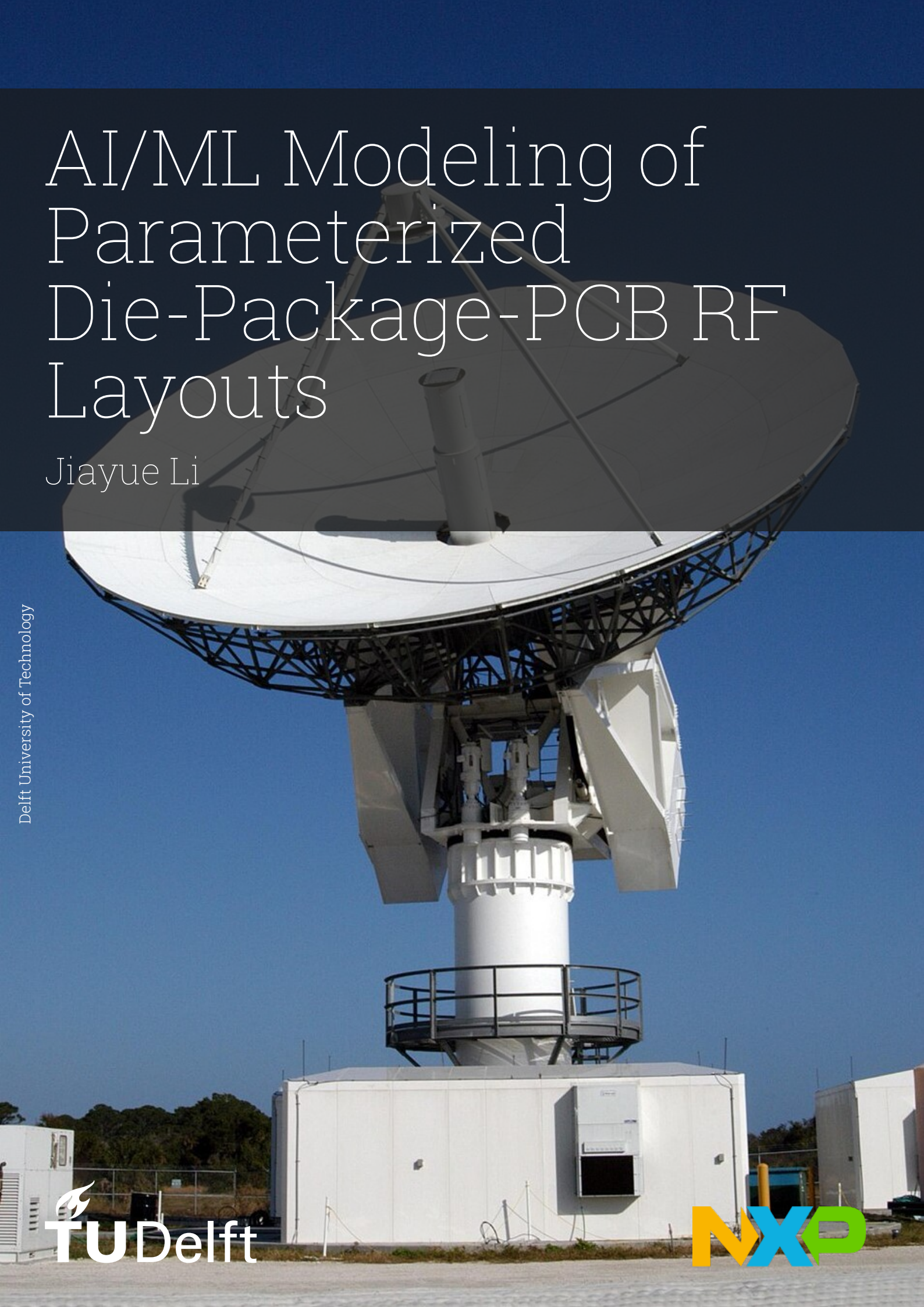
AI/ML Modeling of Parameterized Die-Package-PCB RF Layouts

Jiayue Li



Delft University of Technology

TU Delft

NXP

AI/ML Modeling of Parameterized Die-Package-PCB RF Layouts

by

Jiayue Li

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday September 26, 2025 at 10:00 AM.

Student number:	5955831	
Project duration:	February 1, 2025 – September 26, 2025	
Thesis committee:	Prof. M. Spirito,	TU Delft, supervisor
	Prof. D. Cavallo,	TU Delft
	Dr. L. Ntibarikure,	NXP Semiconductors

This thesis is confidential and cannot be made public until September 31, 2025.

Cover:	NASA or National Aeronautics and Space Administration, Public domain, via Wikimedia Commons
Style:	TU Delft Report Style, with modifications by Daan Zwaneveld

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

TU Delft

NXP

Preface

This master thesis marks the complesion of my studies in the MSc program in Electrical Engineering at Delft University of Technology, within the Faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS). The research presented here was carried out in partial fulfillment of the requirements for the degree, and reflects both an academic journey and a personal one.

The topic of this work emerged from my interest in combining electromagnetic (EM) simulation with modern machine learning techniques for advanced electronic packaging. During my graduation project, I had the opportunity to explore this direction in collaboration with NXP Semiconductors, where I conducted my thesis research. This experience not only allowed me to apply theoretical knowledge to industrially relevant problems but also provided me with valuable exposure to professional engineering workflows.

First and foremost, I would like to express my sincere gratitude to my daily mentor at NXP, Laurent Ntibarikure, for the continuous guidance, encouragement, and invaluable technical support throughout this work. His expertise and patient supervision have been central to the progress of my thesis and to my personal development as an engineer and a teacher.

I would also like to thank my university supervisor, Marco Spirito, for his academic guidance and constructive feedback during our meetings. His role in reviewing and critically assessing the thesis has been essential to bringing this work to completion.

Furthermore, I am grateful to my colleagues at NXP and fellow students at TU Delft for the discussions, collaboration, and motivating environment they created. I also want to thank my friends, both in Delft and elsewhere, for their companionship and encouragement. Their support outside the academic setting has provided balance and motivation throughout this challenging period.

Finally, I am profoundly grateful to my family, whose support and encouragement have been constant throughout my studies. Without their understanding and belief in me, this achievement would not have been possible.

Jiayue Li
Delft, September 2025

Summary

This thesis investigates electromagnetic (EM) variability in advanced die–package–PCB RF structures and proposes a methodology for constructing EM corner models. While process variation is routinely considered at the circuit level through PVT analysis, package-level effects are often neglected despite their significant impact at millimeter-wave frequencies.

A parameterized antenna-in-package model was developed in Ansys HFSS 3D Layout, and space-filling Latin Hypercube Sampling was used to explore the design space efficiently. Surrogate models were trained in OptiSLang and compared, after which a Gaussian Process framework in GPyTorch was implemented to predict S-parameters and antenna gain with high accuracy.

The results show that the surrogate models reproduce full-wave simulations with low error while reducing computational cost by orders of magnitude. This enables fast prediction of EM performance under process variation and systematic extraction of EM corners. The work demonstrates a scalable flow that integrates EM simulation and machine learning, providing an effective tool for robust package-level design.

Contents

1 Introduction

As the push for higher operating frequencies continues, especially in the millimeter-wave (mmw) range, ensuring optimal electromagnetic (EM) performance of the system being designed has become increasingly challenging. At these frequencies, even minor manufacturing variations in geometric parameters at package and PCB levels can significantly affect RF performance, leading to poor yields or limiting system capabilities. Traditionally, process-voltage-temperature (PVT) corner analyses are widely applied at the transistor and circuit levels to ensure robustness across manufacturing variations and operating conditions. However, systematic analysis of process variations, due to manufacturing tolerances, is rarely extended to the package and PCB domains, although their impact on high-frequency signals is large. This research aims to bridge this gap by introducing a structured approach to model parameterized Die+package+PCB RF transitions using a combination of electromagnetic simulations and machine learning (ML) techniques, potentially allowing cheaper package technologies to be exploited also for millimeter-wave designs.

To address this challenge, the study utilizes parametric modeling within Ansys HFSS 3D Layout and employs sophisticated sampling strategies and regression modeling methods available in Ansys OptiSLang [1]. Further innovation is introduced through integration with Gaussian process regression using the GPyTorch framework [2], allowing for easier model portability across computing platforms. By systematically varying geometrical parameters and analyzing their effects, this work facilitates efficient prediction of electromagnetic behavior under manufacturing uncertainties, significantly reducing computational costs compared to traditional Monte-Carlo methods. Therefore, the methodology presented facilitates the early stage design exploration while ensuring reliability in high-frequency packaging designs.

1.1. Motivation

Advanced semiconductor packaging technologies such as 2.5D interposers, 3D chip stacking, and high-density system-in-package (SiP) [3] have made the packaging level an increasingly critical domain for performance and reliability. In these complex assemblies, variations introduced during the manufacturing process of the package (e.g., in substrate wiring, bump dimensions, material properties) can significantly impact the electrical behavior of the system. Traditionally, circuit designers focus on process-voltage-temperature (PVT) corners at the silicon level to ensure that chips meet specifications under transistor process shifts, supply swings, and temperature extremes. However, process variation at the packaging level (outside the silicon die) can be equally impactful on system performance and has frequently been overlooked in standard design flows. Research has demonstrated that discrepancies in manufacturing tolerances of packages (e.g., variations in trace width or dielectric thickness on an IC substrate) can result in substantial signal integrity degradation, leading to waveform distortion and timing errors [4]. In other words, even if an IC is robust across PVT corners, real-world performance may be compromised if the package introduces unmodeled electromagnetic (EM) variability.

The need to model packaging process variations is especially urgent as industry pushes toward higher

frequencies and faster data rates. At gigahertz (GHz) frequencies, small geometric or material deviations in the package can alter impedance and coupling enough to violate design margins. For instance, in high-speed differential interconnects, slight changes in conductor geometry or dielectric constants due to manufacturing can skew impedance and induce waveform distortion [5]. Figure 1.1 illustrates how the center frequency (Fctr) of an Edge Coupled Microstrip Filter (ECMF) model changes as various manufacturing and material parameters varies. The slope of each line indicates the sensitivity of the center frequency to changes in that particular variable. Unlike on-chip device variations (which are well characterized by foundry PVT models), packaging variations involve a different set of parameters (e.g. substrate line/space width, via size, metal and dielectric thickness, etc.) and often lack a similarly established corner-modeling methodology. This means current package-level modeling lacks the standardized corners/methodologies that silicon PVT analysis enjoys. At the package/system level, variation modeling is often ad hoc, with a tendency to employ broad guardbands and overdesign rather than conducting precise corner analysis. Guardbands are extra design margins added to account for variations in manufacturing or operating conditions, ensuring reliable operation under worst-case conditions [6]. This practice can yield designs that are either overly conservative (pessimistic) or insufficiently robust, since engineers must rely on large safety margins in the absence of mature package-level corner models [7]. As a result, products may face unexpected issues such as signal integrity (SI) failures or electromagnetic interference (EMI) problems in later integration, highlighting the significance of explicitly modeling process variation at the package and board level.

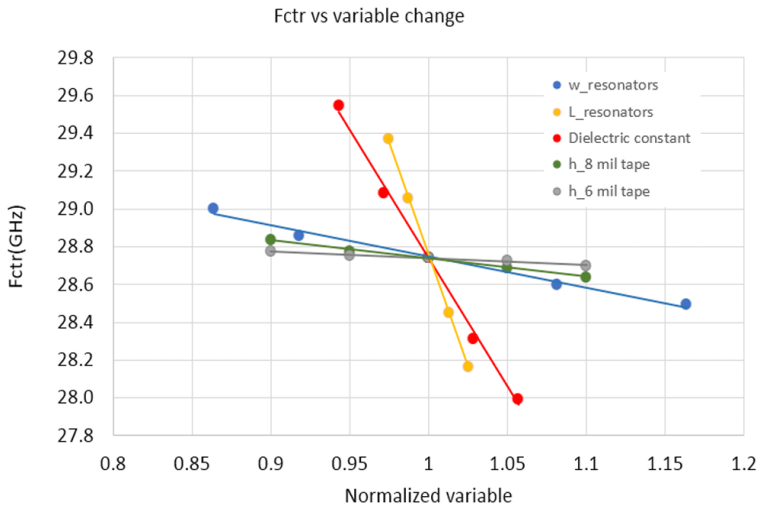


Figure 1.1: center frequency change vs normalized variable change in a coupled Microstrip [5]

1.2. State of the Art in EM Modeling

Recognizing these challenges, researchers and industry practitioners have begun addressing package-level variability and its EM effects. Early studies, conducted within the last two decades, demonstrated that package manufacturing variations can exert a critical impact on signal integrity. This observation underscores the necessity for comprehensive full-wave simulations and measurements to assess the impact of these variations [4]. Until recently, however, comprehensive variation analysis at the package level was limited by the sheer complexity of electromagnetic simulation. A full-wave 3D EM solver (e.g., Ansys HFSS) provides gold-standard accuracy for modeling high-frequency packaging structures from solder bumps and bondwires to interposer microvias. However, each simulation can be computationally expensive, often requiring hours or days of runtime for a single scenario. The exploration of a multitude of variation scenarios through brute-force simulation rapidly becomes intractable. To illustrate, a contemporary 2.5D interposer with numerous parameters (e.g., trace widths, spacing, dielectric properties) exerts substantial demands on EM tools, effectively testing the limits of their capabilities. Indeed, conventional EM simulators struggle to execute simulations at this scale of complexity without resorting to simplifications [8]. This has resulted in a status quo in which only a limited number of manual corner cases are simulated for packaging, or designers rely on experience-driven rules of thumb,

leaving a significant gap in our ability to systematically quantify and mitigate package-induced variation effects.

Concurrently, there has been a surge of interest in applying machine learning (ML) and advanced modeling techniques to problems in electronic design automation (EDA), including electromagnetics. Surrogate modeling (or metamodeling) has emerged as a key approach to handle expensive simulations. The idea is to run a limited set of high-fidelity simulations and use those results to train a more economical approximate model that can predict outcomes for new parameter settings almost instantly. Surrogate models have been employed in numerous engineering disciplines, including aerospace, RF/microwave circuit design, and mechanical systems, with the objective of reducing computational expense [9]. Common surrogate modeling techniques include response surface modeling (low-order polynomials), radial basis function (RBF) interpolation, support vector regression (SVR), artificial neural networks, and Kriging [9]. In the high-frequency electronics realm, researchers have applied these techniques to create fast EM response models that replace direct simulation for tasks like design optimization and yield analysis. Recent literature reports that machine learning models accurately capture the complex behavior of passive microwave components and antennas, thereby dramatically accelerating the design cycle [8]. In the field of advanced packaging, there is a growing trend of leveraging machine learning (ML) for the purpose of design co-optimization and the creation of "digital twins." A digital twin, in this context, refers to a data-driven model that mirrors the behavior of the package under various conditions [10]. This approach enables the execution of virtual experiments and what-if analyses with a level of efficiency that significantly surpasses that of building and testing numerous physical prototypes.

Despite these advances, a clear gap remains in the implementation of such surrogate machine learning approaches specifically to EM modeling under manufacturing variability at the package level. Traditional PVT analysis tools are not equipped to handle package geometry or process variations, and full-wave solvers are too slow to exhaustively simulate them. While surrogate modeling is a well-established strategy within the fields of RF and antenna communities, its adoption in the context of package electromagnetic variability modeling has been restricted. There is thus an opportunity to develop a methodology that marries full-wave EM simulation with modern machine learning to fill this gap. The objective is to develop a model that maintains the precision of electromagnetic analysis while concurrently acquiring the functionality to learn the behavior. This model will then be capable of rapid evaluation for any novel set of process parameters.

1.3. Thesis Overview and Contributions

This thesis proposes a novel design flow to address the above problem. In the proposed methodology, the initial step involves the parameterization of an EM model. Then simulations are performed using Ansys HFSS 3D layout integrated in OptiSLang, with the simulations conducted on an evenly distributed set of sample points. These sample points span the range of manufacturing variations. These variations may include, but are not limited to, varying trace widths, dielectric constants, and copper thicknesses, all within their respective tolerance ranges. Utilizing the data from these simulations, which constitutes a sparse sampling of the "variation space," we then employ a machine learning surrogate model to capture the mapping from process parameters to EM behavior. Specifically, we employ Gaussian Process Regression (GPR), implemented via the GPyTorch library, as our surrogate modeling technique. GPR is particularly well-suited to this task due to its capacity to generate interpolative predictions accompanied by quantified uncertainty. The trained GP model has the capacity to rapidly predict EM characteristics, including S-parameters and far-field gain total of antenna, for novel combinations of process parameters that were not explicitly simulated. This functionality effectively replaces exhaustive full-wave simulations with rapid machine learning inferences.

The contributions of this approach are anticipated to be manifold. Firstly, it exemplifies a pragmatic approach to incorporating packaging process variation in the preliminary design stage, thereby addressing the discrepancy between manufacturing tolerances and system electrical performance. By employing this approach, we can identify the worst-case EM corners at the package level. These are the most deleterious EM effects produced by the package. This identification process can be achieved without brute-force searching. Secondly, the Gaussian process surrogate provides uncertainty estimates for each prediction, which is invaluable in an engineering context. In instances where the model exhibits

high variability in its predictions, this approach can serve as an indicator of the necessity for additional simulation data within the specified region. This, in turn, facilitates an adaptive refinement of the model, thereby enhancing its accuracy and reliability. This approach ensures the efficacy of the methodology in terms of data efficiency. Thirdly, the integration of this surrogate into an EDA flow effectively creates a rapid "EM variation analysis" tool. For instance, designers could perform a Monte Carlo analysis of thousands of package builds in seconds through the surrogate, whereas doing the same with full HFSS runs would be computationally prohibitive. This kind of capability aligns with industry's move toward simulation-driven design and digital twins, which use machine learning (ML) to significantly shorten design iterations and amortize the cost of generating data.[11].

In summary, the significance of this work is that it introduces process variation awareness to electromagnetic modeling at the package level in a practical, scalable manner. The system utilizes cutting-edge machine learning algorithms (specifically, GPyTorch-based Gaussian processes) to ensure optimal accuracy while significantly enhancing operational efficiency. This approach can be regarded as an extension of the traditional PVT corner concept to the domain of packaging, thereby providing a method for defining and evaluating EM corners in the context of manufacturing variability. The remainder of this thesis is structured as follows:

1. Background Theory (Chapter 2): Presents the core concepts and background knowledge underpinning this work. Defines process variation and its effects in advanced packaging, clarify the notion of EM corners versus traditional PVT corners, outline relevant EDA design flows for packaging, and review surrogate modeling techniques with a focus on Gaussian Process Regression.
2. Methodology (Chapter 3): Details the proposed design flow. It covers the parameterized HFSS simulation setup, the design of experiments (DOE) for sampling the variation space, metamodeling methods comparison and choosing.
3. Machine Learning modeling based on Gpytorch (Chapter 4) Illustrates the training process for the GPR surrogate (including kernel selection and hyperparameter tuning), and the integration of python into OptiSLang.
4. Results Analysis and Discussion (Chapter 5): Showcases the application of our method to one or more case studies. Compares the surrogate's predictions against ground-truth EM simulations, demonstrate the speed/accuracy trade-offs, and illustrate how the surrogate can be used to find worst-case corners and perform sensitivity analysis.
5. Conclusion (Chapter 6): Summarizes the findings, discusses the implications for design practices, and suggests future extensions (such as applying the approach to different package types or incorporating other machine learning models).

2

Background Theory

This chapter provides the theoretical background for the research. It begins with an overview of process variation phenomena and their impact at the packaging level, highlighting the limitations of traditional PVT-based analysis. The distinction between electromagnetic corner analysis in packaging and the traditional process-voltage-temperature (PVT) corners used for on-chip design is then clarified. Next, the chapter outlines relevant electronic design automation (EDA) flows for modeling high-frequency packaging structures under manufacturing variability. Finally, the chapter reviews surrogate modeling techniques for performance prediction. These range from classical regression methods to modern machine learning approaches, with particular emphasis on Gaussian Process Regression (GPR) as a key method employed in this work.

2.1. Process Variation in Advanced Packaging

Process variation is defined as the inevitable deviations in manufacturing processes that cause physical parameters to differ from their intended nominal values. In the context of silicon IC fabrication, process variation encompasses discrepancies in transistor channel length, threshold voltage, metal line width, and other parameters across diverse wafers or within a chip. These variations are attributable to the constraints imposed by lithography and etching processes. At the level of packaging, process variations are evident in the materials and structures that connect and encapsulate the ICs. For instance, substrate fabrication tolerances, solder bump size variation, connector and bondwire placement accuracy, printed circuit board (PCB) manufacturing variations, and other factors contribute to variations in the packaging process. As advanced packaging technologies like flip-chip Ball Grid Array (BGA), 2.5D interposers, and 3D stacked dies have become prevalent, the performance impact of these package-level variations has grown. High-density interconnect features in a package substrate (e.g. microvias, fine-line routing, thin dielectric layers) are now more akin to PCB traces or even on-chip interconnect in scale, and their variability has the potential to significantly alter electrical characteristics. For instance, a slight increase in a trace's cross-sectional area (due to etching tolerance) will lower its impedance and insertion loss, whereas a small misalignment or spacing reduction could increase crosstalk or coupling to adjacent traces [12]. Similarly, variations in the dielectric constant or thickness of package insulation layers will change the capacitance and propagation delay of transmission lines [13]. These effects are not just hypothetical; empirical and simulation-based studies confirm that package manufacturing variations can degrade signal quality and timing. In one study, a ±10% process change in a package's differential line geometry led to noticeable distortion of a high-speed signal waveform [4]. In power delivery networks, variations can give rise to impedance spikes or resonances that threaten stability. Consequently, system integrators must take into account slow/fast package corners (e.g., minimum vs. maximum line thickness or dielectric constant) in their designs, as chip designers do with slow/fast transistor corners.

It is worth noting that process variation in packaging can be systematic or random. Systematic variations might include panel-to-panel or lot-to-lot shifts in etch depth or material properties. For example, a whole batch of substrate material having a dielectric constant slightly above spec. However, random variations cover the stochastic differences from one feature to the next, like the roughness or edge

profile differences along a long trace. Advanced packaging often involves multiple material systems (silicon, organic substrates, solder, copper, etc.) and multiple assembly steps, each introducing its own variability [7]. In comparison to on-die variations, packaging variations typically manifest on larger geometric scales (ranging from microns to millimeters as opposed to nanometers). However, it is noteworthy that the tolerance percentages associated with packaging variations can be relatively substantial. To illustrate, a PCB manufacturer might stipulate a tolerance of ±10-20% percentage points on trace width and spacing [14]. This range can yield substantial electrical deviation. Another salient point is that environmental factors (e.g., humidity or temperature during packaging processing) can induce variation, such as warpage of a substrate or die shift during bonding, which further complicates prediction [15]. The aforementioned factors collectively contribute to a distribution of outcomes in electrical performance. As technology progresses and we approach multi-GHz and sub-nanosecond rise times, these variations can lead to discrepancies in insertion loss, skew, bandwidth, and impedance matching that are not acceptable if not properly addressed.

In advanced packages such as 2.5D interposers (Figure 2.1) and chiplet-based systems, the interaction between chip and package variations also becomes important. The interposer or package routing serves as an extension of the on-die interconnect; thus, its variability can affect timing closure and signal integrity just like on-chip RC variation does. A noteworthy aspect in contemporary practice is that packaging variations are not traditionally co-modeled with silicon variations. Nevertheless, these variations are situated between the domains of chip design (managed by foundry PDK corners) and board design (frequently managed by PCB design rules) [16][17]. This discrepancy necessitates that engineers must either execute ad-hoc simulations of package perturbations or depend on conservative margins. The significance of our work lies in formally incorporating these variations into modeling. By treating the process parameters of the package as variables in simulations, we can quantify the extent to which they might cause performance spread or degradation. This is essential for robust design-for-manufacturability at the system level.

