METAMODELING ATOMIC MODELS IN DISCRETE EVENT SYSTEM SPECIFICATION (DEVS) FORMALISM USING MULTIVARIATE ADAPTIVE REGRESSION SPLINES (MARS)

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

CUMHUR DORUK BOZAĞAÇ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
COMPUTER ENGINEERING

JANUARY 2014

Approval of the thesis:

METAMODELING ATOMIC MODELS IN DISCRETE EVENT SYSTEM SPECIFICATION (DEVS) FORMALISM USING MULTIVARIATE ADAPTIVE REGRESSION SPLINES (MARS)

submitted by **CUMHUR DORUK BOZAĞAÇ** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Adnan Yazıcı _____
Head of Department, **Computer Engineering**

Assoc. Prof. Dr. Halit Oğuztüzün _____
Supervisor, **Computer Engineering Department, METU**

Prof. Dr. İnci Batmaz _____
Co-supervisor, **Statistics Department, METU**

Examining Committee Members:

Prof. Dr. İsmail Hakkı Toroslu
Computer Engineering Department, METU _____

Assoc. Prof. Dr. Halit Oğuztüzün
Industrial Engineering Department, METU _____

Prof. Dr. Nur Evin Özdemirel
Computer Engineering Department, METU _____

Prof. Dr. Sibel Tarı
Computer Engineering Department, METU _____

Assist. Prof. Dr. Kayhan İmre
Computer Engineering Department, Hacettepe University _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: CUMHUR DORUK BOZAĞAÇ

Signature :

ABSTRACT

METAMODELING ATOMIC MODELS IN DISCRETE EVENT SYSTEM SPECIFICATION (DEVS) FORMALISM USING MULTIVARIATE ADAPTIVE REGRESSION SPLINES (MARS)

Bozağaç, Cumhur Doruk
Ph.D., Department of Computer Engineering
Supervisor : Assoc. Prof. Dr. Halit Oğuztüzün
Co-Supervisor : Prof. Dr. İnci Batmaz

January 2014, 94 pages

Computer simulations are widely used for design optimization purposes. The problem becomes challenging when design variables are high dimensional and when the simulation is computationally expensive. In this work we propose a methodology for metamodeling of dynamic simulation models via Multivariate Adaptive Regression Splines (MARS). To handle incomplete output processes, where the simulation model does not produce an output in some steps due to missing inputs, we have devised a two-level metamodeling scheme. The methodology is demonstrated on a dynamic radar simulation model. The prediction performance of the resulting metamodel is tested with four different sampling techniques (i.e., experimental designs) and 16 sample sizes. We also investigate the effect of alternative coordinate system representations on the metamodeling performance. The results suggest that MARS is an effective method for metamodeling dynamic simulations, particularly, when expert judgment is not readily available. Results also show that there are interactions

between the coordinate system representations and sampling techniques, and some sampling-representation-size combinations are very promising in the solution to this type of problem. The technique is then applied to develop proxy models (PMs) of atomic models in Discrete Event System Specification (DEVS) simulations to replace an atomic model with a PM that uses the fitted metamodel. The mechanisms required for replacing an atomic model and integrating the PM into a simulation are described in detail. The methodology is tested by replacing a radar atomic model in a military engagement simulation with the PM and the method's benefits and challenges are discussed thoroughly. As a final step of this research, the response surfaces of the original simulation and the PM integrated simulation is analyzed. Both simulations are used in a Particle Swarm Optimization based optimization procedure and the results are compared. Results obtained from this particular case suggest that metamodeling of computationally intensive atomic models is feasible, and even with relatively small number of observations, we can apply metamodeling to sub-components of DEVS simulation models successfully.

Keywords: metamodeling, DEVS, MARS, dynamic simulation model, discrete simulation, statistical experimental design

ÖZ

KESİKLİ OLAY SİSTEM BELİRTİMİ (DEVS) FORMALİZMİNDEKİ ATOMİK MODELLERİN ÇOK DEĞİŞKENLİ UYARLANABİLİR REGRESYON EĞRİLERİ (MARS) İLE METAMODELLENMESİ

Bozağaç, Cumhur Doruk
Doktora, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Doç. Dr. Halit Oğuztüzün
Ortak Tez Yöneticisi : Prof. Dr. İnci Batmaz

Ocak 2014 , 94 sayfa

Bilgisayar benzetimleri tasarım eniyilemesinde sıkça kullanılmaktadır. Tasarım değişkenlerinin çok boyutlu olması ve simülasyanların çok işlem gücü gerektirmesi durumunda ise problem daha zor olmaktadır. Bu çalışmada dinamik benzetim modellerinin çok değişkenli uyarlanabilir regresyon eğrileri (MARS) ile metamodellenmesi için bir yöntem önerilmektedir. Benzetim modellerinin her adımda çıktı üretmediği durumlarda eksik çıktı işlemlerinin ele alınabilmesi için iki seviyeli bir metamodelleme düzeni geliştirilmiştir. Yöntem dinamik bir radar benzetim modelinde uygulanmış ve sonuçlar dört farklı örnekleme yöntemi ve 16 farklı örneklem genişliği için test edilmiştir. Ayrıca alternatif koordinat sistemlerinin metamodelleme üzerindeki etkileri de araştırılmıştır. Sonuçlar MARS'ın dinamik benzetimler için ve özellikle uzman görüşüne başvurmanın zor olduğu durumlarda uygulanabilir olduğunu göstermektedir. Sonuçlar ayrıca koordinat sistemleri ile örnekleme yöntemleri arasında et-

kileşimler olduğunu ve bazı koordinat-örnekleme yöntemi-örneklem genişliği kombinasyonlarının örnek problem için daha başarılı olduğunu göstermektedir. Geliştirilen metamodelleme yöntemi, kesikli olay sistem belirtimi (DEVS) kullanan benzetimlerde atomik modellerin metamodel kullanan bir vekil model (VM) ile değiştirilmesinde kullanılmıştır. Bir atomik modelin değiştirilmesi için gereken mekanizmalar ve VM'in kullanılmasının faydalı ve zararlı yönleri ayrıntılı olarak tartışılmıştır. Yöntem bir angajman benzetimindeki radar atomik modelinin VM'i ile değiştirilmesi sağlanarak test edilmiştir. Bu araştırmanın son adımı olarak orjinal benzetimin ve VM kullanılan benzetimin tepki yüzeyleri karşılaştırılmış ve iki benzetim parçacık sürü optimizasyonunda kullanılmıştır. Örnek problemden elde edilen sonuçlar, gerçek benzetim ile karşılaştırıldığında atomik modellerin metamodellemesinin mümkün olduğunu ve göreceli olarak küçük örneklem genişlikleri ile bile başarılı metamodellerin oluşturulabildiğini göstermektedir.

Anahtar Kelimeler: metamodelleme, DEVS, MARS, dinamik benzetimler, deney tasarımı

Achievement is impossible without dedication. I lovingly dedicate this thesis to my wife who supported me each step of the way and to my son who is my biggest inspiration.

ACKNOWLEDGMENTS

This thesis has been kept on track and been seen through to completion with the support and encouragement of numerous people including my well wishers, my friends, colleagues and my institution. At the end of my thesis, it is a pleasant task to express my thanks to all those who contributed in many ways to the success of this study and made it an unforgettable experience for me.

At this moment of accomplishment, first of all I pay homage to my guide, Assoc. Prof. Dr. Halit Oğuztüzün. This work would not have been possible without his guidance, support and encouragement. Under his guidance I successfully overcame many difficulties and learned a lot. His unflinching motivation and faith in me was inspiring. I would like to thank him for his constructive comments and support throughout this work.

I am also extremely indebted to my co-supervisor and my guide Prof. Dr. İnci Batmaz, for providing necessary knowledge and resources to accomplish my research work. I am very much thankful to her for picking me up as a student at the critical stage of my Ph.D. Her extensive discussions around my work and interesting explorations in statistics have been very helpful for this study.

I also want to thank my former co-supervisor Dr. M. Nedim Alpdemir for his encouraging and constructive approach together with his efforts and patience throughout my thesis work.

Special thanks to my thesis monitoring committee members, Prof. Dr. İsmail Hakkı Toroslu and Prof. Dr. Nur Evin Özdemirel for their support, guidance and helpful suggestions. Their guidance has served me well and I owe them my heartfelt appreciation.

I would also like to thank my committee members, Prof. Dr. Sibel Tarı and Assist. Prof. Kayhan İmre, for reviewing and evaluating my thesis.

I want to thank my family for their unflagging love and support throughout my life; this dissertation is simply impossible without them. I am indebted to my father, Ali Bozağaç, for his care, love and support. I cannot ask for more from my mother, Berrin Bozağaç, who is the greatest well wisher and I have no suitable word that can fully describe her everlasting love to me. I would like to thank my big brother Burak Bozağaç, for his support, encouragement and positive spirit. I also want to thank to my in-laws Alpaslan Gülsoy and Eda Gülsoy for their unconditional support.

I should also mention TÜBİTAK / BİLGEM / İLTAREN for allowing me to be part of a great professional community and supporting me during my academic work. I would also like to thank Fatma Yerlikaya-Özkurt for her help to me during my efforts with the R environment for building the MARS metamodel.

Last, but not least, I would like to thank my wife Gülin and my son Kaan for their understanding and love during the past few years. My wife's support, encouragement, quiet patience and unwavering love were undeniably the bedrock upon which the past seven years of my life have been built. Her tolerance of my occasional vulgar moods is a testament in itself of her unyielding devotion and love. Her support and encouragement was in the end what made this dissertation possible. She helped me to concentrate on completing this dissertation and supported mentally during the course of this work. She already has my heart so I can just give her a heartfelt gratitude.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

NOMENCLATURE

AA	Anti aircraft
AFD	Asymmetrical factorial design
ANN	Artificial neural networks
ATC	Analytical target cascading
BFs	Basis functions
BMDS	Ballistic missile defense system
BN	Bayesian network
C2	Command and control system
CCD	Central composite design
CMBD	Combined coordinate system
CO	Colloborative optimization
CRTSN	Cartesian coordinate system
CTBN	Continuous time bayesian network
d_{AAB}	Minimum AA bullet distance to aircraft
d_{ASM}	Minimum missile distance to ship
DBN	Dynamic bayesian network
DEVS	Discrete event system specification
FN	False negative
FP	False positive
HD	Hybrid design
HMM	Hidden markov model
IBDTBN	Interval based discrete time bayesian network
ITU	International Telecommunication Union
LM	Logger (atomic) model
LHS	Latin hypercube design
MAD	Median absolute deviation
MAE	Mean absolute error
MAM	Meta atomic model

MARS	Multivariate adaptive regression splines
MC	Monte carlo methods
MSE	Mean square error
N	Number of rows in the complete observation table ($N = n \times \omega$)
n	Cardinality of the set of points selected by the sampling technique
n_{AAB}	Total number of AA bullets fired by the ship
OS	Original simulation
PM	Proxy (atomic) model
PMIS	Proxy model integrated simulation
POLAR	Polar coordinate system
PSO	Particle swarm optimization
P	Number of predictors in the complete observation table ($P = p + q + r$)
p	Number of design variables of the simulation model
q	Number of dynamic inputs of the simulation model
r	Number of outputs of the simulation model
r	Correlation coefficient
R^2	Coefficient of determination
RNN	Recurrent neural network
SIMA	Simulation modeling architecture
SSM	State space model
STB	Stability of training and test results
TBN	Temporal bayesian network
TN	True negative
TP	True positive
TR	Training data set
TE	Test data set
V	Set of all dynamic inputs or input events of a simulation model
V_t	Set of all inputs received at time t by a simulation model
v_{it}	Dynamic input received from port i at time step t
X	Set of all design variables of a simulation model
x_{ij}	Design variable i of sample number j
Y	Set of all output produced by a simulation model
Y_t	Set of all outputs produced at time t by a simulation model
y_{it}	Output produced from port i at time step t by a simulation model

Y'	Set of all previous time step outputs produced by a simulation model
y'_{it}	Output i produced at time step $t - 1$ by a simulation model
UD	Uniform design
z_{ij}	Knot for the jth observation of the ith predictor in a MARS model
ω	Number of time steps in one simulation run

CHAPTER 1

INTRODUCTION

Computer simulation is frequently used in solving real world problems by evaluating models and optimizing responses. Large-scale engineering design problems can include a large number of variables and finding the optimum solution in these problems can be very difficult and time consuming. Computer simulations enable analysts to explore the behavior of complex systems through the use of mathematical models. Modeling a complex system often requires composition of models where each model represents a particular aspect or component of the system. The system is in general determined by a set of design variables and parameters (system inputs). Optimization of a complex system involves determining the values for design variables that produce the best system response based on some criteria. Design variables can be changed by the designer, while input parameters are considered fixed during the design process.

Optimization of a system is harder if we do not know the closed-form definition of the system (either because system is too complex or there may be encapsulated components in the system). Designer cannot use analytical optimization methods such as those using derivatives of the function, therefore performance evaluation is limited with system simulation execution. Shan and Wang [104] carried out a survey on modeling and optimization strategies to solve high-dimensional, expensive, black-box simulations. Metamodeling and decomposition are two common approaches used for solving these kinds of problems. Hierarchical decomposition allows designers to exploit the intrinsic structure of the problem in terms of smaller sub-problems, but if the subproblems are highly-coupled, then integration or coordination will be harder. Another way of dealing with the computational intensity involved in such problems is

to use model approximation with simulation metamodels, which are also called surrogate models or emulators [42, 45]. Then the simulation is treated like an input-output function and separate metamodels need to be built for each of the outputs [5].

An early metamodeling technique is fitting polynomial functions by regression analysis for approximating response surfaces [5]. There are a substantial number of studies in the literature that use this technique, such as [85, 89, 102]. Other metamodeling methods used contain, for example, kriging [67], radial basis functions [38], bayesian methods [90], artificial neural networks (ANNs) [91], and support vector regression machines [35]. Among them, ANN is the most popular choice for metamodeling due to its ability to provide a good fit with nonlinear models. However, ANNs require structure specification [58], and appropriate hidden layers selection [23] to avoid overfitting. Yet, there are other techniques, reported in [44, 50]. Friedman's method, called multivariate adaptive regression splines (MARS) depends on recursive partitioning of the input space and competes with the performance of ANNs [30, 78, 84, 116].

Most metamodeling techniques predict system performance using an average value or the final state of a simulation. These metamodels neither represent the time evolution of a simulation, nor can they give information about the simulation states at specific time instances. Consequently, most metamodels cannot represent dependencies between simulation states at different time instances. Dynamic bayesian network (DBN) [31] or recurrent neural network (RNN) [25] are two attempts to achieve a time series prediction. Nevertheless, DBN may require *a priori* knowledge about state of the system [94] and RNN is not guaranteed to reach stability during training [115].

Conti et al. [26] propose predicting time series output of a dynamic simulation using a multi-output Gaussian emulator. Our study is inspired by their emulator, but we build a metamodel with certain novelties: a) we use MARS instead of their Gaussian approach, and b) we devise a two-level metamodeling scheme to handle an incomplete output process where simulation does not always generate an output, while the first metamodel decides *whether to output at this time step*, the second predict *what to output*. To demonstrate our approach, a dynamic radar simulation is modeled using

MARS [44] metamodels. MARS is selected in this work, because it is flexible enough to model complex (i.e, high-dimensional involving non-linear relationships) systems successfully without any expert intervention. The performance of the metamodel is analyzed by considering various sampling techniques and different sample sizes. In addition, the effect of alternative coordinate system representations on modeling performance is also investigated.

After analyzing the effectiveness of the dynamic system metamodel, we tackle complex system simulations problem with a divide and fit approach and combine decomposition and metamodeling. Instead of building a metamodel of the complete system simulation, the simulation system is decomposed into sub-components (atomic models) using the Discrete Event System Specification (DEVS) [121] formalism. Atomic models usually generate time dependent output in accordance with input events, hence metamodeling of an atomic model requires dynamic system metamodeling. The metamodeling approach presented in Section 4 is applied to metamodel atomic models in a DEVS simulation. Note that in literature emulator, surrogate model and metamodel are used interchangeably to refer to modeling of a simulation model. From this point on, we prefer to use the term 'metamodel' for this purpose.

Our divide and fit approach consists of three stages: i) Metamodeling of a selected atomic model, ii) Integrating the whole (composite) simulation with a proxy model (PM) by replacing the atomic model, and iii) Running the composite simulation. Of course, the process is the same when we target more than one atomic model for the metamodeling effort, but in this work, to illustrate our approach better, we simplify our presentation by considering one atomic model only. Our approach is applicable to any simulation framework that allows a hierarchical composition/decomposition of models where (sub)models (with encapsulated states) interact through port-based interfaces by exchanging messages. We employ DEVS as it is a well-established formalism and we have the development tools and execution environment at hand.

PM replacement technique is applied to a dynamic radar atomic model in a one-on-one military engagement simulation. The radar atomic model is replaced by a PM implementation using the trained metamodels. Prediction performance of the PM depends on the performance of the metamodels, which in turn depends on the type

of the sampling technique used during the training. PM integrated simulation (PMIS) performance is evaluated with respect to prediction accuracy, complexity and stability according to the simulation results obtained by using different sampling techniques for training. We also compare the computation time of the original simulation (OS) and PMIS.

After selecting the proper sampling technique and sample size of PM, the PMIS is used in an optimization procedure. The aim is finding the best design variables of a radar system to detect an engaging aircraft for providing the anti aircraft (AA) gun the best position information. The success of the radar system is measured by the minimum miss distance of the AA bullets to the aircraft (d_AAB). We apply single objective optimization to the PMIS without any constraints.

This work is organized as follows: Chapter 2 provides background information about the DEVS formalism, sampling techniques, metamodeling techniques, and particularly, MARS metamodeling. Insight about the previous works on radar simulation metamodeling and dynamic simulation metamodeling are given in Chapter 3. Chapter 4 gives detailed information about our dynamic system metamodeling approach and, particularly, Section 4.2 explains the case involved for analyzing the approach. In Section 4.3, we present our experimental setup and the results for the metamodel selection. The divide and fit approach and the sample simulation for the approach is presented in Chapter 5 and Section 5.2, respectively. In Section 5.3, we present our experimental setup and the results for PM. The application of the developed methods to optimization is provided in Chapter 6 with background information on the optimization technique and results of the experiments. Finally, the conclusion and future work are given in Chapter 7.

CHAPTER 2

BACKGROUND

In this chapter, we present background information about the DEVS formalism and the selected sampling techniques. We enumerate some of the methodologies for handling computationally expensive simulations and give detailed information about selected metamodeling technique which is, MARS.

2.1 Discrete Event System Specification (DEVS)

DEVS is a formalism introduced by Bernard Zeigler et al. in 1976 [121] to describe discrete event systems. In this formalism there are two types of models: atomic models and coupled models. States are maintained and state transitions are performed in atomic models. On the other hand, coupled models consist of other models and connections between those models. An atomic model in classical DEVS formalism consists of a set of input events, a state set, a set of output events, internal and external transition functions, an output function and a time advance function. Formal definition of an atomic model in classical DEVS formalism is defined as follows [121]:

$$M = \langle V, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta \rangle ,$$

where

V is the finite set of input events,

S is the finite set of states,

Y is the finite set of output events,

$\delta_{int} : S \rightarrow S$ is the internal transition function,

$\delta_{ext} : Q \times V \rightarrow S$ is the external transition function, where

$Q = \{(s, t_e) \,|s \in S, 0 \leq t_e \leq ta(s)\}$ is total state set,

$\lambda : S \rightarrow Y^{\Phi}$ is the output function, where

$Y^{\Phi} = Y \bigcup \Phi$ and $\Phi \notin Y$ is a state where model does not generate output

$ta : S \rightarrow \mathbb{T}$ is the time advance function.

Note that, t_e is the elapsed time since last external or internal transition, $\mathbb{T}$ is the set of nonnegative real numbers with the special 'infinity' element, designated ∞.

In DEVS formalism, simulation models communicate with each other using their ports, which are interfaces of models. External input events (V) are received by input ports and output events (Y) are sent by output ports. The state set S is valid for a time interval, which is determined by the time advance (ta) function. After the completion of each time interval, the output function (λ) is executed to send the output events that belong to the current state S, and then, the internal transition function (δ_{int}) is executed to calculate a new state. If a model receives an external event during this time interval, the external transition function (δ_{ext}) is executed and state of the model $s \in S$ is updated to reflect the effects of the incoming event.

2.2 Selected Sampling Techniques

The success of metamodeling technique depends on the sampling technique (experimental design) used in simulation to gather information about the function modeled. Most relevant works on design of simulation experiments are McKay et al. [81], Stein [107], Sacks et al. [100], Kleijnen [66] and Morris and Mitchell [86].

Sampling techniques used in simulation are typically classified in two groups: *classical* and *space filling* [106]. In this study we use an asymmetrical factorial design (AFD) and hybrid design (HD), which are classical, and the latin hypercube design (LHS) and uniform design (UD), which are space filling, in order to evaluate their

effects on the metamodeling performance. Brief descriptions of the designs are provided below.

The AFD is a factorial experiment with p factors each with l_i levels where $1 \leq i \leq p$. Different sample sizes with distinct factor levels can be selected and the AFD allows for the testing of the factorial design technique with small increments in the sample sizes [122]. It requires $n = l_1 \times l_2 \times ... \times l_p$ number of design points to run. Factorial designs require a fairly large number of distinct design points. An alternative approach is to use central composite design based HD as suggested by Roquemore [99]. Batmaz and Tunalı [7] show that HD is rotatable, efficient and has good predictive capabilities. Hence, we include it in our tests to compare its effect with those of the other designs studied. The design matrices in this study are taken from Myers et al. [89].

Classical design approaches tend to place most design points on or near the boundaries of an experimental region, leaving the interior region unattended. Space filling designs, on the other hand, fill the experimental region by selecting design points at equal distances. LHS [55] is a popular space filling design, where the range of each one of p factors is divided into n intervals. From each interval of each factor we take a uniform sample. These samples are then matched across all factors to form n design points such that a factor interval is not used more than once. We use a maximin LHS design [51] so that the distance between the closest pair of points is as large as possible. On the other hand, UD [41] provides uniformly scattered design points in the design space. It is similar to LHS, but LHS requires one-dimensional balance of all levels for each factor, while UD requires one-dimensional balance and p-dimensional uniformity. In this respect, the design points can be very different from LHS for high-dimensional problems [61].

2.3 Computationally Expensive Simulation Handling Methods

Computer simulations are widely used for solving real life problems. Using computer simulations becomes challenging when the problem (hence the input parameters) are high-dimensional and the simulation or analysis is computationally expensive [72].

There are strategies for handling design problems with combinations of these tackles.

2.3.1 Decomposition

Decomposition is reformulating the problem into a set of coordinated or independent sub-problems for reducing high dimensionality of input parameters to tackle "curse-of-dimensionality" [104]. There are three types of decomposition; product decomposition, process decomposition and problem decomposition [71].

Product decomposition partitions a product into physical components [47]. Process decomposition is applied to problems with flow of information, problems defined like a process [59]. Problem decomposition divides multidisciplinary problem models into different sub-problem models. Decomposition-based system design depends on problem decomposition. The general strategy is forming a design structure matrix [17] or functional dependency matrix [112] to reflect the dependency relationship between input parameters and components of the problem model to other components of the problem model. After that, the matrices are analyzed using methods like graph algorithms [82] or tree [21] algorithms. The matrix rows and columns are repositioned so that dependant rows and columns are brought together in diagonals. Then, the problem model can be solved as aggregation of sub-problem models.

Most of the time designers cannot have an ideal decomposition (i.e. there would be rows or columns apart from the clusters, namely coordinating variables, showing dependency to more than one clusters), then the optimization process will also include a coordination step [2]. The methodology is solving master problem to obtain optimal coordinating variables and solving subproblems to obtain optimal local variables and iteratively applying this process until optimization reaches convergence [112]. The problem with these techniques is that, the problem details needs to be well-analyzed so that the designer can efficiently identify dependencies. What is more, these techniques do not handle recurrent relationships between sub-models.

Decomposition is applied using problem specific information. For example, Konstantinidis et al. [69] decomposed the deployment and power assignment problem for wireless sensor networks into three subproblems. One subproblem is dedicated

to increasing network lifetime performance, other is dedicated to increasing coverage quality and final subproblem is dedicated to finding solutions with tradeoffs.

The procedure of solving decomposed individual disciplines or subsystems of a large multidisciplinary design problem is called decentralized design [92]. There are numerous approaches in the literature for using decentralized designs such as; game theory based approaches [97], collaborative optimization (CO) [16], concurrent subspace optimization [12] and analytical target cascading (ATC) [83]. Methodology is selected according to the characteristics of the problem. For example, game theory based approaches are preferred when there is little communication between the subsystem; CO is preferred when design is complex and discipline based analysis is required in subsystems and ATC is preferred for hierarchical architectural designs. The subsystems are dependent with each other through the coupled design variables.

Du and Chen [37] define external uncertainties as uncertainties in input parameters and internal uncertainties are uncertainties in models. They use interval methods for propagating the effect of uncertainties. In [36], Du and Chen define concurrent subsystem uncertainty analysis. They calculate mean and variance of the subsystems and propagated these in the system. Gurnani and Lewis [53] apply game theory based approach and defined a rational reaction set for subsystems. Kokkolaras et al. [68] extend ATC and define an advanced mean value method to generate the cumulative distribution function of subsystem non-linear response.

2.3.2 Metamodeling

Metamodeling is using model approximation techniques for reducing the computational complexity of functions [89]. Simulations are used when prototyping a real system is not practical or experimenting with the system has difficulties. If the simulation is complex and time consuming, then simpler approximations of the simulations can be constructed by using metamodels. If we do not know the simulation as a closed form definition, then the simulation is an input-output function and separate metamodels need to be built for each of the outputs [5].

Metamodel construction includes the following steps: [6]

1. Experimental design for sampling
2. Running the simulation at selected points
3. Constructing the metamodel
4. Validating the metamodel and repeating steps 1, 2 and 3 if necessary
5. Using the metamodel

The major issues in metamodeling are the choice of metamodeling method, sampling technique and validating the metamodel. Most metamodeling methods require the target function to be continuous and smooth. If the nature of the true function is not known a priori, it is not clear which metamodel will be the most accurate. In addition, there is no consensus on how to obtain the most reliable estimates of the accuracy of a given metamodel [113].

