University of Cincinnati

Date: 2/28/2020

I, Sourabh Shende, hereby submit this original work as part of the requirements for the degree of Master of Science in Mechanical Engineering.

It is entitled:

Bayesian Topology Optimization for Efficient Design of Origami Folding Structures

Student's name: **Sourabh Shende**

This work and its defense approved by:

Committee chair: Kumar Vemaganti, Ph.D.

Committee member: Philip Buskohl, Ph.D.

Committee member: Manish Kumar, Ph.D.

36575

Bayesian Topology Optimization for Efficient Design of Origami Folding Structures

A thesis submitted to the

Graduate School

of the University of Cincinnati

in partial fulfillment of the

requirements of degree of

Master of Science

in the Department of Mechanical and

Materials Engineering

2020

by

Sourabh Shende

Bachelor of Technology (B.Tech.)

Visvesvaraya National Institute of Technology, India, 2015

Committee Chair: Dr. Kumar Vemaganti

Abstract

Bayesian optimization (BO) is a popular method for solving optimization problems involving expensive objective functions. Although BO has been applied across various fields, its use in structural optimization area is in its early stages. Origami folding structures provide a complex design space where the use of an efficient optimizer is critical. In this research work for the first time the ability of BO to solve origami-inspired design problems is demonstrated. A Gaussian process (GP) is used as the surrogate model that is trained to mimic the response of the expensive finite element (FE) objective function. The ability of this BO-FE framework to find optimal designs is verified by applying it to two well known origami design problems: chomper and twist chomper.

The performance of the proposed approach is compared to traditional gradient-based optimization techniques and genetic algorithm methods in terms of ability to discover designs, computational efficiency and robustness. BO has many user-defined components and parameters, and intuitions for these for structural optimization are currently limited. In this work, the role of hyperparameter tuning and the sensitivity of Bayesian optimization to the quality and size of the initial training set is studied. Taking a holistic view of the computational expense, various heuristic approaches are proposed to reduce the overall cost of optimization.

A methodology to include derivative information of the objective function in the formulation of the GP surrogate is described, and its advantages and disadvantages are discussed. Additionally, an anisotropic GP surrogate model with independent length scales for each design variable is studied. A procedure to reduce the overall dimension of the problem using information from anisotropic models is proposed.

The results show that Bayesian optimization is an efficient and robust alternative to traditional methods. It allows for the discovery of optimal designs using fewer finite element

solutions, which makes it an attractive choice for the non-convex design space of origami fold mechanics.

Acknowledgments

First and foremost, I would like to thank my research advisor and committee chair, Dr. Kumar Vemaganti for all the guidance and support he has provided to me on this research project. I am grateful to him for finding ample time to discuss research ideas. Without his expertise, persistent help and motivation this thesis would not have been possible.

I would also like to thank Dr. Philip Buskohl and Dr. Manish Kumar for being part of my thesis defense committee.

My Master's thesis project work was done in collaboration with the Air Force Research Laboratory (AFRL), Dayton, Ohio. I especially would like to thank Dr. Philip Buskohl, Dr. Andrew Gillman and Dr. David Yoo who were actively involved in the supervision of my research work. The quality of this project has been greatly enhanced by all the feedback and valuable suggestions provided by them.

I would like to pay my special regards to my parents Mr. Charudatta Shende and Mrs. Rohini Shende and my sister Mrs. Anagha Gurav for all the loving support and encouragement provided to me throughout my graduate studies.

I wish to acknowledge the opportunity provided by the University of Cincinnati - Simulation Center to work on the R&D projects from the Procter & Gamble company. I would like to thank them for giving financial assistance during most of the time here at University of Cincinnati.

Finally, I would like to thank my friends Ashutosh Roy, Sayali Kedari, Krishna Kenja, Mitra Lanka, Teng Long, Kedar Naik, Roshan Vincent, Khushang Mehta and Pushkar Wadagbalkar for sharing wonderful time with me at the University of Cincinnati.

Contents

List of Figures

Chapter 1

Introduction

Global optimization of nonlinear functions is a pervasive need and challenge across many applications and research areas. In many problems of interest, the objective function $f(\boldsymbol{x})$ is not only non-convex in nature but also expensive to evaluate. The scope of this thesis is to study the stochastic Bayesian optimization technique and its application to discovering optimal designs of origami folding structures.

1.1 Bayesian Optimization

Bayesian optimization (BO) techniques are particularly suitable for optimizing expensive problems. Bayesian optimization uses a computationally cheap surrogate model to construct a posterior distribution over the original expensive objective function $f(\boldsymbol{x})$ using a limited number of observations. The statistics of the surrogate model are used to sample a new point where the original objective function $f(\boldsymbol{x})$ is then evaluated. The incorporation of such a surrogate model makes Bayesian optimization an efficient global optimization technique in terms of the number of function evaluations, as shown by several researchers including Mockus [5], Jones et al. [6], Streltsov and Vakili [7], and Sasena [8]. An excellent tutorial on BO is presented by Brochu et al. [4]. Although Bayesian optimization has been used in various fields, its application to structural optimization remains largely unexplored. In this work, the use of Bayesian optimization for origami mechanics problems is studied. This is motivated by the rich design space afforded by such problems and the potential efficiencies to be gained from Bayesian optimization. The main focus of this research is to

adopt a holistic view of the computational costs in BO and provide insights into the factors that affect the efficiency of the approach.

1.1.1 Surrogate Models

Surrogate models used in Bayesian optimization can be broadly classified into two types: parametric and non-parametric regression models. Parametric surrogate models such as nth-degree polynomials usually make strong assumptions about the behavior of the objective function. This is very restrictive in the real world as the complexity of the problem is usually unknown. Non-parametric surrogate models do not make such assumptions and the surrogate adapts itself with newly observed data. Various surrogates are reviewed and compared by Bhosekar and Ierapetritou [9].

Gaussian processes (GPs) are the most popular non-parametric model [4] because of their flexibility and utility. Gaussian process regression is also known as Kriging and these terms are often used interchangeably in the literature. The Gaussian process regression model utilizes a covariance or kernel function to dictate the structure of the response surface. The covariance function is controlled by a finite number of parameters, commonly known as hyperparameters. These are generally unknown and are tuned in order for the surrogate model to match the expensive objective function.

Random forests [10] are another well known non-parametric model. This is an ensemble method where different learning methods are trained on particular features of the data to ignore inactive features and the outcomes from each of the individual learners are then combined. The thorough review by Criminisi et al. [11] indicates that a random forest regression model can outperform a Gaussian process regression model in some cases with bi-modal data. This is because of the inherently unimodal nature of the Gaussian distribution. Nonetheless, the results of this research demonstrate the compatibility of GPs with the origami design space.

1.1.2 Hyperparameter Estimation

Typically GP hyperparameters are estimated by maximizing the marginal likelihood as detailed by Rasmussen and Williams [12]. The likelihood function is the probability of the observed data given the hyperparameters. The maximum of the likelihood function thus indicates the most probable hyperparameters for the observed data. The information added

by the estimated hyperparameters can be measured using Fisher information (see Efron and Hinkley [13], Abt and Welch [14], Vemaganti et al. [15]), which can be seen as the curvature of the logarithm of the likelihood function. However, hyperparameters estimated by maximizing the log-likelihood function often have the drawback of overfitting the observed data [12]. Moreover the unrestricted domain of the hyperparameters often leads to numerical challenges. In order to counter this, Wang and de Freitas [16] derived a regret bound on the hyperparameters in a stochastic setting while estimating it using the marginal likelihood.

1.1.3 Acquisition Function

The GP surrogate model is stochastic in nature and must be scalarized in order to optimize it. The resulting scalar function is known as the acquisition function, since its optimization leads to the acquisition of the next point at which the expensive objective function is evaluated. Several acquisition functions have been used in the literature, including probability of improvement [17], expected improvement [18], and lower or upper confidence bound [19]. The acquisition functions may themselves be tuned using additional parameters, so as to balance exploration and exploitation of the design space.

1.1.4 Efficient Variants

To further improve the performance of the BO algorithm, derivative information about the objective function can be included to provide essential directions in the design space to locate the global solution. Such surrogates with derivative information are studied by Wu et al. [20], Eriksson et al. [21] and Wu et al. [22].

Another way to boost the performance of BO is to use an anisotropic variant of the surrogate, where a unique length scale hyperparameter is associated with each design variable. This makes the surrogate more flexible and can help in faster convergence. Another benefit of using anisotropic models is to determine the relevancy of each design variable, which is commonly known as automatic relevance determination (ARD) [12]. For high-dimensional optimization problems, the relevance information about each design variable can be used to drop any irrelevant variables and reduce the overall dimension of the problem.

This research works explores both these variants and applies it to the optimization problem of the origami folding structures.

1.1.5 Scalability of BO

While Bayesian optimization has been successfully applied to low-dimensional problems, scaling it up to higher dimensions is a major challenge and an open question. This is because the number of observations required by GP grows exponentially as the dimension increases [23], which in turn increases the computational cost and is often infeasible. Additionally, the global optimization of the high dimensional acquisition function becomes a hard problem. In order to tackle this, Rana et al. [24] used an elastic GP with an adaptive length scale parameter to overcome the flat region of the acquisition function in high dimensions and essentially solved a lower dimension optimization of the acquisition function. Wang et al. [25] introduced high-dimensional Bayesian optimization via random embedding (REMBO), where a higher dimensional search space was projected onto a lower dimension using a random transformation matrix. Kandasamy et al. [26] tackled the high-dimensional problem for objective functions with an additive structure wherein the original function is decomposed as the sum of multiple functions dependent on lower dimensional components. Li et al. [27] borrowed the dropout idea from deep neural networks to solve high-dimensional problems where only a subset of variables are optimized at each iteration of BO. These algorithms are reviewed and compared for various synthetic high-dimensional test problems by Choffin and Ueda [28].

This work does not focus on tackling the scale up issue but instead conducts a holistic cost analysis and provides guidelines for the efficient usage of the overall computational budget to solve the optimization problem.

1.1.6 Applications of BO

A number of applications of BO are summarized by Shahriari et al. [3]. BO is extensively used by the machine learning community (Snoek et al. [29], Swersky et al. [30], Bergstra et al. [31]) and is a fairly recent addition to the structural optimization field. Im and Park [32] used particle swarm optimization with a surrogate model for structural optimization problems. Fan et al. [33] adopted Kriging (Gaussian process) surrogates for reliability based

design optimization of crane bridges. Guirguis et al. [34] used a Kriging-interpolated level set approach to reduce the number of design variables in the multi-objective topology optimization of multi-component structures. Zhang et al. [35, 36] implemented a Kriging-assisted multiscale topology optimization methodology for inhomogeneous cellular structures. Dominguez et al. [37] applied BO to solve shape optimization problems in a non-linear finite element framework. Liu et al. [38] also used BO in a finite element framework to design for structural crashworthiness. Another study using Kriging surrogate models for topology optimization of crash structures was conducted by Raponi et al. [39]. Our work seeks to further this line of research, with an emphasis on taking a holistic view of the overall computational cost of BO as applied to origami design problems.

1.2 Origami Folding Structures

Origami, an ancient Japanese paper-folding art form that transforms a plain sheet of paper to complex three-dimensional fold patterns, provides an appropriate test case for topology optimization given the highly nonlinear motions and critical connection to fold pattern topologies. Beyond a rich design problem, origami structures have proven utility in a variety of applications including packaging, optics, biomedical devices, deployable structures, energy absorption (see Turner et al. [40] and Peraza-Hernandez et al. [41] for complete review). There have been numerous structural models used for modeling origami motions from purely rigid body motion (see Tachi [42, 43]) to higher fidelity shell element models for modeling facet deformation (see Song et al. [44]). Recent works by Liu and Paulino [45] and Gillman et al. [1] introduce models that capture the large rotations of origami structures. These models are ideal for efficient topology optimization, and Gillman et al. [2], Fuchi et al. [46, 47] have demonstrated the complex and non-convex nature of these design problems. These works use evolutionary optimization algorithms to find optimal design solutions. The downside of using evolutionary methods like the genetic algorithm (GA) is that it requires many function evaluations to find the optimal fold pattern. This makes finding an efficient optimization algorithm critical.

1.3 Specific Aims

The main goal of this study is to formulate an efficient optimization framework to solve the origami folding structure problem put forward by Gillman et al. [1]. In order to meet this goal, the following specific aims are pursued in this research work:

- Implement Bayesian optimization framework to solve origami folding structures problem.
- Compare the efficiency of BO to previously used gradient-based optimization and GA techniques.
- Test the compatibility of Gaussian process (GP) surrogate model to fit the origami optimization data.
- Study the role of the choice of GP covariance functions on BO.
- Evaluate the sensitivity of the BO algorithm to the initial training set for robustness.
- Quantify the value of estimating hyperparameters at various intervals or holding them fixed.
- Utilize objective function derivative information in the proposed optimization framework and study its potential benefits in solving the origami problem.
- Study the advantages and disadvantages of using anisotropic covariance functions to solve the origami problem, and apply automatic relevance determination to reduce the problem dimension.

1.4 Outline of Thesis

The rest of the thesis document is divided into following chapters.

- Chapter 2 describes the modified nonlinear truss formulation used to model the origami mechanics in this work. Chapter 2 also describes the optimization framework and algorithms implemented in previous work.

- Chapter 3 presents the details of Bayesian optimization, including the covariance and acquisition functions. It also provides the mathematical formulation to include derivative information in the GP and discusses an anisotropic version of the GP surrogate.

- Optimization results from BO for two origami actuation problems, chomper and twist chomper, are presented in Chapter 4 and compared with previously used algorithms. Chapter 4 also presents some heuristic approaches for improving the performance and efficiency of the BO algorithm. It also shows the benefits of using derivative information and using the anisotropic variant of the GP in finding solutions to the chomper and twist chomper problems.

- Chapter 5 summarizes the thesis with concluding remarks. Some future directions of research are also presented in this chapter.

Chapter 2

Problem Formulation and Optimization Framework

2.1 Nonlinear Finite Element Formulation

The structural model of origami folding structures used in this work is a modified nonlinear truss model as presented by Gillman et al. [1]. This model accounts for geometric nonlinearities (small strain/large rotation) in the deformation of of the structure. The standard truss element formulation is modified by including a torsional spring around it, which helps in modeling the fold stiffness between adjacent facets of the origami structure. Additionally, the positional finite element method (FEM) formulation (Greco et al. [48]) is used for numerical modeling. The positional formulation simplifies the representation of fold angles and also helps in assigning the constraints. A summary of the formulation from Gillman et al. [1] is provided here for completeness.

2.1.1 Principle of Minimum Potential Energy

The total energy, Π, of a single truss element in terms of the internal energy, U_t, and external energy, P, is:

$$\Pi = U_t - P, \tag{2.1}$$

where the internal energy is defined as

$$U_t = l_0 \int_0^1 \frac{EA}{2} \varepsilon \left(\boldsymbol{X}_1, \boldsymbol{X}_2\right)^2 + \frac{G}{2} \tilde{\phi} \left(\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_3, \boldsymbol{X}_4\right)^2 \, \mathrm{d}\zeta, \tag{2.2}$$

$$= l_0 \int_0^1 u_t + u_h \, \mathrm{d}\zeta. \tag{2.3}$$

Here, E and A are the Young's modulus and cross-sectional area of the truss, G is the

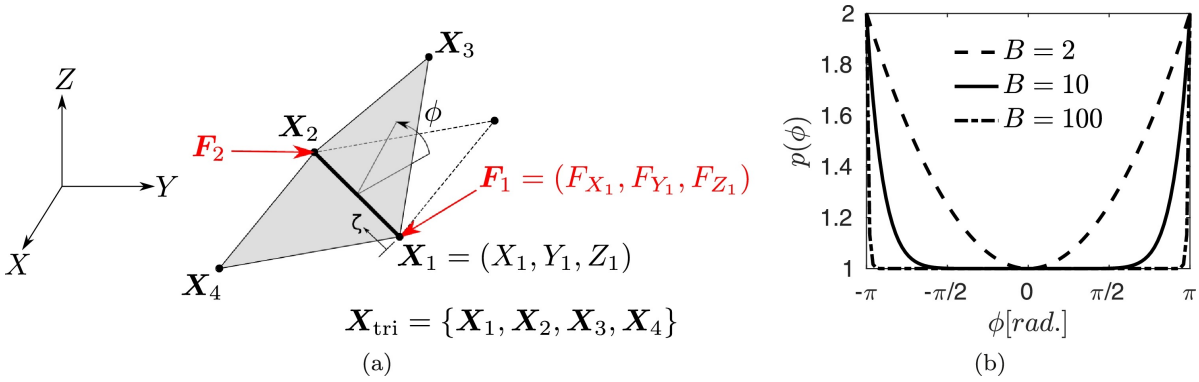


Figure 2.1: (a) Schematics of the base truss element, (b) Penalty function, $p(\phi) = C((\phi/\pi)^B) + 1$ for different exponent values, B $(C = 1)$. This figure is borrowed from Gillman et al. [1].

torsional spring constant (per unit length), and ζ is the non-dimensional integration length along the axial direction of the truss. The local node numbers $\boldsymbol{X}_1$, $\boldsymbol{X}_2$, $\boldsymbol{X}_3$, and $\boldsymbol{X}_4$, as shown in Fig. 2.1(a), define the extensional strain, $\varepsilon(\boldsymbol{X}_1, \boldsymbol{X}_2)$, and the nonlinear spring rotation, $\tilde{\phi}(\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_3, \boldsymbol{X}_4)$:

$$\varepsilon(\boldsymbol{X}_1, \boldsymbol{X}_2) = (|\boldsymbol{X}_2 - \boldsymbol{X}_1| - l_0)/l_0, \tag{2.4}$$

$$\tilde{\phi}(\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_3, \boldsymbol{X}_4) = \phi(\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_3, \boldsymbol{X}_4) - \phi_0. \tag{2.5}$$

Here, l_0 is the original (stress-free) length of the element, ϕ_0 is the original angle, and $\phi(\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_3, \boldsymbol{X}_4)$ is the current fold angle. Note that both the axial strain, ε, and the rotation, $\tilde{\phi}$, are nonlinear functions with respect to nodal coordinates. Additionally, a penalty function $p(\phi) = C((\phi/\pi)^B) + 1$ is introduced to enforce local contact and avoid singularity. Here B and C are constants that affect the sharpness of the fold stiffness near the closed state and the nominal amount of stiffness respectively. Figure 2.1(b) shows variation of the penalty function with different values of B keeping C constant $(C = 1)$. With the inclusion of

penalty function the total energy function takes following form:

$$U_t^p = l_0 \int_0^1 u_t + p(\phi) u_h \, \mathrm{d}\zeta. \tag{2.6}$$

The external energy, P is defined as

$$\begin{aligned} P =& F_{X_1}\left(X_1 - X_1^0\right) + F_{X_2}\left(X_2 - X_2^0\right) + F_{Y_1}\left(Y_1 - Y_1^0\right) \\ &+ F_{Y_2}\left(Y_2 - Y_2^0\right) + F_{Z_1}\left(Z_1 - Z_1^0\right) + F_{Z_2}\left(Z_2 - Z_2^0\right), \end{aligned} \tag{2.7}$$

where the external forces are denoted by $\{F_{X_k}, F_{Y_k}, F_{Z_k}\}$. The equilibrium state of the structure is obtained by applying the principle of minimum potential energy, leading to the nonlinear system of equations:

$$\begin{aligned} \frac{\partial \Pi}{\partial X_i} &= l_0 \int_0^1 \frac{\partial u_t}{\partial X_i} + \frac{\partial (p(\phi) u_h)}{\partial X_i} \, \mathrm{d}\zeta - F_{X_i}, \\ &= l_0 \int_0^1 \frac{\partial u_t}{\partial X_i} + \left(G p(\phi) \tilde{\phi} + G \frac{\tilde{\phi}^2}{2} \frac{\partial p(\phi)}{\partial \phi} \right) \frac{\partial \phi}{\partial X_i} \, \mathrm{d}\zeta - F_{X_i}, \\ &= 0. \end{aligned} \tag{2.8}$$

2.1.2 Linearization of Nonlinear Equations

The nonlinear system of equations (2.8) is solved by linearization through a Taylor series expansion:

$$\begin{aligned} R_i = \frac{\partial \Pi}{\partial X_i} = R_i\left(X_{tri}, F_i\right) = f_i\left(X_{tri}\right) - F_i = 0, \\ R_i \approx R_i\left(X_{tri}^0\right) + \nabla R_i\left(X_{tri}^0\right) \Delta X_{tri} = 0, \end{aligned} \tag{2.9}$$

where $X_{tri} = \{X_1, Y_1, Z_1, X_2, Y_2, Z_2, X_3, Y_3, Z_3, X_4, Y_4, Z_4\}$ is the set of global coordinates of the nodes used to define the fold angle ϕ. The tangent term,

$K_{ik} = \nabla R_i \left(X^0_{tri}\right) = f_{i,k}\left(X^0_{tri}\right) - F_{i,k}$, $F_{i,k} = 0$, is defined as

$$\begin{aligned} K_{ik} = f_{i,k} = l_0 \int_0^1 & \frac{\partial^2 u_t}{\partial X_k X_i} + Gp\left(\phi\right)\left(\frac{\partial \phi}{\partial X_k}\frac{\partial \phi}{\partial X_i} + \tilde{\phi}\frac{\partial^2}{\partial X_k \partial X_i}\right) \\ & + G\frac{\partial^2 p(\phi)}{\partial \phi^2}\frac{\tilde{\phi}^2}{2}\left(\frac{\partial \phi}{\partial X_k}\frac{\partial \phi}{\partial X_i} + \tilde{\phi}\frac{\partial^2}{\partial X_k \partial X_i}\right) \\ & + 2G\tilde{\phi}\frac{\partial p(\phi)}{\partial \phi}\left(\frac{\partial \phi}{\partial X_k}\frac{\partial \phi}{\partial X_i}\right)\, \mathrm{d}\zeta. \end{aligned} \tag{2.10}$$

These linearized set of equations (2.9) are solved in an iterative manner using an arc-length type method.

2.1.3 Constraints

A Lagrange multiplier approach is used to implement periodic boundary conditions to constrain the strains of the mirrored truss elements on the opposite boundaries of the unit cell. Traditional nodal constraints (Dirichlet boundary conditions) are also accommodated in the same Lagrange multiplier approach. These linear constraints for the nonlinear mechanics model are formulated as

$$\begin{aligned} f_i\left(X_{tri}\right) - F_i + A^T_{ij}\lambda_j &= 0, \\ A^T_{ij}X_j - \overline{X}_i &= 0, \end{aligned} \tag{2.11}$$

where a matrix of constants, A_{ij}, represents the coefficients in the constraint equations, and $\overline{X}_i$ are prescribed locations of select nodes at a given loading step. To enforce the periodic displacements, constraints are formulated as follows:

$$\mathbf{X}_2 - \mathbf{X}_1 = \mathbf{X}'_1 - \mathbf{X}'_2, \tag{2.12}$$

where $\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}'_1, \mathbf{X}'_2$ are nodes on the matching boundaries as shown in Fig. 2.2. Also in order to define the fold angles at the boundary truss elements, "ghost nodes" are introduced as shown in Fig. 2.2, where $\mathbf{X}_4 = \mathbf{X}_2 + \left(\mathbf{X}'_3 - \mathbf{X}'_1\right)$. The linearized set of equations including the periodicity constraint are

$$\begin{bmatrix} {}^n_k\mathbf{K} & \mathbf{A}^T \\ \mathbf{A} & \mathbf{0} \end{bmatrix} \begin{Bmatrix} {}^n_{k+1}\Delta\mathbf{X} \\ {}^n_{k+1}\Delta\boldsymbol{\lambda} \end{Bmatrix} = \begin{Bmatrix} {}^n_k\mathbf{f} - {}^n\mathbf{F} - \mathbf{A}^T \cdot {}^n_k\boldsymbol{\lambda} \\ {}^n\overline{\mathbf{X}} - \mathbf{A}^T \cdot {}^n_k\boldsymbol{\lambda}, \end{Bmatrix}, \tag{2.13}$$

where n and k are the loading step and nonlinear iteration step, respectively, and $\mathbf{K}$ refers to the tangent term as defined in Eq. (2.10).

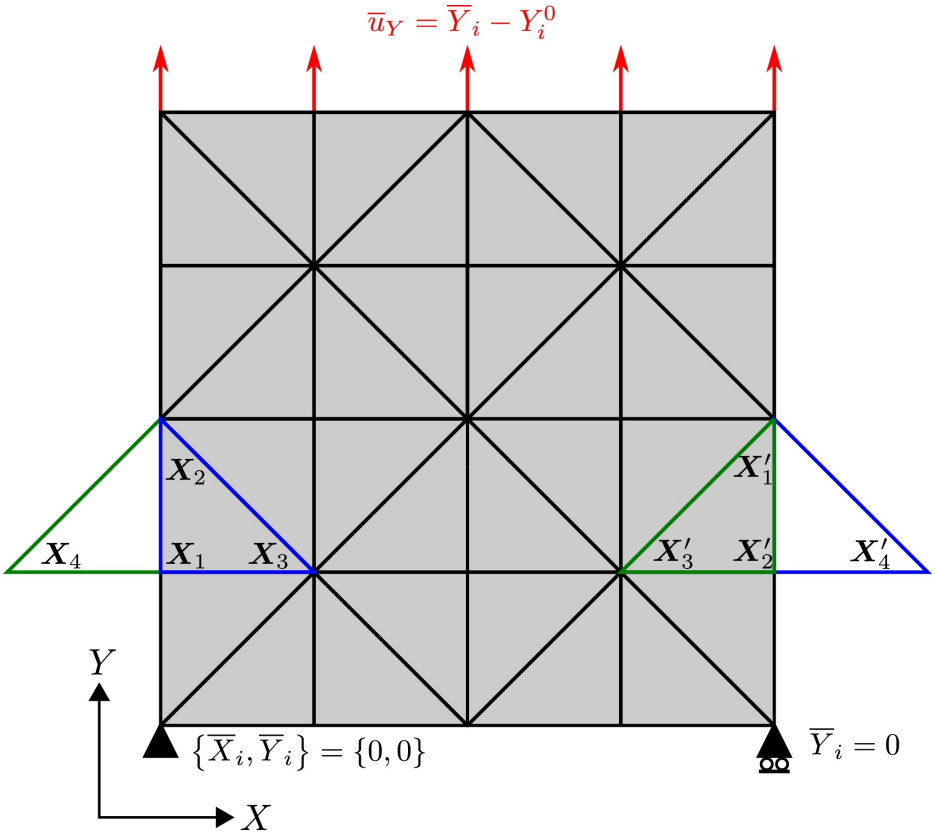


Figure 2.2: Schematics for the periodic and Dirichlet boundary conditions using Lagrange multiplier approach. Ghost nodes ($\mathbf{X}_4, \mathbf{X}_4^{'}$) are introduced to define the fold angles of the corresponding boundary truss element. This figure is borrowed from Gillman et al. [1].

2.1.4 Generalized Displacement Control Method

Origami structures have complex nonlinear load-displacement profiles with critical "snap-back" points. In order to accurately capture such critical points, a robust arc-length with generalized displacement control method (GDCM) is implemented. For the arc-length approach, a scalar Lagrange multiplier, λ_F, is used to scale the applied force vector $\mathbf{F}$. With this additional Lagrange multiplier the system of equations takes following form:

$$\begin{aligned} f_i\left(X_{tri}\right) - \lambda_F F_i + A_{ij}^T \lambda_j &= 0, \\ A_{ij}^T X_j - \overline{X}_i &= 0. \end{aligned} \tag{2.14}$$

2.1.5 Selection of Bifurcating Branch from Flat State

The flat state of the origami problem is another critical point with many bifurcating branches towards multiple equilibrium states. The correct choice of the path is selected using modal analysis. A perturbation force field is determined from the modal analysis of the flat state of the origami structure. For the first nonlinear step $(k = 1)$ of the first load step $(n = 1)$, an augmented stiffness matrix is defined as :

$$ {}^{n}_{k}\mathbf{K}_{aug} = \begin{bmatrix} {}^{n}_{k}\mathbf{K}_{aug} & \mathbf{A}^{T} \\ \mathbf{A} & 0 \end{bmatrix}. \tag{2.15} $$

Eigenvalue decomposition of the augmented stiffness matrix gives out all eigenvalues ${}_{m}\theta_{v}$ and eigenvectors ${}_{m}\mathbf{v}$:

$$ \left({}^{1}_{1}\mathbf{K}_{aug} - {}_{m}\theta_{v}\mathbf{I}\right) {}_{m}\mathbf{v} = \mathbf{0}. \tag{2.16} $$

The eigenvector corresponding to the lowest eigenvalue represents the lowest energy deformation mode of the flat state and so is used to compute the perturbation force field, $\mathbf{F}_{perturb}$. The perturbation force field is applied only on the first load step $(n = 1)$ and is set to zero for all other load steps.

2.2 Optimization Framework

Utilizing this efficient truss-based formulation, an iterative optimization problem can be formulated. This work uses the same design optimization problem as specified in Gillman et al. [2] to find the fold pattern that results in the largest actuation by varying the fold stiffness, where the design variables are the fold stiffness values of the torsional springs. The origami fold pattern is computed by turning each fold line on or off, making the design problem binary. But for gradient-based algorithms, the fold stiffness is considered as a continuous and differentiable function: $G_s\left(\alpha_k\right) = 10^{\overline{\alpha_1}+\alpha_k(\overline{\alpha_2}-\overline{\alpha_1})}$. The minimum fold stiffness value is $G_{soft} = 10^{\overline{\alpha_1}}$ and the maximum stiffness value is $G_{stiff} = 10^{\overline{\alpha_2}}$, where the design variable α_k is constrained to lie in the range $[0, 1]$. A small stiffness value is assigned to soft active fold lines while inactive fold lines are given large stiffness values. The output actuation nodes and their selected degrees of freedoms are represented by vector $\boldsymbol{c}$. The optimization problem is further constrained by the number of allowable active fold lines. This is

represented by ν_0 which denotes the fraction of inactive fold lines. Mathematically, this optimization problem can be represented as:

$$\begin{aligned}
&\text{Find} && \boldsymbol{\alpha} = \alpha_k, k = 1, 2, \ldots, N_f \text{ that} \\
&\text{Minimizes} && J = -\boldsymbol{c}^T \boldsymbol{u} \\
&\text{Subject to} && g = \nu_0 - \tfrac{1}{N_f} \sum_{k=1}^{N_f} \alpha_k \leq 0, \\
& && 0 \leq \alpha_k \leq 1 \, \forall \, k \\
& && R_i \left(\boldsymbol{X} \right) = 0 \\
& && u_i = X_i - X_i^0, \; i = 1, 2, \ldots, 3N_n,
\end{aligned} \tag{2.17}$$

where N_n denotes total number of nodes in the truss structure. Different gradient and evolutionary algorithms are implemented in [2] to solve the origami topology optimization problem. The flowchart for the framework implemented in [2] is shown in Fig. 2.3. The different optimization algorithms utilized are discussed briefly in the subsequent subsections.

2.2.1 Gradient Based Algorithms

As presented in Gillman et al. [2], the optimization problem (2.17) was solved using various well established gradient-based algorithms like the method of moving asymptotes (MMA) [49], sequential quadratic programming (SQP) and interior-point (IP) methods. The MMA algorithm to solve origami folding problem is implemented using Svanberg's [49] MATLAB code. The sequential quadratic programing (SQP) and interior point (IP) methods use MATLAB's optimization toolbox [50] to solve the problem. A brief overview of these algorithms is given in the following subsections.

2.2.1.1 Interior Point Method

In the interior point method, the inequality constraint minimization problem is converted to an equality constraint minimization problem. This is done by using slack variables and a barrier

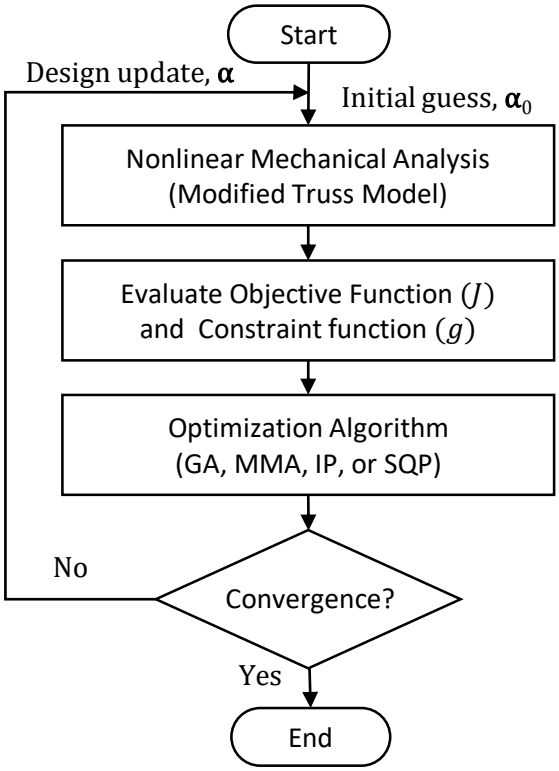


Figure 2.3: Flowchart of topology optimization framework. This figure is borrowed from Gillman et al. [2].

function. Typically the original problem is framed as follows:

$$\begin{aligned} &\text{Minimize} && f(\mathbf{x}), \\ &\text{Subject to} && h(\mathbf{x}) = 0 \text{ and } g(\mathbf{x}) \leq 0, \end{aligned} \tag{2.18}$$

and is converted to an approximate problem as shown below:

$$\begin{aligned} &\text{Minimize} && f_\mu(\mathbf{x}, \mathbf{s}) = f(\mathbf{x}) - \mu \textstyle\sum_i \ln(s_i), \\ &\text{Subject to} && h(x) = 0 \text{ and } g(\mathbf{x}) + \mathbf{s} = 0, \end{aligned} \tag{2.19}$$

The number of slack variables, s_i, is equal to the number of inequality constraints g. In order to keep $\ln(s_i)$ bounded, s_i are restricted to be positive. The added logarithmic term is called a barrier function. The inclusion of the barrier function keeps the search point $\mathbf{x}$ in the interior region of the constraint space, hence the name interior point method. As the μ value decreases to zero, the minimum of f_μ approaches the minimum of f in the constraint region.

The approximate problem defined in Eq. (2.19) is solved using Newton method or conjugate gradient iterations. More details of the algorithm may be found in [51–53].

2.2.1.2 Sequential Quadratic Programing

Sequential quadratic programing (SQP) forms one of the most effective gradient techniques to solve nonlinear constrained optimization problems. The principal idea in SQP is to formulate quadratic programming (QP) subproblem using quadratic approximation of the Lagrangian function. The general minimization problem is defined as:

$$\begin{aligned} &\text{Minimize} && f(\mathbf{x}), \\ &\text{Subject to} && g_i = 0 \quad i = 1, ..., m_e, \\ & && g_i \leq 0 \quad i = m_e + 1, ..., m. \end{aligned} \tag{2.20}$$

The objective function and constraints can be combined using the Lagrangian function:

$$\mathbf{L}(\mathbf{x}, \lambda) = f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i \cdot g_i(\mathbf{x}). \tag{2.21}$$

The QP subproblem is then formulated as:

$$\begin{aligned} &\min \tfrac{1}{2}\mathbf{d}^T \mathbf{H}_k \mathbf{d} + \nabla f(\mathbf{x}_k)^T \mathbf{d}, \\ &\nabla g_i(\mathbf{x}_k)^T \mathbf{d} + g_i(\mathbf{x}_k) = 0 \quad i = 1, ..., m_e, \\ &\nabla g_i(\mathbf{x}_k)^T \mathbf{d} + g_i(\mathbf{x}_k) \leq 0 \quad i = m_e + 1, ..., m. \end{aligned} \tag{2.22}$$

The matrix $\mathbf{H}_k$ is the positive definite approximation of Hessian matrix of the Lagrangian function (2.21). The solution of this QP subproblem produces a vector $\mathbf{d}_k$, which is used as a search direction for new point so that:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k d_k, \tag{2.23}$$

where α_k is the step length parameter. For an overview of the SQP algorithm see [54–57].

In these gradient approaches the design variable, $\boldsymbol{\alpha}$, varies continuously in $[0, 1]$. The disadvantage of using gradient-based methods is that they are sensitive to the initial point and often converge to a local minimum for the non-convex origami optimization problem. Therefore, a non-gradient based evolutionary genetic algorithm (GA) was studied in Gillman et al. [2], using MATLAB's in-built genetic algorithm toolbox [58]. The general GA approach is briefly discussed in the next subsection.

2.2.2 Genetic Algorithm

Genetic algorithms are a family of heuristic global optimization methods inspired by Darwin's theory of biological evolution. The algorithm begins by randomly generating a population for the first generation. Each member of the current population is given a fitness score based on its objective function value. The population of subsequent generation is selected based on different strategies: retention, crossover and mutation. The members of the populations are commonly called chromosomes. For the retention strategy, some chromosomes with better fitness values are carried to the next generation. Such chromosomes are sometimes called the elite population. For the crossover strategy, some randomly selected members referred to as parents are used to produce new members or children. These children are generated by combining genes from both parents. Different variants of selecting and combining genes from parents can be followed. Typically for the mutation strategy, new members are generated by altering the genes of randomly selected members of current population. The fitness value of the newly generated population is computed and the whole process of generating subsequent generations is repeated. Typically the algorithm is stopped when the number of generations reaches a particular limit. Genetic algorithms are reviewed and discussed in detail by Whitley [59], Houck et al. [60] and Tanese et al. [61].

For the origami problem, the GA approach gives superior solutions compared to gradient-based techniques [2]. However, the GA approach requires an order of magnitude more function evaluations to achieve convergence, making it computationally expensive. In order to address this drawback, this work implements a probabilistic approach known as Bayesian optimization (BO).

Chapter 3

Bayesian Optimization

Bayesian optimization (BO) is an optimization technique that uses a stochastic surrogate model to emulate an expensive objective function based on a finite number of function observations. It is a global method used mostly to optimize functions that are expensive to evaluate or are non-convex in nature. It is called Bayesian because it is based on *Bayes theorem*, which states that the posterior probability of a model M given evidence E is directly proportional to the product of the likelihood of E given M and the prior probability of M as explained in Sivia and Skilling [62] :

$$P\left(M|E\right) \propto P\left(E|M\right) P\left(M\right). \tag{3.1}$$

The prior $P\left(M\right)$ represents the space of the possible objective functions in Bayesian optimization. Considering the objective function to be f, the prior takes the form $P(f)$. As more observations are accumulated $\mathbb{D} \in (\mathbf{x}^{1:n}, \mathbf{f}^{1:n})$, the prior is combined with the likelihood function $P(\mathbb{D}|f)$ resulting in the posterior distribution $P(f|\mathbb{D})$,

$$P\left(f|\mathbb{D}\right) \propto P\left(\mathbb{D}|f\right) P\left(f\right). \tag{3.2}$$

The posterior captures the updated belief about the unknown objective function. This step can also be considered as mimicking the true objective function response using a surrogate model. At each iteration, the expensive objective function is evaluated at the optimum resulting from the surrogate model and this new information is used to retrain the surrogate. The unique methodology of using a non-deterministic surrogate model makes Bayesian optimization (BO)

an efficient global optimizer capable of both design space exploration and exploitation.

As summarized by Shahriari et al. [3] and Brochu et al. [4], BO has two main parts. The first part is the probabilistic surrogate model that replicates the behavior of the expensive objective function. The second part is the scalarization of the probabilistic model using an acquisition function to predict the next point of evaluation. Figure 3.1 illustrates the key ideas in BO. The dashed line in Fig. 3.1(a) indicates the expensive objective function f that we seek to minimize, along with two initial training points where the function is evaluated. Next, the top panel of Fig. 3.1(b) shows the first iteration of the surrogate model whose mean passes through the two initial training points. The shaded area indicates the uncertainty in the surrogate model in areas away from the initial training points. In the lower panel, we see the acquisition function, with a triangular marker indicating the location of its minimum. The expensive function is now evaluated at this point and the surrogate model is updated in Fig. 3.1(c). For this second iteration, there is less uncertainty overall in the surrogate model. The acquisition function is again optimized to obtain the next training point and the process is repeated. Figure 3.1(d) shows the eighth iteration with the surrogate model closely mimicking the original expensive function.