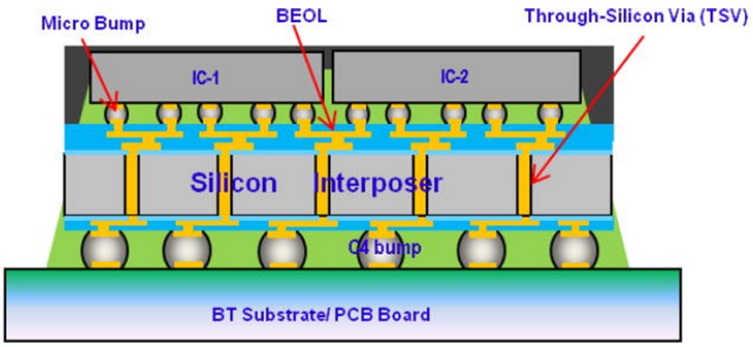


Figure 2.1: A typical cross-section schematic view of a 2.5D interposer-based package structure [18]

2.2. Electromagnetic Corners vs. Traditional PVT Corners

The concept of a “corner” in design refers to an extreme combination of conditions used to test worst-case performance. In traditional PVT corners for ICs, designers combine worst-case process parameters (e.g. slow transistors with high threshold voltage, or fast transistors with low threshold), worst-case supply voltages (min or max Vdd), and temperature extremes (cold or hot) to create scenarios that a chip must survive. These PVT corners, which include SS (slow-slow silicon), FF (fast-fast silicon), and TT (typical-typical silicon), are designed to guarantee that the circuit fulfills timing, power, and functionality requirements across the manufacturing and operating range. Figure 2.2 shows the statistical distribution of threshold voltages (V_{TH}) for NMOS and PMOS transistors in a 28nm technology node, in which the gray scatter points are the results from a Monte Carlo simulation reflecting the realistic statistical variations of device parameters due to process variability. The labeled circles (such as FF, SS, TT, etc.) represent different "corners" in the process model. These corners are deterministic shifts in model parameters corresponding to worst-case or edge-case scenarios, typically used to quickly evaluate circuit behavior under extreme conditions. it can be obeserved that the Monte Carlo points form a continuous distribution representing the actual random nature of process variations, whereas the corners are

discrete points located around the edges or specific regions of the parameter space. The PVT corner methodology has evolved over time, becoming increasingly intricate, particularly at advanced nodes. This evolution has introduced additional considerations, such as on-chip variability and interconnect corners (minRC, maxRC for parasitics), which have led to a substantial increase in the number of corners, with values reaching into the dozens or even hundreds. However, it should be noted that these are electrical corners for the semiconductor devices and wires on-chip. A notable omission in their analysis is the explicit consideration of off-chip wiring and packaging, a crucial aspect in the design and implementation of such systems. To illustrate, a chip might be verified at a "worst timing" corner of SS devices, low Vdd, high temperature, with maximum RC parasitics. However, it should be noted that this simulation likely assumes an ideal or nominal package model. In practice, if the package or PCB introduces additional delay or loss (due to its own variations), the true worst-case scenario may be worse than predicted. However, given that package variation is not included in the foundry PVT specification, it may not be adequately addressed. An industry expert has noted that "things like EM coupling are not... modeled [in standard corners]. They do not model inductance, so this is an unknown factor." This means that, traditionally, electromagnetic issues (coupling, radiation, etc.) are handled outside the normal corner analysis and often rely on designer experience [19].

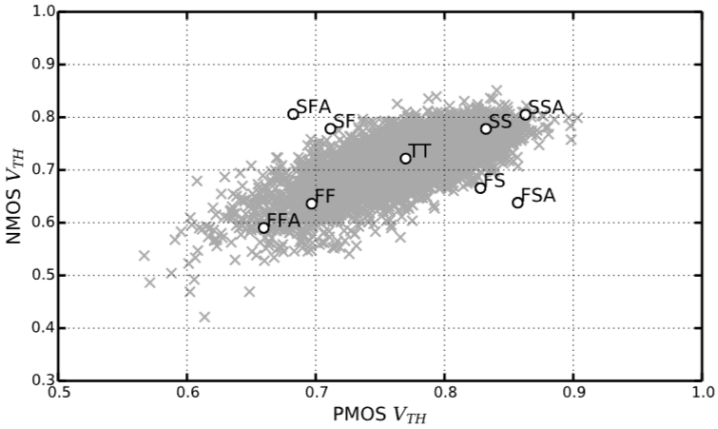


Figure 2.2: Corners and Monte-Carlo scatter plot of NMOS and PMOS V_{TH} (normalized) [20]

EM corners can be thought of as the counterpart to PVT corners, focusing on electromagnetic performance metrics rather than transistor-level metrics. Within the context of packaging, an EM corner may signify the extreme case of signal degradation or cross-coupling, attributable to process-induced physical variances. To illustrate this point, consider a high-speed differential pair routed in a package substrate. In this scenario, one EM corner could be designated as the "slow" EM corner, where the trace geometry and material deviations result in maximum signal delay and loss. These deviations may be attributed to factors such as minimum copper thickness, maximum dielectric constant, and smallest trace-width. This configuration leads to the highest resistance and capacitance. Conversely, a "fast" EM corner might involve maximum copper thickness and minimum dielectric constant, thereby yielding lower loss and faster signal propagation. Similarly, an EM corner for crosstalk might pair worst-case alignment of aggressor and victim lines (e.g., both at minimum spacing due to fab tolerance) with material properties that maximize coupling. These combinations are analogous to SS or FF corners in silicon, but tailored to electromagnetic outcomes such as insertion loss, return loss, or coupling coefficients.

One key difference is how these corners are determined. In silicon PVT, the foundry generally furnishes models for extreme process corners (frequently derived from statistical data, such as 3σ variations for device parameters). In the context of package design, there is not always a direct correlation between the physical dimensions of a package and the available space in a corner library. Instead, designers must rely on fabrication tolerance specifications to ensure the compatibility of the package with the available space. To illustrate, a substrate vendor might stipulate that trace width can vary by $\pm X\ \mu m$, and dielectric permittivity by $Y\%$. From these, a designer could define a corner model (min width, max permittivity, etc.). However, the fabrication of packages entails numerous variables that could interact, and the worst-case electrical impact might not be solely attributable to all-min or all-max values. In such cases, the surrogate modeling approach proves advantageous. Rather than preemptively determining

the most unfavorable combination, the model can be utilized to identify the extremes by sampling the space. Moreover, a conceptual understanding of EM corners facilitates the selection of the parameter bounds and combinations to simulate.

It's also important to consider voltage and temperature in an EM context. PVT covers voltage and temperature explicitly for chip performance. For package EM performance, temperature could affect material properties (e.g. copper resistivity increases with temperature, dielectric loss tangent might increase), and supply voltage mostly influences the chips rather than the passive package. So while "P" and "T" have analogues (process variations in package, and temperature's effect on materials), "V" is usually an external factor not directly altering passive EM behavior (aside from subtle effects like bias-dependent capacitances in some contexts). Therefore, EM corner analysis in packaging often centers on the manufacturing process side and environmental temperature, rather than supply voltage variation.

In summary, EM corners extend the corner analysis concept to include package-induced extremes in electrical performance. The subject under discussion differs from traditional PVT corners in that it is not concerned with transistor speed or leakage; rather, it is concerned with passive interconnect behavior under geometric and material uncertainty. By explicitly defining and analyzing EM corners, engineers can guarantee that the full system (i.e., the chip, the package, and the board) will function robustly. This thesis' approach contributes to the automation of the identification and evaluation of such EM corners by using a learned model of the EM behavior under process variation. This effectively widened the envelope of corner analysis to encapsulate the packaging domain.

2.3. EDA Design Flow for Electromagnetic Packaging Analysis

Electronic Design Automation flows have historically separated chip design and package/board design into different stages with different toolsets. Subsequent to the design and verification of an IC with PVT corners on the silicon, package engineers proceed to design the IC's package or the PCB on which it will be mounted. This configuration guarantees the presence of signal escape routing and power delivery, among other critical components. At this stage, electromagnetic (EM) simulation is employed to validate the signal integrity (SI), power integrity (PI), and electromagnetic interference (EMI) of the package. A typical workflow for packaging EM analysis involve:

1. extract a layout of the package (traces, planes, vias) from a CAD tool;
2. import it into a 3D EM solver like Ansys HFSS, CST Microwave Studio, or Cadence Clarity;
3. run full-wave simulations to obtain S-parameters or impedance profiles for critical nets (e.g. high-speed differential links, memory interfaces, or power distribution networks);
4. use those results in a circuit simulation (like SPICE or a channel simulator) together with the chip's I/O models to check eye diagrams, timing, noise, etc

This process is designed to prevent the degradation of signals or the occurrence of issues such as excessive voltage drop or radiated emissions [21].

However, the execution of EM simulation on large package structures necessitates substantial computational resources. A contemporary ball-grid array (BGA) package or a component of an interposer may comprise thousands of nets and features, with dimensions spanning centimeter-scale, yet exhibiting fine details on the order of microns [22]. The solution of Maxwell's equations on such a structure can necessitate millions of mesh elements. High-fidelity simulation software (HFSS) boasts sophisticated functionalities, including domain decomposition and substantial parallelization capabilities, which enable effective management of complex simulations [23]. However, these advanced features come at the expense of considerable runtime and memory consumption. The complexity of the system is further compounded by the necessity of conducting multiple simulations, particularly in scenarios involving manufacturing variations. Performing one hundred variation simulations of HFSS would require one hundred times the time of a single simulation, which could amount to weeks of computing time. This is clearly impractical in a fast-paced design cycle.

To address this challenge, EDA workflows are progressively integrating more sophisticated strategies that go beyond brute force. One such strategy that has been employed in the field is the utilization

of parameterized simulation and design of experiments (DOE). HFSS and analogous tools enable the parameterization of specific geometry or material properties. This functionality allows users to specify variables (e.g., "trace width" or "dielectric constant") in lieu of fixed values, enabling the tool to iterate over these variables. In conjunction with DOE or optimization software, the flow can be automated to execute a series of simulations that traverse the parameter space. For instance, Ansys offers optiSLang, an AI-driven optimizer, which can interface with HFSS. OptiSLang is capable of intelligently selecting simulation points with the objective of either optimizing a performance metric or constructing a response surface model. As demonstrated in an example documented by Ansys, an interposer model was established with six variables (trace widths, via radius, etc.), and optiSLang was employed to perform a smart DOE—a method of determining which parameters influence outcomes and converging on an optimal solution with far fewer runs than a full grid sweep [24]. This approach has been documented as achieving a 5-fold enhancement in simulation time for a multivariate optimization, in comparison to the conventional brute force method.

The concept of a metamodel or surrogate is also relevant in this context. optiSLang employs a "Metamodel of Optimal Prognosis," which is essentially a surrogate constructed from HFSS runs. In the context of advanced flows, there is an observed emergence of a phenomenon that could be designated as "simulation acceleration through AI." This very phenomenon is the focal point of the present thesis. Rather than treating the EM solver as a black box that must be run for every scenario, the solver is used strategically to generate data to train an internal model. As indicated by certain industry initiatives, such as that of EMA3D Connect, machine learning (ML) models have been demonstrated to predict the outcomes of electromagnetics (EM) simulations with a high degree of accuracy. These models are capable of achieving this prediction within seconds, a significant improvement over the hours or weeks typically required by conventional numerical solvers. This efficiency is achieved through the learning process, which involves the analysis of a limited set of simulations. This underscores a fundamental shift: EDA tools are beginning to integrate classical physics-based simulation with data-driven modeling to achieve both accuracy and speed.

Within the context of process variation at the packaging level, an envisioned EDA flow typically follows a structured sequence. After a package layout is defined, key variation sources and their ranges are identified. A design of experiments (DOE) is then created to sample the multidimensional space without resorting to exhaustive combinations, often by applying space-filling strategies. Selected points are simulated in a full-wave solver such as HFSS, and the results are used to train a surrogate model. Once established, this model can be exploited for broader analyses, for example scanning for worst-case corners, estimating yield through Monte Carlo evaluations, or exploring sensitivity trends. In this way, a limited set of simulations is leveraged to predict a much larger space of outcomes, significantly reducing computational cost while improving insight. Figure 2.3 illustrates this workflow, showing how DOE sampling and surrogate modeling replace exhaustive simulation.

It is important to ensure that the surrogate is validated within the flow. EDA validation may entail executing supplementary HFSS simulations at arbitrary points to substantiate the model's precision. In the event that the model is found to be inaccurate within a specific region, that region can be subjected to additional simulations (an iterative refinement). Contemporary flows have the capacity to integrate this adaptively. There are evident parallels in the practices of adaptive meshing in HFSS (refining the mesh where fields change rapidly) and adaptive sampling in the design space (refining the surrogate where response changes are large or uncertain). The integration of artificial intelligence and simulation, as evidenced by the evolution of electronic design automation (EDA) techniques, signifies a future direction that this thesis's methodology directly addresses.

In summary, contemporary electronic design automation (EDA) flows for packaging are beginning to treat electromagnet (EM) simulation under variability not as a brute-force task but as a modeling task. By combining parameterized solvers with carefully designed sampling strategies, such as space-filling DOE, engineers can reduce the number of simulations required while still capturing the effects of process variations. This thesis builds on that paradigm, using GPR as the modeling engine to achieve fast yet accurate EM analysis for packaging corners.

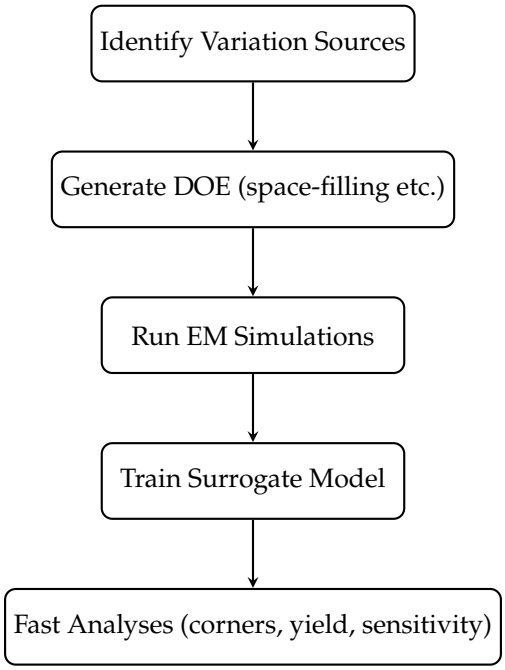


Figure 2.3: General EDA flow for packaging variation analysis.

2.4. Surrogate Modeling and Common Metamodeling Techniques

A surrogate model, also referred to as a metamodel, is a model that emulates the input-output behavior of a more complex simulation or physical experiment. The utilization of surrogates is employed in scenarios where the direct implementation of the simulation is either cost-prohibitive or excessively time-consuming for tasks such as optimization, Monte Carlo analysis, or real-time prediction. The general approach to surrogate modeling involves the following steps: first, a set of sample data is generated (i.e., inputs and corresponding outputs from the expensive model); second, a functional approximation technique is selected; third, the approximation is trained or fit to the data; and fourth, the surrogate is used to predict outputs for new inputs expeditiously. Surrogate models, in this context, function as stand-ins or proxies for the authentic model. This trade-off involves a compromise in fidelity for the sake of achieving substantial gains in processing speed.

In essence, if the true relationship is $y = f(x)$, a metamodel aims to provide an approximation $\hat{y} = g(x)$ such that $y \approx \hat{y}$ with acceptable error [25]. The general process of surrogate modeling involves:

1. generate a set of sample inputs and obtaining corresponding outputs from the expensive model;
2. select a suitable approximation method;
3. fit or train the surrogate on the sample data;
4. use the surrogate to predict outputs for new inputs much more quickly

A variety of regression and machine learning techniques have the capacity to function as surrogates, including but not limited to classical polynomial regression, Gaussian process models, support vector machines, and neural networks [26]. Each method comes with its own assumptions, strengths, and weaknesses, so it is important to choose an appropriate technique for the problem's complexity and data available. In the following section, we will review several common metamodeling techniques employed in ANSYS optiSLang, ranging from simple regression to advanced hybrid models. We will then proceed to delineate their theoretical underpinnings and distinctive characteristics.

2.4.1. Simple Polynomial Regression

Polynomial response surface models are among the oldest forms of surrogates in engineering design. In this approach, one assumes the response can be approximated by a polynomial function of the inputs, and determines the coefficients by least-squares regression. For example, a second-order polynomial

model in variables $x_1, x_2, ..., x_d$ can be written as:

$$\hat{y}(x) = \beta_0 + \sum_{i=1}^{d} \beta_i x_i + \sum_{i=1}^{d} \sum_{j \leq i} \beta_{ij} x_i x_j + \cdots , \tag{2.1}$$

including constant, linear, interaction, and possibly higher-order terms as needed. The coefficients β are fitted to minimize the squared error against the observed data. Polynomial surrogates are defined as global models, which are defined as follows: a single formula is intended to cover the entire design space [27]. The appeal of polynomial regression lies in its simplicity and low computational cost. It is easy to construct and interpretable, since the polynomial coefficients directly indicate the effect of each input on the output. This makes polynomials useful for sensitivity analysis and insight into parameter influence [26]. However, low-order polynomials often lack the flexibility to capture complex or highly nonlinear behaviors. Indeed, studies have found that while polynomial metamodels are cheap and straightforward, they tend to be less accurate than more advanced models like kriging on problems with strong nonlinearity [26]. In summary, a polynomial response surface provides a quick approximate model that works adequately for smooth, relatively simple responses, but it can suffer reduced accuracy if the true response is complex or involves significant interaction effects beyond the polynomial basis.

2.4.2. Moving Least Squares (MLS)

Moving least squares (MLS) is a local regression technique that improves on global polynomials by fitting localized approximations around the prediction point [27]. Instead of using one polynomial for the whole domain, MLS constructs an approximation on the fly at a target location x by weighting nearby sample data more heavily. In MLS, the estimate at x is typically expressed as:

$$\hat{y}(x) = \mathbf{p}(x)^T \mathbf{a}(x) \tag{2.2}$$

where $p(x)$ is a vector of polynomial basis functions (e.g. $\left[1, x_1, x_2, x_1^2, \ldots\right]^T$) and $a(x)$ are coefficients determined by solving a weighted least-squares fit using the known data [27]. The weights are higher for sample points near x and taper off for points farther away, typically using a decaying kernel or influence radius. In other words, MLS "moves" the fitting window across the design space, creating a piecewise-defined surface that is smooth and continuous. This approach can reduce interpolation error because local fitting allows the model to adapt to different regions of the input space without forcing a single global polynomial to fit everywhere [27]. Researchers have found that MLS surrogates often achieve better accuracy than a single global polynomial and can be competitive with other local methods like kriging in certain cases. For instance, MLS was shown to produce accurate response surfaces with fewer samples than a global quadratic model in some structural reliability problems [27]. One important consideration in MLS is the choice of weight function and support size D, which determines how "local" the approximation is. Smaller values of D lead to very local fitting, while larger values produce smoother, more global fits. Figure 2.4 illustrates this effect: with $D = 0.3$ the regression follows the data too closely and overfits, with $D = 5.0$ it becomes overly smooth and underfits, and with $D = 2.0$ it achieves a balanced compromise. This explicit dependence on the support size highlights why parameter selection is critical in practice. Overall, moving least squares provides a flexible compromise: it retains the smoothness of polynomial functions but with a locality that lets the model better capture complex variation by focusing on relevant neighboring data.

2.4.3. Support Vector Regression (SVR)

Support vector regression (SVR) is a statistical learning method that extends the principles of support vector machines to regression problems. In the SVR model, the objective is to identify a function $f(x)$, which may be a linear or nonlinear function characterized by kernels, that exhibits a maximum deviation of at most ϵ from the training targets y_i for all data points. This is achieved while maintaining a minimal complexity, or flatness, in the function. In the SVR model, the objective is realized through the introduction of an ϵ-insensitive loss. As illustrated in Figure 2.5, a tube of width 2ϵ is constructed around the regression function. Training points that fall inside this tube incur no penalty, while those lying outside are treated as support vectors and contribute to the loss. By adjusting the value of ϵ,

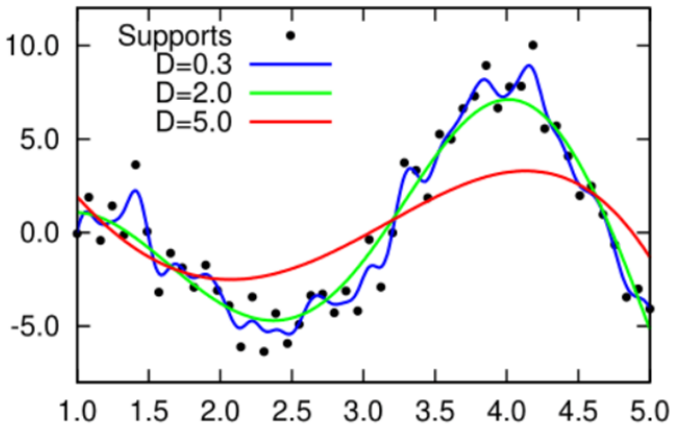


Figure 2.4: Influence of support size D on the Moving Least Squares regression. Three choices of D are compared: $D = 0.3$ (overfitting), $D = 2.0$ (balanced fitting), and $D = 5.0$ (underfitting). The example highlights how the support size controls the trade-off between local adaptability and global smoothness. [28]

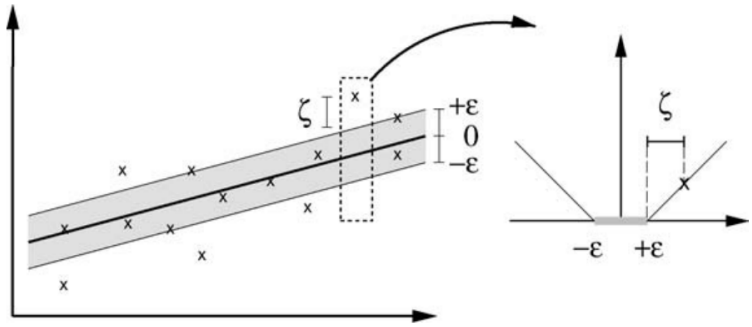


Figure 2.5: Illustration of the ϵ-insensitive tube in support vector regression. [29]

one can control the tolerance of the model and the sparsity of support vectors, thereby trading off accuracy against generalization capability. The optimization problem of SVR balances model complexity (often via a term $\frac{1}{2}\|w\|^2$ for a linear model weight vector) against the total error beyond ϵ, scaled by a regularization constant C. The solution to this convex optimization problem results in a sparse model, in which only a subset of the training points (i.e., the support vectors) influence the predictions. SVR can be combined with kernel functions (e.g. Gaussian RBF, polynomial kernel) to capture nonlinear relationships in the input features, analogous to the kernel trick in SVM classification. The resulting regression function is a weighted sum of kernel evaluations at support vectors.

$$f(x) = \sum_{i \in SV} \left(\alpha_i - \alpha_i^*\right) K\left(x_i, x\right) + b \tag{2.3}$$

In engineering and high-frequency design applications, several studies have shown that SVR can deliver high prediction accuracy and often outperform polynomial response surfaces, RBF models, and Kriging. For example, in high-speed link analysis SVR combined with active subspace methods has been used to accelerate sensitivity analysis while preserving fidelity [30], and in surrogate modeling with multi-fidelity setups SVR has compared favorably against Co-Kriging and Co-RBF in design optimization tasks [31]. This superior accuracy is often attributed to SVR's capacity to handle high-dimensional feature spaces and control overfitting through margin maximization. However, SVR models do require careful tuning of hyperparameters (the kernel type, kernel parameters, C, and ϵ) to perform well, and training involves solving a quadratic programming problem that can be computationally intensive for very large datasets. In summary, support vector regression provides a robust, kernel-based approach to nonlinear function approximation, offering excellent generalization in many cases, though at the cost of

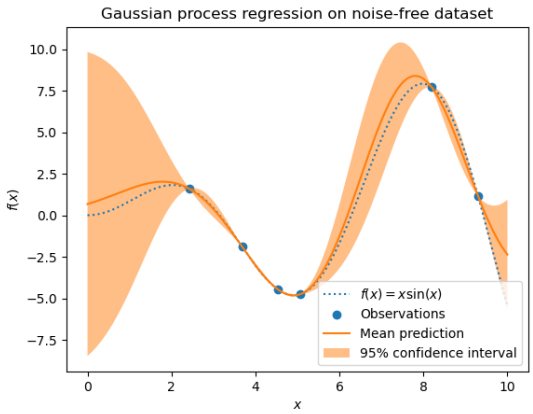


Figure 2.6: Example of Gaussian Process (kriging) surrogate modeling a 1-D function [33]

more complex training and less interpretability than simpler regression models.