There are different techniques proposed for metamodeling of simulations. The first metamodeling technique is the usage of polynomial functions as response surfaces by Box and Wilson [15]. Cressie [28] proposes the use of a geo-statistical model, called kriging for interpolation of stochastic system responses. Other types of metamodeling methods include radial basis functions [38], support vector regression machines [35], bayesian approaches [24] and MARS [44].

2.4 Multivariate Adaptive Regression Splines

In the classical regression approach, a global parametric function is fitted to the available data to approximate the underlying relationship between the target (response) variable (y) and predictor variables (z). Although these methods are relatively easy to develop and interpret, they have limited flexibility and work well only in cases where the true underlying relationship is close to the pre-specified approximating function in the model. To overcome the shortcomings of global parametric approaches, non-parametric models, which do not have any distributional assumptions, have been developed locally over specific subregions of the data, and the data is searched to find the optimum number of subregions [120].

MARS is one of the nonparametric methods that uses a set of linear regressions. The nonlinearity of the model is approximated with different regression slopes in the corresponding intervals of each predictor. The intervals are closed and non-overlapping except for their boundaries, which are called *knots*. In other words, the slope of each regression line is allowed to change from one interval to another with the condition that there is a knot defined in between [44].

The intervals are defined using linear basis functions (BFs) of the form $(z-\tau)_+$ and $(\tau-z)_+$ such that:

$$(z-\tau)_+ = \begin{cases} z-\tau, & \text{if } z > \tau \\ 0, & \text{otherwise} \end{cases} \tag{2.1}$$

$$(\tau-z)_+ = \begin{cases} \tau-z, & \text{if } z < \tau \\ 0, & \text{otherwise,} \end{cases} \tag{2.2}$$

where τ is a knot value and z is a predictor for the MARS model. Assuming that z_{ij} is a knot for jth observation of ith predictor, where $i = 1, 2, ..., P$, $j = 1, 2, ..., N$, N is the total number of observations included in the regression and P is the total number of predictors, the collection of BFs for all predictors can be defined as:

$$B = \{(z_i-\tau)_+, (\tau-z_i)_+ | \tau \in \{z_{i1}, z_{i2}, ..., z_{i(N)}\}, i \in \{1, 2, ..., P\}\} \tag{2.3}$$

Then the MARS model is of the following form:

$$y = \beta_0 + \Sigma_{m=1}^{M}(\beta_m b_m) + \epsilon, \quad \text{and} \quad b_m \in B, \tag{2.4}$$

where y is the output (response) to be predicted, M is the number of BFs in B, β_0 is a constant term, β_m are a set of coefficients for the BFs and ϵ is an error term having zero mean and finite variance.

The MARS method generates a model in a two-stage process: forward and backward. In the first stage, MARS constructs a model with an extra large number of BFs, which deliberately overfits the data. These BFs represent distinct intervals of every predictor divided by knots, and in an intensive search, every possible knot location is tested. The MARS model is actually, in each dimension, a linear summation of certain BFs, and interactions among them if needed. Then, some of the BFs are removed as they contribute least to the overall performance. Therefore, the forward construction may

initially include insignificant model terms. In the backward pruning step, these terms are excluded. Thus, the backward step reduces the complexity of the model without degrading the fit to the data [44].

By allowing arbitrary shapes of BFs and their interactions, MARS has the capacity of reliably tracking very complex data structures that often hide in high dimensions [120]. Compared with other widely used modeling techniques such as multivariate linear regression models, regression trees and support vector machines, MARS has a better prediction accuracy [1, 3, 74]. The MARS method is also applied to several problems including the modeling of simulation data [4, 60, 75, 96, 118], weather forecasting [1, 120], ecological modeling [73], biomedical prediction [3, 32] and intrusion detection [88].

CHAPTER 3

RELATED WORKS

In this chapter, we give insight about the previous work on dynamic simulation metamodeling. Further, we present previous work on metamodeling of radar simulations.

3.1 Metamodeling of Dynamic Simulations

As stated in Chen et al. [22], metamodeling methods generally deal with univariate output variables, however, dynamic simulations require relating the inputs to a multivariate output (multi-output) which is a function of time-space coordinates. There are diverse approaches for metamodeling dynamic simulations. One such example is treating the time-space coordinates as additional input variables as in Zhang et al. [123], which results in a very large number of data points.

Bayesian network (BN) is a collection of random variables represented as network nodes while interconnecting arcs indicating dependency between nodes. Temporal Bayesian networks (TBN) account for temporal dependencies allowing dependency between nodes in different slices [14]. There are instant-based (time sliced) approaches and interval-based (event-based) approaches. Instant-based approaches divide the time into successive slices, and a node is associated with each time instant. To illustrate, DBN [31] is an instant-based TBN. Interval-based approaches divide the time line into disjoint time intervals. A node represents an output event of a state variable, and it can happen within a specific time interval. Interval-based discrete time BN (IBDTBN) [13] and continuous time BN (CTBN) [14] are examples of interval-based TBN.

A Markov process is a stochastic process that satisfies the Markov property and takes values from a set called the state space. The Markov property stipulates that at any times $s > t > 0$, the conditional probability distribution of the state of the process at time s given the whole history of the process up to and including time t, depends only on the state of the process at time t. In effect, the state of the process at time s is conditionally independent of the history of the process before time t, given the state of the process at time t.

A state space model (SSM) is a metamodel of D-dimensional real valued outputs (observations) Y assuming that at each time step, $Y_t, t \in \{1...T\}$ is generated from a hidden state variable X_t, where a sequence of X is a Markov process [48]. A hidden Markov model (HMM) is the discrete state counterpart of SSM. Bengio and Frasconi [9] propose an input-output HMM by forming a recurrent network of state variable nodes [10].

DBN [31] is a Bayesian network, which accounts for temporal dependencies allowing dependency between nodes in different slices. Poropudas and Virtanen [95] suggest using DBN as metamodels. They also give an example of DBN usage for time dependant state variable analysis [94]. They assume that state variable infrastructure is known, and an expert can build the structure beforehand. Weber and Jouffe [114] use DBN to create a dynamic fault tree for reliability modeling. Boudali and Dugan [14] use CTBN for reliability analysis of a system with known state variables.

Monte Carlo (MC) methods are a class of computational algorithms that rely on repeated random sampling to compute their results. Sequential MC methods (SMC), are model estimation techniques based on simulation outputs. SMC are used to estimate Bayesian models in which the state variables are defined as a SSM [19]. Particle filtering is an SMC method used for determining the distribution of a latent variable at a specific time, given all observations up to that time using a set of 'particles' (differently-weighted samples of the distribution). Doucet and Johansen [34] give a detailed tutorial about particle filters. Gu and Hu [52] apply particle filter method to a fire simulation based on cellular-DEVS formalism. They estimate the wind parameter by using simulation outputs.

RNN is a special type of ANN, which can be used for time series prediction [25]. One

disadvantage of RNN models is that like feed-forward networks, they very easily get trapped in local optimal solutions [29, 33]. Since the feedback does not contain a delay, the input values must be propagated many times through the network until the output reaches an equilibrium. This process of repeatedly applying the inputs is called internal recurrence, and it is required since the feedback value directly depends on the output of the hidden layer (without delay). The network may oscillate over many cycles, and there is no guarantee that it will reach stability; it may keep oscillating between the certain values [115].

Kennedy et al. [65] view the underlying code of black-box computer model as an unknown process and model it as a Gaussian process with a specified mean and covariance function. Bhattacharya [11] extends the dynamic emulators with the idea of forcing (dynamic) inputs. For continuous input-output spaces they evaluate the model on a fine grid, and interpolate within the resulting look-up table to obtain an approximate version of the dynamic sequence.

Liu and West [77] propose using time varying auto-regression series for the dynamic computer models (i.e. simulators that model a system evolving over time). They add current time (t) to the function definition and apply BN to this structure.

Conti et al. [26] use single time step emulators to model dynamic computer models. They argue that dynamic simulators operate iteratively over fixed time steps to model a system that is evolving over time. A single run of such a simulator typically consists of a simulation over many time steps, which can be considered as a simpler, single-step simulator being run iteratively many times. Conti and O'Hagan [27] compare the performance of multi-output emulator against a time input emulator, which accepts time as an extra input. The multi-output emulator has been found to be more efficient.

3.2 Previous Radar Simulation Metamodels

Ender et al. [40] propose a system of systems approach for the analysis of the ballistic missile defense system (BMDS). They develop an ANN metamodel to predict a sensor model and track its performance. Lum [80] uses response surface methodology to form a metamodel of an air combat model. The detection range data is collected

using an experimental design, then regressed to develop an equation to predict radar detection range.

Hall and Clark [54] present a simulation study in which they use linear interpolations of the bi-static radar development simulation to populate a static radar performance look-up table as a metamodel. Rodriguez [98] proposes the aggregation of high-resolution simulation models with low-resolution simulations. A mathematical representation of the simulation model based on network theory is presented and procedures are proposed for simulation aggregation using statistical techniques.

All these metamodels measure system performance using either the average detection performance of the sensor model over time or the detection performance on completion of the simulation. Their metamodels do not represent the time evolution of a simulation, hence they do not give information about the simulation outputs at specific time instances. In this work, we model a dynamic radar system model using a MARS based metamodel which can predict different outputs (responses) of a radar model during the course of a simulation run.

CHAPTER 4

DYNAMIC SIMULATION METAMODELING

In this chapter, we give detailed information about our dynamic simulation metamodeling approach, explain the used sample case and present the experimental studies made for selection of the metamodel.

4.1 The Methodology for Dynamic Simulation Metamodeling

A dynamic system generates outputs over time using the inputs and/or the state of the system. Accordingly, inputs may be fixed (known) design variables and/or time-based (dynamic) inputs. In order to simulate such a system, in this study, we adopt a discrete-event simulation approach with a fixed-increment time advance mechanism. Here, we assume that there are $p + q$ inputs, p of which are known and q of which are dynamic, and r outputs measured at ω different time points. We also assume state output is the same as the state at each instant. Then, the methodology consists of the following stages:

1. Gathering simulated data:
   - 1.1. Selecting design points,
   - 1.2. Running simulation at selected design points,
   - 1.3. Preparation of observation tables:
     - 1.3.1. *Complete observation table* which may include incomplete observations, to be used in stage 2.1,
     - 1.3.2. *Reduced observation table* to be used in stage 2.2,
2. Metamodeling the data obtained at stage 1 with MARS:

2.1. Developing metamodel(s) for predicting the existence of output(s) at each time step,

2.2. Developing metamodel(s) for predicting output(s).

In the first stage, the problem owner (or decision maker) specifies the range values of design variables to form a fixed independent variables' space, and design points are selected from this design space by utilizing sampling techniques such as those discussed in Section 2.2. Then, for each design point, the simulation is run and dynamic inputs and generated outputs are recorded. On completion of simulation runs these observations are recorded in the *complete observation table* in which the cells denote the values of design variables, dynamic inputs and outputs at ω time points for different samples (labeled as $SampleID$ in Table 4.1). Table 4.1 presents a sample complete observation table produced by running the simulation for n design points of p design variables, q dynamic inputs and r outputs.

As shown in Table 4.1, a dynamic simulation model generates output over time denoted by the set $Y = \{y_{it} | i = 1, 2, ..., r \text{ and } t = 1, 2, ..., \omega\}$ during the simulation. The output may be affected by a set of design variables $X = \{x_{ij} | i = 1, 2, ..., p \text{ and } j = 1, 2, ..., n\}$ and time-based inputs obtained during the course of the simulation $V = \{v_{it} | i = 1, 2, ..., q \text{ and } t = 1, 2, ..., \omega\}$. Hence, the metamodel should take both the design variables and dynamic inputs into account. At time-step t, the simulation model output may be written as a function of $Y = f(X, V, Y')$, where $y'_{it} = y_{it-1}, i = 1, 2, ..., r$ and $t = 1, 2, ..., \omega - 1$, is the output of the model at the previous step. In other words, the design variables of the model, the dynamic inputs at the current time step and the output at the previous time step are used to estimate the output at the current step.

Table 4.1 includes simulation runs at ω time steps for n different design points where $N = n \times \omega$ is the total number of rows in the observation table. However, for some time steps, the simulation model may receive no inputs, thus, generate no output. Hence, some of the values in the table may not be observed. We call this an *incomplete output process*. To handle this situatin, we prepare another observation table containing only the observed outputs, called *reduced observation table*. In other words, the *complete observation table* is reduced so that only the output occurring rows remain in the

Table 4.1: Layout of an example complete observation table

Sample ID	Design Variables (X)				Time	Dynamic Inputs (V)				Prev. Dyn. Outputs (Y')				Dynamic Outputs (Y)			
	x_1	x_2	...	x_p	t_e	v_1	v_2	...	v_q	y'_1	y'_2	...	y'_r	y_1	y_2	...	y_r
1	x_{11}	x_{21}	...	x_{p1}	t_1	v_{11}	v_{21}	...	v_{q1}	-	-	...	-	y_{11}	y_{21}	...	y_{r1}
					t_2	v_{12}	v_{22}	...	v_{q2}	y_{11}	y_{21}	...	y_{r1}	y_{12}	y_{22}	...	y_{r2}
					...	...	...	...	...	...	...	...	...	...	...	...	...
					$t_{\omega-1}$	$v_{1\omega-1}$	$v_{2\omega-1}$	...	$v_{q\omega-1}$	$y_{1\omega-2}$	$y_{2\omega-2}$	...	$y_{r\omega-2}$	$y_{1\omega-1}$	$y_{2\omega-1}$	...	$y_{r\omega-1}$
					t_ω	$v_{1\omega}$	$v_{2\omega}$	...	$v_{q\omega}$	$y_{1\omega-1}$	$y_{2\omega-1}$	...	$y_{r\omega-1}$	$y_{1\omega}$	$y_{2\omega}$	...	$y_{r\omega}$
2	x_{12}	x_{22}	...	x_{p2}	t_1	v_{11}	v_{21}	...	v_{q1}	-	-	...	-	y_{11}	y_{21}	...	y_{r1}
					t_2	v_{12}	v_{22}	...	v_{q2}	y_{11}	y_{21}	...	y_{r1}	y_{12}	y_{22}	...	y_{r2}
					...	...	...	...	...	...	...	...	...	...	...	...	...
					$t_{\omega-1}$	$v_{1\omega-1}$	$v_{2\omega-1}$	...	$v_{q\omega-1}$	$y_{1\omega-2}$	$y_{2\omega-2}$	...	$y_{r\omega-2}$	$y_{1\omega-1}$	$y_{2\omega-1}$	...	$y_{r\omega-1}$
					t_ω	$v_{1\omega}$	$v_{2\omega}$	...	$v_{q\omega}$	$y_{1\omega-1}$	$y_{2\omega-1}$	...	$y_{r\omega-1}$	$y_{1\omega}$	$y_{2\omega}$	...	$y_{r\omega}$
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
n	x_{1n}	x_{2n}	...	x_{pn}	t_1	v_{11}	v_{21}	...	v_{q1}	-	-	...	-	y_{11}	y_{21}	...	y_{r1}
					t_2	v_{12}	v_{22}	...	v_{q2}	y_{11}	y_{21}	...	y_{r1}	y_{12}	y_{22}	...	y_{r2}
					...	...	...	...	...	...	...	...	...	...	...	...	...
					$t_{\omega-1}$	$v_{1\omega-1}$	$v_{2\omega-1}$	...	$v_{q\omega-1}$	$y_{1\omega-2}$	$y_{2\omega-2}$	...	$y_{r\omega-2}$	$y_{1\omega-1}$	$y_{2\omega-1}$	...	$y_{r\omega-1}$
					t_ω	$v_{1\omega}$	$v_{2\omega}$	...	$v_{q\omega}$	$y_{1\omega-1}$	$y_{2\omega-1}$	...	$y_{r\omega-1}$	$y_{1\omega}$	$y_{2\omega}$	...	$y_{r\omega}$

table. The reduction improves the prediction performance, since the observation rows without any output are useless for predicting the output values.

In the second stage of the methodology, developing a metamodel to generate output for any time step is inspired by the idea of multi-output emulator by Conti et al. [26]. We metamodel the dynamic simulation model by correlating the current outputs with the current inputs as well as the previous outputs.

In our study, we use MARS for three reasons: i) MARS is efficient in problems with large number of inputs and huge training datasets; ii) MARS is non-parametric (does not require any distributional assumptions regarding the model or the model parameters), thus, can be used without any knowledge of the problem characteristics, and iii) MARS is flexible enough to model complex problems (i.e., involving large number predictors with nonlinear relations).

In addition, our approach can handle an incomplete output process in which the simulation may not generate output at some steps. This is particularly useful for cases where output generation depends on the dynamic inputs of the simulation model (e.g. sensor generates output when it senses an object). We also use the previous output as a predictor. During the preparation of the observation dataset, for each time step of the observation dataset, we include the one-step previous output of the simulation model as extra dynamic inputs to the metamodel (instead of providing all previous outputs for each step). Thus, we guide the metamodel to detect and incorporate the correlation only between the current and one-step previous outputs.

In the metamodeling stage since the output process may be incomplete as mentioned above, a two-stage metamodeling scheme is devised; while the first metamodel decides *whether to output at this time step*, the second one predicts *what to output*. In developing the first metamodel (*OutputControl* metamodel), the complete observation table prepared at stage 1.3.1 is used. For developing the second metamodel *Output* metamodel, the reduced table prepared at stage 1.3.2 is utilized. The *OutputControl* model is used to predict whether the simulation model generates output at this step. If it does, then *Output* metamodel is used to predict the output value.

Using the observation table (Table 4.1), the MARS model in (2.4) can be fitted. In equation 2.4, z_{ij} is a predictor for the MARS model, where $z_{ij} \in X \cup V \cup Y'$ and z_{ij} is the knot for the jth observation of the ith predictor where $i = 1, 2, ..., P$, $j = 1, 2, ..., N$, $P = p + q + r$. Here, X, V and Y' are the same as defined above.

It is worth to mention here that implementation of MARS method involves some technical difficulties. A simulation model can have any type of inputs or outputs such as string, list, matrix, double, integer or enumeration, but available MARS tools only accept numeric data such as integer and double or categorical data. To resolve this mismatch, an input data with a complex structure is split into single pieces and each component is treated as a separate independent variable. If an output has a complex structure, then the response is also split into single pieces. Thus, before the training stage, the columns of both (complete and reduced) observation tables are split into columns that are primitive data types (e.g., integer, double, float, enumeration).

4.2 The Case: Metamodeling Dynamic Radar Simulation

In this study, the methodology developed for dynamic simulation metamodeling with MARS in the previous section, is implemented for a radar simulation model against an aircraft. The simulation model is dynamic, because the radar model in the simulation uses the current location and signal information for performing target detection analysis and generates radar track information over time during the simulation. If the radar detects a platform two out of three times in three consecutive attempts (i.e., time steps), it issues a detection event.

In the simulation, the purpose of the user is to find the best radar configuration parameters for earlier, more accurate and precise detection. The user can control the frequency, transmit power and bandwidth of the radar system model. The aircraft model's flight way points are specified before the simulation.

Thus, the radar simulation model has three design variables, two dynamic inputs and one output. As shown in Table 4.2, the inputs include the host platform information (*RadarPlatformInfo*) and the signal information from the target (*TargetSignal*), while output is the detection information (*TargetDetection*).

Table 4.2: Inputs and outputs of the radar simulation model

Input/Output	Set Label	Variable	Name
Input: Design Variables	X	x_1	RadarFrequency (GHz)
		x_2	RadarPower (dBW)
		x_3	RadarBandwidth (MHz)
Input: RadarPlatformInfo	V	v_1	Position $\langle sposx, sposy, sposz \rangle$ (m)
		v_2	Velocity $\langle svelx, svely, svelz \rangle$ (m/s)
Input: TargetSignal	V	v_3	TargetPosition $\langle tposx, tposy, tposz \rangle$ (m)
		v_4	TargetVelocity $\langle tvelx, tvely, tvelz \rangle$ (m/s)
		v_5	TargetRCS (W/m^2)
Output: TargetDetection	Y	y_1	TargetDetectedPosition $\langle dx, dy, dz \rangle$ (m)

The radar simulation model is implemented in accordance with the propagation model recommendations of the International Telecommunications Union (ITU): ITU-R P.452 (The prediction procedure for the evaluation of interference at frequencies above about 0.1 GHz), ITU-R P.525 (Free space attenuation), ITU-R P.526 (Propagation by diffraction), ITU-R P.676 (Attenuation and ducting by atmospheric gases) and ITU-R P.1407 (Multi-path propagation and parameterizations of its characteristics).

4.2.1 Training the metamodel

As stated in the previous section, the radar simulation model is dynamic and generates output over time (e.g., radar track information) during the simulation. The output is affected by the design variables (e.g., radar power), and time-based inputs from other models during the simulation (e.g., position of the target aircraft).

The radar model generates output only when there is a detection. The output is the position of the detected target. During the training stage, two observation tables are prepared as explained in Section 4.1 and used in training by means of the formula in (2.4). The *OutputControl* metamodel is trained using the complete observation table using all the rows at once. Then, *Output* metamodel is trained using the reduced observation table (e.g., time steps in which the radar detects the aircraft).

4.2.2 Metamodel coordinate system representations

The observations include the position information of the radar and the aircraft. During the training stage, the position information of both predictors and responses are split into components. Although a MARS model can undertake the feature selection and interaction analysis between predictors, the coordinate system is modified before the training to investigate the coordinate system's effect on the metamodel performance.

For this purpose, we use three different coordinate systems. Firstly, the observations are used as they are. By default the simulation uses the cartesian coordinate system, so the acronym *CRTSN* is used for this representation. Secondly, the radar system uses range, azimuth and elevation information during the detection analysis. The coordinate system of radar, aircraft and target detection is converted from a cartesian to polar coordinate system before the training, leading to a polar (*POLAR*) representation. Thirdly, aggregation operators are used for reducing the number of predictors and increasing the metamodel performance [18]. For this purpose, several mean operators are considered, and based on the results, the geometric mean of the position information is selected, and called, *CMBND*. Evidently, this representation has no physical interpretation. The predictions are all converted back to the cartesian coordinate system to obtain performance measures that can be compared.

4.3 Experimental Study for Metamodel Selection

In this section, we present the experimental studies made for selection of the metamodel by comparing the sampling techniques and discussing the results.

4.3.1 Training and Test Data

Four sampling techniques are used in our experiments: AFD and HD, UD and LHS. In order to analyze the effect of the number of observations (i.e. sample size) on the overall performance, several sample sizes are tried, which are primarily chosen for the AFD and applied to the UD and LHS sampling techniques for easy comparison. In this study HD is used with three factors and sample size 11. The full list of sample

sizes used in this study are: $\{4, 6, 8, 10, 11, 12, 15, 18, 24, 30, 36, 40, 42, 45, 48, 50\}$.

The simulation is run during the final phase of the engagement of the aircraft with the ship, which takes roughly 10 seconds with 0.1 second time steps (fixed time increment) for each combination of the parameters. For a single design point, we have 100 ($= 10/0.1$) observations, which increase as the sample size increases and becomes 5000 ($= 50 \times 10/0.1$) for the largest sample size.

The design variables of the radar are defined as follows:

i. Radar Frequency(GHz) : $[1.0, 8.0]$

ii. Radar Transmit Power(dBW) : $[50.0, 80.0]$

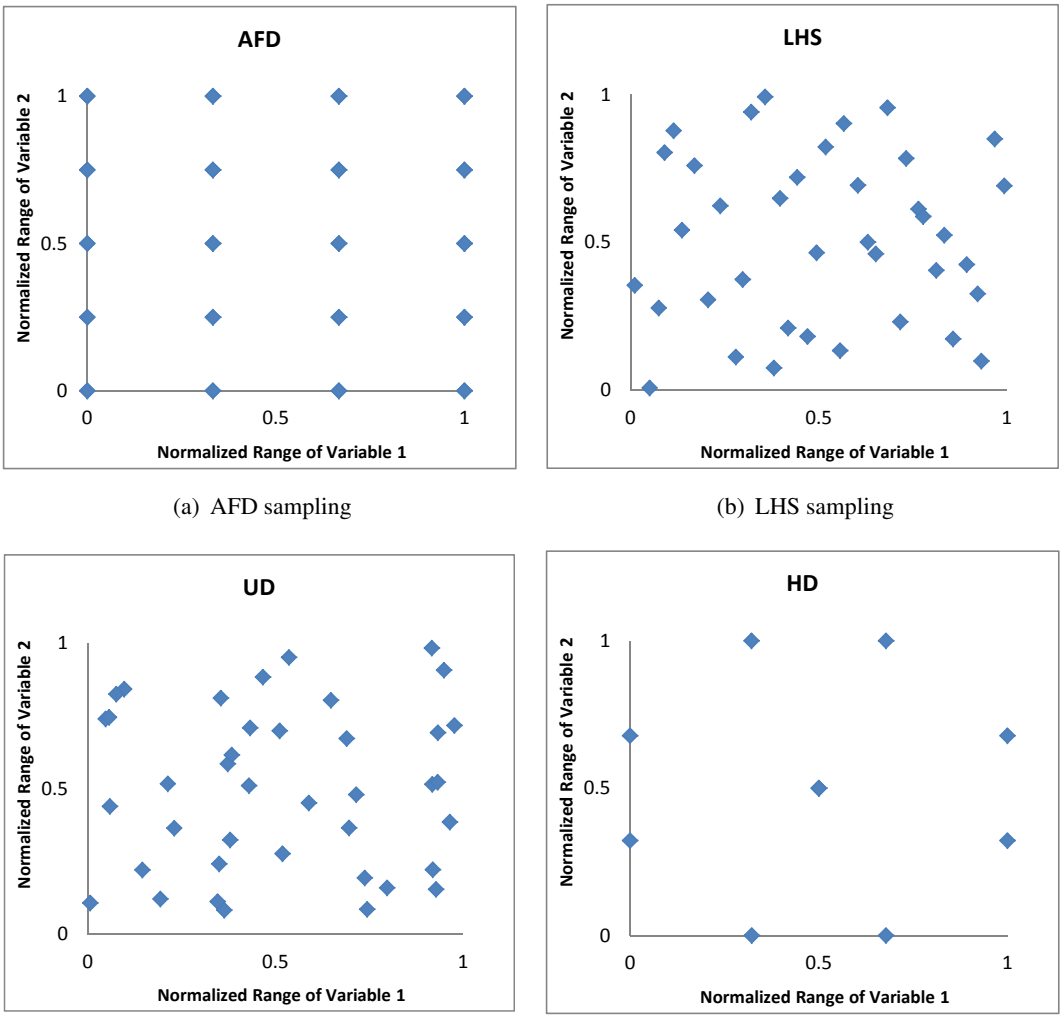
iii. Radar Bandwidth(MHz): $[0.4, 1.0]$

For each of the determined sampling techniques and sample sizes, the design points (i.e. the levels of design variables) are determined, and the simulation is run for all design points. Figure 4.1 shows the representative distributions of the design points in a cross section of the design space. After the simulation runs, the inputs and outputs of the radar simulation model are recorded at each time step to build the *complete observation table*. In order to investigate the correlation between the coordinate system representations and metamodeling performance, the three coordinate systems explained in Section 4.2.2 are applied to the observations before training the metamodels mentioned.

