University of Southampton

Faculty of Engineering, Science and Mathematics, School of Electronics and Computer Science

Support-vector-machine based automatic performance modelling and optimisation for analogue and mixed-signal designs

by

Xianqiang Ren

A thesis submitted for the degree of Doctor of Philosophy

October, 2008

UNIVERSITY OF SOUTHAMPTON
FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

ABSTRACT

Doctor of Philosophy

Support-vector-machine based automatic performance modelling and optimisation for analogue and mixed-signal designs

by Xianqiang Ren

The growing popularity of analogue and mixed-signal (AMS) ASIC and SoC designs for communication applications has led to an increasing requirement for high efficiency performance modelling and optimisation methodologies in AMS synthesis systems. Recently, the support vector machine (SVM) method has been introduced into this challenging field. This research has studied the application of SVMs to AMS performance modelling in terms of the computational cost and prediction accuracy. A novel, general performance modelling methodology which could be applied to an arbitrary AMS system has been developed and integrated into an AMS performance optimisation system.

The contributions of this research can be summarised as follows: firstly, a new performance modelling methodology based on automatic generation of knowledge databases for AMS performance modelling using SVM techniques has been developed. Two performance model construction methods have been implemented: a linearly graded method and a support vector regression method. They can provide a basis for efficient design space exploration. Both methods construct performance models for AMS designs in a fully automatic way. A simulator, a performance extractor and an SVM trainer have been developed and integrated into a practical demonstrator system.

Secondly, a knowledge-based AMS performance optimisation system has been developed for system-level and circuit-level designs. Knowledge data bases created using the proposed methodology are reusable. This has been verified by the application of two optimisation methods, a dedicated genetic optimisation algorithm and the standard pattern search technique.

Finally, the proposed performance model construction methodology and the underlying performance optimisation system have been validated using two complex case studies. The first example is a high-level model of a mixed-signal sigma-delta modulator which comprises most of the practical design nonlinearities and imperfections that are known to affect the performance. The proposed method is able to find designs with significantly superior key performance figures compared to those obtained by the standard sigma-delta modulator design procedure. The second example is a radio frequency (RF) range Colpitts filter for silicon implementations. The non-standard structure of this filter with a lossy spiral inductor makes this type of circuit difficult to handle by standard filter synthesis methods. The proposed performance optimisation method has found solutions which satisfy performance requirements in cases where a standard manual filter design procedure failed.

Contents

List of Figures

List of Tables

Abbreviation

A/D	**A**nalogue/**D**igital
AC	**A**lternating **C**urrent
ADC	**A**nalogue-to-**D**igital **C**onverter
AMS	**A**nalogue and **M**ixed-**S**ignal
ASIC	**A**pplication **S**pecific **IC**
BDG	**B**alanced **D**ata **G**rading
CAD	**C**omputer **A**ided **D**esign
CGS	**C**oarse **G**rid **S**earch
CMOS	**C**omplementary **M**etal-**O**xide **S**emiconductor
DAC	**D**igital to **A**nalogue **C**onverter
DAE	**D**ifferential and **A**lgebraic **E**quation
DC	**D**irect **C**urrent
DR	**D**ynamic **R**ange
ENOB	**E**ffective **N**umber **O**f **B**its
ERM	**E**mpirical **R**isk **M**inimization
FFT	**F**ast **F**ourier **T**ransform
GA	**G**enetic **A**lgorithm
HDL	**H**ardware **D**esign **L**anguage
I/O	**I**nput and **O**utput
IC	**I**ntegrated **C**ircuit
ISAR	**I**nput **S**ignal **A**mplitude **R**ange
KKT	**K**arush **K**uhn **T**ucker
MASH	**M**ulti-st**A**ge noise-**SH**aping
MSE	**M**ean **S**quared **E**rror
NTF	**N**oise **T**ransfer **F**unction
OpAmp	**Op**erational **Amp**lifier
OSR	**O**ver**S**ampling **R**atio
PSD	**P**ower **S**pectral **D**ensity

RBF	**R**adial **B**asis **F**unction
RF	**R**adio **F**requency
RGS	**R**efined **G**rid **S**earch
RMS	**R**oot-**M**ean **S**quare
RTL	**R**egistor **T**ransistor **L**evel
SC	**S**witched **C**apacitor
SDM	**S**igma-**D**elta **M**odulator
SFG	**S**ignal-**F**low **G**raph
SNR	**S**ignal to **N**oise **R**atio
SNRDR	**SNR D**egration **R**atio
SoC	**S**ystem-**o**n-**C**hip
SR	**S**lew **R**ate
SRM	**S**tructured **R**isk **M**inimization
STF	**S**ignal **T**ransfer **F**unction
SV	**S**upport **V**ectors
SVM	**S**upport **V**ector **M**achine
UBW	**U**nity-gain **B**and**W**idth
VC	**V**apnik **C**hervonenkis
VHDL	**V**HSIC **HDL**
VHDL-AMS	**VHDL**-**A**nalogue and **M**ixed-**S**ignal
VHSIC	**V**ery **H**igh **S**peed **I**ntegrated **C**ircuit
VLSI	**V**ery **L**arege **S**cale **I**ntegration

Acknowledgements

走过这三年多的学习，我要感谢很多人。没有你们我不可能完成这样一个长期而艰苦的过程。远在中国山东济宁的家人给我最大的支持，在我因困惑而心生逃遁的时候，我的父亲，母亲，哥哥，姐姐给我最无私的鼓励让我坚持下来。我的妻子邓春旭陪我度过了两年多的时光，给了我最贴心的支持，我们的生活在磨砺中不断前行，让我向一个更优秀的人进步。我的导师汤姆·J·凯日密斯基在学术上给了我很多的指导使我可以顺利的完成这个艰难的学位。还有给了我学习和生活上帮助的朋友们包括南安普顿大学的同学和南安普顿大学电子和计算机系电子系统组的同事们，虽然我没有一一列出他们的名字，但我发自内心的感激他们每一个人，希望他们都能健康快乐。

In the three years study, I have to express my thanks to many people. Without your help, it is impossible for me to accomplish this long and hard study. My family in Ji Ning have given me the greatest support. When I was stuck in the study and lost the strength to continue, my father, mother, brother and sister encouraged me to keep on going and gave me the bravery to overcome the difficulties. Their selfless support is my most previous wealth. My wife Deng Chun Xu stays with me in the U.K for over two years. Her love accompanies me all over the time and our life together becomes more and more harmonic. She is like a mirror and I can see myself in it so that I could be a better person. My supervisor Dr Tom J Kazmierski has provided me brilliant academic supervisions so that I can accomplish the task. I appreciate for his guidance on the research. Also all of my friends who have given me help on my studies and life including my schoolmates from the University of Southampton and my colleagues from the Electronics Design Group of the Electronics and Computer Science department of the University of Southampton. Although I am not listing out their names, I do want to express my thanks from my heart to every one of them and wish them health and happiness.

To My Family

Chapter 1

Introduction

1.1 Challenges in AMS design

In the design of modern electronic systems, AMS circuits are important parts in the interfacing between digital circuitries and the real world [1]. Although the main applications of AMS design methods are analogue signal processing and data conversion components, a complete system with both digital and AMS circuits may contain dozens of such components on a silicon die and are designed in highly scaled digital fabrication technologies [2]. This exposes the major design challenge: the increasing integration level in a large system-on-chip environment with advanced fabrication processes. Such designs, usually called system-on-chip (SoC), are widely applied in consumer electronics such as DVD systems [3] and mobile media applications [4].

A typical SoC is the cell phone shown in figure 1.1 [5]. Although, the primary IC components here are microprocessors and memory, analogue and RF blocks also play an important role. High level of integration in such a complex design can reduce the cost and promote high volume production. However, it is very difficult to achieve such a goal mainly due to the following two aspects. Firstly, analogue design methods need to be refined with the development of the fabrication technology. The current trend is to design analogue components using deep submicron digital fabrication technologies for high speed and high frequency applications. Secondly, the high integration level of analogue designs has to be achieved with low power supply voltages [6].

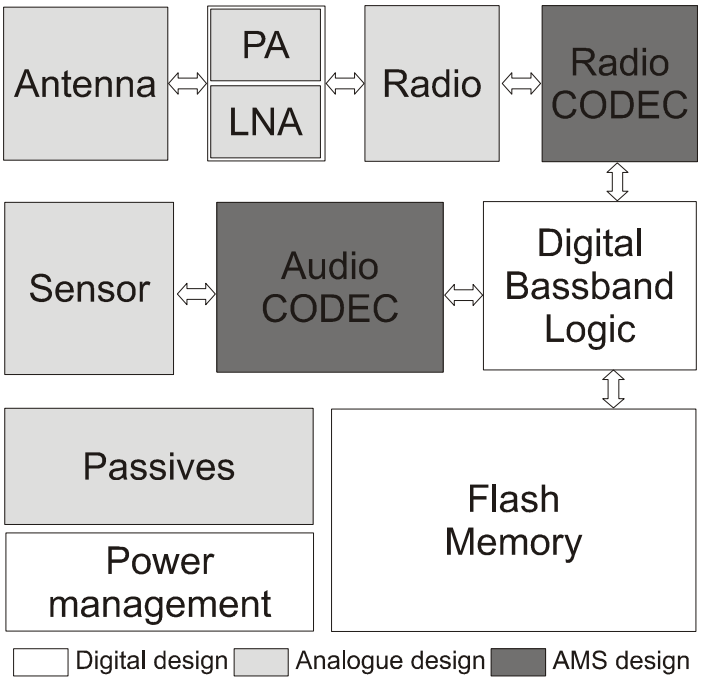


FIGURE 1.1: An illustrative AMS SoC example.

Major design tasks, such as verification, optimisation and synthesis of complex SoC systems need to be carried out at both system and circuit design levels in the VLSI design flow. One particular characteristic of analogue designs is that typically they need to be optimised using dozens of competing continuous-valued performance specifications. A successful optimisation is dependent on the circuit designer's ability to explore a range of nonlinear behaviours across abstraction levels in the VLSI design flow [7]. This means that, firstly, designers need to have adequate means to evaluate the behaviour of the designs and, secondly, they should be able to explore their designs thoroughly at different abstraction levels. In most stages of the VLSI design flow, various simulation approaches are available for design evaluation. In most cases, designs are presented in the form of a description language including classical SPICE netlists, extended digital hardware description languages (HDLs) such as VHDL-AMS [8], SystemC-A [9] as they can cover almost all the VLSI design stages. Simulators for these HDLs, either from industry or academia, are widely available. From the point of view of this research, it is important to stress that it is the lack of methods for sound design exploration that severely limits the designer's efficiency. This research is motivated by this. Specifically the main aim here is to investigate the possibility to develop methodologies for effective AMS design exploration.

1.1.1 AMS synthesis

The hierarchical approach to VLSI design using multiple abstraction levels of cells has become the established way for designers to control the explosion of complexity [10]. Using this well-known methodology, a top-down design procedure [11] is often adopted. It starts from conceptual formulation at the system design level and goes down to block function designs then to circuit cell designs until layout and fabrication. As the design is refined, specifications are transformed into the forms suitable to the corresponding design stage.

In every design stage, a manual iterative process can often be observed [12] as shown in figure 1.2. The design from the previous stage is passed to the current stage. It is then redesigned by considering the design specifications transformed from those at the previous stage and using the library components at the current stage. The main design loop is to use simulations as a means to evaluate and verify the performance of the design iteratively until it fulfils the design specifications. If not, the design will be returned to the previous stage for more refinement.

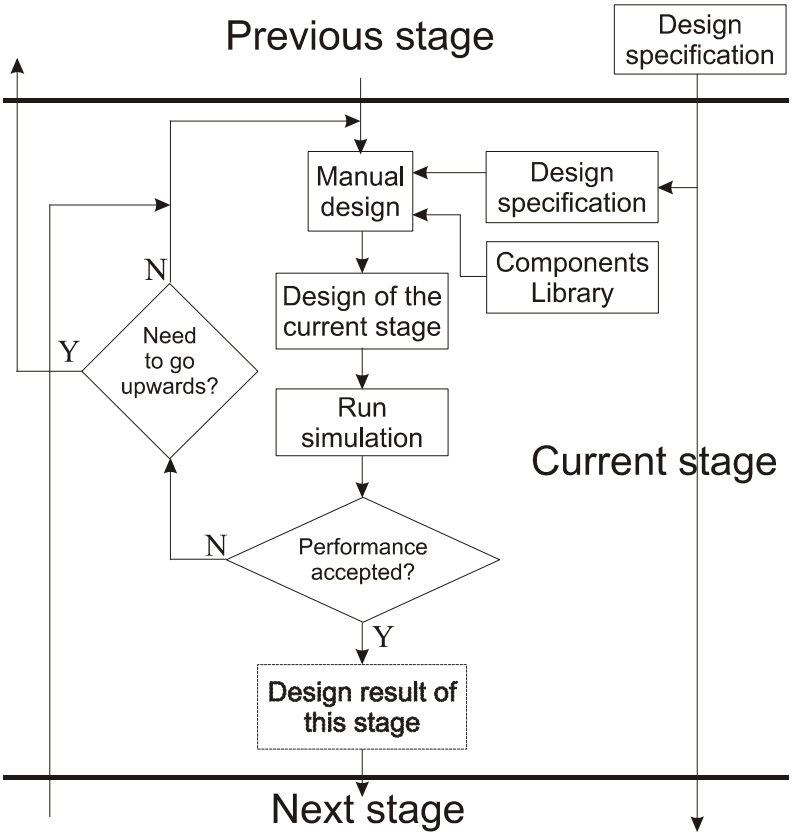


FIGURE 1.2: Manual design flow graph of the general VLSI design stage in detail.

As the design methodology is so well structured, it seems that AMS synthesis can be easily implemented. However, AMS synthesis is still in its infancy and falling far behind its purely digital counterpart [12]. Although digital designs have sufficient support by computer aided design (CAD) methodologies at all the design levels, AMS synthesis has only just moved from conceptual demonstrations to first-generation commercial offerings [13] and only capable of handling analogue cell designs with 10-100 devices [10]. This slow evolution of AMS synthesis essentially reflects the nature of analogue designs where numerous and competing design objectives exist at different design levels. One of the reasons that the AMS synthesis development lags behind is because there is no uniform HDL for synthesis at different design stages in the VLSI design flow. Most AMS HDLs including VHDL-AMS [8], Verilog-A [14], SystemC-A [9] are created for the purpose of modelling and simulation [15]. Therefore, when they are adopted in synthesis systems, a subset is selected to enable writing synthesisable HDL code. Although the new HDLs have attractive features for high-level simulation, up to now, they are not mature enough for the purpose of AMS synthesis.

More importantly, today's modelling and optimisation techniques are not powerful enough to replace designer's experience. At system and circuit design level, AMS designers often use the following six dimensional evaluation matrix for their synthesis methodologies [16]: prediction accuracy, application generality, capability on complexity, synthesis time, automation and preparatory effort needed. However, some of these principles are very subjective and difficult to be quantified. So many famous systems [17–19] have to emphasise some aspects of the evaluation principles while sacrificing some others. Numerous systems are dedicated to specific analogue designs [20, 21], and they use specialised optimisation algorithms [22] or simulators [16] at various design levels [23].

This research is not attempting to solve a general AMS synthesis problem. However, the performance modelling and optimisation methodology applicable to a wide class of AMS designs developed in this research represents a concrete step towards a full general AMS synthesis system. Such a methodology is an important underlying part of any general AMS synthesis system.

1.2 Motivation for this research

As stated above, the aim of this research is to develop a general performance modelling methodology which could be applied to an arbitrary synthesis system for a wide class of AMS designs. It is well known that simulation is the main method to evaluate the performance of analogue IC designs especially when circuits operate at high frequencies with ever shrinking transistor geometry [24]. However, applying simulation directly in analogue synthesis is not always efficient because the number of iterative executions in a synthesis process is large [25]. This is especially true for certain designs that need long simulation times at circuit level such as sigma-delta modulators [20].

Most of the current performance modelling techniques for AMS synthesis are applicable to only one type of circuit. Operational amplifier (OpAmp) and other analogue cells are usually modelled by symbolic expressions [19, 26] or posynomial functions [27]. These methods are essentially the extension of manual design experience. Although they can cover a category of design problems at circuit-level, the generality is restricted to particular problems. The current modelling techniques for analogue filters are another typical example. Transfer function analysis and synthesis is critical in the decomposition and implementation of complex analogue filter structures [28, 29]. Although some analogue-to-digital converter (ADC) designs can also be modelled by transfer functions [30], ADC synthesis methodologies differ significantly [31, 32] because of the lack of general performance modelling methods for different ADC structures, implementation techniques and application specifications. Most systems have to use specifically developed simulation techniques [33] and sacrifice the reusability and generality. Such systems are usually interactive tools which assist designers in design exploration as they provide specific design information in a dedicated way.

There have been attempts to develop general performance modelling techniques for AMS synthesis based on neural networks [17] and fuzzy logic [34]. In such systems, subjective human design expertise, either in a symbolic or a numerical form, can be extracted by the training process. However, neural networks and fuzzy logic have a number of drawbacks which are listed and discussed in detail in the next section. This research aims to investigate an alternative approach to extract design expertise from simulations based on SVMs.

1.2.1 Why support vector machines?

SVMs [35] were proposed originally in the context of machine learning for classification problems on large sets of data which have complex and unknown relationships between variables. The SVM approach can be used both for classification and regression estimation [35]. The technology has been successfully applied to pattern recognition [36], fault diagnosis [37], nonlinear control [38] and many other problems.

SVMs are easier to integrate into an automated design environment than corresponding fuzzy logic systems. The preparatory overhead in the application of fuzzy logic to synthesis is very high because special design experience is needed to convert numerical evaluations of the designs to fuzzy rules. The selection of membership functions is almost always dependent on AMS synthesis system designers' knowledge and specific applications [34, 39, 40]. This impedes a wide adoption of this technique and creates extra difficulties in automation. In contrast to fuzzy logic, SVM training parameters can be determined automatically by applying search algorithms without the designers' manual intervention.

The so called structured risk minimisation (SRM) [41] strategy is used in the SVM training to construct optimal performance models. In a nutshell, the SRM builds the optimum regression function by minimising the prediction error and maximising the distance between the nearest samples and the regression function. In contrast to this strategy, traditional neural network technology uses empirical risk minimisation (ERM) [41] that minimises the approximation error only. It is clear that the SRM strategy automatically solves the optimisation problem by balancing the competing objectives of model generality and accuracy using the well-studied quadratic programming. For ERM methods, this trade-off has to be carried out by dedicated algorithms as seen in some examples [17]. SVM models tend to be more compact than corresponding neural network models because optimal support-vector regression functions include only a subset of the whole training data set, namely the support vector set [35, 41, 42]. It has been reported that SVM based approaches are able to find global optima in cases where neural networks converge to a local one [43]. As some comparisons between the SVM and neural network [44] indicate that the two methods work equally well in terms of prediction accuracy, in summary, it can be conservatively concluded that SVMs perform at least as well as neural networks in terms of model accuracy. Also, SVM

methods seem to perform better in terms of training and prediction errors than simple linear least-square regression and posynomial models [45].

Another advantage of SVMs is that the model construction can be readily automated. As the SVM SRM strategy has implemented the trade-off between the generality and accuracy of the models, the training process would ensure to generate performance models that are very suitable for the design. In additional, the SVM trainer can be configured in an easy tuning way with only two training parameters to explore.

1.3 Research contributions

SVMs have already been suggested as an efficient means to model analogue performance restricted to the classical "good-bad" classification model construction [46, 47] and performance space regression [45]. In this research, SVM application has been extended to the automatic generation of a general performance optimisation system for AMS designs. Compared to most state-of-the-art applications of SVMs [45–47], which use manual and "try-it-out" methods for SVM training parameters determination that may lead to non-optimal results, the method presented in this thesis needs minimum human intervention thus is labour effective. A new, accurate model construction algorithm has been developed and implemented. The algorithm has been demonstrated to provide almost an order of magnitude speed up for SVM training parameter determination compared with the standard grid search approach recommended by SVM trainer's creaters [116]. In addition, the proposed performance modelling methodologies have been implemented in a way that makes the model construction system independent of the underlying optimisation system to achieve high generality and flexibility in the choice of an optimisation system. The proposed algorithms lend themselves easily to effective performance optimisation. As the SVM technique is applicable to any design modelling, the performance modelling methodologies developed in the course of this research are also applicable to a very wide range of problems. The purpose of developing different performance modelling methodologies is to explore the possibilities of applying the SVM technique to the area of AMS performance modelling.

The main contributions of this research can be outlined as follows:

1. **Linearly graded automated performance model construction using SVMs:**

   A linearly graded performance modelling approach has been developed to extend the classical 'good-bad' binary classification model. The idea behind the method is to partition a multi-dimensional design space into a few sub-spaces; all the sub-spaces are modelled separately. The partitioning is done by constructing boundaries between the sub-spaces using SVM classification method; each sub-space is modelled by the SVM regression method to provide performance predictions at design points in the sub-space. The partitioning divides the complex design problem into smaller sub-problems which are easier to be solved in terms of computational cost overhead. In addition, the model can give more insight into the design than the classical 'good-bad' method by providing a linearly graded evaluation result. This approach has been presented in two conference papers [48, 49].