2.4.4. Kriging

Kriging, also known as Gaussian Process (GP) regression, is a probabilistic metamodeling technique that originated in geostatistics and has become popular in engineering design for its accuracy in modeling complex responses. Kriging assumes that the unknown response function can be modeled as a realization of a stochastic Gaussian process [26]. In practice, this means we imagine $y(x)$ as $\mu + Z(x)$, where μ is a constant trend (or a low-order polynomial trend) and $Z(x)$ is a zero-mean Gaussian process with a chosen covariance kernel. The covariance function $K\left(x_i, x_j\right)$ is defined such that points with inputs that are close in distance have strongly correlated outputs. During training, the covariance hyperparameters are tuned so that the model interpolates the data and captures the observed variability. One key feature of GP regression is that it provides not just a prediction but also an estimate of uncertainty (variance) for each predicted point [32]. Given a set of N training samples, the GP posterior mean at a new point x can be written in closed form. For example, in the case of an ordinary kriging model with constant mean, the predictor can be expressed as:

$$\hat{y}(x) = \mu + r(x)^T R^{-1}(\mathbf{y} - \mu\mathbf{1}) \tag{2.4}$$

where μ is the estimated mean, $r(x)$ is the correlation vector between the new point and each training point, R is the $N \times N$ correlation matrix of the training inputs, y is the output vector of the training data and 1 is all-1 vector, multiplied with μ denotes the mean vector [32]. This predictor is derived by requiring the estimator to be unbiased and to minimize the variance of the prediction error (the best linear unbiased predictor principle). Kriging is an exact interpolator for noise-free data. It is capable of reproducing the training outputs with precision and interpolating predictions with a smooth transition between known points, while also widening the uncertainty band further from these points [33]. Because of its flexibility, kriging often yields more accurate approximations for highly nonlinear responses than global polynomials [26]. It has become a standard in global optimization (e.g. the Efficient Global Optimization algorithm) due to its ability to quantify uncertainty and balance exploration vs exploitation. Figure 2.6 illustrates a Gaussian process regression model trained on a small set of samples for a 1-D function. The model's mean prediction (orange curve) passes through all training points (blue dots) and is surrounded by a 95% ($\pm 1.96\sigma$) confidence interval (shaded region) that grows larger in regions away from any observation data.

Despite its accuracy and valuable uncertainty estimates, Gaussian process regression has some limitations. A practical drawback is the computational cost: training involves inverting an $N \times N$ covariance matrix, which is $O\left(N^3\right)$ time and $O\left(N^2\right)$ memory, making kriging difficult to apply directly when the number of training samples is large (e.g., thousands of points) [32]. Another challenge is that one must assume a form for the covariance (kernel) and estimate its hyperparameters; if the chosen kernel is not flexible enough, the GP may struggle to capture varying behavior of the true function across the input space. In

response to these issues, researchers have developed techniques like sparse GPs, as well as mixture of experts models (which we will discuss later) to partition the problem and use multiple local GPs [32]. Overall, kriging is a powerful metamodeling method that provides an optimal interpolation for deterministic data and a probabilistic framework to quantify model confidence. It often serves as a benchmark for accuracy against which newer surrogate methods are compared.

2.4.5. Genetic Aggregation of Response Surfaces (GARS)

Genetic Aggregation of Response Surfaces (GARS) is an ensemble-based surrogate modeling approach that seeks to automatically combine the strengths of multiple modeling techniques. Instead of relying on a single surrogate model, GARS uses a pool of candidate models and tries to construct an aggregated response surface that yields the best predictive performance for a given dataset [34]. The name "genetic aggregation" comes from the use of genetic algorithms or evolutionary strategies to select and weight the candidate models, although the exact implementation can vary. In a typical GARS approach, one might generate several surrogates (for example, a polynomial fit, a kriging model, an SVR model, etc., possibly with different hyperparameter settings), and then optimize a weighted linear combination of these surrogates. The aggregated model can be written as a weighted sum:

$$\hat{A}_m(x) = \sum_{l=1}^{m} \omega_l \hat{s}^{(l)}(x) \tag{2.5}$$

where each $\hat{s}^{(l)}(x)$ is one of the candidate surrogate models and ω_l is its weight in the ensemble. The weights are determined by minimizing an error metric (or a combination of metrics) on the training data. For instance, minimizing cross-validation error or AIC/BIC is one possible approach, while enforcing smoothness or simplicity in the aggregate is another [34]. Because this selection of models and weights can be posed as an optimization problem, genetic algorithms are a convenient method to evolve good combinations, hence "genetic aggregation." The GARS methodology effectively performs a form of model averaging/model selection, attempting to leverage the best aspects of each model in the ensemble. If one model is very accurate in a certain region or for certain output behavior, it can dominate the ensemble there, whereas other models may cover other regions, yielding a robust overall predictor. GARS has been reported to improve prediction accuracy by selecting the most appropriate surrogate form for the data at hand and even blending models to capture different response characteristics. For instance, in engineering and computationally expensive simulation settings, a GARS ensemble surrogate combining Support Vector Machine Regression, Gaussian Process Regression, Moving Least Squares have exhibited superior prediction accuracy in comparison to the performance of any individual model when utilized independently [34]. The ensuing ensemble A notable advantage of GARS is its adaptive flexibility, which allows it to function across diverse domains and kernels. However, the process can be computationally intensive since it trains many models and requires an additional optimization loop to tune the ensemble. There is also a risk of overfitting if the aggregation process becomes too complex (for example, using too many models or overly tuning weights), although cross-validation criteria typically mitigate this. In Ansys optiSLang, GARS is one of the available metamodel options (with an "auto-refinement" capability) to help find a high-quality surrogate by blending approaches. In summary, GARS provides a systematic ensemble approach to surrogate modeling, using evolutionary optimization to select and combine multiple response surface models, aiming to yield a surrogate that is more accurate and robust than any individual model.

2.4.6. Deep Feed Forward Networks (DFFN)

Deep Feed Forward Networks (DFFNs) are a class of machine learning models inspired by biological neural networks and widely used as surrogate models. They consist of an input layer, multiple hidden layers of interconnected neurons with non-linear activation functions, and an output layer that produces the prediction (Figure 2.7). Each neuron computes a weighted sum of its inputs followed by a non-linear activation such as a sigmoid or ReLU. The term "deep" refers to networks with several hidden layers, enabling the extraction of high-level features and the representation of highly non-linear functions. Training is performed by minimizing a loss function, commonly the mean squared error, using backpropagation and stochastic gradient descent [25].

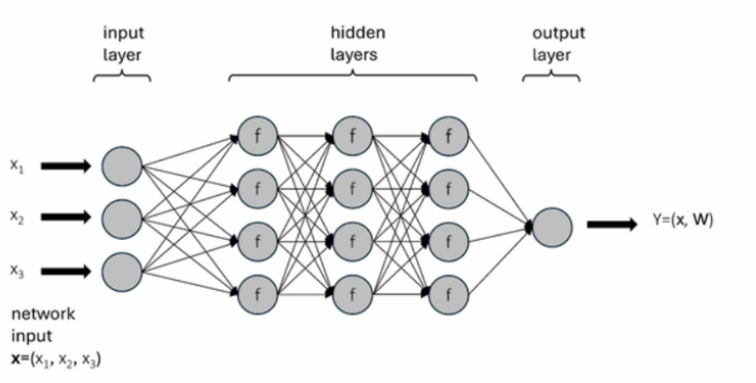


Figure 2.7: Schematic view of a Deep Feed Forward Network with 2 inputs and 3 hidden layers of 4 neurons with activation function f. [28]

DFFNs are known as universal approximators, meaning that with sufficient layers and neurons they can approximate any continuous function to arbitrary accuracy, provided adequate data. This expressive power allows them to model complex input–output relationships that may be difficult for parametric surrogates. Indeed, neural networks have been successfully applied in many engineering domains, including aerodynamic modeling and manufacturing process simulations, where they have shown excellent predictive performance even in high-dimensional problems [35]. They also naturally accommodate multiple outputs through multiple neurons in the output layer, which is convenient for multi-response data.

The main advantages of DFFNs as surrogates are flexibility and scalability. Unlike kernel-based models, they do not require a predefined basis and can learn features directly from data. Modern training algorithms and GPU acceleration further allow efficient training on very large datasets, while evaluations after training are extremely fast since they only involve matrix multiplications. However, DFFNs also have drawbacks: they often require large datasets to avoid overfitting, their performance depends heavily on careful tuning of the architecture and optimization parameters, and they generally lack interpretability compared with statistical models like kriging. Moreover, they do not provide uncertainty estimates by default, although methods such as dropout or Bayesian neural networks can partly address this limitation. Despite these challenges, DFFNs remain powerful and adaptive surrogate models that complement Gaussian process regression and other statistical approaches, often achieving state-of-the-art accuracy when sufficient data are available [26].

2.4.7. Deep Infinite Mixture of Gaussian Processes (DIM-GP)

Deep Infinite Mixture of Gaussian Processes (DIM-GP) is a recent surrogate modeling method that was introduced into optiSLang around 2022 through the Stochos library by PI Probaligence GmbH [36]. It combines neural networks with Gaussian processes in a mixture-of-experts framework: a deep neural network partitions the input space in a data-driven way, while local GP experts provide probabilistic regression with quantified uncertainty [37][32]. This design effectively creates a rich non-stationary GP model where the covariance can vary across the domain, guided by the deep network. The neural network portion enables scalability and feature learning for high-dimensional problems, while the GP portion ensures robustness, data efficiency, and the ability to provide confidence intervals.

The advantages reported for DIM-GP include high predictive accuracy even with relatively small datasets and robustness to noise or outliers. Compared to traditional surrogates such as polynomial response surfaces, radial basis functions, or standard kriging, DIM-GP has been shown to achieve superior performance in challenging scenarios. The “infinite mixture” terminology highlights the flexibility to activate an unbounded number of GP experts as needed, allowing the model to capture heterogeneous behaviors across different parameter regimes more effectively than a single global kernel. A schematic overview of this hybrid architecture is shown in Figure 2.8.

Despite its potential, DIM-GP is also more complex to train than conventional GPs or neural networks. Training likely involves adjusting both the neural network architecture and weights together with

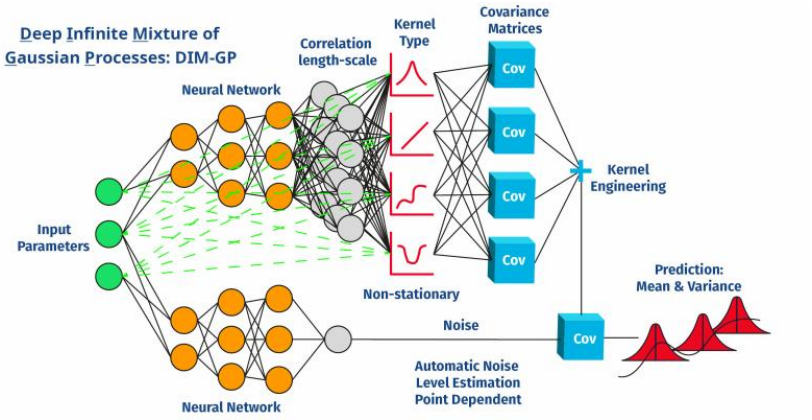


Figure 2.8: Schematic representation of DIM-GP [37]. A deep neural network adaptively partitions the input space, while Gaussian process experts model the outputs in those regions with quantified uncertainty.

GP hyperparameters, using advanced optimization or variational inference techniques. These details remain proprietary, but in practice the implementation in optiSLang hides such complexity from the user: the model can be trained in the same manner as other metamodels, with the internal algorithm handling the tuning. Overall, DIM-GP represents a state-of-the-art surrogate that combines scalability, uncertainty quantification, and strong generalization, making it especially promising for complex engineering problems where accurate predictions must be obtained from limited simulation data [37].

2.5. Gaussian Process Regression (GPR): Theory and Properties

2.5.1. Gaussian Processes: Distributions over Functions

A Gaussian Process (GP) is a probabilistic model for functions. Formally, a GP defines a distribution over possible functions $f(x)$, such that for any finite set of input points, the corresponding function values have a joint multivariate Gaussian distribution [38]. We often write this as:

$$f(x) \sim \mathcal{GP}\left(m(x), k\left(x, x'\right)\right) \tag{2.6}$$

meaning $f(x)$ is assumed to be a GP with mean function $m(x)$ and covariance function $k\left(x, x'\right)$ [39]. In other words, for any collection of inputs $x_1, \ldots, x_n$, the vector of outputs follows $\left(f\left(x_1\right), \ldots, f\left(x_n\right)\right) \sim \mathcal{N}(m, K)$, where $m = \left[m\left(x_1\right), \ldots, m\left(x_n\right)\right]^T$ and K is the $n \times n$ positive-definite matrix with entries $K_{ij} = k\left(x_i, x_j\right)$. The GP is fully specified by the choice of mean and covariance functions. In practice one often assumes a constant or zero mean prior for simplicity, so that all the *structure* of the function is encoded by the covariance kernel [38]. The kernel $k\left(x, x'\right)$ defines the prior correlation between function values at inputs x and x', thereby encapsulating our assumptions about the function's behavior [40]. For example, the kernel might assume the function is very smooth, periodic, or has other traits, as discussed below. Each observed data point is thus modeled as a realization of a Gaussian random variable (the function value plus possibly noise) with variance given by the kernel's self-covariance, and correlation with other points given by the kernel [40].

Gaussian Process Regression (GPR) refers to using a GP as a prior over an unknown function and updating this prior with data to obtain a posterior GP. It is a Bayesian non-parametric regression approach. Given a training dataset $D = \{X, \mathbf{y}\} = \left\{\left(x_i, y_i\right)\right\}_{i=1}^{N}$, with (for example) $y_i = f\left(x_i\right) + \epsilon_i$ and ϵ_i being i.i.d. Gaussian observation noise, we can analytically compute the posterior distribution of the function. The result is also a GP : $f(x) \mid D \sim \mathcal{GP}\left(m_*(x), k_*\left(x, x'\right)\right)$. In particular, the prediction at a new input x^* is Gaussian with mean and variance given by the well-known formulas:

$$\begin{aligned} m_*\left(x^*\right) &= m\left(x^*\right) + k\left(x^*, X\right)\left[K(X, X) + \sigma_n^2 I\right]^{-1}(\mathbf{y} - m(X)) \\ \operatorname{Var}\left[f\left(x^*\right)\right] &= k\left(x^*, x^*\right) - k\left(x^*, X\right)\left[K(X, X) + \sigma_n^2 I\right]^{-1} k\left(X, x^*\right) \end{aligned} \tag{2.7}$$

where $K(X, X)$ is the covariance matrix of the training inputs computed by the kernel, and σ_n^2 is the noise variance. Intuitively, the predictive mean is a weighted interpolation of the observed data values, and the predictive variance gives an uncertainty that grows away from observed points. Close to training data, the GP's confidence is high (low variance), and far away it reverts to the prior mean with large uncertainty. This property makes GPR a powerful tool for modeling engineering systems: it provides not only a prediction but also a confidence interval for that prediction [40].

2.5.2. GPs in Electromagnetic Modeling

In the context of electromagnetic (EM) systems under process variation, GPR offers an efficient way to model expensive simulations or measurements as random functions of manufacturing or design parameters. For instance, one can treat an S-parameter frequency response $S(p)$ (a metric of an EM device's performance) as a function of a vector of process parameters p (geometrical or material variations), and place a GP prior on $S(p)$. The GP is "trained" on a limited set of high-fidelity simulations or measurements, and can then predict the output for new parameter settings with an uncertainty estimate. This yields a fast surrogate model for tasks like yield estimation [39]. Notably, GPs tend to be data-efficient and less prone to overfitting than high-capacity models like neural networks, because the kernel acts as a built-in regularizer reflecting prior knowledge [40]. The key to success is that the kernel must be chosen to reflect the known physical behavior, which will be discussed in the next section.

2.5.3. Covariance Kernels and Prior Assumptions

The choice of covariance kernel $k(x, x')$ is central in GPR, as it encodes assumptions about the underlying function's properties [38][40]. The kernel defines how outputs at two input points are correlated a priori, effectively imposing smoothness, periodicity, or other structural qualities. It is evident that kernels impose certain constraints on the function that is to be modeled. For example, a Gaussian kernel assumes the function is very smooth, a Matérn kernel allows less smoothness (finite differentiability), and a periodic kernel assumes the function repeats with some period [38]. By selecting or designing a kernel, we inject domain knowledge into the GP. In an engineering context, if we expect a response that is fairly smooth except for some periodic oscillations, we can choose a kernel that encodes exactly that. Below we describe two common kernels used in this work – the Radial Basis Function (RBF) kernel and the Periodic kernel – and how they can be combined to model structured yet smooth behaviors. We also briefly note other kernel types and GP variants at the end.

Radial Basis Function (RBF) Kernel

One widely used covariance is the Radial Basis Function (RBF) kernel (also known as the squared exponential or Gaussian kernel). In its isotropic form, it is given by:

$$k_{\mathrm{RBF}}(x, x') = \sigma_f^2 \exp\left(-\frac{\|x - x'\|^2}{2\ell^2}\right) \tag{2.8}$$

where ℓ is a length-scale parameter and σ_f^2 is the output variance (signal variance) [39][41]. The RBF kernel assigns high covariance to inputs that are close in the input space (relative to ℓ), and the covariance decays as a Gaussian function of distance. Intuitively, ℓ controls the smoothness (how quickly the function can vary): a large length-scale means the function changes only gently over input space, while a small ℓ allows rapid variation in the function (short-wavelength features). The variance σ_f^2 controls the overall vertical scale of function variations (how much outputs vary around the mean). The RBF kernel is stationary and isotropic, depending only on $\|x - x'\|$. It produces very smooth functions – in fact, sample functions from an RBF GP prior are infinitely differentiable. This kernel is often considered a "default" because of its universal approximation ability and its simplicity (only two hyperparameters) [41].

In practice, using an RBF kernel implies a strong belief that the target function has no discontinuities or sharp changes. Points that are close in the input space are assumed to have nearly equal outputs. This is a reasonable assumption in many EM modeling scenarios where the device response varies smoothly with geometry or material parameters. Indeed, in the yield estimation study of Fuhrländer and Schöps [39], a squared-exponential (RBF) covariance was used to model how an RF filter's S-parameter changes

with manufacturing parameters. The authors treated the S-parameter $S(p)$ as a GP with an RBF kernel, reflecting the assumption that small changes in the fabrication parameters lead to small, smooth changes in the electrical response. In general, an RBF kernel GP will interpolate the training data with a smooth curve (or surface in higher dimensions) and tends to smooth out noise as well. One must be cautious: if the true function has non-smooth behavior (e.g. sharp resonances or edges), a pure RBF kernel can struggle – it will force a smooth fit and may need a very small ℓ to fit sharp features, which in turn can make the model behave poorly elsewhere [41]. In such cases, one may choose a less smooth kernel (like a Matérn) or add additional kernels to capture the structure.

Periodic Kernel

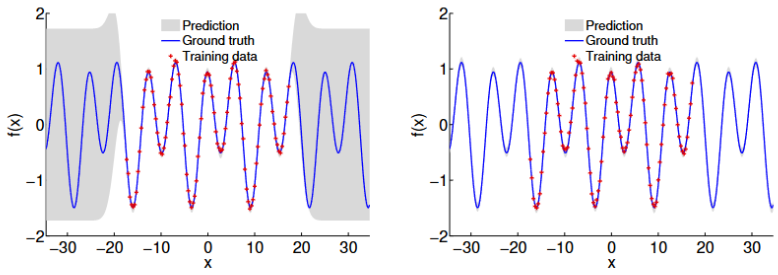
The Periodic kernel (sometimes called the exp-sine-squared kernel) encodes the assumption that the function repeats itself exactly with some period. One formulation of a periodic covariance in one dimension is

$$k_{\text{Periodic}}\ (x, x') = \sigma_p^2 \exp\left(-\frac{2\sin^2\left(\frac{\pi}{p}(x - x')\right)}{\ell_p^2} \right) \tag{2.9}$$

where p is the period of repetition and ℓ_p is a length-scale parameter controlling the smoothness within a period [42]. Intuitively, this kernel first maps the distance $|x - x'|$ into a sinusoidal space $\left(\sin^2\left(\frac{\pi}{p}(x - x')\right)\right)$ so that inputs separated by the period p (or its multiples) are mapped to a difference of zero (hence high covariance). The exponential then ensures that if two points are out of phase (not aligned on the periodic pattern), their covariance decays, with ℓ_p governing the rate of decay. The result is that $k_{\text{Periodic}}\ (x, x')$ gives high similarity whenever x and x' are at nearly the same position within a cycle of length p, enforcing exact periodicity. Essentially, under this kernel the GP prior considers functions that repeat every p (with possible smooth variations from cycle to cycle if combined with another kernel, as we discuss later). The hyperparameters σ_p^2 and ℓ_p play roles analogous to those in the RBF kernel: σ_p^2 is the signal variance and ℓ_p controls how quickly the function can vary within one period (larger ℓ_p means the function shape within each period is smoother, whereas smaller ℓ_p allows more jagged periodic oscillations).

Periodic kernels are useful for modeling repeating patterns in data, such as seasonal effects in signals or frequency responses with regular resonances. They have been applied in signal processing and time-series modeling where a strong periodic component is known a priori [38]. In our electromagnetic modeling context, one can imagine using a periodic kernel to capture resonant behavior or frequency responses that show ripple patterns. For example, Garbuglia et al. (2023) used a GP with a periodic kernel to model wide-band S-parameter curves of high-speed interconnects, which exhibit oscillatory behaviors due to resonances and crosstalk. A standard RBF kernel assumes high smoothness and would fail to confidently extrapolate or interpolate these oscillations[40]. By contrast, a periodic kernel GP can represent the oscillations inherently. In fact, it's known that a GP with a pure periodic kernel will extrapolate a learned periodic signal with confidence – since the kernel "knows" the function repeats, the GP can predict beyond the training range without uncertainty growing, as long as the periodic pattern is clear [38]. Figure 2.9 contrasts a GP with an RBF kernel versus a GP with a periodic kernel on a periodic signal: the RBF GP's predictions revert to the prior with large uncertainty outside the training region, whereas the periodic GP continues the learned oscillation with low uncertainty, tracking the signal.

One limitation of the basic periodic kernel is that it enforces strictly periodic functions of a fixed period. Real physical systems might show quasi-periodic behavior – approximately repeating patterns that slowly evolve or have an underlying trend. For example, an EM signal might oscillate in a pseudo-periodic way where the amplitude or baseline drifts over time, or a frequency response might have a repeating ripple whose magnitude changes across the band. A pure periodic kernel cannot capture such non-repeating trends, so we often combine it with a more flexible kernel to model structured but non-stationary behavior.



(a) Gaussian kernel. (b) Periodic kernel.

Figure 2.9: Comparison of GPs with a Gaussian and periodic kernels to model a periodic signal. Red crosses denote the training targets y_i and the blue line the true latent function. The shaded areas represent the 2σ bound of the marginal predictive distribution of the GP models.