3.1 Gaussian Processes

In this work, we employ the widely used Gaussian process (GP) as the surrogate model. GPs are attractive because mathematically they are easy to model and they provide information about the expected value and uncertainty of the objective function. GPs have been the central component of Bayesian methods as used by Osborne et al. [63], Snoek et al. [29], Wang et al. [25] and Lizotte et al. [64]. Different variants of GP models are discussed in Bhosekar and Ierapetritou [9].

A Gaussian process can be seen as an extension of the Gaussian distribution to the functional space. A GP is a distribution over functions completely defined by its mean function m and covariance function (or kernel) k (Shawe-Taylor and Cristianini [65]):

$$f(\mathbf{x}) \sim \mathcal{N}\left(m(\mathbf{x}), k\left(\mathbf{x}, \mathbf{x}^{'}\right)\right), \tag{3.3}$$

where $\mathcal{N}$ denotes normal distribution. For simplicity we assume the prior mean function

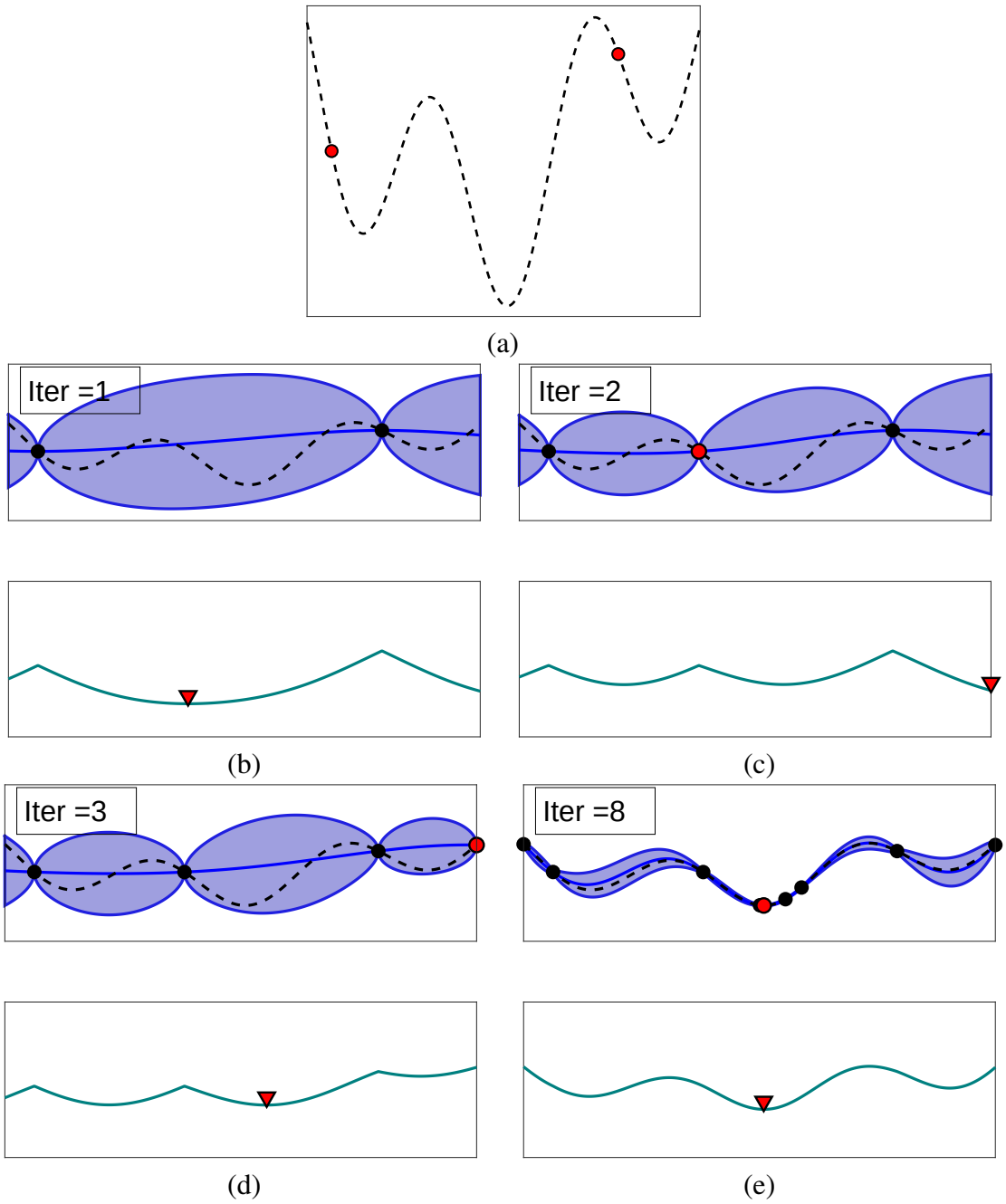


Figure 3.1: Schematic of Bayesian optimization in one dimension: (a) Expensive objective function and two initial training points, shown with red markers. (b) The GP surrogate model (top) is trained with the two points and is stochastic, with the blue bands indicating the uncertainty levels. The acquisition function (bottom) scalarizes the surrogate model and is minimized to find the next training point shown with the red triangle. (c) The expensive function is evaluated at the new training point (red circle) and added to the training data. The process is repeated in (d) and (e), with the surrogate model mimicking the expensive function better with each iteration. The uncertainty in the surrogate model is also reduced as more training points are added. The illustration is inspired by the example presented in Shahriari et al. [3].

to be zero: $m(\mathbf{x}) = 0$. This assumption is not restrictive because as more training points are observed the prior is updated and becomes more informative. The following covariance functions are considered in this work (Shahriari et al. [3]):

$$k_{squared-exponential}\left(\mathbf{x}, \mathbf{x}^{'}\right) = \theta_0^2 \exp\left(-\frac{1}{2}r^2\right), \tag{3.4}$$

$$k_{\text{Matérn1}}\left(\mathbf{x}, \mathbf{x}^{'}\right) = \theta_0^2 \exp\left(-r\right), \tag{3.5}$$

$$\begin{aligned} k_{\text{Matérn3}}\left(\mathbf{x}, \mathbf{x}^{'}\right) &= \theta_0^2 \exp\left(-\sqrt{3}r\right) \\ &\quad \left(1 + \sqrt{3}r\right), \end{aligned} \tag{3.6}$$

$$\begin{aligned} k_{\text{Matérn5}}\left(\mathbf{x}, \mathbf{x}^{'}\right) &= \theta_0^2 \exp\left(-\sqrt{5}r\right) \\ &\quad \left(1 + \sqrt{5}r + \frac{5}{3}r^2\right), \end{aligned} \tag{3.7}$$

where $r^2 = \left(\mathbf{x} - \mathbf{x}^{'}\right)^T \mathbf{\Lambda} \left(\mathbf{x} - \mathbf{x}^{'}\right)$ and $\mathbf{\Lambda}$ is a diagonal matrix of d squared length scales l_i^2, d being the dimension of $\mathbf{x}$. For isotropic covariance functions same value of length scale is considered for all dimensions, so that $l_i = l$. There are many other choices of covariance functions available that may be better suited to a given problem. The covariance function gives the correlation between the input points $\mathbf{x}$ and $\mathbf{x}^{'}$ and is parameterized by the amplitude parameter θ_0 and the length scale parameter l. These parameters are commonly known as hyperparameters. Given n observations of the objective function $f(\mathbf{x})$ at points $\mathbf{x}^i$, the complete covariance/kernel matrix is given by:

$$\mathbf{K} = \begin{bmatrix} k(\mathbf{x}^1, \mathbf{x}^1) & \dots & k(\mathbf{x}^1, \mathbf{x}^n) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}^n, \mathbf{x}^1) & \dots & k(\mathbf{x}^n, \mathbf{x}^n) \end{bmatrix}. \tag{3.8}$$

To predict the value of the function f at a new point $\mathbf{x}_{n+1}$, we enforce the GP property that $\mathbf{f}_{1:n}$ and $f_{n+1} = f(\mathbf{x}_{n+1})$ are jointly Gaussian, so that:

$$\begin{bmatrix} \mathbf{f}^{1:n} \\ f^{n+1} \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \mathbf{K} & \mathbf{k} \\ \mathbf{k}^T & k(\mathbf{x}^{n+1}, \mathbf{x}^{n+1}) \end{bmatrix}\right), \tag{3.9}$$

where,

$$\mathbf{k} = \begin{bmatrix} k(\mathbf{x}^{n+1}, \mathbf{x}^1) & k(\mathbf{x}^{n+1}, \mathbf{x}^2) \dots k(\mathbf{x}^{n+1}, \mathbf{x}^n) \end{bmatrix}^T. \tag{3.10}$$

The predictive distribution is given by Rasmussen and Williams [12]:

$$P\left(f^{n+1}|\mathbb{D}^{1:n}, \mathbf{x}^{n+1}\right) \sim \mathcal{N}\left(\mu_n\left(\mathbf{x}^{n+1}\right), \sigma_n^2\left(\mathbf{x}^{n+1}\right)\right), \tag{3.11}$$

where the mean and variance at $\mathbf{x}^{n+1}$ are

$$\mu_n\left(\mathbf{x}^{n+1}\right) = \mathbf{k}^T\mathbf{K}^{-1}\mathbf{f}_{1:n}, \tag{3.12}$$

$$\sigma_n^2\left(\mathbf{x}^{n+1}\right) = k(\mathbf{x}^{n+1}, \mathbf{x}^{n+1}) - \mathbf{k}^T\mathbf{K}^{-1}\mathbf{k}. \tag{3.13}$$

The mean and the variance at $\mathbf{x}_{n+1}$ predicted by the GP are used to scalarize the surrogate model, which is discussed next in Section 3.2. The GP for a one-dimensional problem is illustrated in Fig. 3.2, which shows the mean (solid blue line) and the uncertainty (shaded blue area) of the GP. The mean of the GP model matches the objective function values at the three observed values shown by black dots.The uncertainty at these three observed location is zero. Apart from these three observed locations, the uncertainty forms a Gaussian distribution about the mean. These distributions are illustrated at locations x^1, x^2 and x^3 in Fig. 3.2.

As mentioned earlier, we consider the use of squared exponential and different Matérn covariance functions in our study. One of the main difference between these covariance function is the smoothness parameter, ν. The squared expoential function has infinite smoothness, $\nu = \infty$, whereas Matérn 1 has the least smoothness parameter value, $\nu = \frac{1}{2}$. The smoothness parameter value for Matérn 3 and Matérn 5 are $\nu = \frac{3}{2}$ and $\nu = \frac{5}{2}$, respectively. The difference in the nature of the posteriors obtained from each of these covariance functions is illustrated in Fig. 3.3. The four panels in the figure show three random samples from the posterior distribution obtained with each of the four covariance functions. As seen in Fig. 3.3(a), samples from the surrogate using the squared exponential function are the smoothest, followed by Matérn 5 (3.3(d)), Matérn 3 (3.3)(c) and lastly by Matérn 1 (3.3)(d). There is no absolute rule to choose the covariance function for a particular objective function. As a rule of thumb, the squared exponential function is preferred if the objective function is infinitely differentiable and other Matérn family functions are chosen depending on the roughness of the objective function.

The GP model is sensitive to the hyperparameters used in the covariance function, which are generally not known. Different values of the hyperparameters result in different behaviors

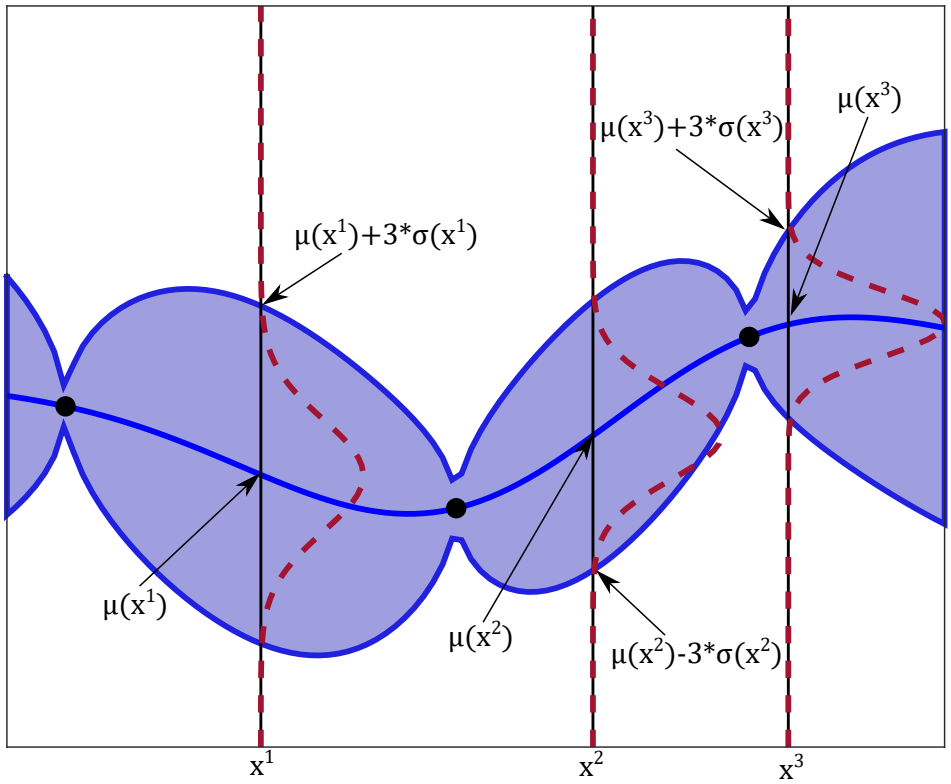


Figure 3.2: One-dimensional Gaussian process with three observations shown with solid black dots. The mean of the surrogate is shown using a solid black line. The uncertainties are denoted by the shaded blue areas. The dotted red curves at $\mathbf{x}^1$, $\mathbf{x}^2$ and $\mathbf{x}^3$ shows the Gaussian distribution superimposed on the mean $\mu(\cdot)$ and standard deviation $\sigma(\cdot)$. The illustration is inspired by the example presented in Brochu et al. [4].

of the surrogate model. For the isotropic Gaussian model, as the length scale value, l, increases the uncertainty seems to decrease as seen in Fig. 3.4. Similarly, from Fig. 3.5 it can be noted that the uncertainty increases as the amplitude parameter, θ_0, increases.

One way to estimate hyperparameters is by maximizing the marginal likelihood (Jones et al. [6], Rasmussen and Williams [12]), which is the probability of the surrogate model, given the observed data $\mathbf{x}^{1:n}$ and the hyperparameters l and θ_0. The hyperparameters estimated by this method help in selecting the most probable model for the given data. The analytical expression for the logarithm of the marginal likelihood is:

$$\mathcal{L} = \log p\left(\mathbf{f}|\mathbf{x}^{1:n}, l, \theta_0\right) = -\frac{1}{2}\mathbf{f}^T\mathbf{K}^{l,\theta_0}\mathbf{f} - \frac{1}{2}\log|\mathbf{K}^{l,\theta_0}| - \frac{n}{2}\log(2\pi)\,. \tag{3.14}$$

Maximizing the log-likelihood function (3.14) requires the solution of a smaller (two-dimensional) optimization problem and may be performed using a local or global

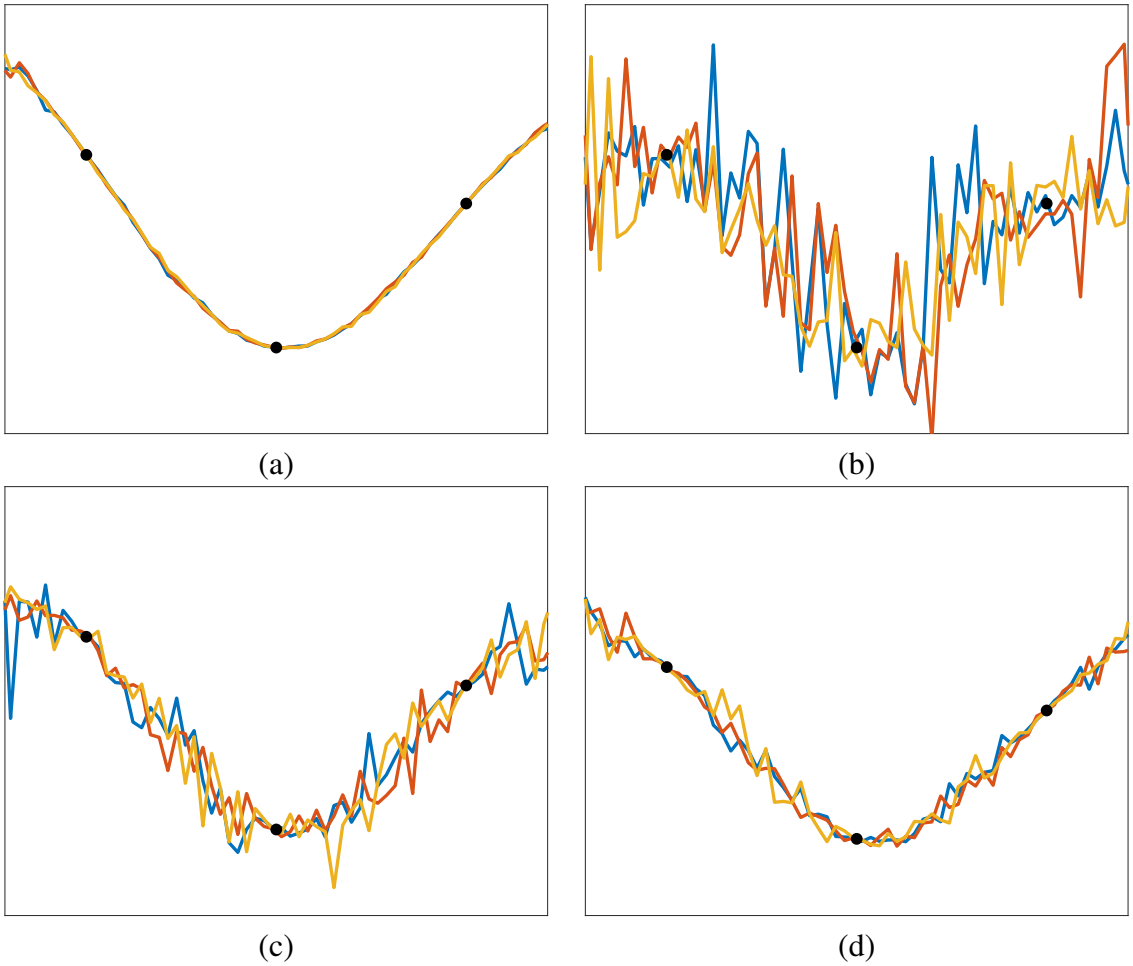


Figure 3.3: Posterior sample functions of the Gaussian process surrogate with various covariance functions: (a) Squared exponential, (b) Matérn 1, (c) Matérn 3, and (d) Matérn 5. The three black dots are the observed points, i.e., data $\mathbb{D}$.

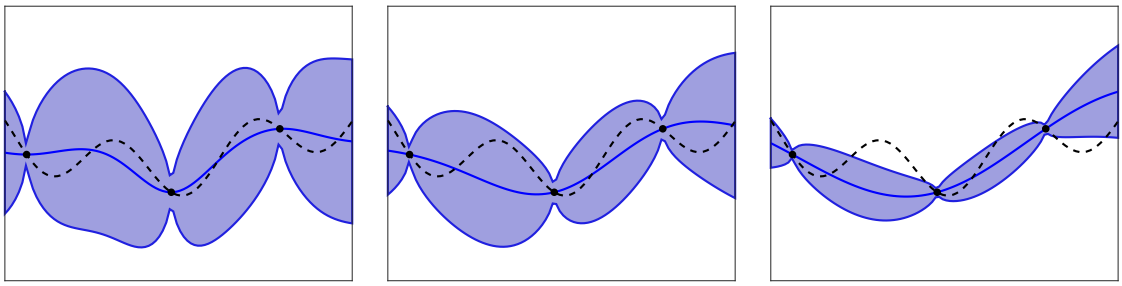
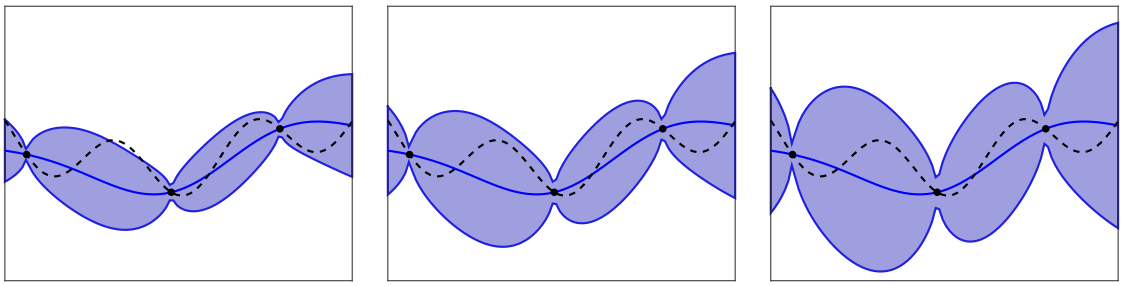


Figure 3.4: Gaussian process surrogate model for different length scale parameter values: (a) $l = 0.5$, (b) $l = 1$, and (c) $l = 2$. The dashed black line is the objective function. The mean of the surrogate is plotted with a solid blue line and the uncertainties are represented by shaded blue areas.

method. This involves computing the inverse of covariance matrix $\mathbf{K}^{l,\theta_0}$. As the size of $\mathbf{K}^{l,\theta_0}$ increases, so does the computational cost involved in hyperparameter estimation. This cost



(a) (b) (c)

Figure 3.5: Gaussian process surrogate model for different amplitude parameter values: (a) $\theta_0 = 0.5$, (b) $\theta_0 = 1$, and (c) $\theta_0 = 2$. The dashed black line is the objective function. The mean of the surrogate is plotted with a solid blue line and the uncertainties are represented by shaded blue areas.

can be reduced if the hyperparameters are estimated less frequently, say, every p iterations.

3.2 Scalarization

The probabilistic model is then scalarized using an acquisition function $\alpha(\mu, \sigma)$, which seeks to balance exploration and exploitation in the selection of the next design point. Probability of improvement (PI) (Kushner [17]) and Expected Improvement (EI) (Lizotte [18]) are two well-known improvement-based acquisition functions, which tend to favor points that are likely to improve the solution based on a target value τ. These acquisition functions have following analytical expressions:

$$\alpha_{PI} = \Phi\left(\frac{\tau - \mu_n\left(\mathbf{x}^{n+1}\right) + \zeta}{\sigma_n\left(\mathbf{x}^{n+1}\right)}\right), \tag{3.15}$$

$$\begin{aligned}\alpha_{EI} = &\left(\tau - \mu_n\left(\mathbf{x}^{n+1}\right) + \zeta\right)\Phi\left(\frac{\tau - \mu_n\left(\mathbf{x}^{n+1}\right) + \zeta}{\sigma_n\left(\mathbf{x}^{n+1}\right)}\right) \\ &+ \sigma_n\left(\mathbf{x}^{n+1}\right)\phi\left(\frac{\tau - \mu_n\left(\mathbf{x}^{n+1}\right) + \zeta}{\sigma_n\left(\mathbf{x}^{n+1}\right)}\right),\end{aligned} \tag{3.16}$$

where Φ is the standard normal cumulative distribution function, ϕ is the standard probability density function, τ is a user-selected target value, and ζ is a parameter used for exploration. Often, it is difficult to set a target value if not enough information is known about the objective function. In such a case, it is convenient to use the lower confidence bound (LCB) acquisition

function (Cox and John [19]):

$$\alpha_{LCB} = \mu_n\left(\mathbf{x}^{n+1}\right) - \kappa\sigma_n\left(\mathbf{x}^{n+1}\right), \tag{3.17}$$

where κ is a user-defined parameter that determines the trade-off between exploration and exploitation.

3.3 Design Space Update and Stopping Criteria

In order to find the next point of evaluation, the acquisition function needs to be optimized. This means one needs to maximize the probability of improvement (PI) and expected improvement (EI), whereas one needs to minimize the lower confidence bound (LCB).

$$\mathbf{x}^{n+1} = \text{argmax}\ \alpha_{PI}\left(\mu_n\left(\mathbf{x}^{n+1}\right), \sigma_n\left(\mathbf{x}^{n+1}\right)\right), \tag{3.18}$$

$$\mathbf{x}^{n+1} = \text{argmax}\ \alpha_{EI}\left(\mu_n\left(\mathbf{x}^{n+1}\right), \sigma_n\left(\mathbf{x}^{n+1}\right)\right), \tag{3.19}$$

or,

$$\mathbf{x}^{n+1} = \text{argmin}\ \alpha_{LCB}\left(\mu_n\left(\mathbf{x}^{n+1}\right), \sigma_n\left(\mathbf{x}^{n+1}\right)\right). \tag{3.20}$$

Figure 3.6 compares the evolution of Bayesian optimization for a one-dimensional problem using these three acquisition functions. As mentioned before, setting a target value τ for probability of improvement (PI) and expected improvement (EI) acquisition functions is difficult without prior knowledge of the objective function. So, often the lowest observed value is considered as the target τ. When this minimum does not change for many iterations, the overall algorithm using PI or EI may be trapped in a local solution. In order to avoid this, the value of the exploration parameter ζ needs to be appropriately chosen. In contrast, for the lower confidence bound (LCB) one just needs to set the exploitation-exploration parameter κ. For a given problem, there is no general rule to select the best acquisition function. Different acquisition functions may be appropriate for different problems.

After finding $\mathbf{x}^{n+1}$ by optimizing the acquisition function, the expensive objective

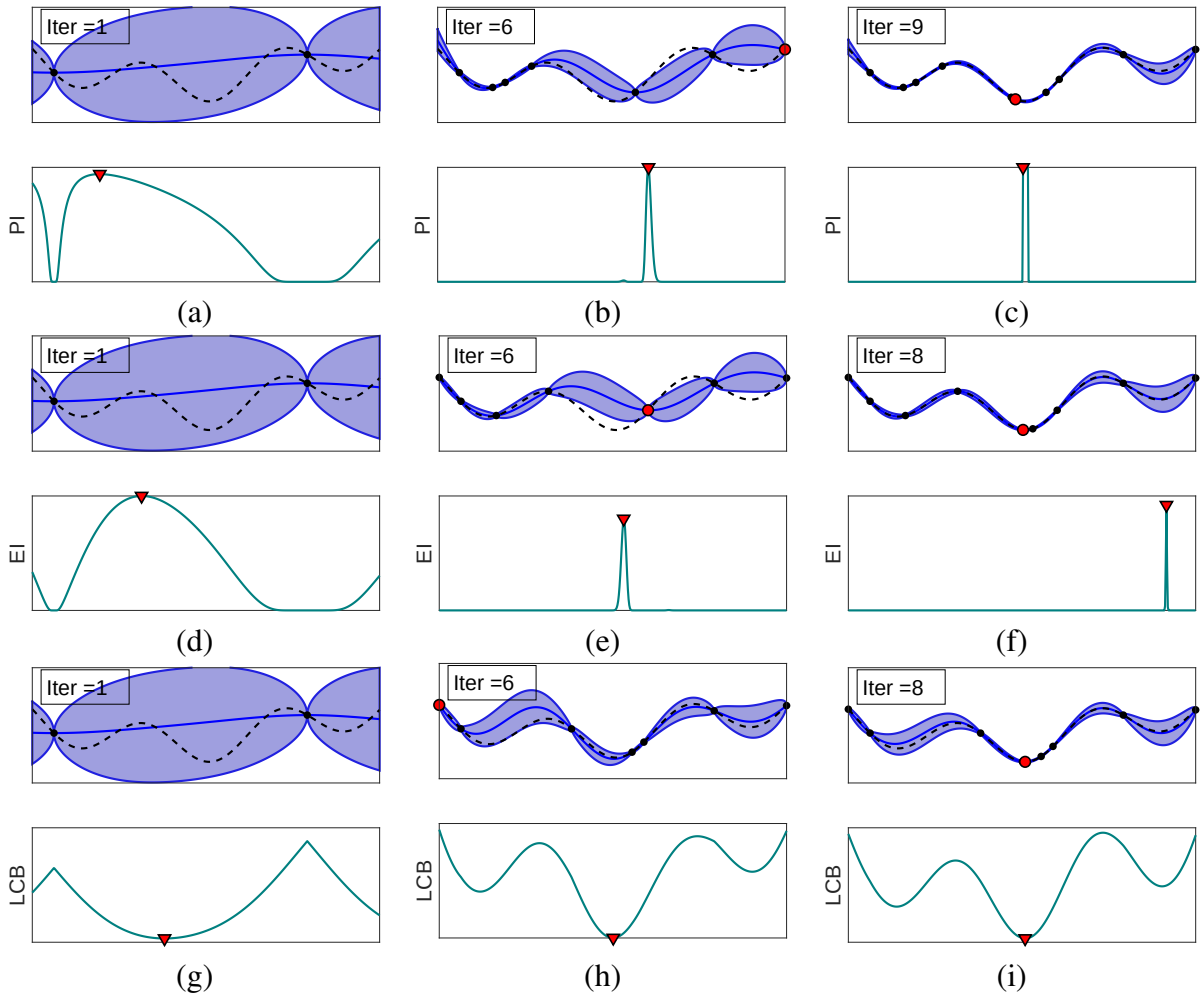


Figure 3.6: Evolution of Bayesian optimization for a one-dimensional problem using various acquisition functions. In the top subplot, the objective function is represented by the dashed black line with black dots as the observed points. The mean of the surrogate is plotted with a solid blue line and the uncertainties are represented by shaded blue areas. The red circle represents newly added observations. In the bottom subplot, the acquisition function is plotted, with a red triangular mark denoting the location of next point to be evaluated. The iterations (a)-(c) use probability of improvement (PI) as the acquisition function, iterations (d)-(f) use expected improvement (EI) as the acquisition function, and iterations (g)-(i) use lower confidence bound (LCB) as the acquisition function.

function is evaluated at this point: $f_{n+1} = f(\mathbf{x}^{n+1})$. This new observation is then appended to the original observations. This process is repeated until the stopping criteria are met. The BO evolution can be stopped when the absolute error is less than some tolerance value. As calculating the error is not always feasible, the maximum number of iterations (number of training points) can be considered as a stopping criterion. The flowchart for the overall Bayesian optimization algorithm is shown in Fig. 3.7.

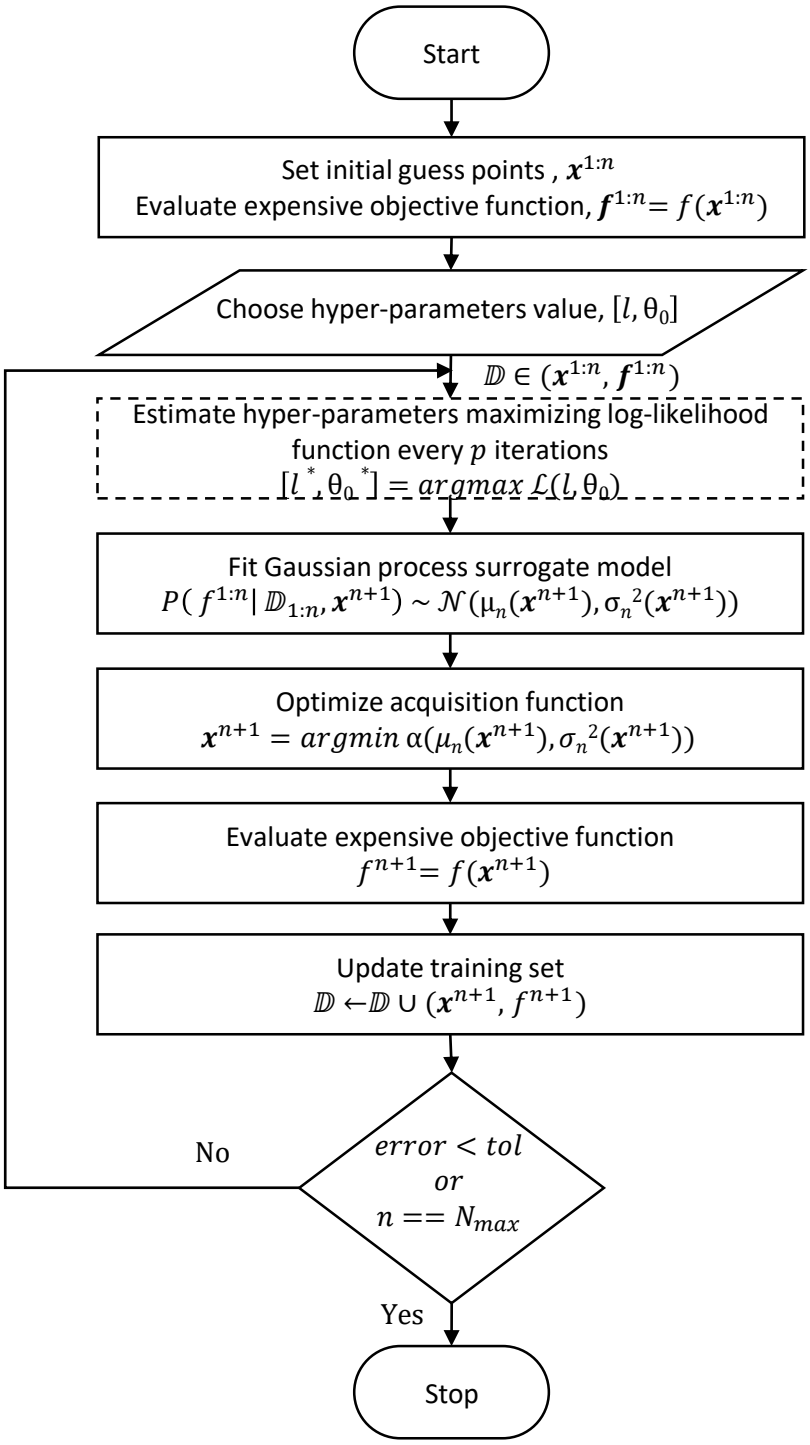


Figure 3.7: Flowchart for Bayesian optimization. The tuning of hyperparameters is optional and shown with a dashed line box. The flowchart highlights how BO improves the efficiency by optimizing the acquisition function using the surrogate model instead of the expensive objective function. The BO workflow provides several options to expend the computational budget of the design problem and requires a holistic analysis to determine best usage for a given problem.

3.4 Parameters for Bayesian Optimization

The expensive objective function in our work is the non-linear FE solver for the origami deformation. The initial training points are chosen randomly. The initial values of the hyperparameters are set either based on prior knowledge or by maximizing the log-likelihood function, $\mathcal{L}$ (3.14). Given the non-convex nature of the log-likelihood function, the maximization problem may become trapped in a local optimum. This is not a major concern since the training data evolve continuously with every iteration and so do the resulting hyperparameters as mentioned in Rasmussen and Williams [12]. In this work we implement the gradient-based interior-point (IP) method (Byrd et al. [51]) for maximizing the log-likelihood function. As mentioned earlier we use four different covariance functions, viz. squared exponential, Matérn1 (commonly known as exponential), Matérn3 and Matérn5. The lower confidence bound acquisition function is considered for the studies done in this research work. The value of the LCB exploitation and exploration trade-off parameter κ is taken as 3 because 3σ for GP covers 99.7% of uncertainty. The LCB function itself can be non-convex multi-modal in nature. So we optimize the acquisition function using a derivative-free genetic algorithm (GA). For this, a population of 200 chromosomes is considered. The elite population size retained for the next generation is taken to be 5% of the population size. Intermediate crossover strategy is used to produce 80% of the population other than the elite population for the next generation. In this strategy children are produced by taking a weighted average of the parents. Remaining chromosomes are generated by mutation where genes of random parents are altered. The GA stops when a maximum of 100 generations is reached. Another stopping criterion for GA is when the change in the best fitness value is less than a function tolerance of 10^{-6} for 50 generations (maximum stall generations).

During the optimization of the acquisition function, the inverse of the kernel matrix needs to be computed. This matrix becomes ill-conditioned if the same point appears repeatedly. To avoid this, if an existing point in the training set re-appears as the solution to the acquisition function minimization, we select a random point as the new training point.

Another source of ill-conditioning is a large value of the hyperparameter l. If this parameter becomes too large, especially during the hyperparameter optimization step, then the rows of the covariance matrix become close to linearly dependent. We overcome this

difficulty by constraining the length scale parameter to be less than twice the maximum Euclidean distance possible in the design space: $l \leq 2 \times \sqrt{d}$. This constraint is justified as the length scale parameter l scales the relative distance between two training points and a maximum value of $2\sqrt{d}$ is physically reasonable.

The objective function and the terms in the covariance matrix are of the order of 10^{-2} for the problems considered in this work. To avoid ill-conditioning, especially as the number of training points increases, we scale the objective function by a factor of 100 in our implementation.

As a stopping criterion of the overall algorithm, we limit the number of FE solutions. The Bayesian optimization procedure is implemented in MATLAB [66], and the built-in MATLAB Genetic Algorithm Toolbox [58] is used to maximize the acquisition function.

3.5 Gaussian Processes with Derivative Enrichment

The approach discussed up to this point does not involve the use of derivative information in exploring the design space for the global optimum. In many optimization algorithms, gradient information is widely used in the search for the optimum as summarized by Snyman [67]. The derivatives of the objective function provide essential directions of the response surface at each location in the design space. In d-dimensional problems, the partial derivative with respect to each d dimension along with the function lead to a set of $d + 1$ values per training point. This additional information is advantageous in closely fitting the surrogate model to the high-dimensional objective function. In the methodology discussed previously, we placed the GP surrogate model as a prior over the objective function f. However, if the derivative information of the expensive objective function is available with little additional cost, it can be incorporated into the GP surrogate and the additional information can be leveraged in finding the global optimum in Bayesian optimization.

As the gradient is the linear operator, Rasmussen and Williams [12] show that the gradient of a GP is also a GP. For the mathematical formulation we consider a function of d variables x_1, $x_2, \ldots x_d$. The first order partial derivatives of the expensive objective function f are denoted as $\nabla(f(\mathbf{x})) = \left[\frac{\partial f}{\partial x_1} \ \frac{\partial f}{\partial x_2} \ \ldots \ \frac{\partial f}{\partial x_d}\right]^T$. The joint process $[f(\mathbf{x}) \ \nabla(f(\mathbf{x}))]$ has a distribution (see

Wu et al. [22]) with four blocks of covariance functions:

$$\begin{aligned}
k_{[f,f]}(\mathbf{x},\mathbf{x}^{'}) &= \text{cov}(f(\mathbf{x}), f(\mathbf{x}^{'})) &= k(\mathbf{x},\mathbf{x}^{'}) &= \mathbf{K} \\
k_{[f,\nabla f]}(\mathbf{x},\mathbf{x}^{'}) &= \text{cov}(f(\mathbf{x}), \nabla f(\mathbf{x}^{'})) &= \nabla_{\mathbf{x}} k(\mathbf{x},\mathbf{x}^{'}) &= \mathbf{K}_{[\mathbf{f},\nabla\mathbf{f}]} \\
k_{[\nabla f,f]}(\mathbf{x},\mathbf{x}^{'}) &= \text{cov}(\nabla f(\mathbf{x}), f(\mathbf{x}^{'})) &= \nabla_{\mathbf{x}^{'}} k(\mathbf{x},\mathbf{x}^{'}) &= \mathbf{K}_{[\nabla\mathbf{f},\mathbf{f}]} \\
k_{[\nabla f,\nabla f]}(\mathbf{x},\mathbf{x}^{'}) &= \text{cov}(\nabla f(\mathbf{x}), \nabla f(\mathbf{x}^{'})) &= \nabla_{\mathbf{x}}\nabla_{\mathbf{x}^{'}} k(\mathbf{x},\mathbf{x}^{'}) &= \mathbf{K}_{[\nabla\mathbf{f},\nabla\mathbf{f}]},
\end{aligned} \tag{3.21}$$

where $\mathbf{K}$ is the covariance matrix between the function values at the training points, $\mathbf{K}_{[\nabla\mathbf{f},\mathbf{f}]}$ is the covariance matrix between gradient components and function values at the training points, and $\mathbf{K}_{[\nabla\mathbf{f},\nabla\mathbf{f}]}$ covariance matrix between gradient components at the training points.