2. **SVM regression based automated performance modelling methodologies:**

   Two approaches have been developed for SVM performance model construction using the SVM regression method:

   - **Automatic generation of SVM regression performance models for AMS and RF designs:** the approach uses SVM regression models on the whole design space to simplify the performance model construction process and eliminate classification errors introduced in the linearly graded approach. The approach is fully automatic with no need for manual intervention and lends itself naturally to a knowledge-based AMS performance optimisation system. This approach has been published and presented in a conference [50].
   - **Computational-cost aware SVM training parameter determination:** the computational cost analysis of the standard grid search training parameter determination algorithm, which is used in the previous two methods, has revealed that the computational cost of an accurate determination of SVM training parameters can be dramatically reduced. The training parameters have significant influence on the computational cost which can vary by up to three or four orders

of magnitude. Thus a very efficient gradient-based and heuristic algorithm has been developed for high speed training parameter determination. An improvement of up to 7 times in terms of the CPU time has been achieved which means days of training time can be saved in practical applications. The new algorithm has been applied in the SVM regression performance model construction system. Results have been summarised in a journal paper submitted to IET CDS Proceedings (see appendix A).

3. **Knowledge-based performance optimisation using SVM models:**

   A very efficient performance optimisation system has been developed. The performance models have the form of an automatically constructed knowledge database so that no simulations are required during the exploration of the design space. Two popular optimisation systems have been implemented in the practical case studies. One uses a dedicated genetic optimisation algorithm and the linearly graded SVM performance models, and the other utilises a standard pattern search technique with SVM regression performance models. Both can converge to an optimal design extremely quickly. In the analysis of the 2^{nd} order sigma delta modulator (SDM) case study, pattern search optimisations required only about two to five minutes of CPU time. The optimisation system has been presented at a conference [50] and an extended version has been described in the submission to the IET CDS Proceedings mentioned above.

4. **Verification of the proposed performance modelling and optimisation methodology using two practical case studies of complex AMS systems:**

   Two main case studies have been modelled and optimised to validate the proposed methodologies. This includes the specifications of the design and performance spaces, SVM model construction and performance optimisation experiments. The mixed-signal example is a 2^{nd} order SDM. It includes most of the imperfections typical for a switched-capacitor implementation: non-linear OpAmp DC gain, finite OpAmp DC gain, slew rate, clock jitter, OpAmp and switch thermal noise, OpAmp dynamic range etc. This example is representative for AMS designs with moderate complexity for which advanced design methodologies are highly required. The second example is a Colpitts RF bandpass filter. The main challenge in this case study is

that it is not suitable for a standard analogue filter transfer function synthesis method. It contains a number of nonlinearities and a complex on-chip spiral inductor with very high losses. The optimised designs have been verified by full simulation. Comparisons of optimisation results with standard manual design results are provided to demonstrate the effectiveness of the proposed methods. Using the proposed computational cost aware SVM training parameter determination algorithm, it has achieved almost an order of magnitude efficiency improvement with less than 5% prediction accuracy degradation.

In addition, the computational cost model of the SVM regression model construction using the standard grid search algorithm has been further confirmed by extra experiments using five public data sets.

1.4 Thesis structure

This thesis is composed of seven chapters. The following is a chapter-by-chapter explanation with each chapter's main points and contributions.

Chapter 2 is a literature review of the topics related to the research. It covers details of state-of-the-art AMS synthesis systems, modelling and performance optimisation techniques. Chapter 3 introduces the underlying technique used in this research, i.e. the support vector machines. Chapter 4 outlines the first contribution of this research: the linearly graded SVM performance modelling methodology. The automated model construction process is presented in detail including a heuristic grading algorithm. A simple case study has been included for conceptual demonstration. Chapter 5 has a twofold purpose. The first goal is to present the proposed methodology for the construction of SVM regression performance models for AMS designs. The second one is to illustrate the computational cost analysis using the standard grid-search algorithm. This analysis leads to an empirical SVM training process computational cost model and an effective SVM training parameter determination algorithm which has been shown to be up to 7 times faster than the classical grid-search method. Chapter 6 presents the proposed performance optimisation system and the two optimisation algorithms used in the underlying optimisation engine. Chapter 7 presents the case studies and experiments. The development of the two cases studies are illustrated in appendix B and C.

Finally, chapter 8 summarises the presented work. The main achievements are discussed and highlighted. A number of conclusions are drawn based on the contents of the previous chapters. Further work is also suggested as a basic guide for a future research on this subject.

Chapter 2

Literature review

Chapter 1 highlights the importance of performance modelling in AMS synthesis systems, its challenges and proposes a general methodology and an optimisation system. Before explaining the work, this chapter provides details of the background literature review as the following: section 2.1 presents a general introduction of AMS synthesis methodologies. It includes the review of the classical structure of AMS synthesis systems and specific synthesis techniques for analogue filters and ADCs. As an important part of an AMS synthesis system, the most popular performance modelling approaches and optimisation algorithms are explained in section 2.2. The HDLs used in the case studies are introduced in section 2.3. Finally, in section 2.4, a brief introduction of the design of SDM is presented.

2.1 AMS synthesis methodology

The design of VLSI system can not been successfully done without the help by CAD tools [51]. The purpose of using CAD tools is to implement a rational methodology to address the challenges of design complexity encountered in modern VLSI designs. One of the design methodologies is a hierarchical top-down strategy [12]. The whole VLSI design process is divided into several stages and each stage corresponds to a form of representation of the design with the specifications projected at that level. There are different opinions of dividing the VLSI design flow [52–54], however, in general, the following process, shown in figure 2.1, is acceptable to cover all the stages. It illustrates the top-down VLSI design

methodology using the hierarchical decomposition method in the space formed by the 'specification' and 'abstraction level'. Seven design stages present and each contains one design step and one verification step (fabrication is verified by testing). Verifications can result in a redesign loop back in the design process, so the design is refined at every step by this forward and backward flow until fulfils the design specifications. The documentation is independent but essential for every design stage to make the design traceable in this complex flow.

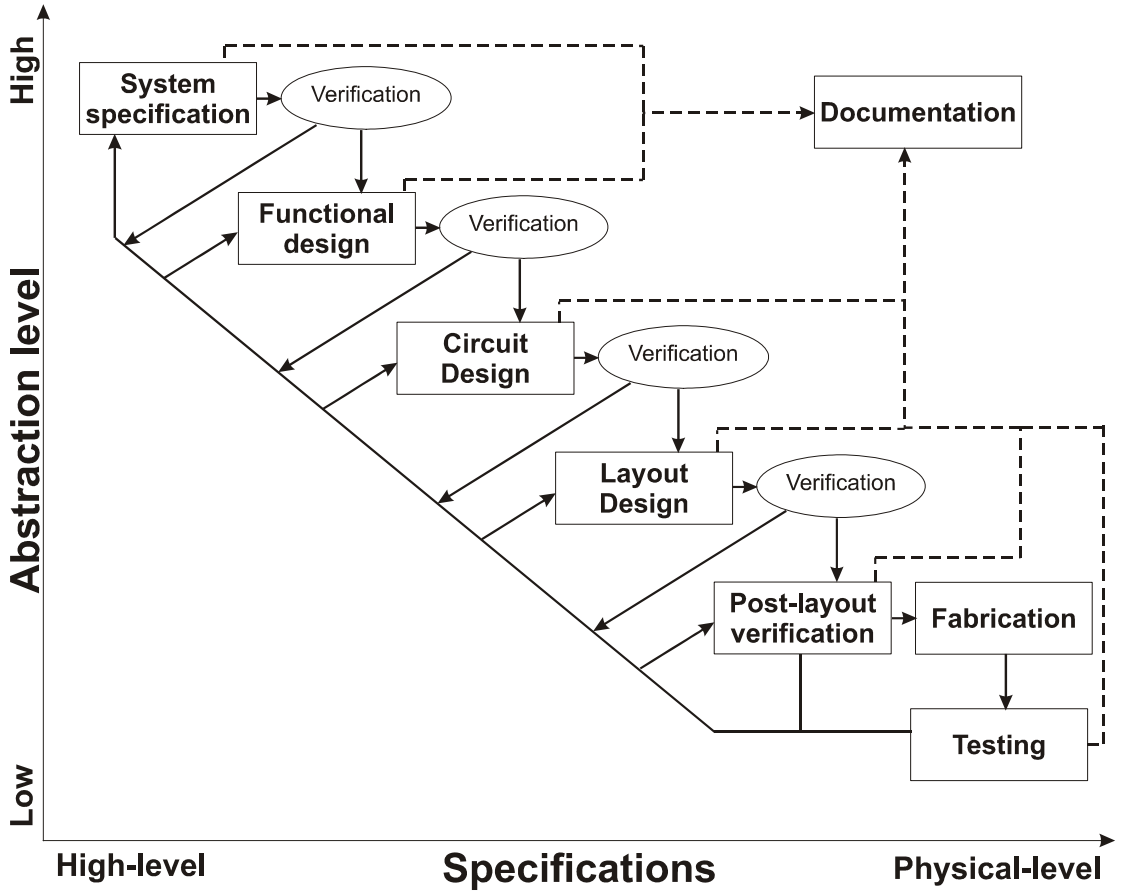


FIGURE 2.1: VLSI design flow composed of six design stages in the two-dimensional space formed by the abstraction level and the specification level.

Synthesis systems are designed to automatically accomplish this design flow to generate electronic circuits from high-level specifications [55]. They are roughly categorised as digital and analogue synthesis systems depending on the type of circuits to be synthesised. AMS synthesis is a mixed synthesis involving both analogue and digital parts in one design. The tasks include: choice of a structure at the block diagram level, analogue/digital (A/D) partition, determination of parameters and properties for analogue and digital blocks [56]. All of these can be separate research fields.

2.1.1 Overview of AMS synthesis

In general, AMS synthesis is still in its infancy and falling far behind its digital counterpart [7, 12]. This predicament is essentially because of the nature of analogue designs. Although analogue units are usually small with dozens of transistors [57], the design needs highly specialised expertise to work on numerous design objectives and trade-offs. This deeply affects the development of performance modelling techniques and synthesis. The first IEEE standard AMS modelling language - VHDL-AMS [8] was released more than ten years later than the first version of its digital counterpart VHDL [58]. Other major AMS HDLs such as Verilog-A and SystemC-A are also modelling and simulation oriented [15]. Therefore, when they are adopted for synthesis designs, some modifications will have to be applied to create synthesisable HDL code. There are two main approaches to do this task. The first approach is to create an intermediate language as an interpretation of the simulation-oriented HDL [59]. The new language is designed to be easily synthesised as a detailed model of design components' functionality and performance descriptions. The second approach is to map synthesisable syntax elements of AMS HDL languages to the components in a pre-defined library [60]. Both approaches have not yet achieved breakthrough developments.

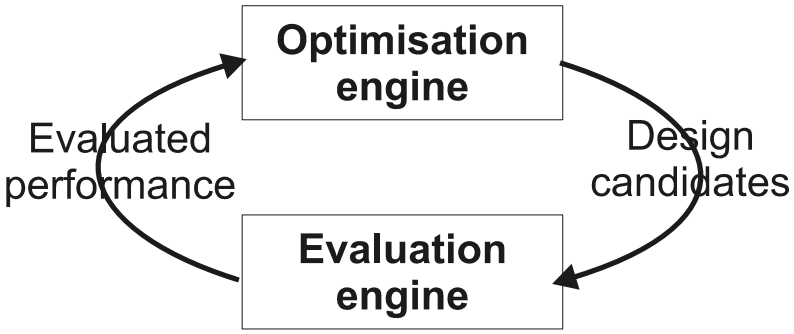


FIGURE 2.2: Classical flow chart of the interaction between the optimisation engine and the evaluation engine in an AMS synthesis system.

AMS synthesis systems usually contain two blocks as shown in figure 2.2. The optimisation engine block is to explore the design space for the optimal design with a set of specifications. The evaluation engine block is to estimate the performance figures of design sets. Optimisation engine modifies the values of the design parameters and sends them to the evaluation engine; then accepts the performance estimations to evaluate the design. The number of interactions between the optimisation and evaluation engine blocks is usually large because the design

space has a multi-dimensional feature. Both of these two blocks need to be highly efficient for quick AMS synthesis.

2.1.2 Knowledge-based vs. simulation-based AMS synthesis system

AMS synthesis systems can be categorised as knowledge-based and simulation-based systems. Systems of these two types have the same structure. The difference mainly lies in the techniques used to evaluate the performance of the designs, i.e. the evaluation engine in figure 2.2. Simulation-based systems use simulations directly at various design levels in the evaluation engine [25, 61]. This method works very well for analogue cell synthesis because the computational cost for simulations of small scale designs is usually negligible using modern workstations. However, it is not always efficient especially for certain designs that need long simulation time such as SDMs [20, 62]. For knowledge-based system, a knowledge database is utilised in the evaluation engine to provide estimations of the designs rather than simulations. Generation of the knowledge database is usually carried out separately using analytical [19, 26, 27] or supervised learning technologies [17].

TABLE 2.1: Comparison between the knowledge-based and simulation-based synthesis methodologies.

	Simulation-based	Knowledge-based
Prediction accuracy	High	Technique dependent
Application generality	High	High
capability on complexity	High	Low
Synthesis time	High	Low
Preparatory effort	Low	High
Reusability	Low	High

Comparison of these two methods using the six evaluation figures is summarised in table 2.1. According to the comparison, in a knowledge-based synthesis system, the modelling technique should be able to provide high prediction accuracy; at the same time, the preparatory effort should be minimised.

2.1.3 Analogue filter synthesis

As the communication sector in nowadays worldwide semiconductor market has exceeded that of the computing sector [63], analogue filter, which has very wide applications in telecommunication systems [64], gets increasingly important. Most analogue filter synthesis systems use standard transfer function analysis techniques [28, 29], which is usually utilised by manual designs. Roughly, the synthesis process is to decompose high order transfer functions into first order terms. The first order terms can be mapped to some standard circuit level cells so that the high order system can be constructed in a bottom-up way.

As shown in figure 2.3 [65], filter specifications are transformed to differential and algebraic equations (DAEs) or Laplace transfer function forms firstly. This is because it is easier to analyse the behaviour of the analogue design in the frequency domain. The synthesis strategy is to convert the system specifications to mathematical forms. Then the DAEs are transformed to suitable forms so that the structures in the equations can be mapped to pre-stored cell circuits. Every cell in the library is parameterised so that they possess flexibility to be redesigned. The cells are combined to form larger structures until the whole system is constructed. During this composition process, architecture and parameter optimisations are performed to ensure the design quality [28, 66].

This standard synthesis methodology does not have many variations for different designs. One of the major variations usually considered is the implementation technique [29]. The characteristics of the first order cell have to be studied and considered in great detail in the synthesis design [67, 68]. In summary, analogue filter synthesis is more mature than most of other analogue synthesis systems.

2.1.4 Analogue-to-digital converter synthesis

ADCs are found in almost every SoC designs as the interface between the digital and real world. According to application specifications (such as input sampling frequency and resolution), the architectures of ADCs can be flash [69], successive approximation [70], subranging [71] and sigma delta [72]. These architectures can be best applied for different sampling frequencies and resolution specifications as shown in figure 2.4. Unlike analogue filters that need only very few analogue building cells, ADCs need many basic components such as integrator, quantiser,

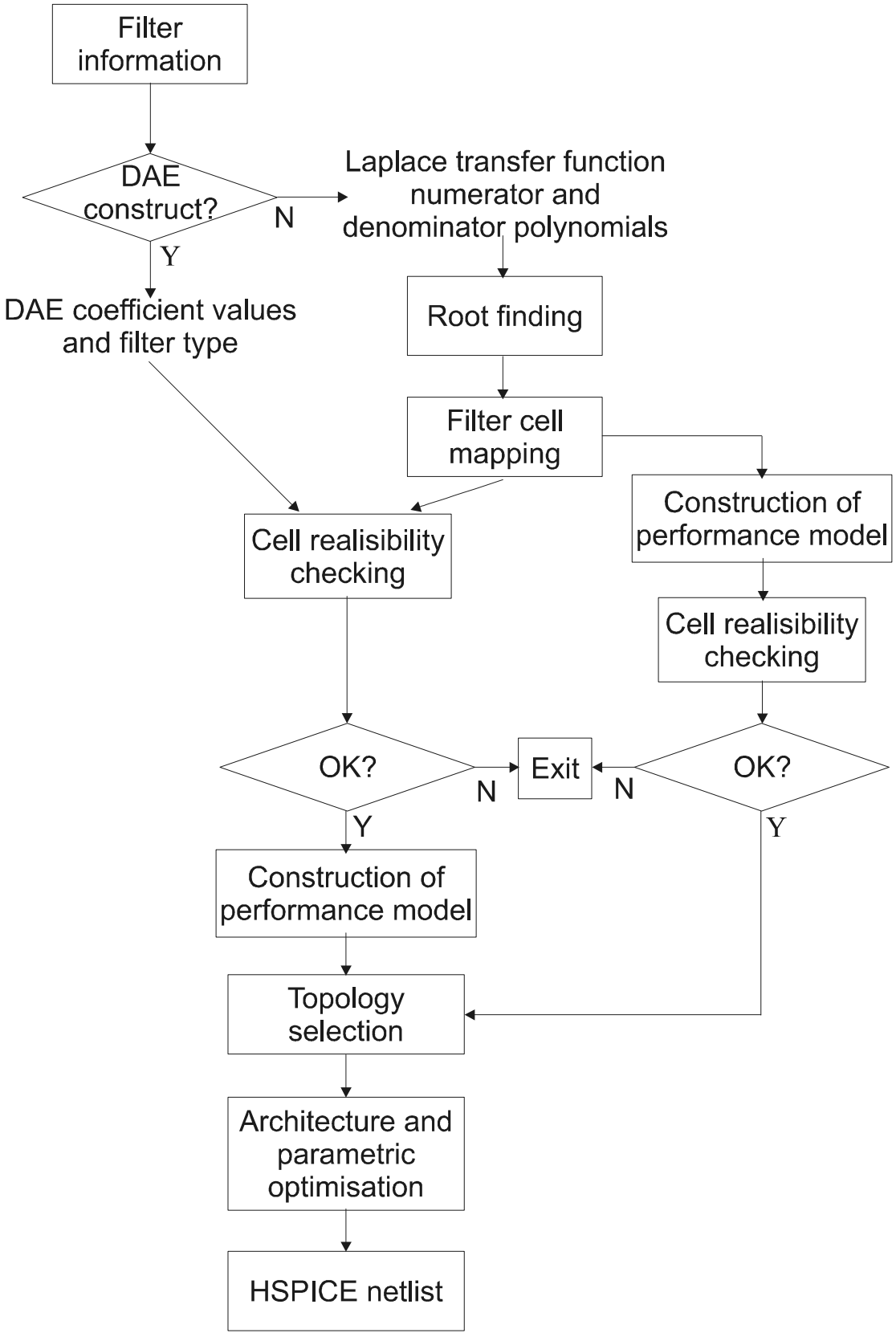


FIGURE 2.3: General analogue filter synthesis procedure [65].

each of which has to be constructed by smaller analogue cells. State-of-the-art ADC designers use specifications to select the architecture of the design roughly and consider performance degradation induced by mismatch. The whole design is very customised and needs expertise badly.

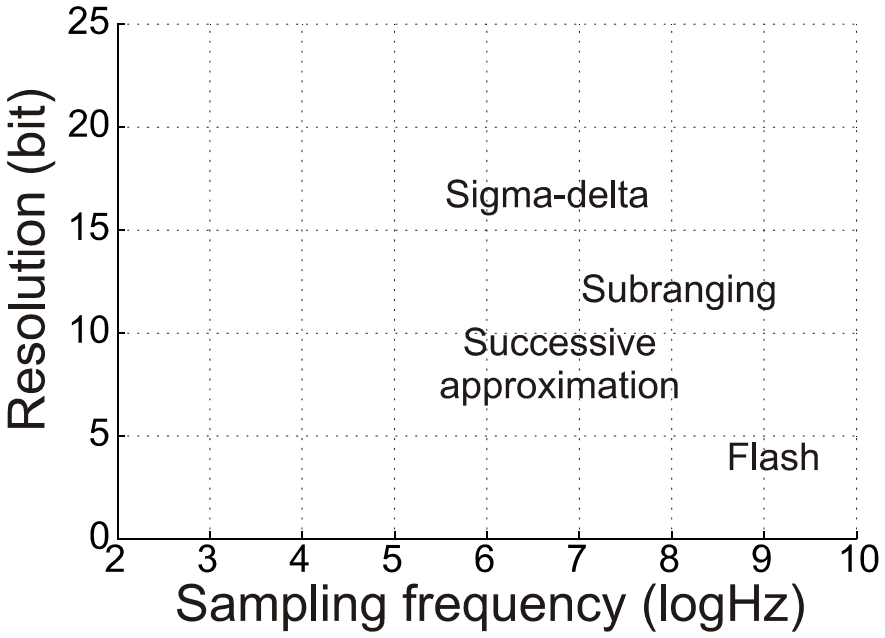


FIGURE 2.4: Typical sampling-frequency vs. resolution range for different ADC architectures.