Combining Kernels for Structured, Smooth Behavior

One of the powerful aspects of GPs is that kernels can be combined (through addition or multiplication) to create richer covariance functions that capture multiple aspects of the behavior [41]. For instance, if we believe our function has an overall smooth trend plus a periodic fluctuation around that trend, we can use a sum of kernels:

$$k_{\text{sum}}(x, x') = k_{\text{smooth}}(x, x') + k_{\text{periodic}}(x, x') \tag{2.10}$$

This results in a GP that is effectively the sum of two independent GP components – one smooth, one periodic. The smooth kernel could be an RBF (or Matérn, etc.), and the periodic kernel handles the oscillatory part. The model will then explain covariance between points as a combination of long-lengthscale variation and periodic variation. Such a model might be appropriate, for example, if an S-parameter has an overall frequency response shape (captured by an RBF) with superimposed ripple due to impedance mismatch (captured by a periodic term). Another way to combine is multiplication:

$$k_{\text{prod}}(x, x') = k_{\text{Periodic}}(x, x') \times k_{\text{RBF}}(x, x') \tag{2.11}$$

sometimes called a locally periodic or quasi-periodic kernel [41]. In this product, the RBF acts as an envelope that modulates the periodic kernel. The resulting GP produces functions that are periodic but whose amplitude or shape can vary gradually over time (or input space). In other words, the function repeats with period p, but the repeating pattern itself can drift slowly. This is often a very realistic assumption for physical signals that are not perfectly stationary. Indeed, the so-called quasi-periodic GP kernel (product of periodic and RBF) has been used to model e.g. stellar activity signals that have periodic stellar rotation modulations with evolving amplitude, and it is equally applicable to engineering signals that show periodic-like behavior under slow variation [2]. In our context, combining an RBF kernel with a periodic kernel allows the model to capture smooth global trends together with local cyclic variations.

Kernel design is problem-dependent. The above combinations can be expanded as needed, for example, one could add multiple periodic kernels of different periods to model a sum of periodic components, or include linear and RBF kernels together to allow both linear trends and smooth nonlinear deviations. The guiding principle is that each kernel encodes an assumed pattern or property, and the sum/product constructs a covariance that has all those properties. By examining the kernel formula, one can often interpret what kinds of variation the GP can represent. A benefit of this modular approach is that it keeps the model interpretable to engineers: we can say, for instance, that "our model assumes the device response consists of a slowly-varying baseline plus an oscillatory component with period X", which is more insightful than a black-box model. In summary, combining the RBF and periodic kernels (either additively or multiplicatively) provides a flexible prior for structured yet smooth behaviors – capturing periodic structure without losing the general smoothness expected of physical responses.

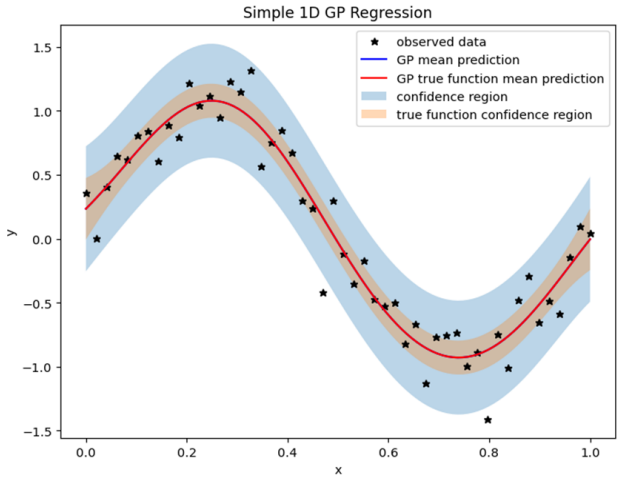


Figure 2.10: Example of Exact Gaussian Process (GP) regression with predictive mean and confidence intervals

2.5.4. Exact GP Model in GPyTorch

For this thesis, we employ an Exact Gaussian Process regression model as implemented in GPyTorch. "Exact" GPR implies that we perform the standard GP inference without resorting to approximations – in other words, we use the full kernel covariance on all training points and compute the posterior predictions as above. GPyTorch's framework allows us to specify a mean function (e.g. constant mean), a covariance kernel (in our case, we use the kernels described above, implemented as RBFKernel, PeriodicKernel, etc.), and a likelihood (we use a Gaussian likelihood for regression). The model is then trained by maximizing the marginal likelihood (or equivalently, minimizing negative log likelihood), which adjusts the kernel hyperparameters (length-scales, variance, etc.) to fit the data [38]. GPyTorch under the hood leverages modern linear algebra techniques and GPU acceleration to handle exact GP inference even for reasonably large datasets. Notably, it uses a method called Blackbox Matrix-Matrix (BBMM) inference to reduce the complexity of solving the linear system for the posterior from the naive $O(N^3)$ to $O(N^2)$ in many cases, enabling scaling to larger N than traditional implementations [2]. In our application, the number of training samples is relatively moderate (since each sample may come from an expensive EM simulation), so exact inference is feasible – the cubic complexity of factorizing the K matrix is not prohibitive at this scale [43]. This exact GP approach has the advantage of preserving full predictive accuracy and correctly estimated uncertainty, as opposed to approximate GP methods which trade some accuracy for speed.

Figure 2.10 demonstrates an simple case of exact Gaussian Process (GP) regression using GPyTorch. The original function is:

$$\begin{aligned} y &= \sin(2\pi x) + \epsilon \\ \epsilon &\sim \mathcal{N}(0, 0.04) \end{aligned} \tag{2.12}$$

The plot shows the observed data points (black stars), the predicted GP posterior mean (blue line), and the true underlying function mean (red line). The shaded regions represent the corresponding confidence intervals: the blue band indicates the GP's predictive uncertainty, while the red band shows the true function's confidence region. This example demonstrates the efficacy of GPyTorch's ExactGP model, which utilizes an RBF covariance kernel and Gaussian likelihood, in accurately capturing the underlying function behavior and quantifying predictive uncertainty. This is achieved by following the standard workflow of training and inference outlined in the GPyTorch regression tutorial.

It is worth noting that many GP variants exist beyond the exact model. For example, sparse Gaussian Processes introduce inducing points to approximate the covariance and reduce complexity, enabling GP regression on tens of thousands of points or more. Variational GPs and approximate inference techniques (like those in GPyTorch's VariationalStrategy modules) are used when data are large or when

one needs to handle non-Gaussian likelihoods. There are also Deep GPs (hierarchical compositions of GPs) that can model more complex, non-stationary behavior, and multi-output GPs for vector-valued functions (relevant in EM if modeling frequency responses at multiple frequencies simultaneously, for instance). In this thesis, however, we focus on the classical exact GP regression model with Gaussian likelihood, as it is sufficient for the problem sizes and provides a clear interpretation.

2.5.5. Limitations of GPR

The most cited limitation is computational scalability. Training a GP (which involves inverting the $n \times n$ covariance matrix) has computational complexity $O(n^3)$ and memory complexity $O(n^2)$, where n is the number of training points. Here, the notation $O(\cdot)$ denotes the asymptotic growth rate commonly used in computational complexity analysis; it expresses how the required resources scale with n, ignoring constant factors and lower-order terms. This cubic scaling means that for large n (say $n > 10^4$), the direct GPR becomes computationally very slow and memory-intensive [44]. For example, 1,000 points is usually fine (inverse of a 1000x1000 matrix is fast on modern hardware), but 10,000 points might be borderline, and 100,000 points is generally intractable without approximations. This is why many research efforts focus on sparse Gaussian processes or approximation methods that reduce complexity (by using inducing points, low-rank approximations, etc.). In our thesis context, we anticipate on the order of perhaps hundreds of simulations – which is within the feasible range – but if one envisioned scaling to thousands, we would need to incorporate such approximations. Modern frameworks like GPyTorch are designed to handle larger datasets by leveraging GPUs and implementing scalable kernel matrix algebra; for instance, GPyTorch can use iterative solvers or exploit special structure in the covariance matrix to go beyond the naive $O(n^3)$ inversion.

Another limitation is that GPR models, by virtue of using a fixed kernel, assume a certain smoothness and stationarity of the function which might not hold. If the actual EM response has sharp discontinuities or non-stationary behavior (e.g., sudden jumps or regime changes at certain parameter thresholds), a single GP with a standard kernel might struggle to fit that. One can sometimes address this by using adaptive length scales or a mixture of GPs (piecewise modeling), but it adds complexity. In our application, EM responses (like S-parameters as a function of geometry) are generally smooth and continuous, so GPR's assumptions are reasonable.

Finally, in high-dimensional input spaces, GPR (like any regression) faces the curse of dimensionality. The kernel might become almost zero for most pairs of distant points in a high-D space, making it hard to learn without an exponential increase in data. However, for packaging variations, the dimensionality (number of independent variation parameters) is typically modest – perhaps a few to maybe a dozen at most significant variables – which is manageable.

2.5.6. Summary and References

In summary, a Gaussian Process regression model is a principled way to interpolate an expensive EM simulation or measurement dataset, treating it as a random function with a prior defined by a covariance kernel. By choosing kernels such as the RBF and Periodic kernels – and combinations thereof – we can encode expectations of smoothness and periodic structure in the EM responses under process variation. This yields a surrogate model that not only fits the data but also generalizes in a physically sensible way (e.g. extrapolating periodic patterns) and provides uncertainty estimates. The theory presented here is grounded in established literature: Rasmussen and Williams' textbook offers a comprehensive introduction to GPs, while numerous research papers demonstrate GPR in engineering contexts [38]. For instance, GP surrogates have been used in yield estimation of microwave filters [43], in modeling high-speed interconnect S-parameter curves [40], and in many other high-dimensional modeling tasks where data are sparse but smoothness or structural priors can be assumed. We will cite specific use-cases and results in later sections. Figures illustrating GP behavior (mean and variance) can be found in open-source tutorials and publications. For example, Figure 2.9 of Nooshin et al. for a visualization of GP predictions using RBF vs. periodic kernels, or the GPyTorch documentation for examples of GP regression on sinusoidal data (Figure 2.10). These visualizations help build intuition on how the GP's confidence intervals tighten near training points and how kernel choice affects the learned function characteristics.

In conclusion, the theory presented in this chapter – covering process variation, EM corner concepts,

EDA EM flows, surrogate modeling, and GPR – lays the groundwork for the methodology we develop. With this theoretical understanding, we can now proceed to describe how we implement the GPR-based EM corner modeling flow in practice, and how these concepts come together to yield an effective solution to modeling electromagnetic behavior under packaging process variations.

3

Methodology

This chapter presents the specific implementation of the proposed EM corner modeling methodology. Building on the concepts introduced in the previous chapter, we describe how the flow is realized using commercial EDA tools. The package layout is parameterized in Ansys HFSS 3D Layout, and systematic sampling and surrogate modeling are carried out within the OptiSLang environment. In addition, integration with the GPyTorch framework is introduced to enable Gaussian Process Regression. Together, these elements form a workflow that connects parameterized EM simulation, machine-learning-based surrogate modeling, and variation analysis. The following sections detail each stage of the flow, highlighting the role of the individual tools and their interaction within the overall methodology.

3.1. EDA flows

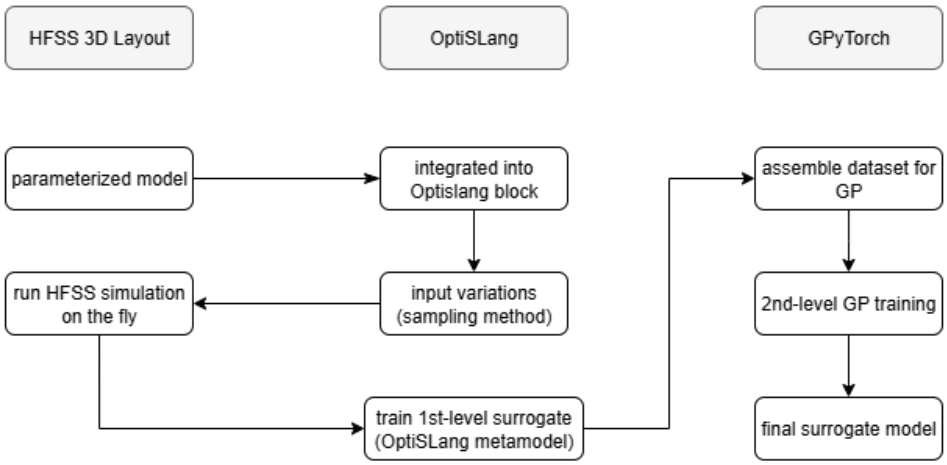


Figure 3.1: EDA flow utilized in this thesis

To address process variation at the package level, this research proposes a structured EDA flow that enables early-stage electromagnetic (EM) corner extraction through systematic parameterized modeling. Process variation in advanced packaging has a significant impact on circuit performance, yet traditional back-end EM simulation struggles to efficiently capture the full range of geometric perturbations. The proposed method introduces a scalable parametric framework that encodes physical uncertainties into simulation-ready models, allowing meaningful EM behavior to be predicted without relying on exhaustive full-process Monte Carlo analysis.

In Figure 3.1, the flow begins with a parameterized package-level EM model that encodes the key geometric and material variables subject to process variation. From this model, input variants are generated using a space-filling Latin Hypercube Sampling Method so that the design space is covered

efficiently with a limited number of simulations. Each sample is evaluated by the EM solver to obtain the required responses (for example S-parameters anf far-field antenna gaintotal), which serve as approximate training targets when high-fidelity runs are too costly. The resulting input–output pairs are used to train a Gaussian Process surrogate in GPyTorch, which provides accurate prediction with calibrated uncertainty while remaining scalable on modern hardware. The trained surrogate then acts as a fast predictor that maps process perturbations to EM behavior and supports early "corner" extraction at the package level. To generalize, here a two-level surrogate is used: OptiSLang trains a first-level metamodel from HFSS DOE data; its outputs are then used to train a second-level GP (GPyTorch) that serves as the final predictor.

3.2. Introduction of Software Tools

To effectively realize and validate the proposed workflow, this research utilizes several key software tools, each chosen for their capabilities relevant to different stages of the design and analysis process. Ansys HFSS 3D Layout was selected due to its advanced capabilities in performing accurate and reliable three-dimensional electromagnetic (EM) simulations and its flexibility in parametric model construction. Ansys OptiSLang, a robust optimization and parameter exploration tool, was employed to systematically handle complex parameter spaces and optimize simulation efficiency through sophisticated sampling techniques. Additionally, the machine learning modeling was implemented using the Gpytorch framework, chosen for its capability in constructing efficient Gaussian process-based predictive models. The machine learning models were developed and trained within the Visual Studio Code integrated development environment (IDE), facilitating efficient coding, debugging, and integration into the broader workflow.

Since Visual Studio Code is a widely used programming tool, it will not be detailed here. Ansys HFSS 3D layout and Ansys Optislang will be introduced separately in this chapter.

3.2.1. HFSS 3D layout

Ansys HFSS 3D Layout is an industry-standard simulation tool for full-wave electromagnetic field analysis. It employs the finite element method (FEM) to solve Maxwell's equations in complex 3D structures, offering high accuracy and reliability. In this study, HFSS 3D Layout was used to construct a parameterized model of the Antenna in Package(AiP) geometry. Key features include:

1. Parametric Modeling Support: HFSS 3D Layout allows users to define geometric and material parameters as variables, enabling automated sweeps and model adjustments across high-dimensional design spaces.
2. Full-Wave EM Simulation: The FEM-based solver ensures high accuracy in computing electromagnetic field distributions and S-parameters across a wide frequency range.
3. Integrated Workflow Environment: HFSS provides a consistent platform for geometry creation, meshing, simulation execution, and post-processing, supporting efficient iteration during optimization and learning phases. [23]

3.2.2. Optislang

Ansys OptiSLang is an advanced tool for design exploration, optimization, and meta-modeling. It was chosen for its capability to handle high-dimensional parameter spaces and automate simulation workflows. Key features include:

1. Automation of External Tools: It provides seamless integration with third-party solvers such as HFSS, allowing parameter combinations to be automatically simulated without manual intervention.
2. Sampling and Design of Experiments (DoE): OptiSLang supports advanced sampling techniques such as Space-Filling Latin Hypercube Sampling, enabling efficient coverage of large parameter spaces with minimal simulation points.
3. Metamodel Construction and Evaluation: The tool supports multiple metamodeling techniques—such as SVR, kriging, DIM-GP and DFFN. It offers quantitative metrics to evaluate and select the most accurate surrogate models. [36]

Together, these tools enabled the construction of a scalable and automated EDA workflow capable of translating package-level process variations into physically meaningful EM corner models.

3.3. Parameterized Model Construction

In the design of advanced electronic packages, small changes in geometry, such as via dimensions, layer thicknesses, or antenna shapes, can lead to significant variations in electromagnetic (EM) behavior. To evaluate and control these effects efficiently, this research adopts a parameterized modeling approach, where key structural features are encoded as adjustable variables. From a higher-level perspective, the core idea is to replace static geometries with flexible, variable-driven structures. This enables engineers to efficiently evaluate "what-if" scenarios, perform sensitivity analysis, and integrate statistical or machine learning models without the need to manually rebuild simulation models.

This modeling concept is widely applicable to a range of EM-related problems such as connector design, PCB stack-up optimization, and antenna packaging, where physical geometry plays a critical role. In this study, it is applied to simulate and generalize the impact of process variations at the package level, enabling the construction of scalable EM corner models in a data-efficient manner.

A cutoff of one Antenna in Package(AiP) model [45][46][47] is used in this research (Figure 3.2), which represents a package structure with embedded antennas and routing layers. With Figure 3.3, the detailed structure of the AiP model can be listed as below:

1. two layers of ADL embedded in Mold Compound and HL972LF respectively. The ADL layer has a feature of an array of small squares arranged 6*6.
2. one layer of antenna connected with feed layer through 4 vias. The slot antenna is embedded in HL972LF.
3. feed layer is connected with the redistribution layer (RDL) through 2 vias. RDL layer consist of a differential transmission line. All of the above layer in this item are embedded in Die_rdl material.
4. below the above layers are the solder ball array surrounded by vacuum with a sheet of backplane at the most bottom.
5. all the metals including ADL, antenna, via, feed, transmission line and solder balls are made of copper.

The parameterized model was implemented in Ansys HFSS 3D Layout, which provides robust support for variable-based geometry and material control.

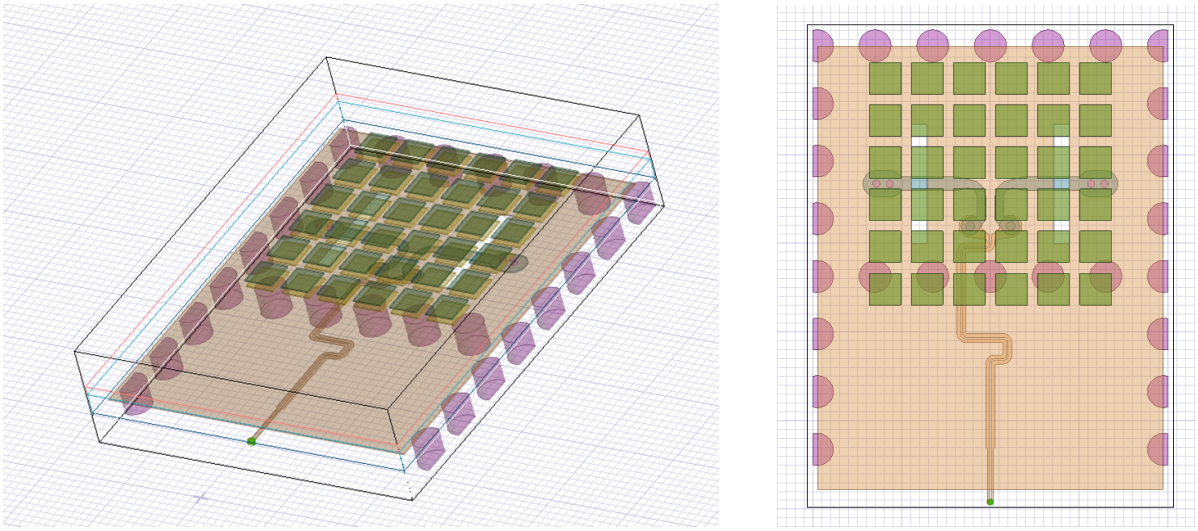
A total of 32 geometrical parameters were identified and incorporated into the model, including via radius, slot antenna dimensions, and the position and size of the artificial dielectric layer (ADL). These parameters were carefully selected based on their physical relevance and observed sensitivity to EM performance within the package structure. The parametric model was constructed in the HFSS 3D Layout environment and integrated with optimization tools to enable automated simulation across the defined design space.

By systematically varying combinations of these parameters, the proposed flow effectively captures the potential fluctuation space of manufacturing-induced variations at the package level. This approach not only improves simulation efficiency and reduces computational cost but also provides a solid foundation for subsequent sensitivity analysis and machine learning–based prediction. The resulting model supports a generalized EM corner construction strategy applicable across a wide range of design scenarios.

Rather than listing all parameters, a representative view of the model is shown below to illustrate the design space complexity and the granularity of control enabled by parameterization(Figure 3.4.

Each parameter was implemented as a named variable in HFSS 3D layout, allowing for automated sweep, batch simulation, and integration with external tools such as Ansys OptiSLang. This framework significantly reduces computational cost by allowing efficient exploration of the design space and supports the construction of accurate surrogate models and machine learning–based predictors.

In light of the temporal and resource limitations imposed on this experiment, a total of five parameters were selected to proceed to the subsequent flow of simualtion. In Figure 3.5, the parameters are



(a) 3D View

(b) Top View

(c) Front View

(d) Right View

Figure 3.2: The structure of the antenna cutoff with different views

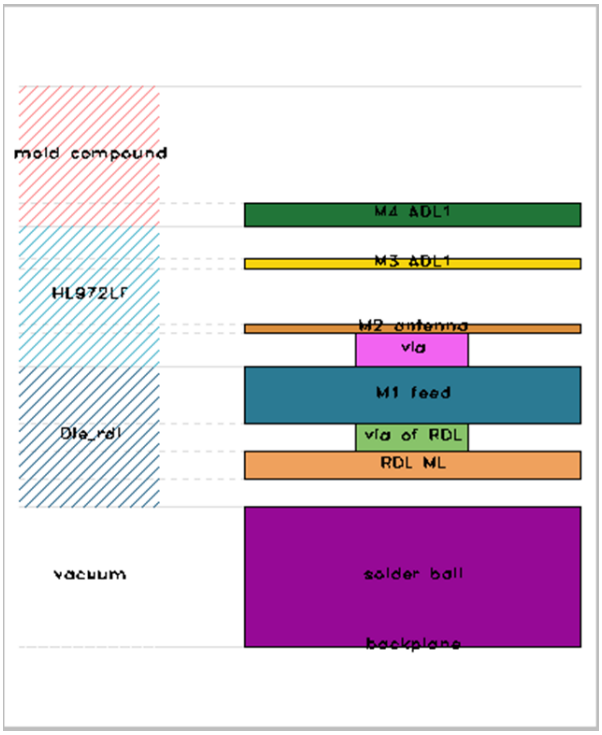


Figure 3.3: the layer structure of the antenna cutoff model

Name	Value	Unit	Evaluated V...	Type
dx	0		0	Design
dy	0		0	Design
dx_M1_l	0		0	Design
dx_M1_r	0		0	Design
dx_center_r	0		0	Design
dy_center_r	0		0	Design
dx_center_l	0		0	Design
dy_center_l	0		0	Design
width_M4	355	um	355um	Design
height_M4	355	um	355um	Design
width_M3	355	um	355um	Design
height_M3	355	um	355um	Design
dx_M4_gap	0		0	Design
dy_M4_gap	0		0	Design
dx_M4_center	0		0	Design
dy_M4_center	0		0	Design
dx_M2_center	0		0	Design
dy_M2_center	0		0	Design
dx_M3_gap	0		0	Design
dx_M3_center	0		0	Design
dy_M3_gap	0		0	Design
dy_M3_center	0		0	Design
dx_via_gap	0		0	Design
via_r	40	u	40u	Design
viardl_r	50	u	50u	Design

(a) part 1

Name	Value	Unit	Evaluated V...
$mold_thick	108	um	108um
$ADL_thick	17.5	um	17.5um
$solderball_thick	310	um	310um
$dierdl_thick	43	um	43um
$HL972LF_thick	255	um	255um
$antenna_thick	17.5	um	17.5um
$via_thick	60	um	60um

(b) part 2

Figure 3.4: Information of the key parameters set in HFSS 3D layout

meticulously delineated. dx and dy are the length and height increase of the slot respectively with slot center point fixed. dx_M1_l and dx_M1_r are the displacement of the two branch of feed in the positive direction (right) while the shape remains the same. via_r is the radius of the vias that connect the antenna and feed.