The test data set is generated with UD sampling (with different random seed from training) and a sample size of 50. For every combination of sampling technique, coordinate system representation and sample size, we use the same data set to test the performance of the metamodel.

4.3.2 Performance Measures

The performance of the developed MARS metamodels is evaluated with respect to these three criteria: accuracy, complexity and stability. The measures used for these criteria, their meanings, interpretations and formulas are presented in Table 4.3. The measures are calculated for each output of the model and the notation y_{kt} specifies the



(a) AFD sampling

(b) LHS sampling

(c) UD sampling

(d) HD sampling

Figure 4.1: Design point distribution in a cross section of sampling techniques used

output value during the simulation run for design point k and time step t. Robustness of the methods under different data features is also assessed with the help of the spread of performance measures used. For this purpose, median absolute deviation (MAD) of the measures is utilized.

4.3.3 Applications and Findings

The holdout method is applied to each of the four sampling techniques for the sample sizes given in Section 4.3.1. As explained in Section 4.1 two distinct metamodels are used, one to predict the occurrence of a detection (i.e., whether to output) called *OutputCntrl* metamodel and the other for predicting the detection location (i.e., what to output) called *Output* metamodel.

MARS models are built using the stand-alone *Earth* package ($v3.2 - 6$) for the R environment [109]. After the models are built, the comparison measures listed in Table 4.3 are calculated for training and testing samples as well as the stabilities of the measures. An example MARS metamodel BF list (knots) for UD sampling with CMBD coordinate system is presented in Appendix section, Tables 7.1-7.2.

Run time is an important concern for computationally expensive simulations. For that reason, we strive to determine the smallest sample size for sampling techniques such that the performance measures of the model are relatively better and more resolute. Figures 4.2 and 4.3 show training, test and stability performances of different coordinate system representations for different sample sizes. For each sample size, the median of each output and each measure obtained from running different sampling techniques are plotted. To obtain the average, the median is preferred because of the existence of outliers. The values in the charts for sample size 11 are the results obtained using the HD sampling.

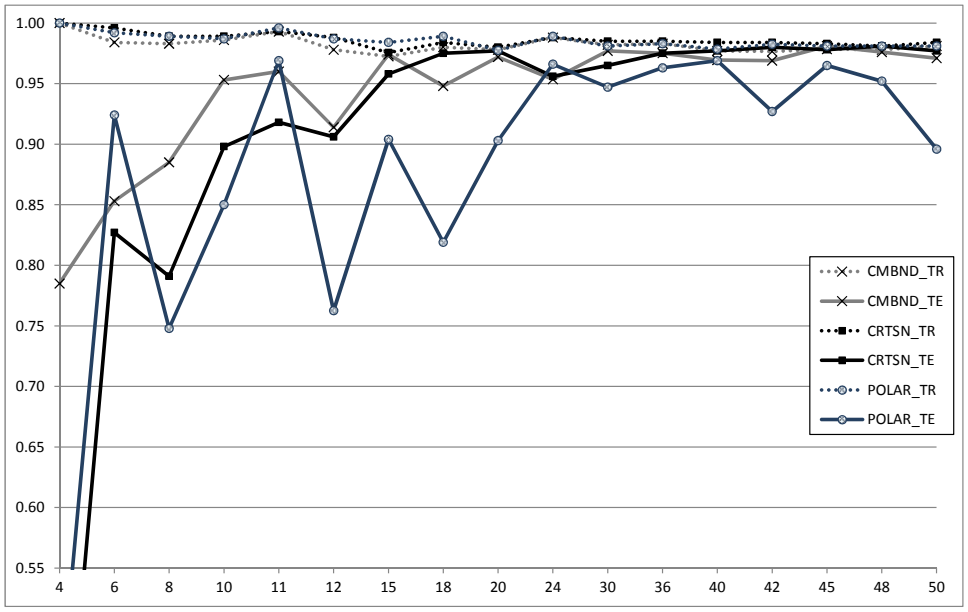
When these plots are closely examined, the following conclusions can be drawn:

i. The three representations perform well and behave similarly with respect to the four measures under consideration as the sample size increases for the training data.

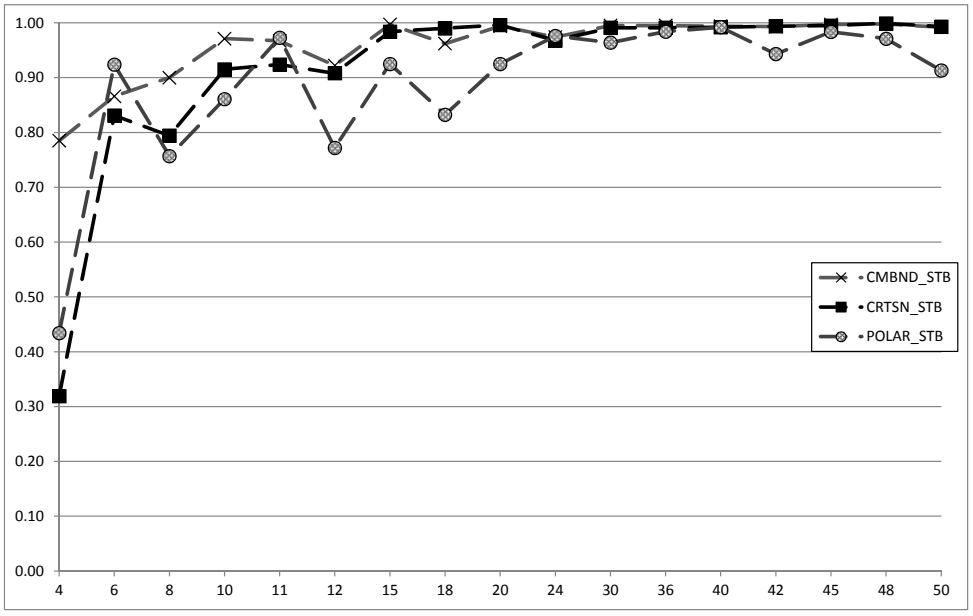
Table 4.3: Prediction performance criteria and measures used

Criterion	Abbr.	Measure	Explanation	Interpretation	Formula
Accuracy	R^2	Adjusted Coefficient of Determination	Percentage of variation in response explained by the model	Values closer to one (1) are better	$R^2 = 1 - (1 - \frac{\sum_{k=1}^{n}\sum_{t=1}^{\omega}(y_{kt}-\widehat{y}_{kt})^2}{\sum_{k=1}^{n}\sum_{t=1}^{\omega}(y_{kt}-\overline{y})^2})(\frac{N-1}{N-P-1})$
	r	Correlation coefficient	Straight line relation between observed and predicted response	Values closer to one (1) are better	$r(y,\widehat{y}) = \frac{\sum_{k=1}^{n}\sum_{t=1}^{\omega}(y_{kt}-\overline{y})(\widehat{y}_{kt}-\overline{\widehat{y}})}{(N-1)s_{\widehat{y}}s_y}$, where $s_{\widehat{y}}$ and s_y are the sample standard deviations of $\widehat{y}$ and y
	MAE	Mean Absolute Error	Average magnitude of errors	Smaller values are better	$MAE = \frac{1}{N}\sum_{k=1}^{n}\sum_{t=1}^{\omega}\lvert y_{kt} - \widehat{y}_{kt}\rvert$
Complexity	MSE	Mean Squared Error	Average of the squared errors by regarding the number of terms in the model	Smaller values are better	$MSE = \frac{1}{N-P-1}\sum_{k=1}^{n}\sum_{t=1}^{\omega}(y_{kt} - \widehat{y}_{kt})^2$
Stability		Stability of measure	Compares the performance of a method on training and test data sets	Values closer to one (1) are better	$min\{\frac{CR_{TR}}{CR_{TE}}, \frac{CR_{TE}}{CR_{TR}}\}$, where CR_{TR} is the performance of the measure in training data set and CR_{TE} is the measure's performance on test data set

Notes: n is the is the number of samples in the sampling, ω is the number of time steps for each simulation run, $N = n \times \omega$ is the total number of rows in the observation table and P is the total number of predictor variables in the model, y_{kt} is the actual output from the dynamic system and $\widehat{y}_{kt}$ is the predicted output from the metamodel

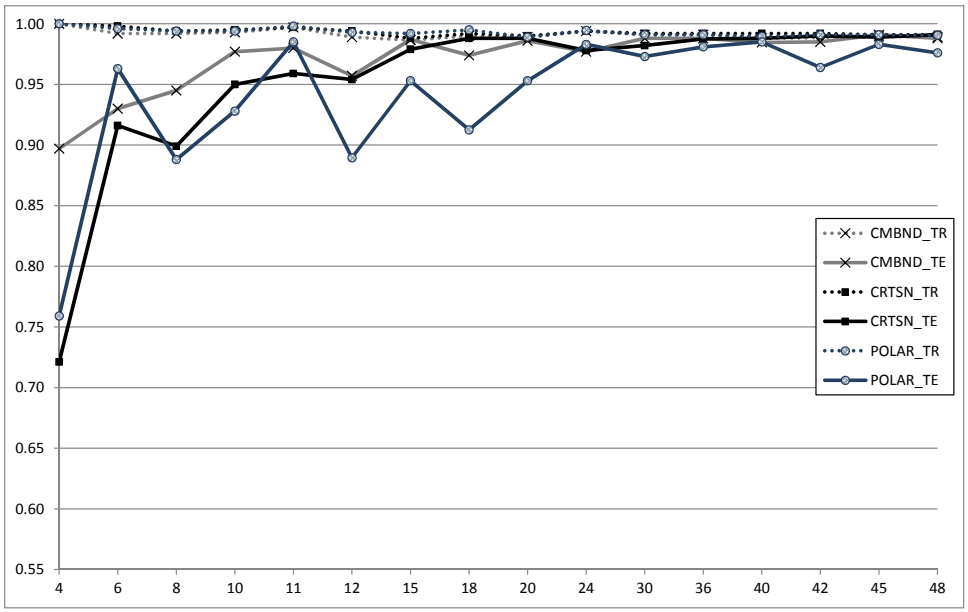


(a) R^2 training and test performance

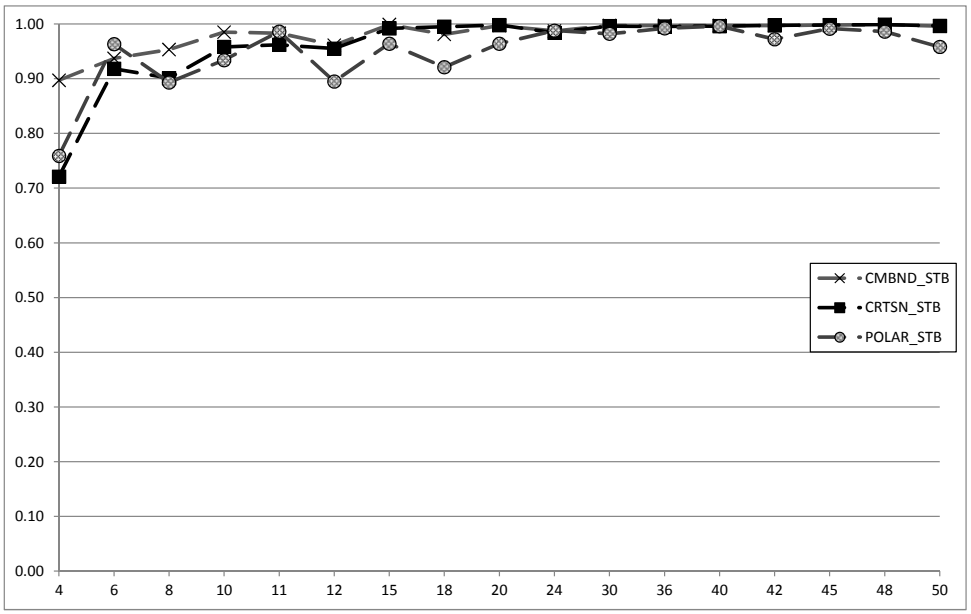


(b) R^2 stability performance

Figure 4.2: R^2 performances with respect to different sample sizes

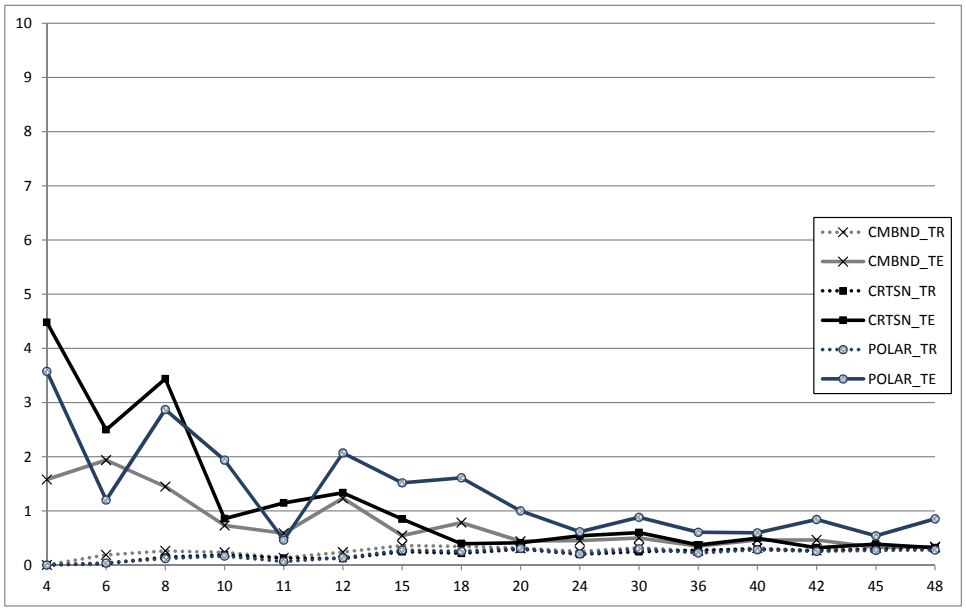


(a) r training and test performance

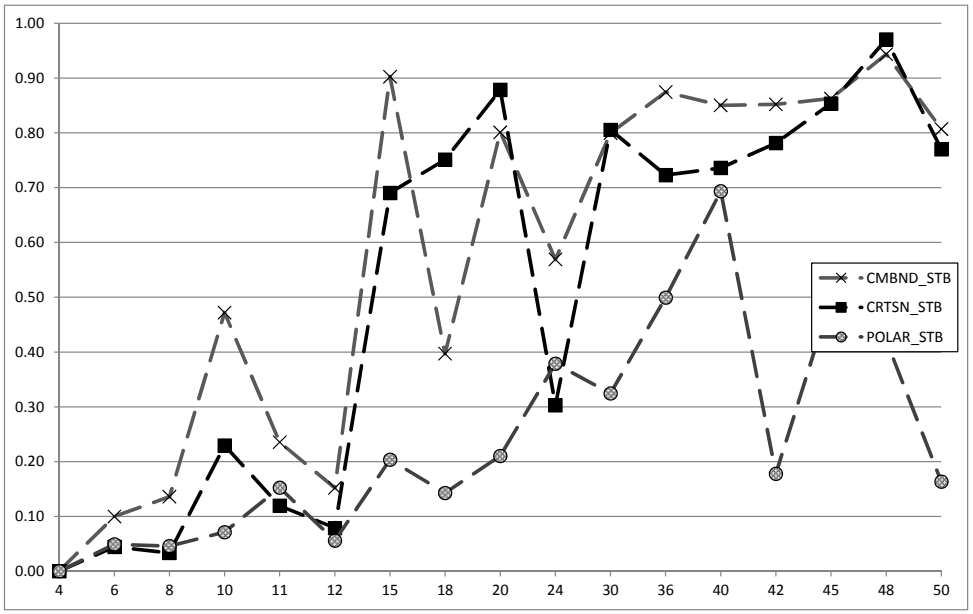


(b) r stability performance

Figure 4.3: r performances with respect to different sample sizes

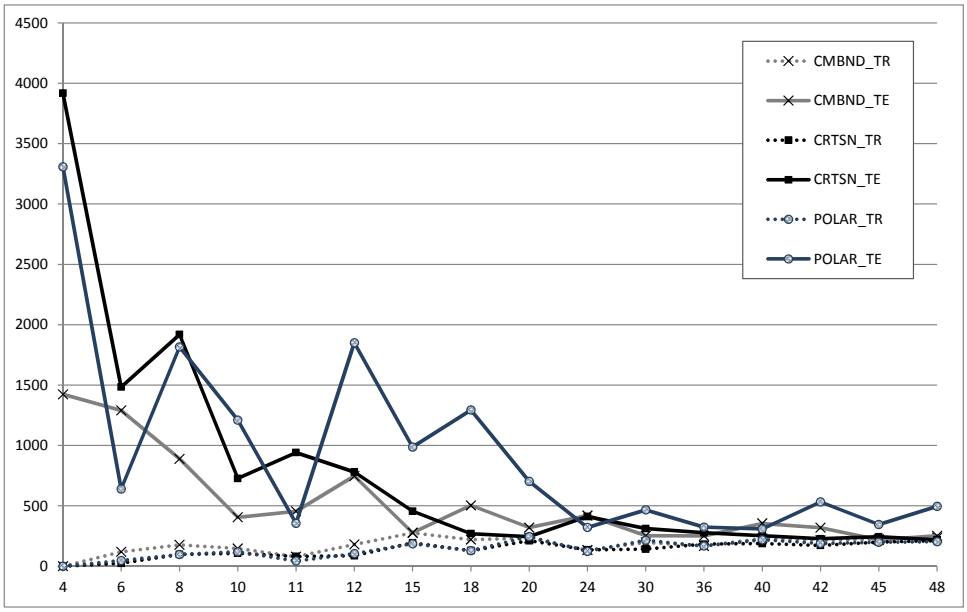


(a) *MAE* training and test performance

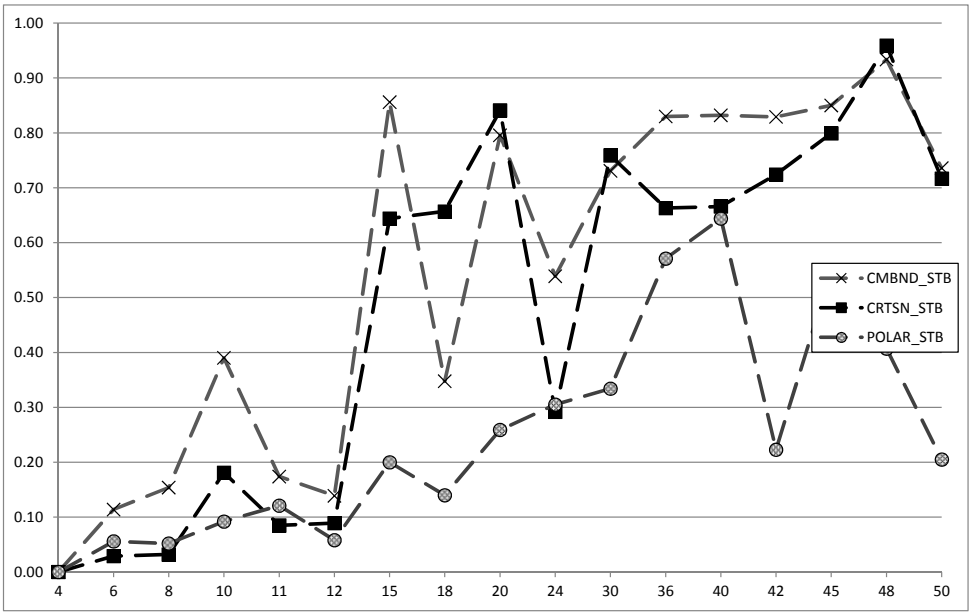


(b) *MAE* stability performance

Figure 4.4: *MAE* performances with respect to different sample sizes



(a) *MSE* training and test performance



(b) *MSE* stability performance

Figure 4.5: *MSE* performances with respect to different sample sizes

ii. As expected, the performance of methods for the test data are not as good as to those of training data, especially for small sample sizes. As the sample size increases, the performance of the methods approaches to that of the training data with respect to all measures under consideration.

iii. The stability of the R^2 and r measures are very high and almost the same for all three representations for $n \geq 24$ where n is the sample size. However, this is not the case for the *MAE* and *MSE* measures since their stabilities fluctuate more as sample size increases.

iv. *MAE* and *MSE* stabilities are considerably lower in comparison to those of R^2 and r.

v. According to the test data and stabilities of the measures, POLAR representation has the worst performance results, particularly when $n \geq 15$ for *MAE* and *MSE* and when $n \leq 24$ for R^2 and r. Interestingly, it performs even better than the other representations when combined with HD with a sample size of 11.

vi. When $n \geq 15$, stabilities of *MAE* and *MSE* for the two representations other than the POLAR sharply increase with large deviations and become relatively regular for $n \geq 30$.

The plots indicate that, apart from the POLAR representation results, which are in general worse than the other representations, a sample size of 15 yields relatively better performances for AFD, LHS and UD. Therefore, 15 is chosen as our candidate sample size for these three samplings and together with a sample size of 11 for the HD. The samplings and representations are analyzed on this basis. Table 4.4 shows overall results for different coordinate representations and sampling techniques.

4.3.4 Results

The performance results of metamodels with respect to three coordinate system representations and four sampling techniques for selected sample sizes (15 for AFD, LHS, UD and 11 for HD) are presented in Tables 4.5 and 4.6, respectively. These tables show the median and median absolute deviation (MAD) of the training, test and stability results for our measures. These statistics are specifically selected since they

Table 4.4: Performance of metamodels for the selected sample sizes

			Training		Test		Stability	
			OutputCntrl	Output	OutputCntrl	Output	OutputCntrl	Output
CMBND	AFD[1]	R^2	0.978	0.974	0.968	0.964	0.991	0.989
		r	0.989	0.986	0.985	0.984	0.998	0.999
		MAE	0.005	1.228	0.006	1.412	0.900	0.912
		MSE	0.005	336.251	0.006	403.317	0.895	0.902
	LHS[1]	R^2	0.973	0.968	0.978	0.915	0.961	0.945
		r	0.987	0.984	0.968	0.958	0.981	0.973
		MAE	0.006	1.486	0.014	3.708	0.429	0.401
		MSE	0.006	370.573	0.014	974.290	0.431	0.380
	UD[1]	R^2	0.981	0.981	0.981	0.978	1.000	0.997
		r	0.991	0.990	0.990	0.989	1.000	0.999
		MAE	0.004	0.938	0.004	1.038	0.952	0.903
		MSE	0.004	216.100	0.004	252.321	0.958	0.856
	HD[2]	R^2	0.992	0.993	0.969	0.960	0.978	0.967
		r	0.996	0.997	0.985	0.980	0.989	0.983
		MAE	0.002	0.400	0.007	1.743	0.275	0.229
		MSE	0.002	79.285	0.007	455.089	0.278	0.174
CRTSN	AFD	R^2	0.988	0.991	0.890	0.869	0.901	0.877
		r	0.994	0.995	0.947	0.936	0.952	0.941
		MAE	0.003	0.545	0.023	5.817	0.116	0.094
		MSE	0.003	112.010	0.023	1473.027	0.117	0.076
	LHS	R^2	0.975	0.973	0.969	0.972	0.991	0.996
		r	1.000	0.999	1.000	0.999	1.000	1.000
		MAE	0.006	1.358	0.008	1.542	0.732	0.837
		MSE	0.006	266.568	0.008	326.534	0.738	0.859
	UD	R^2	0.978	0.975	0.971	0.963	0.994	0.988
		r	0.989	0.988	0.986	0.982	0.997	0.994
		MAE	0.005	1.121	0.006	1.610	0.753	0.696
		MSE	0.005	274.390	0.006	421.965	0.760	0.650
	HD	R^2	0.992	0.993	0.942	0.918	0.950	0.924
		r	0.996	0.997	0.971	0.959	0.975	0.962
		MAE	0.002	0.384	0.013	3.410	0.147	0.113
		MSE	0.002	79.803	0.013	941.175	0.149	0.085
POLAR	AFD	R^2	0.985	0.984	0.875	0.851	0.888	0.864
		r	0.993	0.992	0.940	0.927	0.947	0.935
		MAE	0.003	0.790	0.026	6.593	0.128	0.120
		MSE	0.003	195.212	0.026	1677.305	0.129	0.116
	LHS	R^2	0.982	0.977	0.909	0.904	0.925	0.925
		r	0.991	0.989	0.955	0.953	0.964	0.964
		MAE	0.004	1.036	0.020	4.507	0.206	0.230
		MSE	0.004	271.965	0.020	1086.720	0.208	0.250
	UD	R^2	0.984	0.984	0.925	0.921	0.940	0.936
		r	0.992	0.992	0.963	0.961	0.971	0.969
		MAE	0.003	0.762	0.016	3.764	0.208	0.202
		MSE	0.003	177.017	0.016	886.912	0.210	0.200
	HD	R^2	0.996	0.996	0.975	0.969	0.979	0.973
		r	0.998	0.998	0.988	0.985	0.990	0.986
		MAE	0.001	0.201	0.005	1.377	0.168	0.146
		MSE	0.001	42.902	0.005	354.813	0.171	0.121

[1] AFD, LHS and UD results are taken with sample size of 15

[2] HD results are taken with sample size of 11

are resistant to outliers. For R^2 and r, higher median values, and for *MAE* and *MSE* lower median values indicate better performances for training and test data. On the other hand, the median stabilities that are close to one reveal better performances for the corresponding measures. Here it should be noted that lower median *MSE* values indicate less complex methods, and lower MAD values indicate more robust methods.

4.3.4.1 Performance with respect to representations

Table 4.5 shows the overall results for performance with respect to coordinate systems, in terms of the median and MAD values of the sampling technique.

According to the results:

i. For the training data set, the POLAR and CRTSN representations give best results for the R^2 and r measures, while CRTSN representation has better results for *MAE* and *MSE* measures. POLAR and CRTSN representations are also the most robust.

ii. For the test data set, the CMBND and POLAR representations yield the best and worst results, respectively, with respect to all measures.

iii. The CMBND representation is more stable than the other coordinate systems; its stability is more robust with respect to the R^2 and r measures. Furthermore, it also seems to produce less deviation in results.