Therefore, ADC synthesis is very specialised. Many systems are designed for one type of architecture only [73, 74]. As the synthesis is an iterative process and simulations are expensive at circuit level for full ADC designs, synthesis environments commonly tend to explore the design space at behavioural level with the considerations of the circuit level imperfections. Another method is to develop high speed simulators so that simulation-based system can be developed [75].

2.2 Performance modelling and optimisation in AMS synthesis

2.2.1 Analogue performance modelling

Simulation is essential to assess the performance of analogue IC designs especially with the ever smaller transistors used for high frequency applications. Both circuit

level and behavioural level simulators are applied to cooperate with the AMS VLSI design flow [76]. In simulation-based AMS synthesis systems, circuit level simulation is mainly used for cell synthesis because for large scale designs, simulation cost is so expensive that full circuit simulation is not practical [25]. Behavioural level simulation is thus developed [74] and the development on AMS HDLs is rapid. As introduced in section 2.1.2, simulation-based synthesis methodology has some disadvantages compared to the knowledge-based methodology.

Even for the systems using behavioural level simulation, performance modelling techniques are very critical to enhance the design efficiency [77]. Especially in knowledge-based synthesis systems, performance modelling is the key technique to extract knowledge from simulation results thus is an underlying part of such AMS synthesis systems. In recent years, the evolution of analogue performance modelling has been through its initial stage [78, 79] and is progressing swiftly. There have been many advanced techniques applied for this task. Some classical and important methods will be introduced in this section.

2.2.1.1 Symbolic analysis method

One of the most extensively studied performance modelling techniques is the symbolic analysis approach [80]. Symbolic expressions of the circuit parameters such as AC characteristics can be derived for analogue circuits. With symbolic analysis method, analogue circuits are treated as nodal networks connected by electrical elements. The CAD tools of this kind derive transfer functions and other relationships between the design and performance parameters for the network as symbolic forms in terms of complex Laplace variable s or z-transform variable z, as given by:

$$H(x) = \frac{N(x)}{D(x)} = \frac{\sum_i x^i \cdot a_i(p_1, \cdots, p_m)}{\sum_i x^i \cdot b_i(p_1, \cdots, p_m)} \tag{2.1}$$

where x represents s or z, a_i and b_i are symbolic polynomial functions of the circuit elements p. The circuit elements are represented by symbols instead of numerical values. As an example, ISAAC [26, 81] is introduced for the purpose of understanding the symbolic analysis process.

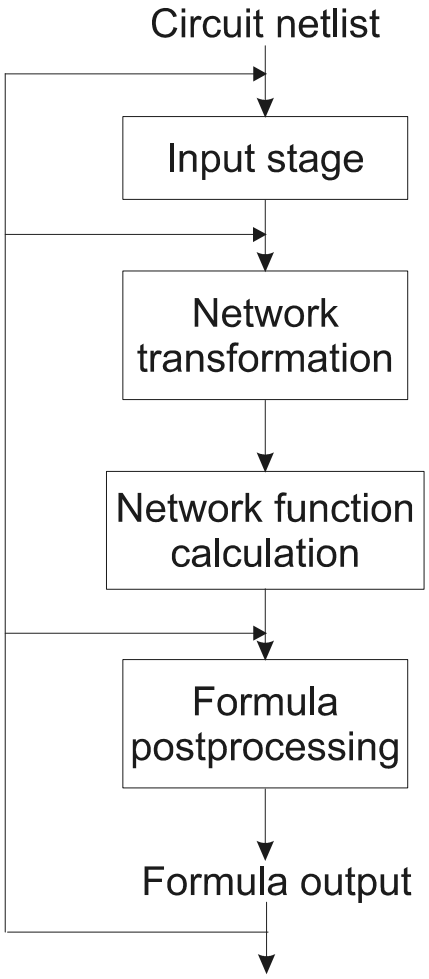


FIGURE 2.5: Symbolic analysis process in the ISAAC system.

The input of the system is a SPICE netlist. At the input stage, the circuit netlist is firstly read in then expanded if any sub-circuit is included. All the sub-circuits in the netlist are replaced by their complete descriptions. Still in the stage, all the recognisable components are replaced by the corresponding small-signal models. After verifying the connections, the netlist is transformed to a network form and the symbolic description of the network is simplified for the first time. For instance, parallel elements, like the elements of current mirror pairs, are represented by the same set of symbols because their small-signal models are exchangeable. After the network transformation stage, equations are set up. The network function is then derived by solving the equations using a dedicated equation deriving routine. Results of this stage are usually very complex equations, which are usually impossible for human to understand. Heuristic programs work on the complex resultant equations and simplify them further with a user-defined error tolerance in

the formula post-processing stage. The formula output may need to be re-derived if they cannot satisfy the specifications. The redesign can happen at a stage in the middle of the process as shown in the figure.

This approach gets very popular since it was reported in later 80s [26] because it can provide more insight of the behaviour of the designs than numerical simulations [26]. Great attention was attracted during 90s. With the increment of calculation power from personal computers and workstations, the method has achieved significant improvements to deal with lumped, linear or linearised and weakly nonlinear systems, and is still improving [12]. However, the methodology encounters great difficulty in recently years. Firstly, the application field is confined to circuit-level designs as at system design level, some performance parameters are difficult to be expressed explicitly by circuit-level design parameters and need to be calculated with simulation results using complex algorithms such as fast Fourier transform (FFT). Secondly and more importantly, the methodology shows limited capability in cooperating with the fast developing fabrication technologies. To handle large scale circuits, symbolic analysis technique needs two main capabilities: hierarchical analysis and good approximation capability [82]. However, as device scaling has a great impact on the transistor modelling for analogue designs [83] such that approximations of the symbolic expressions become more and more difficult.

2.2.1.2 Neural network method

Another very important approach to capture the design's behaviour is to use automatically constructed neural network models for the linear and non-linear relationships that exist between design and performance parameters [17, 84, 85] of analogue designs. This technique is attractive because a neural network with sufficient size can provide estimations at arbitrary precision level given a finite discrete set of training data [86].

In a fully connected neural network, as shown in figure 2.6, the output is related to the inputs by the hidden layer neurons, each of which is related to all the inputs with different weights.

Typically, neural networks are used to construct the database in a knowledge-based analogue synthesis system. In the application, the outputs of the neural networks represent the performance parameters of the analogue system and the

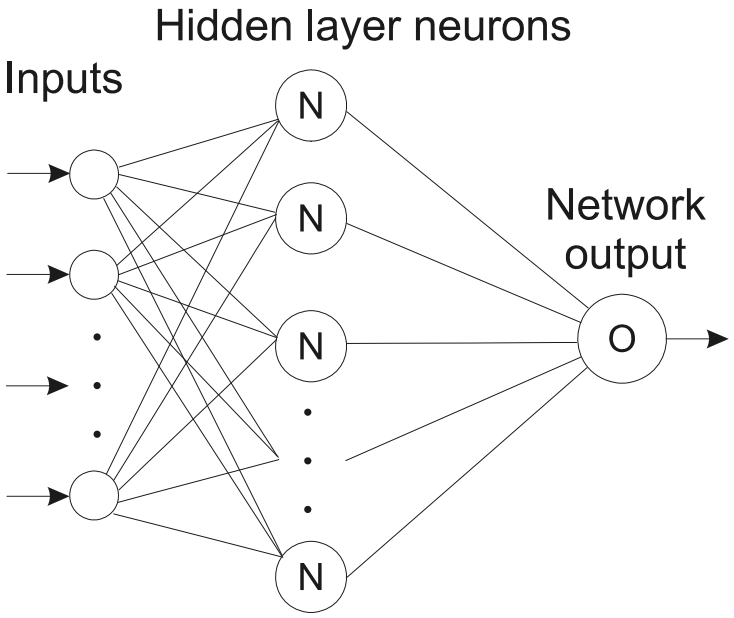


FIGURE 2.6: Classical two layer multi-input single-output feed forward neural network structure.

design parameters are used as the inputs. Several neural networks are constructed for each of the performance parameters so that the design parameters are connected to them by weighted and fully connected networks. The collection of all the networks forms the underlying knowledge database in the design automation system.

The neural networks' training process can be automated. The goal of the training process is to obtain the optimal neural networks in terms of training error and model complexity. The training process, also is called model construction process, is likely to be similar to that shown in figure 2.7. The process starts from a data generation phase. There are two data sets needed for neural network construction. One is the training data set and the other is the verification data set or called testing data set. Depending on the design level and accuracy requirements that the data can be generated from behavioural level or SPICE level simulations. The testing data set should not have any overlap to the training data set to ensure the effectiveness of the verification.

The quality of the models are characterised by two features: generality and accuracy. Generality is used to evaluate how well the model can still perform when it is applied to unknown design sets; accuracy reflects the quality of the model on predicting the outputs and it is measured by the error between the estimation and the real output. Generality and accuracy are a pair of tradeoff. Higher accuracy in the

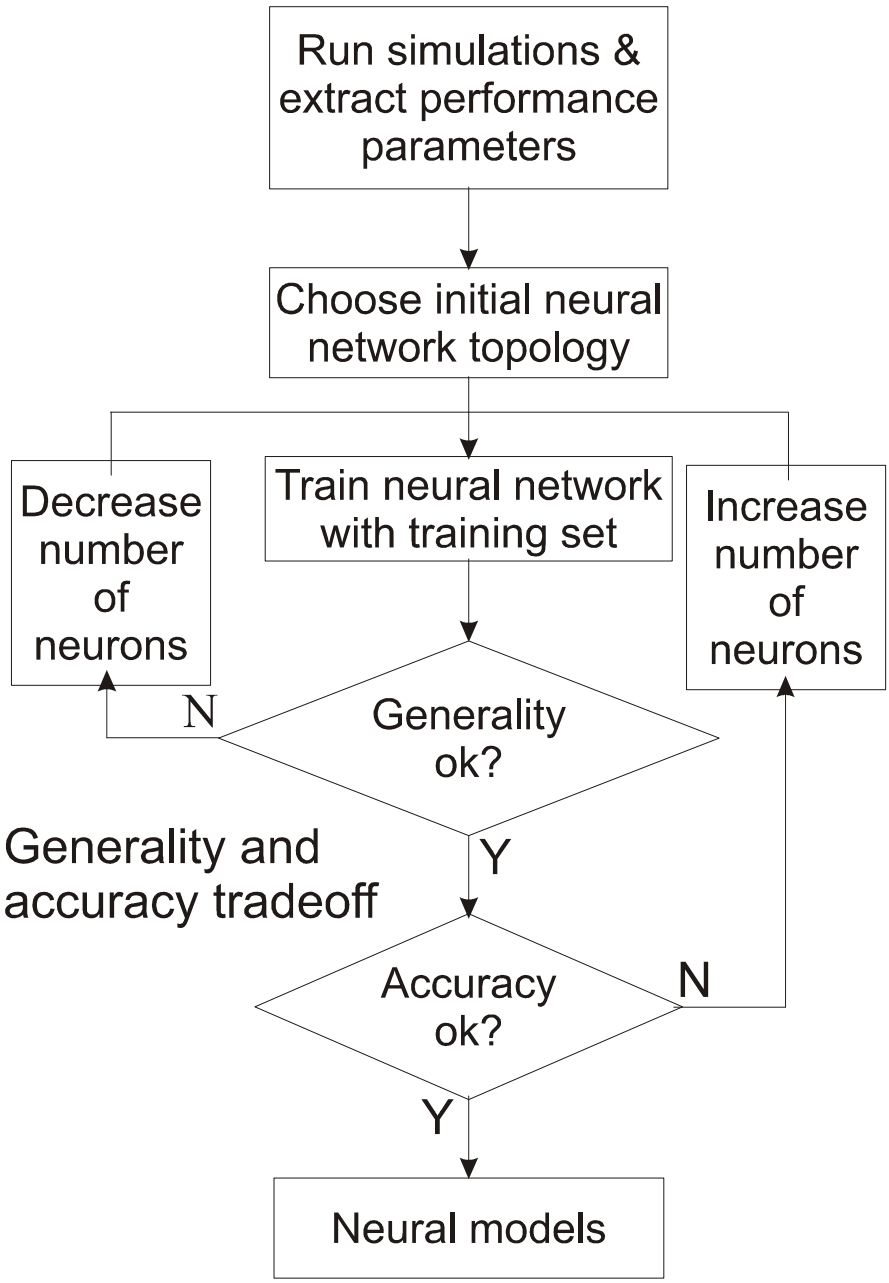


FIGURE 2.7: Model construction process using neural network technique.

training process can be achieved by sacrificing the generality. During the training process, the number of neurons is the tradeoff parameter: a smaller number of neurons can have better generality while a larger number, better accuracy. The two loops on conditions of 'generality' and 'accuracy' in figure 2.7 show a sample process that can optimise neural network models. A well verified neural networks model features the characteristics of adequate neurons with suitable generality for

the training set. These models are essentially the coefficients on every neuron connections in the neural networks. After model construction, the application simply feeds unknown data to the models and collects the predictions.

The advantages of the technique include that neural networks are very powerful in modelling non-linear relationships; the method is flexible and general for vast designs and different design stages; the knowledge modelling process can be standard and easy for automation.

2.2.1.3 Other methods

- **State space method** State space method is widely used in modern control theories and applications [87]. Because of the generality of the method, state space method has also been applied in analogue synthesis designs [88, 89]. The state space descriptions of AMS systems provide dynamic representations of the transfer functions therefore the information about the controllability, observability and stability features of the system. The well-known Laplace transform and z-transform can be used to convert the transfer function between the time domain and frequency domain easily. For time domain implementation, optional time domain components for structural implementations are limited to the functional blocks like adders, amplifiers, integrators and delay units, it is usually enough for analogue systems such as some ADCs or filters to be synthesised.

- **Fuzzy logic method** Fuzzy logic technology has been applied to deal with "uncertain" information and provides an effective means of capturing the approximate and inexact nature of human reasoning [90]. The fuzzy inference process derives conclusions from a set of fuzzy rules in an "IF-THEN" format. To work with the numerical input and output signals, the system needs to convert inputs to the corresponding fuzzy levels that the fuzzy inference system can understand. At the same time, fuzzy inference results are converted back to numerical values at the output. Application of the models in AMS synthesis is very similar as in the neural network. Compared to the neural network method, the disadvantage is that the preparatory overhead of this method is high because expertise is needed to select the member function and define the input and output conversions.

- **Signal flow graph method** Signal flow graphs (SFGs) are functional notations used traditionally in control theories to define system structures [91]. The method is widely used in the design of linear systems. In an SFG, system functionality is represented by the flow of the signal processing. The system structures can be derived from specifications explicitly but none of the features of the signals and the implementation details of the signal processing blocks will be indicated. Thus the technology is suitable for structural exploration of the design at the system level. A recent development for analogue synthesis [59] using the SFG method extends the concept of simple SFG to "characterised" SFG with performance attributes and requirements integrated into the individual blocks. In this way, it is possible to explore the architectures of analogue designs with the considerations of the performance. Also, it is very advantageous that SFGs for analogue designs can be easily described by HDLs such as VHLD-AMS. However, developing a highly accurate SFG model for non-linear analogue and RF design is very labour intensive and needs expertise a lot. For circuit level designs with sub-micron technologies, the situation will get even worse. This indicates the reason why applications of the method are limited at system level.

2.2.2 Optimisation algorithms

Analogue designs can have a large number of solutions with different trade-offs. If the performances of the analogue designs are evaluated by a cost function that rates the solutions according to their performance parameters, the design problem can be considered as a combinatorial optimisation problem. It is commonly accepted that exhaustive search of the analogue design space is not a practical method to obtain the optimal solution especially when the design space has a high dimensional feature. The purpose of using optimisation algorithms is to search the design space effectively. Some optimisation algorithms are introduced in this section.

2.2.2.1 Cost function

The analogue optimisation problem is considered as a constrained optimisation as both of the design and performance parameters are bounded by inequality constraints that must be met [18]. At this point, it can be accommodated in the

following standard constrained nonlinear programming problem:

$$minimise: \ C(x)$$

subject to:

$$\begin{aligned} x_1 &\in [V_{1-low}, V_{1-high}] \\ x_2 &\in [V_{2-low}, V_{2-high}] \\ &\cdots \\ x_n &\in [V_{n-low}, V_{n-high}] \end{aligned}$$

where $C(x)$ is the cost function to be optimised with design parameter vector x, x_i represents the i^{th} design parameter, V_{i-low} and V_{i-high} are the lower and upper constraints of the i^{th} design parameter.

Cost function is a special function used as the optimisation objective in analogue synthesis systems. It represents an overall evaluation of the whole design including all the performance parameters. So it is usually constructed as a combination of all the performance parameters. The weights in the equation balance the competing performance parameters. Generally, there are two forms of cost functions. The first type is a weighted linear summation form:

$$C(x) = \sum_{i}^{n} w_i R_i(x) \tag{2.2}$$

where $w_i, i \in [1 \cdots n]$ are the weight coefficients for all the performance parameters and R_i are performance parameters. This form can be found in many systems [16, 17, 92].

The second form is a weighted scalar error function. This form can be one of the two kinds as the following:

$$C(x) = \sum_{i}^{n} w_i |R_i(x) - R_i^{'}|^2 \tag{2.3}$$

or

$$C(x) = \sum_{i}^{n} w_i |\frac{R_i^{'}}{R_i(x)}| \tag{2.4}$$

where $R_i^{'}$ is the designer-specified objective value of i^{th} the performance parameter. Both of these two kinds of the cost functions are suitable for optimisations with explicit specifications [39, 59].

The two forms of cost functions are capable of exploring design spaces of analogue systems. The first form is more suitable for the cases where performance parameters have open specifications such as "the design area is the smaller the better". The second form can be used to find solutions that have the smallest deviations from the specified performance targets. In practice, it is possible to combine the two forms in one cost function if the weights are assigned properly. In this way, the constraints of the performance parameters of a design to be optimised can be openly or exactly specified.

2.2.2.2 Genetic algorithm

As its name suggests, the genetic algorithm (GA) attempts to develop solutions for the optimisation problems by mimicking the process of biological evolution. GA has been shown to have promising performance in a wide variety of problems when very large difficulties arise and there is no obvious way to calculate solutions [93]. The general flow chart of a GA is shown in figure 2.8. The concepts of fitness, crossover and mutation are borrowed from the biological terminology.

Initially, a number of individuals are generated with random gene values and they form the first generation population. The population scale will remain the same until the optimisation process finishes. This candidate solution set needs to be adequate and usually in a range from 50 to 1000. The individuals are featured by their chromosomes which are coded characteristics of the individuals. Fitness is an indicator calculated for each of the individuals to measure the quality of their chromosome. From the initial population, the first generation parents are selected according to their fitness. There can be variant algorithms to do the selection. One of the methods is to pick up the ones with better performance than others by ranking their fitness. After selection, if the crossover operation, which

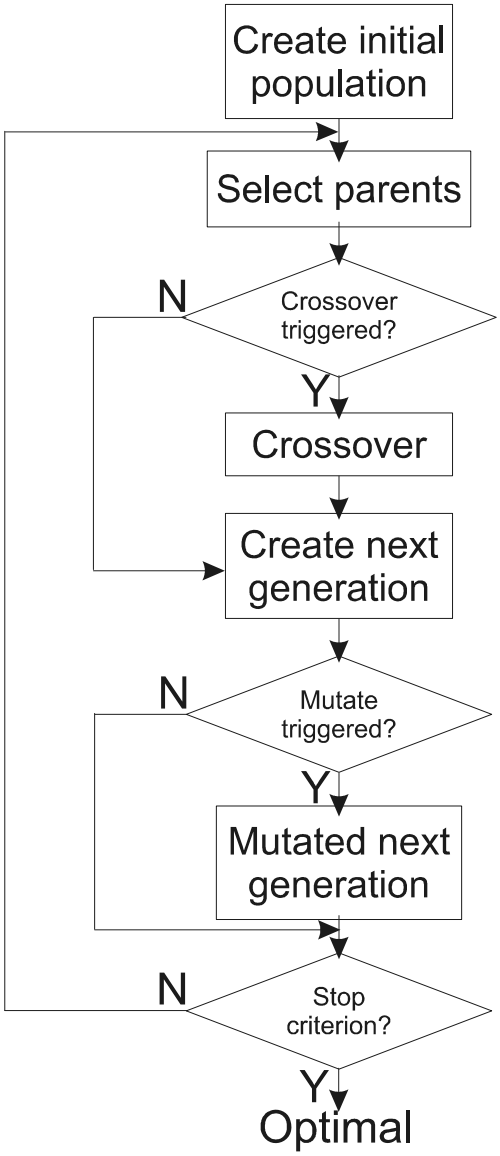


FIGURE 2.8: Flow chart of the canonical genetic algorithm.

happens with a fixed probability, is triggered, some parts of the chromosomes of the selected parents are exchanged and their offspring is produced. When the crossover operation is carried out on the selected parents, the new generation is formed. For each of the individuals in the new generation, the genes on their chromosomes have fixed probability to mutate on random positions. This evolution process finishes when the generation number exceeds the specified number. The optimal solution is then selected from the last generation with highest fitness.