Ultimately, the parameterized model serves as a cornerstone for the entire methodology, enabling systematic EM simulation under process variation and supporting the generalization of results to diverse packaging scenarios.

3.4. Simulation Workflow in OptiSLang

To support the construction of a generalized EM corner modeling framework, the simulation flow in this study was designed to efficiently explore a high-dimensional design space while minimizing the total number of full-wave simulations. This was achieved through an automated workflow built on Ansys OptiSLang, which supports advanced sampling strategies, surrogate modeling, and optimization techniques. Besides the realization of the workflow, two other components are discussed in this section: the choice of sampling strategy and the comparison of metamodel approximation methods.

3.4.1. Workflow Implementation

The surrogate modeling pipeline was implemented using Ansys OptiSLang's Adaptive Metamodel of Optimal Prognosis (AMOP) framework. In this workflow, the full-wave electromagnetic simulation in Ansys Electronics Desktop (EDT) (HFSS 3D Layout) is integrated as a subsystem within OptiSLang's project flow. In Figure 3.6, the AMOP module (green block) encapsulates the HFSS simulation and manages the design of experiments and surrogate model generation. As shown above, the HFSS model (via Ansys EDT integration) receives design parameters as inputs and produces electromagnetic response data as outputs. The AMOP process automatically orchestrates the sampling of input parameters, execution of HFSS simulations, and construction of a metamodel (surrogate) – labeled "MOP" in the diagram – which represents the EM behavior under variation. This adaptive loop continues until a satisfactory surrogate accuracy is achieved, using Coefficient of Prognosis (CoP) as a convergence metric. The overall goal of this automated workflow is to efficiently build a surrogate model that predicts

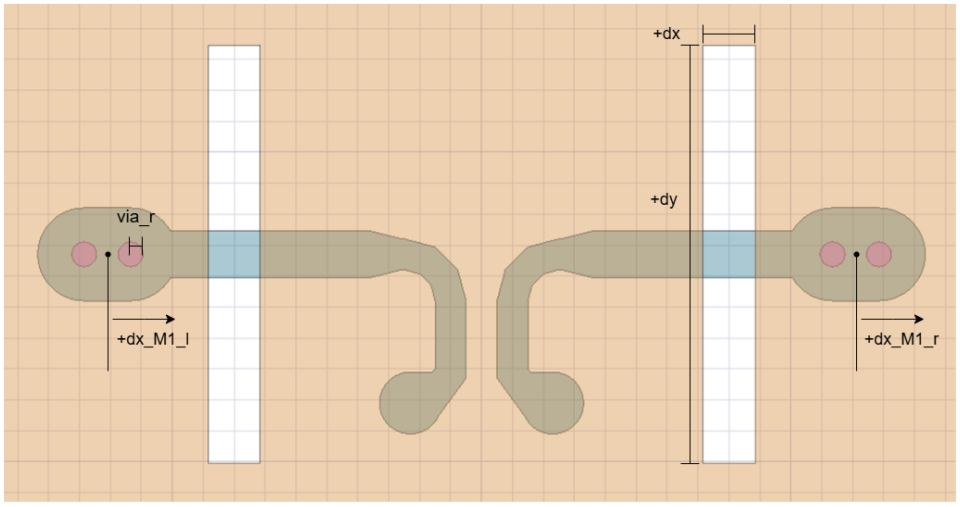


Figure 3.5: parameters used in the following experiment

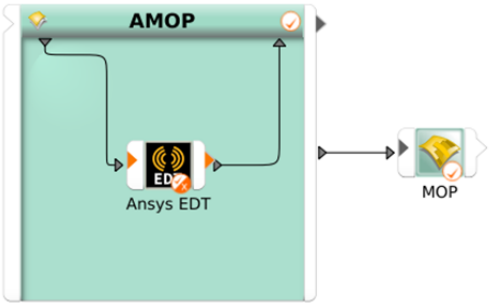


Figure 3.6: Overview of the OptiSLang workflow for surrogate modeling. Ansys EDT integrated in AMOP followed by a MOP block.

electromagnetic responses under package-level process variations, without requiring a full HFSS solve for each new design point.

The full-wave simulation of EM model is linked to OptiSLang through Ansys EDT integration. OptiSLang's interface to HFSS was configured by loading the parametric HFSS 3D Layout project (.aedt file) into an Electronics Desktop block. Figure 3.7 shows the Ansys EDT module setup: the left panel lists the input parameters (automatically imported from the HFSS design) and their current values, while the right panel displays the output responses extracted from HFSS. Each input (e.g. geometric offsets and via dimensions) is mapped to a corresponding parameter in the HFSS model. Similarly, critical EM performance metrics are defined as outputs; in this study they included S-parameter values (e.g. S11 real and imaginary parts across frequency points) and antenna radiation characteristics (e.g. peak gain and gain patterns at key angles). For example, gain_max (the maximum realized gain) and sampled values of the radiation pattern (gaintotal_Phi_'0deg' and '90deg') were specified as responses. Through this integration, OptiSLang can automatically execute HFSS simulations for any sampled combination of input parameters and retrieve the resulting EM performance metrics. The tight coupling between OptiSLang and HFSS (AEDT) ensures that the entire DOE runs seamlessly without manual intervention, enabling efficient data generation for surrogate modeling.

Subsequent to the preparation of the aedt setting, the focus shall be directed towards the AMOP setting. A crucial first step of AMOP configuration was the parameter definition, where key geometrical variables of the package design were identified and their variation ranges specified. Five primary design parameters (e.g. dx, dy, dx_M1_r, dx_M1_l, and via_r) were defined as continuous input variables

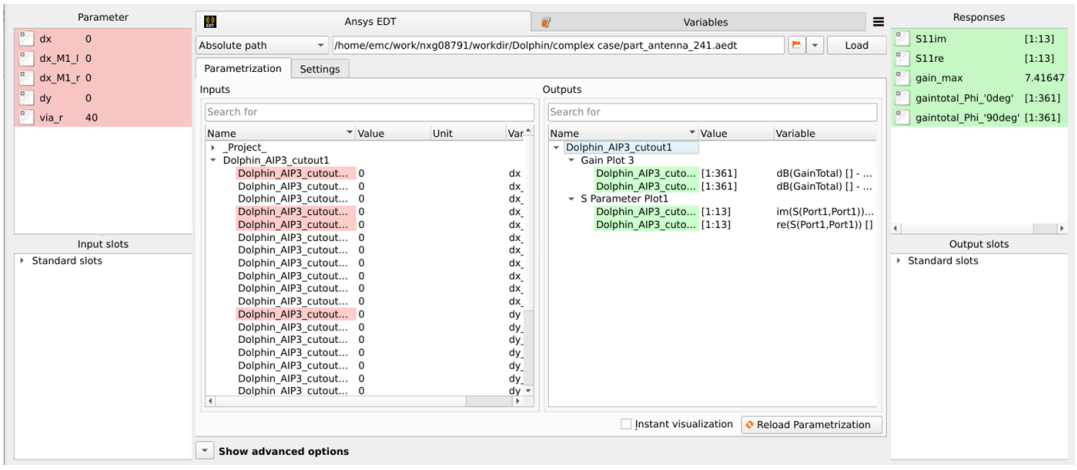


Figure 3.7: Integration of Ansys Electronics Desktop (HFSS) with OptiSLang. Input parameter and output response registration

	Name	Parameter type	Reference value	Constant	Value type	Resolution	Range		Range plot
1	dx	Optimization	0	☐	REAL	Continuous	-17	17	
2	dy	Optimization	0	☐	REAL	Continuous	-135	135	
3	dx_M1_r	Optimization	0	☐	REAL	Continuous	-100	100	
4	dx_M1_l	Optimization	0	☐	REAL	Continuous	-100	100	
5	via_r	Optimization	40	☐	REAL	Continuous	36	44	

Figure 3.8: Definition of design parameters and their ranges in OptiSLang

in OptiSLang. Each parameter was assigned a realistic range based on manufacturing tolerances (for instance, slot dimension offsets and via radius were allowed to vary within ±10% limits). Figure 3.8 shows the parameters table in OptiSLang, listing each variable's name, type (all treated as optimization inputs), and value range. By enumerating these ranges, the workflow establishes the design space to be explored. These parameters are passed into the HFSS model, enabling parametric simulation of the RF structure under different process variation scenarios. The clear definition of inputs ensures that the surrogate model will be trained on a well-bounded and physically meaningful domain of variation.

Space-filling Latin Hypercube Sampling (LHS) is selected for the design of experiments. With the parametric HFSS model in place, an appropriate sampling strategy was employed to explore the input design space. We chose a Space-Filling Latin Hypercube Sampling (LHS) scheme for the initial design of experiments within OptiSLang's AMOP settings. As shown in Figure 3.9, the sampling method is set to Space-filling Latin Hypercube, which ensures a well-distributed coverage of the multi-dimensional parameter space. This approach avoids clumping of sample points and reduces the chance of leaving large voids in the design space. Key parameters of the DOE were configured: for instance, an initial batch of LHS points was generated (on the order of a few dozen simulations) to broadly cover all variable ranges. The adaptive refinement option was enabled (via the "Advanced settings"), meaning the workflow would add additional samples in iterations if needed. The AMOP module monitors the surrogate model's accuracy (using CoP as an indicator) and triggers refinement loops by adding, say, 10 new LHS points per iteration until the improvement stagnates or a target accuracy (e.g. CoP ≈ 0.99) is reached. This adaptive sampling balances efficiency with thoroughness – it starts with a space-filling global exploration and then focuses computational effort where the surrogate needs improvement. By using LHS, the workflow leverages a space-filling DOE that is well-suited for high-dimensional problems and unknown response landscapes, which is essential for building a reliable metamodel of the HFSS responses. Further reseadrch on Space Filling LHS sampling methods will be discusssed in next section.

Here, the Genetic Aggregation Response Surface (GARS) model is activated. In configuring the surrogate, we leveraged OptiSLang's AI Toolkit, which offers a variety of regression modeling algorithms. Figure 3.10 shows the selection of metamodel types available, including polynomial response surfaces, Moving Least Squares (MLS), Kriging (Gaussian Process), Support Vector Regression (SVR), Genetic Aggregation

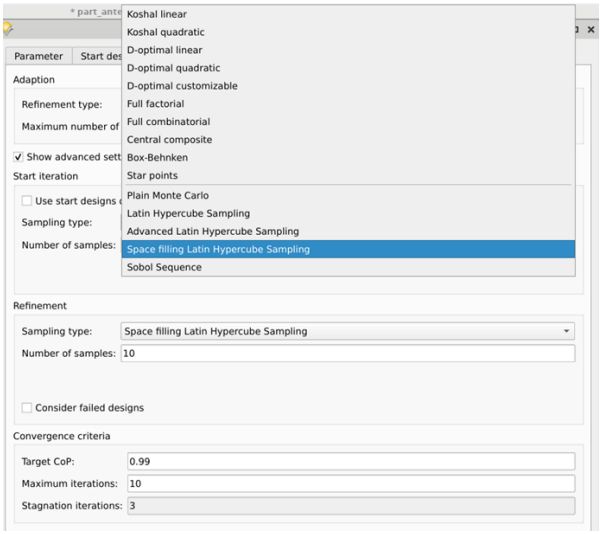


Figure 3.9: Configuration of the sampling strategy in the AMOP settings

Response Surface (GARS), Deep Feedforward Neural Network (DFNN), and Deep Infinite Mixture Gaussian Process (DIM-GP). We activated the more advanced modeling techniques for this study. In particular, the GARS model was used in many experiments due to its robust performance – GARS is an ensemble approach that uses a genetic algorithm to blend multiple basis functions, achieving high accuracy on complex responses. Likewise, the DIM-GP model (a multi-expert Gaussian Process method) was explored for its ability to capture intricate, non-linear patterns in the data and predict vectors such as signal. Both GARS and DIM-GP are state-of-the-art surrogate models introduced in recent versions of OptiSLang, making them well-suited for modeling the highly non-linear electromagnetic behavior in our application. For each model type, OptiSLang automatically trains a surrogate on the collected simulation data and evaluates its predictive accuracy. The Adaptive MOP process can compare these models based on the CoP or validation errors and identify the best-performing surrogate (the "metamodel of optimal prognosis"). In our implementation, final surrogate experiments were conducted using the GARS and DIM-GP models, reflecting both the top performer and a GP-based approach compatible with later ML integration. By trying multiple metamodels, we ensured that the chosen surrogate would be sufficiently accurate and generalizable for predicting the RF responses under process variations. Detailed experiment will be in next section.

As the workflow executes, OptiSLang maintains a Results Design table logging each simulation and its outcomes. A portion of that table is depicted in Figure 3.11, where each row (indexed by design ID) represents a particular combination of input parameters generated by the LHS-based DOE. The inputs (dx, dy, dx_M1_r, dx_M1_l, and via_r) vary across the sampled designs, spanning the specified ranges. For each design, the HFSS simulation returns the requested outputs: here we record complex S-parameter values (S11re and S11im over 13 frequency points), the maximum gain ($gain\_max$), and sampled radiation pattern values ($gaintotal\_Phi0$ and 90 over 361 angles). Most runs completed successfully (Status = "Succeeded"), providing a rich dataset of input-output pairs. These simulation results form the training data for the surrogate models. Once the desired number of samples was collected (ensuring broad coverage and meeting accuracy criteria), the AMOP module built the final surrogate metamodel. This surrogate can predict outcomes (such as gain or S-parameters) for new sets of input parameters nearly instantaneously, which is invaluable for rapid what-if analyses and understanding process sensitivities. In summary, the implemented OptiSLang workflow — from parameter definition through automated sampling and model training — enabled the construction of high-fidelity surrogate models for package-level EM response prediction under manufacturing variations.

3.4.2. Space Filling Latin Hypercube Sampling (LHS) method

When dealing with a large number of design variables, the efficiency and coverage of the sampling strategy become critical to both the quality and cost of the overall modeling process.

Property	Value
CoP tolerance	
Transformation	
Models	
Polynomials	
Use	☐ False
Order	2
Coefficient factor	2.00
Moving least squares	
Use	☐ False
Order	2
Coefficient factor	8.00
Kriging	
Use	☐ False
Anisotropic	☐ False
Coefficient factor	8.00
Genetic Aggregation Response Surface	
Use	☑ True
Support Vector Regression	
Use	☐ False
Deep Feed Forward Network	
Use	☐ False
Deep Infinite Mixture Gaussian Process (DIM-GP)	
Use	☐ False

Figure 3.10: Metamodel method selection

AMOP (on apcw103-10-59-174-180.nxdi.ie-awsc1.nxp.com)

Parameter | Start designs | Criteria | Adaption | MOP | Other | Result designs

	Id	Feasible	Duplicates	Status	dx	dx_M1_l	dx_M1_r	dy	via_r	S11im	S11re	gain_max	gaintotal_Phi_'0deg'	gaintotal_Phi_'90deg'
1	0.1	true		Succeeded	-11.22	58	70	-24.3	42.8	[1:13]	[1:13]	7.21734	[1:361]	[1:361]
2	0.2	true		Succeeded	-12.58	-14	26	-2.7	40.56	[1:13]	[1:13]	7.13106	[1:361]	[1:361]
3	0.3	true		Succeeded	-0.34	70	-62	-121.5	40.4	[1:13]	[1:13]	7.42727	[1:361]	[1:361]
4	0.4	true		Succeeded	-1.7	-94	6	83.7	42.96	[1:13]	[1:13]	6.4779	[1:361]	[1:361]
5	0.5	true		Succeeded	-5.78	90	62	105.3	38.16	[1:13]	[1:13]	7.29547	[1:361]	[1:361]
6	0.6	true		Succeeded	16.66	-34	66	-89.1	40.08	[1:13]	[1:13]	7.05793	[1:361]	[1:361]
7	0.7	true		Succeeded	-9.86	14	-10	51.3	38.64	[1:13]	[1:13]	7.41198	[1:361]	[1:361]
8	0.8	true		Succeeded	-16.66	50	-74	78.3	39.6	[1:13]	[1:13]	7.64243	[1:361]	[1:361]
9	0.9	true		Succeeded	5.78	94	-42	-40.5	37.04	[1:13]	[1:13]	7.61564	[1:361]	[1:361]
10	0.10	true		Succeeded	11.22	86	30	-99.9	43.44	[1:13]	[1:13]	7.46123	[1:361]	[1:361]
11	0.11	true		Succeeded	9.18	-58	18	-62.1	39.44	[1:13]	[1:13]	6.93506	[1:361]	[1:361]
12	0.12	true		Succeeded	14.62	10	-82	-8.1	43.12	[1:13]	[1:13]	7.403	[1:361]	[1:361]
13	0.13	true		Succeeded	-15.98	-62	54	24.3	43.28	[1:13]	[1:13]	6.46729	[1:361]	[1:361]
14	0.14	true		Succeeded	-5.1	46	14	-67.5	42.48	[1:13]	[1:13]	7.47282	[1:361]	[1:361]
15	0.15	true		Succeeded	-11.9	-82	-46	126.9	40.24	[1:13]	[1:13]	6.83342	[1:361]	[1:361]
16	0.16	true		Succeeded	-3.74	-50	86	-126.9	41.68	[1:13]	[1:13]	6.58336	[1:361]	[1:361]
17	0.17	true		Succeeded	-7.14	82	-90	35.1	42.16	[1:13]	[1:13]	7.56013	[1:361]	[1:361]
18	0.18	true		Succeeded	12.58	-78	78	72.9	37.52	[1:13]	[1:13]	5.66792	[1:361]	[1:361]
19	0.19	true		Failed	3.06	-98	-86	2.7	38.8					
20	0.20	true		Succeeded	-4.42	-6	-66	116.1	36.24	[1:13]	[1:13]	7.39692	[1:361]	[1:361]
21	0.21	true		Succeeded	10.54	-42	-98	89.1	40.72	[1:13]	[1:13]	7.3466	[1:361]	[1:361]
22	0.22	true		Succeeded	-14.62	78	90	-51.3	38.96	[1:13]	[1:13]	7.05016	[1:361]	[1:361]
23	0.23	true		Failed	-13.94	2	-34	-110.7	42					
24	0.24	true		Succeeded	4.42	38	-26	99.9	43.6	[1:13]	[1:13]	7.59433	[1:361]	[1:361]
25	0.25	true		Succeeded	15.3	-66	-30	40.5	38.32	[1:13]	[1:13]	7.02703	[1:361]	[1:361]

Figure 3.11: Results of the Design of Experiments in OptiSLang (partial list of sampled designs and outcomes). Each row corresponds to a simulated design with its input parameters and key output metrics.

In this research, the Space-Filling Latin Hypercube Sampling (LHS) method was selected as the basis for the design of experiments. Unlike naive random sampling or grid-based sampling, space-filling LHS ensures that each variable's range is well covered while avoiding clustering or under-sampled regions. This balance between uniformity and randomness makes it particularly suitable for building accurate surrogate models with limited samples.

Overview of the LHS Strategy and Uniform Coverage

Latin Hypercube Sampling (LHS) is a stratified random sampling technique designed to ensure that each input variable's range is thoroughly explored while maintaining randomness. It is developed by McKay in 1979 [48]. In an LHS design, the range of each of the d input factors is partitioned into N equally spaced intervals (strata), and one sample is drawn from each interval for every factor. The construction of the system ensures uniform, one-dimensional coverage for each variable. This guarantees that every interval of a factor's range contains exactly one sampled point. As a result, oversampling occurs in some portions while others remain empty [44]. In other words, the marginal distribution of samples for each input is uniform across its domain. This property contrasts with simple Monte Carlo sampling, which can randomly cluster points or leave large gaps in some regions (especially in high dimensions) [49].

While the uniform coverage property is straightforward to explain in one dimension, in higher-dimensional design spaces it is less intuitive to visualize. To address this, we applied Uniform Manifold Approximation and Projection (UMAP) to project the multi-dimensional sampling sets into two dimensions. Figure 3.12 shows that the LHS points maintain a more even spread after dimensionality reduction compared with random sampling, illustrating the improved space-filling behavior of LHS in practice. Obviously, random sampling is more uneven then LHS. With Figure 3.13 it can be more clearly seen that space filling LHS has a uniformly distributed characteristic.

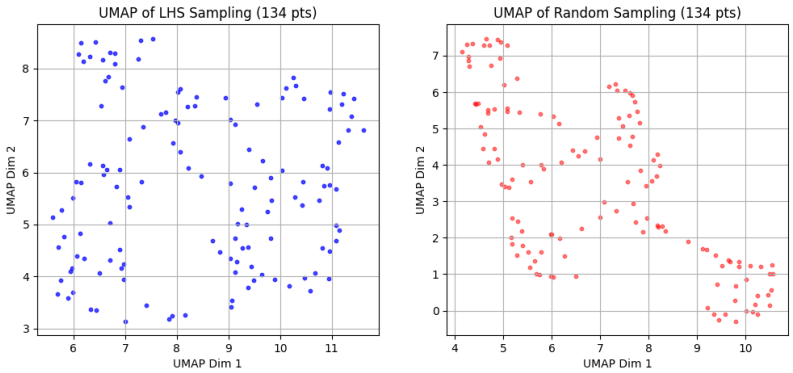


Figure 3.12: Comparison of LHS and random sampling in a high-dimensional design space, visualized in 2D using Uniform Manifold Approximation and Projection (UMAP). The 134 sampling points, later used in experiments, demonstrate that LHS maintains a more uniform spread than random sampling.

To illustrate the LHS layout, consider a two-dimensional example with N points. Each axis (factor X or Y) is divided into N segments, and points are placed such that there is only one point in each row and each column of the resulting $N \times N$ grid. For instance, Figure 3.14 shows an example of a 2D Latin hypercube design: the red dots are arranged so that no two points share the same band along X or Y. This stratification covers each factor's range uniformly, in contrast to naive random sampling which might leave some rows or columns empty. Therefore the method greatly reduces the chance of points clustering together and ensures a broad spread along each axis of the input space.

Mathematically, one way to construct an LHS is as follows: for each dimension $j = 1, \ldots, d$, generate a random permutation π_j of the indices $1, 2, \ldots, N$. Then assign the i-th sample's coordinate in dimension j as:

$$x_{ij} = \frac{\pi_j(i) - U_{ij}}{N}, \quad i = 1, \ldots, N \tag{3.1}$$

where each U_{ij} is an independent uniform random number in $[0, 1]$. This formula ensures that x_{ij}

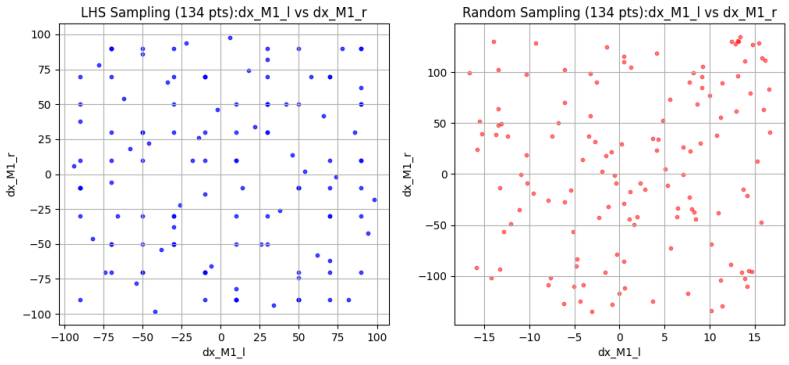


Figure 3.13: Distribution of the same 134 LHS and random samples projected onto the two most sensitive design parameters, dx_M1_l and dx_M1_r. Compared with random sampling, the LHS design achieves a more uniform coverage along both parameter axes.