This can be represented more compactly in matrix form as:

$$\begin{bmatrix} \mathbf{f} \\ \nabla\mathbf{f} \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \mathbf{K} & \mathbf{K}_{[\mathbf{f},\nabla\mathbf{f}]} \\ \mathbf{K}_{[\nabla\mathbf{f},\mathbf{f}]} & \mathbf{K}_{[\nabla\mathbf{f},\nabla\mathbf{f}]} \end{bmatrix}\right), \tag{3.22}$$

where the first block $\mathbf{K}$ is the covariance matrix for the original GP (see Eq. (3.8)). For simplicity let $\mathbf{K}_D$ denote the joint covariance matrix for the observed function values and its gradients.

$$\mathbf{K}_D = \begin{bmatrix} \mathbf{K} & \mathbf{K}_{[\mathbf{f},\nabla\mathbf{f}]} \\ \mathbf{K}_{[\nabla\mathbf{f},\mathbf{f}]} & \mathbf{K}_{[\nabla\mathbf{f},\nabla\mathbf{f}]} \end{bmatrix} \tag{3.23}$$

The individual block matrices in Eq. (3.23) take the following form:

$$\begin{aligned}
\mathbf{K}_{[\nabla\mathbf{f},\mathbf{f}]} &= \frac{\partial \mathbf{K}(\mathbf{x},\mathbf{x}^{'})}{\partial \mathbf{x}} \\
&= \begin{bmatrix}
\frac{\partial k(x^1,x^1)}{\partial x_1^1} & \cdots & \frac{\partial k(x^1,x^n)}{\partial x_1^1} \\
\vdots & \ddots & \vdots \\
\frac{\partial k(x^n,x^1)}{\partial x_1^n} & \cdots & \frac{\partial k(x^n,x^n)}{\partial x_1^n} \\
\vdots & \ddots & \vdots \\
\frac{\partial k(x^1,x^1)}{\partial x_d^1} & \cdots & \frac{\partial k(x^1,x^n)}{\partial x_d^1} \\
\vdots & \ddots & \vdots \\
\frac{\partial k(x^n,x^1)}{\partial x_d^n} & \cdots & \frac{\partial k(x^n,x^n)}{\partial x_d^n}
\end{bmatrix}_{nd\times n},
\end{aligned} \tag{3.24}$$

$$\mathbf{K}_{[\mathbf{f},\nabla\mathbf{f}]} = \frac{\partial \mathbf{K}(\mathbf{x},\mathbf{x}^{'})}{\partial \mathbf{x}'} = \mathbf{K}_{[\nabla\mathbf{f},\mathbf{f}]}{}^{T}, \tag{3.25}$$

where $\frac{\partial k}{\partial x_i^j}$ is the derivative of the covariance function with i indicating the dimension and j indicating the training point. Similarly,

$$\begin{aligned}
\mathbf{K}_{(\nabla\mathbf{f},\nabla\mathbf{f})} &= \frac{\partial^2 \mathbf{K}(\mathbf{x},\mathbf{x}^{'})}{\partial \mathbf{x}\partial \mathbf{x}'} \\
&= \begin{bmatrix}
\frac{\partial^2 k(x^1,x^1)}{\partial x_1^1 \partial x_1^1} \cdots \frac{\partial^2 k(x^1,x^n)}{\partial x_1^1 \partial x_1^n} \cdots \frac{\partial^2 k(x^1,x^1)}{\partial x_1^1 \partial x_d^1} \cdots \frac{\partial^2 k(x^1,x^n)}{\partial x_1^1 \partial x_d^n} \\
\vdots \ddots \vdots \\
\frac{\partial^2 k(x^n,x^1)}{\partial x_1^n \partial x_1^1} \cdots \frac{\partial^2 k(x^n,x^n)}{\partial x_1^n \partial x_1^n} \cdots \frac{\partial^2 k(x^n,x^1)}{\partial x_1^n \partial x_d^1} \cdots \frac{\partial^2 k(x^n,x^n)}{\partial x_1^n \partial x_d^n} \\
\vdots \ddots \vdots \\
\frac{\partial^2 k(x^1,x^1)}{\partial x_d^1 \partial x_1^1} \cdots \frac{\partial^2 k(x^1,x^n)}{\partial x_d^1 \partial x_1^n} \cdots \frac{\partial^2 k(x^1,x^1)}{\partial x_d^1 \partial x_d^1} \cdots \frac{\partial^2 k(x^1,x^n)}{\partial x_d^1 \partial x_d^n} \\
\vdots \ddots \vdots \\
\frac{\partial^2 k(x^n,x^1)}{\partial x_d^n \partial x_1^1} \cdots \frac{\partial^2 k(x^n,x^n)}{\partial x_d^n \partial x_1^n} \cdots \frac{\partial^2 k(x^n,x^1)}{\partial x_d^n \partial x_d^1} \cdots \frac{\partial^2 k(x^n,x^n)}{\partial x_d^n \partial x_d^n}
\end{bmatrix}_{nd\times nd},
\end{aligned} \tag{3.26}$$

so that

$$\mathbf{K}_D = \begin{bmatrix}
[\mathbf{K}]_{n\times n} & [\mathbf{K}_{(f,\nabla f)}]_{n\times nd} \\
[\mathbf{K}_{(\nabla f,f)}]_{nd\times n} & [\mathbf{K}_{(\nabla f,\nabla f)}]_{nd\times nd}
\end{bmatrix}_{n(d+1)\times n(d+1)}. \tag{3.27}$$

Thus, the overall covariance matrix $\mathbf{K}_D$ becomes a square matrix of size $n(d+1)\times n(d+1)$. Now, we can apply the GP property to the joint distribution of n observed function values and its gradients to predict the value of the function f at a new point $\mathbf{x}^{n+1}$ as:

$$\begin{bmatrix} \mathbf{f}_{1:n} \\ \nabla\mathbf{f}_{1:n} \\ f_{n+1} \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \mathbf{K}_D & \overline{\mathbf{k}}_D \\ \overline{\mathbf{k}}_D^T & k(x^{n+1},x^{n+1}) \end{bmatrix}\right), \tag{3.28}$$

where $[\overline{\mathbf{k}}_D]_{n(d+1)\times 1} = [\mathbf{k}^T, \mathbf{k}_{[\mathbf{f}^{\mathbf{n+1}},\nabla\mathbf{f}]}]^T$. Similar to the earlier analysis in the Section 3.1, the predictive posterior on f^{n+1} takes following form:

$$P\left(f^{n+1}|\mathcal{D}^{1:n},\mathbf{x}^{n+1}\right) \sim \mathcal{N}\left(\overline{\mu}_n\left(\mathbf{x}^{n+1}\right), \overline{\sigma}_n^2\left(\mathbf{x}^{n+1}\right)\right), \tag{3.29}$$

where

$$\overline{\mu}\left(x^{n+1}\right) = \overline{\mathbf{k}}_D^T \mathbf{K}_D^{-1}\begin{bmatrix} f^{1:n} & \nabla f^{1:n} \end{bmatrix}^T, \tag{3.30}$$

and

$$\overline{\sigma}^2\left(x^{n+1}\right) = k(x^{n+1}, x^{n+1}) - \overline{\mathbf{k}}_D^T \mathbf{K}_D^{-1} \overline{\mathbf{k}}_D. \tag{3.31}$$

The mean $\overline{\mu}$ and variance $\overline{\sigma}^2$ are then used to optimize the acquisition function to find the next evaluation point in the design space. The benefits of utilizing derivatives in GP and in overall BO are studied by Wu et al. [20], Eriksson et al. [21] and Wu et al. [22].

For the work presented here, the derivative analysis of BO is done using the squared exponential covariance function. This can easily be extended to other covariance functions whose first and second order derivatives can be readily determined.

The isotropic squared exponential covariance function has following mathematical form:

$$k(\mathbf{x}, \mathbf{x}^{'}) = \theta_0^2 \exp\left(-\frac{1}{2}\frac{(\mathbf{x} - \mathbf{x}^{'})^2}{l^2}, \right) \tag{3.32}$$

where l is the length scale hyperparameter and θ_0 is amplitude scaling hyperparameter. The first order partial derivatives of the squared exponential covariance function are given by:

$$\frac{\partial k(\mathbf{x}, \mathbf{x}^{'})}{\partial x_d} = -\frac{k(\mathbf{x}, \mathbf{x}^{'})(x_d - x_d^{'})}{l^2}, \tag{3.33}$$

$$\frac{\partial k(\mathbf{x}, \mathbf{x}^{'})}{\partial x_{d'}^{'}} = \frac{k(\mathbf{x}, \mathbf{x}^{'})(x_{d'} - x_{d'}^{'})}{l^2}, \tag{3.34}$$

where d and $d^{'}$ are the dimensions of $\mathbf{x}$ and $\mathbf{x}^{'}$ in consideration respectively.

The second order partial derivatives for the squared exponential covariance function have following form when the dimensions of $\mathbf{x}$ and $\mathbf{x}^{'}$ are not the same, i.e., $d \neq d^{'}$:

$$\frac{\partial^2 k(\mathbf{x}, \mathbf{x}^{'})}{\partial x_d \partial x_{d'}^{'}} = \frac{k(\mathbf{x}, \mathbf{x}^{'})(x_d - x_d^{'})(x_{d'} - x_{d'}^{'})}{l^4}. \tag{3.35}$$

On the other hand, when taking the second partial derivative with respect to the same dimension ($d = d^{'}$),

$$\frac{\partial^2 k(\mathbf{x}, \mathbf{x}^{'})}{\partial x_d \partial x_d^{'}} = \frac{k(\mathbf{x}, \mathbf{x}^{'})}{l^2}\left[1 - \frac{(x_d - x_d^{'})^2}{l^2}\right]. \tag{3.36}$$

The hyperparameters l and θ_0 are estimated by maximizing the log-likelihood $\mathcal{L}$ (see Eq. (3.14)) similar to the earlier approach without derivatives. The $\overline{\mu}$ and $\overline{\sigma}$ calculated using Eqs. (3.21)–(3.36) are used by the acquisition function to scalarize the probabilistic GP surrogate. Any of the acquisition functions – PI, EI or LCB – can be used for scalarization.

In this work we use the lower confidence bound as the acquisition function.

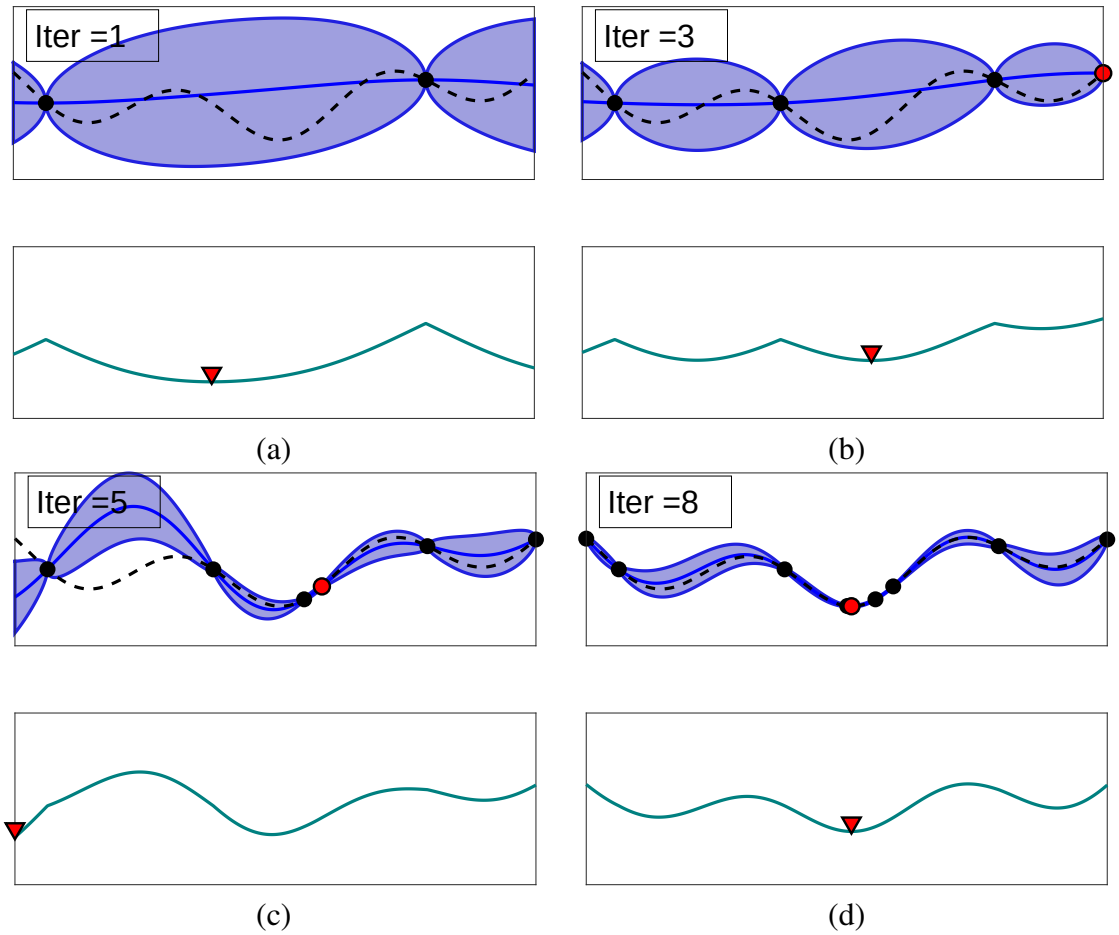


Figure 3.8: Schematic of Bayesian optimization without derivative information for a one-dimensional problem: In panels (a)-(d), the GP surrogate model (top) is trained with the observed points (black and red dots) and is stochastic, with the blue bands indicating the uncertainty levels. The acquisition function (bottom) scalarizes the surrogate model and is minimized to find the next training point shown with the red triangle. As seen in (a)-(d) the GP mean (dark blue line) matches only the true objective function (dotted black line) value at the training points. BO without derivatives eight iterations to find the optimum point.

The fundamental differences between traditional BO and BO with derivative enrichment are summarized in Figs. 3.8 and 3.9 using a one-dimensional problem. The dashed line indicates the expensive one-dimensional objective function. The solid blue lines are the mean of the surrogate models and the shaded blue areas denote the associated uncertainty of the GP surrogate model. As seen in Fig. 3.8(a) to 3.8(d), when derivative information is not included, the mean of the surrogate matches just the true objective function value at the observed points. BO in this case takes eight iterations to find the optimum point (see Fig. 3.8(d)). In contrast when the derivative of the objective function is included, the mean of

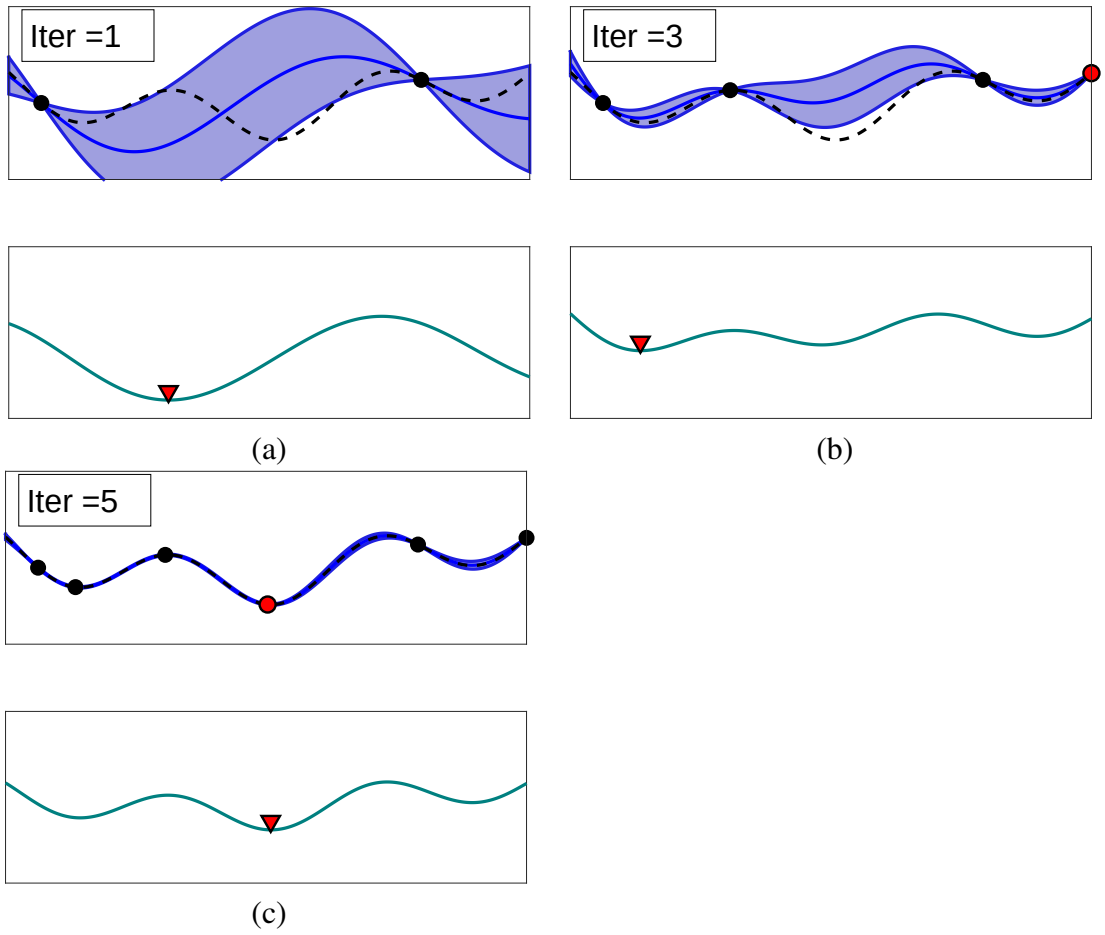


Figure 3.9: Schematic of Bayesian optimization with derivative information for a one-dimensional problem: In the panels (a)-(c), the GP surrogate model (top) is trained with the observed points (black and red dots) and is stochastic, with the blue bands indicating the uncertainty levels. The acquisition function (bottom) scalarizes the surrogate model and is minimized to find the next training point shown with the red triangle. As seen in (a)-(c) the GP mean (dark blue line) matches the true objective function (dotted black line) as well as its derivative at the training points. BO with derivative takes five iterations to find the optimum point.

the surrogate matches both the true objective function value as well as its derivative at the training points, as seen in Fig. 3.9(a) to 3.9(c). Moreover, BO with derivative information is able to find the optimum solution in five iterations (see Fig. 3.9(c)), thus supporting the idea that BO with derivative information can converge faster. Figure 3.9 also indicates that the associated uncertainty is reduced with use of the derivative information.

A more illustrative two-dimensional example showing the usage of derivative information is presented in Figs. 3.10– 3.13. In Figs. 3.10(a) and 3.11(a) the translucent blue surface denotes the expensive two-dimensional objective function with two initial training points marked by black dots. The mean of the surrogate models is represented by the opaque solid

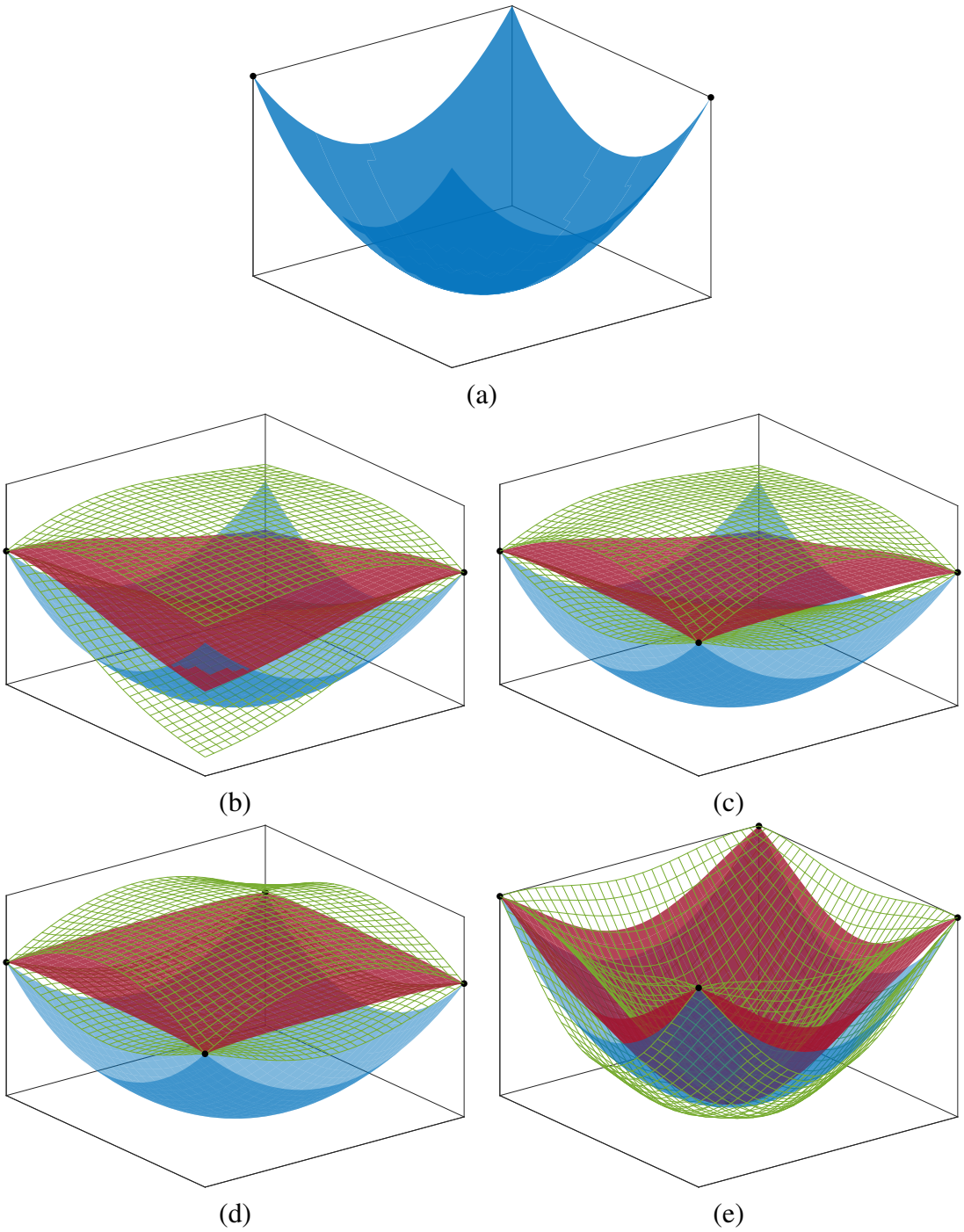


Figure 3.10: Schematic of Bayesian optimization without derivative enrichment for a two-dimensional problem: (a) Surface plot for the true objective function. In panels (b)-(e), the GP surrogate model is trained with the observed points (black dots) and is stochastic, with the solid red surface denoting the mean of the surrogate and the green wireframe surfaces indicating the uncertainty levels. The acquisition function scalarizes the surrogate model and is minimized to find the next training point shown. As seen in (b)-(e) the GP mean (solid red surface) matches only the true objective function (translucent blue surface) value at the training points. BO without derivatives takes four iterations (e) to find the optimum point.

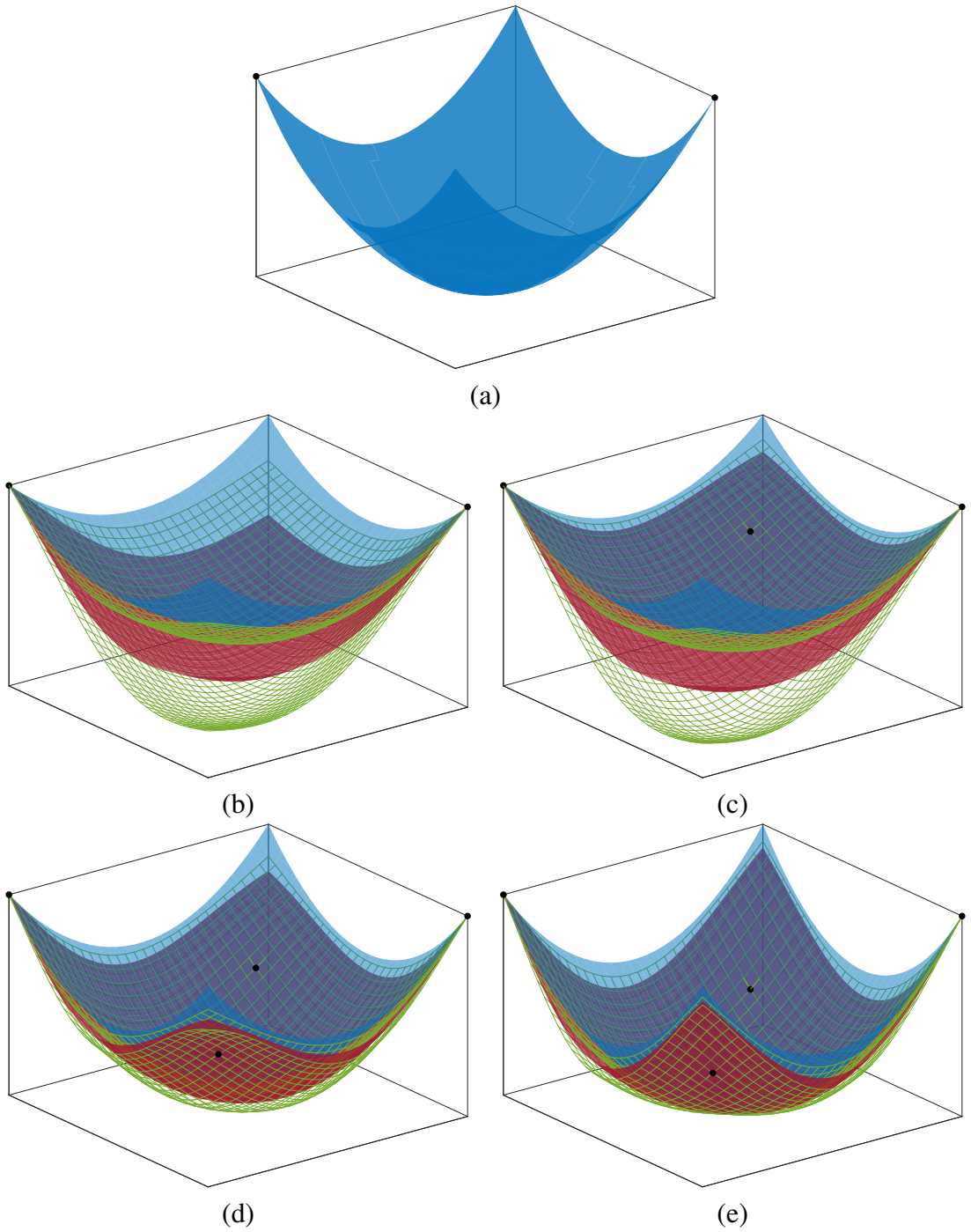


Figure 3.11: Schematic of Bayesian optimization with derivative enrichment for a two-dimensional problem: Surface plot for the true objective function. In panels (b)-(e), the GP surrogate model is trained with the observed points (black dots) and is stochastic, with the solid red surface denoting the mean of the surrogate and the green wireframe surfaces indicating the uncertainty levels. The acquisition function scalarizes the surrogate model and is minimized to find the next training point shown. As seen in (b)-(e) the GP mean (solid red surface) matches the true objective function (translucent blue surface) value as well as its derivative at the training points. BO with derivatives takes four iterations (e) to find the optimum point.

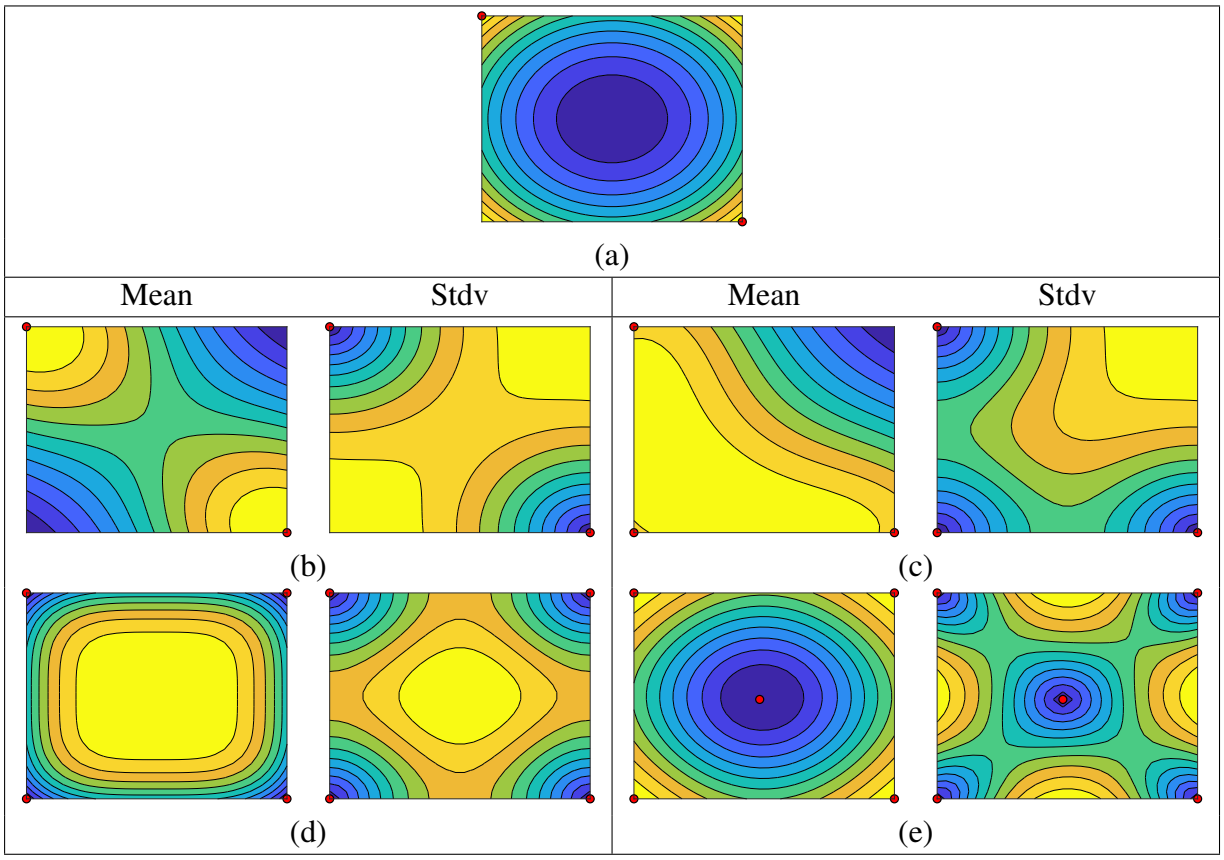


Figure 3.12: Schematic of Bayesian optimization without derivative enrichment for a two-dimensional problem: (a) Contour plot for true objective function. In panels (b)-(e), the GP surrogate model is trained with the observed points (solid red dots) and is stochastic, with the contour plots of the GP mean and standard deviation represented for each iteration. The acquisition function scalarizes the surrogate model and is minimized to find the next training point shown. BO without derivative takes four iterations (e) to find the optimum point.

red surface. The green wireframe surfaces denotes the upper and lower uncertainty bounds of the surrogate models. As seen in Fig. 3.10 the mean matches only the objective function at the observed training points when no derivative information is included in formulating the surrogate model. In contrast, Fig. 3.11 shows that when partial derivative information is included in forming the surrogate model, the mean of the surrogate matches the objective function as well as its derivatives at the the observed locations. Similar remarks apply to Figs. 3.12 and 3.13 where the contours of the functions are plotted. Figures 3.12(a) and 3.13(a) show the contour plots of the true two-dimensional objective function with two initial training points marked with red circles and whose minimum lies at the center. Other panels in Figs. 3.12 and 3.13 show contour plots of the mean and standard deviation of the surrogate models for subsequent iterations. Comparing these figures one can infer that the

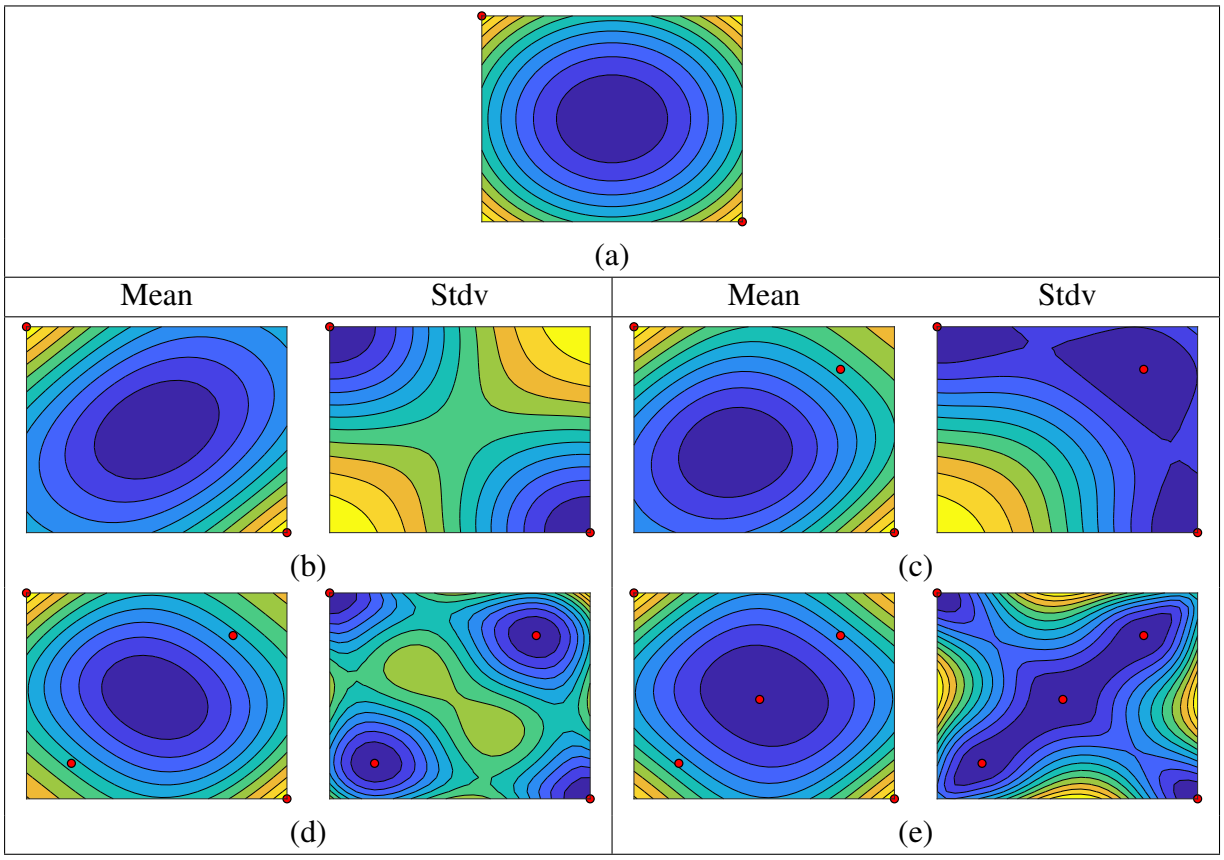


Figure 3.13: Schematic of Bayesian optimization with derivative enrichment for a two-dimensional problem: (a) Contour plot for true objective function. In panels (b)-(e), the GP surrogate model is trained with the observed points (solid red dots) and is stochastic, with the contour plots of the GP mean and standard deviation represented for each iteration. The acquisition function scalarizes the surrogate model and is minimized to find the next training point shown. BO without derivative takes four iterations (e) to find the optimum point.

surrogate shows close resemblance to the objective function surface when derivative information is included.

Computing the mean $\overline{\mu}$ and the variance $\overline{\sigma}^2$ of the GP surrogate model with derivative enrichment involves computing the inverse of the full covariance matrix $\mathbf{K}_D$ (see Eqs. (3.30) and (3.31)). The size of the covariance matrix is $n(d+1) \times n(d+1)$, which not only increases with more training points, but also multiplies as the dimension of the problem d increases. So for high-dimensional problems, the computational cost of fitting a derivative enriched GP surrogate is much higher than a GP surrogate without derivative information. But with the incorporation of derivatives the number of expensive function evaluations required to find the optimum could be less, thus compensating for the extra cost of inverting a bigger covariance matrix K_D.

3.6 Anisotropic Models

The methodology described up to this point uses an isotropic covariance function, which means the same length scale parameter value is assigned to all directions in the parameter space. In other words, the same weight is given to all directions or design variables in an isotropic model. But in reality, for most optimization problems, various design variables affect the objective function differently. For such scenarios, an anisotropic model may be more appropriate. In such models, the anisotropy is introduced by assigning different length scale values to different directions. Considering the squared exponential covariance function defined as:

$$k_{squared-exponential}\left(\mathbf{x}, \mathbf{x}^{'}\right) = \theta_0^2 \exp\left(-\frac{1}{2}r^2\right), \tag{3.37}$$

where $r^2 = \left(\mathbf{x} - \mathbf{x}^{'}\right)^T \mathbf{\Lambda} \left(\mathbf{x} - \mathbf{x}^{'}\right)$, the directional nature of the model is determined by the matrix $\mathbf{\Lambda}$. For isotropic models, $\mathbf{\Lambda}$ takes following form:

$$\mathbf{\Lambda} = l^{-2}\boldsymbol{I}, \tag{3.38}$$

where $\boldsymbol{I}$ is the identity matrix of size d equal to the dimension of the problem. For anisotropic models, $\mathbf{\Lambda}$ is defined as :

$$\mathbf{\Lambda} = diag(\boldsymbol{l})^{-2} = \begin{bmatrix} l_1^{-2} & 0 & \dots & 0 \\ 0 & l_2^{-2} & \dots & 0 \\ \vdots & \dots & \ddots & \vdots \\ 0 & 0 & \dots & l_d^{-2} \end{bmatrix}, \tag{3.39}$$

where $\boldsymbol{l}$ is a vector with different length scale values $l_1, l_2, \dots, l_d$. Anisotropic models are briefly discussed by Rasmussen and Williams [12]. As different length scales are associated with different design variables in anisotropic models, the surrogate model is given more freedom to adapt and fit the observed data, which results in a closer match with the objective response surface.

Figures 3.14 and 3.15 point out the difference between isotropic and anisotropic surrogate model behavior using a two-dimensional objective function. For illustration, the two-dimensional problem has a dominant behavior in one of the coordinate directions.

Figure 3.14(a) shows the objective function surface in translucent blue color with four observed training points shown with black dots. The mean surface of the surrogate is shown in opaque red color and the upper and lower uncertainty bounds are shows using green wireframe in Figs. 3.14(b) and (c). Figure 3.14(b) shows the isotropic surrogate model considering a length scale value $l = 5$ for both directions, whereas Fig. 3.14(c) shows the anisotropic surrogate model with different length scale values $\mathbf{l} = [5, 10]$ for the two directions. Here, a higher length scale value is deliberately assigned to the dormant direction and the dominant direction is assigned a lower length scale value. Because of these different length scale values, the anisotropic surrogate also has an active direction, which helps it in closely resembling the objective response surface. Contour plots of the illustration are shown in Fig. 3.15. The contour plot of objective function is shown in Fig. 3.15(a) with four observed data points marked with red circles, which clearly shows the variation of the objective function in one of the two directions. Figures 3.15(b) and (c) plot the mean and standard deviation of the isotropic and anisotropic surrogate models, respectively. The mean of the anisotropic surrogate model in Fig. 3.15(c) shows most of the variation in the dominant direction. Both Figs. 3.14 and 3.15 suggest that anisotropic surrogates can approximate the objective response surface better than isotropic surrogates if the active direction are aligned with the axes of the input hypercube. However, one needs to assign appropriate length scale values to the different directions for proper behavior of the anisotropic surrogates.