GA has been applied in analogue synthesis systems to search for optimal solutions

[17, 94] in design spaces. It is commonly used in AMS synthesisers that the performance parameters of the designs are combined in one cost function, which is the objective function in the GA controlled synthesis process. The formation of the cost function can be simply a linear combination or some specifically designed type like a fuzzy membership approach [17]. GA is applicable for general purpose systems and is effective for the problems with no obvious formal analytical solutions. However, there are reported discussions [94] showing that for analogue circuit sizing problem, some specified numerical optimisations can be more effective than the general purpose GA. Even though, GA still can be a good option as a powerful optimisation engine in analogue synthesis systems.

2.2.2.3 Pattern search algorithm

Pattern search algorithms belong to the family of direct search methods [95]. They can be applied to a number of optimisation problems that are not well suited for standard optimisation algorithms [96] including problems in which the objective function is discontinuous [95], non-differentiable or highly nonlinear [97].

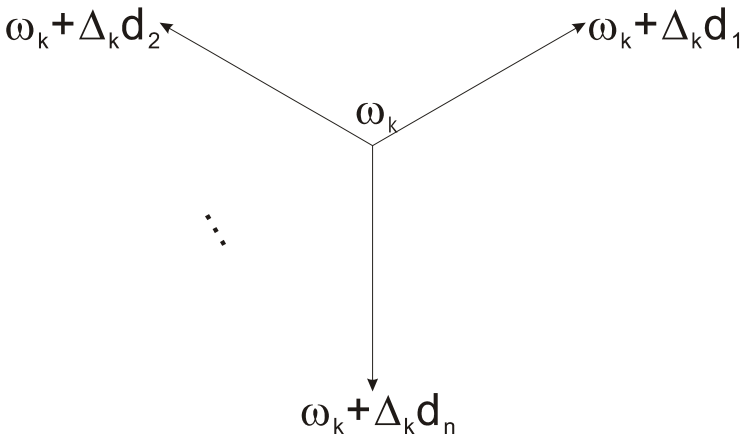


FIGURE 2.9: Mesh exploration of pattern search optimisation.

As shown in figure 2.9, pattern search explores a set of points, called a mesh, around the central point ω_k, Δ_k is the distance on direction of $d_i, i \in 1, \cdots, n$. All the directions form a direction vector D. The mesh points are evaluated, and their objective values are compared to the one of the central point in order to select the next central point. If an improvement is found among the points, the iteration is declared successful and the mesh size is retained or coarsened; otherwise, the mesh size is refined and a new set of points will be constructed. The process terminates

when either the mesh size is smaller than the pre-defined mesh tolerance or the performance of the system has better performance than the specifications.

A very important issue is to select the direction set. Research [98] has proved that if the number of directions n is within the following range:

$$m + 1 \leq n \leq 2m \tag{2.5}$$

where $m + 1$ and $2m$ are the lower and upper limits for the number of directions respectively, an effective exploration of a m-dimensional design space $\Re^m$ can be achieved. The significance of the conclusion is that given a design space, there are a number of ways to choose a direction set with different number and weights of the directions.

Since the pattern search approach was first introduced in the 1960s, the algorithm could not get enough popularity because of the lack of convergence verification. However, in 1997, the algorithm was analyzed and its convergence proved by Torzcon [99] for derivative-free unconstrained optimisation problems. Then the proof has been extended to handle constrained optimisation problems [100], including problems with a finite number of linear constraints [101]. This justifies that the algorithm is applicable for AMS design optimisations.

2.3 HDLs for AMS modelling and simulation

2.3.1 Modelling capabilities of HDLs

HDLs are designed for various simulation and design problems. They are applicable to different abstraction levels in the VLSI design flow and various design domains such as digital and analogue. Figure 2.10 shows the abstraction levels and designs covered by some of the main HDLs. The abstraction levels on the x-axis start from analogue systems that cover all analogue designs then change to digital designs from low gate level to high algorithm level. Most of the well-known HDLs such as VHDL, Verilog and SystemC, are for digital designs while some are for analogue only like SPICE. There is a trend to extend standard digital HDLs by adding new language syntax elements. This is mainly to cover AMS design issues. The new languages include VHDL-AMS, Verilog-AMS, Verilog-A and SystemC-A

for complex AMS system simulation and modelling. HDLs with AMS extensions are mainly developed for the purpose of filling the gap between the digital domain and the analogue domain. Among them, VHDL-AMS has been standardised by the IEEE; SystemC-A has been applied to complex simulation and modelling problems [9]; others are still under development.

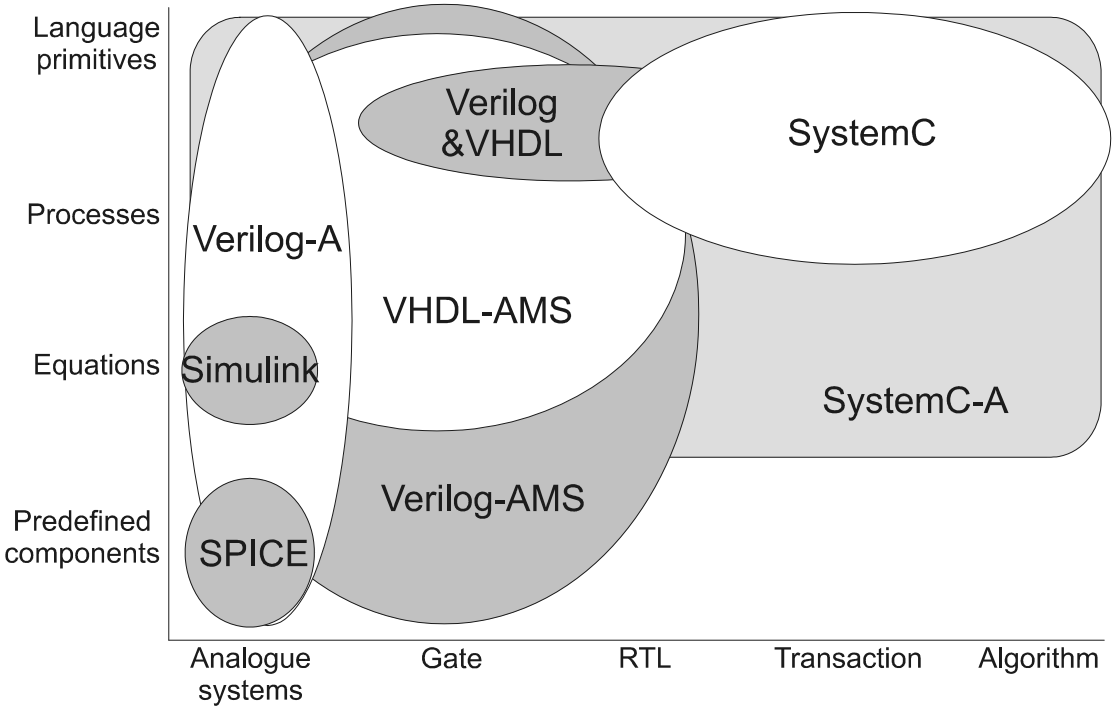


FIGURE 2.10: Application fields covered by different HDLs [9].The x-axis is the abstraction levels and the y-axis is the design representations.

In figure 2.10, it clearly shows that different HDLs are applicable for various design stages and representations. For AMS synthesis applications, unfortunately none of the HDLs can be synthesisable without any dedicated modifications at the moment.

2.3.2 VHDL-AMS

The standard IEEE 1076.1 language, also informally known as VHDL-Analogue and Mixed-Signal (VHDL-AMS) extension, is based on the IEEE std 1076-1993 (VHDL) language. The aim of the extension is to provide capabilities for modelling and simulation of AMS designs [8]. IEEE std 1076.1 1999 contains not only the extensions and modifications of IEEE std 1076 1993 but is a superset of VHDL, which is widely used for digital system simulation and synthesis. The language

supports modelling at various abstraction levels of electrical and nonelectrical energy domains [102]. The systems to be modelled are lumped systems that can be described by ordinary DAEs in the following form [103]:

$$F(x, dx/dt, t) = 0; \tag{2.6}$$

where F is a vector of expressions, x is a vector of unknowns, dx/dt is the derivatives of the unknowns with respect to time t. VHDL-AMS defines the notations for DAEs without indicating any method to solve them. This provides maximum flexibility to simulator development.

VHDL-AMS models As in VHDL, a VHDL-AMS model consists of an entity with one or more architectures in one design. The interfaces and generic definitions are in the entity, which define the input/output (I/O) characteristics of the design. The behaviour of the system is defined in the architecture. At behavioural level, event-driven behaviour is described by concurrent statements or sequential processes while continuous behaviour by simultaneous statements. At structural level, different configurations can be implemented for one entity to describe register transistor level (RTL) netlists. The language supports hierarchical design methodology by providing a series of language elements for components' interconnection.

VHDL-AMS simulation cycle The time domain simulation cycle using VHDL-AMS is shown in figure 2.11 [102]. The simulation cycle of VHDL-AMS is a process that advances on the time axis with a nested delta-cycle iteration. It is based on event-driven simulation mechanism. The simulator maintains an event list recording the new values assigned to the signals and the corresponding time points. At one time point T'_n, the process executes all the related signals and processes triggered by active events. If more signals or processes are triggered by the executed events, the current time will not move on, i.e. T_n will not be updated. The simulation will add delta cycles to propagate the signals to the components that they affect. After all the delta cycles are processed, the simulation process will move on to the next time point that an event happens. The process terminates when the simulation stop time is reached.

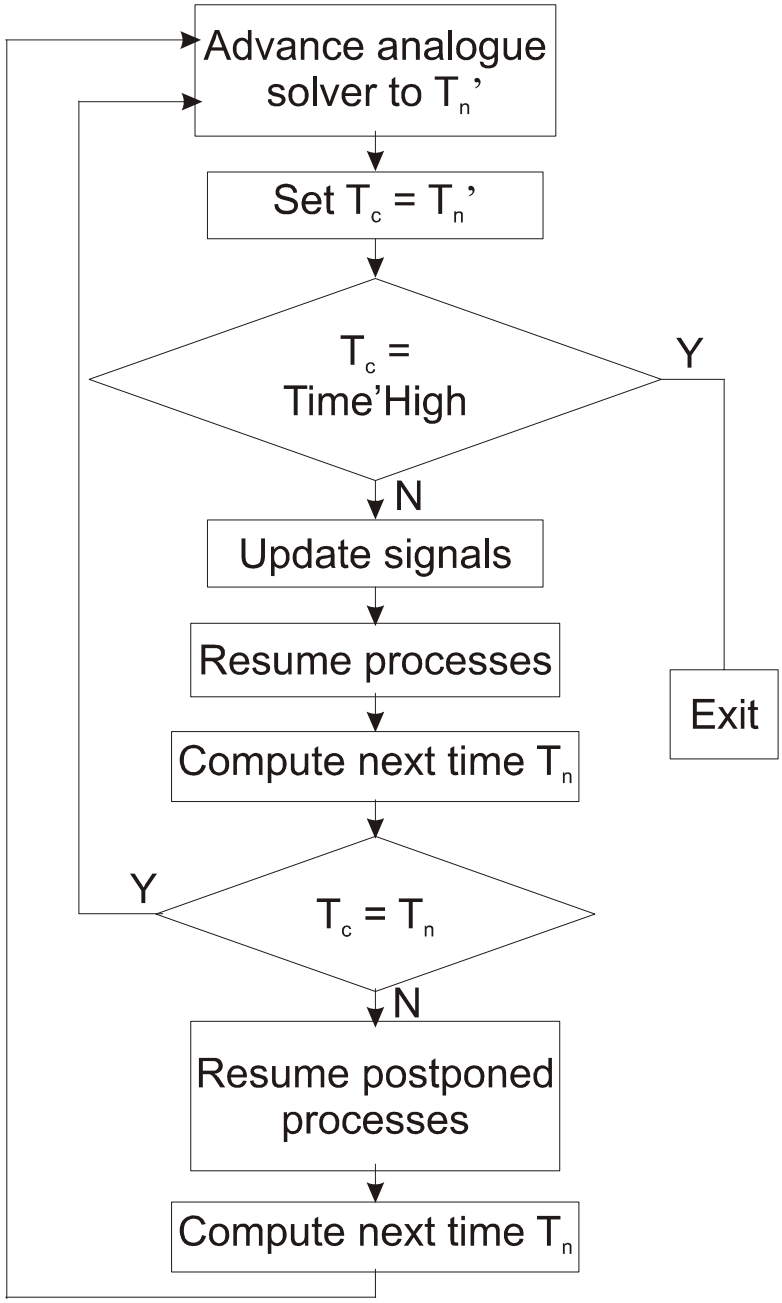


FIGURE 2.11: Time domain simulation cycle of VHDL-AMS.

Applications Although VHDL-AMS has been standardised for only few years, it has been widely used in AMS simulations [15] and synthesis systems [56, 59, 104]. Simulation problems covered by the language spread from electronics and electrical to non-electrical problems. However, as it has been introduced previously, because of the lack of mature simulators, the synthesis tools based on VHDL-AMS are still in their infancy. A good example of using VHDL-AMS is its application as the

main simulation language [15] to multi-discipline complex problems related with aircraft design.

2.3.3 SystemC

SystemC is a C++ library and a methodology that can be used to effectively create a cycle-accurate model of software algorithms, hardware architectures and interfaces of SoC and system-level designs [105]. Applications of SystemC are mainly digital designs. Nevertheless, the extension of the language for analogue designs is under development [9] so that complex systems involving a number of hardware and software components can be handled.

SystemC language elements The basic building block in SystemC is a module. Designers can facilitate the hierarchical decomposition of complex systems using simple modules to reduce design complexity. The definition of a module includes its interfaces and implementations of member functions. Processes are the basic units that define the execution of functions in a design. The execution order of the processes changes the sequential feature of C++ and makes SystemC suitable for concurrent programming. Every process maintains a sensitivity list. When variables in the sensitivity list are modified, the corresponding process will be triggered. SystemC has very flexible techniques to connect different modules. There are defined ports and signals for direct connections. Channels can be created for specific connections with specially defined behaviour. Another flexibility of using SystemC comes from its rich data types with C++ default and user defined types. Relying on the widely used objective-oriented software language, SystemC gets more and more popular for its powerful features inherited from C++.

SystemC design methodology The current SystemC design flow does not cover on automatic synthesis stage, even for digital designs. There are missing connections between SystemC simulation and synthesis. Figure 2.12 shows a draft design flow. It starts from SystemC system level design. Then the design is simulated and verified in the design loop. This design loop is for a complete system level design including system level architectural design, system level simulations and verifications. However, the system level design results need to be converted to

other synthesisable HDLs such as VHDL or Verilog manually for synthesis [105] and further refinement.

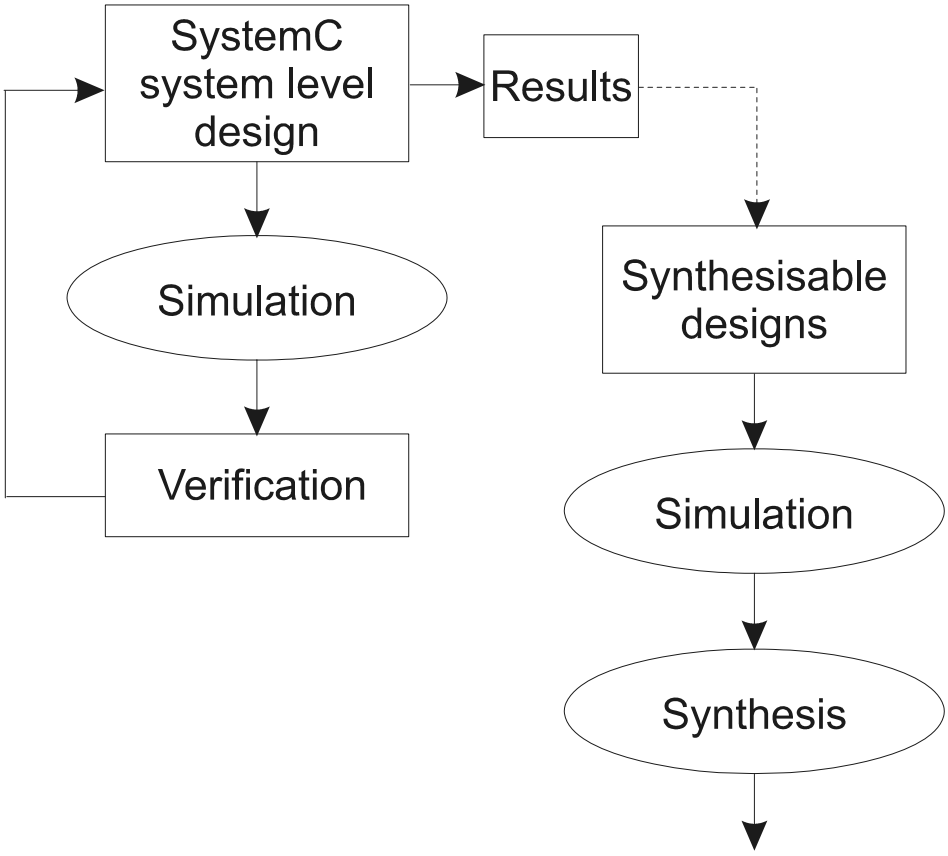


FIGURE 2.12: A typical SystemC design and synthesis process.

SystemC provides a platform to enhance productivity. Both high level and RTL level designs can be implemented in SystemC. Low level designs can be refined from system specifications. Because designers can work on the design at different levels using one uniform language, this makes the design easier to be managed.

SystemC-A: extending SystemC for AMS simulation and modelling Although SystemC is designed for digital system simulation, recently developments try to extend the language capable for AMS simulation and modelling [9, 106, 107]. One important aspect is to develop a class package for analogue modelling. A successful development with practical examples can be found in recent works [9, 106]. Three key elements are utilised for analogue designs. The first one is called system variable. Because it is an effective method to represent the system using DAEs, a system variable is needed for the unknowns in the equations. Secondly, a class

of analogue components are developed to facilitate the representation of analogue nodes, component instances and analogue sources. Thirdly, some methods need to be built to solve the DAEs. The three elements form a complete set of utilities for the representation and simulation needs.

To cooperate with AMS designs, the mechanism of handling the synchronisation between the digital and analogue parts is essential. Firstly, there should be some means to convert the signals from one domain to another. This has been implemented [9] with two specific SystemC modules: one for the digital to analogue conversion and one for the opposite direction. Secondly, because the events in the two domains are in different forms, it is necessary for the system to recognise as well as synchronise them between the analogue and digital solvers. Digital simulation uses traditional delta cycle concept while analogue events are evaluated only when certain behaviour is detected. The exchange of the notification of events in both digital and analogue domains should ensure that extra simulation cycles will be added to propagate events.

2.4 Sigma delta modulation

ADCs can be categorised as: Nyquist-rate and oversampling converters. The sampling frequency of Nyquist-rate converters must be at least twice of the maximum signal bandwidth. One sample in the input signal corresponds to a batch of bits on all the output pins [108]. The oversampling ADCs can achieve higher resolution than Nyquist-rate ADCs and release critical requirements on the IC fabrication process by sacrificing the signal bandwidth. Oversampling and noise shaping are the two main techniques employed to achieve their advantages [109, 110]. One input sample corresponds to a long sequence of digital bits on one output pin and further digital signal processing techniques generate a good estimation on the output based on these bits. Traditionally, as one of the oversampling modulators, SDMs are applied for low frequency designs such as audio, instrumentation and sonar systems because they can have higher output resolution than other ADC structures such as flash, pipelined and successive approximation.

2.4.1 Delta and sigma-delta modulation

There are two kinds of modulators often seen in oversampling converters. The first one is the Delta modulator whose structure is shown in figure 2.13 a). Modulators of this kind have a feedback loop that contains one ADC, one digital to analogue converter (DAC) and one filter. The filter is often implemented by an integrator and placed on the feedback path. The whole system modulates the input signal onto the difference between two successive input samples plus a first order noise difference generated from the quantiser. The linearised model of the quantiser is an adder with an external noise source E as shown in figure 2.13 b). The following difference equation is used to describe the behaviour of the system:

$$v(i) = u(i) - u(i-1) + n(i) - n(i-1) \tag{2.7}$$

where vector v is the output signal, vector u is the input signal, n is the noise signal and i is the index.

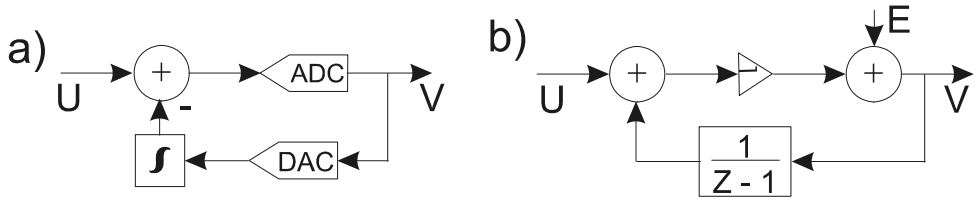


FIGURE 2.13: a) structure of the Delta modulator as an ADC; b) the linearised model.