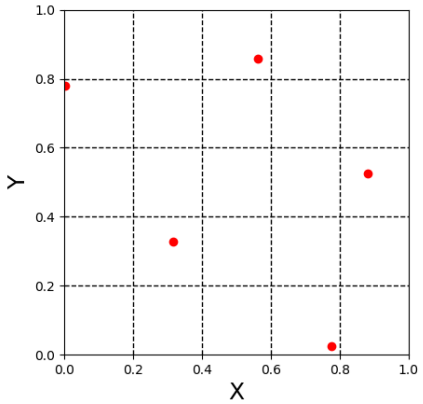


Figure 3.14: Example of a 2D Latin hypercube sample with one point in each row and each column of the grid.

falls in the $[\frac{\pi_j(i)-1}{N},, \frac{\pi_j(i)}{N}]$ interval (the $\pi_j(i)$-th stratum) of factor j. As a result, each of the N strata of every input factor j contains exactly one sample. The randomness of U_{ij} adds stochastic variation within each interval, so the design is not simply a regular grid – it preserves randomness while enforcing coverage of all portions of each range [50]. This balance between enforced uniform coverage in one dimension and randomness within those constraints is a key feature of LHS. In practice, LHS designs are model-independent (not tailored to any specific response surface) and are widely used in computer experiments for their flexibility in sampling an arbitrary number of points without rigid grid structures[25]. Unlike full factorial grids, which grow exponentially with dimensionality and have inflexible sample sizes, an LHS can generate any number of sample points as needed and still maintain stratification [50]. This makes LHS especially scalable when dealing with many design variables.

Space-Filling Criterion and Theoretical Optimality

While a basic LHS guarantees uniform coverage in each individual dimension, it does not automatically optimize the multidimensional space-filling unless additional steps are taken [50]. In higher dimensions, an LHS could still by chance place samples relatively close together (clustering in a local region) and leave other regions of the design space under-sampled [51]. To address this, researchers often enhance LHS with space-filling criteria so that the sample points are as evenly dispersed as possible in the overall d-dimensional domain [52][53]. A common optimality criterion is the maximin distance: the goal is to maximize the minimum pairwise distance between any two sample points [50]. Under the maximin design criterion, the "worst-case" nearest-neighbor distance is made as large as possible, yielding a set of points that are spread out to cover the space broadly. In formula form, one seeks to choose sample locations $x^{(1)}, \ldots, x^{(N)}$ that maximize:

$$\Phi_{\min} = \min_{1 \le i < j \le N} \left\| x^{(i)} - x^{(j)} \right\| \tag{3.2}$$

for an appropriate distance norm (often Euclidean). An LHS that achieves a high $\Phi_{\min}$ ensures no two points are too close, effectively avoiding clustering and large voids in the design [52][52]. Such designs are referred to as space-filling LHS or maximin LHS designs. They retain the Latin hypercube structure (one point per stratum in each dimension) but use algorithmic optimizations (e.g. iterative swaps, simulated annealing, or evolutionary algorithms) to reposition points and improve the spread. For example, one simple improvement procedure is to iteratively swap coordinates of two points along a dimension if the swap increases the minimum distance between all point pairs. This procedure preserves the Latin property while enhancing the space-filling quality, as illustrated in research by Morris and Mitchell (1995) and others who pioneered optimized LHS designs [50].

The "space-filling" LHS is thus theoretically well-founded as a near-optimal strategy for exploration: it strives for multidimensional uniformity. Every region of the input space receives coverage, and no subset of the domain is left with a sparse sampling of points [52][52]. In effect, a space-filling LHS combines the strengths of uniform grids (covering the space systematically) with the advantages of random sampling (statistical robustness and flexibility) [51]. Statistically, LHS belongs to the class of low-discrepancy designs (it can be shown to produce lower discrepancy than purely random sampling) and achieves variance reduction in Monte Carlo estimates by ensuring stratification across the range of each input [54]. In practical terms, this means that for a given number of samples, an LHS tends to provide more information about the response behavior than the same number of purely random samples, because it "fills" the domain more evenly [52][53]. Numerous studies have demonstrated faster convergence and higher accuracy with LHS in tasks like numerical integration and uncertainty quantification: by avoiding redundant clustering of points, LHS can drastically reduce the number of runs needed to achieve a desired level of accuracy compared to standard Monte Carlo sampling [51]. One analysis notes that LHS requires far fewer samples than unstructured sampling to reach the same estimation error in high-dimensional problems [55]. In summary, from a theoretical standpoint, a space-filling LHS is optimal in the sense of providing maximal coverage of the design space for a given sample size – no other sampling strategy can cover all directions and regions as uniformly without sacrificing randomness.

Advantages for Surrogate Modeling and Engineering Simulations

Because of these properties, space-filling LHS has become a de facto standard for design of experiments in complex engineering simulations and surrogate modeling. It is widely used across many engineering domains – including uncertainty quantification, reliability analysis, and global optimization – as a way to sample expensive black-box models efficiently and without bias [56][57]. In the context of building surrogate models (metamodels) for high-fidelity simulators, using an LHS design for the training data offers two key benefits. First, it ensures that the surrogate sees a diverse and representative set of input combinations covering the whole feasible range of each parameter [25]. This reduces the chance that the surrogate will be inaccurate in a region because of missing data; essentially, LHS "spreads out" the training points to avoid leaving blind spots in the input space [52][53]. Second, LHS is flexible and scalable as the number of input variables grows. Traditional factorial designs become impractical with many variables (the sample size grows exponentially), whereas LHS can handle many variables by sampling each independently in a stratified manner [50]. In practice, space-filling Latin hypercube sampling is often considered a suitable choice for high-dimensional design problems. Compared with full-factorial or response-surface based designs, it provides better coverage of the input space across multiple parameters while avoiding the rapid growth in sample size that makes other methods less efficient.

Empirically, the efficacy of LHS in surrogate modeling has been validated in numerous studies. Afzal et al. (2017) observe that "Latin hypercube sampling is [a] widely used design-of-experiment technique to select design points for simulation which are then used to construct a surrogate model." [56] It has been shown to yield surrogate models with better global accuracy for a given sample budget, compared to designs where points might cluster or miss entire regions of the domain. In structural reliability analysis and other uncertainty studies, LHS is similarly praised for improving efficiency: "Latin hypercube sampling is a widely-used method to generate controlled random samples" for such

simulations, significantly reducing the number of runs necessary to achieve accurate estimates [57]. By covering each percentile range of every input, LHS captures extremes and intermediate values without bias, which is especially important when the model responses are highly non-linear or unknown a priori. This makes LHS well-suited for black-box problems like full-wave electromagnetic simulations (e.g. HFSS models), where the underlying response function is complex or analytically intractable [50]. In such cases, one cannot tailor the sample points to specific features of the response (since those features are not known in advance), so a general space-filling approach is theoretically justified to explore all possibilities uniformly.

In the present work, these advantages motivated the choice of a space-filling LHS design for the OptiSLang–HFSS simulation workflow. The LHS method allowed us to automatically generate N sample combinations of input parameters that are well-spread over the allowable range of each parameter. Those N points were then simulated in HFSS to produce training data for the surrogate models. Because space-filling LHS minimizes "holes" in the sampled domain, the resulting surrogate is built on a foundation of data that covers all regions of interest, which improves its generalization quality. Furthermore, using LHS kept the number of required simulations to a minimum: thanks to the space-filling property, even a limited set of samples (which is constrained by computation time) can capture most of the important variation in the response [52][53]. This aligns with best practices in simulation-driven design, where "space-filling designs can be used to sample as much of the design space as possible with the minimum number of evaluations" [54]. Overall, the theoretical optimality of space-filling LHS – in terms of uniform coverage and maximal information gain per sample – is well supported by the literature and has made it a top choice for general-purpose design of experiments in engineering. By employing a space-filling LHS in our methodology, we ensure that the surrogate modeling stage starts with a scientifically sound and widely trusted sampling plan, one that is justified by both theory and extensive usage in high-dimensional engineering problems [56].

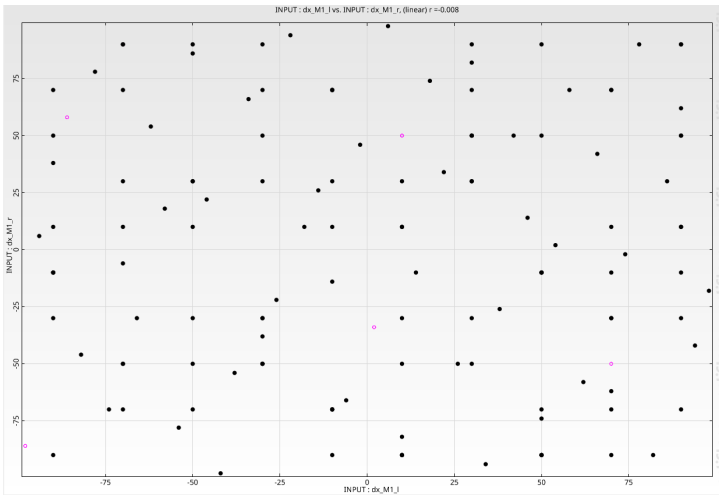


Figure 3.15: 2D design space of a simulation using Space Filling Latin Hypercube Sampling method

3.4.3. Metamodel Methods Comparison

The comparison of metamodel focus on a particular output of interest ($gain\_max$ in this case). All previously defined input parameters were included as potential predictors for the surrogate, and the output response ($gain\_max$) was marked for metamodeling. OptiSLang automatically evaluates the importance of each input (e.g. via sensitivity analysis and the CoP metric) to determine how strongly they influence the output. In this adaptive metamodeling process, the tool can even exclude negligible parameters to simplify the model, though here all variables were kept "selectable" to allow a comprehensive analysis. The AMOP algorithm then constructed a surrogate that maps the five-dimensional input space (the geometry variations) to the output (maximum gain). Internally, multiple modeling techniques can be tried and compared to maximize the accuracy (Prognosis) for the chosen response. Throughout this setup, the integration ensures that any new sample generated by the AMOP is automatically fed into HFSS and the resulting $gain\_max$ is returned, gradually improving the

surrogate's fidelity.

Once simulation data was collected, multiple metamodel (regression model) approximation methods were tested to identify the most suitable one for predicting EM responses under parameter variation. All methods were trained on the same set of LHS sampling points, ensuring a fair and controlled comparison. The methods evaluated include:

1. Support Vector Regression (SVR)
2. Kriging (Gaussian Process)
3. Genetic Aggregation Response Surface(GARS)
4. Deep Feedforward Neural Networks (DFNN)
5. Deep Infinite Mixture Gaussian Process(DIM-GP)

Simple polynomial regression and Moving Least Squares(MLS) models often struggle to achieve sufficient accuracy in highly nonlinear electromagnetic problems, thus not being considered. Each model's performance was assessed using four quantitative criteria: maximum error, mean error, root mean square error (RMSE), and the coefficient of prognosis (COP) on the prediction values. These metrics provided a more comprehensive understanding of each model's predictive behavior, covering both average performance and worst-case deviations.

Prediction	GARS	DIM-GP	DFFN	SVR	Kriging
Max error	0.169824	0.260724	0.350222	0.365523	0.341528
Mean error	0.0390157	0.0420252	0.0545105	0.0794277	0.0539983
RMSE	0.0516965	0.0582642	0.0726734	0.104851	0.0761086
COP	0.985817	0.981984	0.971972	0.941658	0.96926

Table 3.1: Comparison of Prediction Metrics between different metamodel methods

Among all candidates, the GARS method consistently outperformed others across all four metrics, demonstrating the lowest RMSE and maximum error, and the highest COP (Table3.1). COP has such equation:

$$\mathrm{CoP} = 1 - \frac{\sum_{i=1}^{q} \sum_{j \in \mathcal{V}_i} \left(\hat{y}_j^{(-i)} - y_j \right)^2}{\sum_{j=1}^{N} \left(y_j - \bar{y} \right)^2} \tag{3.3}$$

N is the total number of training samples; $\bar{y}$ is the average of the outputs y_j in the training set; - the training points are divided into q subsets $\mathcal{V}_i$ (cross-validation); - $\hat{y}_j^{(-i)}$ denotes the predicted value after excluding the i-th subset; the numerator is the sum of squared cross-validation prediction errors and the denominator is the total variance.

From a theoretical standpoint, GARS would be the most appropriate choice for downstream modeling and generalization.

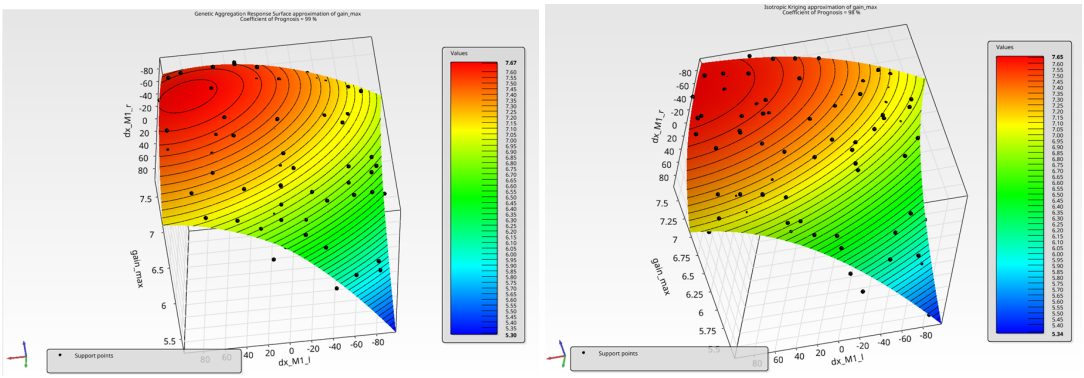
However, the subsequent machine learning stage (Section 4) demanded a surrogate capable of predicting an entire output signal for each new set of input parameters, rather than just single-point values. In this context, a "signal" refers to a continuous electromagnetic response (such as an S-parameter curve across frequency or an antenna gain pattern across angles) which is represented in the simulation by a vector of discrete data points. Among the tested models, only the DIM-GP approach (specifically its signal-output variant) could directly accommodate such vectorized outputs; methods like GARS are inherently single-output predictors, meaning they would require building separate models for each frequency or angle point to reconstruct a full signal. In theory, one could train independent GARS models at each output point and then stitch together their predictions to form the complete response. However, modeling each point in isolation would ignore the inherent correlations among points in the signal, leading to a loss of valuable information and potentially less consistent overall predictions [58]. By contrast, DIM-GP treats the entire output vector jointly, leveraging the correlation structure across

the signal to produce more coherent and physically realistic predictions of the full response. Moreover, DIM-GP was the second-best performer in the benchmarking study, with error metrics only slightly higher than GARS, so adopting it incurred only a minor trade-off in accuracy.

Therefore, DIM-GP was ultimately selected for final deployment as it was uniquely suited to the project's signal-based output requirements while still delivering high predictive fidelity. This choice highlights a practical trade-off often encountered in engineering modeling: the model with the absolute lowest error is not always the optimal choice if it cannot be seamlessly integrated into the required workflow or output format. In such cases, a nearly as accurate method that aligns with the problem's data structure and downstream needs is the more viable and pragmatic option.

addition of 3.4.3

In addition to comparing models trained on the same sampling data, this research also investigated cases where sampling points could not be unified across methods due to tool limitations or algorithm-specific constraints. In such cases, a secondary validation procedure was introduced: the response surfaces of two models were overlaid, and the region of maximum difference between their predictions was identified. New simulation points were then added in these regions using HFSS 3D layout to determine which model more accurately reflected the true EM response.



(a) GARS **(b)** kriging

Figure 3.16: Response Surface of $gain\_max$ generated by different model in Optislang

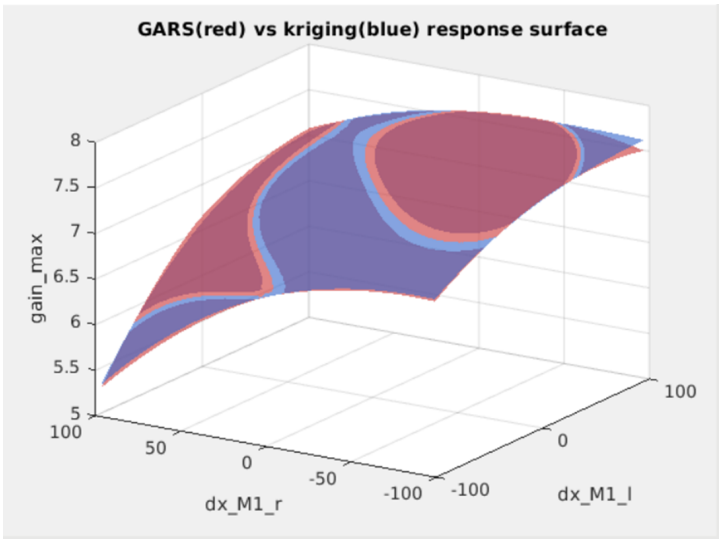


Figure 3.17: overlapping response surface of GARS model and kriging model

The GARS and Kriging predicted $gain\_max$ surfaces are in close agreement overall, nearly coinciding in shape and magnitude. However, the differential response surface (absolute value) reveals five distinct localized regions of discrepancy, with a maximum difference of $\tilde{0}.1163$ in $gain\_max$. Validation at

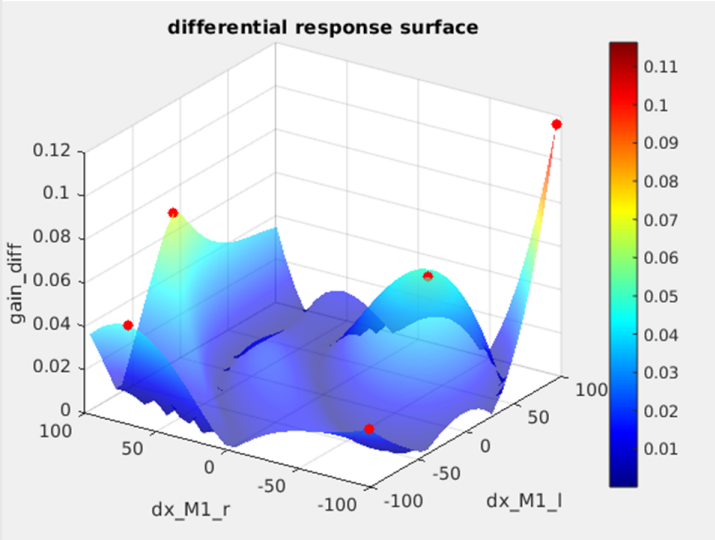


Figure 3.18: absolute differential of GARS and kriging response surface

	dx_M1_l	dx_M1_r	simulation	GARS	kriging	G_error	k_error
1	98	-98	7.6169	7.6271	7.5108	0.134%	-1.393%
2	-10.1379	98	6.6273	6.6195	6.5495	-0.118%	-1.174%
3	43.9310	-43.9310	7.5934	7.6479	7.5972	0.718%	0.05%
4	-98	70.9655	5.7357	5.7736	5.8185	0.661%	1.444%
5	-98	-98	7.0302	6.9117	6.9379	-1.686%	-1.313%

Table 3.2: Summary of simulation results with errors.

these discrepancy "hotspots" using full simulation data confirms that both metamodels remain highly accurate (errors on the order of 0.1-1.7%), though each model is slightly more accurate in different regions. For example, at the largest difference location GARS predicted 7.627 vs. 7.617 actual (only +0.13% error) while Kriging gave 7.511 (-1.39% error); conversely, at another high-discrepancy point GARS had õ.72% error against Kriging's 0.05%. Thus, while both response surfaces align very closely overall, the remaining differences are structured and non-negligible in those specific regions.

This validation strategy establishes an objective framework for assessing the model's generalization capability beyond the training dataset. Furthermore, it emphasizes the significance of uncertainty-aware refinement, particularly in contexts where the surrogate model is deployed in safety-critical or yield-sensitive applications

4

Machine Learning modeling based on Gpytorch

To further enhance the generality and predictive capability of the proposed modeling framework, machine learning (ML) was integrated into the EDA flow. While traditional surrogate models can efficiently approximate known regions of the parameter space, they often struggle to generalize to unseen designs or quantify predictive uncertainty. In contrast, ML methods—particularly those based on probabilistic frameworks—are better suited to capture complex, nonlinear relationships and provide uncertainty-aware predictions, especially under sparse and noisy data conditions common in electromagnetic (EM) simulation tasks.

From a broader engineering perspective, incorporating ML into EM modeling enables the construction of scalable, data-efficient systems that can improve as more data becomes available, while significantly reducing the number of required full-wave simulations. This methodology is not limited to the current packaging scenario but is also applicable to other high-cost simulation domains such as thermal modeling, antenna optimization, or cross-layer signal integrity analysis.

In this study, Gaussian Process (GP) regression was adopted as the core ML model due to its strong theoretical foundation in function approximation and uncertainty quantification. The implementation was based on the Gpytorch framework, which offers modular, scalable tools for building GP models in high-dimensional spaces. Training data was derived from the parameter exploration conducted in OptiSLang, and the GP model was used to learn the mapping between design parameters and EM response curves. The resulting model not only captured the nonlinear structure in the data but also enabled predictive inference beyond the original simulation set, effectively supporting the construction of EM corner models at the package level.

The integration of Gpytorch into the EDA workflow constitutes a key innovation of this research, bridging the gap between computationally expensive EM simulation and scalable predictive design automation.

4.1. Realization of ML model

The ML model in this study is implemented using the Gpytorch framework, which is optimized for scalable and modular Gaussian Process (GP) modeling. The foundational model used is the Exact GP, a Bayesian regression model well-suited for cases where the full covariance structure of the training data can be calculated. Exact GP offers two key advantages: (1) the ability to provide both mean and uncertainty estimates for predictions, and (2) a flexible kernel-based structure that can be adapted to reflect physical behavior (e.g., smoothness, periodicity).

4.1.1. Model Design and Training

In the early stage of development, the ML model was constructed and tested in standalone Python scripts using Visual Studio Code. This phase served two main purposes: (1) to verify that the GP model could accurately learn the relationships between parameter inputs and EM response outputs; and (2) to validate that the signal form of the output (e.g., S-parameter curve) could be handled efficiently by the GP model. Preprocessed simulation data from OptiSLang was used to train the model, and the performance was evaluated using standard regression metrics and prediction plots.

Training Data: The surrogate model was trained on data generated from the parameterized electromagnetic simulations in OptiSLang. In total, 140 design instances (sampled via Latin Hypercube, as described earlier) were simulated with the full-wave solver, and their results were used as training points. Each design instance is defined by a set of input features – in this case, a subset of up to 5 key geometric parameters (selected from the 32 available parameters based on sensitivity analysis) – and the corresponding EM output response curve. For example, one set of outputs consists of the reflection coefficient S11 of the RF package across frequency (with real and imaginary parts), while another set corresponds to the antenna gain as a function of angle (radiation pattern). These response "signals" provide a rich target for learning, as they encapsulate the frequency-dependent or angle-dependent behavior of the system for each design. Before training, the raw simulation data were preprocessed (normalized and aligned) to ensure the GP model could be efficiently trained and that all features shared comparable scales.