The downside of using an anisotropic model is the need for estimation of more hyperparameters. In the anisotropic case, the number of hyperparameters increases to $d + 1$, d length scale hyperparameters l_i for each dimension, and one amplitude parameter θ_0. Similar to the isotropic case, hyperparameters in anisotropic models can be estimated by maximizing the log-likelihood function $\mathcal{L}$. The log-likelihood maximization problem becomes a $d + 1$-dimensional maximization problem for anisotropic models, which increases the computational cost involved in hyperparameter estimation. But the idea of implementing an anisotropic model is that the overall algorithm might require fewer expensive function evaluations to find the global optimum. In other words, the additional cost involved in estimating different length scale values is balanced by the time saved in performing fewer expensive function evaluations.

Another benefit of using anisotropic models is the ability to determine the relevance of each design variable to the objective function. For the squared exponential covariance

function Eq. (3.37) with anisotropy using $\mathbf{\Lambda}$ from Eq. (3.39), each of the different length scale values $l_1, l_2, \ldots, l_d$ gives the relevance information for the corresponding design variable. In simple terms, each length scale informs how far one needs to move along a particular direction in the input space for the function value to become uncorrelated. This is called automatic relevance determination (ARD) as the inverse of the length scale value denotes how important a design variable is: if the length scale has a very high value, the covariance function becomes independent of that input. ARD was introduced by Neal [68] and is briefly discussed by Rasmussen and Williams [12]. ARD can be used to reduce dimensionality by removing irrelevant input design variables. This eventually leads to less computational time as the number of design variables is reduced.

In this work we study the use of anisotropic models for solving origami design optimization problems. We only use the squared exponential covariance function and estimate the anisotropic hyperparameters by maximizing the log-likelihood function using the interior-point algorithm. We compare results from the isotropic and anisotropic cases for two origami problems. We also use ARD to reduce the dimension of these two problems and study the improvement in the efficiency of the optimization algorithm.

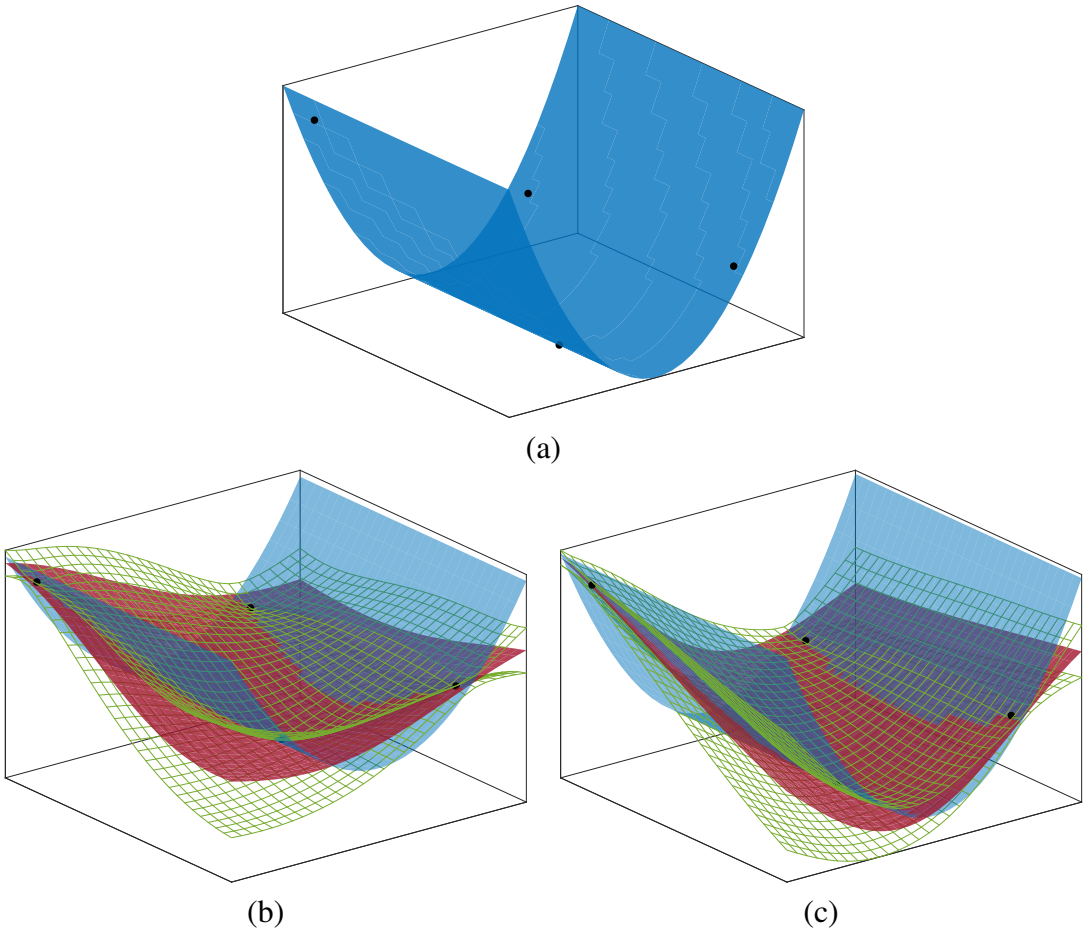


Figure 3.14: Schematic of Bayesian optimization with anisotropic covariance for a two-dimensional problem: (a) Surface plot for the true objective function with dominant direction. (b) The GP surrogate model is trained with the observed points (black dots) with isotropy with length scale $l = 5$, and is stochastic, with the solid red surface denoting the mean of the surrogate and the green wireframe surfaces indicating the uncertainty levels. (c) The GP surrogate model is trained with the observed points (black dots) with anisotropic length scales $\mathbf{l} = [5, 10]$, and is stochastic, with the solid red surface denoting the mean of the surrogate and the green wireframe surfaces indicating the uncertainty levels. The anisotropic model is closer to the original objective function with one dominant direction.

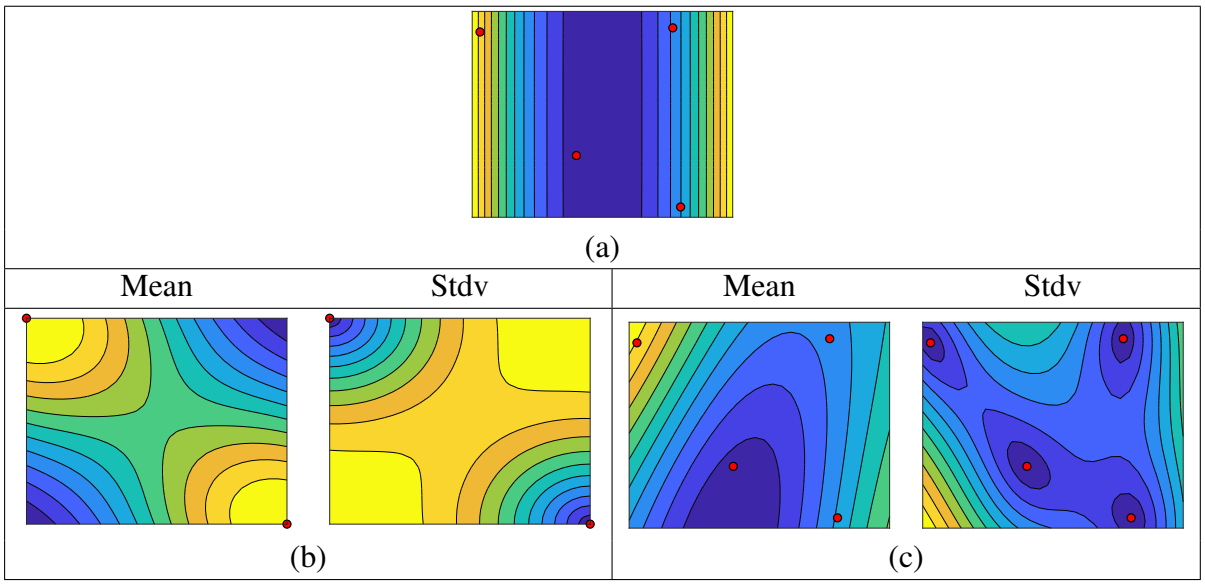


Figure 3.15: Schematic of Bayesian optimization with anisotropic covariance for a two-dimensional problem: (a) Contour plot for the true objective function with a dominant direction. (b) Contour plots of the mean and standard deviation of the GP surrogate model trained with the observed points (solid red dots) with isotropic length scale $l = 5$. (c) Contour plots of the mean and standard deviation of the GP surrogate model trained with the observed points (solid red dots) with anisotropic length scales $\mathbf{l} = [5, 10]$.

Chapter 4

Results

In this chapter, we present results from the application of Bayesian optimization to two origami case studies: the chomper problem and the twist chomper problem. The roles of the covariance function, the initial training set and the hyperparameter estimation are studied in this chapter. Later in the section, a detailed study of the computational cost of BO taking a holistic view is carried out. Various heuristic approaches to reduce the overall computational time are suggested. Improvements in the convergence by utilizing derivative information about the objective function are also shown for the two origami case studies. Results using the anisotropic BO model for these two case studies are presented. Finally, using information from the anisotropic model, a method to eliminate irrelevant design variables and the resulting improvement in computational efficiency using the reduced dimensional form of the objective function are discussed.

Note that BO is not a greedy algorithm and therefore the objective function value does not always improve with every iteration. As such, the evolution of BO is represented using the best objective function value found up to a given iteration. The results from BO are compared with results from the gradient-based sequential quadratic programming (SQP) optimization and GA techniques. The SQP algorithm stops when the function value changes by less than 10^{-14} or when the infinity norm of the gradient is less than 10^{-14} or if the number of iterations exceeds 50. The genetic algorithm considers 560 chromosomes as the population size. For the subsequent generations in GA, 5% of the population is retained as the elite population. New chromosomes are created in each generation by using crossover (80% of population other than elite) and mutation strategies. The algorithm stops when the maximum number of generations

reaches 100 or when the relative change in the function value is less than 10^{-6} for over 50 stall generations.

4.1 Case Studies: Problem Description

The schematic of the two design problems is shown in Fig. 4.1, with the chomper problem on the left and the twist chomper on the right. In both cases, the nodes marked with black triangles are fixed. The blue arrows indicate the locations of the applied forces and we seek to maximize the displacements of the structures at the output nodes shown with green arrows. Each solid line indicates a truss element. The spring stiffness values associated with the internal truss elements constitute the design variables for each problem: 18 for the chomper and 38 for the twist chomper.

The twist chomper problem is a scaled-up version of the chomper problem, where the goal is to maximize the displacements at the corner nodes in the directions indicated in Fig. 4.1(b). Because the output nodes are at the corners, new designs with twisting are favored. This is in contrast to the chomper problem where folding is favored because the output nodes are at the middle of the edges rather than the corners.

For the chomper problem, we take $EA/G_{stiff} = 10$ and $G_{stiff}/G_{soft} = 10^4$. The fold line constraint value is taken to be $\nu_0 = 0.66$ so that a maximum of six active fold-lines are allowed. The displacement of $u_z = 0.03$ is applied in the negative z direction on the input nodes. The maximum number of finite element solutions is set to 100.

For the twist chomper, we use the ratios $EA/G_{stiff} = 10^3$ and $G_{stiff}/G_{soft} = 10^3$. The fold line constraint is set to $\nu_0 = 0.77$, which translates to a maximum of eight allowable active fold lines. Similar to chomper problem, $u_z = 0.03$ displacement is applied in the negative z direction on the input nodes. The maximum number of finite element solutions is set to 200.

4.2 Comparison of Optimization Algorithms

Figure 4.2 compares the performance of the BO and gradient-based sequential quadratic programming algorithm for the chomper problem, with the best objective function value plotted against the number of FE solutions. Note that the number of FE solutions is the same as the size of the training set for the GP surrogate model in Bayesian optimization. The

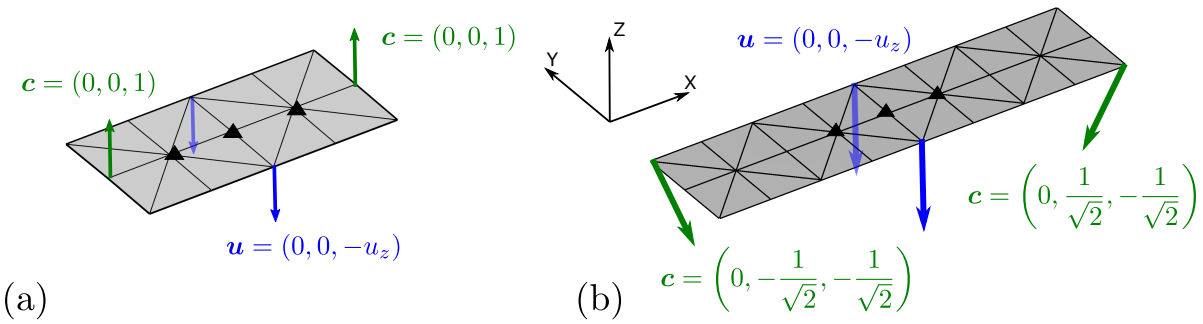


Figure 4.1: Origami problem setup: (a) Chomper problem, (b) Twist chomper problem. The black triangles indicate fixed nodes. Blue arrows denotes applied displacements, $\mathbf{u}$. Green arrows denote the objective function displacements.

results from four choices of covariance function are compared to the result from the SQP method (which is in fact the best of two runs with different initial guesses). The insets show the top views of the resulting fold patterns, and the three-dimensional views are shown on the sides. The red lines indicate the truss elements joining the fixed nodes. To better visualize the results, fold angles less than 2 degrees are ignored. For this study, the hyperparameters are determined at every iteration ($p = 1$) by maximizing the likelihood function. The initial training set consists of five randomly chosen points. We see that the BO results consistently perform better than the SQP algorithm and find new fold patterns (Designs III and IV).

Figure 4.3 shows a similar comparison for the twist chomper problem, along with two- and three-dimensional views of the fold patterns. Again, the BO approach achieves better solutions than the gradient-based SQP method for each of the four covariance functions. The SQP solution seems to find a local optimum early on, with no improvement in subsequent iterations, as can be expected from such a method. However, the stochastic BO approach doesn't suffer from this drawback. The choice of Matérn5 and Matérn3 leads to the best fold pattern (IV), while the squared exponential and Matérn1 covariance functions result in a slightly inferior Design III.

Gillman et al. [2] used a genetic algorithm to find optimal actuations of various origami folding structures. In order to compare its performance with BO, GA was tested on the chomper and twist chomper problems with the same problem parameters. For both problems, 560 chromosomes were considered in the population. For the chomper problem, GA took six generations i.e., a total of $560 \times 6 = 3360$ finite element solutions to converge to Design IV from Fig. 4.2. For the twist chomper problem, GA required 20 generations, i.e.,

$560 \times 20 = 11200$ finite element solves to converge to Design IV from Fig. 4.3. Although GA is able to find the best solution, it requires many more expensive function solutions compared to BO. This clearly illustrates the superior performance of BO.

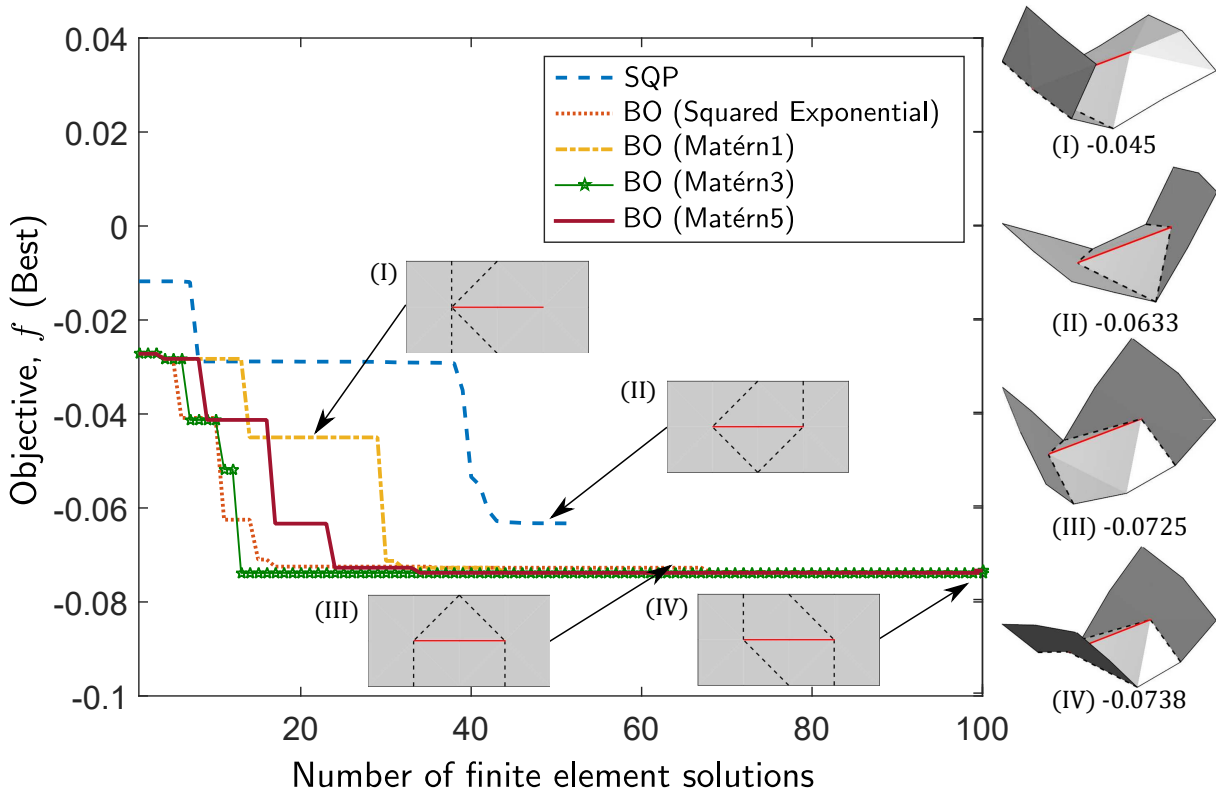


Figure 4.2: Evolution of the objective function and corresponding fold patterns (insets) and actuations (right) from Bayesian optimization and gradient-based SQP method for the chomper problem. Hyperparameters are estimated at every iteration $(p = 1)$ for the BO solutions. Red solid line on the fold patterns indicate truss elements connecting fixed nodes. The dashed black lines indicate the active folds.

4.3 The Role of the Initial Training Set

BO requires an initial training set to train the GP surrogate model. Here we study how the choice of the points in this training set and its size affect the performance of the method.

4.3.1 Sensitivity to the Initial Training Set

To study the sensitivity of BO to the initial training set, we consider five different training sets, each with five randomly selected points. Figure 4.4(a) shows the convergence history

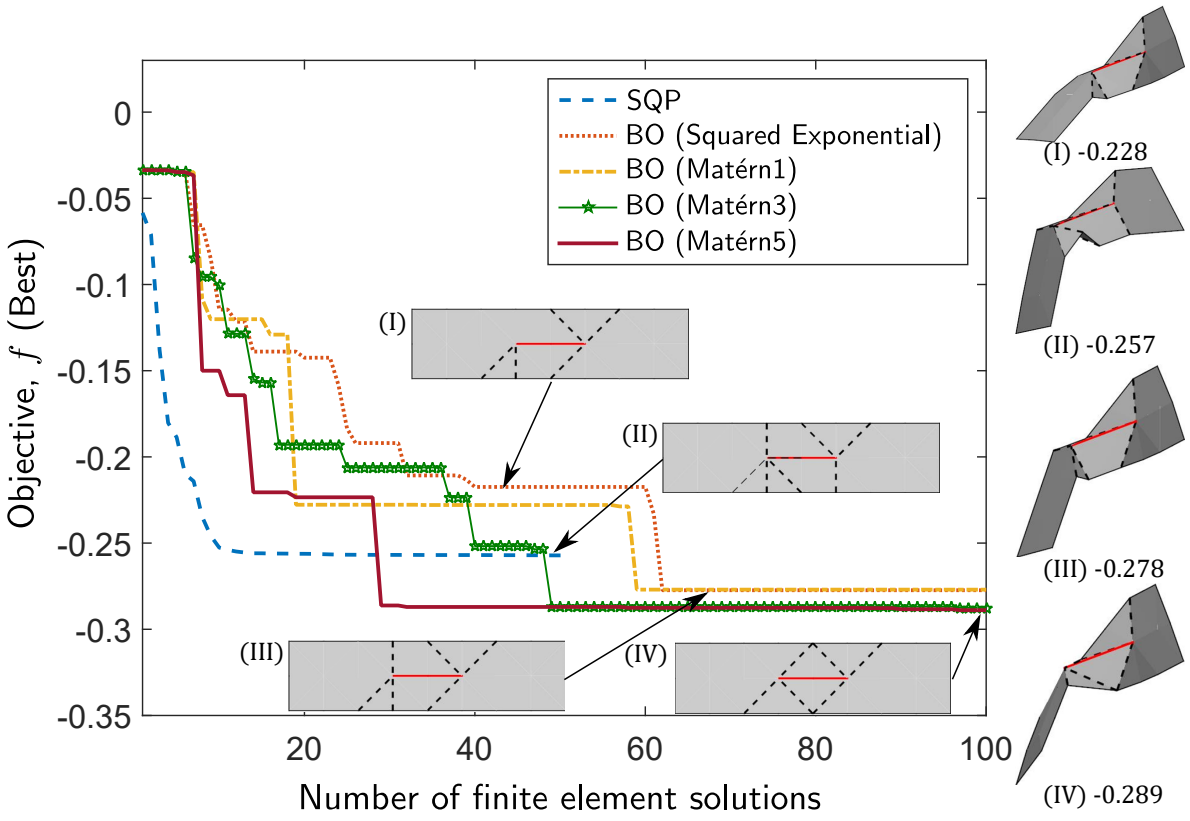


Figure 4.3: Evolution of the objective function and corresponding fold patterns (insets) and actuations (right) from Bayesian optimization and gradient-based SQP method for the twist chomper problem. Hyperparameters are estimated at every iteration $(p = 1)$ for the BO solutions. Red solid line on the fold patterns indicate truss elements connecting fixed nodes. The dashed black lines indicate the active folds.

for the chomper problem using the squared exponential covariance function. In each case, the procedure converges to the best solution we have found to date in our work, with a function value of -0.0738. As a comparison, we study the sensitivity of the gradient based SQP method to the initial point as follows for the chomper problem. From each of the five training sets used for BO, we pick the point with the best objective function value and set it as the initial guess for the SQP method. The resulting evolution of the objective function for the five initial guesses is shown in Fig. 4.4(b), with each run leading to a different optimal design. The best design obtained with the SQP method is with initial guess 1, which converges to Design V with an objective function value -0.0736.

A similar comparison is carried out for the twist chomper using BO with the Matérn1 covariance function and five training sets with five points each. Figure 4.5(a) shows the convergence history for the same. This time, two of the initial training sets find the best solution we have found to date (Design III with $f(\mathbf{x}) = -0.289$). The other training sets lead

to slightly inferior solutions (3.8% and 7.2% lower performance). The results from the same study applied to the twist chomper problem using SQP are shown in Fig. 4.5(b). In each case, this gradient method is trapped in a sub-optimal design. The best design obtained by the SQP technique is with initial guess 3 with a objective function value of -0.257.

All the optimal solutions obtained by BO are better than those discovered by the gradient method. Overall, the BO approach is insensitive to the initial training set $(n = 5)$ for the chomper and twist chomper problems in comparison to the gradient method.

4.3.2 Sensitivity to the Initial Training Set Size

We now study the effect of using initial training sets of different sizes. We heuristically choose 5, 10, 15, 20 and 25 points for each set. Although 25 training points is small when compared to the complete design space, it is 1/4th and 1/8th of the total function evaluations considered for the chomper and twist chomper problems, respectively. The evolution of the objective function from BO using these different sizes is shown in Fig. 4.6(a) for the chomper problem and Fig. 4.7(a) for the twist chomper problem. The dashed vertical line in Fig. 4.6(a) and Fig. 4.7(a) separates the initial training set from the subsequent points predicted by the optimization procedure. For the chomper problem, the best possible solution (Design I in Fig. 4.6(a)) is found regardless of the size of the initial training set. The case with 10 initial guesses converges fastest followed by 15, 25, 5 and lastly 20 initial points. For the twist chomper problem, the case with 15 initial points converges fastest to the best solution (Design III), followed by the case with 5 initial guesses. The cases with 10 and 20 initial points find Design II and the case with 25 initial points finds Design I. In general, there is no discernible trend from these studies and a larger initial training set does not seem to guarantee a better solution or faster convergence. Also, there is no significant difference in the computational time for the cases with different sizes initial training sets as shown in Fig. 4.6(b) and Fig. 4.7(b). [1] This is because the additional computational time with 5 initial training points when compared to 25 initial training points, is the time taken to estimate hyperparameters and optimize acquisition function for iteration 6 to 25. As this computation accounts for very less time we don't see much difference in the overall time between these cases. As such, we use an initial size of five for all subsequent results in this work.

[1] More details of these computational plots are presented in Section 4.4.

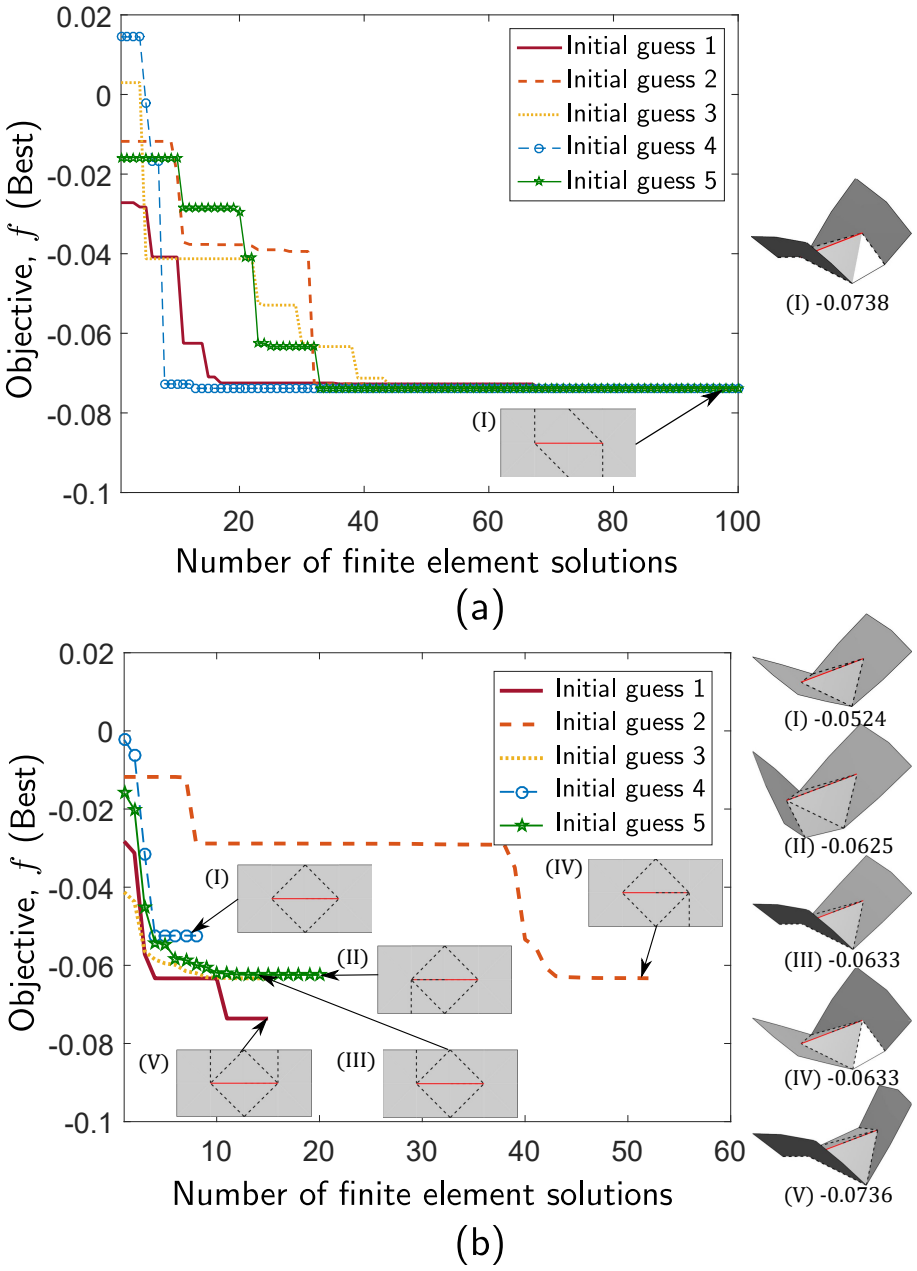


Figure 4.4: Sensitivity to the initial training set for the chomper problem (a) using BO with squared exponential and (b) using SQP. Evolution of the objective function and corresponding fold patterns (insets) and actuations (right) from for different initial training sets. Red solid line on the fold patterns indicate truss elements connecting fixed nodes. The dashed black lines indicate the active folds.

Figures 4.2 through 4.7 highlight the differences in the design space of the chomper and twist chomper problems. The twist chomper problem has a larger design space with fewer symmetries than the chomper problem. This makes optimizing the twist chomper more difficult and challenging. As seen in Fig. 4.2 through 4.7, BO is successfully able to solve

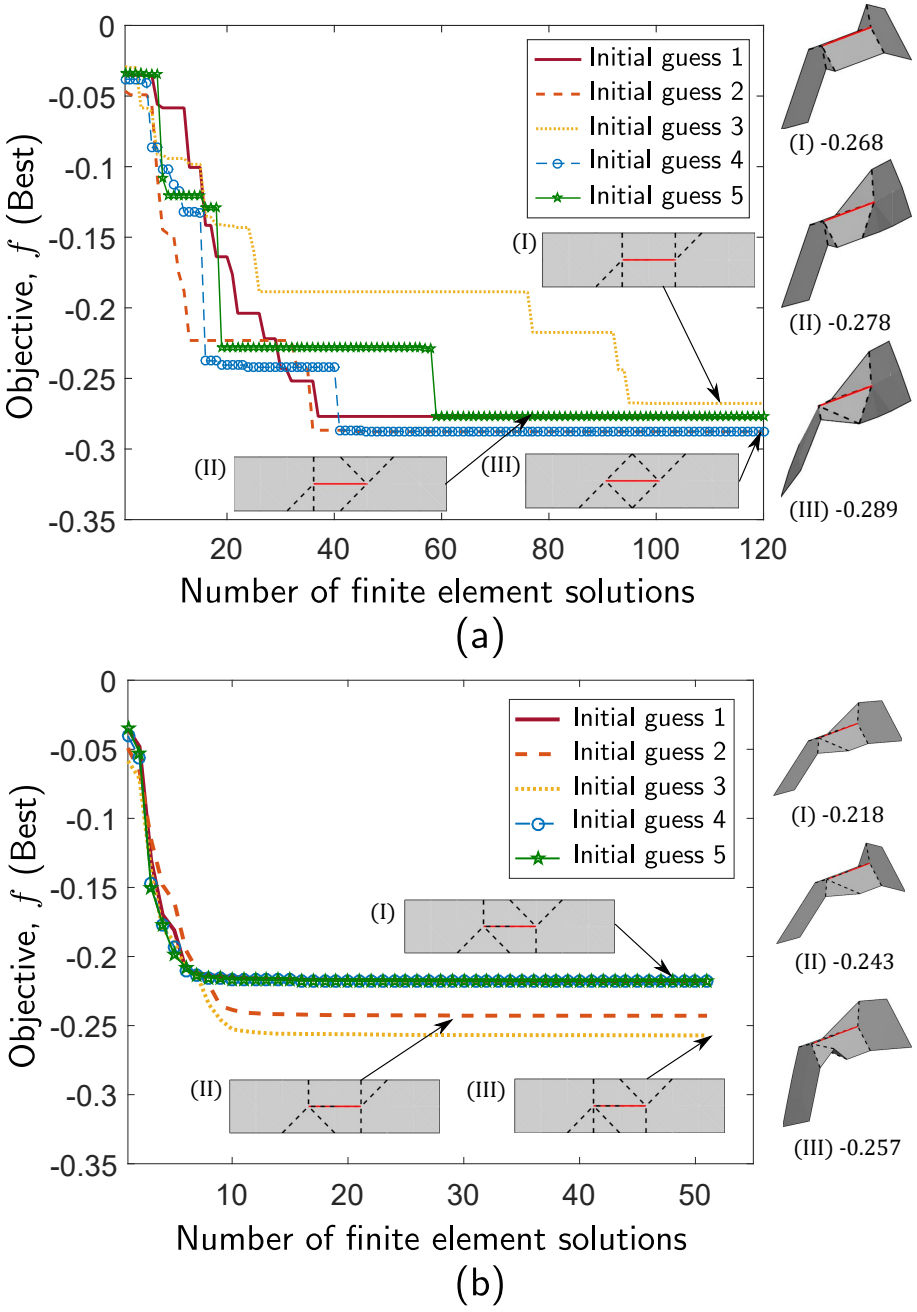
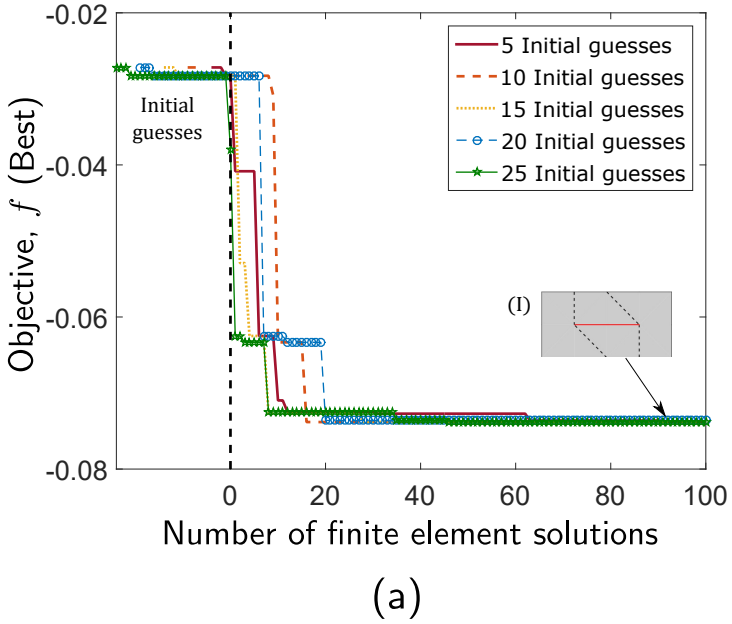
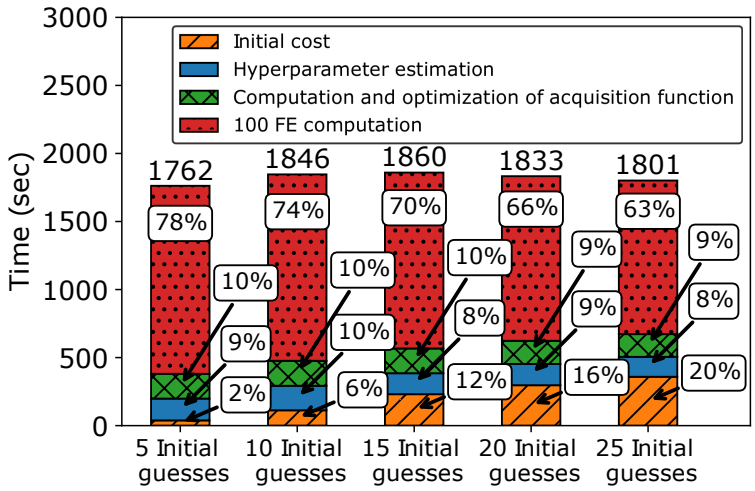


Figure 4.5: Sensitivity to the initial training set for the twist chomper problem (a) using BO with Matérn1 covariance function and (b) using SQP. Evolution of the objective function and corresponding fold patterns (insets) and actuations (right) from for different initial training sets. Red solid line on the fold patterns indicate truss elements connecting fixed nodes. The dashed black lines indicate the active folds.

both these problems.



(a)

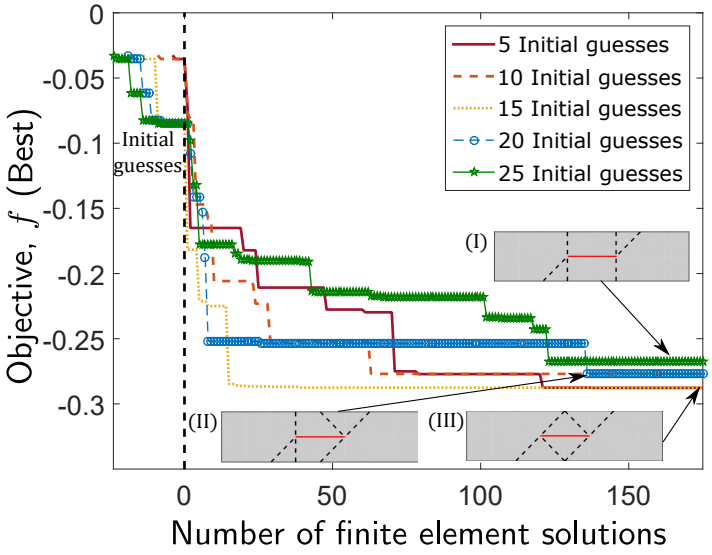


(b)

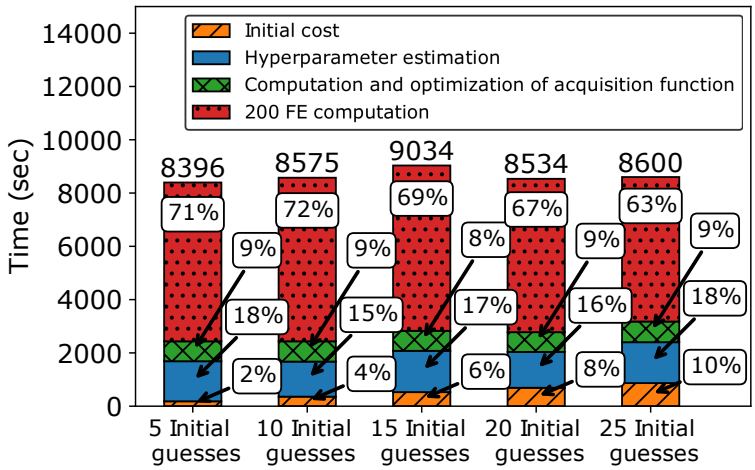
Figure 4.6: Sensitivity to initial training set size for the chomper problem. (a) Evolution of the objective function and corresponding fold patterns (insets) from Bayesian optimization for 5/10/15/20/25 initial training points. Dashed vertical line separates the initial training set from the points found through Bayesian optimization. Red solid line on the fold patterns indicate truss elements connecting fixed nodes. The dashed black lines on the fold patterns indicate the active folds. (b) Overall computational time for the five cases.

4.4 Computational Cost Analysis

For the BO approach, the overall computational time can be divided into three distinct categories: the time required to estimate hyperparameters, the time spent in computing the



(a)



(b)

Figure 4.7: Sensitivity to initial training set size for the twist chomper problem. (a) Evolution of the objective function and corresponding fold patterns (insets) from Bayesian optimization for 5/10/15/20/25 initial training points. Dashed vertical line separates the initial training set from the points found through Bayesian optimization. Red solid line on the fold patterns indicate truss elements connecting fixed nodes. The dashed black line on the fold patterns indicate the active folds. (b) Overall computational time for the five cases.

covariance matrix and optimizing the acquisition function, and the time required to evaluate the expensive objective function. Various heuristic approaches can be applied to reduce the overall computational time. In the next set of studies, we consider the following questions: (a) How frequently should the hyperparameters be estimated? (b) What is the overall benefit

of hyperparameter optimization? and (c) Does limiting the size of the training set affect the performance of the method?