The second type is SDM, whose structure and linearised model are shown in figure 2.14 a) and b) separately. The system is composed by the same components as in delta modulators but with different arrangement. The filter is on the feed forward path. The output of the system is now a delayed input plus a first order difference of the noise signal as the following difference equation:

$$v(i) = u(i-1) + n(i) - n(i-1) \tag{2.8}$$

In these two structures, the DAC is the main component that generates noise and affects the performance of the system [111]. To reduce the non-linear distortion from the DAC, many techniques have been developed such as to use single-bit DAC instead of multi-bit ones or digital correction and dynamic matching [111].

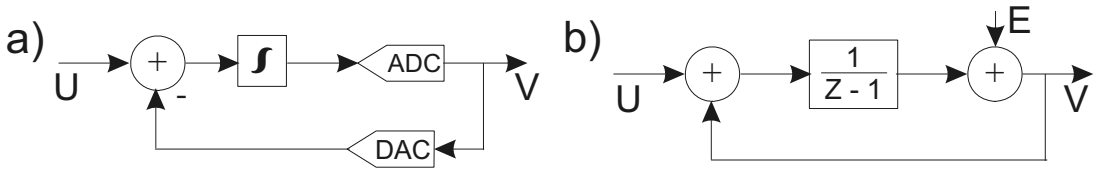


FIGURE 2.14: a) structure of the SDM as an ADC; b) the linearised model.

The structures of delta modulators and SDMs are actually commutative because the adder (subtractor) and the integrator are linear components. Delta modulation does the integration on the output then addition; sigma-delta modulation does addition first then integration. So the result is that the first order difference on the input signal in the Delta modulation is integrated to become the complete input signal in the sigma-delta modulation. This swapping makes it equal if an integrator is added on the input signal in the delta modulators.

2.4.2 Noise-shaping and oversampling

The first order difference of the noise in equation 2.7 and 2.8 is called quantisation noise as it comes from the process when a continuous analogue signal is quantised to a discrete digital signal. This is not avoidable for all the A/D conversion systems. In the z-domain, considering the effect of the filter in the system, the noise $E(z)$ is expressed as $Q(z) = (1 - z^{-1})E(z)$ at the output. To view the relationship of the power spectral density (PSD) of $Q(z)$ and $E(z)$, the amplitudes of $1 - z^{-1}$ is approximated and z is substituted by $e^{2j\pi f/f_s}$, where f is the signal frequency and f_s is the sampling frequency:

$$1 - z^{-1} \quad = \quad 2\sin(\pi\frac{f}{f_s})(\sin\pi\frac{f}{f_s} + j\cos\pi\frac{f}{f_s}) \tag{2.9}$$

therefore, the relationship of the PSD between $Q(z)$ and $E(z)$ is:

$$S_q(f) = (2\sin(\pi\frac{f}{f_s}))^2 S_e(f) \tag{2.10}$$

A well-known conclusion about a "random" quantisation noise is that the root-mean square (RMS) value of the signal follows the following equation:

$$e^2_{RMS} = \frac{\Delta^2}{12} \tag{2.11}$$

where the Δ is the least significant bit of the ADC. With a practical assumption that the noise power spreads over the half of the sampling bandwidth uniformly, the PSD is expressed as:

$$S_e(f) = \frac{e_{RMS}^2}{f_n/2} = \frac{\Delta^2}{6f_n} \tag{2.12}$$

where f_n is the normalised frequency.

Substitute S_e in equation 2.10 by equation 2.12, the result equation shows that the noise power is shaped by the modulator with a factor of $(2\sin(\pi\frac{f}{f_s}))^2$. This factor indicates that the noise power will approach to zero at low frequency range and will increase at high frequency range. This noise shaping technique, as shown in figure 2.15, is one of the two most important concepts behind SDMs.

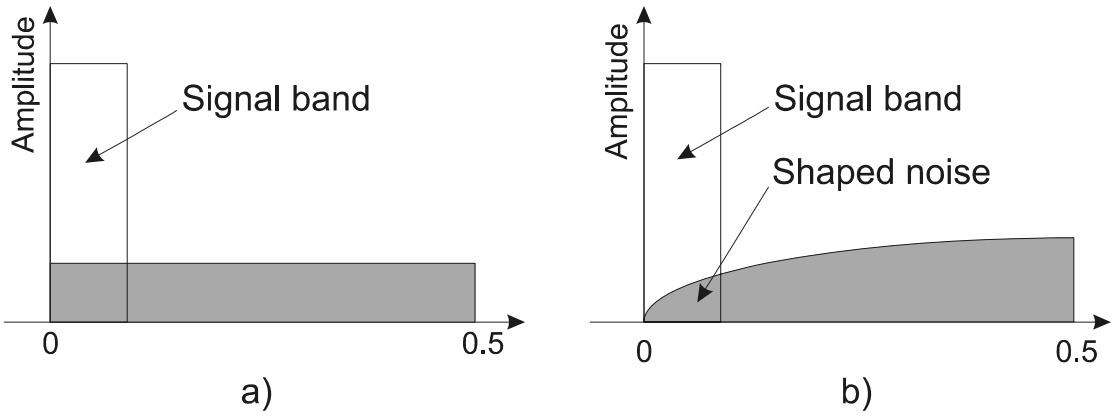


FIGURE 2.15: Noise shaping reduces noise in signal band. a) noise in the signal band before the noise shaping; b) noise is shaped and pushed to high frequency range.

The other key technique in SDM is oversampling. Oversampling can reduce noise level because while the the useful signal keeps its power in the signal band, the noise power always spread over the entire sampling bandwidth, thus the increment of the sampling bandwidth reduces the noise level in the signal band as shown in figure 2.16. This is based on another fact that the power of the noise signal is a constant that can be derived by integrating equation 2.12 from 0 to f_s. The resultant power of the noise in the signal band is:

$$q_{RMS}^2 = \frac{\pi^2 e_{RMS}^2}{3OSR^3} \tag{2.13}$$

where OSR stands for oversampling ratio and is calculated as $OSR = f_s/2f$.

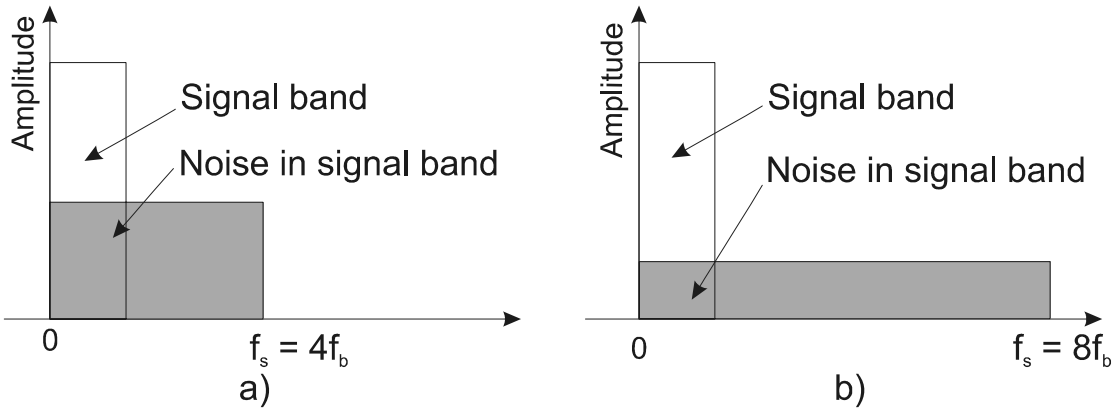


FIGURE 2.16: Oversampling reduces noise in signal band. a) noise distribution when $f_s = 4f$; b) noise distribution when $f_s = 8f$.

OSR has significant influence on the performance. The above equation is logarithmically scaled to give the following relationship:

$$\log_{10}(q_{RMS}^2) = C - 3\log_{10} OSR \tag{2.14}$$

where C is the constant that represents all other values after the logarithmic operation.

A common and qualified measurement of the accuracy of a converter is the signal to noise ratio (SNR). It has a simple relationship with the effective number of bits (ENOB) [111]:

$$SNR = 6.02 \cdot ENOB + 1.76 \tag{2.15}$$

Considering equation 2.14 and 2.15, when the OSR is doubled, an SDM will give 9dB noise decrement thus is equally to get 1.5 ENOB increment on the resolution. However, in practice, as the input signal frequency increases, the difficulty of improving the SDM performance by increasing the OSR gets more and more challenging.

2.4.3 SDM structures

The general structure of a SDM contains a filter block and a quantiser block as shown in figure 2.17 [111]. The filter is a linear component and includes memory

elements. The output of the two-input single-output system is expressed as a linear combination of its inputs U and V:

$$Y(z) = L_0(z)U(z) + L_1(z)V(z) \tag{2.16}$$

where L_0 represents the transfer function of the filter shaping the input signal U and L_1 is the one for the feedback signal.

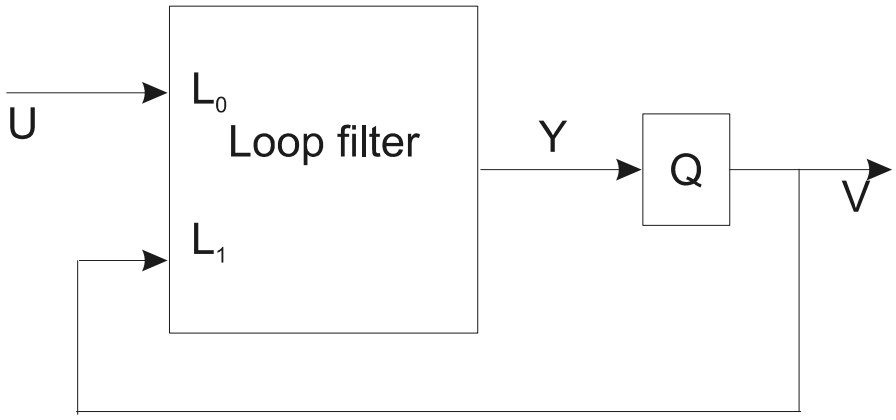


FIGURE 2.17: General structure of a SDM used as an ADC [111].

The quantiser is a single-input single-output system that can be linearised if noise is considered as external and the gain of the quantiser is unity:

$$V(z) = Y(z) + E(z) \tag{2.17}$$

Thus, the overall system has one signal input, one noise input and one output. The relationship between the inputs and output is expressed by the following equation:

$$V(z) = STF(z)U(z) + NTF(z)E(z) \tag{2.18}$$

where NTF stands for noise transfer function and STF for signal transfer function, which are expressed by the following equations for general and highly abstract SDM structures:

$$NTF(z) = \frac{1}{1 - L_1(z)} \quad and \quad STF(z) = \frac{L_0(z)}{1 - L_1(z)} \tag{2.19}$$

Conversely, with given NTF and STF, the corresponding loop filter transfer function can be calculated directly from the following equations:

$$L_0(z) = \frac{STF(z)}{NTF(z)} \quad and \quad L_1 = 1 - \frac{1}{NTF(z)} \tag{2.20}$$

These equations are independent of the structures of SDMs, which means that the input-output characteristics of the modulators are solely determined by the NTF, STF and the performance of the quantiser. Also, because the NTF provides all the poles for both filters, it has the most significant influence on the overall performance [111].

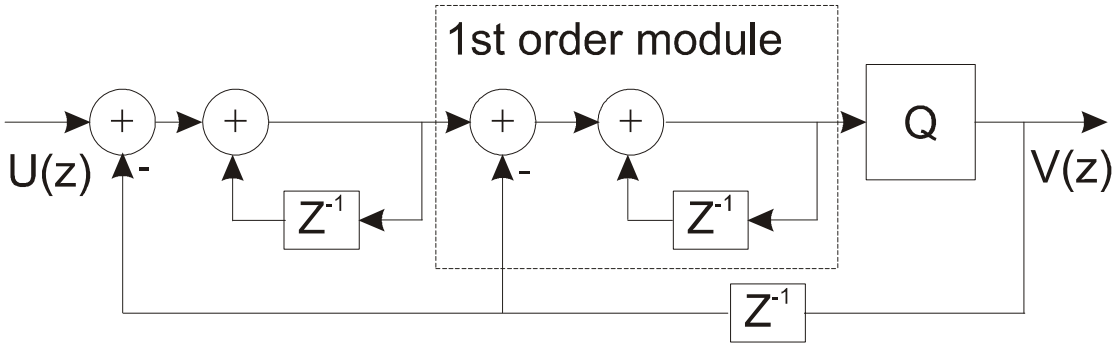


FIGURE 2.18: Detailed z-domain model of the delay free structure of 2^{nd} order modulators constructed by cascading 1^{st} order modules.

One of the strategies to make high-order SDMs is to cascade first-order modulators as shown in figure 2.18. However, typically, first-order modulator is implemented by an integrator with unit delay not a delay free one as in the figure. When the first-order modulators are cascaded on the feed forward path, the input signal is delayed more on the propagation to the output but the absolute form of STF stays the same; noise is differentiated more times and sharper noise shaping function will be achieved. Suppose STF is delayed by k unit clock periods and the NTF is differentiated n times, they take the following form:

$$STF = z^{-k} \quad and \quad NTF = (1 - z^{-1})^n \tag{2.21}$$

by equation 2.20, the loop filters are:

$$L_0 = \frac{z^{-k}}{(1-z^{-1})^n} = \frac{z^{n-k}}{(z-1)^n} \tag{2.22}$$

$$L_1 = 1 - \frac{1}{(1-z^{-1})^n} = \frac{(1-z^{-1})^n - 1}{(1-z^{-1})^n} \tag{2.23}$$

Poles of equation 2.22 and 2.23 are all located at $z = 1$. Zeros for equation 2.22 are at $z = 0$. Zeros for equation 2.23 lie on the solutions of $(1-z^{-1})^n - 1 = 0$, which is $(1-z^{-1})^n = 1$. Considering equation 2.9, noise in higher order modulators will be more attenuated in the signal band and shaped more to high frequency ranges than in low order systems. In practice, the advances of high-order SDMs may only be achieved by sacrificing the dynamic range.

Another structure commonly seen in high order SDMs is called multi-stage noise-shaping (MASH) modulator. Figure 2.19 shows the general structure. The first stage takes the input and the second stage uses the noise from the first stage as the input. The z-transform outputs of the two stages are:

$$V_1(z) = STF_1 U(z) + NTF_1 E_1(z) \tag{2.24}$$

$$V_2(z) = STF_2 E_1(z) + NTF_2 E_2(z) \tag{2.25}$$

The transfer functions of the two digital filters, i.e., H_1, H_2, are designed to cancel the quantisation noise generated from the first stage. According to the above two equations, their relationship should be:

$$H_1 NTF_1 - H_2 STF_2 = 0 \tag{2.26}$$

The usual and practical choice is to make $H_1 = STF_2$ and $H_2 = NTF_1$. For both of the two stages, the STFs are very likely to have delays. Then the overall output can be found to be:

$$V(z) = STF_1 STF_2 U(z) + NTF_1 NTF_2 e_2(Z) \tag{2.27}$$

If both of the two stages use second-order modulators, then $STF_1 = STF_2 = z^{-2}$ and $NTF_1 = NTF_2 = (1-z^{-1})^2$. Thus the overall $NTF = (1-z^{-1})^4$. So the noise is shaped as by a fourth order system while the system keeps the robustness

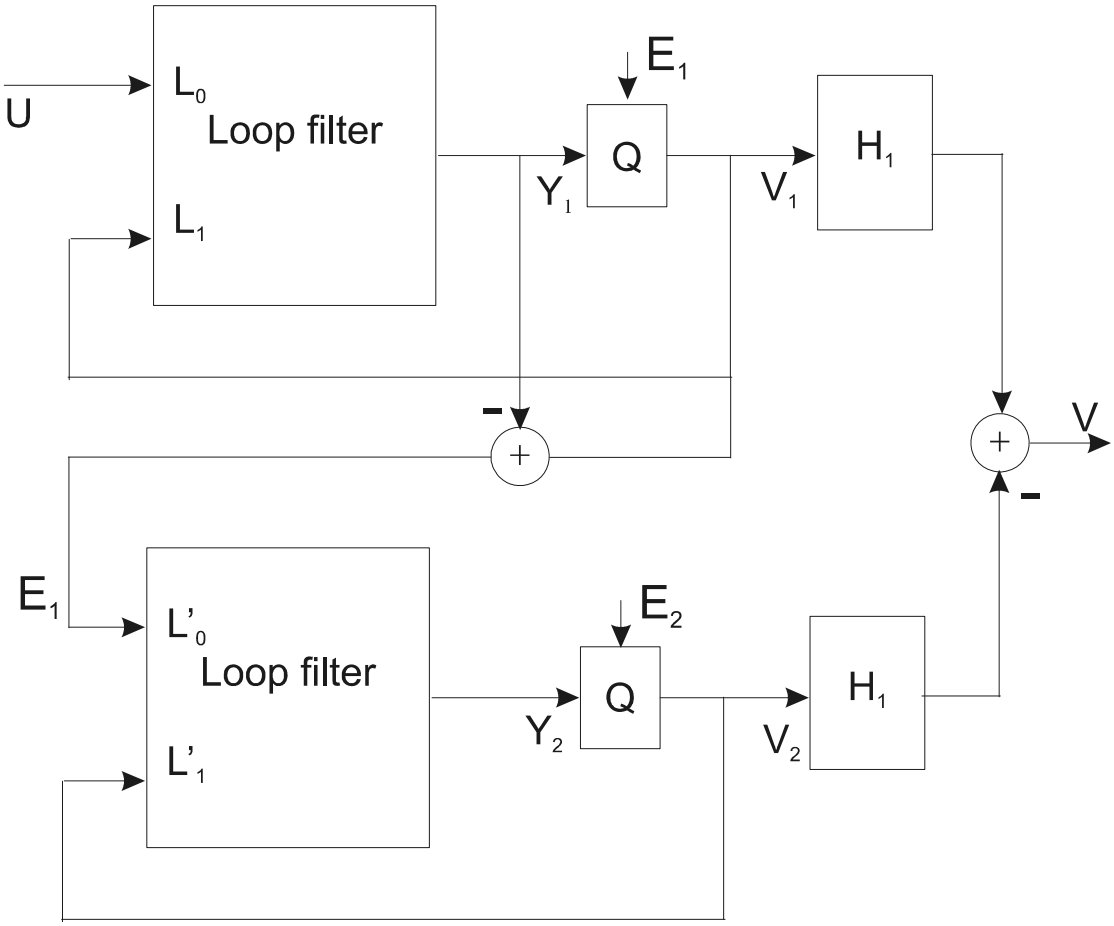


FIGURE 2.19: MASH structure of two-stage SDMs.

of second-order modulators. Another advantage is that as the input to the second stage $e_1(n)$ is "noise" already, the output of the second stage is closer to the white noise. This can reduce the harmonics of the output thus reduce some non-linear effects. Also, as $H_2 = NTF_1$ is a high-pass filter and the non-linear errors from the second stage will be multiplied with H_2 before the final summation, the non-linear errors in the signal band will be suppressed.

2.5 Concluding remarks

This chapter presents a review of the literature related to this research with the exception of the SVM technique which is introduced and reviewed separately in the next chapter for clarity. Regarding AMS-synthesis related techniques, the

review covers general AMS synthesis systems and then narrows down to the specific techniques that are usually seen in many existing research publications. This part highlights the core role of performance modelling and optimization in AMS synthesis and their treatment by some traditional approaches; advancement and generalisation of performance modelling and optimization for the purpose of AMS synthesis is the main focus in this research. In addition, the underlying techniques needed for the development of the case studies are reviewed. This includes theoretical analysis of SDM designs and review of HDLs.

Chapter 3

Support vector machines: an introduction and the state of the art

This chapter has a twofold purpose. Section 3.1 contains an introduction to the support vector machine technology used in this research. It focuses on how the technology is developed from concepts. Section 3.2 is mainly about the state of the art for the application of the technology in AMS performance modelling and optimisation. A SVM trainer is also introduced in this chapter.

3.1 SVM introduction

SVMs [35] were proposed originally in the context of machine learning for classification problems on large sets of data which has complex and unknown relationship with variables. Being supervised learning, the application of SVM models needs firstly a training process to construct the models and then a testing process to verify the models before they can be used to solve practical problems. The constructions of the SVM classification or regression models follow the same approach for different applications: simulations are needed to generate data where the information about the relationship between the design and performance parameters is contained; then SVM trainer extracts knowledge from the data and constructs SVM classification or regression models that can be stored in a database. The

application of the databases, as the working flow shows in figure 3.1, is to provide predictions on unknown design sets with their corresponding classes or numerical approximations.