GPyTorch Exact GP Model: We implemented an Exact Gaussian Process regression model using the GPyTorch library. GPyTorch provides a modular framework for building GP models by specifying the kernel (covariance function), mean function, and likelihood [59]. In our case, we defined a custom GP model class inheriting from gpytorch.models.ExactGP, with a Gaussian likelihood (assuming homoscedastic noise) and a constant mean function. A constant mean (which was initially set to the average of the training outputs) is a common default choice for GP regression, allowing the model to learn deviations from a baseline level. The core of the model is the covariance kernel, which was carefully designed to reflect the known behavior of the EM responses. We employed a composite kernel combining a Radial Basis Function (RBF) kernel times a periodic kernel for gaintotal prediction; while a Radial Basis Function (RBF) kernel times a RBF kernel for S-parameter prediction. The RBF kernel (squared-exponential covariance) captures the smooth variation of the output with respect to the design parameters – it assumes that designs with similar parameter values produce similar responses, with correlations decaying Gaussianly with distance in the input space. The periodic kernel, on the other hand, accounts for the inherently repeating patterns observed in certain outputs. For instance, the gain as a function of angle is naturally 360°-periodic (or 2π in radians); by including a periodic covariance on the angle input, we encode this prior knowledge so that the model "knows" the response repeats every full rotation. In practice, the kernel was implemented as a product of an RBF kernel (with Automatic Relevance Determination, ARD, allowing a separate lengthscale for each design parameter) and a periodic kernel for the angle or frequency dimension, yielding a covariance $k((\mathbf{x}, \theta), (\mathbf{x}', \theta')) = k_{\text{design}}(\mathbf{x}, \mathbf{x}') \times k_{\text{periodic}}(\theta, \theta')$. This structure enables the model to learn complex functions that vary smoothly with design changes and periodically with angle (or with any other cyclic input), effectively handling entire response curves rather than single scalar outputs. As shown in Figure 4.1 and 4.2, with two kernels time each other, the accuracy of prediction can be improved to a big extent. The poor performance of the single-kernel model in Figure 4.1 reflects the limited expressive power of using only an RBF kernel. While the RBF captures smooth correlations in the design space, it cannot account for oscillatory or periodic components in the responses. As a result, the single-kernel GP tends to either underfit the rapid variations or misrepresent the overall trend. By contrast, the composite kernel (RBF × periodic or RBF × RBF) used in Figure 4.2 provides additional flexibility. It models both smooth dependence on design parameters and repeating patterns along the frequency or angle axis. This combination allows the GP to reproduce the detailed shape of the EM responses more accurately. All kernel hyperparameters – the RBF lengthscales for each input dimension, the periodic kernel lengthscale and period, the overall output scale (signal variance), and a small noise variance – were initialized to reasonable values and left trainable for optimization.

Training Procedure: The GP surrogate was trained by maximizing the log marginal likelihood of the training data. We used a stochastic gradient descent approach (Adam optimizer) to tune the kernel hyperparameters and the likelihood's noise parameter. The training loop, analogous to training a neural

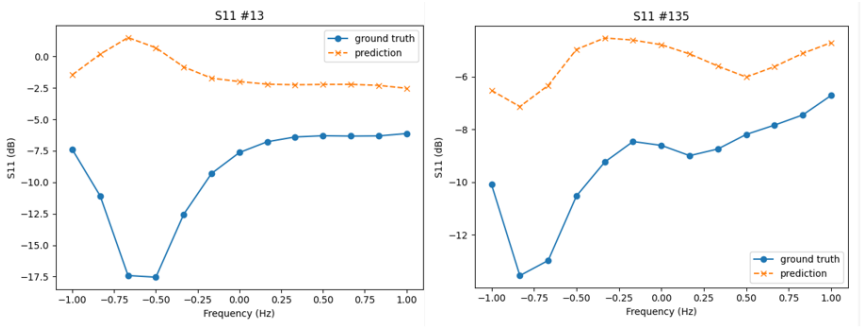


Figure 4.1: Prediction on the training data with ExactGP having one kernel (Poor).

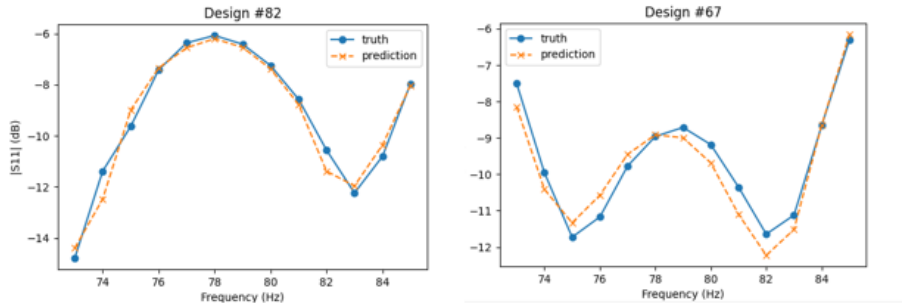


Figure 4.2: Prediction on the training data with ExactGP having two kernels (Improved).

network, iteratively updated the parameters to reduce the negative log-likelihood (NLL). We ran the optimization for up to 700–800 epochs (iterations over the full training set), monitoring the loss to ensure convergence. To prevent overfitting and improve numerical stability, a small noise term (jitter) of about 10^{-4} to 10^{-3} was included in the likelihood – this acknowledges that even deterministic simulation data can have tiny numerical noise, and it helps condition the covariance matrix inversion. During training, the NLL steadily decreased, indicating the model was improving its fit to the data.

Figure 4.3 illustrates the Negative Log Likelihood (NLL) values as a function of the number of epochs for two different learning rates: lr=0.1 and lr=0.2. The blue line represents the NLL values for a learning rate of 0.1, while the red line corresponds to a learning rate of 0.2. The epochs range from 25 to 750, with each data point corresponding to a specific epoch. It can be seen that the NLL values for lr=0.2 (red line) decrease more rapidly in the initial epochs compared to lr=0.1 (blue line), indicating faster convergence. However, the NLL values for lr=0.2 also show more fluctuations, suggesting potential instability in the learning process. Both learning rates eventually reach similar NLL values around -1.5, but the path to convergence differs. The lr=0.2 model achieves lower NLL values earlier but experiences more fluctuations, while the lr=0.1 model converges more steadily. So in later practice we chose the more conservative rate (lr=0.1) for the final model to ensure stability. The choice of learning rate significantly affects the training process. A higher learning rate (lr=0.2) can lead to faster convergence but may also introduce instability, while a lower learning rate (lr=0.1) provides a more stable but slower convergence. This figure provides valuable insights into the impact of different learning rates on the training process, as measured by NLL. It highlights the trade-off between convergence speed and stability, emphasizing the importance of selecting an appropriate learning rate for optimal model performance.

After confirming the GP surrogate's accuracy in the standalone setup, the final trained model was saved and prepared for integration. In the next subsection of the thesis, the deployment of this model into the OptiSLang workflow via a Python block is discussed, enabling direct surrogate predictions within the design exploration loop. Here we focus on the performance and validation of the GP model itself.

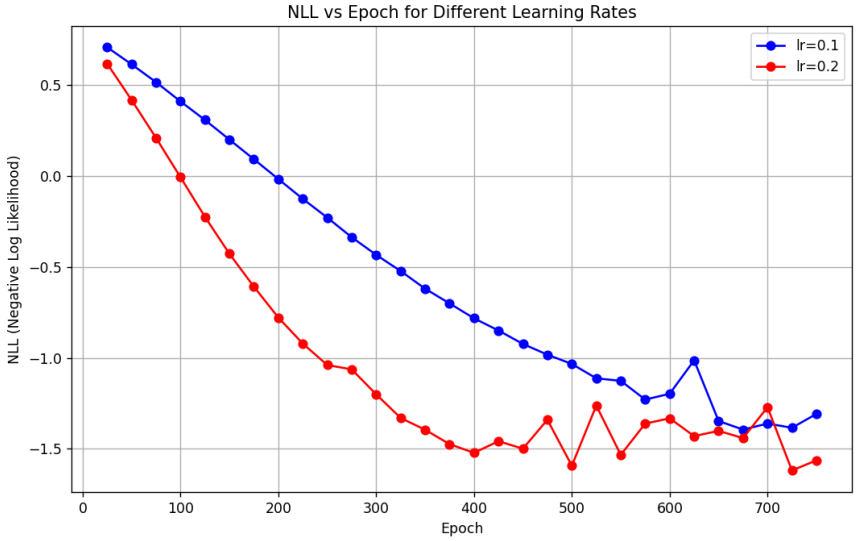


Figure 4.3: NLL vs Epoch for Different Learning Rates

4.2. Surrogate Model Performance

In the final stage, the machine learning model was embedded into the OptiSLang workflow using a custom Python Block. Figure 4.4 demonstrates the whole flow. The integration involved several steps: using the results of AMOP to generate DIM-GP signal surrogate model; then in another new block use a MOP solver to call the model in MOP; the python block followed is to write down the input and output of every inference; Parametric System can feed the input parameters into MOP solver; While Loop is utilized to change the value of parameters in the Parametric System (randomly).

The inference of DIM-GP signal make it possible to evaluate new design points directly through the ML model without invoking HFSS. After that, the recorded data can be used to train the final model. This not only accelerated the evaluation process but also enabled the full pipeline—from parameter definition to EM corner prediction—to be executed within a single automated environment.

This final implementation demonstrates the practical viability of combining physics-based simulation and data-driven learning, bridging the gap between high-fidelity EM modeling and scalable engineering design. Although the final model was based on the DIM-GP structure due to framework constraints (as discussed in Section 3.4.3), the core approach remains applicable to a broader class of GP and ML models.

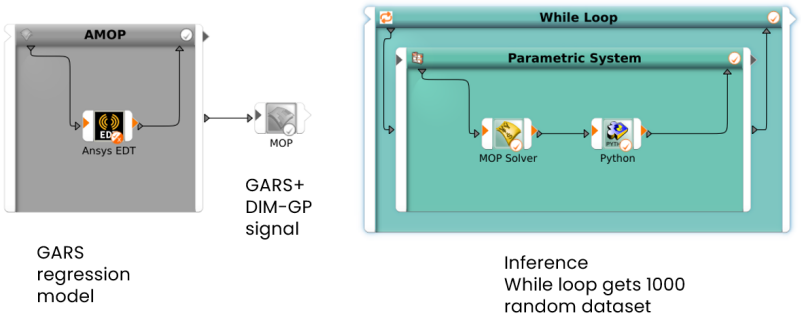


Figure 4.4: OptiSLang workflow with Python block

Evaluation Metrics: To assess the predictive performance of the Gaussian Process surrogate, we evaluated it on a separate test set of 140 unseen design points (these were additional OptiSLang simulation runs not used in training). Four quantitative metrics were used to summarize the errors:

maximum error, mean error, RMSE, and Coefficient of Prognosis (CoP). The maximum error is the worst-case absolute difference between the GP prediction and the true simulation output over all test points – effectively measuring the largest deviation observed (useful for gauging worst-case behavior). The mean error denotes the average absolute prediction error across all test samples, indicating the overall bias or typical discrepancy. The root mean square error (RMSE) provides a single measure of error magnitude that heavily penalizes larger errors (being the square root of the average squared error). Finally, the Coefficient of Prognosis (CoP) is a metric adopted from OptiSLang's validation methodology to quantify predictive accuracy. In simpler terms, CoP reflects the fraction of output variance explained by the model's predictions on unseen data – a CoP of 1 would indicate a perfect surrogate, whereas, for example, 0.95 would indicate that 95% of the variance in the true responses is captured by the model.

Using these metrics, the GP surrogate demonstrated high accuracy in predicting the electromagnetic responses. Table 3.2 already compared several modeling approaches on a similar problem, and the GP-based method achieved a CoP above 0.98 with low errors. In our final evaluations, the Exact GP model showed consistent performance: the RMSE on the test set was on the order of only 5×10^{-2} (in normalized units), and the mean absolute error was around 4×10^{-2}. For context, in the case of antenna gain prediction, this corresponded to roughly 0.8 dB RMSE and about 0.3 dB average error – a very small discrepancy given that the gain values ranged over tens of dB. The maximum error observed for the gain model was approximately 2 dB at a single design (occurring at an extreme corner of the design space), whereas most designs had much lower errors. For the S_{11} predictions, errors were even smaller in absolute terms; the GP could predict the reflection coefficient within a few hundredths of a unit (complex magnitude), which is negligible for practical purposes. The high CoP (typically 0.95–0.99 for the various outputs) confirms that the surrogate explains the vast majority of the variation in the EM responses. These results instill confidence that the GP model has not overfit the training data, but rather generalizes well to new design points.

4.3. Results Analysis

This chapter presents the results of machine learning–based signal modeling using the Gpytorch framework. The goal is to accurately predict electromagnetic responses such as S_{11} and gain patterns based on a limited number of full-wave simulations. All models were trained on datasets generated from OptiSLang simulations and used the DIM-GP structure due to its compatibility with signal-level outputs.

4.3.1. Prediction of S-Parameters and gain pattern

The real and imaginary components of S_{11} and gain patterns at fixed angles (e.g., $\phi = 0$, $\phi = 90$) were modeled using DIM-GP across multiple datasets (140, 500, 1000 points). The output signal was the reflection loss across frequency from 73GHz to 85GHz and gain across θ (elevation) from 0° to 360°.

Table 4.1 presents a comparative analysis of the performance of a machine learning model trained on different dataset sizes when predicting various signal parameters. The performance is measured using the Root Mean Square Error (RMSE) for four key parameters: the real part of the S-parameter (S-re), the imaginary part of the S-parameter (S-im), gain-phi90, and gain-phi0. Additionally, the computational time required for inference on a BSUB cluster with 32 cores is also reported.

The table compares three different training dataset sizes: 140, 500, and 1000. As the training dataset size increases, there is a clear trend of improved prediction accuracy across all four parameters, as indicated by the decreasing RMSE values. For instance, the RMSE for S-re drops from 0.015 with 140 points to 0.005 with 1000 points. Similarly, the RMSE for S-im decreases from 0.0371 to 0.0071, for gain-phi90 from 0.4917 dB to 0.2359 dB, and for gain-phi0 from 0.782 dB to 0.2946 dB.

However, this improvement in prediction accuracy comes at the cost of increased computational time. The inference time on the BSUB cluster rises from 5 minutes for the smallest dataset to 30 hours for the largest dataset. This trade-off between accuracy and computational efficiency is a critical consideration when selecting the appropriate training dataset size for practical applications. Generally, increasing dataset size led to better smoothness and fidelity in signal predictions.

Figure 4.5 and 4.6 shows the prediction versus truth of all four signals in 500 dataset. In this figure, the GP model's predicted response (all the signals) is plotted as a continuous curve, and the true result is

Dataset size	S11-re RMSE	S11-im RMSE	gain-phi90 RMSE	gain-phi0 RMSE	TIME on BSUB -n 32 python
140	0.015	0.0371	0.4917dB	0.782dB	5 min
500	0.006	0.0104	0.2355dB	0.3023dB	50 min
1000	0.005	0.0071	0.2359dB	0.2946dB	30h

Table 4.1: General test on the difference between simulation dataset and DIM-GP inference dataset across different training dataset sizes.

overlaid for comparison. The surrogate predictions (solid lines) are virtually indistinguishable from the ground truth (markers) across the frequency band (or theta band). Choose S11 imaginary for instance, the errors at each frequency point are very small – on the order of 0.01 (1%) or less – which is within the acceptable tolerance for design purposes. Overall, Figure 4.5 and 4.6 highlights that the Exact GP surrogate can faithfully reproduce complex frequency-domain responses, providing both an accurate mean prediction and a quantitative uncertainty estimate.

4.3.2. Impact of Oversampling on sub-range with Different Training Dataset Size

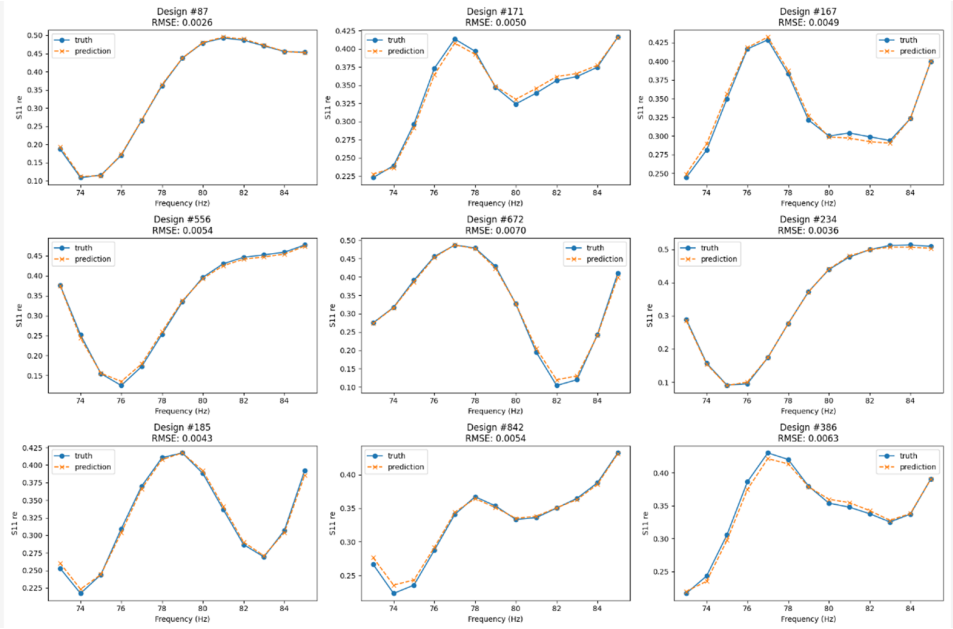
Figure 4.7 demonstrates the impact of oversampling on a particular range, a focal point that engineers frequently examine. In Figure 4.7, the blue line shows the overall RMSE across the full angular range, the red line shows the RMSE restricted to the sensitive angular sub-ranges (0–30° and 330–360°), and the green line shows the RMSE in these sub-ranges when additional oversampling is applied. These angular sectors correspond to the main lobe of the radiation pattern, which antenna engineers typically consider the most critical region, since prediction accuracy in the main lobe directly impacts design and manufacturing performance. The efficacy of oversampling is evident in its capacity to reduce Root Mean Square Error (RMSE) values, particularly in scenarios involving smaller datasets. However, it should be noted that the stability of this effect is not guaranteed. This aligns with physical expectations: the surrogate is most certain where the device behavior was well-sampled, and less certain toward the extrapolation regions. This finding suggests a potential future direction for the rapid and precise prediction of specific signal ranges.

On the other hand, it can be seen that the overall RMSE (blue line) decreases consistently as the dataset size grows, while the sub-range RMSE (red) and the oversampled sub-range RMSE (green) exhibit fluctuations when the dataset size is small. This indicates that performance in localized angular regions is more sensitive to sampling variation. After about 50 samples, all curves converge to stable values, and further improvements become marginal, as the parameter space is already well covered and the model approaches convergence. The larger fluctuations observed in the sub-range RMSE (red and green curves) are mainly due to the smaller effective data coverage in this angular region. Because the RMSE is computed only on a subset of the radiation pattern, the metric is inherently more sensitive to how the training samples are distributed. At small dataset sizes, this limited coverage can lead to irregular accuracy gains as additional points are added. In the case of oversampling, the effect is amplified: while local refinement eventually improves accuracy, at small sample sizes it may cause temporary overfitting to the oversampled region. As the dataset size increases, both effects diminish, and the curves converge more smoothly.

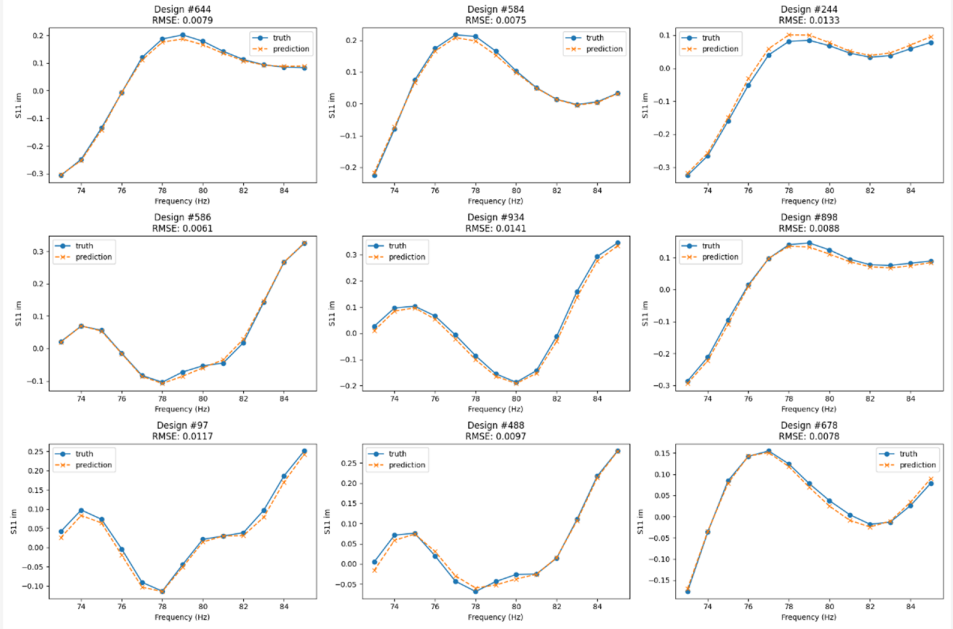
Effect of parameter numbers on convergence

Figure 4.8 illustrates the Root Mean Square Error (RMSE) changes as the number of varied parameters increases for different dataset sizes. The dataset sizes range from 2 to 50. Each curve corresponds to a different number of varied parameters: "Change (1)" refers to varying only the most sensitive parameter dx_M1_l; "Change (12)" means varying dx_M1_l and dx_M1_r; and so on until "Change (12345)" where all five parameters are varied simultaneously. Each line in the chart represents a different change scenario, with distinct colors for clarity. The parameters are listed in decreasing order of sensitivity: dx_M1_l (1), dx_M1_r (2), dy (3), via_r (4), and dx (5).