4.4.1 Frequency of Hyperparameter Estimation

As discussed in Section 3.1, the hyperparameters used to define the covariance function can be estimated by maximizing the log-likelihood function. At a given iteration n, the log-likelihood function depends on the points 1 through $n-1$. Thus the optimized hyperparameters are not representative of the complete design space. Therefore estimating the hyperparameters every iteration may not be very efficient. Instead, it may suffice to estimate them less frequently and save the associated computational expense.

As an example, we consider estimating the hyperparameters every ten iterations $(p = 10)$ and compare the results to the case where the hyperparameters are estimated every iteration $(p = 1)$. For the chomper problem, the evolution of the objective function is shown for these two cases in Fig. 4.8(a), where the squared exponential covariance function is used. The computational costs for the two cases are shown in Fig. 4.8(b). The same initial training set is used for both cases. We see that both cases discover the best fold pattern (Design I). The computational time in estimating the hyperparameters is reduced considerably, from 153 seconds (24% of total time) to 13 seconds (3% of total time) when the hyperparameters are estimated every ten iterations $(p = 10)$. The estimated hyperparameters are shown in Fig. 4.8(c) and Fig. 4.8(d). The hyperparameters from the two approaches are in the same range. Figures 4.9, 4.10 and 4.11 compare the cases of hyperparameter estimation with $p = 1$ and $p = 10$ for the Matérn1, Matérn3 and Matérn5 covariance functions, respectively. In all these cases we see that the hyperparameter estimation time is reduced from around 22% of the total time to around 4% of the total time when the hyperparameters are estimated every 10 iterations. Also, regardless of the covariance function, the best fold pattern (Design I) is discovered for both $p = 1$ and $p = 10$. This indicates that the BO algorithm doesn't suffer when the frequency of estimating hyperparameters is reduced.

A similar study is carried out for the twist chomper problem. Figure 4.12(a) compares the evolution of the objective function for the two scenarios using the squared exponential covariance function and the same set of initial points. For this problem, interestingly, estimating the hyperparameters every ten iterations $(p = 10)$ leads to the discovery of a better

fold pattern (Design II) than by estimating them every iteration (Design I). The hyperparameters estimated every iteration might be over-fitting the data and thus may cause poor performance when new training points are appended in comparison to the the case where hyperparameters are estimated every ten iterations. Figure 4.12(b) shows that the reduction in the frequency of hyperparameter estimation leads to an overall improvement of 31% in computational time. The relative time spent in hyperparameter estimation is reduced from 1238 seconds (33% of total time) to 141 seconds (5% of total time). Figures 4.12(c) and 4.12(d) show the estimated hyperparameters for the two cases. The hyperparameters are relatively flat when tuned every iteration $(p = 1)$ especially after about the first 30 finite element solutions. Similar conclusions can be drawn with Matérn1, Matérn3 and Matérn5 covariance functions. The plots for the evolution of BO, overall computational time and estimated hyperparameters for these cases are shown in Figs. 4.13, 4.14 and 4.15. Interestingly, regardless of the covariance function the case where hyperparameters are estimated every 10 iterations performs better in terms of discovering the best solution. Additionally, estimating hyperparameters every ten iterations instead of every iteration saves around 30% of overall time for the twist chomper problem. Moreover, the length scale parameter, l, and the amplitude parameter, θ_0, seem to follow similar trends when estimated every iteration and every 10 iterations.

All these results strongly indicate that BO stands to gain in terms of efficiency and design discovery by reducing the frequency of hyperparameter estimation.

4.4.2 Fixed Hyperparameters

As seen in Subsection 4.4.1, it is not necessary to update the hyperparameters every iteration in order to find an efficient origami structure. This raises the question whether the tuning of the hyperparameters can be avoided altogether. In an attempt to answer this, we now study the performance of the BO approach with fixed values for the two hyperparameters: the amplitude parameter, θ_0, and the length-scale parameter, l.

The amplitude parameter in the covariance function is related to the overall range of the objective function (T), and the length-scale parameter is related to the maximum Euclidean distance $(\sqrt{d})$ in the parameter space of the given problem. We allow the hyperparameters θ_0 and l to take on the following values: $\theta_0 = \{0.5\sqrt{T}, 1\sqrt{T}, 2\sqrt{T}\}$ and

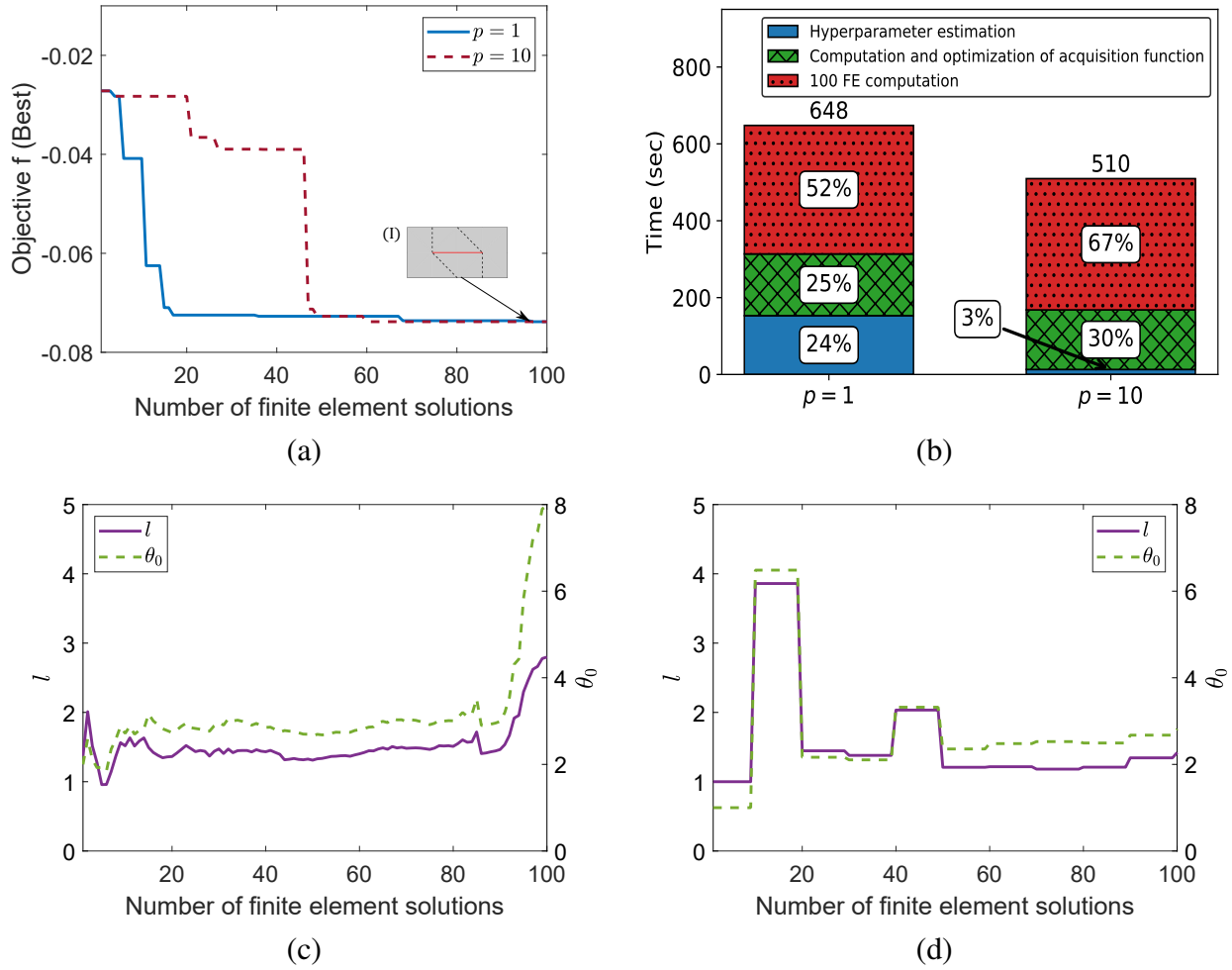


Figure 4.8: Frequency of hyperparameter estimation: Chomper problem with squared exponential covariance function. (a) Evolution of the objective function and corresponding fold pattern (inset) from Bayesian optimization when estimating the hyperparameters every iteration $(p = 1)$ versus every 10 iterations $(p = 10)$. Red solid lines on the fold patterns indicate truss elements connecting fixed nodes. The dashed black lines indicate active folds. (b) Computational time for the two cases. (c) Evolution of the hyperparameters for $p = 1$, and (d) Evolution of the hyperparameters for the case $p = 10$.

$l = \{0.5\sqrt{d},\ 1\sqrt{d},\ 2\sqrt{d}\}$, where T is the size of the range of objective function values and d is the dimension of the problem. Note that $T = 0.16$, $d = 18$ for the chomper problem and $T = 0.6$, $d = 38$ for the twist chomper problem. The choice of these particular values is based on a preliminary analysis that showed the hyperparameters to vary in this range. Of course, for a general optimization problem, the value of T is not known a priori, but information about its range may be known from other optimization techniques. In any case, our goal here is to understand the implications of using fixed hyperparameters.

Figures 4.16 to 4.23 compare the evolution of the cost function among selected sets of fixed

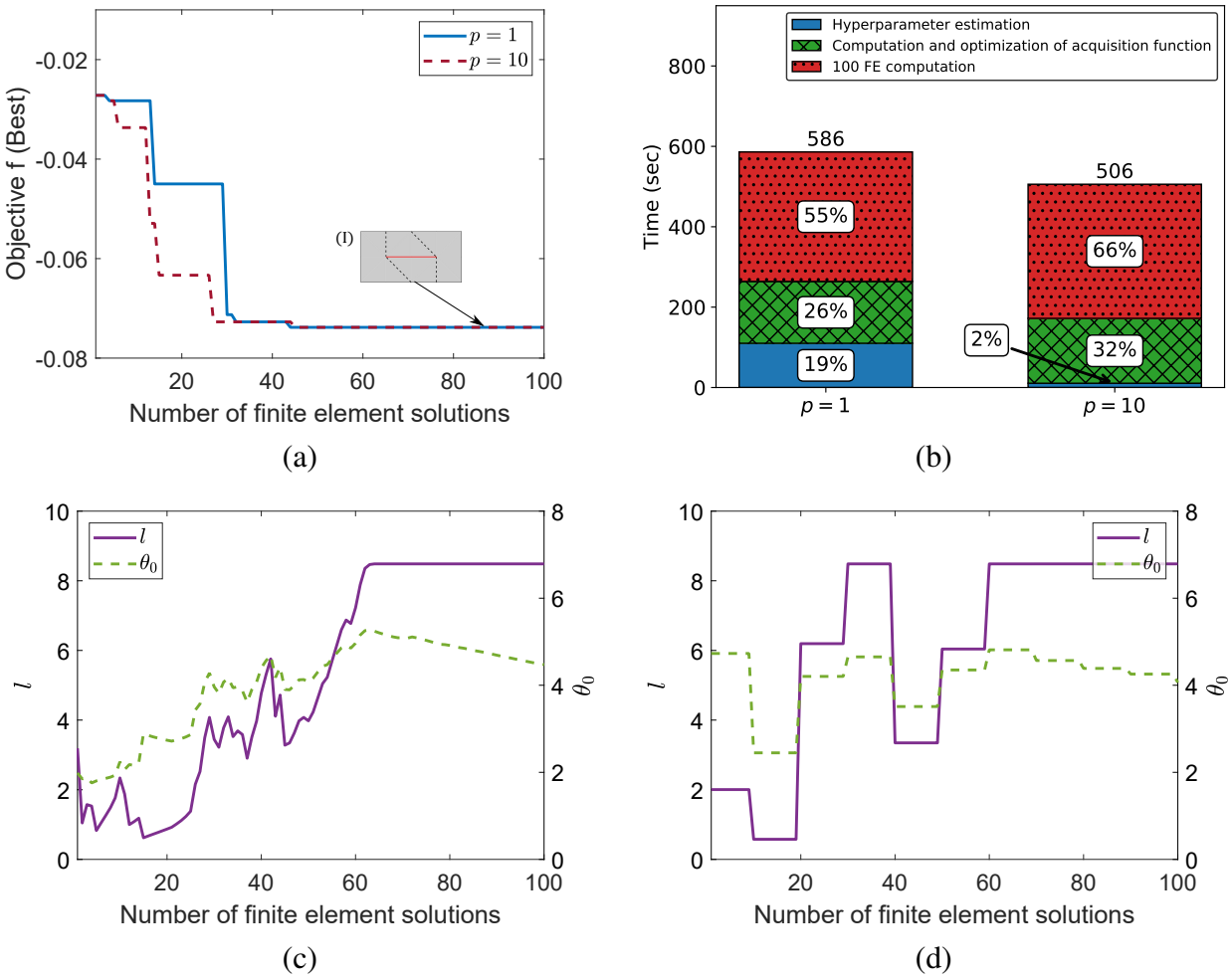


Figure 4.9: Frequency of hyperparameter estimation: Chomper problem with Matérn1 covariance function. (a) Evolution of the objective function and corresponding fold pattern (inset) from Bayesian optimization when estimating the hyperparameters every iteration $(p = 1)$ versus every 10 iterations $(p = 10)$. Red solid lines on the fold patterns indicate truss elements connecting fixed nodes. The dashed black lines indicate active folds. (b) Computational time for the two cases. (c) Evolution of the hyperparameters for $p = 1$, and (d) Evolution of the hyperparameters for the case $p = 10$.

hyperparameters for both problems using various covariance functions. These are compared to the case when the hyperparameters are estimated every iteration $(p = 1)$ by optimizing the log-likelihood.

For the chomper problem, all the cases with fixed hyperparameters find the best fold pattern except the case with $\left[l = 1\sqrt{d}, \theta_0 = 0.5\sqrt{T}\right]$ and squared exponential covariance (see Fig. 4.16(a)), which converges to a sub-optimal solution. For the chomper problem with fixed hyperparameters, the time spent in estimating hyperparameters is completely eliminated saving 22% to 24% of the overall time.

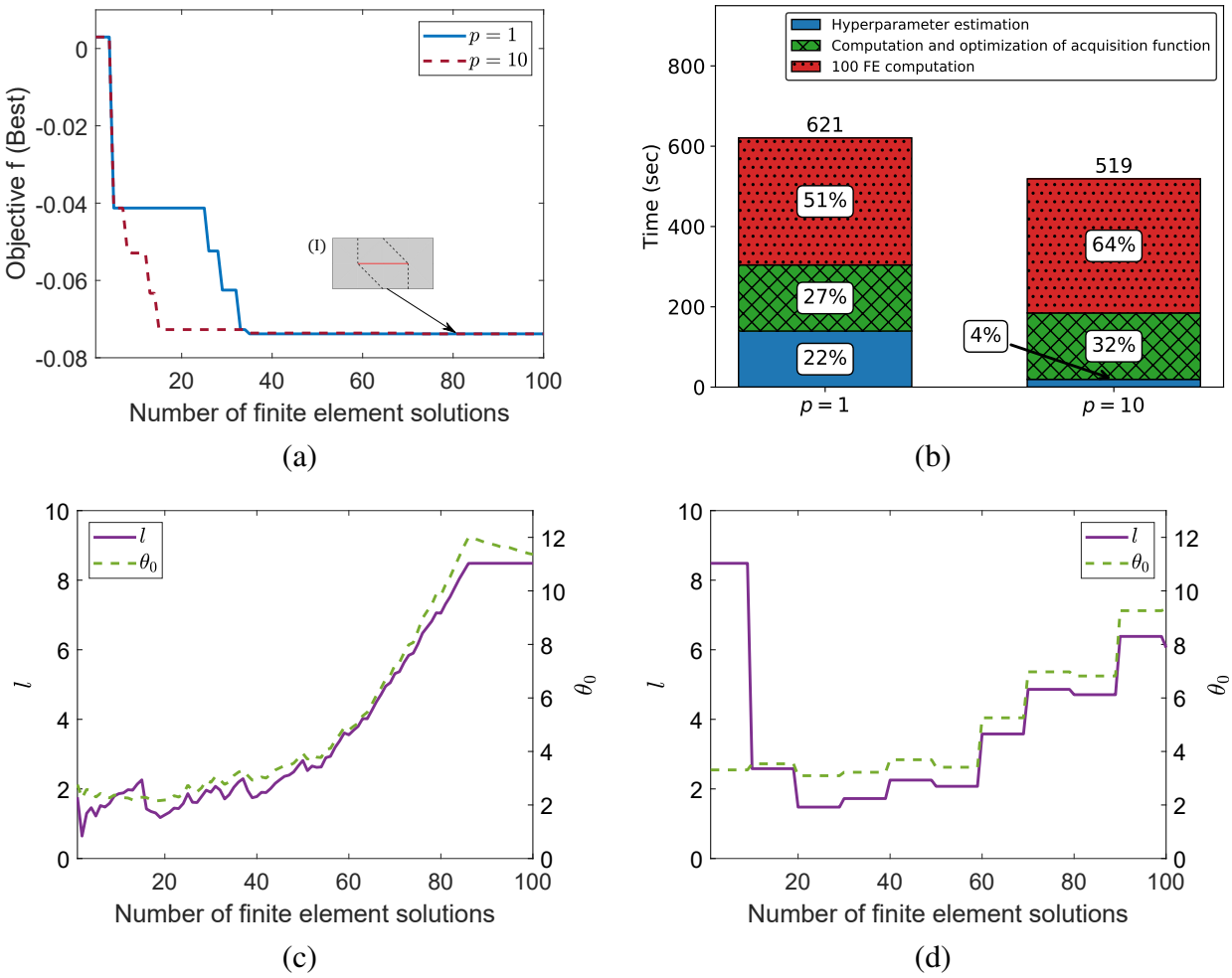


Figure 4.10: Frequency of hyperparameter estimation: Chomper problem with Matérn3 covariance function. (a) Evolution of the objective function and corresponding fold pattern (inset) from Bayesian optimization when estimating the hyperparameters every iteration $(p = 1)$ versus every 10 iterations $(p = 10)$. Red solid lines on the fold patterns indicate truss elements connecting fixed nodes. The dashed black lines indicate active folds. (b) Computational time for the two cases. (c) Evolution of the hyperparameters for $p = 1$, and (d) Evolution of the hyperparameters for the case $p = 10$.

For the twist chomper problem, the fixed hyperparameter solutions find better objective function values than when the hyperparameters are estimated every iteration, as seen in Figs. 4.20 to 4.23 using various covariance functions. However, the best set of fixed hyperparameters depends on the covariance function used. For the squared exponential covariance function, all the fixed hyperparameter cases discover better fold patterns than the case where hyperparameters are estimated every iteration. The fixed hyperparameter set $\left[l = 0.5\sqrt{d}, \theta_0 = 1\sqrt{T}\right]$ discovers the best possible fold pattern for the twist chomper problem. Fixed hyperparameter values $\left[l = 1\sqrt{d}, \theta_0 = 2\sqrt{T}\right]$ and $\left[l = 2\sqrt{d}, \theta_0 = 0.5\sqrt{T}\right]$

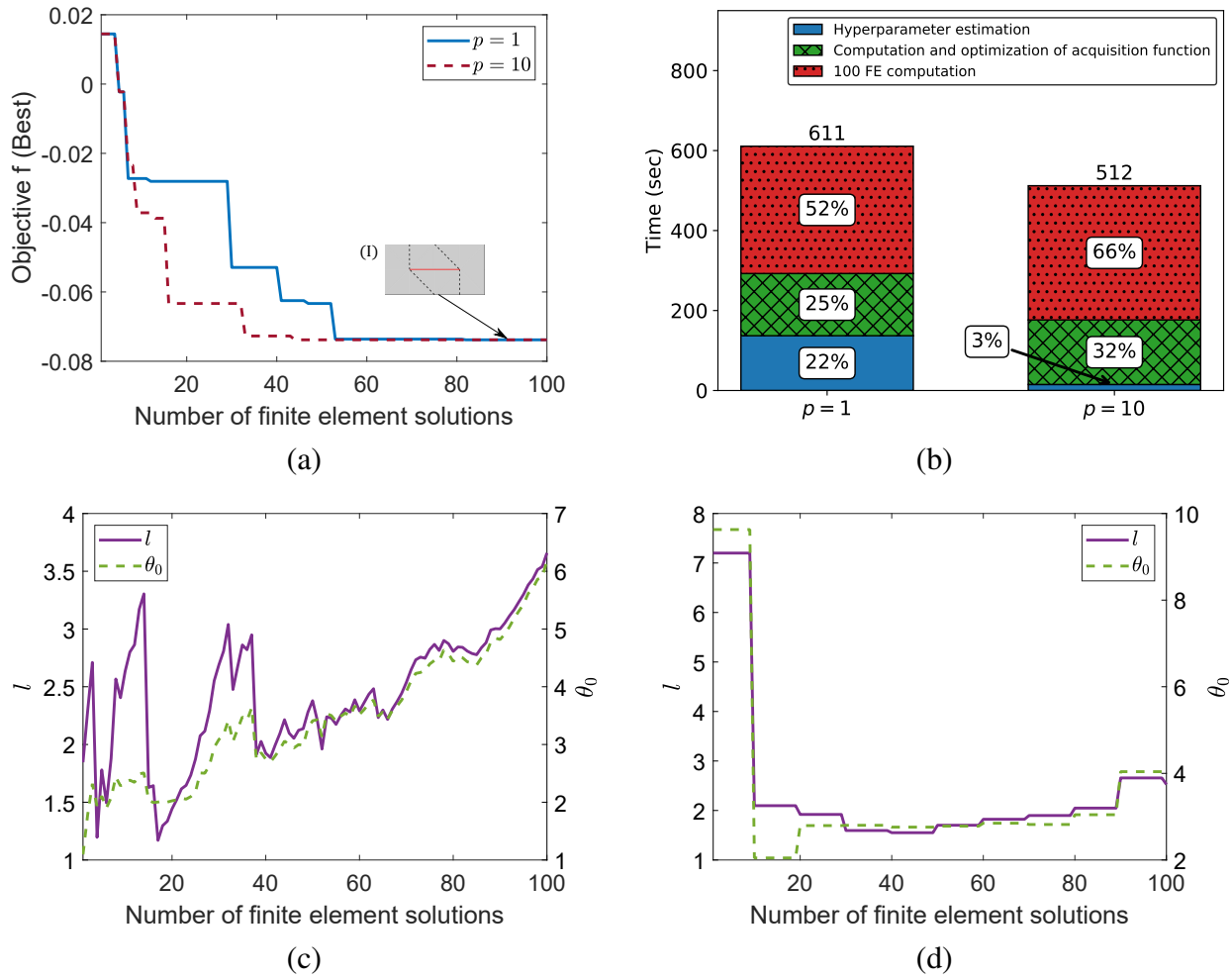


Figure 4.11: Frequency of hyperparameter estimation: Chomper problem with Matérn5 covariance function. (a) Evolution of the objective function and corresponding fold pattern (inset) from Bayesian optimization when estimating the hyperparameters every iteration $(p = 1)$ versus every 10 iterations $(p = 10)$. Red solid lines on the fold patterns indicate truss elements connecting fixed nodes. The dashed black lines indicate active folds. (b) Computational time for the two cases. (c) Evolution of the hyperparameters for $p = 1$, and (d) Evolution of the hyperparameters for the case $p = 10$.

for the Matérn1 covariance function, $\left[l = 0.5\sqrt{d}, \theta_0 = 1\sqrt{T}\right]$ and $\left[l = 2\sqrt{d}, \theta_0 = 0.5\sqrt{T}\right]$ for the Matérn3 covariance function and $\left[l = 0.5\sqrt{d}, \theta_0 = 1\sqrt{T}\right]$ and $\left[l = 1\sqrt{d}, \theta_0 = 2\sqrt{T}\right]$ for the Matérn5 covariance function find the best design for the twist chomper problem. Also as seen in Figs. 4.20 to 4.23, the hyperparameter estimation time of 24% to 45% of the overall time is completely avoided when the hyperparameters are fixed.

From the studies conducted on the chomper and twist chomper problem, the efficiency of the BO increases when hyperparameters are kept fixed as opposed to estimating them every iteration. However, this does not guarantee the global solution. Preliminary knowledge of the

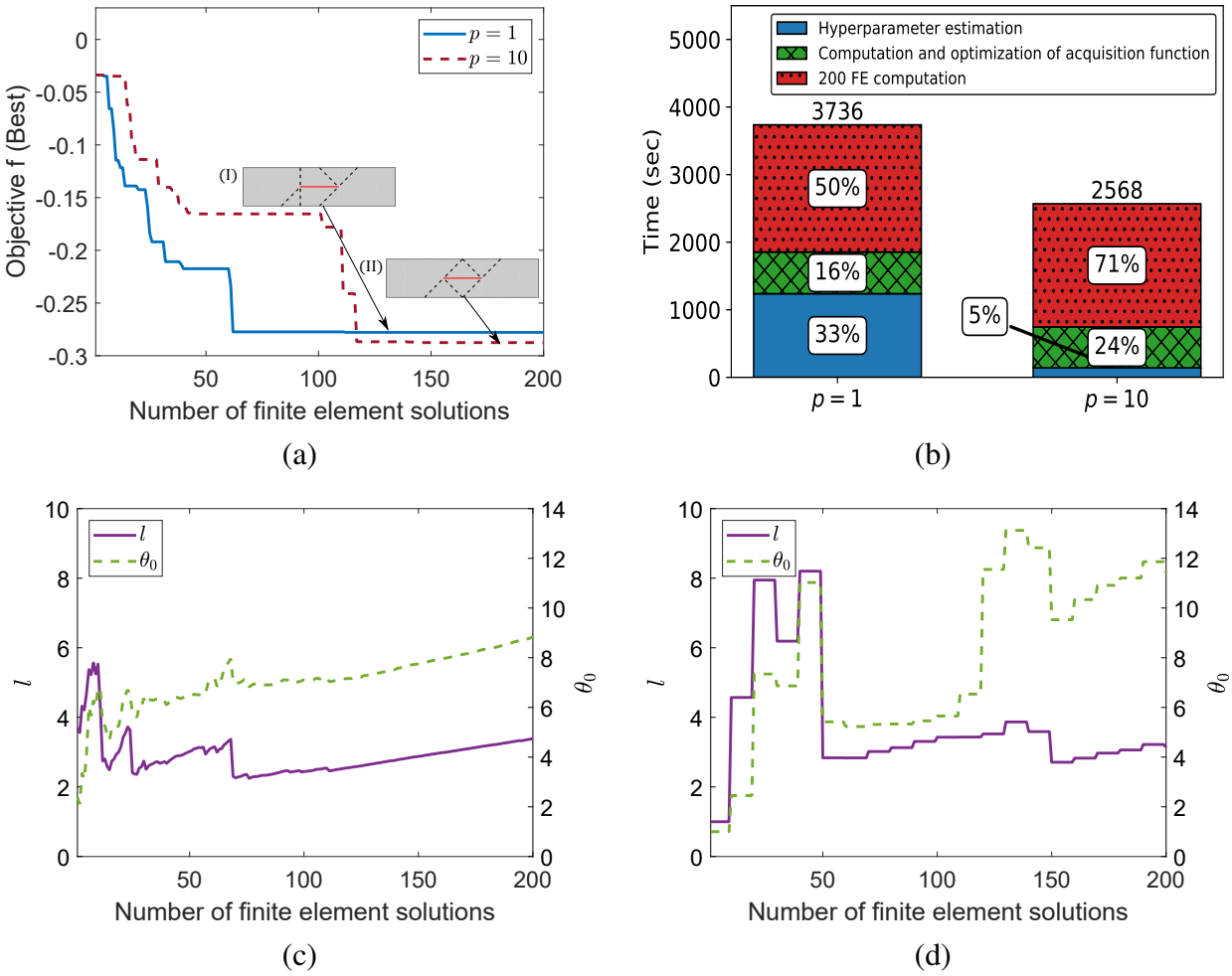


Figure 4.12: Frequency of hyperparameter estimation: Twist chomper problem with squared exponential covariance function. (a) Evolution of the objective function and corresponding fold pattern (inset) from Bayesian optimization when estimating the hyperparameters every iteration $(p = 1)$ versus every 10 iterations $(p = 10)$. Red solid lines on the fold patterns indicate truss elements connecting fixed nodes. The dashed black lines indicate active folds. (b) Computational time for the two cases. (c) Evolution of the hyperparameters for $p = 1$, and (d) Evolution of the hyperparameters for the case $p = 10$.

objective function is necessary in order to fix the hyperparameter values.

4.4.3 Limiting the Size of the Training Set

As discussed earlier, our BO implementation uses a Gaussian process surrogate model to model the original objective function. As seen in Eqs. (3.12) and (3.13), the evaluation of the mean μ_n and standard deviation σ_n of the surrogate model requires the LU decomposition of the covariance matrix $\mathbf{K}$. This is also needed for the calculation of the log-likelihood function used in hyperparameter estimation. The size of this dense matrix is equal to the number

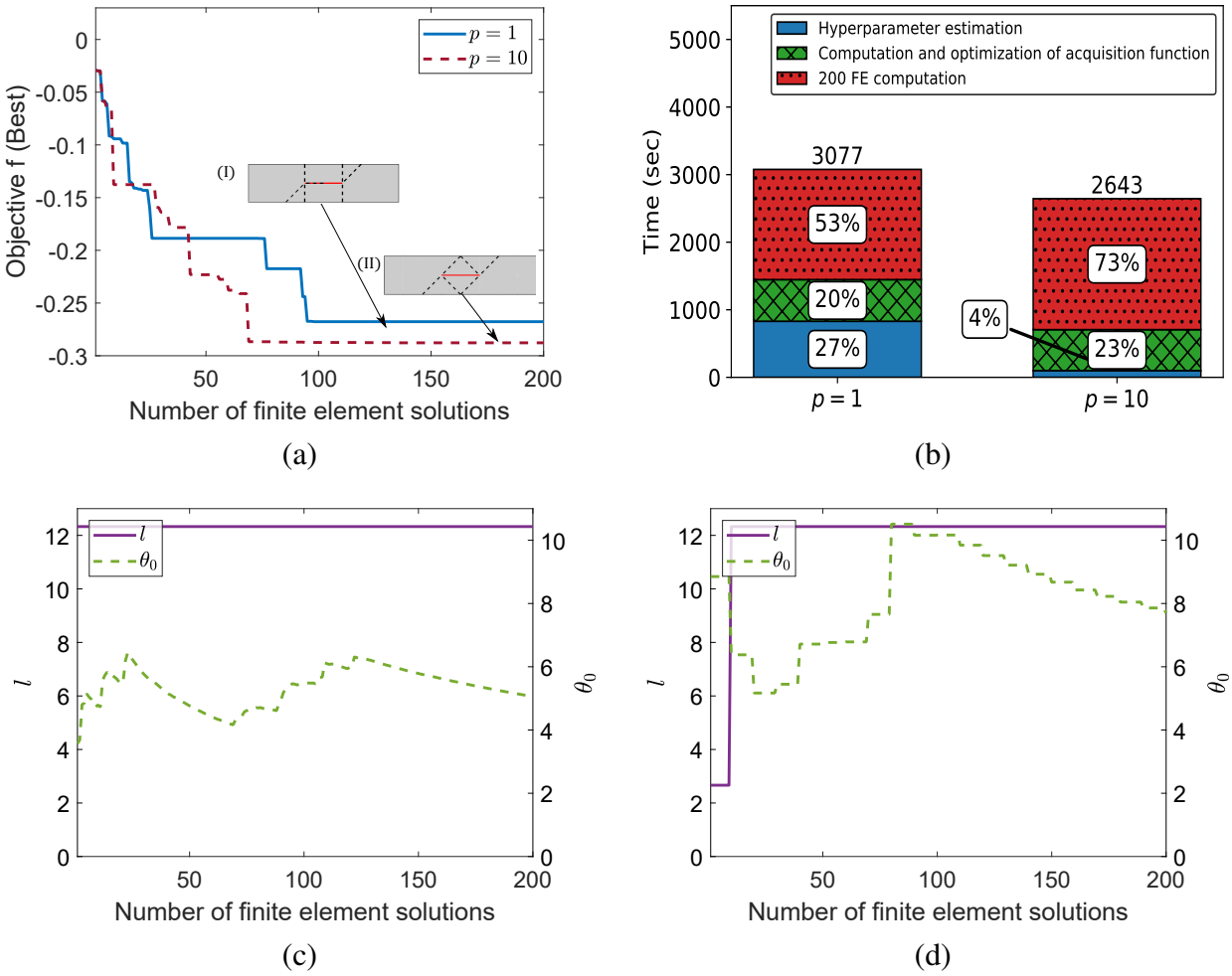


Figure 4.13: Frequency of hyperparameter estimation: Twist chomper problem with Matérn1 covariance function. (a) Evolution of the objective function and corresponding fold pattern (inset) from Bayesian optimization when estimating the hyperparameters every iteration $(p = 1)$ versus every 10 iterations $(p = 10)$. Red solid lines on the fold patterns indicate truss elements connecting fixed nodes. The dashed black lines indicate active folds. (b) Computational time for the two cases. (c) Evolution of the hyperparameters for $p = 1$, and (d) Evolution of the hyperparameters for the case $p = 10$.

of training points and grows with the number of iterations. The cost of decomposing the covariance matrix grows considerably with the size of the training data, since its complexity is $\mathcal{O}(n^3)$, where n is the number of training points.

To improve the surrogate model fit, especially in the vicinity of the best design, and to reduce the overall computational time, we consider removing training points with the poorest objective function values. Other strategies for pruning the training set are of course possible. As a case study, we eliminate 50 training points with the highest objective function values for the twist chomper problem after 100 iterations. This pruning of the training set is carried out

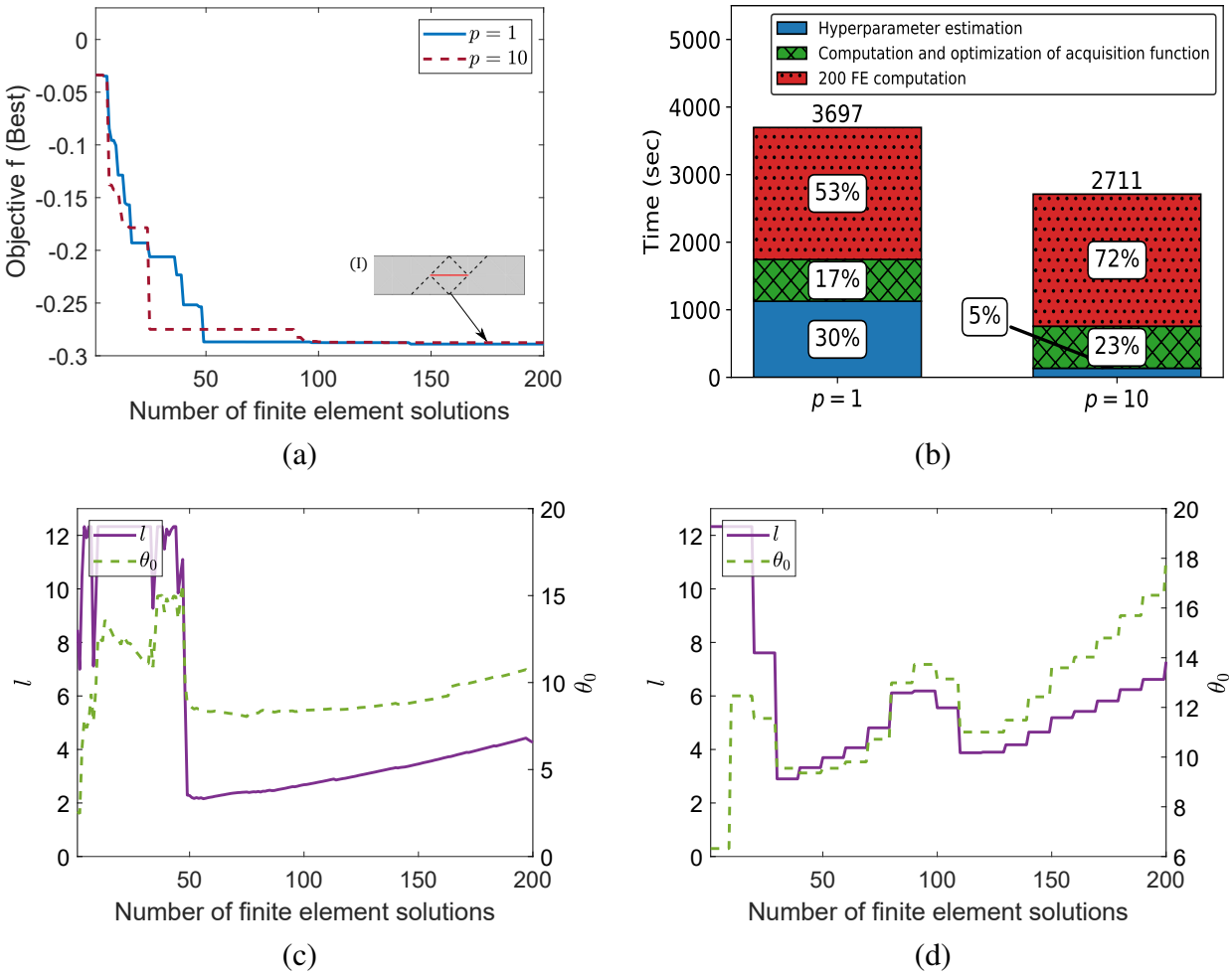


Figure 4.14: Frequency of hyperparameter estimation: Twist chomper problem with Matérn3 covariance function. (a) Evolution of the objective function and corresponding fold pattern (inset) from Bayesian optimization when estimating the hyperparameters every iteration $(p = 1)$ versus every 10 iterations $(p = 10)$. Red solid lines on the fold patterns indicate truss elements connecting fixed nodes. The dashed black lines indicate active folds. (b) Computational time for the two cases. (c) Evolution of the hyperparameters for $p = 1$, and (d) Evolution of the hyperparameters for the case $p = 10$.

just one time. Figure 4.24(a) plots the evolution of the objective function with and without the pruning of the training set. We see that the performance of the optimization procedure improves in this case and a better fold pattern (Design II) is discovered. Figure 4.24(b) compares the computational time for the two cases, and shows a considerable reduction in the time spent in estimating the hyperparameters (632 seconds to 543 seconds) and the time spent in computation and optimization of acquisition function (1386 seconds to 718 seconds).