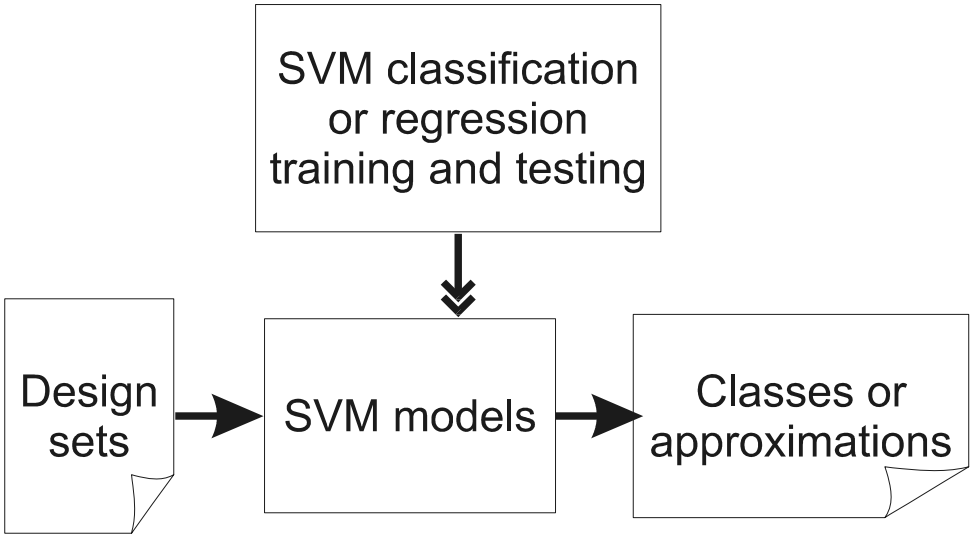


FIGURE 3.1: SVM models in a typical application environment.

This introduction is based on SVM technologies for classification problems.

3.1.1 Background

Classification is to recognise raw pattern data for specific classes by measuring the similarity between the data and the feature of the classes. A distance matrix is commonly employed for this purpose.

Think of an example to use a 3 dimensional plane to classify 2 dimensional patterns in an X-Y space (x_i, y_i), $i \in [1, \cdots, n]$. The plane is defined by:

$$z = \omega x + \upsilon y + c \tag{3.1}$$

So any point that makes $z - \omega x - \upsilon y - c = 0$ lies on the plane. If a pattern (x_i, y_i) has a corresponding positive value in the right hand side of equation 3.1, then the point is above the plane, otherwise below the plane as shown in figure 3.2.

The plane can be generalised for a multi-dimensional space. Considering that there is always an error expected between the prediction and the truth, an extra

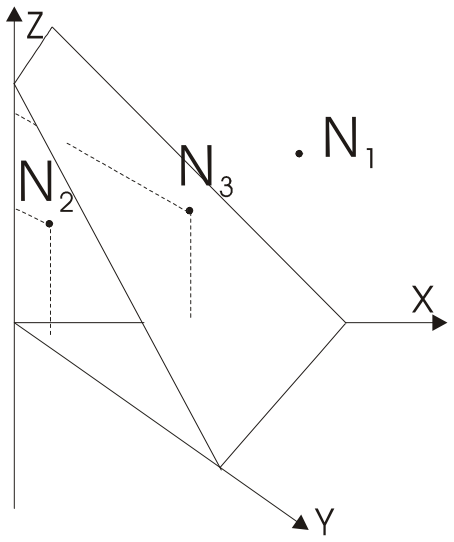


FIGURE 3.2: The 3 dimensional surface plane separates the space into two parts. Three nodes (N_1, N_2, N_3) are projected onto the plane and classified.

term can be added to represent the errors. So the plane is of the following form:

$$O = \sum_{i=1}^{n} \omega_i x_i + \epsilon$$

where O is real numbered output to be used for different classes and ϵ is the error term.

Before applying the plane for classification problems, the key step is to determine the values of the elements in the weight vector ω. As shown in figure 3.3, the weight vector straightly decides the generality capability of the separating line (one dimensional plane). This can become a very complex issue in multi-dimensional cases. It should be noted that the process of constructing the separating plane is the training process.

3.1.2 Statistical learning and kernel method

3.1.2.1 Similarity measurement

To generalise the plane constructed using known data onto unknown data, first of all, it is important to quantify the similarity between known and unknown data.

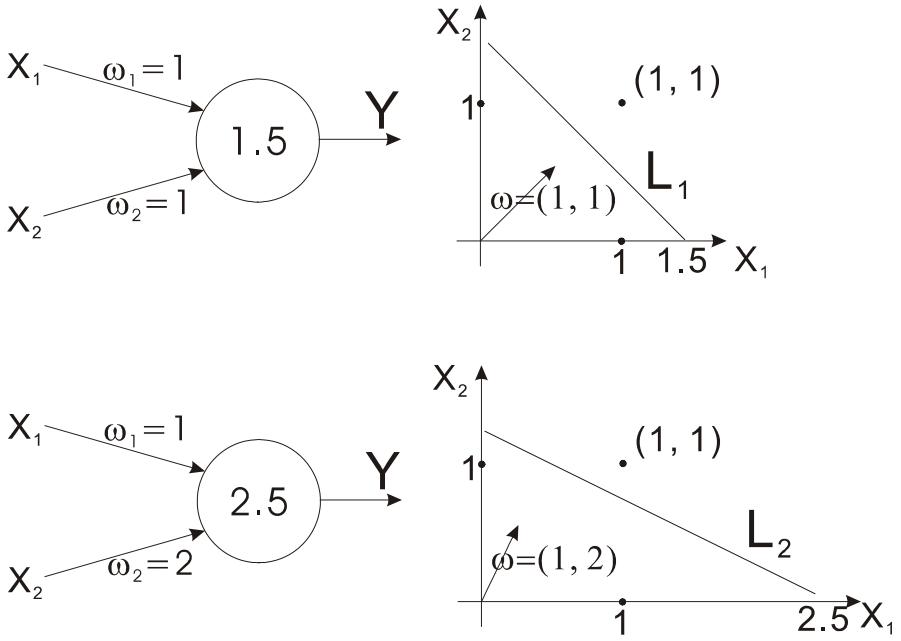


FIGURE 3.3: The two separation lines L_1 and L_2 have different capability features for unknown data as their corresponding weight vectors ω_1 and ω_2 are of different values.

In SVMs, the dot-product is utilised for this purpose, which is defined as:

$$\vec{v}_1 \cdot \vec{v}_2 = \sum_{i=1}^{m} (\vec{v}_1)_i (\vec{v}_2)_i \tag{3.2}$$

where $\vec{v}_1$ and $\vec{v}_2$ are two vectors in an N dimensional space $\Re^N$. Using dot-products, the relationship between data points can be constructed geometrically in terms of angles, lengths and distances. More importantly, a designated feature space can be created so that the separability of the patterns can be maximised when they are mapped to the feature space. As explained next, this mapping simplifies calculation because of the following equivalence:

$$k(x, x^{'}) = \mathbf{x} \cdot \mathbf{x}^{'} = \Phi(x) \cdot \Phi(x^{'}) \tag{3.3}$$

where function $\Phi : \chi \rightarrow F$ is the mapping function for each pattern in the input space and $\mathbf{x}$ and $\mathbf{x}^{'}$ are mapped patterns in the feature space, function k is called the kernel function.

Assume two classes are to be separated without any prior information about the probabilities of the two classes by the following means of the input patterns from

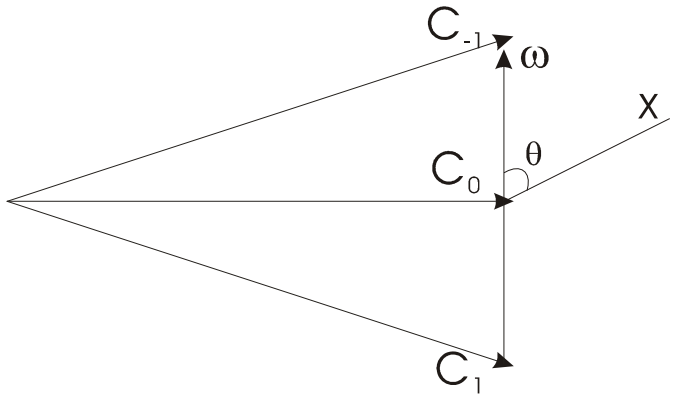


FIGURE 3.4: Geometrical relationship between the two classes C_1, C_{-1}, their distance $\omega = C_1 - C_{-1}$, their mid-point $C = \frac{C_1+C_{-1}}{2}$ and the new pattern $\mathbf{x}$.

classes $y_i = 1$ and $y_i = -1$:

$$C_1 = \frac{1}{m_1} \sum_{i:y_i=1}^{m_1} \mathbf{x}_i \tag{3.4}$$

$$C_{-1} = \frac{1}{m_{-1}} \sum_{i:y_i=-1}^{m_{-1}} \mathbf{x}_i \tag{3.5}$$

where m_1 and m_{-1} are the total number of patterns in the 1 and -1 classes.

As shown in figure 3.4, the label of a new pattern $\mathbf{x}$ depends on the angle enclosed by $\mathbf{x}$ and ω.

$$y = sgn(cos\theta) \tag{3.6}$$

According to the Law of Consines, equation 3.6 can be rewritten as:

$$y = sgn((\mathbf{x} - C) \cdot \omega) \tag{3.7}$$

$$= sgn((\mathbf{x} \cdot C_1) - (\mathbf{x} \cdot C_{-1}) + b) \tag{3.8}$$

where the offset parameter b is

$$b = \frac{1}{2}(||C_1||^2 - ||C_{-1}||^2) \tag{3.9}$$

Equation 3.8 and 3.9 can be rewritten in the input space by substituting equation 3.3:

$$y = sgn\left(\frac{1}{m_1}\sum_{i:y_i=1}(\mathbf{x}\cdot\mathbf{x}_i) - \frac{1}{m_{-1}}\sum_{i:y_i=-1}(\mathbf{x}\cdot\mathbf{x}_i) + b\right) \tag{3.10}$$

$$= sgn\left(\frac{1}{m_1}\sum_{i:y_i=+1}k(x,x_i) - \frac{1}{m_{-1}}\sum_{i:y_i=-1}k(x,x_i) + b\right) \tag{3.11}$$

$$b = \frac{1}{2}\left(\frac{1}{m_1^2}\sum_{i,j:y_{i,j}=1}k(x_i,x_j) - \frac{1}{m_{-1}^2}\sum_{i,j:y_{i,j}=-1}k(x_i,x_j)\right) \tag{3.12}$$

If the two classes C_1, C_{-1} have the same distance to the origin and the kernel function is integral on the input space, then the offset parameter $b = 0$ and equation 3.11 is called the Bayes decision boundary. This separation can be correctly estimated by Parzen windows, i.e the correct label depends on the larger of equation 3.13 and 3.14:

$$P_1(x) = \frac{1}{m_1}\sum_{i:y_i=1}k(x,x_i) \tag{3.13}$$

$$p_2(x) = \frac{1}{m_{-1}}\sum_{i:y_i=-1}k(x,x_i) \tag{3.14}$$

3.1.2.2 Generality capability and Vapnik Chervonenkis dimension

The knowledge of a problem can only be extracted from known data. During the extraction process, the generality capability of the knowledge onto unknown data is a great concern. A function that works well on training data may not be a good one for unknown data. In fact, assume a training set

$$(x_1,y_1),\ldots,(x_m,y_m) \in R^n \to \{\pm 1\} \tag{3.15}$$

and a testing set

$$(\bar{x_1},\bar{y_1}),\ldots,(\bar{x_m},\bar{y_m}) \in R^n \to \{\pm 1\} \tag{3.16}$$

$$subject\ \ to\ \ \{\bar{x_1},\ldots,\bar{x_m}\} \cap \{x_1,\ldots,x_m\} \in \{\} \tag{3.17}$$

there can exist a function f^* that fulfils the following results simultaneously:

$$f^*(x_i) = f(x_i) \tag{3.18}$$

$$but \quad f^*(\bar{x_i}) \neq f(\bar{x_i}) \tag{3.19}$$

So it is concluded that only minimising training error does not consequently imply a small testing error [43]. The two kinds of errors are distinguished as:

$$R_{emp}[f] = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}|f(x_i) - y_i| \tag{3.20}$$

is the training error and called empirical risk and

$$R[f] = \int \frac{1}{2}|f(x) - y|dP(x,y) \tag{3.21}$$

for testing error and called risk.

To choose a classification function for a fixed data set, the statistical learning theory or Vapnik Chervonenkis (VC) theory [35] states:

- a suitable function is restricted to have a capacity that is suitable for the amount of available training data
- VC theory provides bounds on the risk (testing error)
- minimisation of these bounds depends on both the empirical risk R_{emp} and the capacity of the function.

This minimisation method is called structured risk minimization (SRM). The capability of a classification function is defined as how much data the function can be successfully applied on for correct classification. As shown in figure 3.5 and 3.6, the capability of an oriented straight line is 3.

The relationship between risk bounds and VC dimension is: if $h < m$ is the VC dimension of a class of functions that a learning system can implement, then for all functions of that class, with a probability of at least $1 - \eta$, the bound is

$$R(\alpha) \leq R_{emp}(\alpha) + \Phi(\frac{h}{m}, \frac{log(\eta)}{m}) \tag{3.22}$$

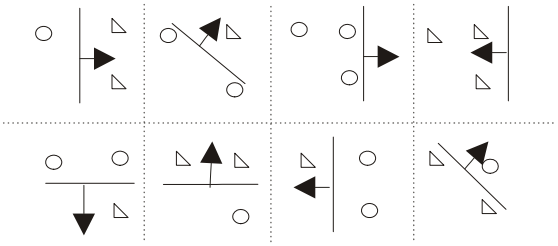


FIGURE 3.5: Two data sets with 3 samples separated by an oriented straight line. Triangles represent class +1 and circles are −1.

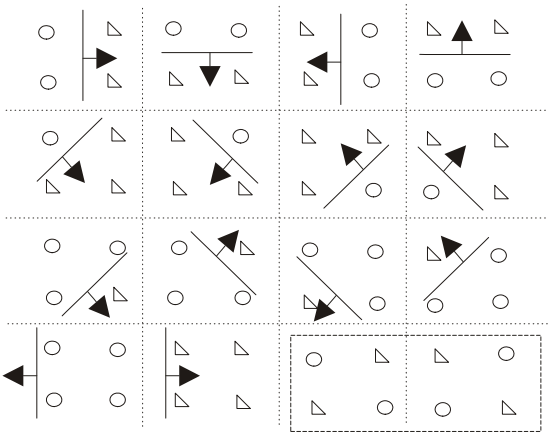


FIGURE 3.6: Two data sets with 4 points can not be separated by the oriented straight line used in figure 3.5. The last two cases in the dashed square can not be separated successfully. Triangles represent +1 and circles are labeled −1.

and coefficient term Φ is

$$\Phi(\frac{h}{m}, \frac{log(\eta)}{m}) = \sqrt{\frac{h(log(\frac{2m}{\eta}) + 1) - log(\frac{\eta}{4})}{m}} \tag{3.23}$$

In equation 3.23, term Φ is monotonic with h. So no training error means a high VC dimension and this will consequently give a high Φ and enlarge the bound according to equation 3.22. So it will not support possible hopes that due to the small training error, we can have a small testing error.

3.1.3 Hyperplane classifier

The hyperplane classifier algorithm performs dot-products in the feature space with VC theory considerations, i.e the capability of the separation functions need

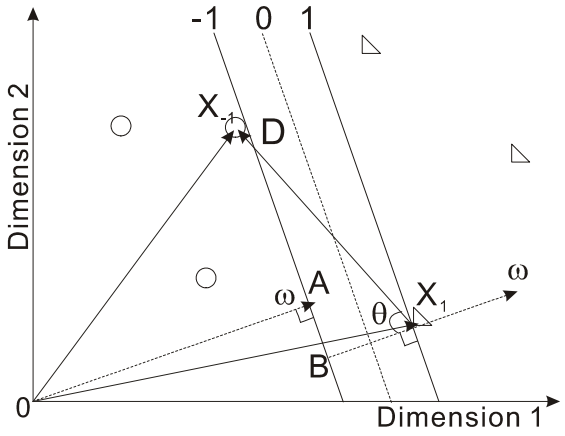


FIGURE 3.7: The hyperplane (labelled as '0') separates two classes: triangles (class 1) and circles (class −1). Geometrical relationships between the patterns and the hyperplane are illustrated.

to be calculated. For instance, the following is a hyperplane:

$$(\omega \mathbf{x}) + b = 0 \tag{3.24}$$

where $\omega \in \Re^N$ is the coefficient vector in the N dimensional feature space $\Re$ as the mapped patters $\mathbf{x}$, and $b \in \Re$ is the bias vector. The separation function using this hyperplane definition can thus be expressed:

$$f(x) = sgn(\omega \mathbf{x}) + b) \tag{3.25}$$

For this separation, two observations need to be emphasised:

- there should exist a unique plane yielding a maximum margin of separation between the classes
- the generality capability decreases with margin increasing

According to the observations, construction of the hyperplane is a constrained optimisation problem. First of all, the margin needs to be defined. Assume a hyperplane is used to separate two classes represented by triangles and circles as shown in figure 3.7. Vector ω defines the hyperplane and is perpendicular to it. Patterns $\mathbf{x}_1$ and $\mathbf{x}_2$ are located on the boundary hyperplanes. The difference vector $D = \mathbf{x}_1 - \mathbf{x}_2$ encloses an angle of θ with the perpendicular ω. The following process derives the distance between the two patterns:

$$\begin{cases} D = \mathbf{x}_1 - \mathbf{x}_2 \\ \omega \cdot \mathbf{x}_1 + b = 1 \\ \omega \cdot \mathbf{x}_2 + b = -1 \end{cases} \quad \Rightarrow \quad \omega \cdot (\mathbf{x}_1 - \mathbf{x}_2) = 2 \tag{3.26}$$

$$\Rightarrow \quad |\omega| \cdot |\mathbf{x}_1 - \mathbf{x}_2| \cdot cos\theta = 2 \tag{3.27}$$

$$\Rightarrow \quad |\mathbf{x}_1 - \mathbf{x}_2| \cdot cos\theta = \frac{2}{|\omega|} \tag{3.28}$$

$$\Rightarrow \quad |D| \cdot cos\theta = \frac{2}{|\omega|} \tag{3.29}$$

So the constrained optimisation problem can be equally expressed by minimising equation 3.30 subject to equation 3.31 as the following:

$$\tau(\omega) = \frac{1}{2}||\omega||^2 \tag{3.30}$$

$$y_i \cdot (\omega \cdot \mathbf{x}_i) + b \geq 1 \quad for \quad i = 1, \ldots, m \tag{3.31}$$

A typical technique to solve the above problem is to use Lagrange optimisation.

$$L(\omega, b, \alpha) = \frac{1}{2}||\omega||^2 - \sum_{i=1}^{m} \alpha_i (y_i \cdot (\omega \cdot \mathbf{x}_i) + b) \tag{3.32}$$

where α_i is a Lagrange coefficient vector. The goal is to find the minimum value of function L with respect to its primal variables (ω and b) or the maximum value with respect to its dual variable (α). When equation 3.31 is not taking the equality condition, intuitively, the corresponding α_is have to be zero in the second term in equation 3.32 for the maximum value of L. This statement is called Karush-Kuhn-Tucker (KKT) complementarity conditions. It is clear that the partial differentiations of function L with the respect to its primal variables ω and b should be zeros:

$$\frac{\partial L(\omega, b, \alpha)}{\partial \omega} = 0 \quad \Rightarrow \quad |\omega| = \sum_{i=1}^{m} \alpha \cdot y_i \cdot \mathbf{x}_i \tag{3.33}$$

$$\frac{\partial L(\omega, b, \alpha)}{\partial \alpha} = 0 \quad \Rightarrow \quad \sum_{i=1}^{m} \alpha \cdot y_i = 0 \tag{3.34}$$

Equation 3.33 indicates that some α_i are not zero, which means the corresponding points contribute to the construction of the hyperplane; however, equation 3.34 indicates that some α_i are zero, which means the corresponding points are not used in the construction of the hyperplane. In SVM, all points that have non-zero Lagrange coefficients are called Support Vectors (SV).

Considering equation 3.33, the decision function is written as:

$$f(x) = sgn(\sum_{i=1}^{m}(y_i \cdot \alpha_i(\mathbf{x} \cdot \mathbf{x}_i) + b)) \tag{3.35}$$

3.1.4 Support vector classifier

Considering equation 3.3, when equation 3.35 is rewritten in the input space, the mapping from the input space and the feature space can be implicitly carried out by using the kernel functions:

$$\begin{aligned} f(x) &= sgn(\sum_{i=1}^{m}(y_i \cdot \alpha_i(\mathbf{x} \cdot \mathbf{x}_i) + b)) && (3.36)\\ &= sgn(\sum_{i=1}^{m}(y_i \cdot \alpha_i \Phi(x)\Phi(x_i) + b)) && (3.37)\\ &= sgn(\sum_{i=1}^{m}(y_i \cdot \alpha_i k(x, x_i) + b)) && (3.38) \end{aligned}$$

Up to here, all the analysis and derivations are based on an assumption that the classes can be perfectly separated without any errors. However, in practice, this assumption is almost never valid and infeasible patterns can be seen very commonly in many cases. So, the original objective function in equation 3.30 should be re-expressed:

$$minimise \quad \frac{1}{2}||\omega||^2 + C\sum_{i=1}^{m}(\xi_i + \xi_i^*) \tag{3.39}$$

$$subject \ \ to \quad \begin{cases} y_i - \omega \cdot \mathbf{x}_i - b \leq \epsilon + \xi_i \\ \omega \cdot \mathbf{x}_i + b - y_i \leq \epsilon + \xi_i^* \end{cases} \tag{3.40}$$

where constant C is the trade-off parameter that determines the weight of the error terms in the optimisation problem. So from equation 3.39, the maximisation

of the generalisation capability and minimisation of the training errors are jointly solved so that a saddle point of the optimisation problem can be found. This is the principle of the SRM. The solving still uses Lagrange optimisation techniques. The new optimisation problem with infeasible samples is as the following:

$$\begin{aligned} maximise \quad & -\frac{1}{2}\sum_{i,j=1}^{m}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)k(x_i, x_j) \\ & -\epsilon\sum_{i=1}^{m}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{m}(\alpha_i - \alpha_i^*)y_i \end{aligned} \tag{3.41}$$

$$subject \ \ to \quad \begin{cases} \sum_{i=1}^{m}(\alpha_j - \alpha_j^*) = 0 \\ \alpha_i, \alpha_i^* \in [0, C] \end{cases} \tag{3.42}$$

where αs are the Lagrangian coefficients.