Across all scenarios, the RMSE decreases as the dataset size increases, indicating improved model performance with larger training sets. The most substantial reductions occur when the dataset size increases from 2 to about 10 samples, after which the improvements become more gradual. In this initial range, adding new points through Latin Hypercube Sampling (LHS) significantly enhances the coverage of the design space and enables the surrogate to capture the dominant trends more accurately. In our

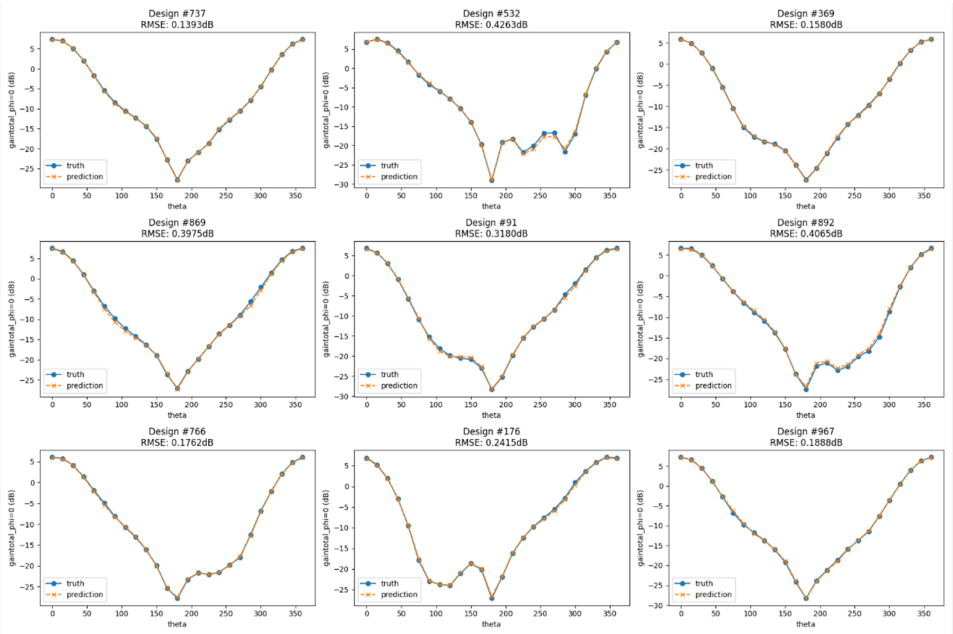


(a) S_{11} real

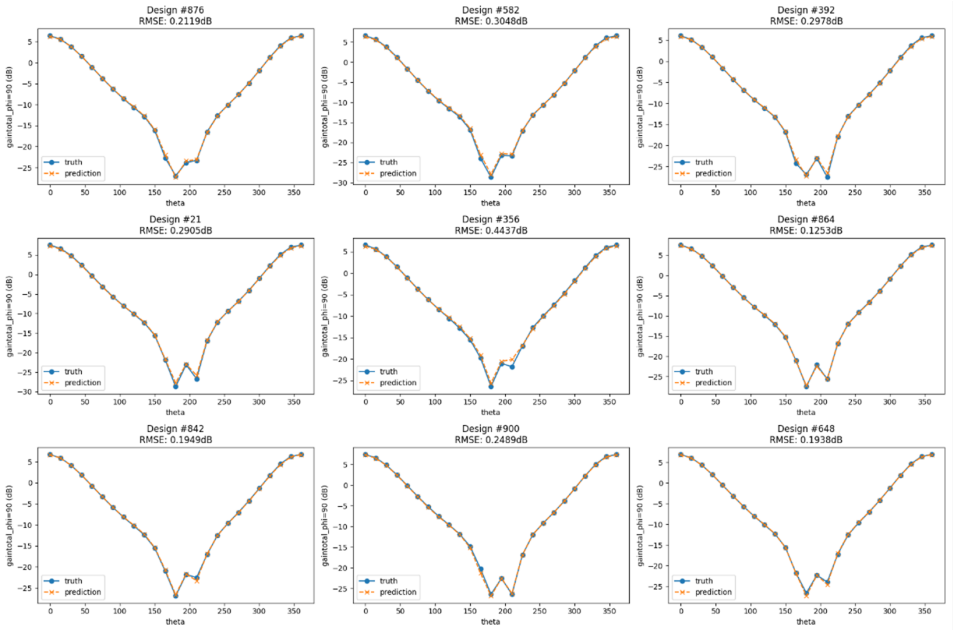


(b) S_{11} imaginary

Figure 4.5: Prediction vs Truth on S_{11} in 500 dataset



(a) gain ($\phi = 0$)



(b) gain ($\phi = 90$)

Figure 4.6: Prediction vs Truth on gain in 500 dataset

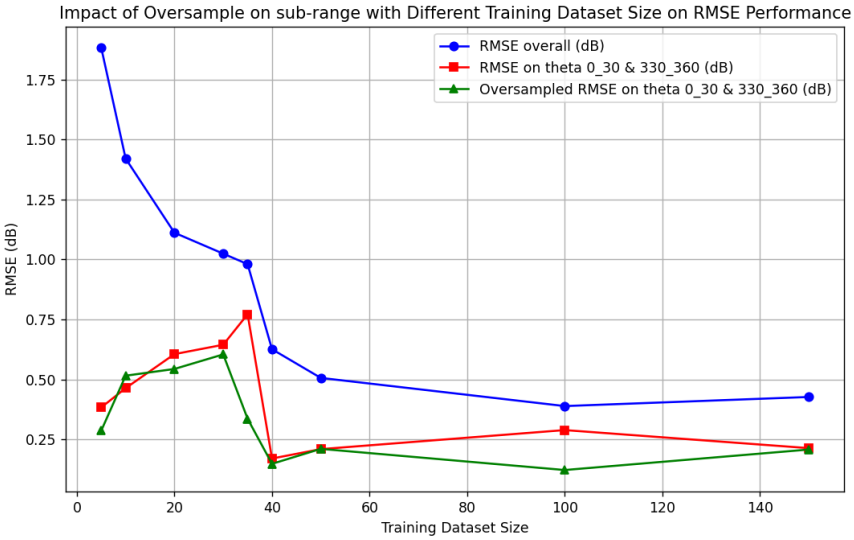


Figure 4.7: Impact of oversampling on RMSE performance with different training dataset sizes. The blue line shows the overall RMSE across the full angular range, the red line shows the RMSE within the main lobe region (0–30° and 330–360°), and the green line shows the RMSE in the same region when additional oversampling is applied.

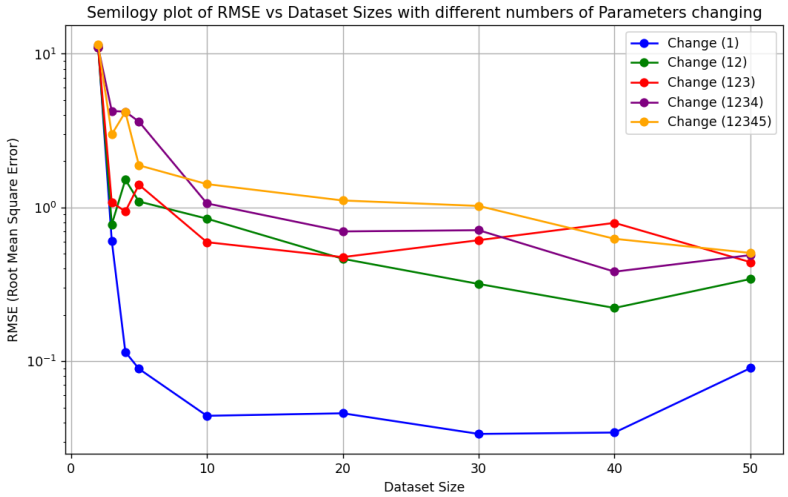


Figure 4.8: Effect of varying the number of design parameters on RMSE across different dataset sizes (log scale on y-axis). "Change (1)" means only the most sensitive parameter dx_M1_l is varied, "Change (12)" means the first two parameters (dx_M1_l and dx_M1_r) are varied, and so on until "Change (12345)" where all five parameters are varied simultaneously.

study, the dataset size was expanded by progressively generating additional LHS points within the same predefined parameter ranges, thereby creating a denser training set as the sample size grew from 2 up to 50. This explains why RMSE values generally decrease with more data: the surrogate becomes increasingly reliable as the sampling more comprehensively represents the underlying electromagnetic behavior.

These results highlight the critical role of even a small number of initial samples in establishing a reliable surrogate. At the same time, varying more parameters (e.g., Change (12345)) consistently leads to higher RMSE values compared to varying fewer parameters (e.g., Change (1)), especially for smaller datasets. The most sensitive parameter (dx_M1_l) has the greatest impact when varied individually, and as additional parameters are included, the RMSE values increase, reflecting the cumulative effect of parameter variations.

Overall, the figure provides valuable insights into the relationship between dataset size, parameter variations, and model performance. It emphasizes the need to carefully select and tune parameters to achieve accurate surrogates, particularly when working with smaller datasets. This also suggests that introducing more variability in the parameters can lead to increased model complexity and potential overfitting. This finding serves to underscore the significant value of the process outlined in my thesis. It is anticipated that in the future, numerous engineering cases will necessitate the consideration of a substantial number of parameters, thereby necessitating the utilization of a voluminous dataset. At this juncture, the efficacy of the process in achieving the objective of reducing time and resource expenditure while generating substantial datasets is evident.

4.3.3. Summary

The DIM-GP model implemented with Gpytorch demonstrated strong predictive capability for signal-level electromagnetic responses. Key findings include:

1. High accuracy across S_{11} and gain signals, with RMSE down to 0.01 dB on S-parameters and 0.54 dB on gain patterns
2. Stable convergence across different datasets and learning rates
3. Scalability, with better performance as more simulation data becomes available
4. Suitability for integration with OptiSLang via Python Block, enabling automated, signal-aware EM modeling
5. These results confirm that the adopted machine learning framework is effective in capturing nonlinear, multi-variable relationships in high-frequency package structures.

5

Future Work

Building on the results of this thesis, several ways for future work are proposed to extend and enhance the methodology.

5.1. Broader Application of the Methodology

The modeling approach can be applied to a wider range of package and interconnect configurations to verify its generality. For example, future studies could evaluate different advanced packaging technologies (such as other flip-chip layouts, 2.5D/3D integration schemes, or antenna-in-package designs) using the same workflow. This would demonstrate how well the Gaussian-process surrogate model adapts to new structures and frequency ranges, and it may highlight any modifications needed to handle more complex or higher-frequency scenarios.

5.2. Advanced Surrogate Modeling Techniques

While this work focused on an exact GPR model in GPyTorch, exploring other machine learning models or improved GP variants could further boost performance. One direction is to investigate sparse or approximate GP techniques (e.g. utilizing GPyTorch's variational strategies) to maintain scalability as the training dataset grows. Another possibility is incorporating alternative models like deep neural networks methods to compare against GPR, especially if future datasets become large or if the problem involves highly nonlinear behavior beyond the Gaussian process assumptions. Such comparisons and extensions would clarify the trade-offs between model accuracy, training cost, and interpretability for this application.

Enhanced Workflow and Automation: The current OptiSLang-based workflow could be extended into a more adaptive loop. In future work, an active learning strategy could be implemented where the surrogate's own uncertainty feedback guides the selection of new simulation points. By iteratively sampling regions with higher predictive uncertainty, the model can be refined efficiently, focusing computational effort where it is most needed. In addition, engineers can direct the iterative loop toward regions of primary interest. They may allocate more EM simulations to these subdomains and retrain the surrogate after each increment. This focused sampling improves local accuracy and reduces predictive uncertainty within the range that matters for design decisions. Regions of interest can be defined by performance targets, constraint margins, or candidate worst case corners. The procedure alternates between targeted sampling and model updating until the estimated error falls below a predefined threshold or the design objectives are met.

6

Conclusions

In this work, a methodology was presented to efficiently model electromagnetic performance variability in advanced electronic packaging. The approach combined parametric full-wave simulations with machine learning to capture the effects of manufacturing variations at the die-package-PCB interface. Using Ansys HFSS 3D Layout, a series of 3D EM simulations were performed over a Latin Hypercube Sampling of geometry parameters to generate a simualtion dataset of S-parameters and far-field patterns. Optislang enabled automated evaluation of new parameter combinations without rerunning expensive simulations based on the DIM-GP metamodel of the simulation data. The evaluation results obtained thus constituted the training dataset. A Gaussian Process Regression (GPR) surrogate model was then trained on this dataset using the GPyTorch library.

The results demonstrate that the GPR surrogate can achieve high accuracy in predicting key RF performance metrics. In the case study, the model predicted return loss (S_{11}) and antenna gain patterns with errors on the order of only a few tenths of a decibel, closely matching the full-wave HFSS simulations. The surrogate also showed stable convergence behavior and improved fidelity as more training samples were incorporated. Once trained, generating a new EM prediction took only milliseconds – an orders-of-magnitude speed-up compared to individual HFSS runs that can take hours. This drastic reduction in computational cost enables extensive variation analyses and rapid "what-if" evaluations that would be impractical with brute-force simulation alone.

Overall, the developed workflow effectively brings a form of electromagnetic corner analysis to the package and board level. Using the surrogate model, designers can quickly identify worst-case performance corners and quantify the impact of package process tolerances, analogous to how PVT corners are used in chip design. The study successfully demonstrates that a GPR-based surrogate can bridge the gap between expensive full-wave EM simulations and the need for fast, scalable prediction. In doing so, it provides a practical tool for early-stage design exploration and robust verification of high-frequency packaging designs under manufacturing variability.

References

[1] Ansys Inc. "Ansys optislang." Version 2023 R2. (2023), [Online]. Available: https://www.ansys.com/products/connect/ansys-optislang (visited on 09/18/2025).

[2] J. Gardner, G. Pleiss, K. Q. Weinberger, D. Bindel, and A. G. Wilson, "Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration," *Advances in neural information processing systems*, vol. 31, 2018.

[3] J. H. Lau, "Recent advances and trends in advanced packaging," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 12, no. 2, pp. 228–252, 2022. DOI: 10.1109/TCPMT.2022.3144461.

[4] S. Ahn, A. Lu, W. Fan, L. Wai, and J. Kim, "Effects of process variation on signal integrity for high speed differential signaling on package level," in *4th Electronics Packaging Technology Conference, 2002.*, 2002, pp. 249–252. DOI: 10.1109/EPTC.2002.1185676.

[5] J. Aguirre, H. Tallo, H. Shigenobu, and J. Uyesugi, "Ceramic process variation impact on electrical design of high frequency components," in *International Symposium on Microelectronics*, International Microelectronics Assembly and Packaging Society, vol. 2014, 2014, pp. 000 205–000 212.

[6] D. V. C. d. Nascimento, K. Georgiou, K. I. Eder, and S. Xavier-de-Souza, "Evaluating the effects of reducing voltage margins for energy-efficient operation of mpsocs," *IEEE Embedded Systems Letters*, vol. 16, no. 1, pp. 25–28, 2024. DOI: 10.1109/LES.2023.3240625.

[7] S. S. Iyer and A. A. Bajwa, "Reliability challenges in advance packaging," in *2018 IEEE International Reliability Physics Symposium (IRPS)*, IEEE, 2018, pp. 4D–5.

[8] K. C. Sahu, S. Koziel, and A. Pietrenko-Dabrowska, "Surrogate modeling of passive microwave circuits using recurrent neural networks and domain confinement," *Scientific Reports*, vol. 15, p. 13 322, 2025.

[9] W. Chen, "The application of different surrogate models in engineering predictions," *Applied and Computational Engineering*, vol. 80, no. 1, pp. 87–93, 2024. DOI: 10.54254/2755-2721/80/2024ch0075.

[10] Y. Li and W. Kim, "The applications of simulation and artificial intelligence in advanced packaging," in *2024 IEEE International Symposium on the Physical and Failure Analysis of Integrated Circuits (IPFA)*, IEEE, 2024, pp. 1–5.

[11] H. Zou, S. Zeng, C. Li, and J. Ji, "A survey of machine learning and evolutionary computation for antenna modeling and optimization: Methods and challenges," *Engineering Applications of Artificial Intelligence*, vol. 138, p. 109 381, 2024.

[12] D. Lee and S. Engineer, "Fast-track chiplet integration with streamlined ucie's electrical layer analysis,"

[13] M. S. Sharawi, "Practical issues in high speed pcb design," *IEEE Potentials*, vol. 23, no. 2, pp. 24–27, 2004.

[14] *IPC-2221A: Generic Standard on Printed Board Design*, IPC International, 2003.

[15] M. Stiteler, C. Ume, and B. Leutz, "In-process board warpage measurement in a lab scale wave soldering oven," in *1996 Proceedings 46th Electronic Components and Technology Conference*, 1996, pp. 247–254. DOI: 10.1109/ECTC.1996.517399.

[16] Y. Liu, X. Li, and S. Yin, "Review of chiplet-based design: System architecture and interconnection," *Science China Information Sciences*, vol. 67, no. 10, p. 200 401, 2024.

[17] K. Zhang, "1.1 semiconductor industry: Present & future," in *2024 IEEE International Solid-State Circuits Conference (ISSCC)*, IEEE, vol. 67, 2024, pp. 10–15.

[18] X. Zhang, J. K. Lin, S. Wickramanayaka, *et al.*, "Heterogeneous 2.5 d integration on through silicon interposer," *Applied physics reviews*, vol. 2, no. 2, 2015.

[19] A. B. Kahng, U. Mallappa, L. Saul, and S. Tong, ""' unobserved corner" prediction: Reducing timing analysis effort for faster design convergence in advanced-node design," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, 2019, pp. 168–173.

[20] A. Papadopoulou, "Variability analysis and yield optimization in deep-submicron mixed-signal circuits," Ph.D. dissertation, UC Berkeley, 2017.

[21] H. Li, S. Moore, and A. T. Markettos, "A simulation methodology for electromagnetic analysis and testing on synchronous and asynchronous processors,"

[22] D. Staiculescu, A. Pham, J. Laskar, S. Consolazio, and S. Moghe, "Analysis and performance of bga interconnects for rf packaging," in *1998 IEEE Radio Frequency Integrated Circuits (RFIC) Symposium. Digest of Papers (Cat. No.98CH36182)*, 1998, pp. 131–134. DOI: 10.1109/RFIC.1998.682064.

[23] Ansys, Inc., *An introduction to hfss*, Ansys, 2024.

[24] M. Commens, J. Mologni, and P. Gasperini. "Electromagnetic simulation and 3d-ic interposers." (2022), [Online]. Available: https://semiengineering.com/electromagnetic-simulation-and-3d-ic-interposers/ (visited on 08/11/2025).

[25] T. Simpson, J. D. Peplinski, P. Koch, and J. Allen, "Metamodels for computer-based engineering design: Survey and recommendations," in *Engineering computations*, 2001. DOI: 10.1007/PL00007198.

[26] G. G. Wang and S. Shan, "Review of metamodeling techniques for product design with computation-intensive processes," 2011. DOI: 10.24908/PCEEA.V0I0.3940.

[27] T. Krishnamurthy, "Comparison of response surface construction methods for derivative estimation using moving least squares, kriging and radial basis functions," 2013. DOI: 10.2514/6.2005-1821.

[28] Ansys, Inc. "Ansys help documentation." Ansys online help portal, includes documentation for Fluent, Workbench, Mechanical APDL, etc. (2025), [Online]. Available: https://ansyshelp.ansys.com/ (visited on 08/12/2025).

[29] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.

[30] H. Ma, E.-P. Li, A. C. Cangellaris, and X. Chen, "Support vector regression-based active subspace (svr-as) modeling of high-speed links for fast and accurate sensitivity analysis," *IEEE Access*, vol. 8, pp. 74 339–74 348, 2020.

[31] M. Shi, S. Wang, W. Sun, L. Lv, and X. Song, "A support vector regression-based multi-fidelity surrogate model," *arXiv preprint arXiv:1906.09439*, 2019.

[32] C. Etienam, K. J. H. Law, S. Wade, and V. Zankin, "Fast deep mixtures of gaussian process experts," *Machine Learning*, vol. 113, no. 3, pp. 1483–1508, 2024. DOI: 10.1007/s10994-023-06491-x.

[33] Scikit-learn developers. "Gaussian processes regression: Basic introductory example." Example from scikit-learn documentation; authors: Vincent Dubourg, Jake Vanderplas, Jan Hendrik Metzen :contentReference[oaicite:1]index=1. (n.d.), [Online]. Available: https://scikit-learn.org/stable/auto_examples/gaussian_process/plot_gpr_noisy_targets.html (visited on 08/12/2025).

[34] M. B. Salem, O. Roustant, F. Gamboa, and L. Tomaso, "Universal prediction distribution for surrogate models," in *SIAM/ASA J. Uncertain. Quantification*, 2015. DOI: 10.1137/15M1053529.

[35] T. Kavzoglu, "Increasing the accuracy of neural network classification using refined training data," *Environmental Modelling amp; Software*, vol. 24, no. 7, pp. 850–858, 2009. DOI: 10.1016/j.envsoft.2008.11.012.

[36] A. optiSLang Algorithm Team, *Recent developments in metamodeling, optimization, and uncertainty quantification in optislang*, Technical report, Accessed: 2025-08-12, n.d. [Online]. Available: https://library.dynardo.de/fileadmin/Material_Dynardo/bibliothek/WOST19/12_WOST22_Recent_Developments_Ansys_Thomas_Most.pdf.

[37] P. P. GmbH. "Stochos – STOCHOS (Software page)." Web page describing the STOCHOS software; copyright date provided (2023). (2023), [Online]. Available: https://probaligence.de/stochos/ (visited on 08/12/2025).

[38] N. HajiGhassemi and M. Deisenroth, "Analytic long-term forecasting with periodic gaussian processes," in *International Conference on Artificial Intelligence and Statistics*, 2014.

[39] M. Fuhrländer and S. Schöps, "A blackbox yield estimation workflow with gaussian process regression applied to the design of electromagnetic devices," *Journal of Mathematics in Industry*, vol. 10, no. 1, Oct. 2020. DOI: 10.1186/s13362-020-00093-1.

[40] F. Garbuglia, D. Spina, T. Reuschel, C. Schuster, D. Deschrijver, and T. Dhaene, "Modeling s-parameters of interconnects using periodic gaussian process kernels," in *Workshop on Signal Propagation on Interconnects*, 2023. DOI: 10.1109/SPI57109.2023.10145548.

[41] D. Duvenaud, "Automatic model construction with gaussian processes," 2014. DOI: 10.17863/CAM.14087.

[42] GPyTorch contributors. "Source code for gpytorch.kernels.periodic_kernel (gpytorch 1.8.1)." Displays implementation and mathematical formulation of the PeriodicKernel. (2025), [Online]. Available: https://docs.gpytorch.ai/en/v1.8.1/_modules/gpytorch/kernels/periodic_kernel.html (visited on 08/12/2025).

[43] A. C. Sanabria-Borbón, S. Soto-Aguilar, J. J. Estrada-López, D. Allaire, and E. Sánchez-Sinencio, "Gaussian-process-based surrogate for optimization-aided and process-variations-aware analog circuit design," *Electronics*, vol. 9, no. 4, p. 685, 2020.

[44] A. Banerjee, D. B. Dunson, and S. T. Tokdar, "Efficient gaussian process regression for large datasets," *Biometrika*, vol. 100, no. 1, pp. 75–89, Dec. 2012. DOI: 10.1093/biomet/ass068.

[45] W. Syed, "On the control of surface waves in integrated antennas: Analysis and design exploiting artificial dielectric layers," Ph.D. dissertation, Ph. D. dissertation, Delft University Of Technology, 2015.

[46] K. Doris, F. Jansen, M. Lont, *et al.*, "Mm-wave automotive radar: From evolution to revolution," in *2021 IEEE International Electron Devices Meeting (IEDM)*, 2021, pp. 25.7.1–25.7.4. DOI: 10.1109/IEDM19574.2021.9720646.

[47] M. Spella, W. H. Syed, D. Cavallo, M. Huang, and L. V. Gemert, "Integrated circuit package," US10615134B2, United States Patent, Apr. 2020. [Online]. Available: https://patentimages.storage.googleapis.com/0c/c4/4c/d571071ff97d59/US10615134.pdf.

[48] M. D. McKay, R. J. Beckman, and W. J. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 42, no. 1, pp. 55–61, 2000.

[49] E. Hale, "Surrogate modeling approaches for use with prognostics models package (intern notes)," 2021.

[50] C. Kamath, "Intelligent sampling for surrogate modeling, hyperparameter optimization, and data analysis," in *Machine Learning with Applications*, 2021. DOI: 10.1016/j.mlwa.2022.100373.

[51] R. B. Gramacy, *Surrogates: Gaussian process modeling, design, and optimization for the applied sciences*. Chapman and Hall/CRC, 2020.

[52] G. E. P. Box and K. B. Wilson, "On the experimental attainment of optimum conditions," in *Breakthroughs in Statistics: Methodology and Distribution*, S. Kotz and N. L. Johnson, Eds. New York, NY: Springer New York, 1992, pp. 270–310, ISBN: 978-1-4612-4380-9. DOI: 10.1007/978-1-4612-4380-9_23.

[53] G. E. P. Box and D. W. Behnken, "Some new three level designs for the study of quantitative variables," *Technometrics*, vol. 2, no. 4, pp. 455–475, 1960. DOI: 10.1080/00401706.1960.10489912.

[54] F. A. C. Viana, "A tutorial on latin hypercube design of experiments," *Quality and Reliability Engineering International*, vol. 32, no. 5, pp. 1975–1985, Nov. 2015. DOI: 10.1002/qre.1924.

[55] M. Stein, "Large sample properties of simulations using latin hypercube sampling," *Technometrics*, vol. 29, no. 2, pp. 143–151, 1987.

[56] A. Afzal, K.-Y. Kim, and J.-W. Seo, "Effects of latin hypercube sampling on surrogate modeling and optimization," 2017. DOI: 10.5293/IJFMS.2017.10.3.240.

[57] A. Hami, B. Radi, and C. Huang, "Overview of structural reliability analysis methods — part ii: Sampling methods," 2017. DOI: 10.21494/ISTE.OP.2017.0116.

[58] H. Liu, J. Cai, and Y.-S. Ong, "Remarks on multi-output gaussian process regression," *Knowledge-Based Systems*, vol. 144, pp. 102–121, 2018.

[59] GPyTorch developers. "Simple gp regression — gpytorch (v1.13) example." Basic tutorial demonstrating Exact GP regression with GPyTorch v1.13, covering training, model setup, prediction, and plotting. (2023), [Online]. Available: https://docs.gpytorch.ai/en/v1.13/examples/01_Exact_GPs/Simple_GP_Regression.html (visited on 08/12/2025).