4.3.4.2 Performance with respect to sampling techniques

Table 4.6 shows the results of the metamodel with respect to the sampling techniques, in terms of the median and MAD values of the coordinate systems. From the results the following conclusions can be drawn:

i. For the training data set, HD performs better and is also the most robust, while AFD and UD have a worse performance than the other samplings.

ii. For the test data set, UD and HD give comparable and better results than the other samplings in terms of the R^2 and r measures, while LHS is better than the

Table 4.5: Performance of metamodels with respect to representations for the selected sample sizes

		Training			Test			Stability		
		CMBND	CRTSN	POLAR	CMBND	CRTSN	POLAR	CMBND	CRTSN	POLAR
R^2	Median	0.980	0.983[a]	0.984[a]	0.967[a]	0.941	0.937	0.987[a]	0.956	0.954
	MAD	0.005	0.009	0.001[b]	0.009[b]	0.026	0.024	0.011[b]	0.036	0.019
r	Median	0.994	0.996[a]	0.995[a]	0.988[a]	0.971	0.973	0.998[a]	0.978	0.978
	MAD	0.005	0.002	0.003[b]	0.005[b]	0.020	0.019	0.001[b]	0.019	0.016
MAE	Median	0.683	0.518[a]	0.560	1.125[a]	2.510	2.2571	0.191[a]	0.301	0.041
	MAD	0.269	0.081[b]	0.216	0.192[b]	1.412	1.451	0.191	0.301	0.041[b]
MSE	Median	147.693	95.907[a]	109.960	277.819[a]	681.570	620.863	0.673[a]	0.368	0.042
	MAD	87.151	30.970[b]	71.547	101.384[b]	441.328	395.334	0.206	0.287	0.042[b]

[a] indicates better values with respect to median

[b] indicates better values with respect to MAD

others in terms of the *MAE* and *MSE* results. AFD is significantly worse than the others.

iii. In terms of the stability, UD is the best and AFD is the worst, with the HD having more robust stability.

4.3.4.3 Interactions between representations and sampling techniques

We also investigate interactions among sampling techniques and coordinate system representations. Figures 4.6 and 4.7 show the median of output measure performance changes with respect to sampling and representation interactions.

The results can be interpreted as follows:

i. According to the R^2 and r measures, test and stability performances indicate that HD performs better when it is used together with the POLAR representation, but the others perform better with CMBND.

ii. *MAE* and *MSE* test and stability performances confirm that HD-POLAR combination performs better than the HD-CMBND combination.

iii. *MAE* and *MSE* training and stability performance results show that AFD, LHS and UD perform better when they are used with the CRTSN representation and have worse performance when used with POLAR. Furthermore, AFD has best results with CMBND representation and LHS has best results with CRTSN.

4.3.5 Discussion

In this section, some results that might need more explanation are discussed further in the light of the above findings. In this work, we first find the minimum number of observations (i.e. sample size) required to develop a metamodel with a good prediction performance. According to the results, the LHS-CRTSN and UD-CMBND combinations with sample sizes of 15 yield the best results together with the HD-POLAR combination with a sample size of 11.

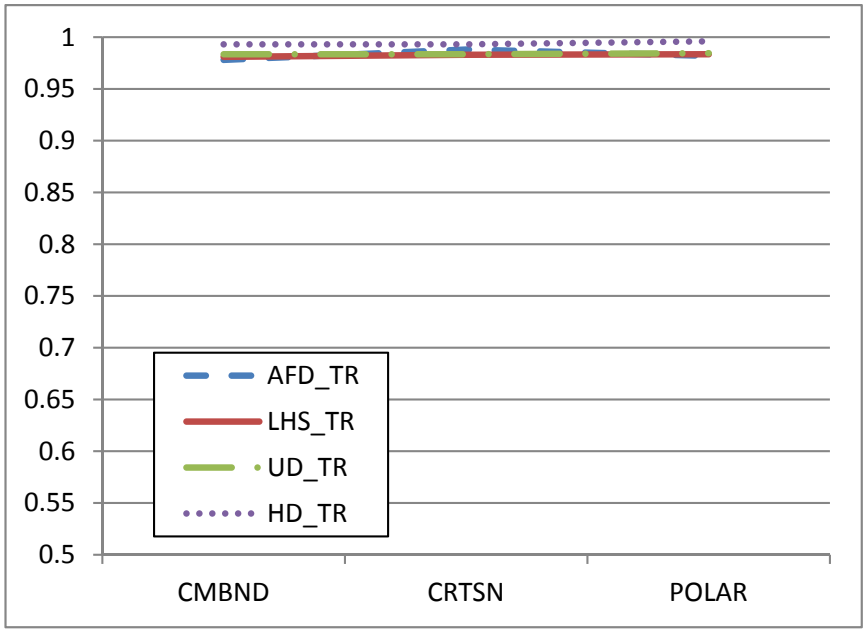
In general, AFD does not produce steady results, as opposed to LHS, UD and HD. This may be due to the uneven distribution of the design points in the AFD sampling

Table 4.6: Performance of metamodels with respect to sampling techniques for selected sample sizes

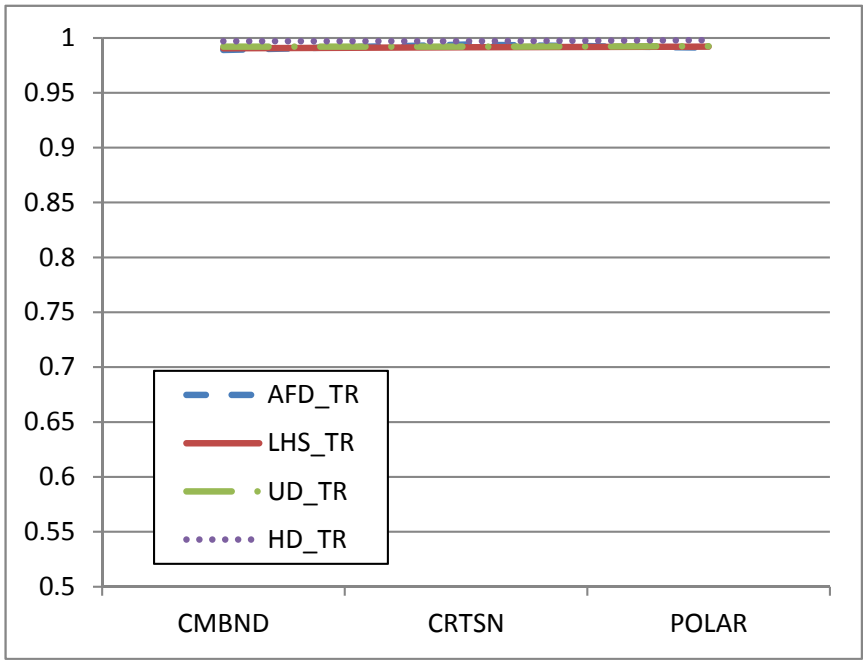
		Training				Test				Stability			
		AFD	LHS	UD	HD	AFD	LHS	UD	HD	AFD	LHS	UD	HD
R^2	Median	0.984	0.978	0.981	0.993[a]	0.869	0.956	0.963[a]	0.960	0.877	0.945	0.988[a]	0.967
	MAD	0.007	0.004	0.003	0.001[b]	0.018	0.002	0.015	0.009[b]	0.013	0.02	0.009	0.006[b]
r	Median	0.992	0.998[a]	0.990	0.997	0.936	0.998[a]	0.982	0.980	0.941	0.973	0.994[a]	0.983
	MAD	0.003	0.001[b]	0.002	0.001[b]	0.009	0.001[b]	0.007	0.005	0.006	0.009	0.005	0.003[b]
MAE	Median	0.790	0.427	0.938	0.384[a]	1.952	1.428[a]	1.610	1.743	0.121	0.530	0.696[a]	0.146
	MAD	0.245	0.0635	0.176	0.016[b]	0.776	0.034[b]	0.572	0.366	0.026[b]	0.0985	0.207	0.033
MSE	Median	195.212	41.799	216.100	39.285[a]	1473.027	391.772[a]	421.965	455.089	0.116	0.491	0.650[a]	0.121
	MAD	83.202	7.8755	39.083	0.518[b]	204.278	112.43	169.644	100.276[b]	0.04	0.13	0.206	0.036[b]

[a] indicates better values with respect to median

[b] indicates better values with respect to MAD

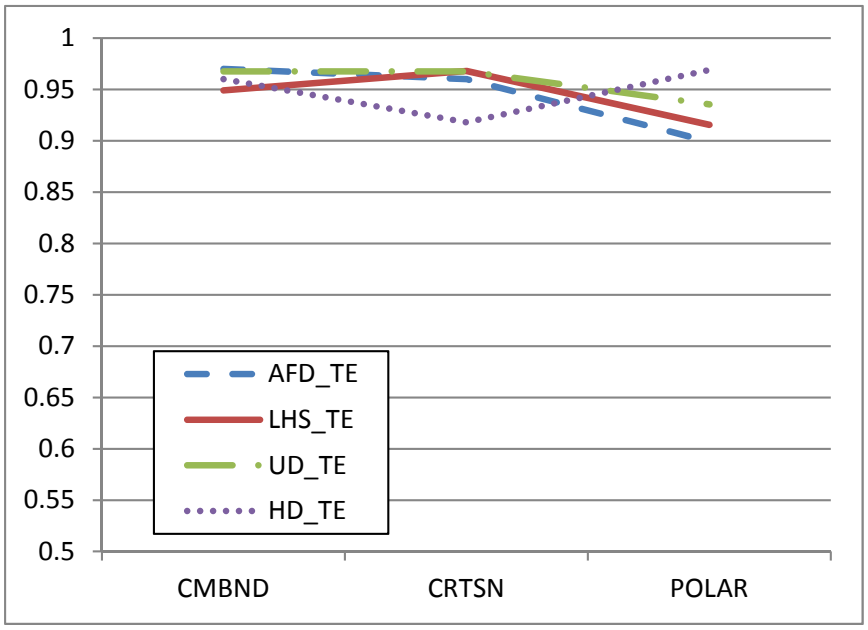


(a) R^2 training interaction

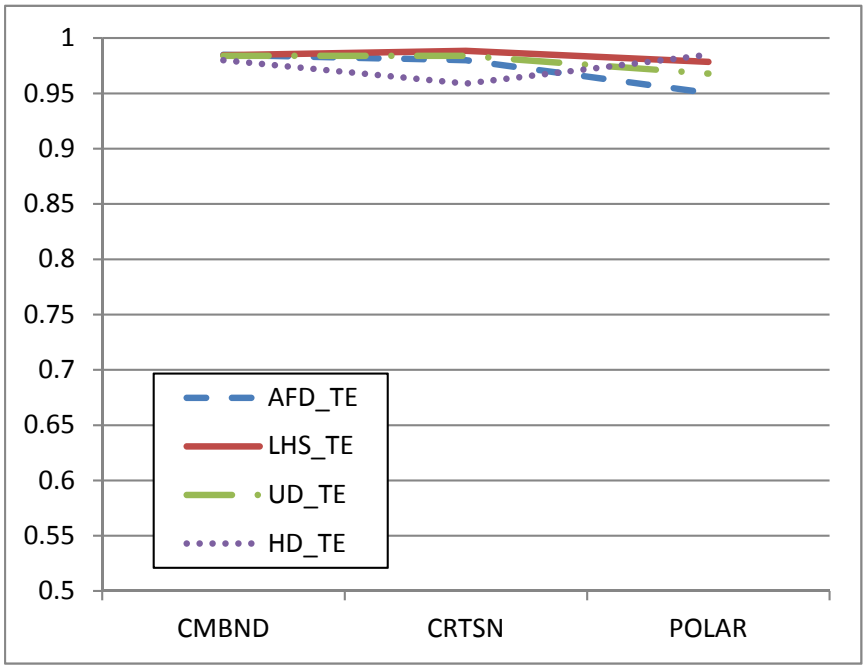


(b) r training interaction

Figure 4.6: Sampling-representation training interaction of R^2 and r measures

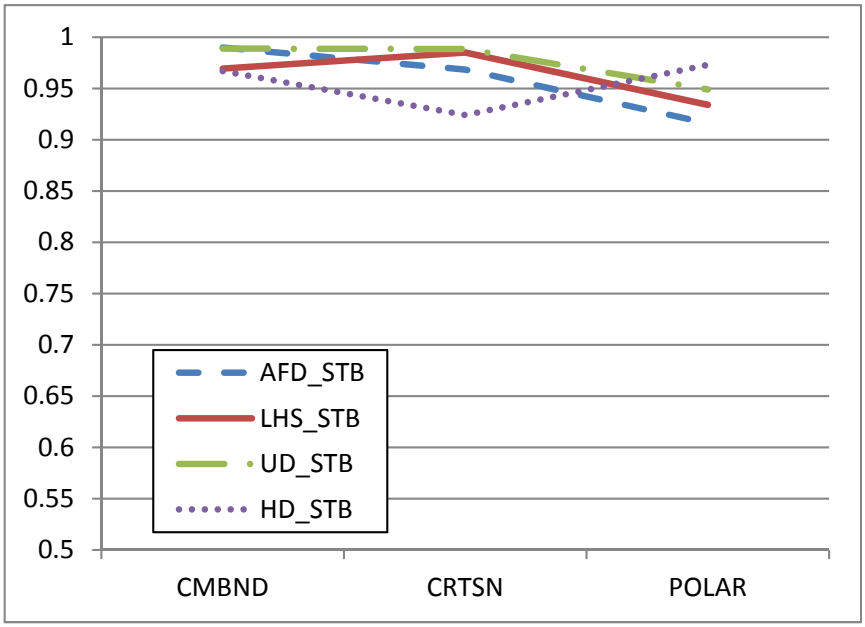


(a) R^2 test interaction

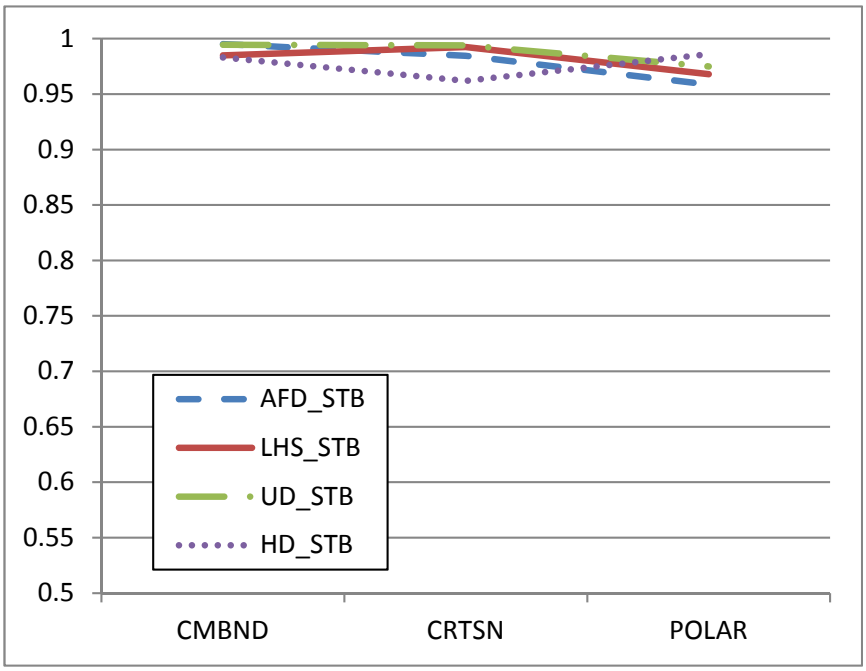


(b) r test interaction

Figure 4.7: Sampling-representation test interaction of R^2 and r measures

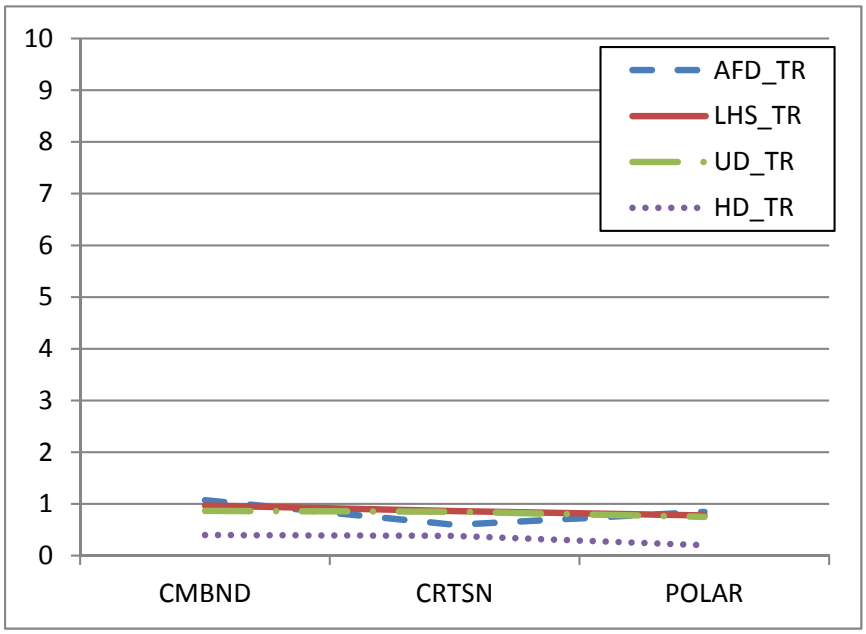


(a) R^2 stability interaction

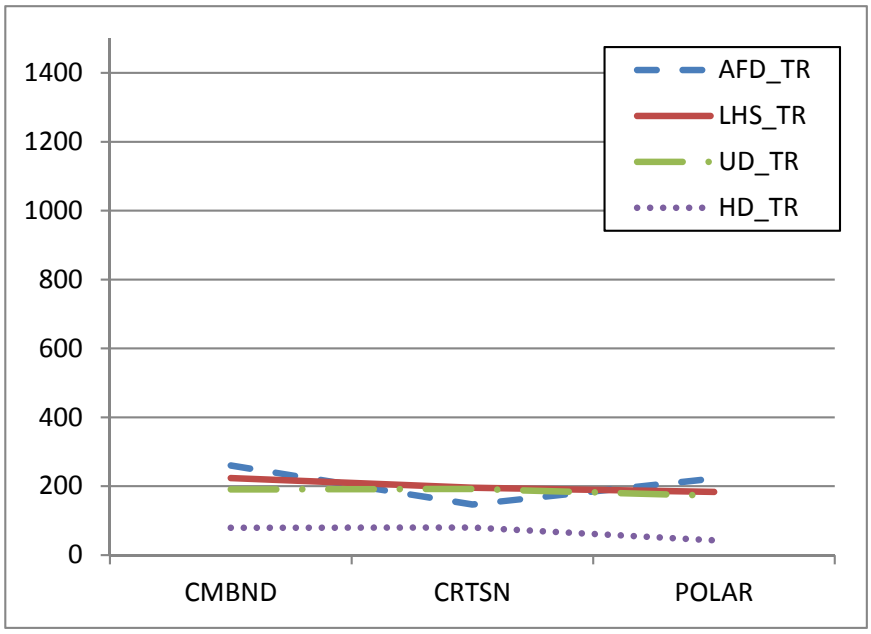


(b) r stability interaction

Figure 4.8: Sampling-representation stability interaction of R^2 and r measures

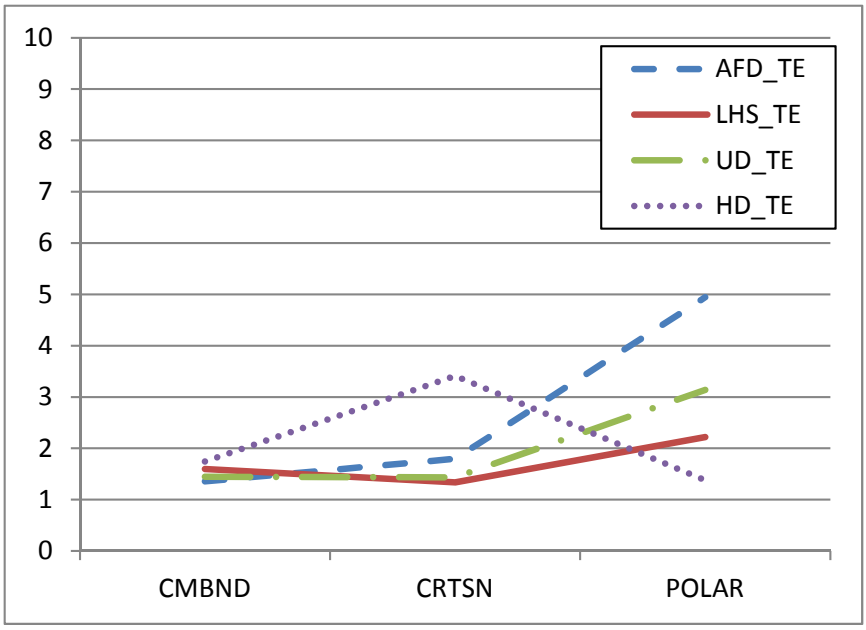


(a) *MAE* training interaction

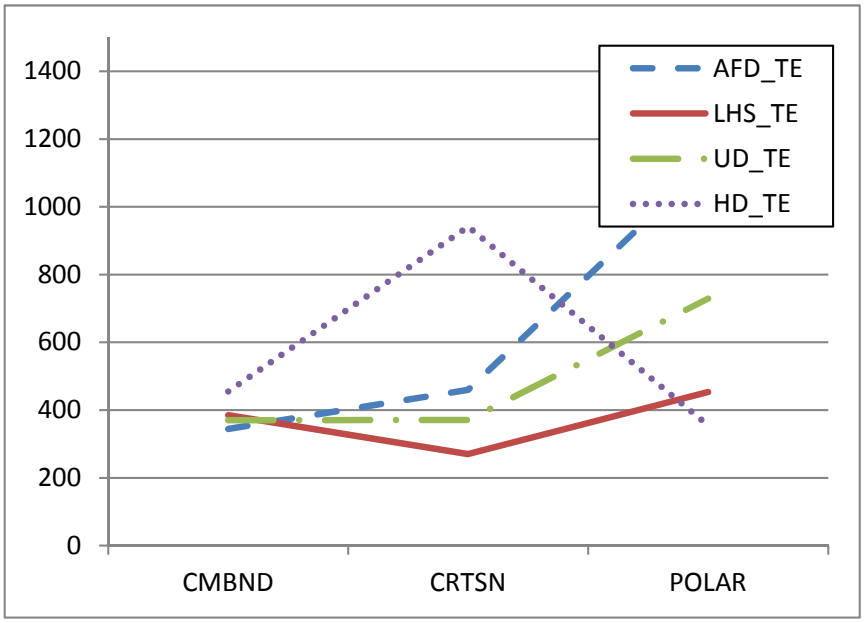


(b) *MSE* training interaction

Figure 4.9: Sampling-representation training interaction of *MAE* and *MSE* measures

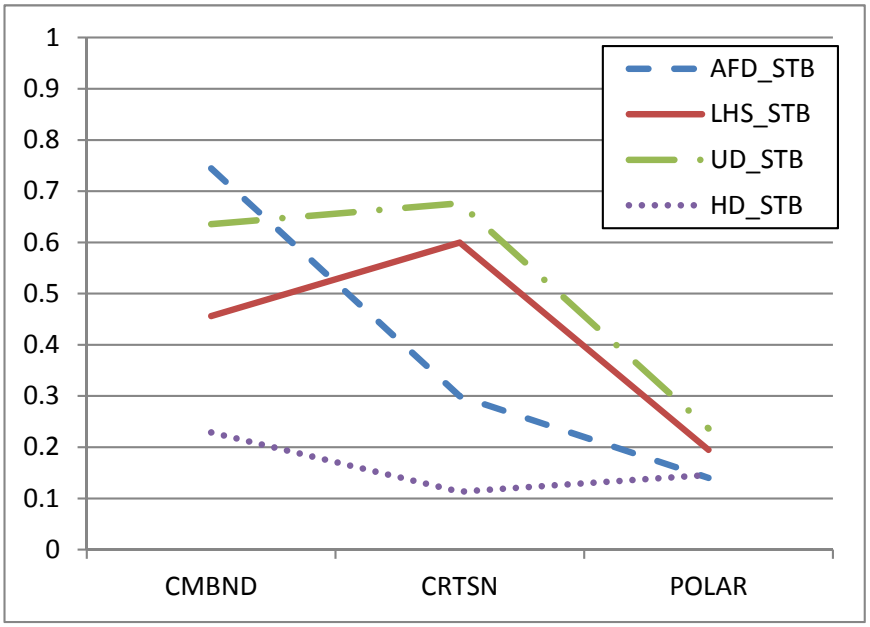


(a) *MAE* test interaction

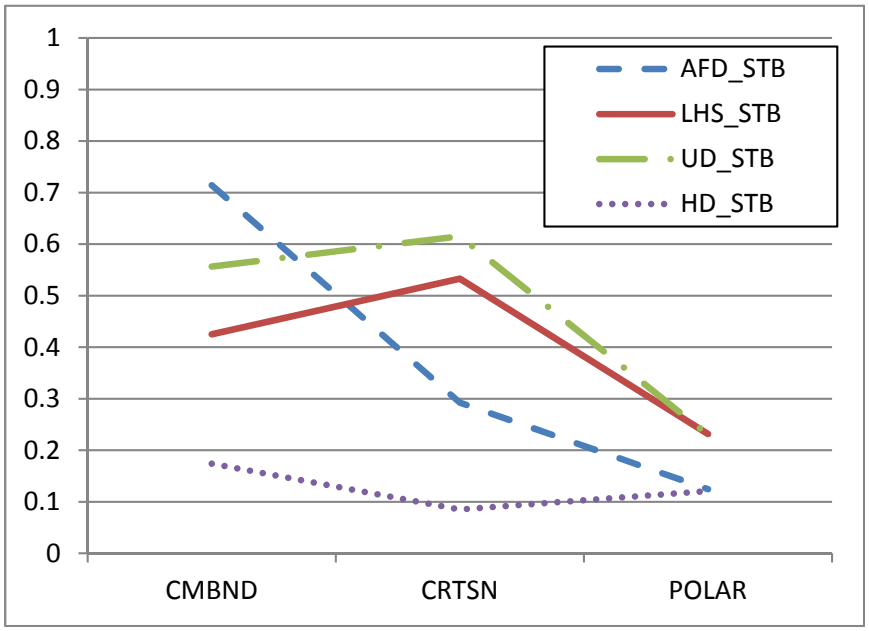


(b) *MSE* test interaction

Figure 4.10: Sampling-representation test interaction of *MAE* and *MSE* measures



(a) *MSE* stability interaction



(b) *MSE* stability interaction

Figure 4.11: Sampling-representation stability interaction of *MAE* and *MSE* measures

space in low sample sizes. AFD performance is affected by the number of levels of each factor. HD on the other hand locates 10 samples at the boundaries and one sample at the center. LHS and UD have n-dimensional uniformity, where n is the number of design points, that locates sample points around the center. We believe that a high number of observations close to the center leads to an increase in the prediction performance of the metamodel.