3.1.5 Kernels

Kernel function is one of the main extensible factors of the SVM technique. Several kernel functions have been developed and more are emerging. Most applications use the following four kinds of kernels:

- Linear Kernel: $k(x_i, x) = x_i \cdot x$ [112]
- Polynomial Kernel: $k(x_i, x) = (\gamma x_i \cdot x + r)^d$ [113]
- Radial Basis Function Kernel (RBF): $k(x_i, x) = exp(-\gamma||x_i - x||^2), \gamma > 0$ [46]
- Sigmoid Kernel: $k(x, x\prime) = tanh(\gamma x_i \cdot x + r)$ [114]

where γ, r, d are training parameters of the kernel functions that determine the functions' characteristics. Only when the parameters are assigned to optimal values for the training data set that the corresponding model constructed can provide optimal modelling accuracy and generality. So the task of finding optimal models is essentially finding the optimal training parameter values.

Among these kernels, the RBF kernel is recommended for the following three main reasons [42]. Firstly, the RBF kernel has better capability to handle non-linear

relationships between the input patterns than the linear kernel. Actually, both of the linear and sigmoid kernels behave like special cases of the RBF kernel with special parameters. Secondly, compared to the polynomial kernel, the RBF kernel has less training parameters so it is easy tuning. Finally, the RBF kernel has less numerical difficulties than the sigmoid and polynomial kernels. In this research, the SVMs are used with the RBF kernel in all the case studies.

3.1.6 SVM regression

SVM regression is of the same idea as in SVM classification. In the ϵ-SVM regression [35], the goal is to construct a function $f(x)$ that has at the most ϵ deviation from the simulation results y_i^{sim} for all the training data samples. So in SVM regression problems, the SRM is minimises the estimation errors and maximises the generality of the regression function, which is usually defined by the so-called ϵ-insensitive loss function $|\xi|_\epsilon$ as the following:

$$|\xi|_\epsilon = \begin{cases} 0 & if|\xi| < \epsilon \\ |\xi| - \epsilon & otherwise \end{cases} \tag{3.43}$$

This is illustrated graphically in figure 3.8.

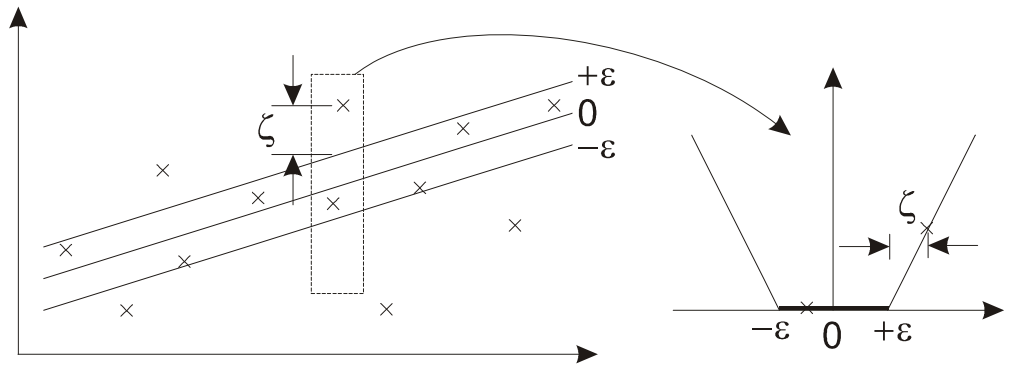


FIGURE 3.8: ϵ-insensitive soft margin loss setting for a linear SVM.

3.2 SVM in AMS performance modelling - the state of the art

SVM technique was firstly applied to solve automatic analogue circuit sizing problems in 2003 [46]. Since then, as a technology with lot of potential, it has attracted a lot of attention [45, 47, 49, 50].

Currently, the application of SVM technology on AMS performance modelling and optimisation is in its initial state. Research has just emerged in the field. A summary has been created in table 3.1 based on the collected publications. A few points can be outlined. Firstly, most of the applications use classical basic analogue circuits such as various amplifiers as the case studies. Secondly, the number of samples for SVM training varies significantly from case to case. Generally, the number of training samples is mainly related to the complexity of the relationship between the design and performance spaces not the dimension of the design and performance spaces. There is no theory to define how many samples are adequate for a design. Many of them are carried out to prove that SVM is applicable for the AMS performance modelling problem.

TABLE 3.1: Summary of recent SVM performance modelling applications.

Ref	Example	Design-to-performance mapping	# of training samples	Time(h) to construct optimal models
[45]	high speed CMOS OTA	13-to-5	243	N/A
[46]	low noise amplifier	7-to-7	up to 50000	1 (approx)
[47]	single stage OTA	6-to-3	7000	0.85
	two stage OTA	11-to-3	7800	2.2
[49]	2^{nd} order Σ-Δ modulator	5-to-5	3125	1.1
[50]	Colpitts RF filter	6-to-5	4096	0.4

Among all the publications, the RBF kernel is solely utilised in all the references for SVM training. The model accuracy is the only factor to evaluate the quality of the models. On this point, the references can show with different cases that the technique is able to construct accurate models for analogue and AMS designs. The

computational cost is not considered in the construction of the SVM performance models except the last two from this research. The computational costs in the last column are for the construction of optimal models, i.e. the optimum pair of SVM training parameters. Other research uses manual methods for training parameter determination or there are no explicit statements that indicate that an automatic method is used. Compared to those work, this research represents an initiative effort for automatic SVM performance model construction, in addition, the computational cost of the SVM training parameter determination has been explored.

3.2.1 LibSVM - an SVM trainer

There have been quite a few software packages available for SVM applications. A good list can be found on the kernel machines web site [115]. Among them, LibSVM [116] is selected in this research for its reliability and flexibility. Firstly, the software has been applied in many SVM related performance modelling initiatives as explained previously. This reflects the reliability of the software for various designs. Also, as the software is created as an open source package, it is fully visible to users. With its Windows-based executable files, it is easy to make the synthesis system independent of the software by using a function for interfacing.

The software package contains executable files to do scaling, training and prediction separately for Windows applications. The execution of the tools is fully controlled by users in a command-line interface. Popular programming languages such as C and MATLAB can call the tool with standard interface functions or commands. This eases the cooperation between LibSVM and user defined functions. Secondly, the documentation of the tool is outstanding and it contains rich samples on its web page. This includes a user guide and research publications which can give an insight into how the tool works.

3.3 Concluding remarks

In this chapter, the SVM technology is introduced. Key concepts such as VC dimensional and SRM principles have been explained with examples. The relationships between the concepts and the SVM have been derived and explained.

The state-of-the-art of SVM in the application of AMS performance modelling has been summarised and reviewed. The main SVM trainer used in this research has also been introduced.

Chapter 4

Linearly graded automated performance model construction using support vector machines

Linearly graded performance modelling methodology is an extension of the classical 'good-bad' classification model. This initiative aims to implement the 'divide and conquer' strategy to partition large analogue design space into smaller subspaces using user specified linear grades, then construct models for each subspace. This chapter contains an introduction of this methodology in section 4.1. An automated model construction algorithm based on a traditional grid-search approach and a new heuristic grading algorithm is presented in section 4.2. Section 4.3 is a preliminary application of the linearly graded performance modelling methodology on a 2^{nd} order lowpass analogue filter. The SVM classification models have been successfully constructed for it.

4.1 Linearly graded performance modelling methodology

4.1.1 Relationship between design and performance spaces

The design space of an AMS design is formed by the design parameters that have influence on the performance of the system. Each design parameter represents one dimension in the design space. When several design parameters are combined, a multi-dimensional design space is then formed. As in most cases, the design parameters are constrained in ranges, so the design space is more likely to be a closed multi-dimensional space. In the same way, the corresponding performance space can be formed by all the performance parameters that designers are interested in.

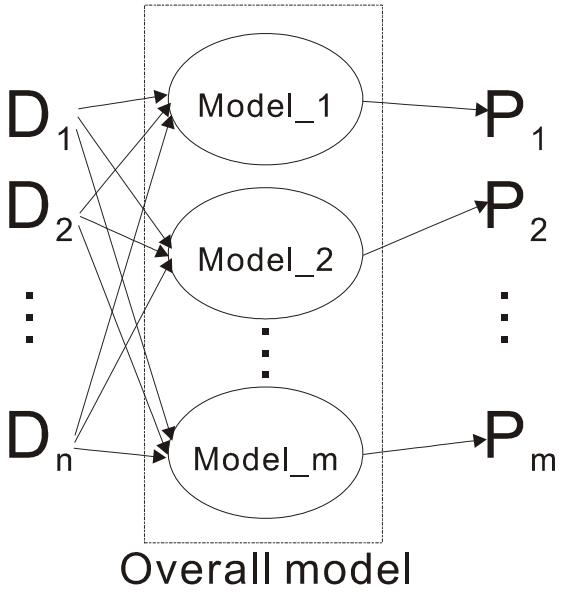


FIGURE 4.1: Structure of the overall model between design and performance parameters of AMS designs.

Performance parameters have complex relationships with design parameters in AMS designs and each design parameter influences different performance parameters with variant degrees. As shown in figure 4.1, the overall performance model of an AMS design is composed of several independent models, each of which corresponds to single relationship between one performance parameter and all the related design parameters. When all these models are combined, the overall performance model represents the relationship between the performance and design spaces.

The performance modelling methodology introduced in this chapter is to construct this overall performance model for the design and performance spaces of general AMS designs automatically for knowledge-based optimisation applications.

4.1.2 Performance modelling methodology

For one design, the overall linearly graded AMS performance models are composed of two kinds of models. For each performance parameter, the SVM classification models are used to model the boundaries between different grades; the SVM regression models are built for each subspace separated by the boundaries so that a numerical performance prediction can be provided at arbitrary design points in that subspace.

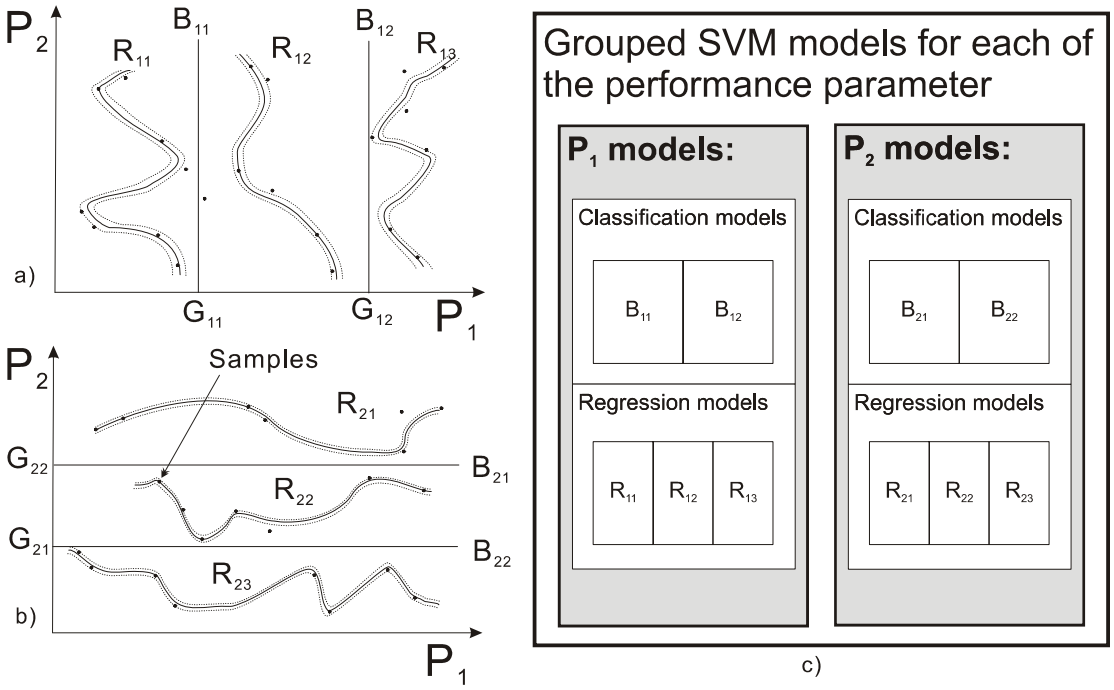


FIGURE 4.2: Illustrative example of the constructed linearly graded SVM classification and regression models in the two dimensional space formed by performance parameter P_1 and P_2. a) linearly graded performance models for performance parameter P_1 includes classification models for boundaries B_{11} and B_{12} and regression models R_{11}, R_{12}, R_{13}; b) linearly graded performance models for performance parameter P_2 with its corresponding classification and regression models; c) organisation of the classification and regression models for the two parameters P_1 and P_2.

An imaginary example is shown in figure 4.2, a) and b) to illustrate what the models really are. The distribution of the samples is a projection from the multi-dimensional performance space of a design to a two-dimensional performance space formed by the performance parameters P_1 and P_2. Subplot a) shows that the performance space is divided into three subspaces according to P_1 by two grades G_{11} and G_{12}, while b) shows the partitioning according to P_2 by grades G_{21} and G_{22}. As shown in the figure, each grade has six or seven or eight samples. The classification models, B_{11}, B_{12} for P_1 and B_{21}, B_{22} for P_2, will be constructed for the boundaries defined by the grades. Within each subspace, SVM regression models are constructed based on the distribution of the samples in the grades, i.e. R_{11}, R_{12} and R_{13} for P_1 and R_{21} to R_{23} for P_2. In the figure, each regression model R is represented by three curves. The central solid line represents the regression itself while the symmetric dashed lines drift away from the regression function by a tolerance distance. c) in the figure summarises the organisation of the models. For each of the performance parameter, P_1 and P_2, the linearly graded SVM models contain their own set of classification and regression models for each grades and subspaces.

4.2 Automated performance model construction

The model construction flow chart is shown in figure 4.3. The main operations including simulation, grading and SVM training are highlighted in grey rectangular in the middle. This chart has been implemented by algorithm 1.

The first element needed by the linearly graded performance modelling is a parameterised design with m design parameters D_i, $i \in 1 \cdots m$ and each has bounds assigned. The n performance parameters P_i, $i \in 1 \cdots n$ and their extraction functions need to be provided. Firstly, the design space is sampled to provide k design samples in the training data set. All the design samples are fed through a simulator, which can be behavioural-level or circuit-level. After simulations, k sets of performance vectors P_v of length n are extracted, either directly from simulations or by complex post-simulation result processing functions. Then one design sample D_v and its corresponding performance vector P_v form one training data sample T_v in the training data set. Applying the balanced data grading algorithm on the user-specified performance grades, a new set of grade vectors $G[1 : l]$ for

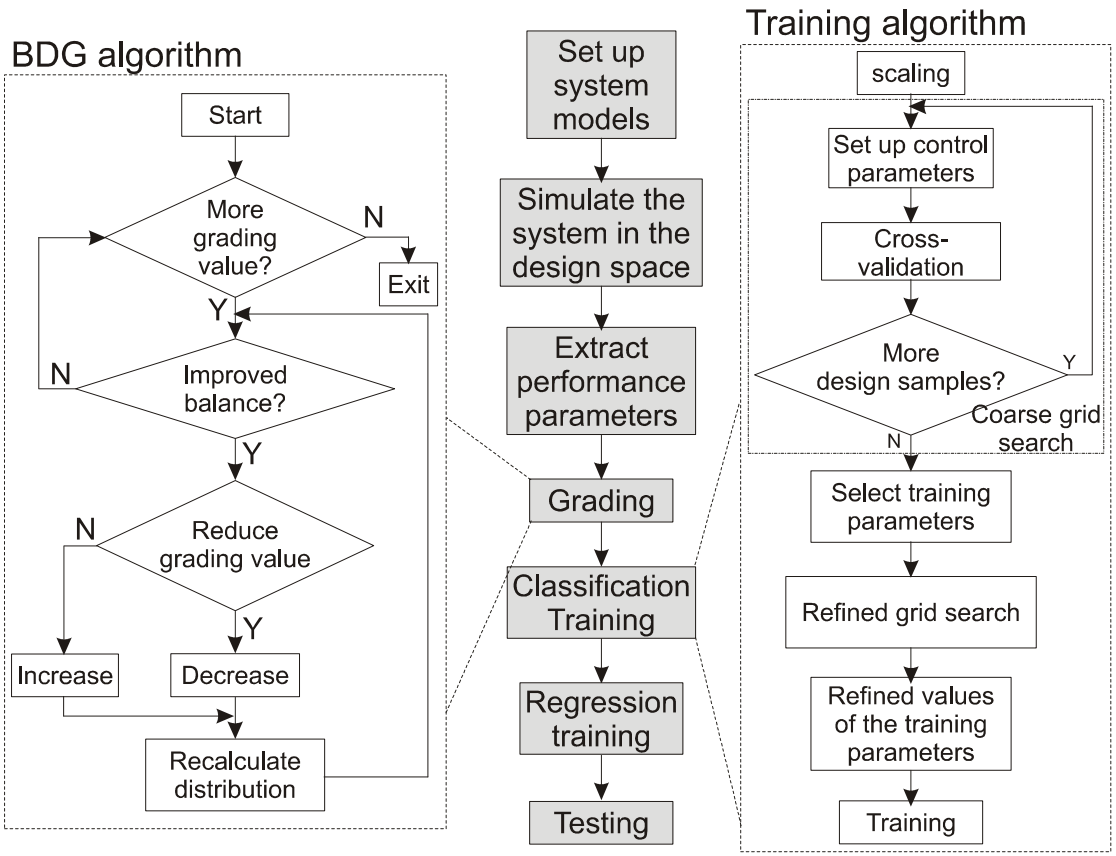


FIGURE 4.3: Flow chart of the model construction process including the BDG and the training algorithm details.

each performance parameter can be generated. The training data set is graded using the new grade vectors $G[1:l]$.

After these preparatory steps, the SVM classification and regression model construction processes commence. The classification model $MC_{i,j}$ for the i^{th} performance parameter P_i's j^{th} grade is constructed using the training data falling into the grade. Following this classification training step, a regression model $MR_{i,j}$ for P_i's j^{th} grade $G(j)$ is trained, which straightforwardly uses training samples in the grade. So, for each grade of each performance parameter, there is one SVM classification model and one regression model. The structure of the whole performance model is shown at the bottom of figure 4.2. These models are stored as the knowledge database in optimisation applications.

To achieve highest prediction accuracy, a few points have been considered. For each grade of the SVM classification training, the training samples in that grade are treated as one class while all other training samples are treated as another.

Algorithm 1 Linearly graded performance model construction algorithm.

Require: design parameters D_i, $i \in 1 \cdots m$;
Require: performance parameters P_i, $i \in 1 \cdots n$;
1: generate k design vectors $D_v = [D_1, \cdots, D_m]$ by sampling design parameters's ranges;
2: **for** $i = 1$ to k **do**
3: run simulation;
4: **for** $j = 1$ to n **do**
5: extract the performance parameter in the vector $P_v[j]$;
6: **end for**
7: form one training data sample $T_i = [D_v[i], P_v[i]]$;
8: **end for**
9: **for** $i = 1$ to m **do**
10: use BDG to generate grading vector $G[1 : l]$ for P_i;
11: grade the design space;
12: **end for**
13: **for** $i = 1$ to n **do**
14: **while** $j = 1$ to l **do**
15: construct SVM classification model $MC_{i,j}$ for P_i's $G(j)$;
16: construct SVM regression model $MR_{i,j}$ for P_i's $G(j)$;
17: **end while**
18: **end for**
19: **return** all MC, MR;

This can make full use of the data for the SVM trainer as all the samples in the design space are used for each grade's binary classification model construction.

To make the model construction fully automatic, it is very essential to develop the assistant algorithms to determine the grades and the SVM training parameters. The balanced data grading (BDG) algorithm (introduced in section 4.2.2) has been developed so that the generated grades can have the following property: the number of samples in each grade is similar. SVM training parameter determination algorithms and cross-validation techniques are used in the training process for the purpose of achieving good trade-off between the models' accuracy and generality.

Compared to other methods that have traditionally been accomplished by manual efforts [45–47], the salient feature of the methodology is that it is fully automatic and needs minimum human intervention, and is therefore labour saving. Apart from the main flow, the training parameter determination algorithm and the BDG algorithms are introduced in the following two sections.

4.2.1 Grid search algorithm

The grid search algorithm is used to determined the SVM training parameters based on the prediction accuracy. It is recommended as a general approach by the LibSVM creators [42]. The contribution of this research is to automate the process and integrate the algorithm in the performance modelling flow. The training parameters include the SRM trade-off parameter C and the kernel parameters, which is, in this research, γ of the RBF kernel. The algorithm has been implemented in MATLAB.