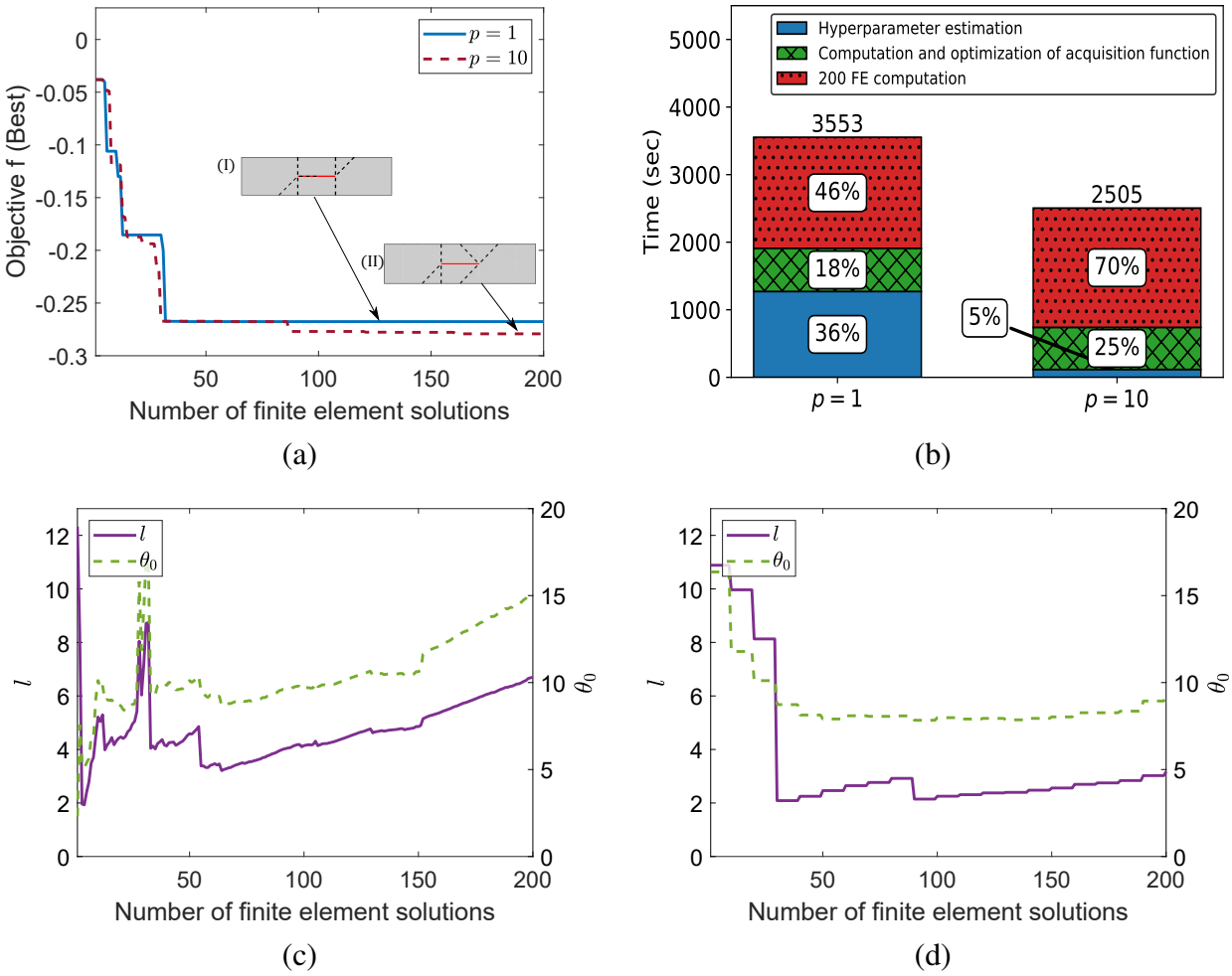
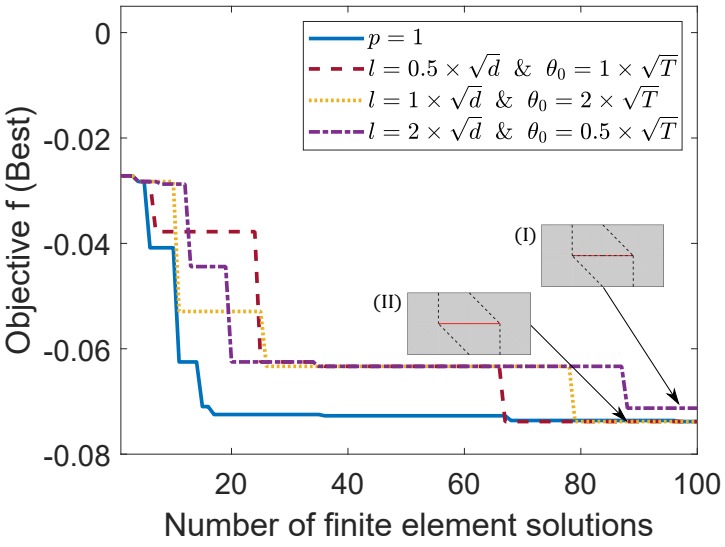


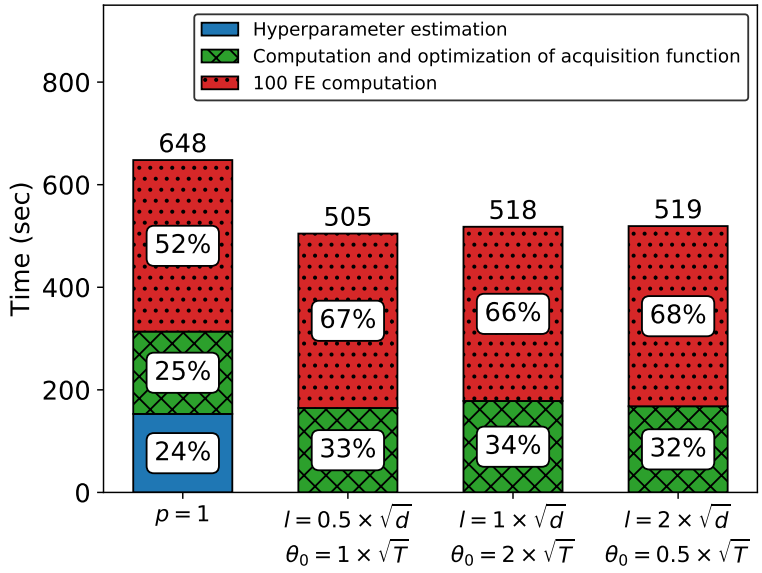
Figure 4.15: Frequency of hyperparameter estimation: Twist chomper problem with Matérn5 covariance function. (a) Evolution of the objective function and corresponding fold pattern (inset) from Bayesian optimization when estimating the hyperparameters every iteration $(p = 1)$ versus every 10 iterations $(p = 10)$. Red solid lines on the fold patterns indicate truss elements connecting fixed nodes. The dashed black lines indicate active folds. (b) Computational time for the two cases. (c) Evolution of the hyperparameters for $p = 1$, and (d) Evolution of the hyperparameters for the case $p = 10$.

4.5 Bayesian Optimization with Derivative Enrichment

As discussed in Chapter 3, derivative information about the expensive objective function can be utilized in the Bayesian optimization framework if readily available. With the derivative information, the covariance matrix now has elements representing correlations between function values, correlations between function and derivative, and correlations between derivative values. The covariance matrix increases in size to $n(d+1) \times n(d+1)$ from $n \times n$ in this framework, where n is the number of training points and d being the dimension of the



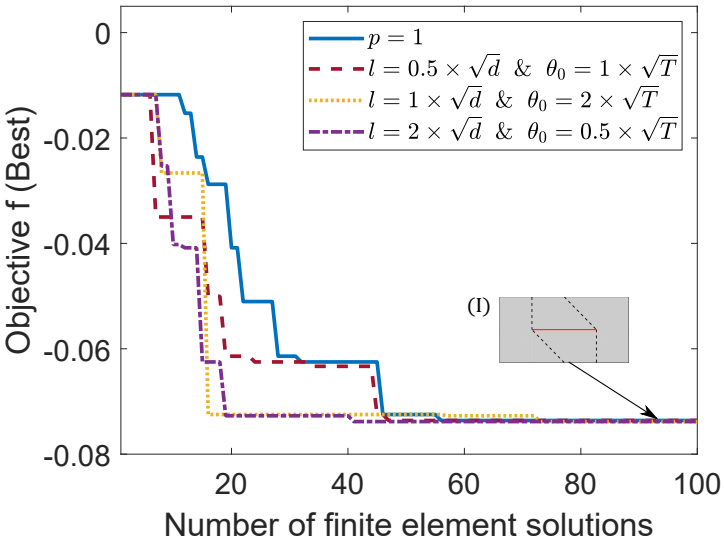
(a)



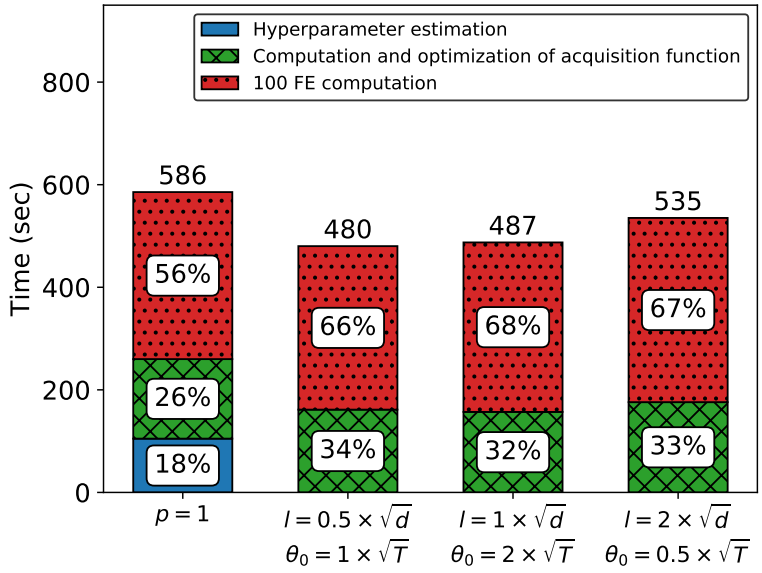
(b)

Figure 4.16: Fixed hyperparameters for the chomper problem with squared exponential covariance function. (a) Evolution of the objective function and corresponding fold patterns (insets) from Bayesian optimization when estimating the hyperparameters every iteration $(p = 1)$ versus holding them fixed throughout. Red solid lines on the fold patterns indicate truss elements connecting fixed nodes. Dashed black lines indicate the active folds. (b) Computational time for the various cases.

problem. In the case of the origami problem, all the partial derivatives at the training points are available with little extra computing time. We make use of these derivatives in the BO algorithm using the squared exponential covariance function and the lower confidence bound acquisition function. For the chomper problem which has 18 design variables, we obtain 18



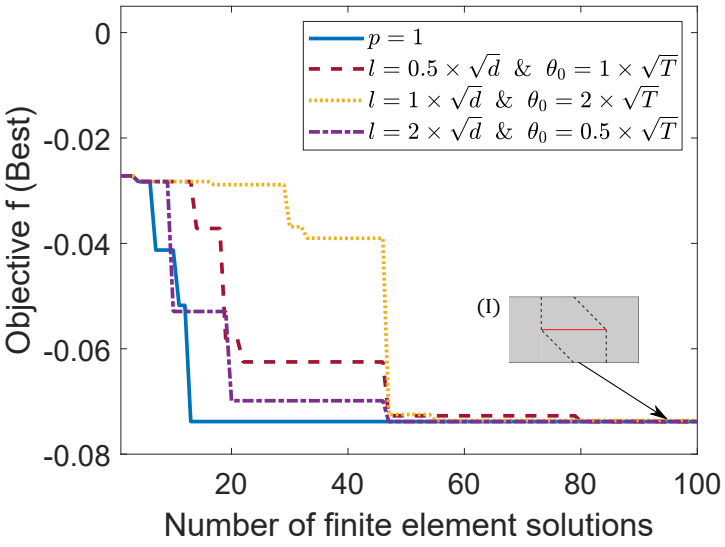
(a)



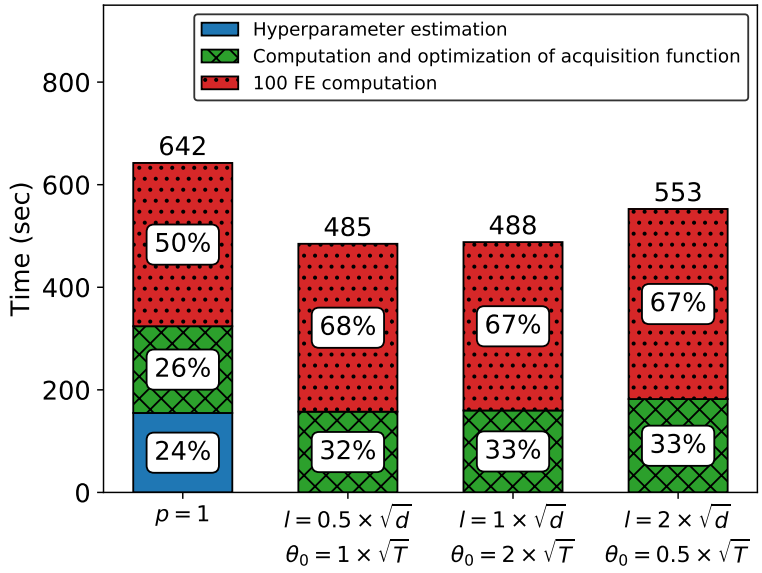
(b)

Figure 4.17: Fixed hyperparameters for the chomper problem with Matérn1 covariance function. (a) Evolution of the objective function and corresponding fold patterns (insets) from Bayesian optimization when estimating the hyperparameters every iteration ($p = 1$) versus holding them fixed throughout. Red solid lines on the fold patterns indicate truss elements connecting fixed nodes. Dashed black lines indicate the active folds. (b) Computational time for the various cases.

partial derivatives at a given training point. For the twist chomper problem which has 38 design variables, we have 38 partial derivatives with respect to the design variables. So, the covariance matrix with derivative information is $d + 1 = 19$ times larger than the original covariance matrix without derivative information for chomper problem. Similarly, for the



(a)

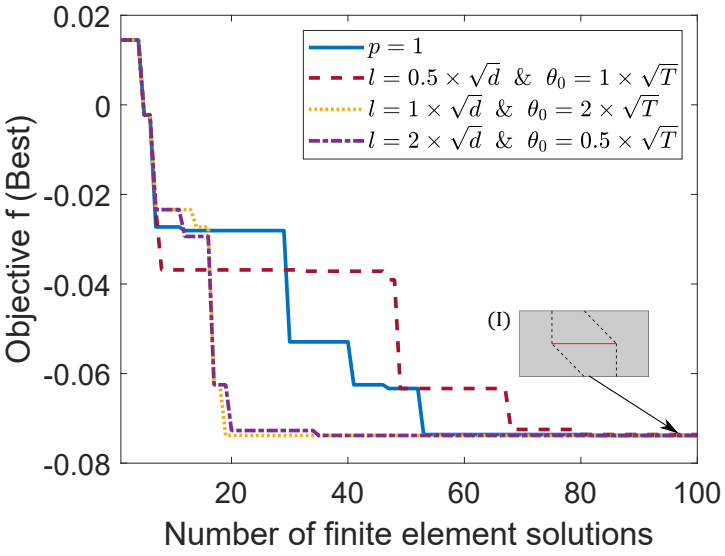


(b)

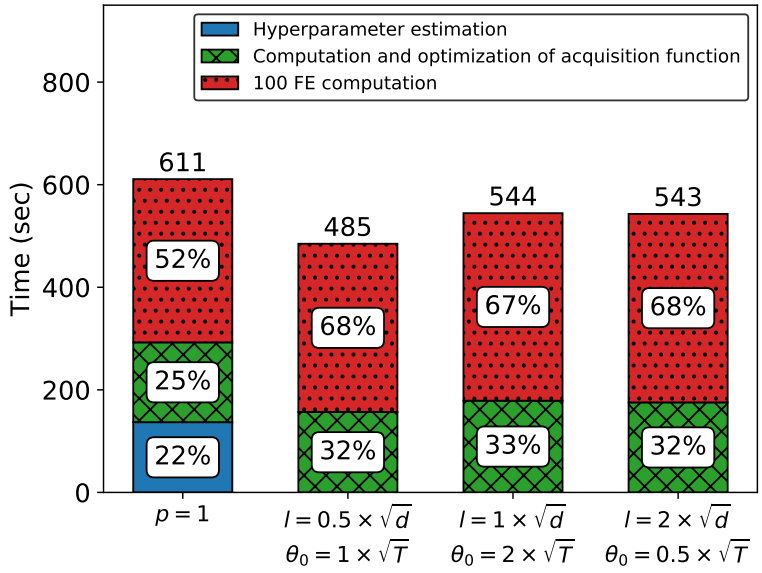
Figure 4.18: Fixed hyperparameters for the chomper problem with Matérn3 covariance function. (a) Evolution of the objective function and corresponding fold patterns (insets) from Bayesian optimization when estimating the hyperparameters every iteration $(p = 1)$ versus holding them fixed throughout. Red solid lines on the fold patterns indicate truss elements connecting fixed nodes. Dashed black lines indicate the active folds. (b) Computational time for the various cases.

twist chomper problem the covariance matrix becomes $d + 1 = 39$ times bigger than the original covariance matrix without derivative information. Thus, the cost of each iteration of Bayesian optimization goes up when derivative information is included.

Figure 4.25 compares the evolution of BO with and without derivative information for the



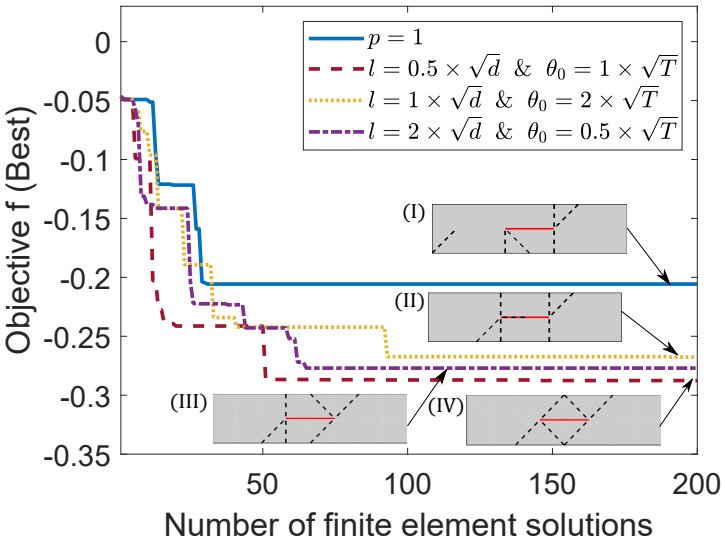
(a)



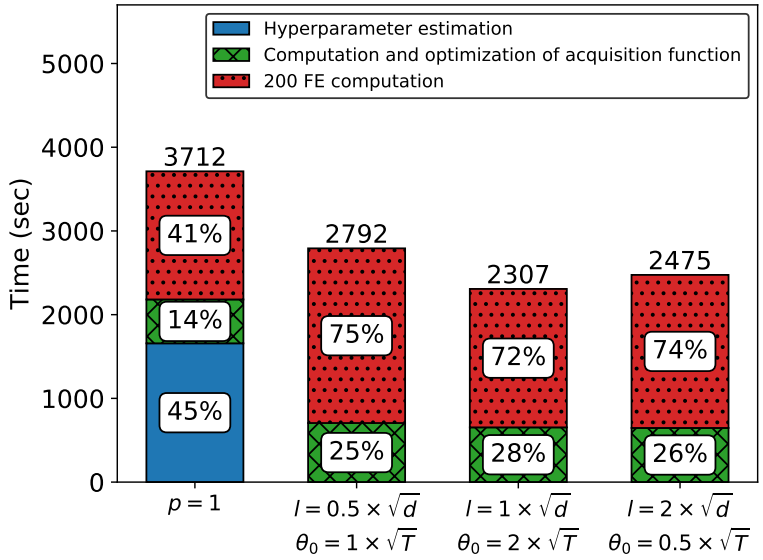
(b)

Figure 4.19: Fixed hyperparameters for the chomper problem with Matérn5 covariance function. (a) Evolution of the objective function and corresponding fold patterns (insets) from Bayesian optimization when estimating the hyperparameters every iteration $(p = 1)$ versus holding them fixed throughout. Red solid lines on the fold patterns indicate truss elements connecting fixed nodes. Dashed black lines indicate the active folds. (b) Computational time for the various cases.

chomper problem. Similar to previous studies, five random training points are considered as the initial guesses for both cases of BO. The best design (Design I) with objective function value $f(\mathbf{x}) = -0.0738$ is discovered in all five panels of Fig. 4.25. Also, BO with and without derivative information find the best design within 10 to 70 finite element solutions for the



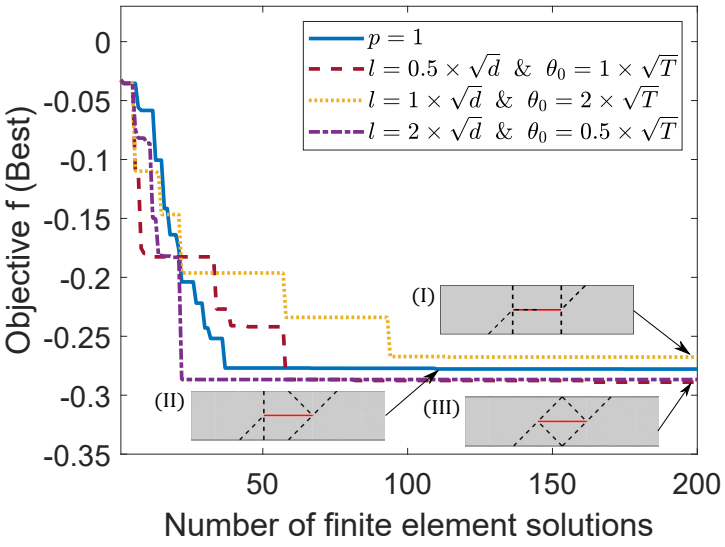
(a)



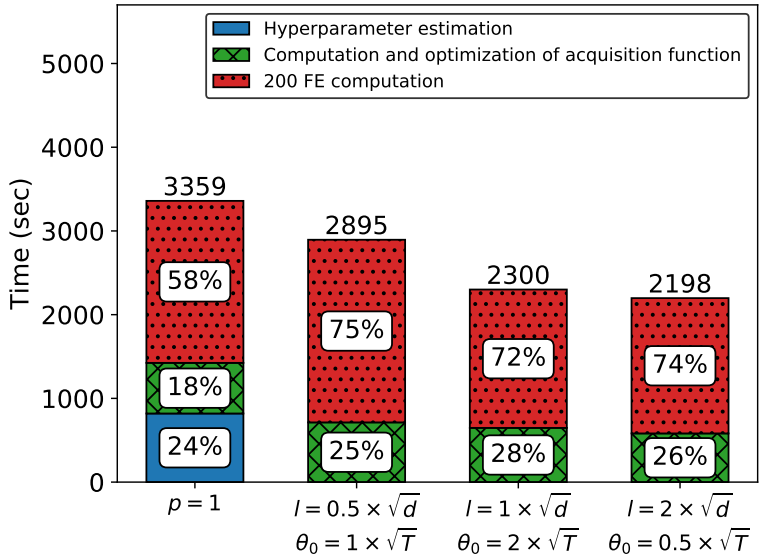
(b)

Figure 4.20: Fixed hyperparameters for the twist chomper problem with squared exponential covariance function. (a) Evolution of the objective function and corresponding fold patterns (insets) from Bayesian optimization when estimating the hyperparameters every iteration $(p = 1)$ versus holding them fixed throughout. Red solid lines on the fold patterns indicate truss elements connecting fixed nodes. Dashed black lines indicate the active folds. (b) Computational time for the various cases.

five initial guesses. Even though BO with derivative enrichment takes a similar number of expensive function evaluations, it requires more computational time as it involves calculating the inverse of a larger covariance matrix. This implies that no additional advantage is gained by including the derivative information in the search of optimal an fold pattern for the chomper



(a)

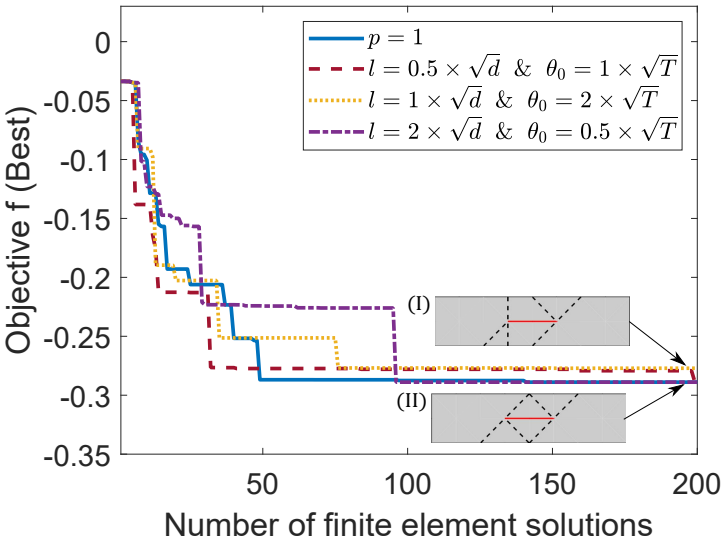


(b)

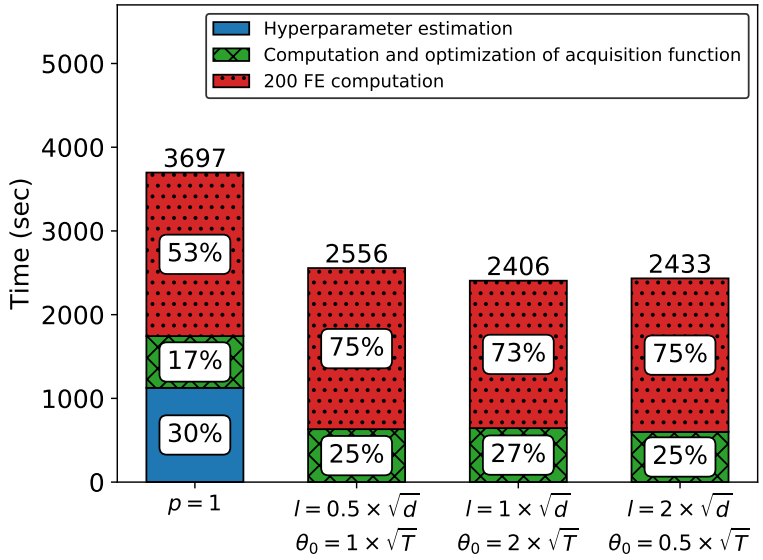
Figure 4.21: Fixed hyperparameters for the twist chomper problem with Matérn1 covariance function. (a) Evolution of the objective function and corresponding fold patterns (insets) from Bayesian optimization when estimating the hyperparameters every iteration $(p = 1)$ versus holding them fixed throughout. Red solid lines on the fold patterns indicate truss elements connecting fixed nodes. Dashed black lines indicate the active folds. (b) Computational time for the various cases.

problem.

Figure 4.26 compares the overall computational time with and without derivative information for the chomper problem. The computational time here is measured till the best design is discovered or the maximum number of training points (100) is reached. The cases



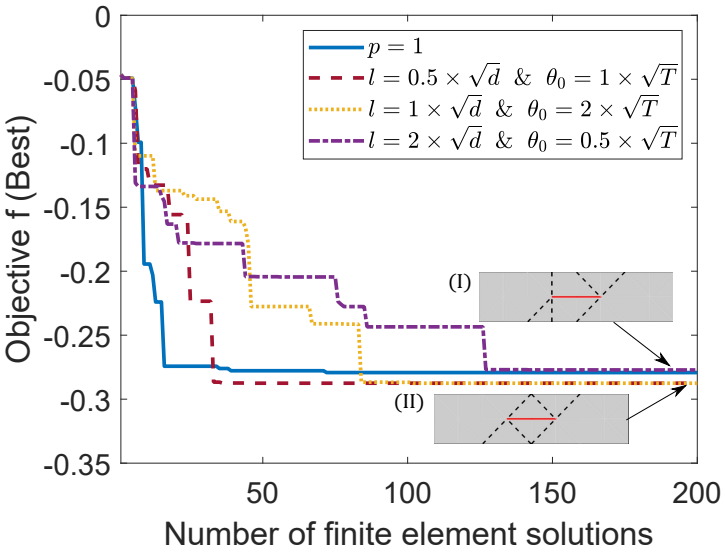
(a)



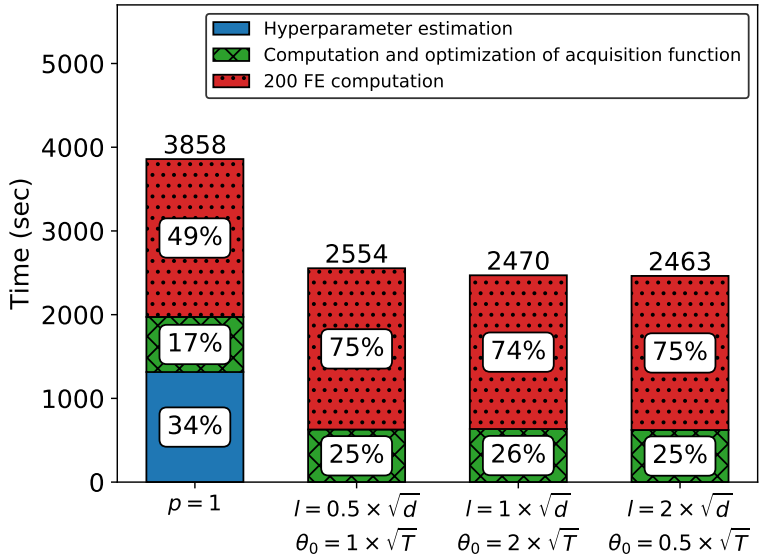
(b)

Figure 4.22: Fixed hyperparameters for the twist chomper problem with Matérn3 covariance function. (a) Evolution of the objective function and corresponding fold patterns (insets) from Bayesian optimization when estimating the hyperparameters every iteration $(p = 1)$ versus holding them fixed throughout. Red solid lines on the fold patterns indicate truss elements connecting fixed nodes. Dashed black lines indicate the active folds. (b) Computational time for the various cases.

with initial guess 2 (Fig. 4.26(b)) and initial guess 5 (Fig. 4.26(e)) take less overall time to find the best possible solution when derivative information is utilized. In all the other cases studied here, BO with derivative enrichment either takes more time (Fig. 4.26(d)) or does not find the global solution (Fig. 4.26(a) and 4.26(c)). This again indicates that utilizing



(a)

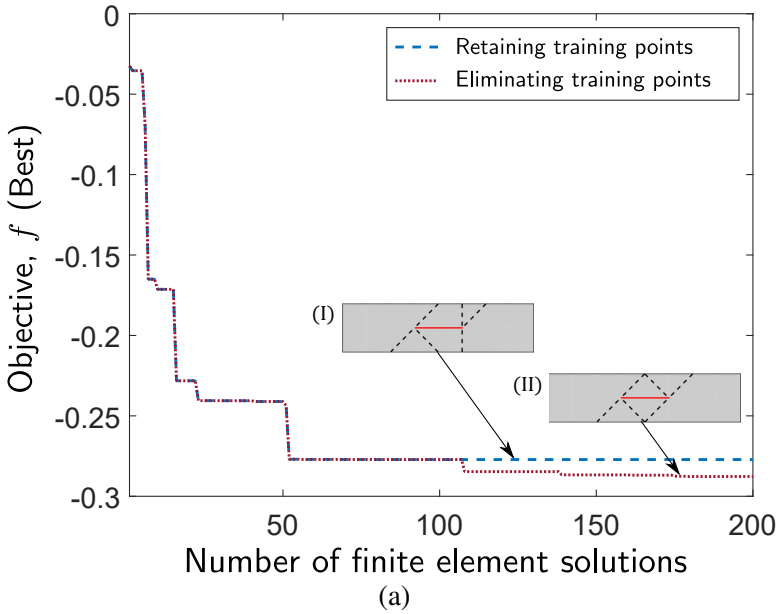


(b)

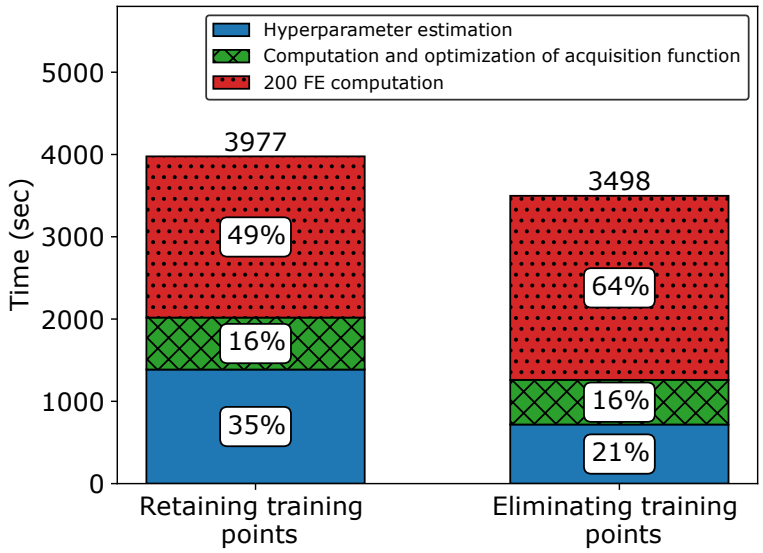
Figure 4.23: Fixed hyperparameters for the twist chomper problem with Matérn5 covariance function. (a) Evolution of the objective function and corresponding fold patterns (insets) from Bayesian optimization when estimating the hyperparameters every iteration $(p = 1)$ versus holding them fixed throughout. Red solid lines on the fold patterns indicate truss elements connecting fixed nodes. Dashed black lines indicate the active folds. (b) Computational time for the various cases.

derivative information is not necessarily beneficial in discovering good solutions for the chomper problem.

A similar study is conducted for the 38-dimensional twist chomper problem. The comparison of the evolution of Bayesian optimization with and without derivative



(a)



(b)

Figure 4.24: Limiting the size of the training set for the twist chomper problem. (a) Evolution of the objective function and corresponding fold patterns (insets) from Bayesian optimization when retaining all the points (dashed blue line) versus eliminating points with the worst objective function values (dotted red line). Red solid line on the fold patterns indicate truss elements connecting fixed nodes. The dashed black lines indicate the active folds. (b) Computational time for the two cases.

information for the five initial guesses is shown in Fig. 4.27. In contrast to the chomper problem, inclusion of derivative information for the twist chomper problem improves the efficiency of the overall Bayesian optimization approach. As seen in Fig. 4.27, the case of BO with derivative information outperforms BO without derivatives in all five scenarios. The

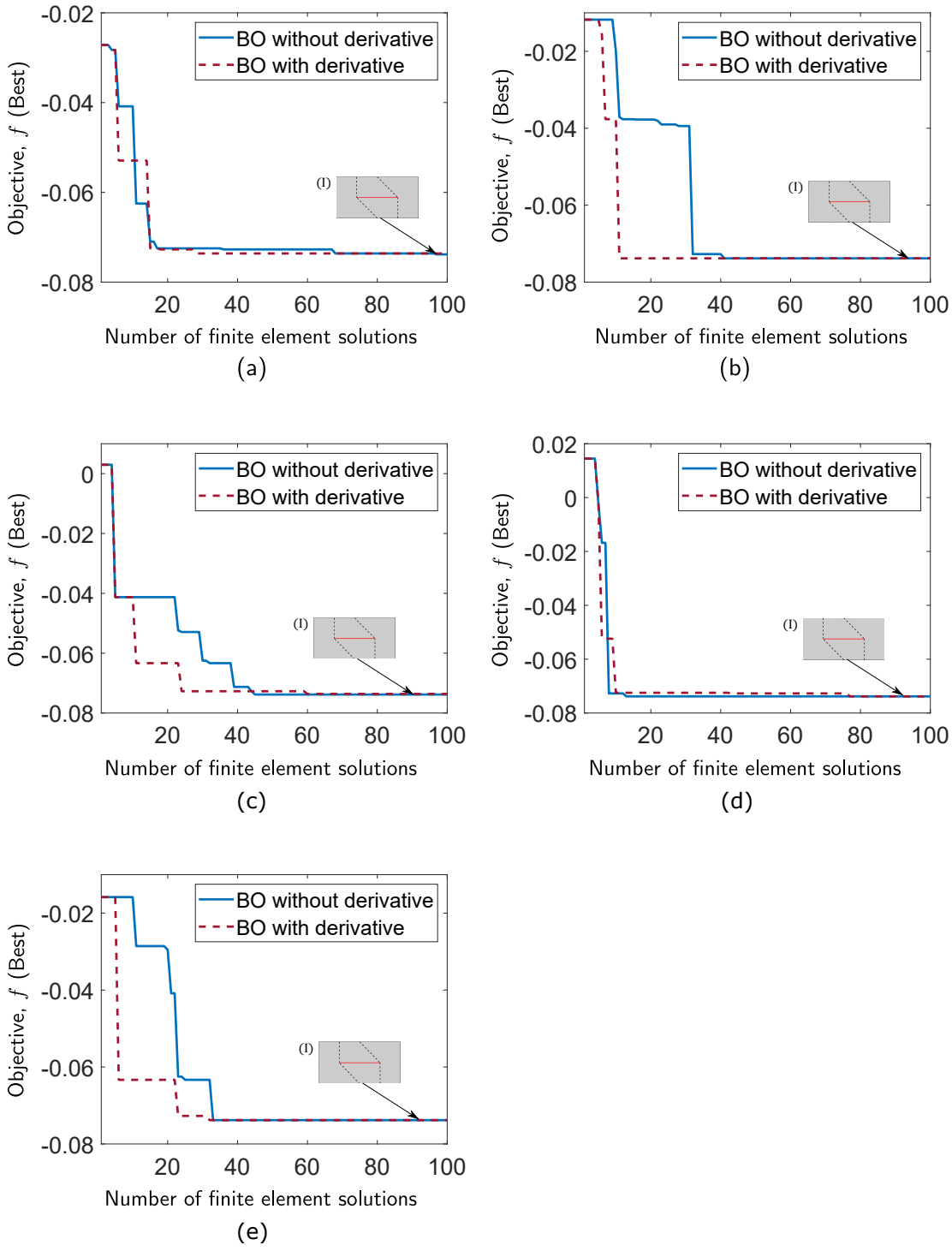


Figure 4.25: Comparison of Bayesian optimization with and without use of derivative information using squared exponential covariance function: Chomper problem. (a) Initial guess 1, (b) Initial guess 2, (c) Initial guess 3, (d) Initial guess 4, and (e) Initial guess 5. The solid blue line indicates the evolution of BO without derivative information and dashed red line indicates the evolution of BO with derivative information along with the corresponding fold patterns (insets).