The CMBND representation generates good results, since the independent variables (predictors) of a position information is reduced before training the metamodel. The POLAR representation produces unsteady results. Notably, it has the best performance with HD. In general, according to the testing data performance, POLAR has worse performance than the other representations for sample sizes less than 24. It also produces the best training and the worst test data performances for the sample of size 15. This might be due to the fact that after the coordinate conversion, small errors that occurred during the prediction of azimuth or elevation angles can cause significant changes in the final detection location. Thus, POLAR generates the worst results with respect to *MSE* measure.

The results show that the metamodeling of a dynamic simulation system using MARS is feasible. The LHS-CRTSN and UD-CMBND combinations with sample sizes of 15 and the HD-POLAR combination with a sample size of 11 can all be used to train a metamodel to replace the dynamic simulation model for radar detection.

CHAPTER 5

THE DIVIDE AND FIT APPROACH

DEVS formalism allows hierarchical specification of simulation models. A DEVS simulation model, also called composite simulation, consists of atomic models connected to form coupled models. In other words, the composite simulation can be decomposed into smaller models connected to each other. By courtesy of DEVS formalism, the connection structure and the types of data exchanged between atomic models are known before a simulation run. The connection structure of atomic models seems suitable for exploitation, but potential recurrent (feedback loop) connections and time-dependent input and output messaging do not leave room for static decomposition techniques like component oriented decomposition [47] or optimal partitioning [2].

We propose building metamodels of computationally expensive atomic models and replacing the original atomic models with their respective metamodels implemented as PMs for faster simulation runs. In many multi-disciplinary simulations we are aware of, only a few simulation models per simulation are predominantly time consuming and worth the metamodeling effort. The designer can specify the list of atomic models for metamodeling and only those atomic models are subjected to the metamodeling process.

5.1 Metamodeling of an Atomic Model

An atomic model is like a black-box function and only the values of design variables in X and the time and values of the input events with external transition function

$\delta_{ext} : s_{t-1} \times v_t \rightarrow s_t$ and output events with output function $\lambda : s_t \rightarrow y_t$ can be observed. The metamodel should predict whether to produce output and the value of output using only these values.

Major assumptions about the atomic model that we are metamodeling are as follows:

i. Atomic model is a dynamic time-invariant system [79], intuitively meaning that model's behavior does not depend explicitly on time.

ii. Atomic model's internal transitions are not observable in accordance with the black-box assumption.

iii. Atomic model generates output in the course of a simulation run, but it may not generate output at some instants of time.

The methodology explained for dynamic simulation metamodeling in Section 4.1 is applied to the atomic model. Metamodel building starts with the selection of design points from the design space using the selected sampling technique. We assume that the range values of the design variables are pre-specified by the decision maker (or problem owner). Then, for each design point, composite simulation is run and the input and output events of the atomic model are recorded at each time step. At the end of the simulation runs, a table of observations is formed called *complete observations table*, where columns specify data from each port and rows specify input-output communication of the atomic model in a time step basis. During this process, the previous time step output, y'_t, is also added to the predictors of the current time step. By this way, the metamodel is provided with information to find the correlations between the previous and the current states of the atomic model.

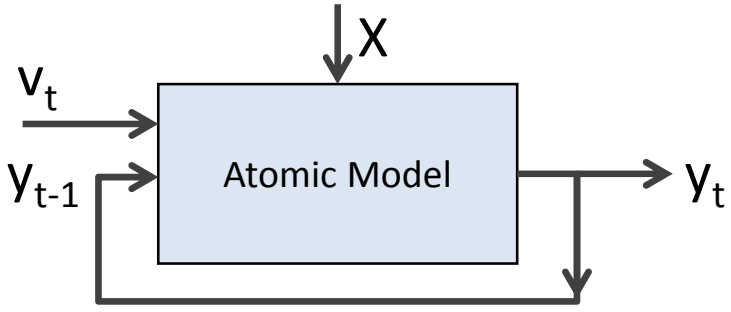


Figure 5.1: Input and outputs of an atomic model

The inputs and outputs of an atomic model is presented by Figure 5.1. At time step t,

the atomic model output may be written in the form $Y_t = f(X, V_t, Y'_t)$, where X is the set of atomic model design variables, $V_t = \{v_{1t}, v_{2t}, .., v_{qt}\}$ is the set of all input events at time step t received from q input ports and $Y'_t = Y_{t-1} = \{y_{1t-1}, y_{2t-1}, .., y_{rt-1}\}$ is the set of all output events produced by r output ports at the previous time step. In other words, the design variables of the atomic model, the input events at a time step and the output events at the previous time step are used to estimate the output events at the current step.

5.1.1 Training of the metamodel

Output metamodels are used for predicting the output events of the atomic model, but an atomic model may not generate output at every time step. We call this as an incomplete output process. For example, a sensor model generates output only when there is a detection or a launcher may only fire a missile when the launch command is received. To handle these cases, as explained in Section 4.1 we have devised a two-level metamodeling approach, where a different metamodel is used for predicting *whether to output or not* in this step, which is called *OutputControl* metamodel.

During the training phase, the observation tables are preprocessed for each metamodel. *Output* metamodels should only be trained using the observation rows (time steps) where atomic model generates output. For this reason, before training, the time step logs without any output are trimmed. On the other hand, the *OutputControl* metamodel requires the boolean response of whether to output in each step. The responses in the observation table are converted to boolean information (e.g. 0 if there is no output and 1 if there is output) before training the metamodel. Hence, the metamodels are trained using the preprocessed observation tables.

As mentioned in Section 4.1, due to limitations of the available MARS tools, input or output events with a complex structure is split into separate pieces and each piece is treated as a separate predictor (e.g., the position vector $\langle sposx, sposy, sposz \rangle$ is separated into three distinct variables).

5.1.2 Proxy Model (PM) Integration

In this section the integration procedure of PM into the PMIS is described. In order to replace an atomic model with its PM in the context of the whole simulation, one must implement mechanisms for

i. Logging the input events and output events of the target atomic model during the training runs.

ii. Emulating the input event handling and output event generation of the target atomic model in the PM.

iii. Using the trained metamodel during the output event generation.

Logging events in a DEVS simulation requires attaching hooks to the events of the DEVS engine for a specific atomic model. The events should be parsed using the types declared in the external transition function (δ_{ext}) of DEVS formal definition. (Standard DEVS event messages do not carry type information, but they are structured according to the types specified). Then, the events should be properly logged in a time step basis. For this purpose we implement the Logger Model (LM), which attaches itself to an atomic model and records inputs and outputs of it to generate the observation table. Later, the observation table is used for training a metamodel by the *Earth* package of the R environment [109].

Replacing the target atomic model requires altering the connection structure of DEVS simulation and attaching the input ports and output ports of the target atomic model to the new model. The new model must correctly parse the input event, use the event to predict the next output and form output event from the predictions of the metamodel. Moreover, the input event handling, prediction and output event generation are performed repeatedly at each time step of the simulation run; so these structures should work efficiently.

We implement PM for this purpose. PM uses the trained metamodel and replaces an atomic model in the DEVS simulation. PM uses the input events from the input port and design values to generate the output events from output ports. Figure 5.2 shows the usage of the LM and PM in our engagement simulation. The whole simulation, with LM and PM included, has been implemented by using the DEVS-based

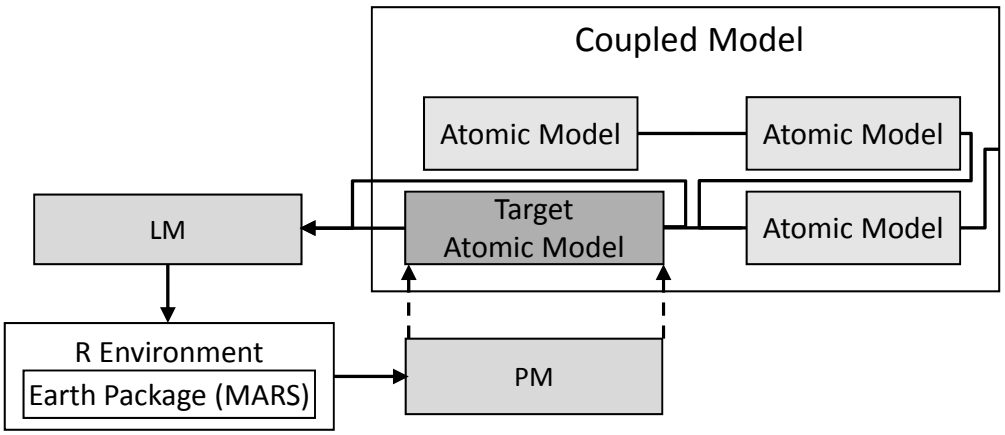


Figure 5.2: PM in DEVS simulation

simulation framework SIMA [57].

The process to replace an atomic model with its metamodel consists of the following steps:

i. PM gets the input events and forms a single row of observation table using previous time step output event and the current time step input events.

ii. PM uses the observation row and the trained *OutputControl* metamodel to decide whether to generate outputs at this time step or not.

iii. PM uses the observation row and trained *Output* metamodels to predict the output values, in case an output should be generated.

iv. PM generates an output event of the required type from the output values and sends the event from the output port.

The architecture of PM is depicted in Figure 5.3. PM binds the event parser to the input port. Event parser extracts the required data from input events since an event can contain other information such as sender id, receiver id and simulation framework dependent data. Input event data and previous time step's prediction are used by the observation collator to form the observation row. The observation row is passed to the output predictor to be used by the MARS metamodels. The output predictor uses the *OutputControl* metamodel and the observation row to decide whether atomic model should generate an output event. If it should, then the predictor uses *Output* metamodels and the observation row to predict the output. Event generator builds the

corresponding output event from these components and propagates the output event to the output port.

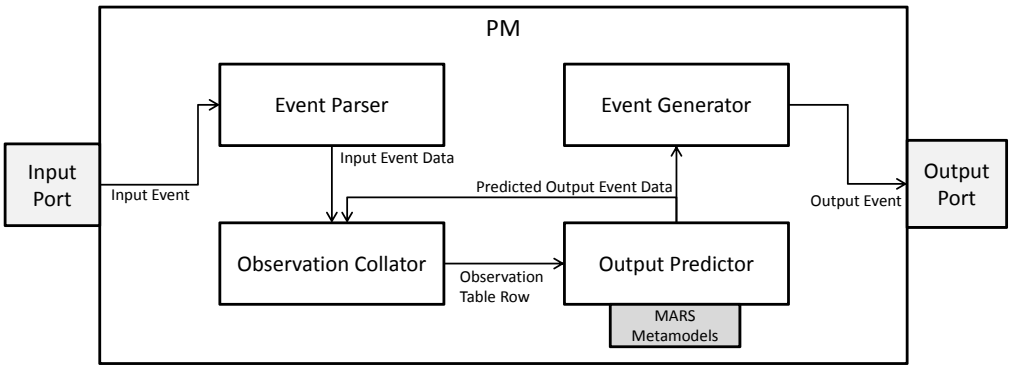


Figure 5.3: PM architecture

5.2 The Case: Radar Simulation in a Military Engagement Simulation

In this study, the sample composite simulation is an engagement simulation between an attacking aircraft and a defending ship. The aircraft engages the ship with a missile from a properly close distance while the ship responds by firing its AA gun. The ship has a radar system for detecting the incoming aircraft. Command and control system (C2) of the ship uses the track information from the radar and issues fire commands to its AA gun with the target location. There is an environment model in the simulation that checks for collisions and radar signal transmission by using platform information (e.g., location) and radar information (e.g., range).

The radar system in the simulation uses the current location and signal information for performing target detection analysis. If the radar detects a platform two times in three consecutive time steps, it issues a *detection* event. Evidently, the ship's defense system relies on the detection performance of the radar. In the simulation, the purpose of the user is to build a defensive strategy for the ship to survive the engagement. Figure 5.4 shows an overall view of the engagement.

The simulation accepts two sets of input parameters. User can define the radar design variables, which are the characteristic parameters of a radar system (i.e. the power, frequency and bandwidth of the radar). The user can also define the scenario of the simulation by specifying the initial height (h_p) and speed (s_p) of the aircraft. The

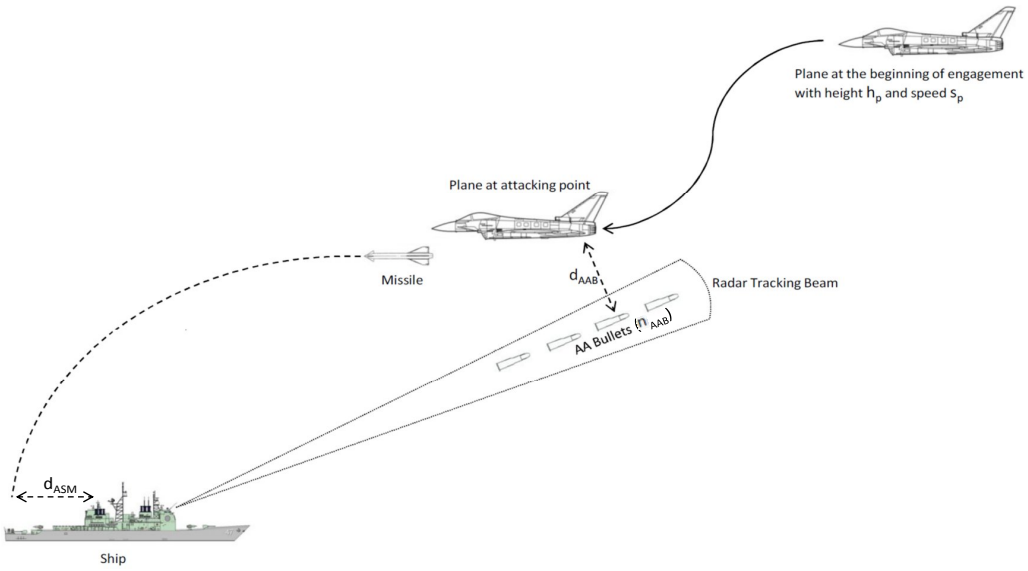


Figure 5.4: An overall view of the engagement simulation

aircraft model calculates the flight way points to successfully attack the ship with a missile. In our model, the flight way points are calculated deterministically. In other words, any given initial position and initial speed always generate the same flight way points. Note that the radar is assumed to be an omni-directional radar, meaning that the approach angle of the aircraft does not affect the radar's detection performance. The AA gun and missile parameters are assumed to be fixed in this simulation. Table 5.1 shows the input and output variables of the simulation. Here, the design variables of the simulation are the design variables of the radar atomic model.

Table 5.1: Inputs and outputs of the simulation

Category	Name
Input:Design Variables	ShipRadarFrequency (GHz) ShipRadarPower (dBW) ShipRadarBandwidth (MHz)
Input:Aircraft Parameters	AircraftInitialDistance (m) AircraftInitialAltitude (m) AircraftInitialSpeed (m/s)
Output:Simulation Results	MinAABulletDistanceToAircraft d_{AAB} (m) NumAABulletsFired n_{AAB} MinMissileDistanceToShip (m) d_{ASM}

We analyze the simulation results with respect to three outputs (see Figure 5.4): the

minimum AA bullet distance (d_{AAB}) to the aircraft, total number of AA bullets (n_{AAB}) fired by the ship and the minimum missile distance (d_{ASM}) to the ship. High rate of detection by the radar system increase the AA bullet accuracy, thus making the d_{AAB} smaller and d_{ASM} larger. The simulation detects bullet-aircraft collisions by using the aircraft body dimensions but d_{AAB} results are calculated according to the geometric center of the aircraft. High number of n_{AAB} indicates inaccuracy of the radar tracking. The radar system can detect the aircraft but without accurate location information the bullets cannot hit the aircraft. The radius of the aircraft body is specified as $3.75m$, which specifies the minimum distance to the center required to hit the aircraft. Any bullet closer than this distance is assumed to destroy the aircraft (there is no partial damage) and the engagement (hence simulation) will be finished.

The overall system is a time-driven simulation implemented as a discrete-event simulation with fixed time increments. As shown in Figure 5.5, there are three main models in the simulation: environment, ship and aircraft. The ship model is further decomposed into ship platform, ship C2, radar and AA gun atomic models while the aircraft model is further decomposed into aircraft platform, aircraft C2, radar and missile models.

Ship's radar atomic model is the most computationally intensive part in the whole simulation making it a worthwhile candidate for our metamodeling technique. Thus, we build the metamodel of the radar atomic model. Radar atomic model has three design variables, two input ports and one output port as listed in Chapter 4.2 Table 4.2. Input events include the host platform information (ShipPlatformInfo) and signal information from target (TargetSignal), while output event is the detection information (TargetDetection).

5.3 Experimental Study on PM-Integrated Simulation

The three sampling technique-coordinate system representation-sampling size combinations specified in Section 4.3.5 are used during the experimental studies: LHS-CRTSN and UD-CMBND with the sample size of 15 and HD-POLAR with the sample size of 11. From this point we will call them as HD, LHS and UD for brevity.

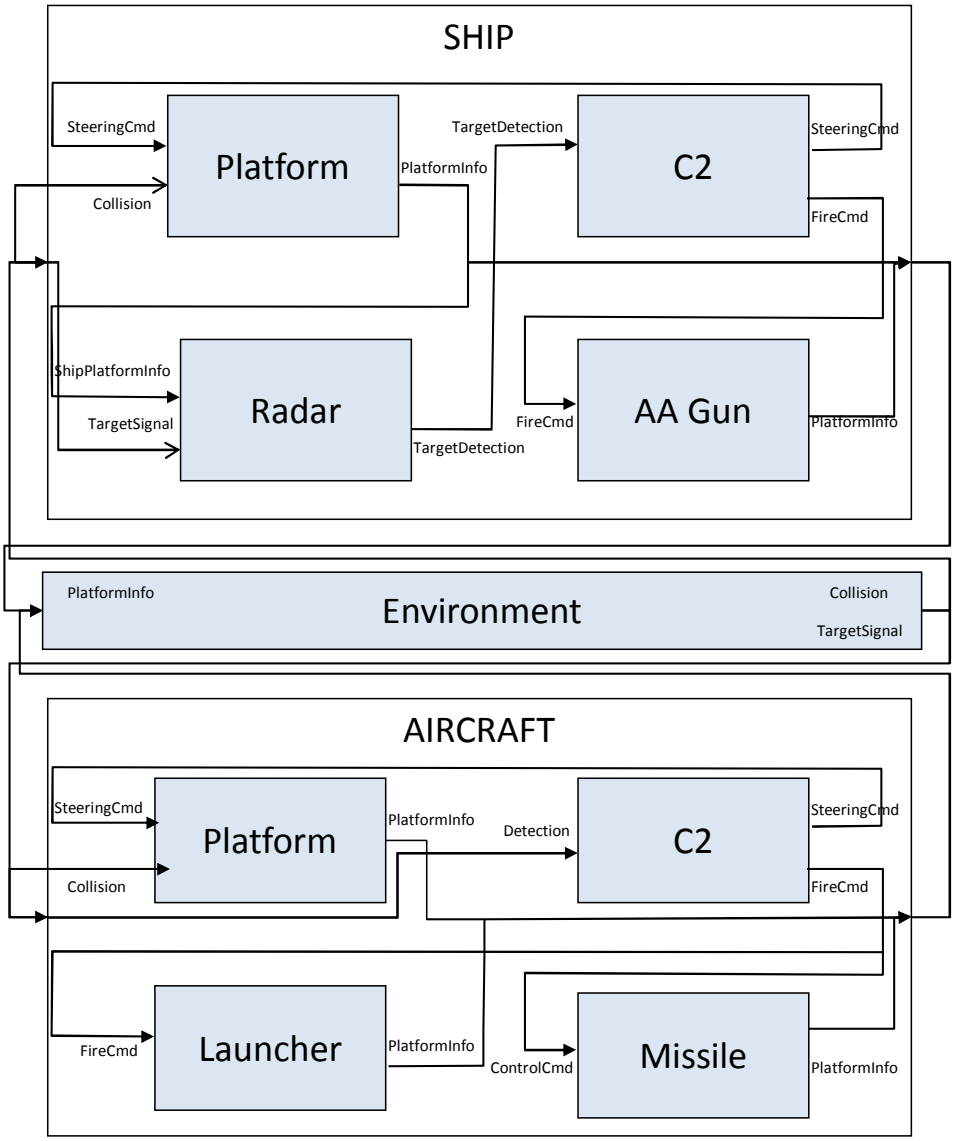


Figure 5.5: The engagement simulation model structure

In order to test the metamodel performance in the context of the composite simulation, we replace the radar model with the PM as described in Section 5.1.2. The prediction performance of the PM is tested using the simulation results of the PMIS with selected combinations and comparing them against the original simulation (OS) results.

5.3.1 Training and Test Data

The scenario parameters, specifically, aircraft parameters, are chosen such that the aircraft starts the simulation at 3000 m distance to the ship (which is just outside the range of the radar system) with 50 m altitude and 200 m/s speed.

During the training phase, MARS models are trained using the simulated data obtained by the specified sampling technique and sample size. The experiments are executed using a test data set which is obtained by random selection from the experimental region with a sample of size 50. The same test data set is used for both PMIS and OS.

5.3.2 Applications and Findings

For each combination, the original simulation is run with the LM model, which builds the observation table. After that, *OutputControl* metamodel and *Output* metamodels are built using the stand-alone *Earth* package of the R environment [109]. The radar atomic model is replaced by the PM containing the MARS models build. PMIS is run for the test data set and the results are collected. This process is repeated for all combinations studied.

The OS is also run using the test data set. OS and PMIS results are evaluated with respect to some critical performance measures. The measures used, their meanings, explanations and formulas are presented in Table 5.2.

The simulation results are analyzed in terms of the three simulation outputs: d_{AAB}, n_{AAB} and d_{ASM}. In order to compare the execution durations of OS and PMIS, simulation is run using the worst case scenario, which guarantees that the ship radar cannot detect the aircraft till the end of the simulaton and the aircraft survives. Using this scenario, OS and PMIS are run for 50 times and the original radar model and PM processing times are measured. The overall simulation execution durations of OS and PMIS are also measured together with the training simulation execution durations.

Table 5.2: Performance measures used for simulation result analysis

Measure	**Explanation / Interpretation**	**Formula**
R^2: Adjusted Coefficient of Determination	Percentage of variation in response explained by the model / the higher the better	$R^2 = 1 - (1 - \frac{\sum_{k=1}^{n}(y_k - \widehat{y}_k)^2}{\sum_{k=1}^{n}(y_k - \overline{y})^2})(\frac{n-1}{n-P-1})$
r: Correlation coefficient	Linear relation between observed and predicted response / the higher the better	$r(y, \widehat{y}) = \frac{\sum_{k=1}^{n}(y_k - \overline{y})(\widehat{y}_k - \overline{\widehat{y}})}{(n-1)s_{\widehat{y}}s_y}$ $s_{\widehat{y}}$ and s_y are the sample standard deviations of $\widehat{y}$ and y
MAE: Mean Absolute Error	Average magnitude of errors / the lower the better	$MAE = \frac{1}{n}\sum_{k=1}^{n}\lvert y_k - \widehat{y}_k \rvert$
MSE: Mean Squared Error	Average of the squared errors by regarding the number of terms in the model / the lower the better	$MSE = \frac{1}{n-P-1}\sum_{k=1}^{n}(y_k - \widehat{y}_k)^2$

Notes: n is the is the number of samples in the sampling technique, P is the total number of predictor variables in the metamodel, y_k is the actual output from OS and $\widehat{y}_k$ is the predicted output from PMIS

5.3.3 Results

The results for the sampling techniques with respect to all three simulation outputs are presented in Table 5.3.

Table 5.3: Performance of sampling techniques with respect to simulation outputs

		HD	LHS	UD
d_{AAB}	R^2	0.955	0.810	0.966*
	r	0.988*	0.928	0.987
	MAE	0.264	0.418	0.215*
	MSE	0.406	1.552	0.318*
n_{AAB}	R^2	0.664	0.878*	0.873
	r	0.947	0.954*	0.951
	MAE	2.220	1.240*	1.840
	MSE	19.891	6.217*	7.609
d_{ASM}	R^2	0.927	0.943*	0.895
	r	0.993*	0.987	0.958
	MAE	1.905*	3.810	7.238
	MSE	5.526*	20.868	67.907

* indicates the best values with respect to the measure

According to the results,

i. Regarding the d_{AAB}, UD has the best performance in terms of R^2, MAE and MSE measures, while LHS has the worst performance. HD has a similar performance with UD but can outperform it only with respect to the r measure.

ii. Regarding the n_{AAB}, LHS yields the best performance results in terms of all measures, while HD yields the worst performance.

iii. Regarding the d_{ASM}, HD gives the best performance results in terms of r, MAE and MSE measures. LHS is the second best according to these measures, while exceeding the performance of HD with respect to the R^2 measure. UD does not exhibit a good performance with respect to MAE and MSE measures.

iv. The selected sampling-representation-size combinations have varying performances in terms of the simulation outputs. The radar model's primary output is the location of detection. At the beginning of the simulation, the radar model cannot detect the aircraft which has just entered into the effective detection

range. After that the range of the first detection point depends on the radar performance for the original radar model. Upon detection, the AA gun fires bullets as long as there is a detection. PM should be accurate enough to predict the detection at a similar range so that the n_{AAB} is close to that of the OS.

v. The radar propagation model takes the atmospheric ducting and multi-path propagation effects into account, which cause the radar model to drop the tracking on the aircraft from time to time as shown in Figure 5.6. The AA gun cannot fire bullets until there is a re-detection. Similarity of the results between the OS and PMIS also depends on the closeness of the detection start and end points of the original radar model and those of the PM model.