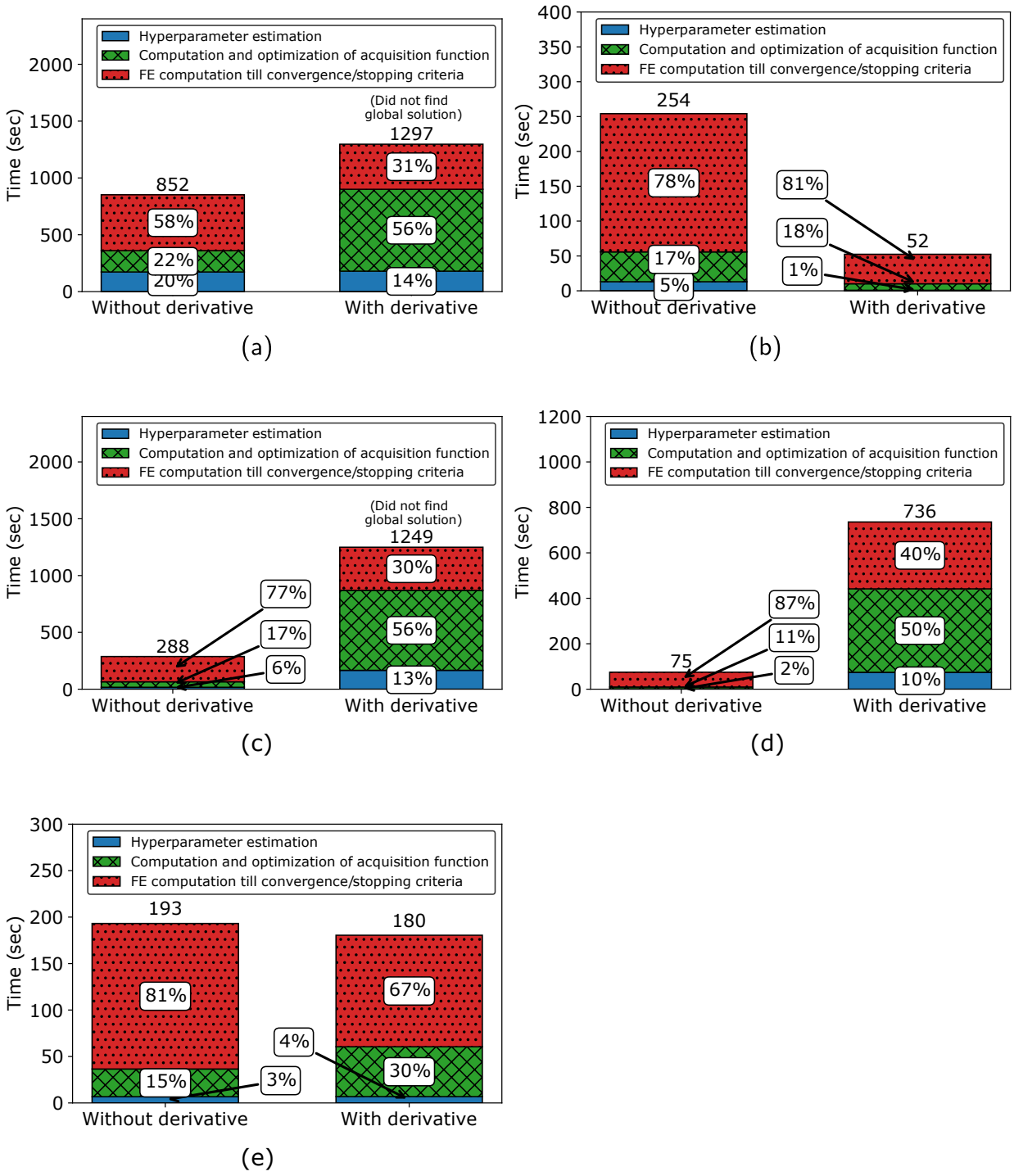


Figure 4.26: Comparison of computational time for Bayesian optimization with and without use of derivative information using squared exponential covariance function: Chomper problem. (a) Initial guess 1, (b) Initial guess 2, (c) Initial guess 3, (d) Initial guess 4, and (e) Initial guess 5.

best possible fold pattern (Design I) with $f(\mathbf{x}) = -0.289$ is discovered with all five initial training sets when derivative information is utilized. The best design is found with 16, 85, 17,

34 and 8 finite element solutions for the five different initial training sets when the derivative information is used. In contrast, Bayesian optimization without the use of derivatives does not find the optimum design within 100 finite element solution except for initial training set 1 (see Fig. 4.27). Even in the case of training set 1, Fig. 4.27(a) shows that BO without derivative information takes 85 finite element solutions which is 69 more expensive function solves than BO with derivative case for same initial training set.

Figure 4.28 compares the overall computational time with and without derivative information for the twist chomper problem. Again, the computational time is measured till the optimum point is discovered or the maximum number of training points (100) is reached. In contrast to the chomper problem, the derivative information helps not only in discovering the best fold pattern but also in converging faster for the twist chomper problem. Also, for the twist chomper problem, when derivative information is utilized, all five initial guesses lead to the best solution whereas its counterpart without derivative information only finds the best fold pattern with initial guess 1 (Fig. 4.28(a)). Moreover because the derivative information helps in faster convergence, the overall time to find global optimum is reduced by 87% for initial guess 1, 90% for initial guess 3, 71% for initial guess 4, and 96% for initial guess 5. But even for the case with initial guess 2, BO without derivative information does not find the best design whereas BO with derivative enrichment is able to discover it. The time taken to compute and optimize the acquisition function greatly increases, as seen in Fig. 4.28(b), because the covariance matrix size increases when the derivative information is included. Thus the computation and acquisition function (green hashed block in the stacked bar plot) for BO with derivative as seen in Fig. 4.28(b) takes almost 74% of the overall time.

To summarize, the inclusion of derivative information in the BO framework does not help the chomper problem but is hugely beneficial for the twist chomper problem. This indicates that the response objective function for the twist chomper problem has a dominant direction in its design space which greatly affects the objective function value. Thus, for problems with a dominant/active direction in the design space, the derivative information is hugely advantageous in finding those directions and helps the BO algorithm converge faster.

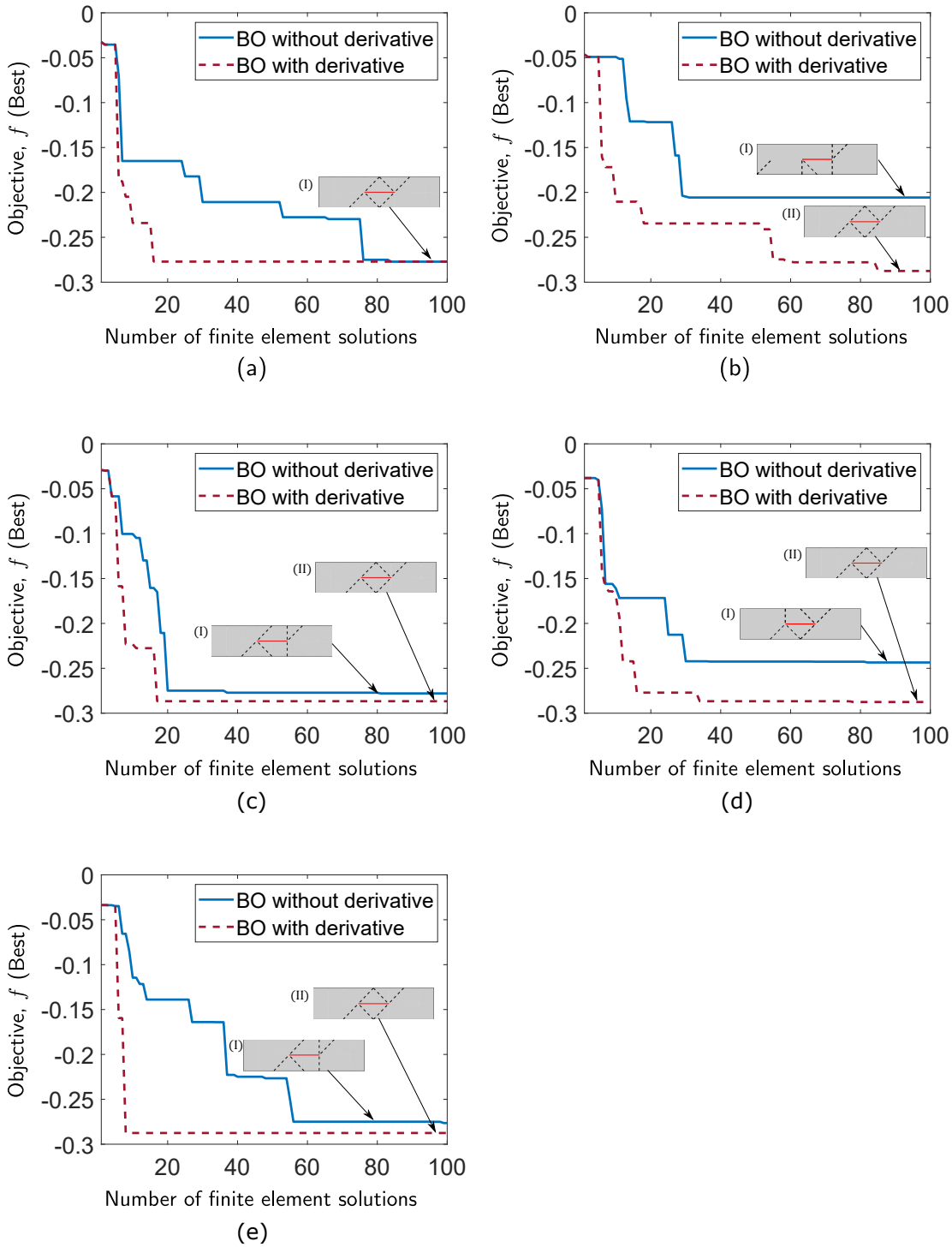


Figure 4.27: Comparison of Bayesian optimization with and without use of derivative information using squared exponential covariance function: Twist chomper problem. (a) Initial guess 1, (b) Initial guess 2, (c) Initial guess 3, (d) Initial guess 4, and (e) Initial guess 5. The solid blue line indicates the evolution of BO without derivative information and dashed red line indicates the evolution of BO with derivative information along with the corresponding fold patterns (insets).

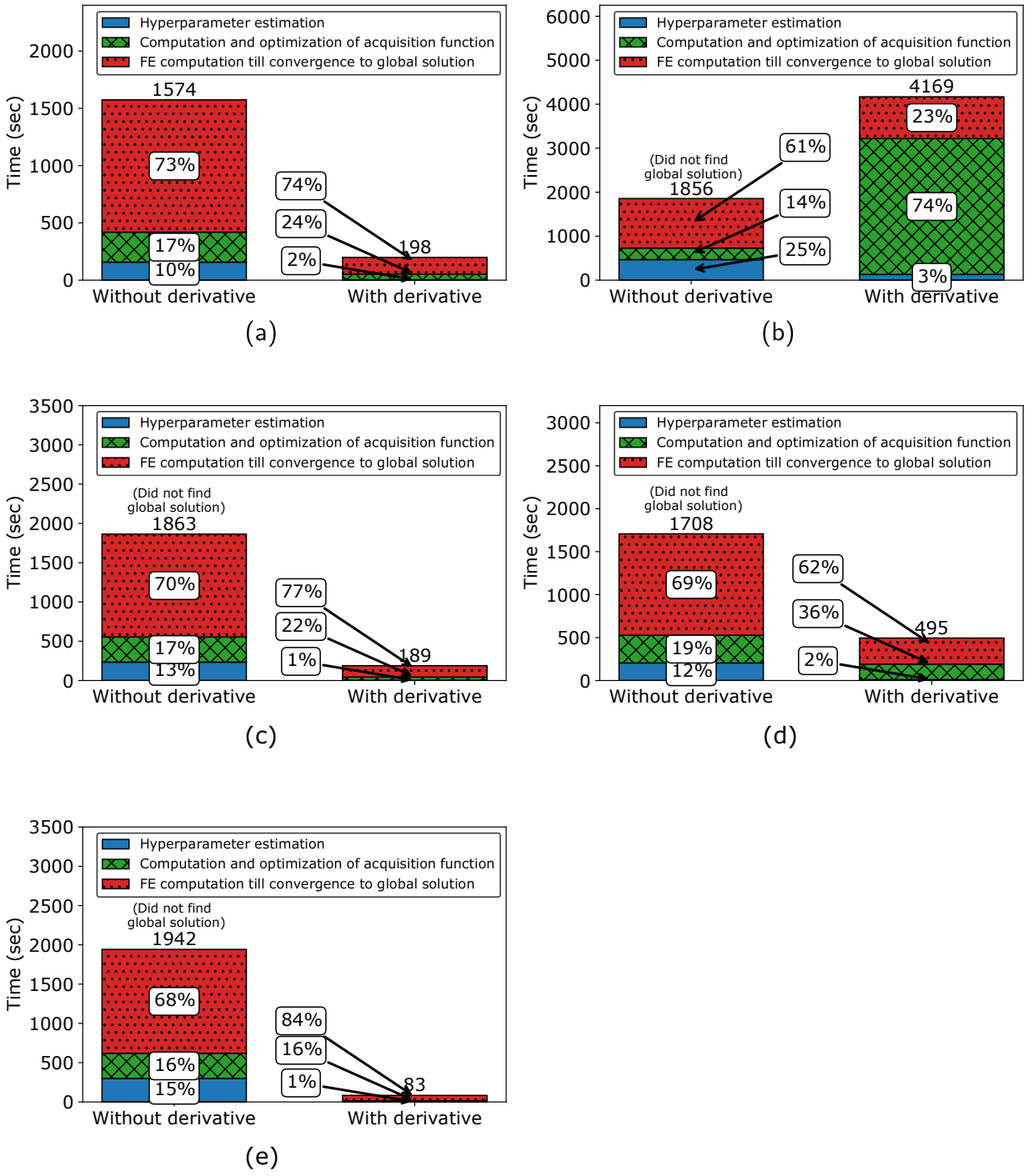


Figure 4.28: Comparison of computational time for Bayesian optimization with and without use of derivative information using squared exponential covariance function: Twist chomper problem. (a) Initial guess 1, (b) Initial guess 2, (c) Initial guess 3, (d) Initial guess 4, and (e) Initial guess 5.

4.6 Bayesian Optimization with Anisotropy

In section 3.6, the mathematics involved in utilizing anisotropic models of BO is described. In anisotropic models, all the design variables are differentiated as different length scale parameters are assigned to each of the design variables. These length scale parameters are estimated by maximizing the log-likelihood function as done in the traditional BO framework. However, as the number of length scales increases to match the number of design variables in the anisotropic BO model, the estimation of hyperparameters now becomes a $(d + 1)$-dimensional optimization problem. This leads to an increase in the overall computational time. However with anisotropy in the BO framework, it may be possible to find a better solution with fewer expensive function solutions, thus saving overall computational time.

Figure 4.29 compares the progress of isotropic BO to anisotropic BO for the chomper problem. Both isotropic and anisotropic models find the best fold pattern with the various initial guesses using similar number of FE solutions. Thus, it can be inferred that introducing anisotropy does not benefit the convergence for chomper problem. As discussed before, the time spent in estimating hyperparameters is expected to increase greatly. Figure 4.30 compares the overall time till convergence or stopping criterion is met for both isotropic and anisotropic models for the chomper problem. It is clear that the anisotropic case takes much more time compared to isotropic BO. As seen in Fig. 4.30, for the anisotropic cases, the estimation of the hyperparameters (solid blue block in the stacked bar plot) at every iteration takes almost 70% to 80% of the overall time. In other words, in some cases (Fig.s 4.30(b), 4.30(c) and 4.30(d)), anisotropic models are more than 10 times as expensive as isotropic models. This clearly indicates that anisotropic BO is not an efficient way to find optimal fold pattern for the chomper problem.

Similar studies are done on the 38-dimensional twist chomper problem. The evolutions of isotropic BO and anisotropic BO are compared in Fig. 4.31. The anisotropic BO model discovers the best solution for all the five different initial guesses whereas the isotropic model is able to find the best solution only with initial guess 1 Fig. 4.31(a). The overall computational time is shown in Fig. 4.32. Similar to the chomper problem case, the estimation of hyperparameters takes the most amount of overall time for the anisotropic models. The tuning of hyperparameters at every iterations takes almost 80% to 90% of the

overall time when the number of training points is more than 150. Thus, even though anisotropic BO discovers better fold patterns than the isotropic model, it takes significantly more time to discover them except for the case with initial guess 5, where it takes just 30 iterations to find the best possible fold pattern. Similar to the chomper problem, for some cases (Figs. 4.32(b), 4.32(c) and 4.32(d)), anisotropic models are more than 10 times as expensive as isotropic models. Overall, anisotropic BO models are able to find better solutions when applied to the twist chomper problem but are computationally expensive, making it an inefficient approach.

4.7 Anisotropic Bayesian Optimization with Automatic Relevance Determination

As discussed in Section 3.6, anisotropic models can be used to determine the irrelevant design variables by implementing automatic relevance determination (ARD). The design variables associated with a high value of length scale have minimal effect on the objective function value, making them irrelevant. In Section 4.6, hyperparameters are estimated every iteration by maximizing the log-likelihood function. For these previous studies (Figs. 4.29, 4.30, 4.31 and 4.32), the average length scale parameters corresponding to every design variable over the complete BO evolution (100 iterations for the chomper and 200 iterations for the twist chomper) can be calculated.

Figure 4.33 shows the average length scale values for the 18 design variables of the chomper problem. The maximum Euclidean distance for the chomper problem $(\sqrt{18})$, is considered to be the threshold value and is marked with a horizontal dashed line in Fig. 4.33. All the design variables with length scale parameter value greater than the threshold are considered irrelevant and are shown with red hashed bar plots, whereas the rest are plotted with solid blue bars. As shown in Fig. 4.33, for the chomper problem, four design variables labeled $1, 13, 17$ and 18 are irrelevant as their average length scale values are greater than the threshold. The remaining 14 design variables are the ones that affect the objective function and are therefore considered relevant design variables. The relevant and irrelevant design variables are shown for the chomper problem in Fig. 4.34. The solid red lines indicate the four irrelevant design variables and the remaining relevant variables are shown with dashed

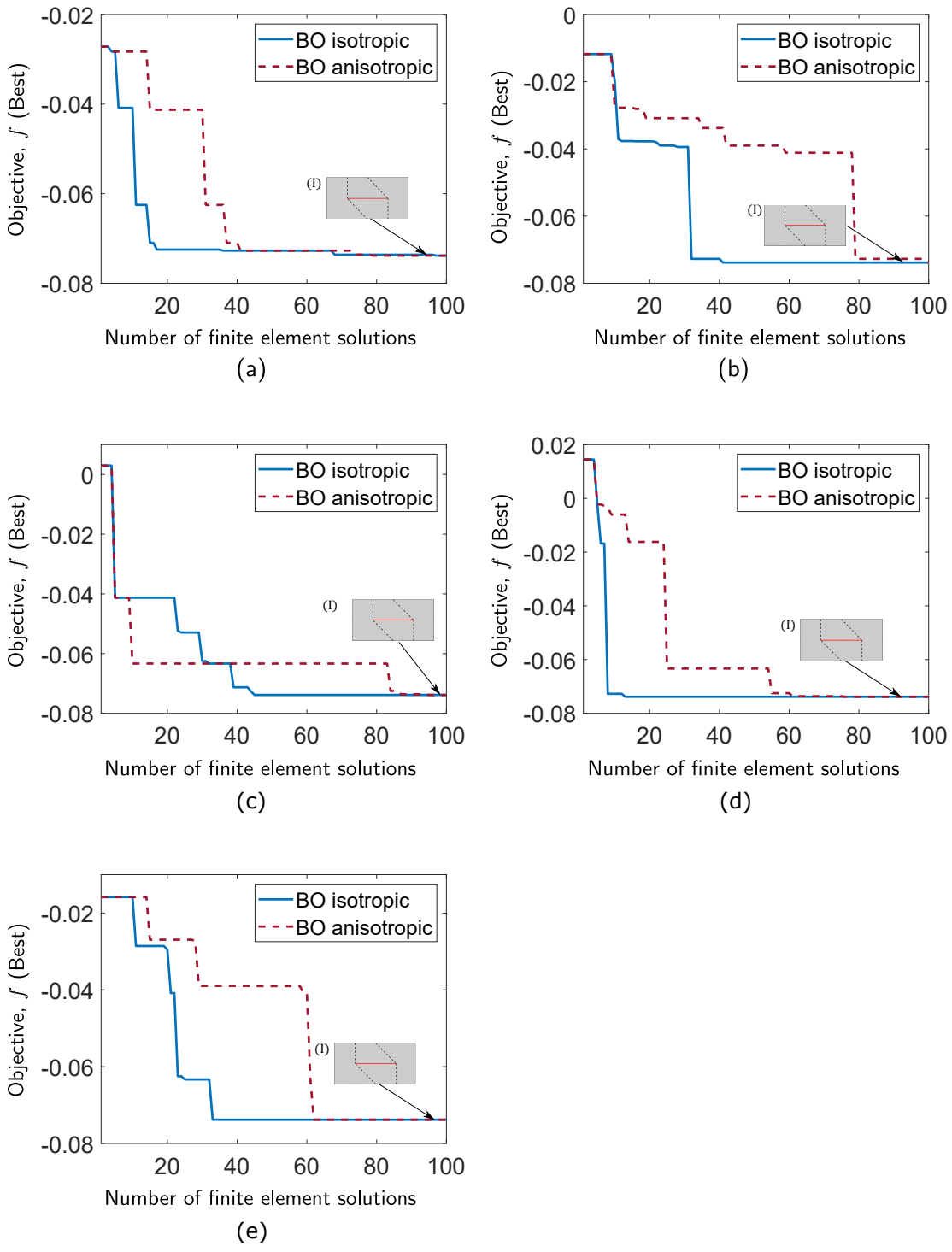


Figure 4.29: Comparison between isotropic and anisotropic models in Bayesian optimization with squared exponential covariance function: Chomper problem. (a) Initial guess 1, (b) Initial guess 2, (c) Initial guess 3, (d) Initial guess 4, and (e) Initial guess 5. The solid blue line indicates the evolution of BO without derivative information and dashed red line indicates the evolution of BO with derivative information along with the corresponding fold patterns (insets).

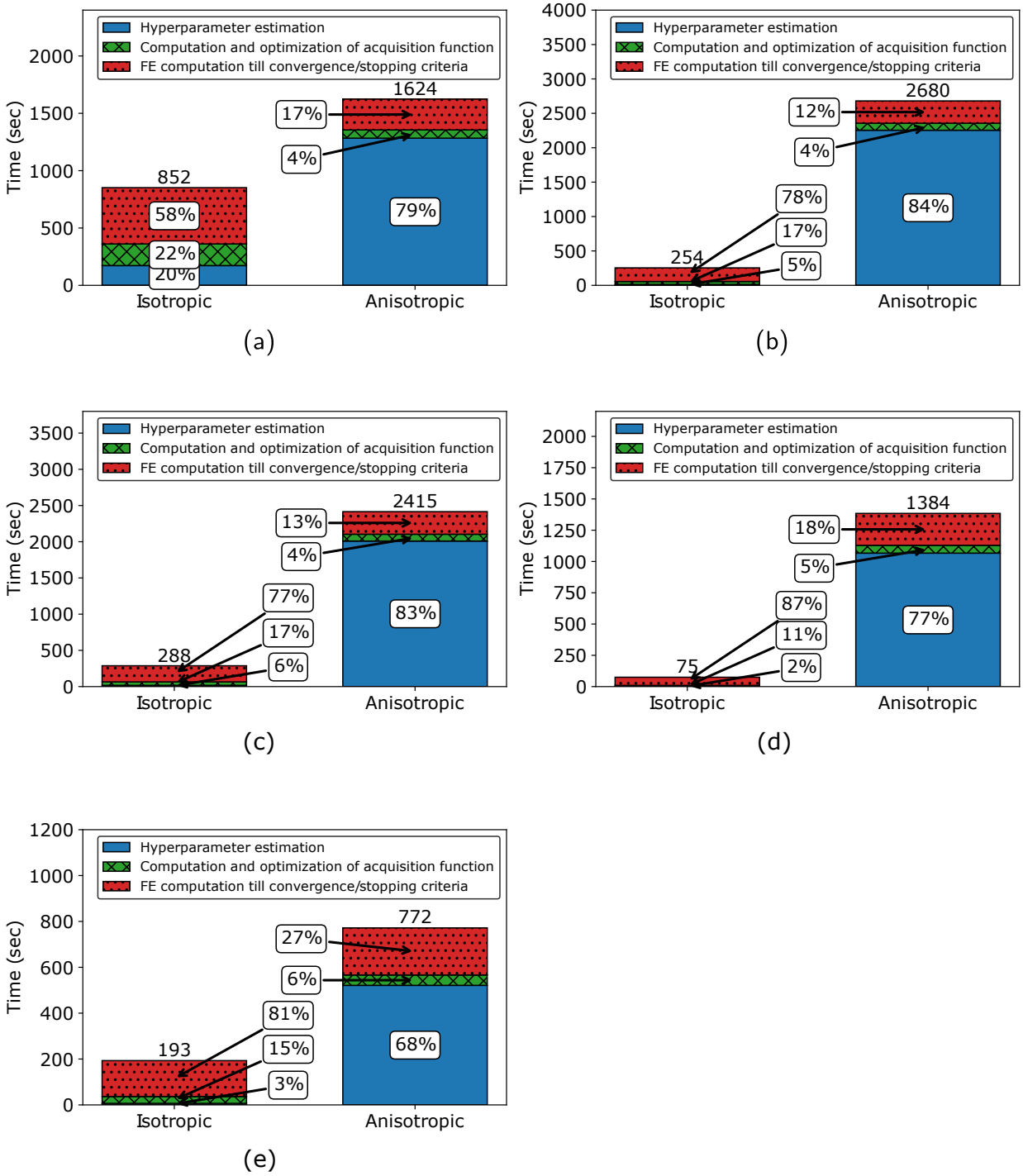


Figure 4.30: Comparison of computational time between isotropic and anisotropic models in Bayesian optimization with squared exponential covariance function: Chomper problem. (a) Initial guess 1, (b) Initial guess 2, (c) Initial guess 3, (d) Initial guess 4, and (e) Initial guess 5.

blue lines. It is interesting to note that the irrelevant design variables or truss elements discovered by ARD technique are all attached to the fixed nodes where two of those trusses connect the fix nodes while the remaining two trusses connect a fixed node to a node where

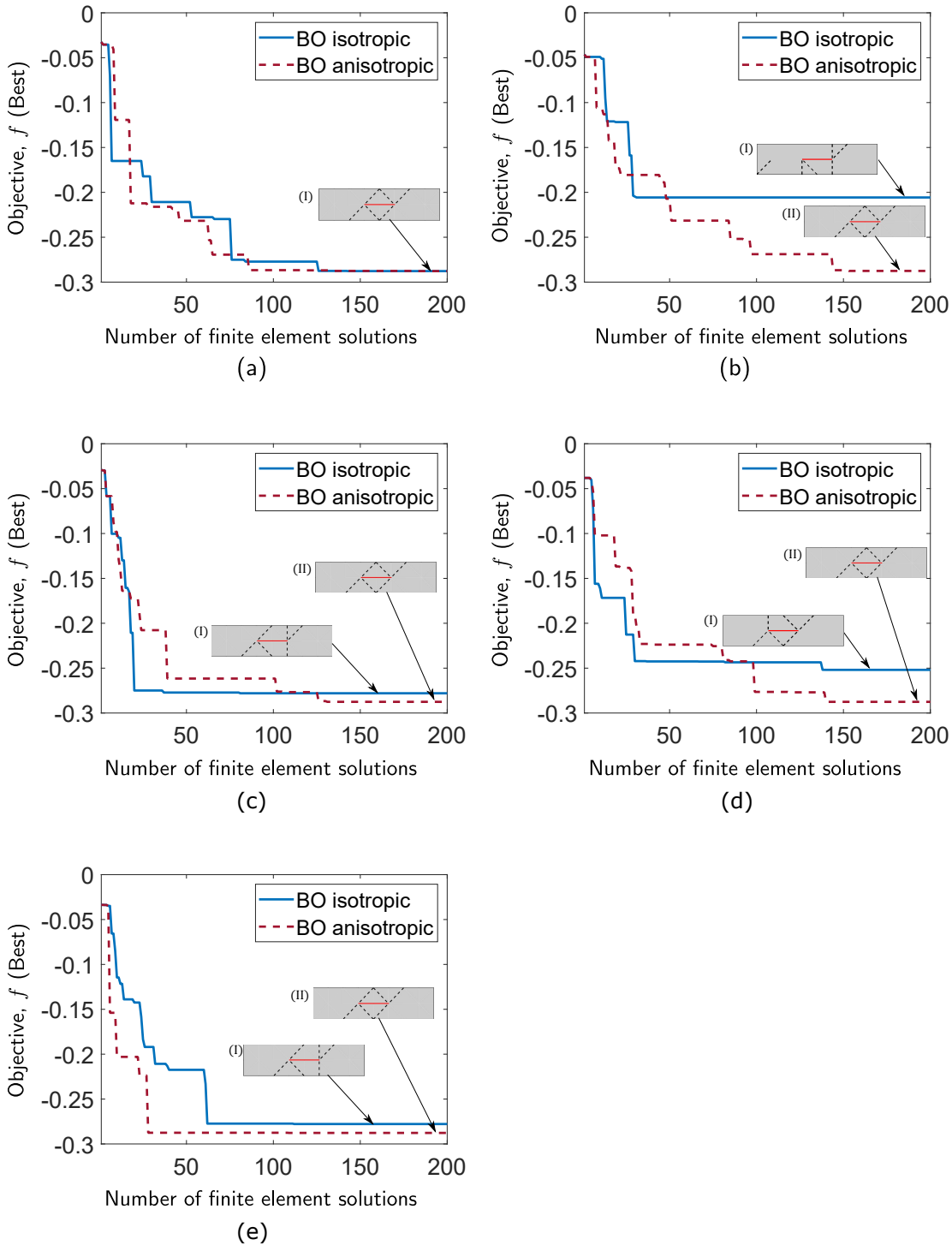


Figure 4.31: Comparison between isotropic and anisotropic models in Bayesian optimization with squared exponential covariance function: Twist chomper problem. (a) Initial guess 1, (b) Initial guess 2, (c) Initial guess 3, (d) Initial guess 4, and (e) Initial guess 5. The solid blue line indicates the evolution of BO without derivative information and dashed red line indicates the evolution of BO with derivative information along with the corresponding fold patterns (insets).

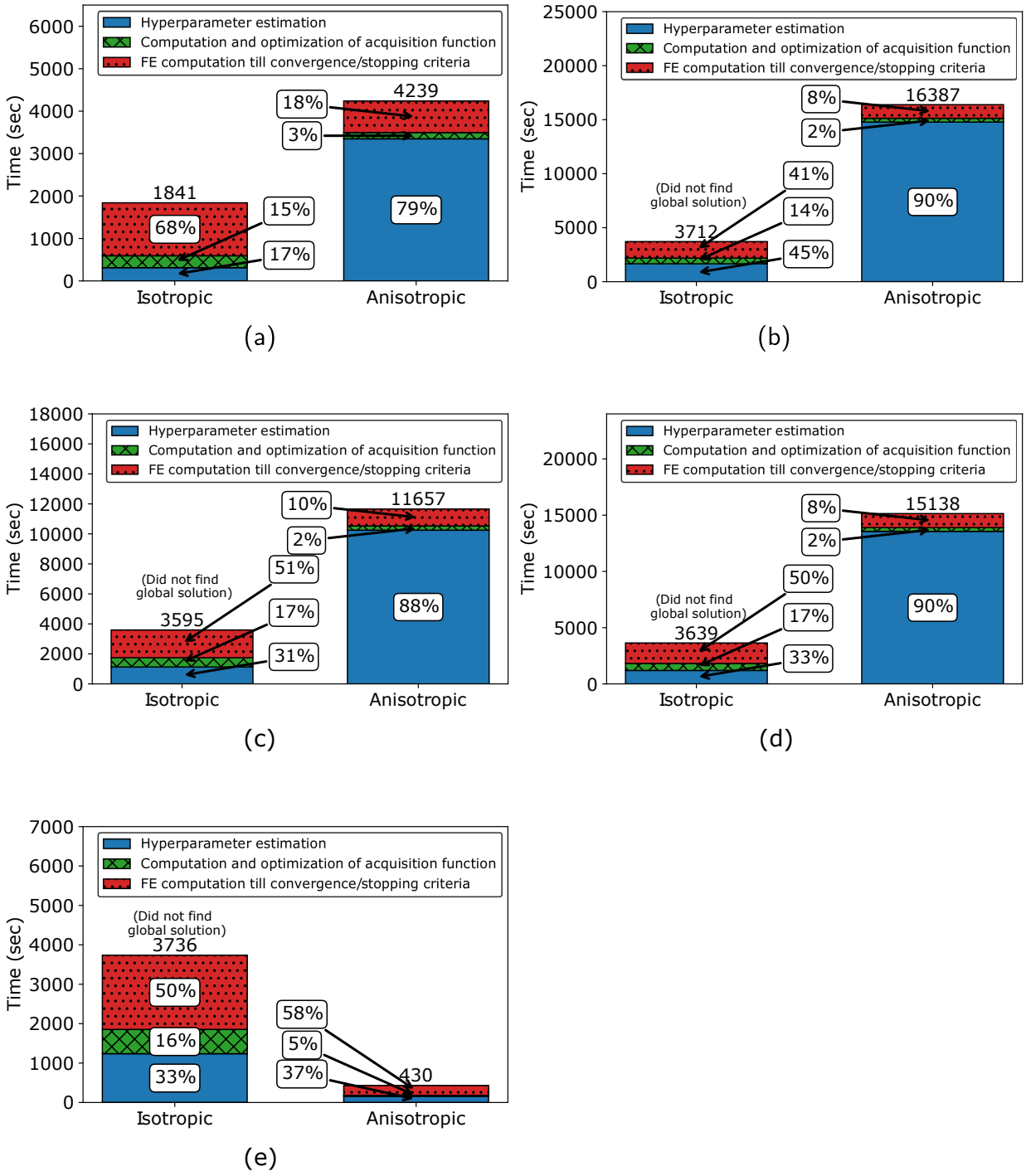


Figure 4.32: Comparison of computational time between isotropic and anisotropic models in Bayesian optimization with squared exponential covariance function: Twist chomper problem. (a) Initial guess 1, (b) Initial guess 2, (c) Initial guess 3, (d) Initial guess 4, and (e) Initial guess 5.

the input displacement is applied. In other words, these truss elements, whether turned on or off, have a minimal effect on the objective function value. These four design variable values

thus can be kept fixed and not be included in the optimization framework. The BO evolution of such a setup with reduced 14 dimensions for the chomper problem is shown in Fig. 4.35(b) using five different initial guesses and is compared with the original problem plotted in Fig. 4.35(a). The BO considered here is isotropic with squared exponential covariance function, with the LCB acquisition function, and with the estimation of hyperparameters done every iteration. The BO with reduced dimensions finds the best fold pattern, which is the same as that discovered with the original 18-dimensional BO for the chomper problem. The overall computational time is compared in Fig. 4.36. As the problem dimension is reduced from 18 to 14, the time taken to optimize acquisition function is reduced as the design space is smaller. Note that even with the reduced problem size, the time taken to estimate isotropic hyperparameters and compute the acquisition function is the same since the size of covariance matrix is only a function of the number of training points and not the problem dimension. Surprisingly, the reduced dimension problem requires more FE solutions to find the best solution compared to original problem. Also, as the reduction in dimension from 18 to 14 is not huge, the advantage of eliminating the irrelevant dimensions for the chomper problem is limited.

A similar study is carried out for the 38-dimensional twist chomper problem where ARD is found to be much more beneficial. The average lengths scale values estimated with different initial guesses for the 38 design variables of the twist chomper problem are shown in Fig. 4.37. This average is calculated over 200 iterations. The threshold value for the lengths scale parameter is considered to be the maximum Euclidean distance $(\sqrt{38})$ for the twist chomper problem. It is represented by the dashed horizontal line in Fig. 4.37, which clearly shows that a majority of the length scale parameters exceed the threshold value and signify irrelevant design variables for the objective function. Figures 4.37(c), 4.37(d) and 4.37(e) indicate that only 12 out of the 38 design variables are relevant (shown by solid blue bars). The relevant design variables from these sub-panels are labeled 10, 11, 12, 14, 15, 16, 18, 19, 20, 22, 23 and 24. Figure 4.37(a) shows that with initial guess 1 the design variable 19 and 23 are additionally irrelevant reducing the number of relevant design variables to ten. However for the case with initial guess 2 (Fig. 4.37(b)) along with the 12 relevant design variables there is an additional design variable labeled 26 which has its average length scale parameter to be less than the threshold and therefore can be considered relevant. For further analysis we consider 12 design variables as seen in Fig. 4.37(c), 4.37(d)

and 4.37(e) to be relevant and the rest as irrelevant design variables. The relevant and irrelevant truss elements are shown in Fig. 4.38, where the red solid lines indicates irrelevant variables and the dashed blue lines indicate the 12 relevant design variables. Even in the twist chomper case the truss elements connecting the fixed nodes with each other and connecting the fixed nodes to the input nodes are irrelevant. Interestingly, for the twist chomper, most of the truss elements away from the fixed nodes are also irrelevant according to ARD. Also, interestingly, the best possible fold pattern forms from a subset of the relevant 12 design variables, which indicates that the reduced 12 variable problem could discover the best solution. The reduction in the dimension of the optimization problem from 38 to 12 is a huge advantage. The optimization problem now can be posed using these 12 design variables, keeping the rest of the truss elements fixed as inactive fold lines. As the number of active design variables decreases to almost a third of the original problem, the overall time to find the optimum is expected to be reduced. Figure 4.39 shows the comparison of BO progress with all 38 design variables and with the 12 relevant design variables determined from ARD. The isotropic squared exponential covariance function is used for modeling the GP surrogate and the lower confidence bound is used as the acquisition function to find subsequent sampling points. The hyperparameters are estimated at every iteration for both these cases. As seen in Fig. 4.39(b), all the cases with various initial guesses discover the best fold pattern with fewer FE solutions (less than 100) except initial guess 5, which takes 185 FE solutions. For the original 38-dimensional problem, only the case with initial guess 1 discovers the best possible solution as seen in Fig. 4.39(a). Figure 4.40 compares the overall computational time for the cases with five different initial guesses with all 38 design variables and with the reduced problem. As the dimension of the optimization problem is reduced, the time spent in optimizing acquisition function decreases as seen in Fig. 4.40(b). Moreover, we see a huge improvement in the overall time for discovering best fold pattern when compared to the original BO without dimensional reduction. Thus, ARD clearly boosts the efficiency of BO for the twist chomper problem.

These studies indicate that anisotropic surrogate models are more beneficial when applied to problems that have some dominant design variables that greatly affect the objective function. In such scenarios ARD can be implemented to reduce the overall dimension of the problem and thus improve the efficiency of the overall BO algorithm.

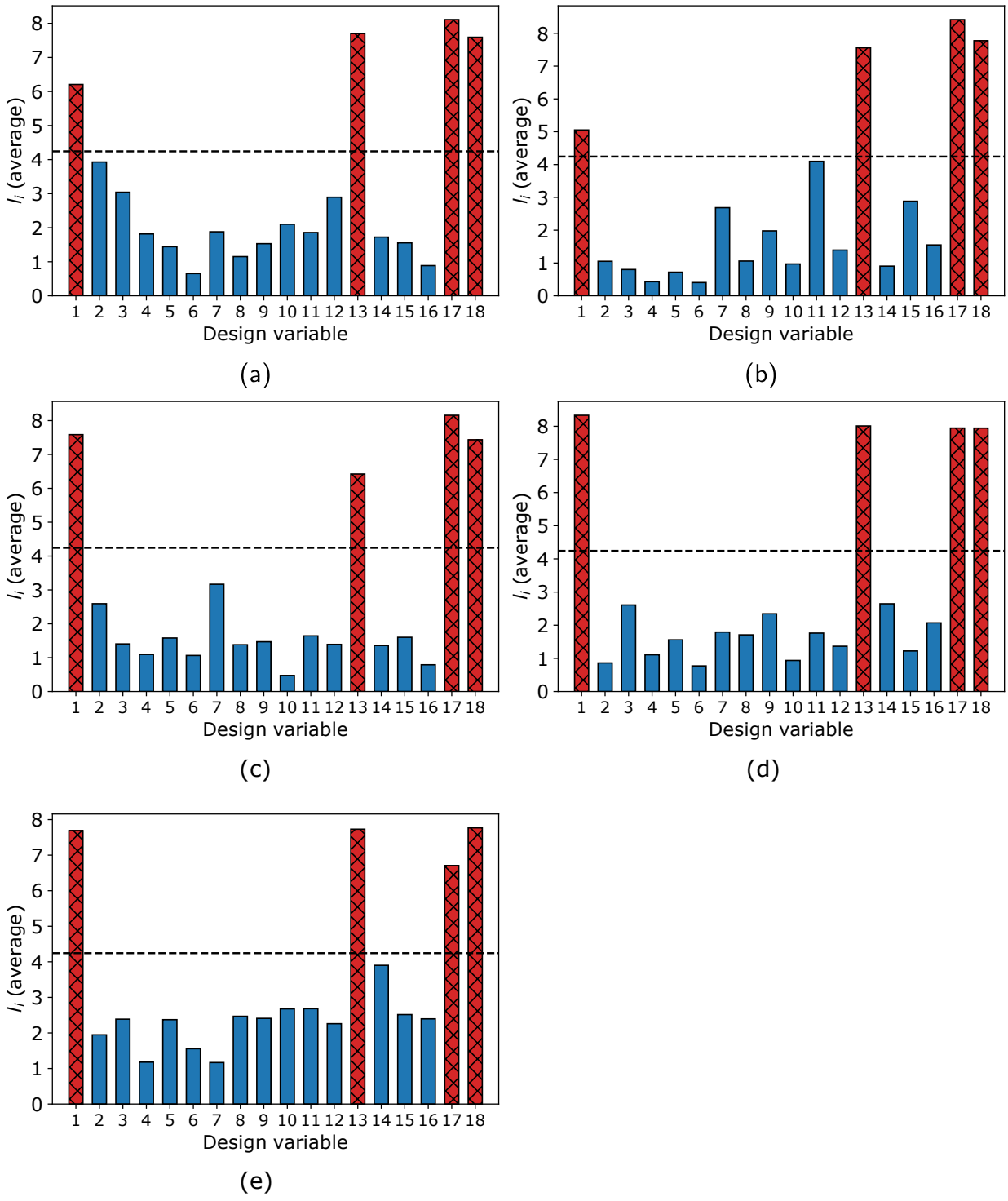


Figure 4.33: Average values of length scale hyperparameters of anisotropic model in Bayesian optimization with squared exponential covariance function: Chomper problem. (a) Initial guess 1, (b) Initial guess 2, (c) Initial guess 3, (d) Initial guess 4 and (e) Initial guess 5. The dashed horizontal line indicates the threshold length scale value. The design variables with length scale value greater than the threshold are irrelevant and are shown with red hash bars. The design variables with length scale value less than the threshold are relevant and are shown with solid blue bars.