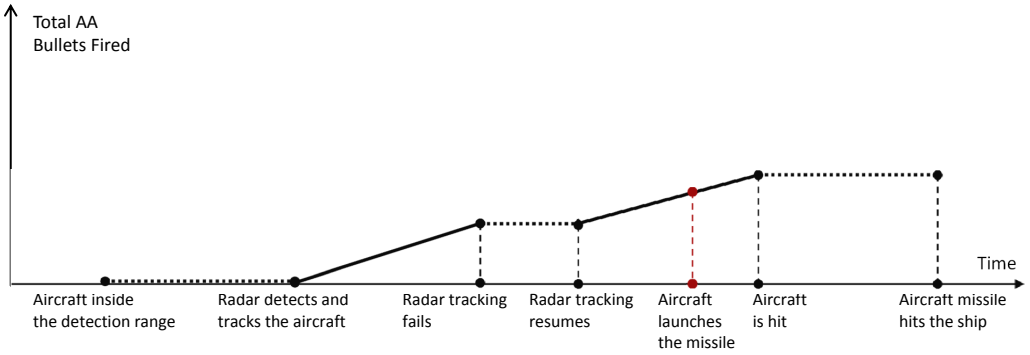


Figure 5.6: Example n_{AAB} with respect to aircraft distance to ship

vi. If the detection point prediction error is so small that it does not affect the occurrence of a aircraft hit/miss event (i.e. both results are larger than the radius of the bounding sphere of the aircraft), then it has no effect on the d_{ASM}. Also there are simulation results where the d_{AAB} is close to zero, but the aircraft is hit after the missile launch and the ship is destroyed.

vii. HD has the best results for predicting the d_{ASM}, LHS has the best results for predicting the n_{AAB}, and UD has the best prediction performance for the d_{AAB}.

Execution time performance results are shown in Table 5.4. According to the results,

i. PM works over 1000 times faster than the original radar atomic model.

ii. PMIS works more than 300 times faster than OS.

iii. As expected, the training costs of LHS and UD are greater than that of HD due to the training sample sizes.

Table 5.4: Execution time comparison of OS and PMIS

	OS	PMIS-LHS	PMIS-HD	PMIS-UD
Training Time	N/A	15×24914.11	11×24795.48	15×24952.43
Radar Model Time	24723.17	20.97	21.64	20.61
Total Simulation Time	24968.48	73.91	74.16	73.35

Note: Units are in milliseconds.

5.4 Discussion

In this section, some of the results that might need closer attention to evaluate the proposed methodology are discussed further in the light of the above findings. The radar atomic model is the most time consuming atomic model in the simulation. By replacing it with a PM, the composite simulation has gained a speed-up of more than 300. According to the execution time performance results, the PMIS is extremely efficient. The processing requirements of the target atomic model are important in this respect. Only the most time consuming atomic models should be replaced by PM. Unlike static metamodel of an overall simulation, metamodel of an atomic model will run through a series of inputs and produce a series of outputs (the time steps of the simulation). Hence, the efficiency of the metamodel is much more important.

The selected sampling-representation combinations have dissimilar performance results for different simulation outputs. The difference in performance may be attributed to distinct allocation schemes of design points in the experimental regions for each sampling technique. To illustrate, HD, LHS and UD provide the best performance for predicting the missile distance to ship, the total AA bullets and the AA bullet miss distance to the aircraft, respectively.

What is best for the accuracy of the metamodel may not be the best for the accuracy of the overall simulation. The PM runs for all the time steps and with inputs other than the design variables (e.g. input events such as aircraft location at each time step). Hence, in order to successfully train the metamodel, a simulation run should include time steps where as many input events as possible has occurred. In our example, consider the case where radar detects the aircraft at the very first time step. Then simulation run will only include one input event and PM will not be trained with

enough observation data. Hence, design points generated by the sampling technique should be distributed enough for the simulation model in the sense that a variety of dynamic inputs can be observed.

Magnitude of errors in the overall simulation results is larger than that in the atomic model outputs. This is due to the accumulation of errors over all the time steps during a simulation run. An early prediction error may cause an unreasonable result such as untimely destruction of the aircraft. On the other hand, if the aircraft is hit on the first time step of the original simulation, and if PM fails to generate the hit event on that first time step, then simulation result will be unreliable.

Metamodeling a dynamic simulation includes some particular properties, too. Since the PM here is a dynamic simulation model, any error in the prediction at some time step may affect subsequent time steps. The observation table includes output data from the previous time step. By this way the metamodel can keep track of the state of the target model. However, if an output has an error, then it may affect the next step. Consider radar's first detection of the aircraft. In the previous time step the radar fails to detect giving no output, and in the next time step the radar outputs the detected location. Let us call this sequence *F-D*, where *F* and *D* represent 'fail-to-detect' and 'detect' respectively. The radar model will not drop the tracking of the aircraft until consecutive no-detection steps occur. Hence, there will only be a few time steps with this information (*F-D* or *D-F*), whereas a lot of time steps will consist of ongoing track (*D-D*) or ongoing fail-to-detect (*F-F*) information. The metamodel will be more successful at predicting the *D-D* or *F-F* cases and may miss the state changes of the output.

The observation table has $n \times \omega$ rows instead of n rows, where n is the cardinality of the set of points selected by the sampling technique and ω is the number of steps in one simulation run. This puts a burden on the observation table collection, observation table preprocessing and metamodeling. Our two-level metamodeling approach (*OutputControl* metamodel and *Output* metamodels) aims to increase the accuracy of the metamodeling with the large observation table, since *Output* metamodels are trained using only the relevant rows.

CHAPTER 6

AN APPLICATION OF THE PROPOSED APPROACH FOR OPTIMIZATION

In this chapter, we use the PMIS in optimization to demonstrate a usage scenario of the proposed approach. In real world problems, computer simulation is frequently used for optimizing the design variables of a system. If the computer simulation is computationally expensive, then the optimization procedure will take too much time or require a lot of computational resources. In Section 2.3, we present a number of methods for handling computationally expensive simulations. The developed method is another way of dealing with the computational intensity. In this chapter, we present the response surfaces of the OS and PMIS, run an optimization procedure on both simulations and compare the results. The simulation has three different outputs but according to the results we found out that only the simulation result of d_{AAB} can be directly optimized. Hitting the aircraft as close to the center as possible has a higher survival value than using minimum n_{AAB}. And maximizing d_{ASM} is not useful since the results are same for all cases where the aircraft is hit before launching the missile. For this reason, we will apply single objective optimization to the radar design variables with no constraints considering the d_{AAB} as response of the simulation.

6.1 Optimization Techniques

A variety of techniques have been proposed for solving the general optimization problem. Comprehensive surveys about these techniques can be found in [46, 108, 110]. The techniques can be divided into two major parts: global optimization techniques

and local optimization techniques [110]. Local optimization techniques require unimodal response surfaces and they fail when the surface model is discontinuous or non-differentiable [101]. On the other hand, global optimization techniques are designed for multi-modal functions; and heuristic search methods like evolutionary computation techniques can work without any information about the response surface and they do not require the closed form definition of the problem.

6.1.1 Evolutionary Computation

Evolutionary Computation techniques are stochastic heuristic search methods that attempt to mimic nature [56]. Genetic algorithms (GA) [49] are the first evolutionary algorithm introduced, where input configurations (solutions) are modeled as genes of chromosomes and candidates of optimal solution are compared, selected and recombined according to their performance. Just like evolution and survival of the fittest in nature, the population of configurations breed for reaching the optimum. The algorithm also maintains mutation concept for increasing exploratory behavior and avoiding local optimums. Although modified versions of GA has been used for continuous problems [20, 117], original algorithms work with discrete decision spaces and are widely used for combinatorial optimization. Furthermore gene coding schemes determine the success of the GA [8], therefore problem specific knowledge is needed for effective usage of GA.

Memetic algorithms are inspired by Darwin's meme notion, and they are hybrid algorithms combining GA with local search techniques [43]. As an analogy, offsprings after the recombination of GA are allowed to gain local experience and selection for next generation is applied afterwards. Just like GA, it is best applied to combinatorial optimization problems (e.g., traveling salesman problem, vehicle routing problem, knapsack problem, minimum spanning tree problem) [70].

6.1.2 Particle Swarm Optimization

Particle Swarm Optimization (PSO) [63] is another heuristic inspired from the nature and mimics the social behavior of migrating birds trying to reach a destination. Each

design configuration (named *particle* in the algorithm) is viewed as a bird in the flock and just like birds determine their speed and direction to avoid collision in flock, to maintain flock velocity and to stay close to center of the flock; particles determine their velocity and update their location (i.e, update value of each design variable) in p-dimensional design space [64]. Unlike GA and MA, PSO particles can readily use real values. The ith particle is represented as $\mathbf{x_i} = \{x_{i_1}, x_{i_2} \ldots, x_{i_p}\}$ and velocity of the particle is represented as $\mathbf{vel_i} = \{vel_{i_1}, vel_{i_2} \ldots, vel_{i_p}\}$. Each particle's local best position and flock's global best position are also maintained and they are represented as $\mathbf{bl_i} = \{bl_{i_1}, bl_{i_2} \ldots, bl_{i_p}\}$ and $\mathbf{bg} = \{bg_1, bg_2 \ldots, bg_p\}$ respectively. The algorithm runs for a number of iterations and at each iteration, the particle velocity and position is updated according to the following formula:

$$vel_{i_d}^{t+1} = w * vel_{i_d}^{t} + c_1 * r_1 * (bl_{i_d}^{t} - x_{i_d}^{t}) + c_2 * r_2 * (bg_d^{t} - x_{i_d}^{t}) \tag{6.1}$$

$$\text{subject to} \quad vel_{max}^{t+1} \geq vel_{i_d}^{t+1} \geq -vel_{max}^{t+1}$$

$$x_{i_d}^{t+1} = x_{i-1_d}^{t} + vel_{i_d}^{t} \quad \text{where} \quad d = 1, \ldots, p. \tag{6.2}$$

Here t is the iteration counter; c_1 and c_2 are positive numbers named *learning constants* c_1 for the affect of the local best position and c_2 for the affect of the global best position; r_1 and r_2 are two random numbers in the range $[0, 1]$ and w is called *inertia weight*, which defines how previous velocity contributes to current velocity. W defines the tradeoff between global and local exploration abilities of the particles. A larger inertia weight relaxes the particle movement and facilitates global exploration while a smaller inertia weight facilitates local exploration for fine tuning optimization. Eberhart and Shi [39] propose decreasing w linearly with time. vel_{max} limits maximum velocity change of a particle. Note that second term in equation 6.1 defines 'private thinking' of a particle, while third term in equation 6.1 defines 'social collaboration' among particles.

There are two versions of PSO according to the communication between particles: local PSO and global PSO. In global PSO, all particles communicate, therefore global best position is the swarm best position. In local PSO, particles can only communicate with neighborhood particles. Kennedy [62] suggests that global PSO converges fast but has large chance of being caught in local optimum while local PSO converges slowly while better searching the solution space. There are a number of proposed local PSO neighborhood structures in literature including pyramid shaped, star shaped,

von Neumann, ring shaped and random shaped [105].

The global optimization algorithm is selected as PSO, due to the following reasons:

i. PSO is a successful continuous variable global search algorithm, which has been applied to large-scale problems in several engineering disciplines [103],

ii. The main algorithm of PSO is relatively simple and there are only two parameters (coefficients) of the algorithm to be specified,

iii. There are several parallel and distributed applications of PSO algorithm which leaves room for scalable optimization runs in distributed environments such as [76, 87, 111, 119].

6.2 Experimental Study on Optimization

The single objective simulation optimization of the military engagement simulation presented in Section 5.2 can be defined as follows:

$$\min_{\mathbf{x} \in X} \quad d_{AAB} = F(\mathbf{x}) \tag{6.3}$$

where $\mathbf{x} = \{x_1, x_2, \ldots, x_p\}$ is a design configuration of the system being simulated, F is the simulation (OS or PMIS) and d_{AAB} is the *AA bullet miss distance to the aircraft* during the simulation. The simulation runs in a deterministic way, so there is no variation in the results for repeated runs. We want to find the design configuration for the radar system which increases the detection performance of the radar and in turn allows the AA gun to hit the aircraft as close to the center of the aircraft (find minimum d_{AAB}) as possible. The simulation detects bullet-aircraft collisions by using the aircraft body dimensions. The d_{AAB} results are calculated according to the geometric center of the aircraft.

PMIS trained with UD design is found to have the best prediction performance for simulation result d_{AAB} in Section 5.3.3. In order to test the performance of the PM integrated simulation in an optimization procedure, we select PSO as the optimization technique and use UD design with the sample size of 15 to train the PM.

The design variables of the radar and their feasible values are as follows:

i. Radar Frequency(GHz) : [1.0, 8.0]

ii. Radar Transmit Power(dbW) : [50.0, 80.0]

iii. Radar Bandwidth(MHz): [0.4, 1.0]

Before running the optimization algorithm, we plot the response surface of the OS and PMIS to compare their optimization performances. The plots represent the d_{AAB} results of the simulation with respect to frequency and power design variables for specified bandwidth values. Figures 6.1-6.3 compare the response surfaces for the bandwidth values of 0.4, 0.7 and 1.0 respectively.

Looking at these figures, one may observe the following:

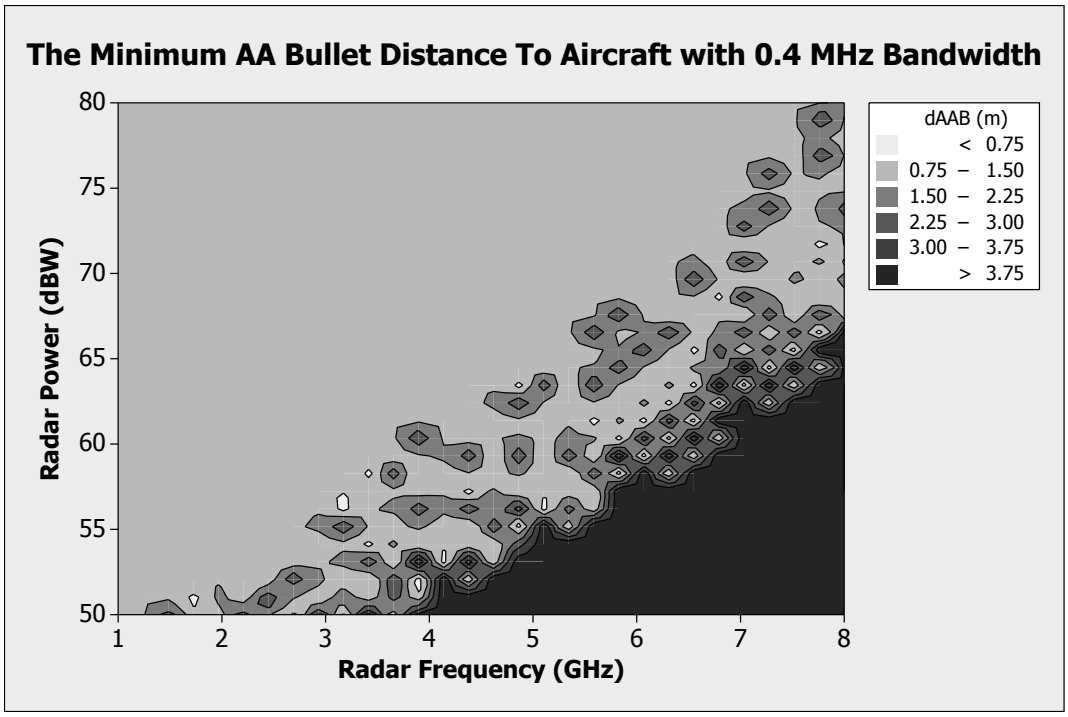
i. Values higher than $3.75m$ are critical in the sense that they represent the failure of the aircraft termination.

ii. The radar system needs higher power for better detection as the frequency increases. This characteristic does not change for different bandwidth values of the radar.

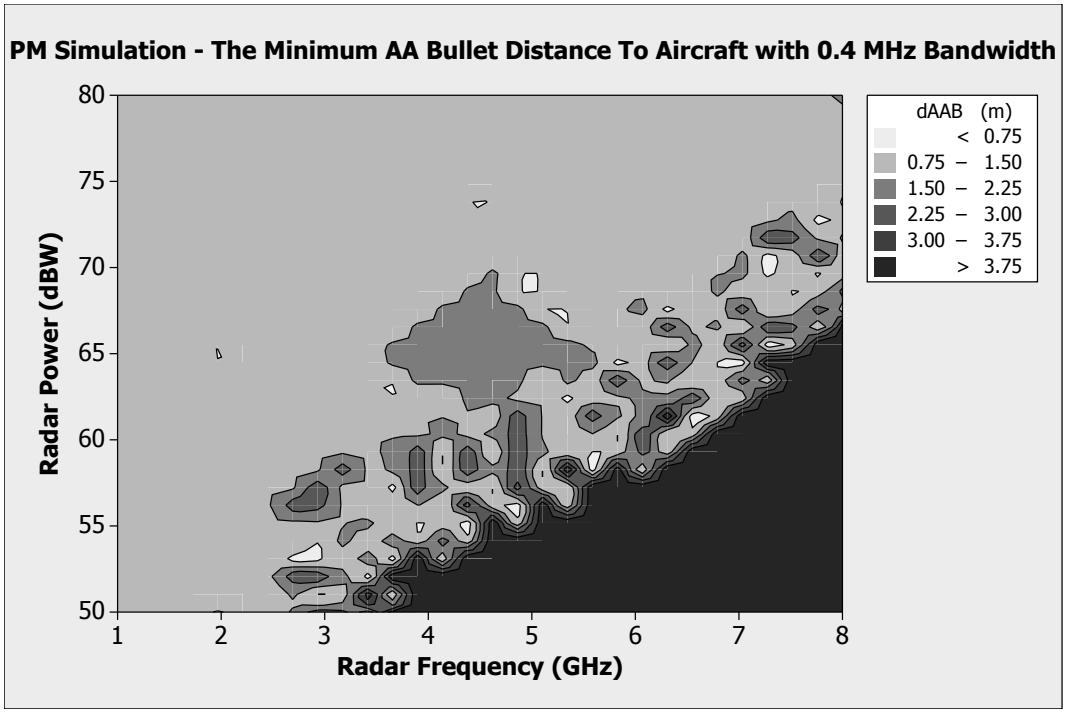
iii. The response surface consists of three zones which can be characterized as *good*, *bad* and *risky*. If we draw a diagonal line from the lower left corner to the upper right corner, the region above that line always has good results, but the values are not best. There is a zone in the lower right corner where the results are always bad (failure). And in between these zones, there is a risky zone which includes a mix of best values and bad values.

iv. There are multiple regions of best values scattered in the risky zone.

There are a number of best values for the minimal bullet distance result for different frequency-power pairs and we want find one such pair using PMIS. According to the results, the optimization results are quite similar to the optimization results of OS. However, there are some points where PMIS gives smaller results than OS. In order to analyze this behaviour, we plot the difference of the response values as contour plots. Figure 6.4 represents the regions where PMIS results are larger (worse) than OS results (false negative values). Figure 6.5 on the other hand, shows the regions where PMIS results are smaller (better) than OS results (false positive values).

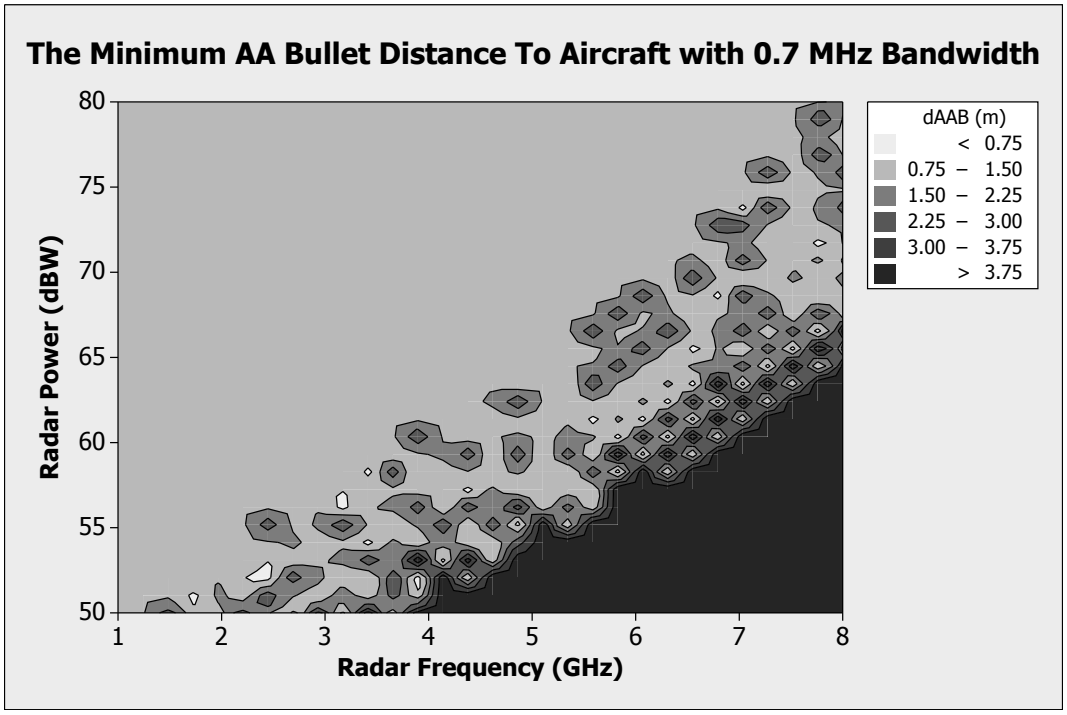


(a) Response surface of OS

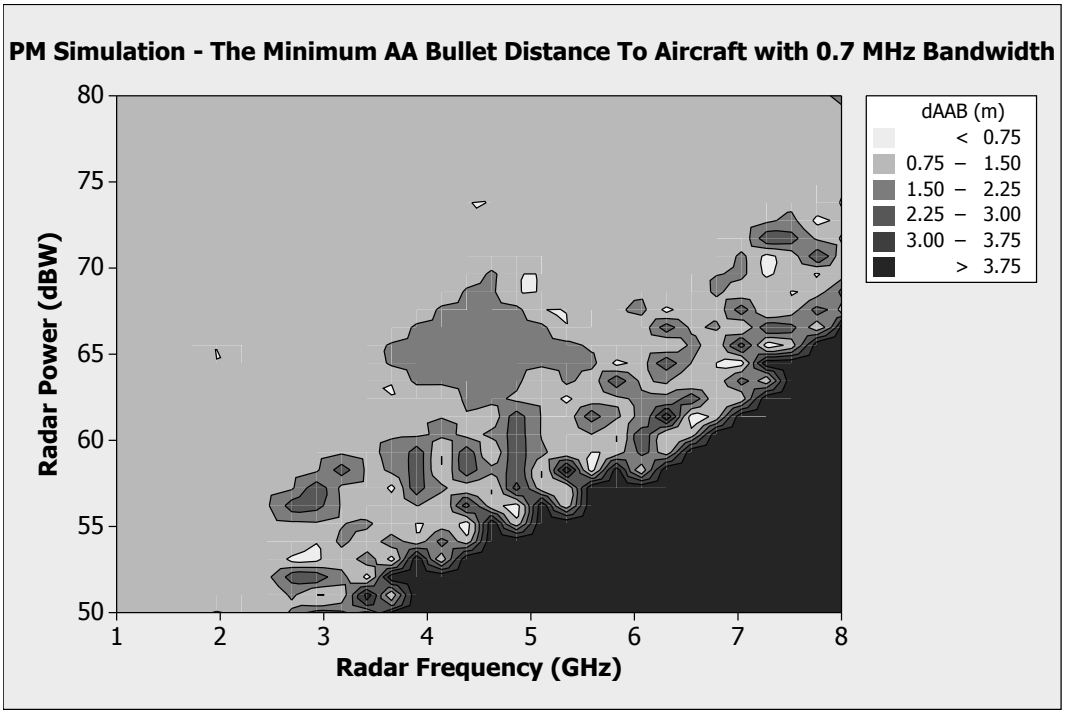


(b) Response surface of PMIS

Figure 6.1: Response surfaces of the simulations for the minimum AA bullet distance to the aircraft with 0.4 MHz Bandwidth

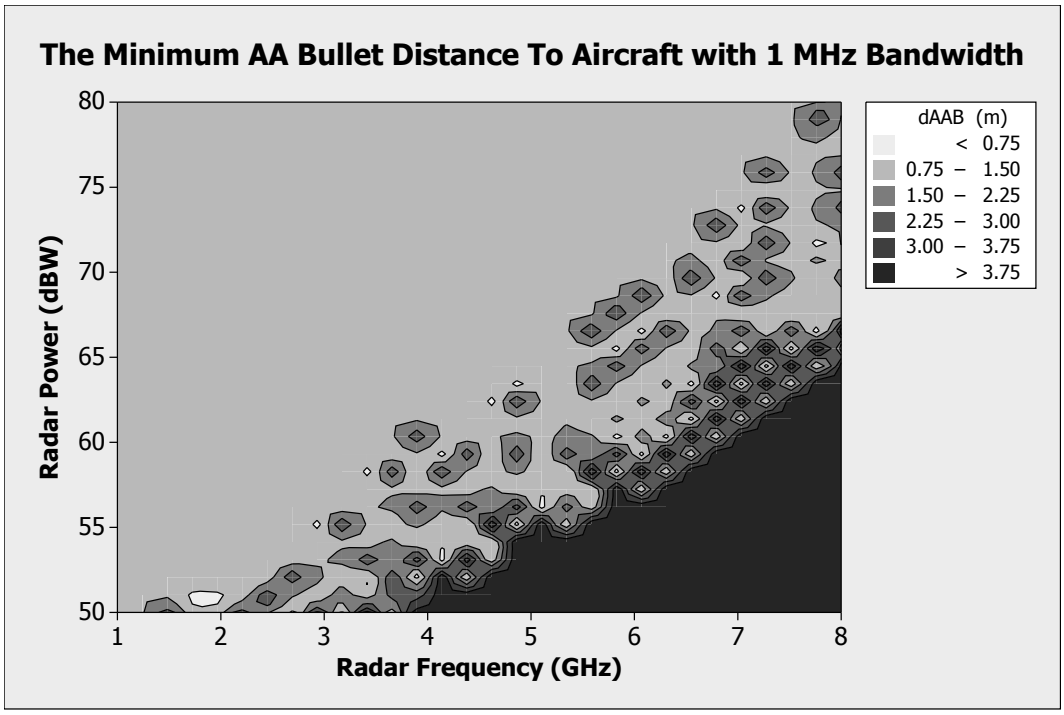


(a) Response surface of OS

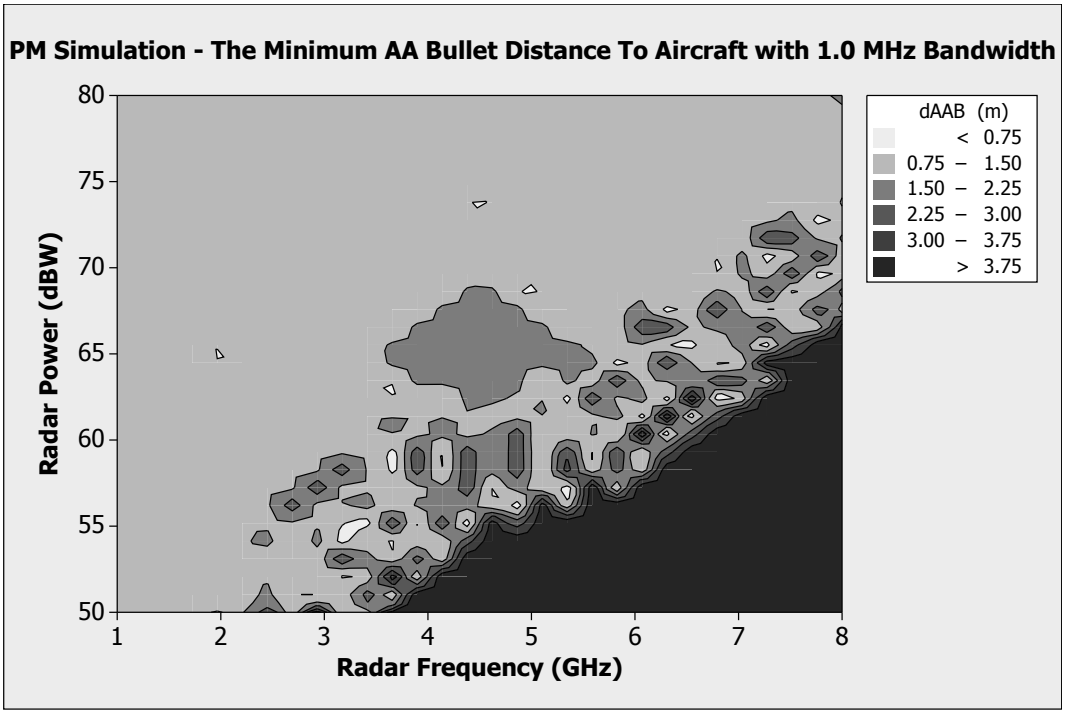


(b) Response surface of PMIS

Figure 6.2: Response surfaces of the simulations for the minimum AA bullet distance to the aircraft with 0.7 MHz Bandwidth



(a) Response surface of OS



(b) Response surface of PMIS

Figure 6.3: Response surfaces of the simulations for the minimum AA bullet distance to the aircraft with 1.0 MHz Bandwidth

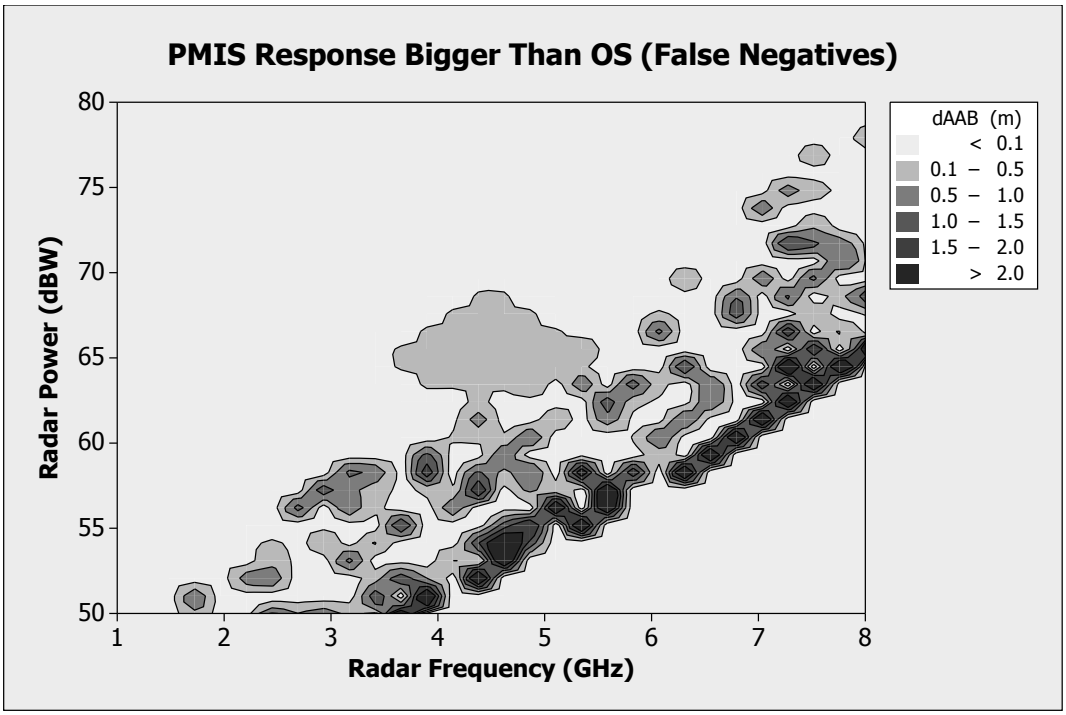


Figure 6.4: False negative value distribution of PMIS

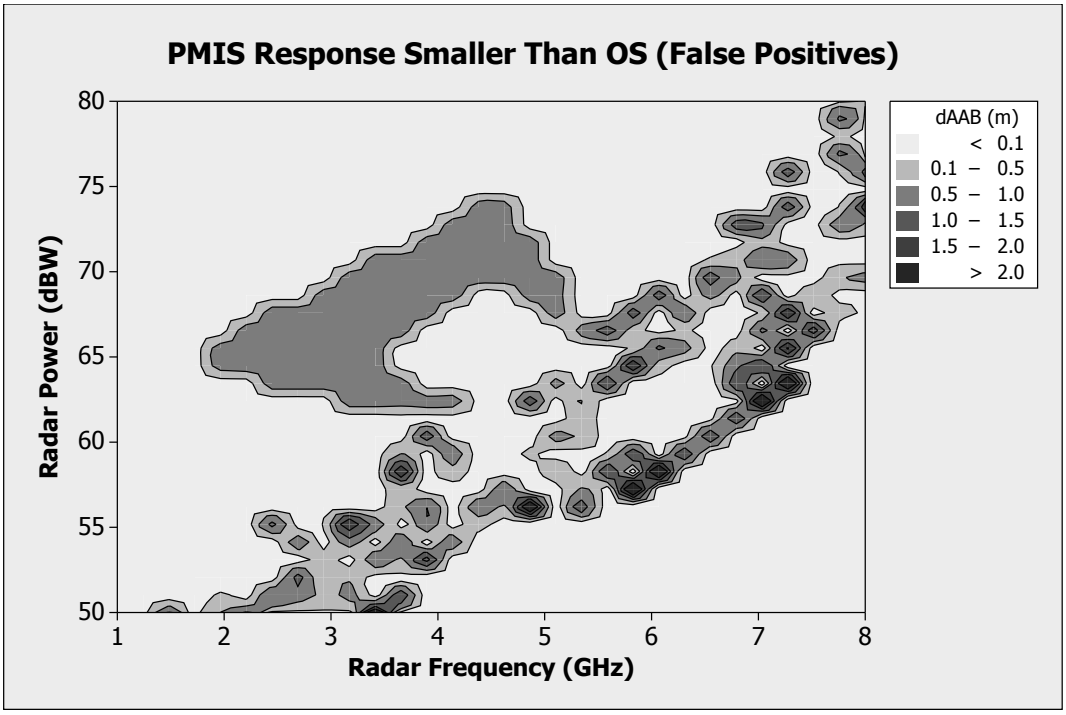


Figure 6.5: False positive value distribution of PMIS

According to these figures, the following may be concluded:

i. The difference in the responses is prominent in the risky zone.

ii. There is a region in the good zone where PMIS can generate false positive values, though the difference of the values are not too high to produce false best results.

iii. The highest values of difference between the results are at the border of the risky and bad zones.

6.2.1 Applications and Findings

In order to test the performance of PMIS during an optimization procedure, we use SwarmOps [93] as the PSO library. The parameters of the PSO are selected as the common values in the literature, which are also similar to the default parameters suggested for three-dimensional problems in the PSO library used:

i. N_p (number of particles) : 25

ii. w (inertia weight) : 0.4

iii. c1 (cognitive acceleration coefficient) : 2

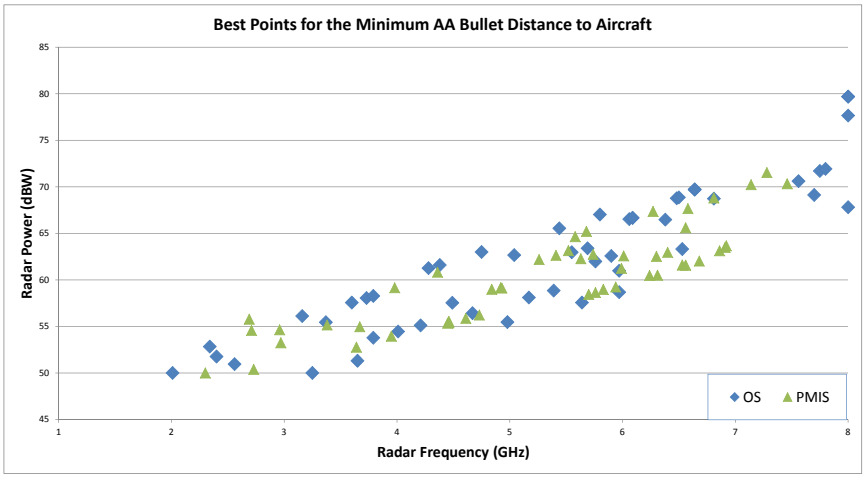
iv. c2 (social acceleration coefficient): 2

The simulation is deterministic but the optimization procedure includes a random behaviour in the particle coefficients. We also know that there are multiple best regions for the selected simulation result. The optimization procedure is run 50 times using OS. Then in order to analyze the performance of PMIS, the PM is trained using UD and the optimization procedure is run 50 times using the integrated simulation.

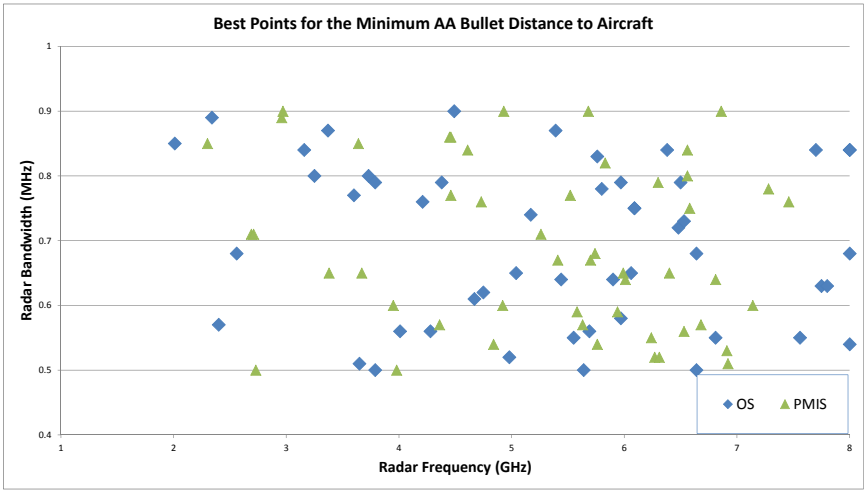
6.2.2 Results and Discussion

The best points found for OS are given in Table 6.1. They are in the range of 0.662 – 0.668. The best points found for PMIS are given in Table 6.2. They are in the range of 0.622 – 0.665, except one result value of 0.805.

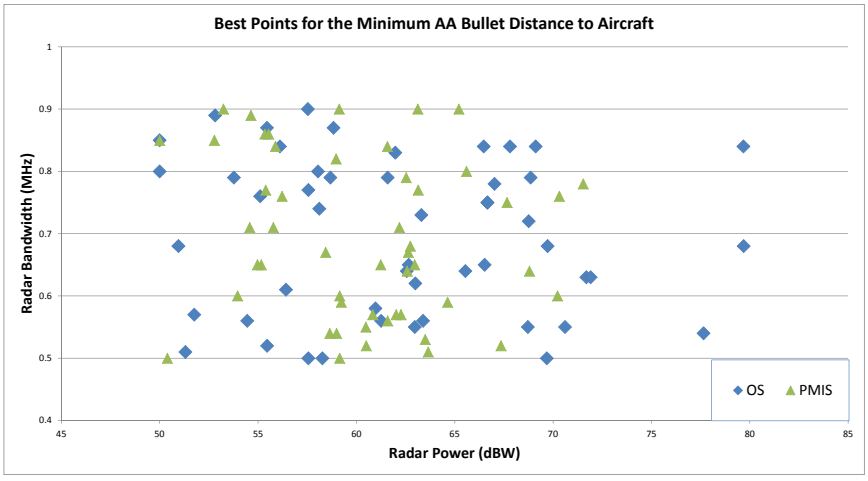
OS and PMIS best points are drawn in Figures 6.6(a)-6.6(c) and each chart represents the results in terms of frequency versus power, frequency versus bandwidth and power versus bandwidth, respectively.



(a) Best point comparison according to frequency and power



(b) Best point comparison according to frequency and bandwidth



(c) Best point comparison according to power and bandwidth

Figure 6.6: Best points comparison of OS and PMIS for d_{AAB} simulation output

Table 6.1: Best points of OS

Freq. (GHz)	Power (dBW)	Bandw. (MHz)	dAAB (m)
2.01	50	0.85	0.663
2.34	52.82	0.89	0.662
2.4	51.76	0.57	0.663
2.56	50.95	0.68	0.663
3.16	56.11	0.84	0.662
3.25	50	0.8	0.668
3.37	55.45	0.87	0.663
3.6	57.56	0.77	0.662
3.65	51.31	0.51	0.668
3.73	58.05	0.8	0.662
3.79	53.77	0.79	0.663
3.79	58.27	0.5	0.662
4.01	54.45	0.56	0.663
4.21	55.11	0.76	0.663
4.28	61.26	0.56	0.662
4.38	61.59	0.79	0.662
4.49	57.53	0.9	0.663
4.67	56.42	0.61	0.663
4.75	63	0.62	0.662
4.98	55.46	0.52	0.668
5.04	62.66	0.65	0.662
5.17	58.11	0.74	0.663
5.39	58.84	0.87	0.663
5.44	65.54	0.64	0.662
5.55	62.97	0.55	0.663
5.64	57.56	0.5	0.668
5.69	63.39	0.56	0.663
5.76	61.98	0.83	0.663
5.8	67.02	0.78	0.662
5.9	62.56	0.64	0.663
5.97	58.68	0.79	0.668
5.97	60.97	0.58	0.663
6.06	66.52	0.65	0.662
6.09	66.66	0.75	0.662
6.09	66.66	0.75	0.662
6.38	66.47	0.84	0.663
6.48	68.76	0.72	0.662
6.5	68.86	0.79	0.662
6.53	63.3	0.73	0.663
6.64	69.68	0.5	0.662
6.64	69.72	0.68	0.662
6.81	68.72	0.55	0.663
7.56	70.61	0.55	0.663
7.7	69.11	0.84	0.663
7.75	71.69	0.63	0.663
7.8	71.91	0.63	0.663
8	67.81	0.84	0.668
8	77.65	0.54	0.662
8	79.68	0.84	0.662
8	79.69	0.68	0.662

Table 6.2: Best points of PMIS

Freq. (GHz)	Power (dBW)	Bandw. (MHz)	dAAB (m)
2.3	50	0.85	0.663
2.69	55.78	0.71	0.663
2.71	54.58	0.71	0.622
2.73	50.39	0.5	0.663
2.96	54.64	0.89	0.622
2.97	53.24	0.9	0.622
3.38	55.16	0.65	0.664
3.64	52.78	0.85	0.665
3.67	54.97	0.65	0.664
3.95	53.96	0.6	0.622
3.98	59.15	0.5	0.662
4.36	60.83	0.57	0.665
4.45	55.35	0.86	0.622
4.46	55.39	0.77	0.662
4.46	55.54	0.86	0.665
4.61	55.88	0.84	0.623
4.73	56.22	0.76	0.663
4.84	58.99	0.54	0.623
4.92	59.15	0.6	0.664
4.93	59.13	0.9	0.665
5.26	62.19	0.71	0.623
5.41	62.64	0.67	0.623
5.52	63.14	0.77	0.623
5.58	64.64	0.59	0.664
5.63	62.27	0.57	0.664
5.68	65.21	0.9	0.625
5.7	58.44	0.67	0.664
5.74	62.74	0.68	0.663
5.76	58.65	0.54	0.664
5.83	58.97	0.82	0.665
5.94	59.23	0.59	0.623
5.99	61.24	0.65	0.663
6.01	62.58	0.64	0.663
6.24	60.48	0.55	0.662
6.27	67.35	0.52	0.625
6.3	62.53	0.79	0.664
6.31	60.5	0.52	0.664
6.4	62.96	0.65	0.622
6.53	61.59	0.56	0.623
6.56	61.58	0.84	0.662
6.56	65.6	0.8	0.664
6.58	67.66	0.75	0.664
6.68	62.02	0.57	0.663
6.81	68.8	0.64	0.805
6.86	63.12	0.9	0.664
6.91	63.5	0.53	0.665
6.92	63.65	0.51	0.625
7.14	70.23	0.6	0.625
7.28	71.53	0.78	0.665
7.46	70.32	0.76	0.622

According to the results,

i. The best points lay in the risky zone in both OS and PMIS runs.

ii. For the best points, we can realize an interaction between the frequency and power design variables, yet there is no obvious interaction between bandwidth and other design variables.

iii. Best points of OS and PMIS are similar in value and distribution. There is a group of best points located above the frequency value of 7.5, but PMIS did not find any best points in this region.

The high amount of false positive (FP) values can be considered as dangerous, since a FP best point value can lead to non-termination of the aircraft and mislead the simulation user. False negatives (FN) on the other hand may hinder the optimization procedure during the search. For this purpose we present the contingency table in Table 6.3 based on the hit/miss results of the simulation. We also calculate the sensitivity and the specificity of the results. Sensitivity or true positive rate $= TP/P = TP/(TP + FN)$, where TP is true-positive. Specificity or true negative rate $= TN/N = TN/(FP + TN)$ where TN is true-negative.

Table 6.3: Contingency table of the simulation results according to the minimum AA bullet distance to the aircraft

		OS Result	
PMIS Result		Hit	Miss
	Hit	722	11
	Miss	25	141
Sensitivity:	96.65%		
Specificity:	92.76%		

The results show that usage of PMIS in an optimization procedure give comparable results to the usage of OS. The sensitivity and specificity of the results are also quite high. There are some regions where PMIS can give FP results, but the threshold where these values can mislead the optimization procedure is quite high (3.75) considering the best values (0.662).

CHAPTER 7

CONCLUSION AND FURTHER STUDIES

This work has proposed a methodology for the MARS based metamodeling of dynamic discrete-event simulations with fixed-increment time advance mechanism. For this purpose, first an emulator was developed to predict the output at any time step using the data collected from the previous time steps during the simulation.

The dynamic simulation is treated like a black-box function and only the values of the design variables, time, values of the inputs and the values of the outputs can be observed. The design variables of the simulation, the input at a time step and the output in the previous time step are used to train a metamodel using the MARS method and estimate the output at the current step. MARS is particularly selected as a metamodeling technique due to its power for modeling complex (i.e, high-dimensional and nonlinear) relations. Furthermore, our approach can handle an incomplete output process where simulation does not necessarily generate an output for every time step.

The model was tested on four sampling techniques: AFD, HD, UD and LHS. In order to analyze the effect of sample size on the overall performance, several sample sizes were tried. During the training stage, the position information of both the predictors and responses were split into components. Although a MARS model can undertake feature selection and interaction analysis between predictors, the coordinate system was transformed before the training to investigate the effect of three different coordinate systems on the metamodel performance. For each of the experimental designs and for the list of sample sizes, the design points were selected, and the simulation was run for all the design points using different coordinate system representations.

The results show that a dynamic simulation model can be successfully metamodeled with MARS for the radar detection problem. The four selected sampling-representation-size combinations provide good prediction performances. The effectiveness of the MARS method for dynamic simulation metamodeling should be further evaluated by other case studies.

The methodology is further extended to decompose the simulation into smaller working pieces using the hierarchical specification of simulation models in the DEVS formalism and to build metamodels of the sub-systems. Atomic models in a DEVS simulation generate time dependent output in accordance with input events, so building a metamodel of an atomic model is essentially the dynamic simulation metamodeling problem. For this purpose, the metamodeling approach based on single time step metamodels via MARS is used. The design variables of the model, the input at the current time step and output at the previous time step are used to train a model using the MARS method and estimate the output at the current step.

Performance of the PM depends on the performance of the metamodels, which in turn depends on the type of the sampling techniques used. For this purpose, different designs are tried for testing the metamodel including LHS, HD and UD. For each of the sampling techniques, the simulation is run for all the design points and metamodels. The resulting metamodels are substituted into the PM to replace the original atomic model in the simulation.

The metamodeling technique proposed is applied to a radar simulation model in a one-on-one military engagement simulation between an attacking aircraft and a defending ship. The approach is tested by comparing the simulation results belonging to three response variables against original simulation results. Results indicate that each design has some advantage over the others for a simulation result, which can be attributed to dynamic behavior of the radar model and characteristics of the simulation results. PM-integrated simulation works significantly faster than the original simulation and PM has good prediction performance as it generates the output events for the original model.

Metamodeling of a dynamic simulation model and replacing it with a PM offer some challenges including: a large number of data values gathered from the observations at

each time step, the need for the high accuracy of the metamodel due to accumulation of errors in the simulation and the need for explorative experiments (not for the design variables but for the dynamic inputs of the atomic model) during training. Furthermore, replacing the target atomic model requires altering the connection structure of DEVS simulation, logging and parsing the events in a DEVS simulation, handling incomplete output processes where an atomic model does not generate output for each time step and forming proper output events from the predictions of the metamodel.

In the final step of this research, we used the PMIS in an optimization procedure. For this purpose, PSO optimization procedure is applied to both the OS and the PMIS to find an optimal solution in the design space for the radar system. The results were analyzed in terms of best point values, sensitivity and specificity, which show that usage of the PMIS in an optimization procedure yields comparable results to the usage of the OS. If we also consider the speed-up gain from using the PM, we can conclude that our method successfully applied to our case.

7.1 Future Research

As a future research of this work, first of all the sample case can be upgraded to allow the usage of multiple PMs in a simulation. By this way, we can see the performance effect of the number of PMs in the PMIS. Furthermore, the error in the output of PM can be analyzed sequentially in a time step basis and compared to the original atomic model for a single run of the simulation. The procedure should be repeated for every sample in the test data set. By comparing the results for single PM simulations and multi PM simulations, one can deduce the effect of PM for error accumulation.

Our metamodeling approach is inspired by the multi-output Gaussian emulator of Conti et al. [26], but we build our metamodel with certain novelties. Their emulator can be implemented and the divide and fit approach based on MARS metamodel can be compared against their gaussian emulator.

Finally, different optimization procedures can be used such as applying multi objective constrained optimization procedure or selecting different optimization methods so that their performance with PMIS can be analyzed.

REFERENCES

[1] A. Abraham, D. Steinberg, and N. Philip. Rainfall forecasting using soft computing models and multivariate adaptive regression splines. *IEEE SMC Transactions, Special Issue on Fusion of Soft Computing and Hard Computing in Industrial Applications*, 2001.

[2] J. Allison, M. Kokkolaras, and P. Papalambros. Optimal partitioning and coordination decisions in decomposition-based design optimization. *Journal of Mechanical Design*, 131(8):81–88, 2009.

[3] P. Austin. A comparison of regression trees, logistic regression, generalized additive models, and multivariate adaptive regression splines for predicting ami mortality. *Statistics in medicine*, 26(15):2937–2957, 2007.

[4] M. Balshi, A. McGuire, P. Duffy, M. Flannigan, J. Walsh, and J. Melillo. Assessing the response of area burned to changing climate in western boreal north america using a multivariate adaptive regression splines approach. *Global Change Biology*, 15(3):578–600, 2009.

[5] R. Barton. Simulation metamodels. In *Simulation Conference Proceedings, 1998. Winter*, volume 1, pages 167–174. IEEE, 1998.

[6] R. Barton and M. Meckesheimer. Metamodel-based simulation optimization. *Handbooks in operations research and management science*, 13:535–574, 2006.

[7] I. Batmaz and S. Tunali. Small response surface designs for metamodel estimation. *European Journal of Operational Research*, 145(2):455–470, 2003.

[8] D. Beasley, R. Martin, and D. Bull. An overview of genetic algorithms: Part 1. Fundamentals. *University computing*, 15:58–58, 1993.

[9] Y. Bengio and P. Frasconi. An input output hmm architecture. In *Advances in neural information processing systems*, pages 427–434, 1995.

[10] Y. Bengio and P. Frasconi. Input-output hmms for sequence processing. *Neural Networks, IEEE Transactions on*, 7(5):1231–1249, 1996.

[11] S. Bhattacharya. A simulation approach to bayesian emulation of complex dynamic computer models. *Bayesian Analysis*, 2(4):783–816, 2007.

[12] C. Bloebaum, P. Hajela, and J. Sobieszczanski-Sobieski. Non-hierarchic system decomposition in structural optimization. *Engineering optimization*, 19(3):171–186, 1992.

[13] H. Boudali and J. Dugan. A discrete-time bayesian network reliability modeling and analysis framework. *Reliability Engineering & System Safety*, 87(3):337–349, 2005.

[14] H. Boudali and J. Dugan. A continuous-time bayesian network reliability modeling, and analysis framework. *Reliability, IEEE Transactions on*, 55(1):86–97, 2006.

[15] G. Box and K. Wilson. On the experimental attainment of optimum conditions. *Journal of the Royal Statistical Society. Series B (Methodological)*, 13(1):1–45, 1951.

[16] R. Braun, P. Gage, I. Kroo, and I. Sobieski. Implementation and performance issues in collaborative optimization. *Analysis*, 1(x1):y1j, 1996.

[17] T. Browning. Applying the design structure matrix to system decomposition and integration problems: a review and new directions. *Engineering Management, IEEE Transactions on*, 48(3):292–306, 2001.