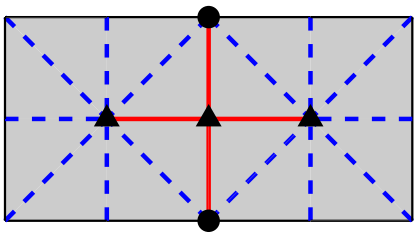
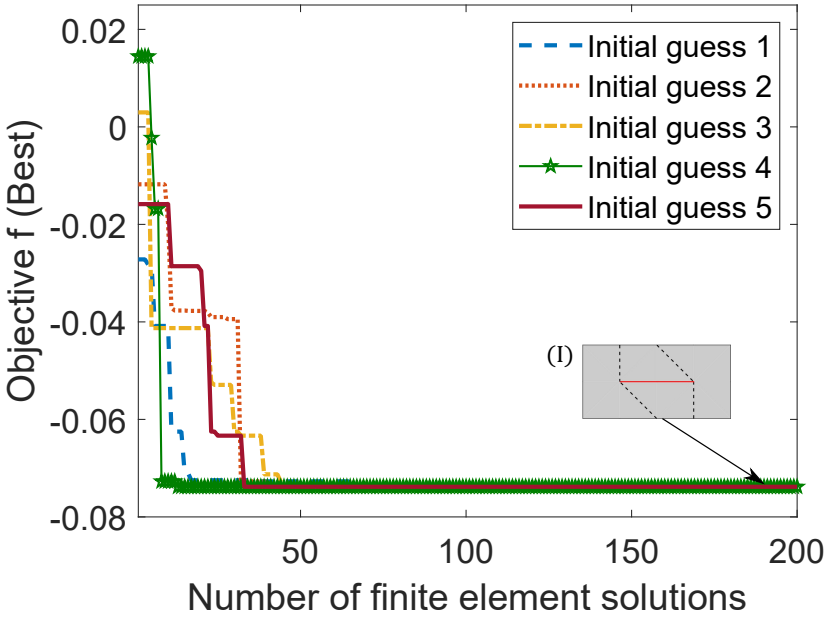
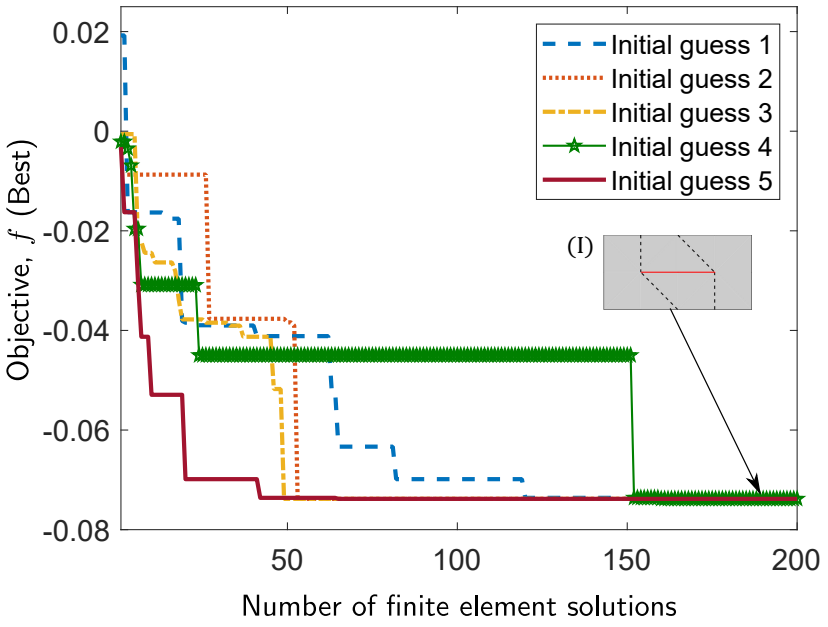


Figure 4.34: Design variables of the chomper problem. The solid red lines indicate irrelevant design variables. The dashed blue lines indicates reduced relevant design variables. The black triangular markers denotes the fixed nodes and the black circular markers denote the input nodes.

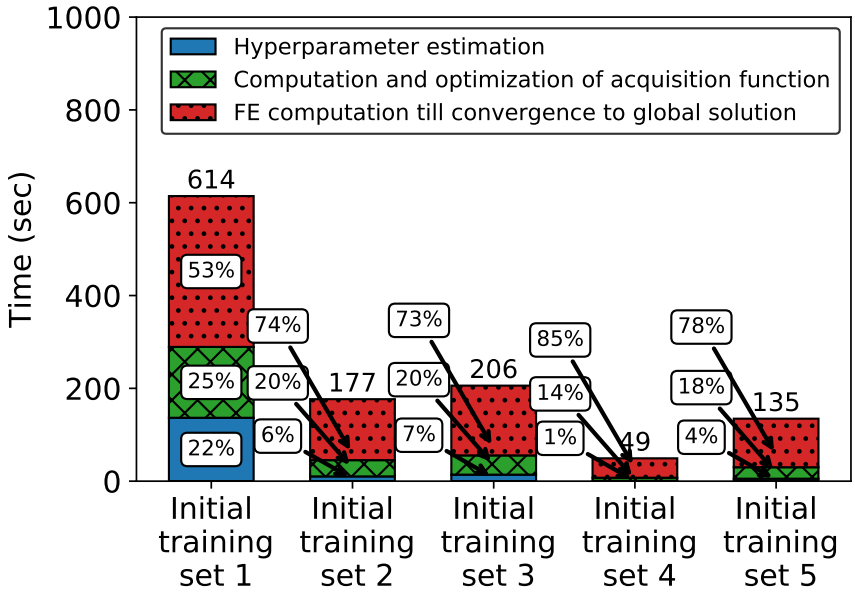


(a)

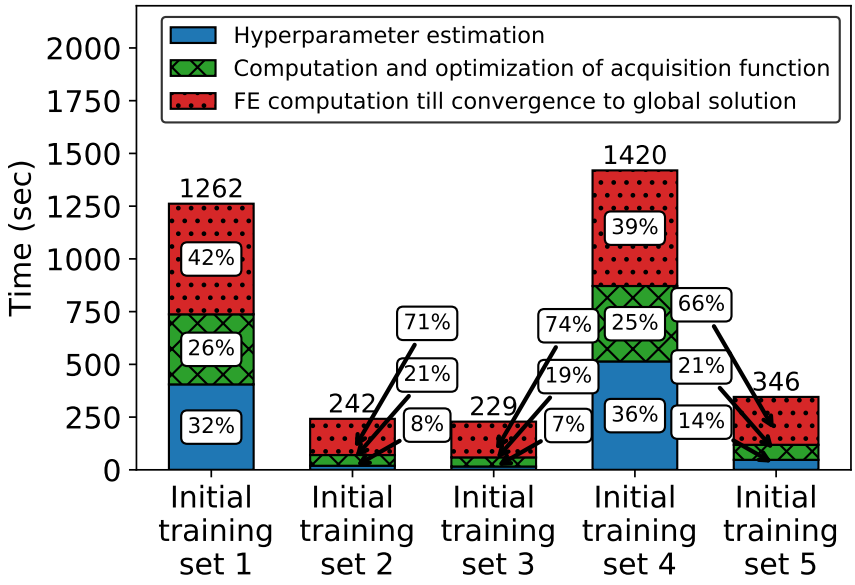


(b)

Figure 4.35: Evolution of Bayesian optimization with squared exponential covariance function for the chomper problem (a) with all 18 design variables, and (b) with reduced 14 dimensions found by ARD for five different initial training sets.



(a)



(b)

Figure 4.36: Computational time of Bayesian optimization with squared exponential covariance function for the chomper problem (a) with all 18 design variables, and (b) with reduced 14 dimensions found by ARD for five different initial training sets.

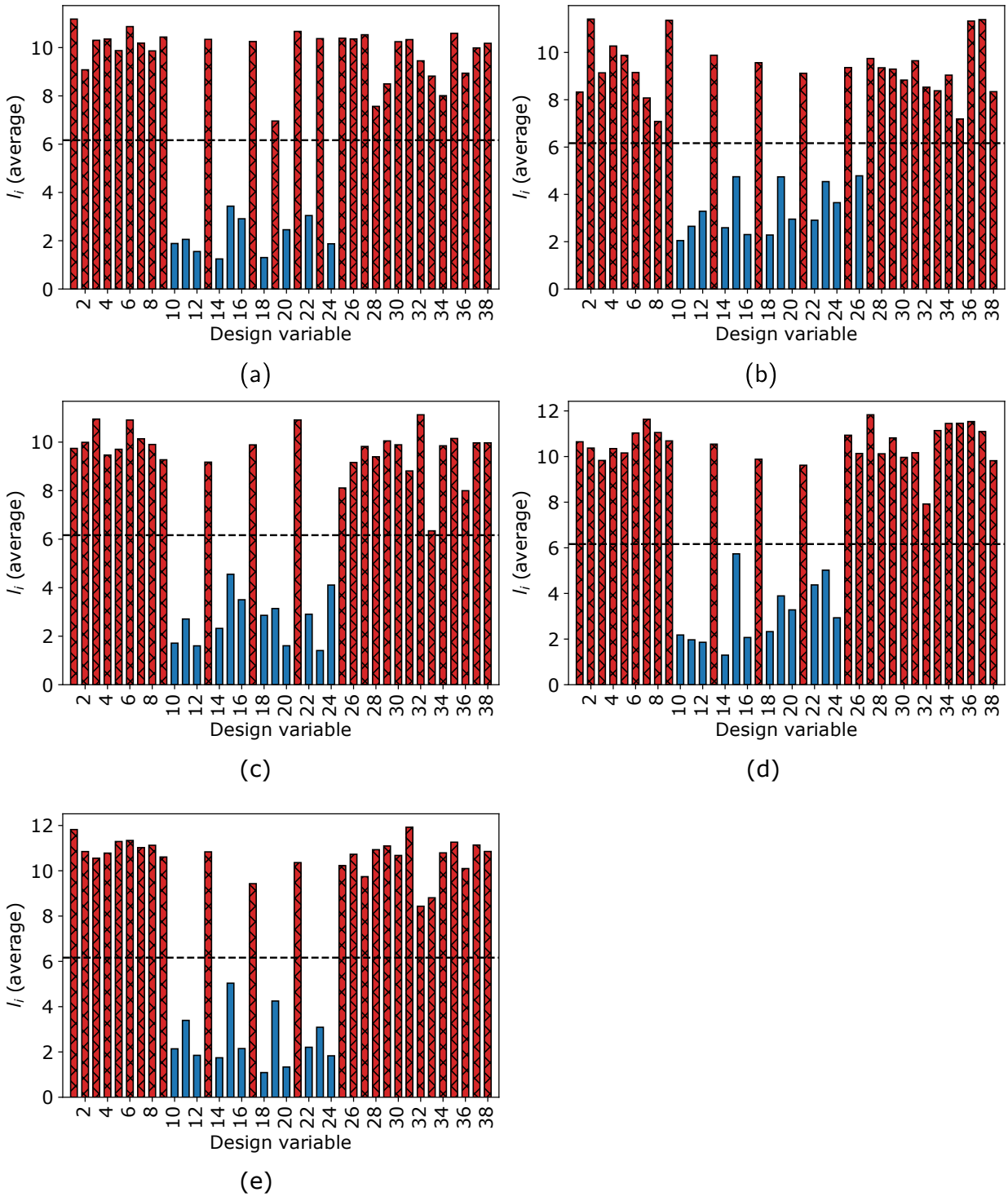


Figure 4.37: Average values of length scale hyperparameters of anisotropic model in Bayesian optimization with squared exponential covariance function: Twist Chomper problem. (a) Initial guess 1, (b) Initial guess 2, (c) Initial guess 3, (d) Initial guess 4 and (e) Initial guess 5. The dashed horizontal line indicates the threshold length scale value. The design variables with length scale value greater than the threshold are irrelevant and are shown with red hash bars. The design variables with length scale value less than the threshold are relevant and are shown with solid blue bars.

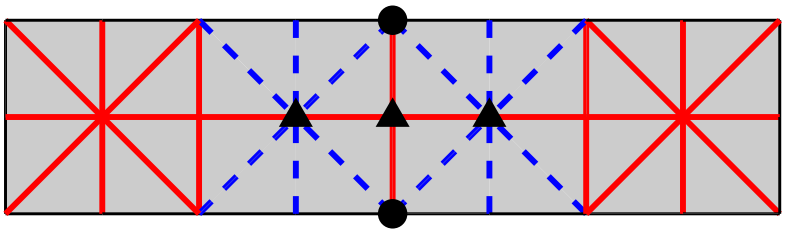
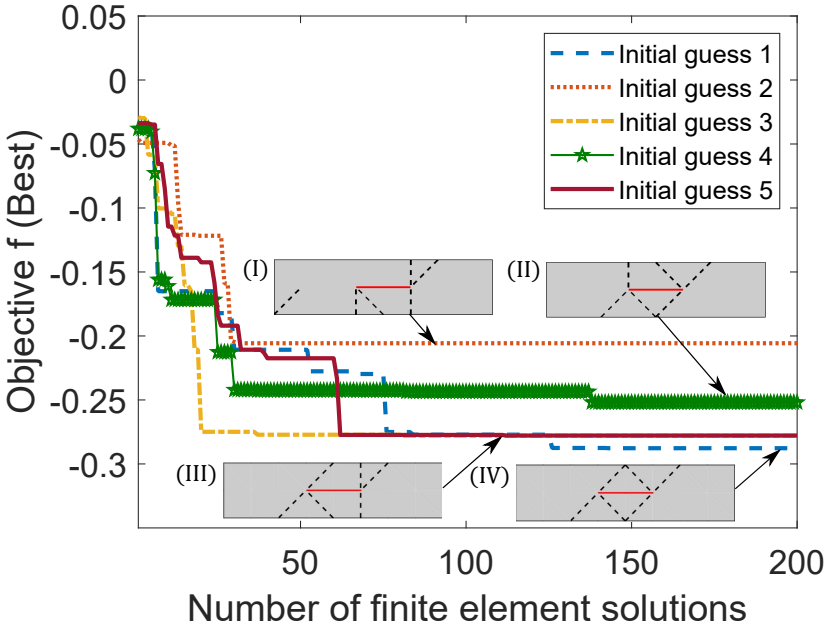
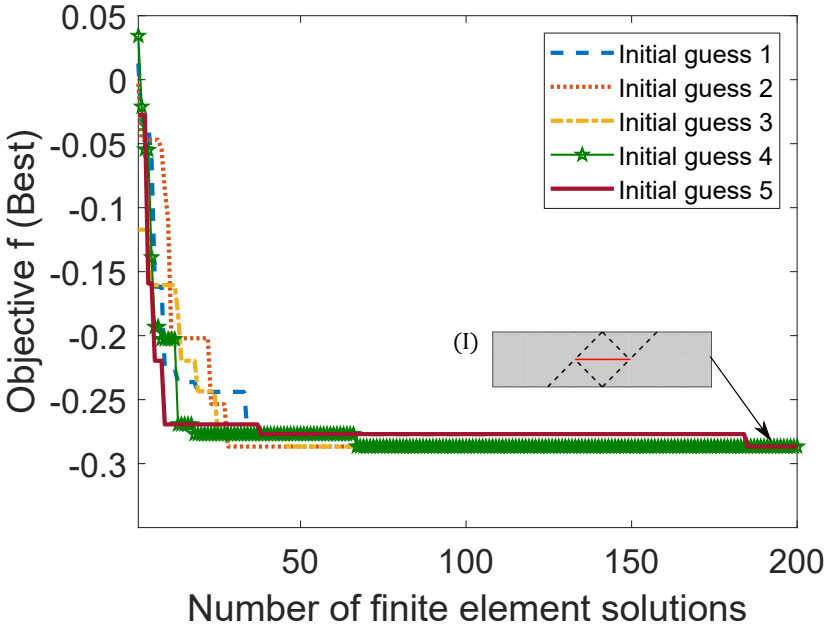


Figure 4.38: Design variables of the twist chomper problem. The solid red lines indicate irrelevant design variables. The dashed blue lines indicates reduced relevant design variables. The black triangular markers denotes the fixed nodes and the black circular markers denote the input nodes.

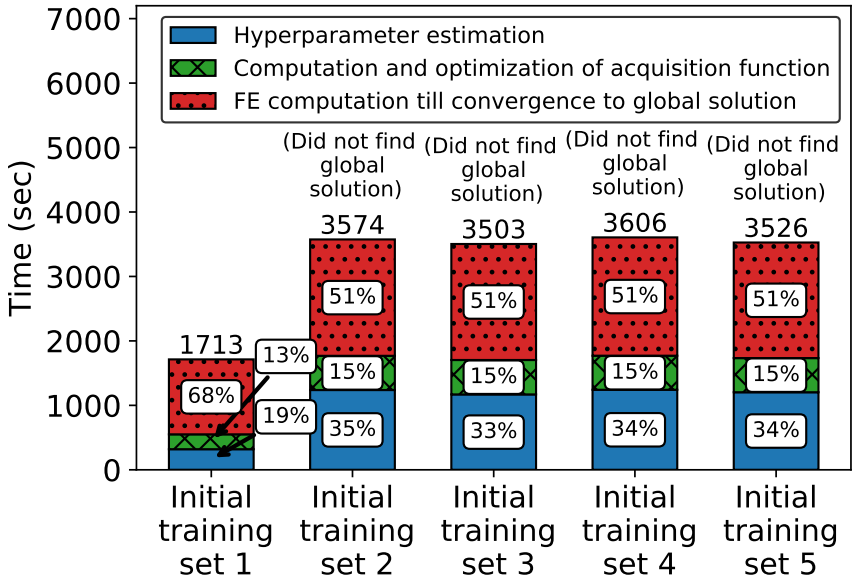


(a)

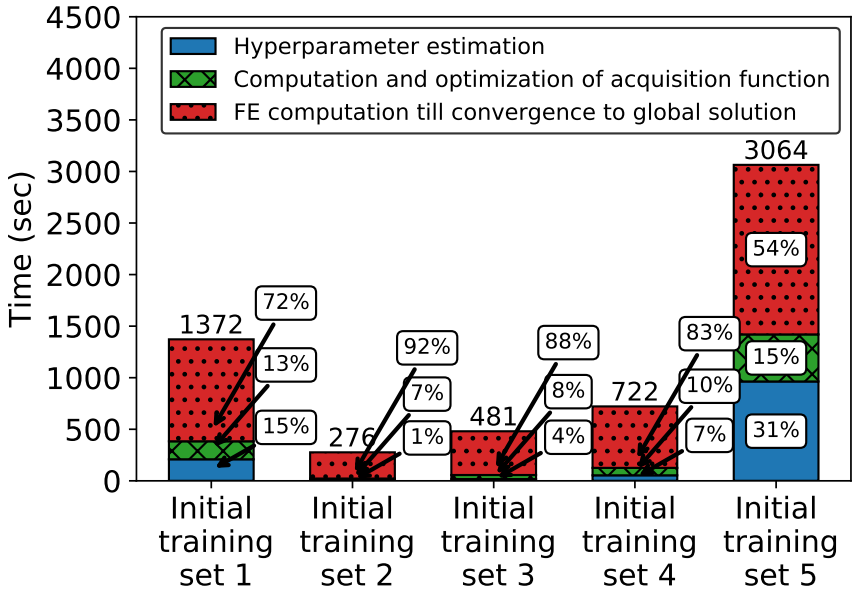


(b)

Figure 4.39: Evolution of Bayesian optimization with squared exponential covariance function for the twist chomper problem (a) with all 38 design variables, and (b) with reduced 12 dimensions found by ARD for five different initial training sets.



(a)



(b)

Figure 4.40: Computational time of Bayesian optimization with squared exponential covariance function for the twist chomper problem (a) with all 38 design variables and (b) with reduced 12 dimensions found by ARD for five different initial training sets.

Chapter 5

Conclusions and Future Work

In this work, we study the use of BO to solve the problem of discovering optimal origami-inspired nonlinear truss structures. BO is a stochastic approach to the optimization problem based on using a less expensive surrogate model to mimic the response of the more expensive function, which in our case is a FE solution for each candidate design. The surrogate model is a Gaussian process, which is specified by a covariance function. The GP is scalarized here using the lower confidence bound acquisition function. The findings of this research may be summarized as follows:

- Bayesian optimization consistently outperforms the gradient-based method, delivering previously undiscovered designs for the structure.
- In comparison to GA, which is also a global algorithm, the BO approach is able to find good designs with far fewer FE solutions for the problems studied here.
- One of the main strengths of BO is its robustness. The method is less sensitive to the initial training set than the gradient-based approach. As the size of the training set increases with the evolution of the optimization procedure, any ill-effects from the randomly chosen initial points are reduced.
- Based on the studies conducted, the Gaussian process (GP) surrogate model works well for origami optimization problems.
- For the origami problem, the choice of the covariance function does not play a significant role. The squared exponential covariance function as well as the Matérn family are able to find good solutions to the optimization problem.

- It is not necessary to tune the BO hyperparameters at every iteration. Increasing the tuning interval does not seem to affect the performance of the method and results in a considerable reduction in the overall computational time – 21% for the chomper problem and 31% for the twist chomper when tuning every 10 iterations.

- The efficiency of BO can also be improved by keeping the hyperparameters fixed, thus completely eliminating the cost of hyperparameter tuning and saving 18% to 24% and 24% to 45% of the overall time for the chomper and twist chomper problems. This is possible provided information is available about the range of values of the objective function and the length scale of the design parameter space.

- Another approach to increasing the computational efficiency of BO is to limit the size of the training set.

- When derivative information is included in the formulation of GP surrogate, it is found to be more beneficial for the twist chomper problem than the chomper problem. The derivative enriched surrogate model finds the best possible fold pattern for the twist chomper within 100 FE solutions regardless of the initial training set.

- The anisotropic GP surrogate also helps the twist chomper problem more in converging faster to the global solution in comparison to the chomper problem. Although the anisotropic BO discovers the best fold pattern for the twist chomper problem for all the initial guesses considered, it requires 60% more computational budget for 200 FE solutions.

- The ARD implementation finds four of the 18 design variables to be irrelevant for the chomper problem, and 26 out of 38 for the twist chomper problem. For the twist chomper problem, the reduced 12-dimensional optimization reduces the overall computational time by as much as 90% when compared to the original 38-dimensional problem. Also, the best possible solution is found with this reduced dimensional optimization problem regardless of the initial training points.

Further research is needed in the use of Bayesian optimization for structural problems. In our study, the covariance function is stationary. The use of non-stationary covariance functions could lead to the discovery of new designs with fewer expensive function solutions.

Another limitation of the current study is that the size of design space is relatively small. The performance of BO for larger design problems and the use of other surrogate models can be part of the future work.

Bibliography

[1] Andrew Gillman, Kazuko Fuchi, and Philip Buskohl. Truss-based nonlinear mechanical analysis for origami structures exhibiting bifurcation and limit point instabilities. *International Journal of Solids and Structures*, 147:80–93, 2018. ISSN 0020-7683. doi: 10.1016/j.ijsolstr.2018.05.011. URL `https://doi.org/10.1016/j.ijsolstr.2018.05.011`.

[2] Andrew S. Gillman, Kazuko Fuchi, and Philip R. Buskohl. Discovering Sequenced Origami Folding Through Nonlinear Mechanics and Topology Optimization. *Journal of Mechanical Design, Transactions of the ASME*, 141(4):1–11, 2019. ISSN 10500472. doi: 10.1115/1.4041782.

[3] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Climate Change 2013 - The Physical Science Basis*, 104(1):1–30, 2015. ISSN 1098-6596. doi: 10.1017/CBO9781107415324.004.

[4] Eric Brochu, Vlad M Cora, and Nando de Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. eprint arXiv:1012.2599, arXiv.org, December 2010.

[5] Jonas Mockus. Application of Bayesian approach to numerical methods of global and stochastic optimization. *Journal of Global Optimization*, 4(4):347–365, 1994. ISSN 09255001. doi: 10.1007/BF01099263.

[6] Donald Jones, Matthias Schonlau, and William Welch. Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization*,

13:455–492, 1998. URL https://link.springer.com/content/pdf/10.1023{%}2FA{%}3A1008306431147.pdf.

[7] Simon Streltsov and Pirooz Vakili. A Non-myopic Utility Function for Statistical Global Optimization Algorithms. *Journal of Global Optimization*, 14(3):283–298, 1999. ISSN 09255001. doi: 10.1023/A:1008284229931.

[8] Michael James Sasena. *Flexibility and Efficiency Enhancements for Constrained Global Design Optimization with Kriging Approximations*. PhD thesis, University of Michigan, 2002.

[9] Atharv Bhosekar and Marianthi Ierapetritou. Advances in surrogate based modeling, feasibility analysis, and optimization: A review. *Computers and Chemical Engineering*, 108:250–267, 2018. ISSN 00981354. doi: 10.1016/j.compchemeng.2017.09.017. URL https://doi.org/10.1016/j.compchemeng.2017.09.017.

[10] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, oct 2001. ISSN 0885-6125. doi: 10.1023/A:1010933404324. URL https://doi.org/10.1023/A:1010933404324.

[11] Antonio Criminisi, Jamie Shotton, and Ender Konukoglu. *Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning*, volume 7. NOW Publishers, 2011. doi: 10.1561/0600000035.

[12] Carl Rasmussen and Christopher Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[13] Bradley Efron and David V. Hinkley. Assessing the accuracy of the maximum likelihood estimator: Observed versus expected fisher information. *Biometrika*, 65(3):457–483, 1978. ISSN 00063444. doi: 10.1093/biomet/65.3.457.

[14] Markus Abt and William J Welch. Fisher information and maximum-likelihood estimation of covariance parameters in Gaussian stochastic processes. *The Canadian Journal of Statistics*, 26(1):127–137, 2018.

[15] Kumar Vemaganti, Sandeep Madireddy, and Sayali Kedari. On the inference of viscoelastic constants from stress relaxation experiments. *Mechanics of Time-Dependent*

Materials, 2019. ISSN 15732738. doi: 10.1007/s11043-018-09403-y. URL http://dx.doi.org/10.1007/s11043-018-09403-y.

[16] Ziyu Wang and Nando de Freitas. Theoretical Analysis of Bayesian Optimisation with Unknown Gaussian Process Hyper-Parameters. pages 1–16, 2014. URL http://arxiv.org/abs/1406.7758.

[17] Harold J. Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 86(1):97–106, 03 1964. ISSN 0021-9223. doi: 10.1115/1.3653121. URL https://doi.org/10.1115/1.3653121.

[18] Daniel James Lizotte. *Practical Bayesian Optimization*. PhD thesis, University of Alberta, 2008.

[19] Dennis D. Cox and Susan John. SDO: A Statistical Method for Global Optimization. *Multidisciplinary Design Optimization: State-of-the-Art*, pages 315—-329, 1997. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.52.4169.

[20] Jian Wu, Matthias Poloczek, Andrew Gordon Wilson, and Peter I. Frazier. Bayesian optimization with gradients. *Advances in Neural Information Processing Systems*, 2017-Decem(3):5268–5279, 2017. ISSN 10495258.

[21] David Eriksson, Eric Hans Lee, Kun Dong, David Bindel, and Andrew Gordon Wilson. Scaling Gaussian process regression with derivatives. *Advances in Neural Information Processing Systems*, pages 6867–6877, 2018. ISSN 10495258.

[22] Anqi Wu, Mikio C. Aoi, and Jonathan W. Pillow. Exploiting gradients and Hessians in Bayesian optimization and Bayesian quadrature. pages 1–20, 2017. URL http://arxiv.org/abs/1704.00060.

[23] Adam D. Bull. Convergence rates of efficient global optimization algorithms. *Journal of Machine Learning Research*, 12:2879–2904, 2011. ISSN 15324435.

[24] Santu Rana, Cheng Li, Sunil Gupta, Vu Nguyen, and Svetha Venkatesh. High dimensional Bayesian optimization with elastic Gaussian process. *34th International Conference on Machine Learning, ICML 2017*, 6:4407–4415, 2017.

[25] Ziyu Wang, Masrour Zoghiy, Frank Hutterz, David Matheson, and Nando De Freitas. Bayesian optimization in high dimensions via random embeddings. *IJCAI International Joint Conference on Artificial Intelligence*, pages 1778–1784, 2013. ISSN 10450823.

[26] Kirthevasan Kandasamy, Jeff Schneider, and Barnabás Póczos. High dimensional Bayesian Optimisation and bandits via additive models. *32nd International Conference on Machine Learning, ICML 2015*, 1:295–304, 2015.

[27] Cheng Li, Sunil Gupta, Santu Rana, Vu Nguyen, Svetha Venkatesh, and Alistair Shilton. High dimensional Bayesian optimization using dropout. *IJCAI International Joint Conference on Artificial Intelligence*, pages 2096–2102, 2017. ISSN 10450823. doi: 10.24963/ijcai.2017/291.

[28] Benoit Choffin and Naonori Ueda. Scaling Bayesian Optimization uo to Higher Dimensions : A Review and Comparison of Recent Algorithms. *2018 IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2018.

[29] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian Optimization of Machine Learning Algorithms. *Advances in Neural Information Processing Systems*, pages 2951–2959, 2012.

[30] Kevin Swersky, Jasper Snoek, and Ryan P. Adams. Multi-task Bayesian optimization. *Advances in Neural Information Processing Systems*, pages 1–9, 2013. ISSN 10495258.

[31] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011, NIPS 2011*, pages 1–9, 2011.

[32] Jongbin Im and Jungsun Park. Stochastic structural optimization using particle swarm optimization, surrogate models and Bayesian statistics. *Chinese Journal of Aeronautics*, 26(1):112–121, 2013. ISSN 10009361. doi: 10.1016/j.cja.2012.12.022. URL `http://dx.doi.org/10.1016/j.cja.2012.12.022`.

[33] Xiaoning Fan, Pingfeng Wang, and FangFang Hao. Reliability-based design optimization of crane bridges using kriging-based surrogate models. *Structural and*

Multidisciplinary Optimization, 59(3):993–1005, Mar 2019. ISSN 1615-1488. doi: 10.1007/s00158-018-2183-0. URL https://doi.org/10.1007/s00158-018-2183-0.

[34] David Guirguis, Karim Hamza, Mohamed Aly mohamed, Hesham Hegazi, and Kazuhiro Saitou. Multi-objective topology optimization of multi-component continuum structures via a kriging-interpolated level set approach. *Structural and Multidisciplinary Optimization*, 51:733–748, 10 2014. doi: 10.1007/s00158-014-1154-3.

[35] Yan Zhang, Hao Li, Mi Xiao, Liang Gao, Sheng Chu, and Jinhao Zhang. Concurrent topology optimization for cellular structures with nonuniform microstructures based on the kriging metamodel. *Struct. Multidiscip. Optim.*, 59(4):1273–1299, April 2019. ISSN 1615-147X. doi: 10.1007/s00158-018-2130-0. URL https://doi.org/10.1007/s00158-018-2130-0.

[36] Yan Zhang, Liang Gao, and Mi Xiao. Maximizing natural frequencies of inhomogeneous cellular structures by kriging-assisted multiscale topology optimization. *Computers & Structures*, 230:106197, 2020. ISSN 0045-7949. doi: https://doi.org/10.1016/j.compstruc.2019.106197. URL http://www.sciencedirect.com/science/article/pii/S0045794919315299.

[37] Sebastian Dominguez, Nilima Nigam, and Bobak Shahriari. A combined finite element and Bayesian optimization framework for shape optimization in spectral geometry. *Computers and Mathematics with Applications*, 74(11):2874–2896, 2017. ISSN 08981221. doi: 10.1016/j.camwa.2017.08.044. URL http://dx.doi.org/10.1016/j.camwa.2017.08.044.

[38] Kai Liu, Tong Wu, Duane Detwiler, Jitesh Panchal, and Andres Tovar. Design for Crashworthiness of Categorical Multimaterial Structures Using Cluster Analysis and Bayesian Optimization. *Journal of Mechanical Design*, 141(12):1–15, 2019. ISSN 1050-0472. doi: 10.1115/1.4044838.

[39] Elena Raponi, Mariusz Bujny, Markus Olhofer, Nikola Aulig, Simonetta Boria, and Fabian Duddeck. Kriging-assisted topology optimization of crash structures. *Computer Methods in Applied Mechanics and Engineering*, 348:730 – 752, 2019. ISSN 0045-7825.

doi: https://doi.org/10.1016/j.cma.2019.02.002. URL http://www.sciencedirect.com/science/article/pii/S0045782519300726.

[40] Nicholas Turner, Bill Goodwine, and Mihir Sen. A review of origami applications in mechanical engineering. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 230(14):2345–2362, 2016. ISSN 20412983. doi: 10.1177/0954406215597713.

[41] Edwin A. Peraza-Hernandez, Darren J. Hartl, Richard J. Malak, and Dimitris C. Lagoudas. Origami-inspired active structures: A synthesis and review. *Smart Materials and Structures*, 23(9), 2014. ISSN 1361665X. doi: 10.1088/0964-1726/23/9/094001.

[42] Tomohiro Tachi. Simulation of Rigid Origami. *Origami 4*, pages 175–187, 2009. doi: 10.1201/b10653-20.

[43] Tomohiro Tachi. 3D Origami Design Based on Tucking Molecules. *Origami 4*, (Figure 1):259–272, 2009. doi: 10.1201/b10653-27.

[44] Jie Song, Yan Chen, and Guoxing Lu. Axial crushing of thin-walled structures with origami patterns. *Thin-Walled Structures*, 54:65–71, 2012. ISSN 02638231. doi: 10.1016/j.tws.2012.02.007. URL http://dx.doi.org/10.1016/j.tws.2012.02.007.

[45] Ke Liu and Glaucio H. Paulino. Nonlinear mechanics of non-rigid origami: An efficient computational approach. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473(2206), 2017. ISSN 14712946. doi: 10.1098/rspa.2017.0348.

[46] Kazuko Fuchi, Philip R. Buskohl, Giorgio Bazzan, Michael F. Durstock, Gregory W. Reich, Richard A. Vaia, and James J. Joo. Origami Actuator Design and Networking Through Crease Topology Optimization. *Journal of Mechanical Design, Transactions of the ASME*, 137(9):1–10, 2015. ISSN 10500472. doi: 10.1115/1.4030876.

[47] Kazuko Fuchi, Philip Buskohl, Giorgio Bazzan, Michael F. Durstock, Gregory W. Reich, Richard A. Vaia, and James J. Joo. Design optimization challenges of origami-based mechanisms with sequenced folding. *Journal of Mechanisms and Robotics*, 8(5):1–6, 2016. ISSN 19424310. doi: 10.1115/1.4032442.

[48] Marcelo Greco, Francisco Gesualdo, Wilson Venturini, and Humberto Coda. Nonlinear positional formulation for space truss analysis. *Finite Elements in Analysis and Design*, 42(12):1079–1086, 2006. ISSN 0168874X. doi: 10.1016/j.finel.2006.04.007.

[49] Krister Svanberg. The method of moving asymptotes - a new method for structural optimization. *International Journal for Numerical Methods in Engineering.*, 24(2):359–373, 1987.

[50] MATLAB Optimization Toolbox. Matlab optimization toolbox, 2016. The MathWorks, Natick, MA, USA.

[51] Richard H. Byrd, Jean Charles Gilbert, and Jorge Nocedal. A trust region method based on interior point techniques for nonlinear programming. *Mathematical Programming, Series B*, 89(1):149–185, 2000. ISSN 00255610. doi: 10.1007/PL00011391.

[52] Richard H. Byrd, Mary E. Hribar, and Jorge Nocedal. An interior point algorithm for large-scale nonlinear programming. *SIAM J. on Optimization*, 9(4):877–900, 1999. ISSN 1052-6234. doi: 10.1137/S1052623497325107. URL `https://doi.org/10.1137/S1052623497325107`.

[53] Richard A. Waltz, José L. Morales, Jorge Nocedal, and Dominique Orban. An interior algorithm for nonlinear optimization that combines line search and trust region steps. *Mathematical Programming*, 107(3):391–408, 7 2006. ISSN 0025-5610. doi: 10.1007/s10107-004-0560-5.

[54] Philip E. Gill, Walter Murray, and Margaret Wright. *Numerical linear algebra and optimization*, volume 1. Addison-Wesley, 1991.

[55] Michael J. D. Powell. Variable metric methods for constrained optimization. In R. Glowinski, J. L. Lions, and Iria Laboria, editors, *Computing Methods in Applied Sciences and Engineering, 1977, I*, pages 62–72, Berlin, Heidelberg, 1979. Springer Berlin Heidelberg. ISBN 978-3-540-35411-6.

[56] Willi Hock and Klaus Schittkowski. A comparative performance evaluation of 27 nonlinear programming codes. *Computing*, 30:335–358, 1983. ISSN 1436-5057. doi: 10.1007/BF02242139.

[57] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, New York, NY, USA, second edition, 2006.

[58] MATLAB Genetic Algorithm Toolbox. Matlab genetic algorithm toolbox, 2016. The MathWorks, Natick, MA, USA.

[59] Darrell Whitley. A genetic algorithm tutorial. *Statistics and Computing*, 4, 10 1998. doi: 10.1007/BF00175354.

[60] Christopher Houck, Jeffrey Joines, and Michael Kay. A genetic algorithm for function optimization: A matlab implementation. *NCSUIE-TR-95-09. North Carolina State University, Raleigh, NC, USA*, 22, 05 1998.

[61] Reiko Tanese, John H. Holland, and Quentin F. Stout. *Distributed Genetic Algorithms for Function Optimization*. PhD thesis, USA, 1989. AAI9001722.

[62] Devinder S. Sivia and John Skilling. *Data Analysis - A Bayesian Tutorial*. Oxford Science Publications. Oxford University Press, 2nd edition, 2006.

[63] Michael A Osborne, Roman Garnett, and Stephen J Roberts. Gaussian Processes for Global Optimization. *3rd International Conference on Learning and Intelligent Optimization LION3*, (x):1–15, 2009. URL `http://www.robots.ox.ac.uk/$\sim$mosb/OsborneGarnettRobertsGPGO.pdf`.

[64] Daniel J. Lizotte, Russell Greiner, and Dale Schuurmans. An experimental methodology for response surface optimization methods. *Journal of Global Optimization*, 53(4):699–736, 2012. ISSN 09255001. doi: 10.1007/s10898-011-9732-z.

[65] John Shawe-Taylor and Nello Cristianini. *Kernel methods for pattern analysis*. Cambridge University Press, 2004. ISBN 0521813972.

[66] MATLAB. *version 9.0.0 (R2016a)*. The MathWorks Inc., Natick, Massachusetts, 2016.

[67] Jan A. Snyman. *Practical Mathematical Optimization. An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms*. Springer, 01 2005.

[68] Radford M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag, Berlin, Heidelberg, 1996. ISBN 0387947248.