[18] T. Calvo, G. Mayor, and R. Mesiar. *Aggregation operators: new trends and applications*, volume 97. Physica Verlag, 2002.

[19] O. Cappé, S. Godsill, and E. Moulines. An overview of existing methods and recent advances in sequential monte carlo. *Proceedings of the IEEE*, 95(5):899–924, 2007.

[20] R. Chelouah and P. Siarry. A continuous genetic algorithm designed for the global optimization of multimodal functions. *Journal of Heuristics*, 6(2):191–213, 2000.

[21] L. CHEN, Z. DING, and S. LI. A formal two-phase method for decomposition of complex design problems. *Journal of mechanical design*, 127(2):184–195, 2005.

[22] T. Chen, K. Hadinoto, W. Yan, and Y. Ma. Efficient meta-modelling of complex process simulations with time-space-dependent outputs. *Computers & Chemical Engineering*, 2010.

[23] V. Chen, K. Tsui, R. Barton, and M. Meckesheimer. A review on design, modeling and applications of computer experiments. *IIE Transactions*, 38(4):273–291, 2006.

[24] R. Cheng. Regression metamodeling in simulation using bayesian methods. In *Proceedings of the 31st conference on Winter simulation: Simulation—a bridge to the future-Volume 1*, pages 330–335. ACM, 1999.

[25] J. Connor, R. Martin, and L. Atlas. Recurrent neural networks and robust time series prediction. *Neural Networks, IEEE Transactions on*, 5(2):240–254, 1994.

[26] S. Conti, J. P. Gosling, J. Oakley, and A. O'Hagan. Gaussian process emulation of dynamic computer codes. *Biometrika*, 96(3):663–676, 2009.

[27] S. Conti and A. O'Hagan. Bayesian emulation of complex multi-output and dynamic computer models. *Journal of Statistical Planning and Inference*, 140(3):640–651, 2010.

[28] N. Cressie. Spatial prediction and ordinary kriging. *Mathematical Geology*, 20(4):405–421, 1988.

[29] M. Cuéllar, M. Delgado, and M. Pegalajar. An application of non-linear programming to train recurrent neural networks in time series prediction problems. *Enterprise Information Systems VII*, pages 95–102, 2006.

[30] R. D. De Veaux, D. Psichogios, and L. Ungar. A comparison of two nonparametric estimation schemes: Mars and neural networks. *Computers & Chemical Engineering*, 17(8):819–837, 1993.

[31] T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(2):142–150, 1989.

[32] E. Deconinck, Q. Xu, R. Put, D. Coomans, D. Massart, and Y. Vander Heyden. Prediction of gastro-intestinal absorption using multivariate adaptive regression splines. *Journal of pharmaceutical and biomedical analysis*, 39(5):1021–1030, 2005.

[33] K. Dimopoulos, C. Kambhampati, and R. Craddock. Efficient recurrent neural network training incorporating a priori knowledge. *Mathematics and Computers in Simulation*, 52(2):137–162, 2000.

[34] A. Doucet and A. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later, 2009.

[35] H. Drucker, C. Burges, L. Kaufman, A. Smola, and V. Vapnik. Support vector regression machines. *Advances in Neural Information Processing Systems*, pages 155–161, 1997.

[36] X. Du and W. Chen. Concurrent subsystem uncertainty analysis in multidisciplinary design. In *Long Beach. The 8th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization. CA: AIAA. AIAA-00-4928*, pages 1–11, 2000.

[37] X. Du and W. Chen. Methodology for managing the effect of uncertainty in simulation-based design. *AIAA journal*, 38(8):1471–1478, 2000.

[38] N. Dyn, D. Levin, and S. Rippa. Numerical procedures for surface fitting of scattered data by radial functions. *SIAM Journal on Scientific and Statistical Computing*, 7:639, 1986.

[39] R. Eberhart and Y. Shi. Comparison between genetic algorithms and particle swarm optimization. *Lecture notes in computer science*, pages 611–618, 1998.

[40] T. Ender, R. Leurck, B. Weaver, P. Miceli, W. Blair, P. West, and D. Mavris. Systems-of-systems analysis of ballistic missile defense architecture effectiveness through surrogate modeling and simulation. *IEEE Systems Journal*, 4:156–166, 2010.

[41] K. Fang. The uniform design: application of number-theoretic methods in experimental design. *Acta Math. Appl. Sinica*, 3:363–372, 1980.

[42] A. Forrester and A. Keane. Recent advances in surrogate-based optimization. *Progress in Aerospace Sciences*, 45(1-3):50–79, 2009.

[43] B. Freisleben and P. Merz. A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems. In *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation*, pages 616–621. IEEE Press, 1996.

[44] J. Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, 19(1):1–67, 1991.

[45] L. Friedman. *The simulation metamodel*. Kluwer Academic Publishers Norwell, Massachusetts, 1996.

[46] M. Fu, F. Glover, and J. April. Simulation optimization: a review, new developments, and applications. In *Proceedings of the 37th conference on Winter simulation*, page 95. Winter Simulation Conference, 2005.

[47] P. Geyer. Component-oriented decomposition for multidisciplinary design optimization in building design. *Advanced Engineering Informatics*, 23(1):12–31, 2009.

[48] Z. Ghahramani. Learning dynamic bayesian networks. *Adaptive Processing of Sequences and Data Structures*, page 168, 1998.

[49] D. Goldberg. *Genetic Algorithms in Search and Optimization*. Addison-wesley, 1989.

[50] R. B. Gramacy and H. K. Lee. Bayesian treed gaussian process models with an application to computer modeling. *Journal of the American Statistical Association*, 103(483):1119–1130, 2008.

[51] A. Grosso, A. Jamali, and M. Locatelli. Finding maximin latin hypercube designs by iterated local search heuristics. *European Journal of Operational Research*, 197(2):541–547, 2009.

[52] F. Gu and X. Hu. Towards applications of particle filters in wildfire spread simulation. In *Proceedings of the 40th Conference on Winter Simulation*, pages 2852–2860. Winter Simulation Conference, 2008.

[53] A. Gurnani and K. Lewis. Decentralized design at the edge of rationality. *DETC2006-99520*, 2006.

[54] J. Hall and C. Clark. Sensorcraft mission simulation study. Technical report, DTIC Document, 2002.

[55] J. C. Helton and F. J. Davis. Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems. *Reliability Engineering & System Safety*, 81(1):23–69, 2003.

[56] J. Holland. *Adaptation in natural and artificial systems*. MIT press Cambridge, MA, 1992.

[57] A. Kara, F. Deniz, D. Bozagac, and N. Alpdemir. SIMA: A DEVS based hierarchical and modular modelling and simulation framework. In *Proceedings of the 2009 Summer Computer Simulation Conference*. Society for Computer Simulation International Istanbul, Turkey, 2009.

[58] M. Karam and M. Zohdy. Nonlinear model-based dynamic recurrent neural network. In *Circuits and Systems, 2001. MWSCAS 2001. Proceedings of the 44th IEEE 2001 Midwest Symposium on*, volume 2, pages 624–626. IEEE, 2001.

[59] P. Karimian and J. Herrmann. Separating design optimization problems into decision-based design processes. *Journal of mechanical design*, 131:011007, 2009.

[60] E. Kartal. Metamodeling complex systems using linear and nonlinear refression methods. Master's thesis, Middle East Technical University, 2007.

[61] E. Kartal-Koç, İ. Batmaz, and G. Weber. Robust regression metamodelling of complex systems: The case of solid rocket motor performance metamodelling. *Advances in Intelligent Modelling and Simulation*, pages 221–251, 2012.

[62] J. Kennedy. Small worlds and mega-minds: effects of neighborhood topology onparticle swarm performance. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 3, 1999.

[63] J. Kennedy, R. Eberhart, et al. Particle swarm optimization. In *Proceedings of IEEE international conference on neural networks*, volume 4, pages 1942–1948. Piscataway, NJ: IEEE, 1995.

[64] J. Kennedy, R. Eberhart, and Y. Shi. *Swarm Intelligence*. Springer, 2001.

[65] M. Kennedy and A. O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 63(3):425–464, 2001.

[66] J. P. Kleijnen. Sensitivity analysis of simulation experiments: regression analysis and statistical design. *Mathematics and Computers in Simulation*, 34(3):297–315, 1992.

[67] J. P. Kleijnen. Kriging metamodeling in simulation: a review. *European Journal of Operational Research*, 192(3):707–716, 2009.

[68] M. Kokkolaras, Z. Mourelatos, and P. Papalambros. Design optimization of hierarchically decomposed multilevel systems under uncertainty. *Journal of Mechanical Design*, 128:503, 2006.

[69] A. Konstantinidis, Q. Zhang, and K. Yang. A subproblem-dependent heuristic in moea/d for the deployment and power assignment problem in wireless sensor networks. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*, pages 2740–2747. IEEE, 2009.

[70] N. Krasnogor and J. Smith. Competent memetic algorithms: model, taxonomy and design issues'. *IEEE Transactions on Evolutionary Computation*, 9:474–488, 2005.

[71] A. Kusiak and N. Larson. Decomposition and representation methods in mechanical design. *Journal of mechanical design*, 117(B):17–24, 1995.

[72] A. Law, W. Kelton, and W. Kelton. *Simulation modeling and analysis*, volume 2. McGraw-Hill, New York, 1991.

[73] J. Leathwick, J. Elith, and T. Hastie. Comparative performance of generalized additive models and multivariate adaptive regression splines for statistical modelling of species distributions. *Ecological Modelling*, 199(2):188–196, 2006.

[74] T. Lee, C. Chiu, Y. Chou, and C. Lu. Mining the customer credit using classification and regression tree and multivariate adaptive regression splines. *Computational Statistics & Data Analysis*, 50(4):1113–1130, 2006.

[75] P. Lewis and J. Stevens. Nonlinear modeling of time series using multivariate adaptive regression splines (mars). *Journal of the American Statistical Association*, pages 864–877, 1991.

[76] D. Lim, Y. Ong, Y. Jin, B. Sendhoff, and B. Lee. Efficient hierarchical parallel genetic algorithms using grid computing. *Future Generation Computer Systems*, 23(4):658–670, 2007.

[77] F. Liu and M. West. A dynamic modelling strategy for bayesian computer model emulation. *Bayesian Analysis*, 4(2):393–411, 2009.

[78] C.-J. Lu, T.-S. Lee, and C.-M. Lian. Sales forecasting for computer wholesalers: A comparison of multivariate adaptive regression splines and artificial neural networks. *Decision Support Systems*, 54(1):584–596, 2012.

[79] D. G. Luenberger. *Introduction to dynamic systems: Theory, models, and applications*. Wiley (New York), 1979.

[80] D. Lum. A methodology for modeling radar detection range in air combat simulation. *MS Thesis Air Force Inst. of Tech., Wright-Patterson AFB, OH.*, 1, 1989.

[81] M. McKay, R. Beckman, and W. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, pages 239–245, 1979.

[82] N. Michelena and P. Papalambros. A hypergraph framework for optimal model-based decomposition of design problems. *Computational Optimization and Applications*, 8(2):173–196, 1997.

[83] N. Michelena, H. Park, and P. Papalambros. Convergence properties of analytical target cascading. *AIAA journal*, 41(5):897–905, 2003.

[84] G. G. Moisen and T. S. Frescino. Comparing five modelling techniques for predicting forest characteristics. *Ecological Modelling*, 157(2):209–225, 2002.

[85] D. C. Montgomery, V. M. Bettencourt, et al. Multiple response surface methods in computer simulation. *Simulation*, 29(4):113–121, 1977.

[86] M. D. Morris and T. J. Mitchell. Exploratory designs for computational experiments. *Journal of Statistical Planning and Inference*, 43(3):381–402, 1995.

[87] S. Mostaghim and J. Teich. Covering Pareto-optimal fronts by subswarms in multi-objective particle swarm optimization. In *2004 Congress on Evolutionary Computation (CEC 2004)*, volume 2, pages 1404–1411, 2004.

[88] S. Mukkamala, A. Sung, A. Abraham, and V. Ramos. Intrusion detection systems using adaptive regression spines. *Enterprise Information Systems VI*, pages 211–218, 2006.

[89] R. Myers, D. Montgomery, and C. Anderson-Cook. *Response surface methodology: process and product optimization using designed experiments*, volume 705. John Wiley & Sons Inc, New Jersey, 2009.

[90] J. Otto, M. Paraschivoiu, S. Yesilyurt, and A. T. Patera. Bayesian-validated computer-simulation surrogates for optimization and design: error estimates

and applications. *Mathematics and computers in simulation*, 44(4):347–367, 1997.

[91] M. Papadrakakis, N. Lagaros, and Y. Tsompanakis. Structural optimization using evolution strategies and neural networks. *Computer methods in applied mechanics and engineering*, 156(1-4):309–333, 1998.

[92] P. Papalambros, N. Michelena, and N. Kikuchi. Distributed cooperative systems design. pages 265–270, 1997.

[93] M. Pedersen. *SwarmOps*. Hvass Laboratories, 3.1 edition, January 2011. Numerical and Heuristic Optimization Tool.

[94] J. Poropudas and K. Virtanen. Analyzing air combat simulation results with dynamic bayesian networks. In *Proceedings of the 39th conference on Winter simulation: 40 years! The best is yet to come*, pages 1370–1377. IEEE Press, 2007.

[95] J. Poropudas and K. Virtanen. Simulation metamodeling through dynamic bayesian networks, 2009.

[96] A. Prasad, L. Iverson, and A. Liaw. Newer classification and regression tree techniques: bagging and random forests for ecological prediction. *Ecosystems*, 9(2):181–199, 2006.

[97] S. Rao. Game theory approach for multiobjective structural optimization. *Computers & structures*, 25(1):119–127, 1987.

[98] J. Rodriguez. Metamodeling techniques to aid in the aggregation process of large hierarchical simulation models, 2008.

[99] K. Roquemore. Hybrid designs for quadratic response surfaces. *Technometrics*, pages 419–423, 1976.

[100] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn. Design and analysis of computer experiments. *Statistical science*, 4(4):409–423, 1989.

[101] F. Schoen. Stochastic techniques for global optimization: A survey of recent advances. *Journal of Global Optimization*, 1(3):207–228, 1991.

[102] L. W. Schruben and V. J. Cogliano. An experimental procedure for simulation response surface model identification. *Communications of the ACM*, 30(8):716–730, 1987.

[103] J. Schutte, J. Reinbolt, B. Fregly, R. Haftka, and A. George. Parallel global optimization with the particle swarm algorithm. *International journal for numerical methods in engineering*, 61(13):2296, 2004.

[104] S. Shan and G. Wang. Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions. *Structural and Multidisciplinary Optimization*, 41(2):219–241, 2010.

[105] Y. Shi. Particle swarm optimization. *IEEE Connections*, 2(1):8–13, 2004.

[106] T. Simpson, D. Lin, and W. Chen. Sampling strategies for computer experiments: Design and analysis. *International Journal of Reliability and Applications*, 2(3):209–240, 2001.

[107] M. Stein. Large sample properties of simulations using latin hypercube sampling. *Technometrics*, 29(2):143–151, 1987.

[108] J. Swisher, P. Hyden, S. Jacobson, and L. Schruben. Simulation optimization: a survey of simulation optimization techniques and procedures. In *Proceedings of the 32nd conference on Winter simulation*, pages 119–128. Society for Computer Simulation International San Diego, CA, USA, 2000.

[109] R. Team et al. R: A language and environment for statistical computing. *R Foundation for Statistical Computing*, (01/19), 2010.

[110] E. Tekin and I. Sabuncuoglu. Simulation optimization: A comprehensive review on theory and applications. *IIE Transactions*, 36(11):1067–1081, 2004.

[111] R. Thakker, M. Patil, and K. Anil. Parameter extraction for PSP MOSFET model using hierarchical particle swarm optimization. *Engineering Applications of Artificial Intelligence*, 22(2):317–328, 2009.

[112] T. Wagner and P. Papalambros. General framework for decomposition analysis in optimal design. *Advances in Design Automation*, 65:315–325, 1993.

[113] G. Wang and S. Shan. Review of metamodeling techniques in support of engineering design optimization. *Journal of Mechanical Design*, 129:370, 2007.

[114] P. Weber and L. Jouffe. Complex system reliability modelling with dynamic object oriented bayesian networks (doobn). *Reliability Engineering & System Safety*, 91(2):149–162, 2006.

[115] R. Welch, S. Ruffing, and G. Venayagamoorthy. Comparison of feedforward and feedback neural network architectures for short term wind speed prediction. In *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*, pages 3335–3340. IEEE, 2009.

[116] C.-C. Yang, S. O. Prasher, R. Lacroix, and S. H. Kim. A multivariate adaptive regression splines model for simulation of pesticide transport in soils. *Biosystems engineering*, 86(1):9–15, 2003.

[117] J. Yang, J. Horng, and C. Kao. A continuous genetic algorithm for global optimization. In *Proceedings of the Seventh International Conference on Genetic Algorithms, Michigan State University, East Lansing, MI, July 19-23, 1997*, page 230. Morgan Kaufmann Publishers, 1997.

[118] T. York, L. Eaves, et al. Common disease analysis using multivariate adaptive regression splines (mars): Genetic analysis workshop 12 simulated sequence data. *Genetic Epidemiology*, 21:S649, 2001.

[119] D. Zaharie, D. Petcu, and S. Panica. A Hierarchical Approach in Distributed Evolutionary Algorithms for Multiobjective Optimization. *Lecture Notes In Computer Science*, 4818:516, 2008.

[120] H. Zareipour, K. Bhattacharya, and C. Canizares. Forecasting the hourly ontario energy price by multivariate adaptive regression splines. In *Power Engineering Society General Meeting, 2006. IEEE*, pages 7–pp. IEEE, 2006.

[121] B. Zeigler, H. Praehofer, and T. Kim. *Theory of modeling and simulation*, volume 100. Academic press, California, 2000.

[122] M. Zelen. The use of group divisible designs for confounded asymmetrical factorial arrangements. *The Annals of Mathematical Statistics*, pages 22–40, 1958.

[123] J. Zhang, A. Morris, E. Martin, and C. Kiparissides. Prediction of polymer quality in batch polymerisation reactors using robust neural networks. *Chemical Engineering Journal*, 69(2):135–143, 1998.

APPENDIX

MARS Metamodel Knots

Table 7.1: Knots of CMBD-UD-15 MARS Output metamodel

Detection X Output:	
0.6035	
+0.953	* max(0, x[SgVPXYZ] - 0.6918)
-0.948	* max(0, 0.6918 - x[SgVPXYZ])
-0.0536	* max(0, x[PlPsXYZ] - 1.019)
+0.0509	* max(0, 1.019 - x[PlPsXYZ])
Detection Y Output:	
0.5147	
+0.928	* max(0, x[SgVPXYZ] - 0.6918)
-0.885	* max(0, 0.6918 - x[SgVPXYZ])
+0.0228	* max(0, x[PlPsXYZ] - 1.164)
+0.0993	* max(0, 1.164 - x[PlPsXYZ])
+0.0995	* max(0, 1.164 - x[PlPsXYZ]) * max(0, x[SgVPXYZ] - 1.227)
-0.0405	* max(0, 1.164 - x[PlPsXYZ]) * max(0, 1.227 - x[SgVPXYZ])
-0.0235	* max(0, 1.164 - x[PlPsXYZ]) * max(0, 1.227 - x[SgVPXYZ]) * max(0, x[DtLsY] - -0.834)
-0.00497	* max(0, 1.164 - x[PlPsXYZ]) * max(0, 1.227 - x[SgVPXYZ]) * max(0, -0.834 - x[DtLsY])
-0.0381	* max(0, x[PlPsXYZ] - -1.498) * max(0, x[SgVPXYZ] - 0.6918)
+0.0764	* max(0, -1.498 - x[PlPsXYZ]) * max(0, x[SgVPXYZ] - 0.6918)
Detection Z Output:	
0.9276	
+1.12	* max(0, x[SgVPXYZ] - 0.6492)
-1.18	* max(0, 0.6492 - x[SgVPXYZ])
-0.0303	* max(0, x[PlPsXYZ] - 1.2)
-0.158	* max(0, 1.2 - x[PlPsXYZ])
-0.147	* max(0, 1.2 - x[PlPsXYZ]) * max(0, x[SgVPXYZ] - 1.187)
+0.0617	* max(0, 1.2 - x[PlPsXYZ]) * max(0, 1.187 - x[SgVPXYZ])
+0.0341	* max(0, 1.2 - x[PlPsXYZ]) * max(0, 1.187 - x[SgVPXYZ]) * max(0, x[DtLsY] - -0.8945)
+0.00773	* max(0, 1.2 - x[PlPsXYZ]) * max(0, 1.187 - x[SgVPXYZ]) * max(0, -0.8945 - x[DtLsY])
+0.0461	* max(0, x[PlPsXYZ] - -1.439) * max(0, x[SgVPXYZ] - 0.6492)
-0.0945	* max(0, -1.439 - x[PlPsXYZ]) * max(0, x[SgVPXYZ] - 0.6492)

Table 7.2: Knots of CMBD-UD-15 MARS OutputCntrl metamodel

Detection Exists:	
1.613	
-0.495	* max(0, x[DtLsZ] - 0.4785)
-1.06	* max(0, 0.4785 - x[DtLsZ])
-0.226	* max(0, x[SgVPXYZ] - -0.2213) * max(0, x[DtLsZ] - 0.4785)
+0.338	* max(0, 0.8533 - x[SgVPXYZ]) * max(0, 0.4785 - x[DtLsZ])
-183	* max(0, x[SgVPXYZ] - 0.957)
-0.5	* max(0, 0.957 - x[SgVPXYZ])
+89.7	* max(0, x[SgVPXYZ] - 0.946) * max(0, 0.4785 - x[DtLsZ])
-260	* max(0, x[PlPsXYZ] - -1.867) * max(0, x[SgVPXYZ] - 0.946) * max(0, 0.4785 - x[DtLsZ])
-0.318	* max(0, x[Freq] - -0.7069) * max(0, x[SgVPXYZ] - 0.8533) * max(0, 0.4785 - x[DtLsZ])
+3.02	* max(0, -0.7069 - x[Freq]) * max(0, x[SgVPXYZ] - 0.8533) * max(0, 0.4785 - x[DtLsZ])
+34.7	* max(0, x[Freq] - -1.414) * max(0, x[SgVPXYZ] - 0.957)
+2.41	* max(0, x[Freq] - -0.7069) * max(0, x[PlPsXYZ] - -1.604) * max(0, x[SgVPXYZ] - 0.8533) * max(0, 0.4785 - x[DtLsZ])
-19.7	* max(0, x[Freq] - -0.7069) * max(0, -1.604 - x[PlPsXYZ]) * max(0, x[SgVPXYZ] - 0.8533) * max(0, 0.4785 - x[DtLsZ])
+0.405	* max(0, x[Power]) * max(0, x[SgVPXYZ] - 0.8533) * max(0, 0.4785 - x[DtLsZ])
-0.415	* max(0, 0 - x[Power]) * max(0, x[SgVPXYZ] - 0.8533) * max(0, 0.4785 - x[DtLsZ])

GLOSSARY

Complete observation table	Observation table produced by running the simulation for all design points of the sample set and recording input and output for all time steps
Design variable	Variable that is controllable from the point of view of the designer (i.e., simulation user)
Design space	Multidimensional combination of design variables with acceptable ranges
Dynamic system	A system that generates outputs over time using the dynamic inputs and/or the state of the system
Incomplete output process	A simulation model which does not produce an output in some steps due to missing inputs or internal behavior
Input parameter	Uncontrollable parameters from the point of view of the designer (e.g., environment conditions)
Original simulation	Simulation with original atomic model (radar system in the sample case)
Predictor variable	Independent variable that can be used to predict the output of a model
PM integrated simulation	Simulation in which a target atomic model is replaced with a proxy atomic model (PM) implementing its metamodel
Reduced observation table	Observation table produced by trimming complete observation table so that only the rows with output data remain in the table
Response	Dependent variable that changes in accordance with changes in the predictors
Response surface	Surface plots that visualize the dependence of a response on the predictors by presenting contours of the response values

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Bozağaç, Cumhur Doruk
Nationality: Turkish (TC)
Date and Place of Birth: 29.10.1981, Mersin
Marital Status: Married
Phone: 0 312 2916059
Fax: 0 312 2916004

EDUCATION

Degree	Institution	Year of Graduation
M.S.	Computer Engineering, Bilkent University	2006
B.S.	Computer Engineering, Bilkent University	2003
High School	İçel Anatolian High School	1999

PROFESSIONAL EXPERIENCE

Year	Place	Enrollment
2011-Present	TUBİTAK BİLGEM İLTAREN	Chief Researcher
2007-2011	TUBİTAK BİLGEM İLTAREN	Senior Researcher
2003-2007	TUBİTAK BİLGEM İLTAREN	Research Scientist

PUBLICATIONS

Bozağaç D., Karaduman G., Kara A., Alpdemir M. N., "Sim-PETEK: A Parallel Simulation Execution Framework for Grid Environments." The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology. SAGE Publications, ISSN=1548-5129, 2012)

Bozağaç D., Karaduman G., Kara A., Alpdemir M. N., "Sim-PETEK: A Parallel Simulation Execution Framework for Grid Environments." Summer Computer Simulation Conference, SCSC 2009, İstanbul, Turkey, 13-16 July 2009, pp.275-282

Bozağaç D., Karaduman G., Alpdemir M. N., "Sim-PETEK: Simulasyonlar için Paralel ve Dağıtık Koşum Altyapısı" 3. Ulusal Savunma Uygulamaları Modelleme ve Simulasyon Konferansı 2009, USMOS 2009, Ankara, Turkey, 17-18 Haziran 2009

Kara A., Deniz F., Bozağaç D., Alpdemir M. N., "Simulation Modeling Architecture (SiMA), A DEVS Based Modeling and Simulation Framework." Summer Computer Simulation Conference, SCSC 2009, İstanbul, Turkey, 13-16 July 2009, pp.315-321

Kara A., Bozağaç D., Alpdemir M. N., "Simulasyon Modelleme Altyapısı (SiMA): DEVS Tabanlı Hiyerarşik ve Modüler bir Modelleme ve Koşum Altyapısı." İkinci Ulusal Savunma Uygulamaları Modelleme ve Simulasyon Konferansı, USMOS 2007, 18-19 Nisan 2007, ODTÜ, Ankara

Bozağaç D., "Ghostware and Rootkit Detection Techniques for Windows." M.Sc Thesis Bilkent University · Institute of Engineering and Sciences. Computer Engineering Ankara, 2006 xi, 51 p.