Considering that both SVM training parameters C and γ need to be scanned in exponential ranges, to enhance the search efficiency, the grid search contains two successive phases. In the first phase, the training parameters are searched with a loose resolution to locate an optimal range; then in a second stage, a detailed grid search with much higher resolution searches the optimal range only to find the optimum. These two phases are the so called *coarse grid search* (CGS) and *refined grid search* (RGS) phases.The process is illustrated in figure 4.4.

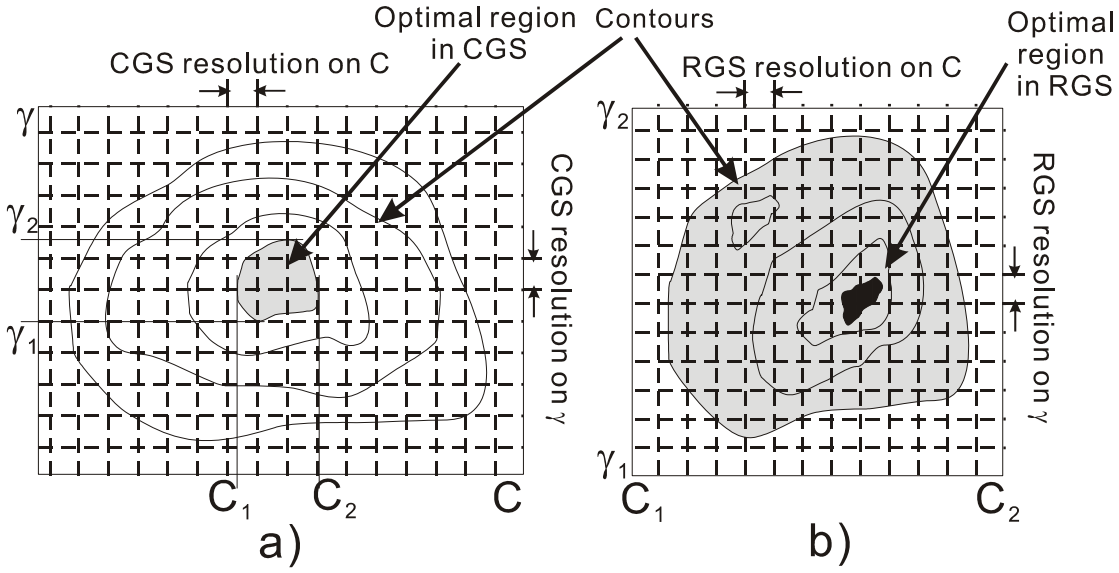


FIGURE 4.4: CGS and RGS in determining the SVM training parameters C and γ. a) optimal region in CGS defined by the lower left point $[C_1, \gamma_1]$ and upper right point $[C_2, \gamma_2]$; b) RGS re-scans the optimal region in higher resolution.

As shown in figure 4.3, in the right-hand column, this model training process starts from the scaling, which is the operation to scale all the training samples to the same interval using proportional mapping. The reason to do this is to eliminate the domination by large numerical values of one design parameter. The CGS and

RGS are then carried out. The training data is cross validated for each C and γ pair in the grid search for the maximum validation on the prediction accuracy and generality using the whole training data set. The cross validation process is to split the whole data set into several folds then testing the model constructed using one fold with other folds on every fold set of the data.

The CGS is outlined in algorithm 2. The algorithm requires the configuration parameters for the SVM trainer. Parameters ϵ and μ configure the SVM trainer as a μ SVM with tolerance ϵ for regression. V in the training command sets the number of folds of the cross validation, $kernelType$ has options for the four kernel functions introduced in section 3.1.5. The lower and upper bounds of the training parameters C and γ are required. In the CGS, C and γ are sampled n and m times exponentially so that $n \cdot m$ pairs of C-γ are created to form the grids. Each pair is fed to the SVM trainer and finally the optimal prediction accuracy A_{opt} and the corresponding optimal parameters C_{opt} and γ_{opt} are output.

Algorithm 2 CGS phase of the standard grid search algorithm for SVM training parameter determination.

Require: SVM trainer setting ϵ, μ;
Require: SVM trainer setting V;
Require: SVM trainer setting $kernelType$;
Require: $C = [C_{lb}, C_{ub}]$, $\gamma = [\gamma_{lb}, \gamma_{ub}]$;
generate C and γ vectors $V_C[1:n]$ and $V_\gamma[1:m]$;
define A_{opt}, C_{opt}, γ_{opt};
for $i = 1$ to m **do**
    for $j = 1$ to n **do**
        involve the SVM trainer with all the setting parameters;
        post processing the resulting file for prediction accuracy A_{temp};
        if $A_{tmep} > A_{opt}$ **then**
            $A_{opt} = A_{temp}$;
            $C_{opt} = V_C[j]$;
            $\gamma_{opt} = V_\gamma[i]$;
        end if
        record the results;
    end for
end for
return C_{opt}, γ_{opt}, A_{opt};

This algorithm can be easily modified for the RGS algorithm. The difference is that the C_{opt} and γ_{opt} found in the CGS are used used as the starting point on

which the RGS sampling is centred on. All other parts are identical to algorithm 2.

4.2.2 Heuristic grading algorithm

The new heuristic grading vector generation algorithm, the BDG, is to modify user-specified grading vectors to obtain a new set of vectors that can divide the training data set into several subsets according to their performances, and each subset has a similar number of design samples in it. The reason to have similar amounts of training samples for each grade classification and regression model construction is to avoid the situation wherein one grade has too many samples or too few samples. These situations can easily create overtrained models. The concept is explained by figure 4.5. The distribution using the original grading vector $[V_1, V_2]$ can generate overtrained models while the new vector V_3, V_4 can avoid the difficulty. The comparison of the distribution on the left in the figure clearly shows the effect of the algorithm.

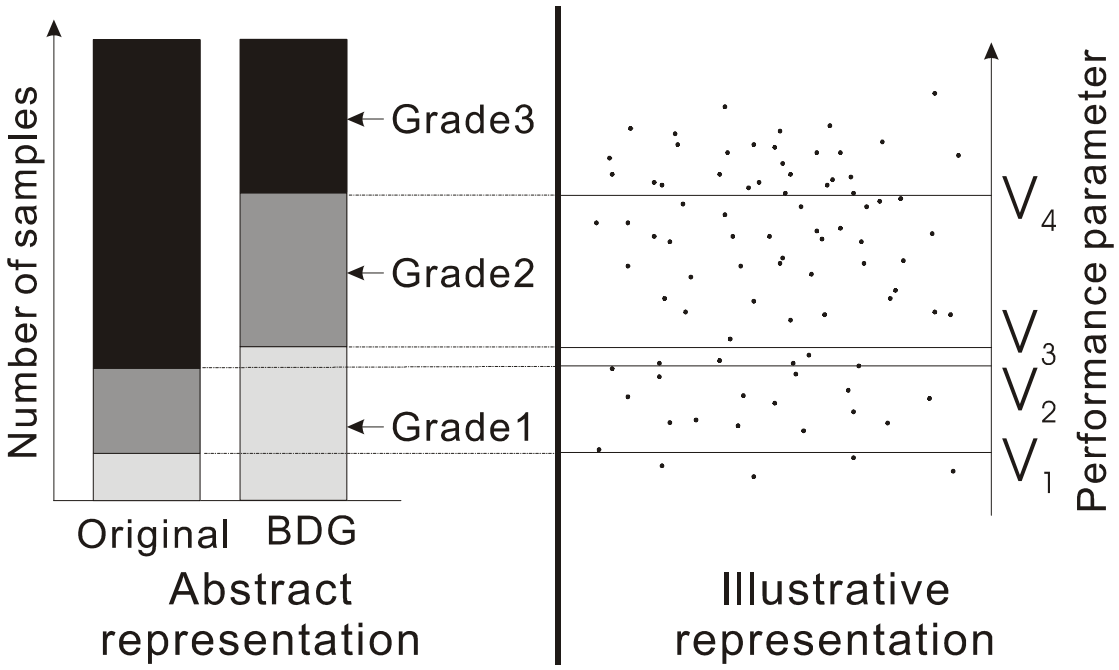


FIGURE 4.5: Illustrative representation is the distribution of the samples according to a performance parameter. Abstract representation is the number of samples. V_1 and V_2 are the original grades; V_3 and V_4 are the new values calculated by the BDG algorithm.

Users specify an initial set of grading elements to the performance parameters according to their experience. As shown in figure 4.3, in the left-hand column, the heuristic algorithm tries to increase or decrease the values of the elements in the grading vector. If a more balanced grading result can be achieved, the algorithm continues the increment or decrement, otherwise it modifies the value in an opposite way until best balance is reached. When this is finished on every element in the grading vector, the process terminates and returns the modified grading vector.

Algorithm 3 BDG algorithm for the generation of new grading elements.

Require: original grading vector $G[1:m]$;
Require: step size s;
Require: direction d;
number of grades $N = length(G)$;
count the total number of samples N_t;
calculate $N_{ini}[1:m]$ using $G[1:m]$;
calculate ideal $N_g = \frac{N_t}{m}$ for each grade;
calculate $N_{tol} = N_g \cdot s$ for each grade;
calculate sample difference $N_d = N_{ini} - N_g$ for E_1;
define temporary difference of number of samples N_{temp} in the adjustment of grading elements;
while $i < m$ **do**
    while $abs(N_{temp}) \leq abs(N_d)$ and $N_{temp} * N_d > 0$ **do**
        $N_d = N_{temp}$;
        if xor(d, N_d)=**false then**
            $G(i) = G(i) + s$;
        else
            $G(i) = G(i) - s$;
        end if
        recalculation the distribution;
        calculate new sample difference N_d;
    end while
    $i = i + 1$;
    calculate new overall distribution;
    update N_d, N_{temp};
end while
return $G[1:m]$;

The process has been implemented in algorithm 3. The original grading vector G with m elements, together with the step size s and direction character d, is provided for a performance parameter. Using G, the original distribution N_{ini} is calculated. The ideal distribution N_g with minimum difference on the number

of samples is derived. The acceptable tolerance for each grade $N_{tol} = N_g \cdot s$ can be obtain by multiplying the ideal distribution and the modification step size. From the first element E_1 in G, every element is modified according to the current sample difference N_d and the direction d feature of the performance parameter using an $xor(d, N_d)$ operator. N_{temp} is used to record the previous number of samples difference. The modification process terminates when the current number of samples difference N_d is larger than the previous one recorded in N_{temp}. The new grading vector G is returned as the output.

4.3 Case study: a 2^{nd} order lowpass analogue filter

As a preliminary example to verify the concept of the linearly graded AMS performance modelling methodology, a 2^{nd} order lowpass analogue filter has been used to construct the SVM classification models of ten grades for each of the three selected performance parameters. The number of grades has been determined randomly as it has no effect on the conceptual proof.

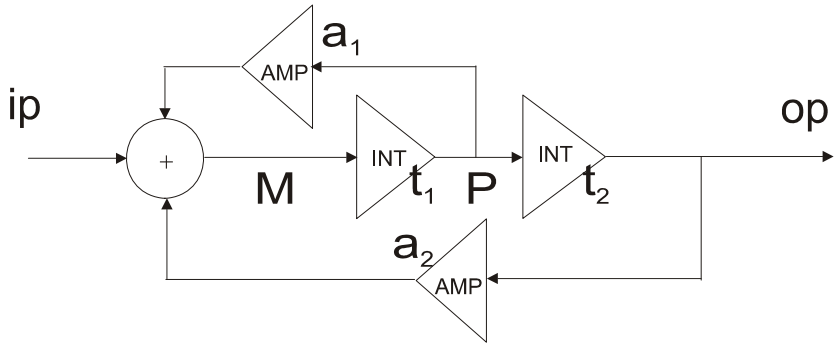


FIGURE 4.6: Signal flow graph of the 2^{nd} order analogue filter.

The signal flow graph of the analogue filter is in figure 4.6. In the figure, terminal "ip" and "op" represent the input and output; "M" and "P" are internal connections. Four design parameters a_1, a_2, t_1, t_2 have been attached to the corresponding components. a_1 and a_2 are the feedback coefficients; t_1 and t_2 are the time constants of the integrators.

The design has been implemented in VHDL-AMS. The four design parameters are defined as the generics in the filter's entity. The connections in the filter are

defined as quantities in the architecture. The behaviour of the system is described by three simultaneous statements for the DAEs as the follows:

$$V_M = V_{ip} + a_1 \cdot V_p + a_2 \cdot V_{op} \quad (4.1)$$

$$V_P^{'} = t_1 \cdot V_M \quad (4.2)$$

$$V_{op}^{'} = t_2 \cdot V_P \quad (4.3)$$

where V_M, V_{ip}, V_P and V_{op} are the voltages at the corresponding nodes, the $V^{'}$ represents the differential operation of the nodal voltage.

A MATLAB based VHDL-AMS simulator, called 'SAMAS' [117], has been used for simulations. The main advantage of using this tool is that it can be easily integrated in the performance model construction process without much overhead dealing with graphical interfaces. However, it is noted that the netlists for design and analysis are separated in the tool. In one simulation, a design netlist is firstly complied and loaded then an analysis netlist file is loaded and applied.

4.3.1 Linearly graded SVM classification model construction

The four design parameters form the design space. The performance space is formed by three performance parameters: gain (G), cut-off frequency (F_c), and over-shoot (R). The goal of this case study is to model the ten grades of each performance parameter in the relationships with the four dimensional design space. Using the linearly graded performance modelling methodology introduced in chapter 4, a training data set is generated and graded for each of the performance parameters. Figure 4.7 shows the performance of the training data samples projected onto three-dimensional spaces that the other two dimensions are two design parameters. The distribution of the samples clearly indicates the nonlinear relationship between the design and performance spaces.

The standard grid search training parameter determination algorithm and cross-validation technique have been used for the model construction. The cross-validation accuracies for each grade are summarised in table 4.1.

For the G parameter, high training accuracies have been achieved for all the grades. For the other two parameters F_c and R, most grades' classification model training

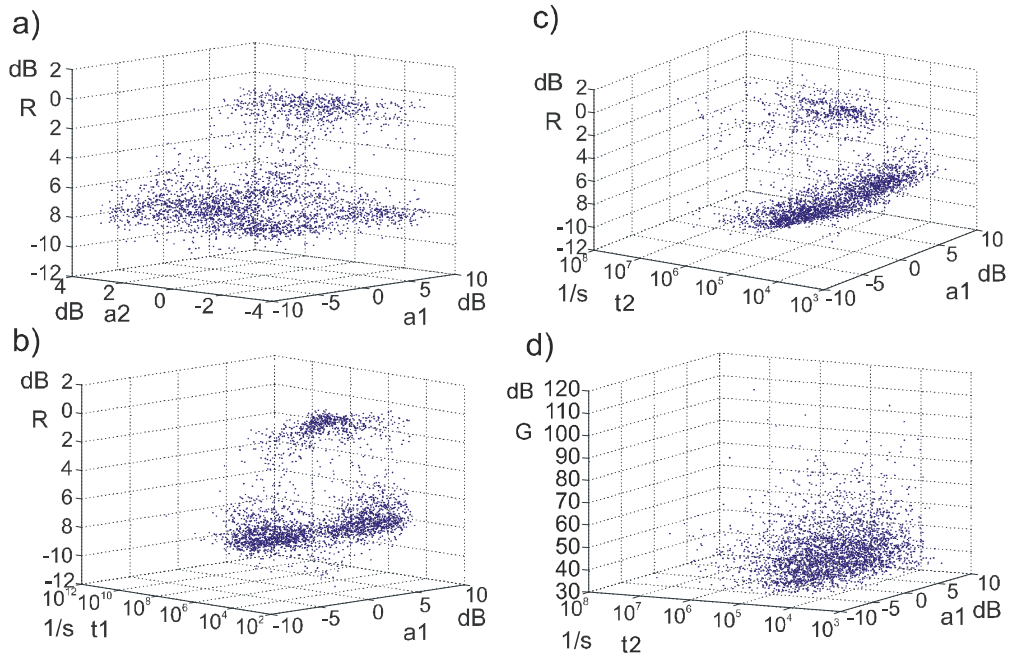


FIGURE 4.7: Sample scatter diagrams showing projection of the performance space onto the 3-dimensional spaces that reflect the relationship between overshoot (R) and the four design parameters. a) $R(a_1, a_2)$, b) $R(a_1, t_1)$, c) $R(a_1, t_2)$, d) $G(a_1, t_2)$.

TABLE 4.1: Cross-validation accuracies of the RGS phase for the ten grades (nine boundaries) of the performance parameters.

Grades	F_c	G	R
1	99.84	99.9	93.38
2	99.39	99.93	90.77
3	98.38	99.91	90.41
4	95.22	99.87	88.25
5	83.86	99.89	84.45
6	79.76	99.8	74.35
7	96.93	99.79	98.52
8	99.49	99.94	99.76
9	99.94	99.99	99.98

accuracies are high. One or two models have comparably low accuracies. This is because the grading in this example is uniform and considers no distribution of the samples in the design space. So the grades that separate high design sample density areas in the design space can have more infeasible errors.

The accuracy contours of the CGS and RGS in the standard grid search training

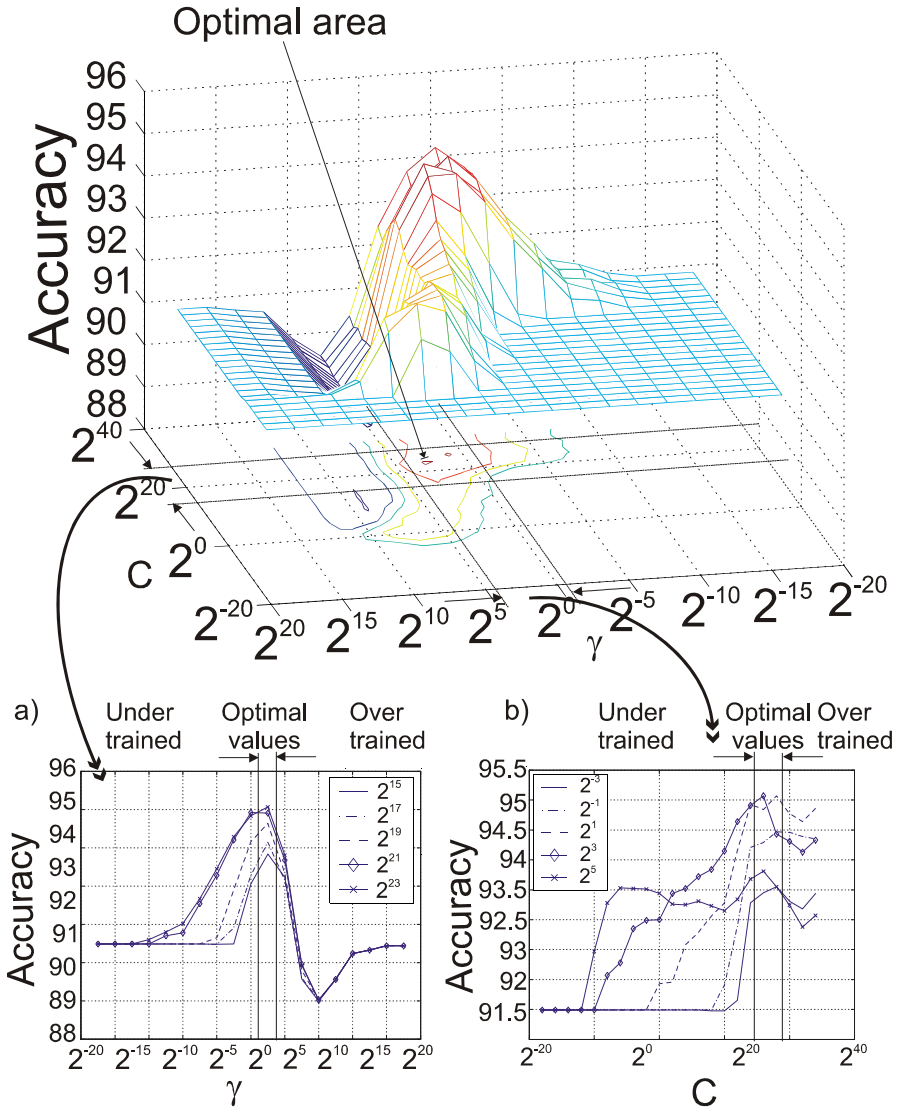


FIGURE 4.8: Top plot is a 3D classification accuracy diagram showing the CGS of the model construction for the boundary $F_c = 4$. Sub-plots a) and b) are projections onto 2D planes showing optimal, under-trained and over-trained regions. a) shows γ vs. classification accuracy. b) shows C vs. classification accuracy.

parameter determination algorithm are presented. Figure 4.8 and 4.9 show an example pair of the CGS and RGS results for the grade 4 boundary of the cut-off frequency F_c.

In figure 4.8, C takes a two based exponential increment from 2^{-19} to 2^{31} while γ has the range from 2^{-29} to 2^{19}. The top plot clearly shows an optimum region centred at $C = 2^{20}$ and $\gamma = 2$. This is the optimal pair for C and γ achieved in

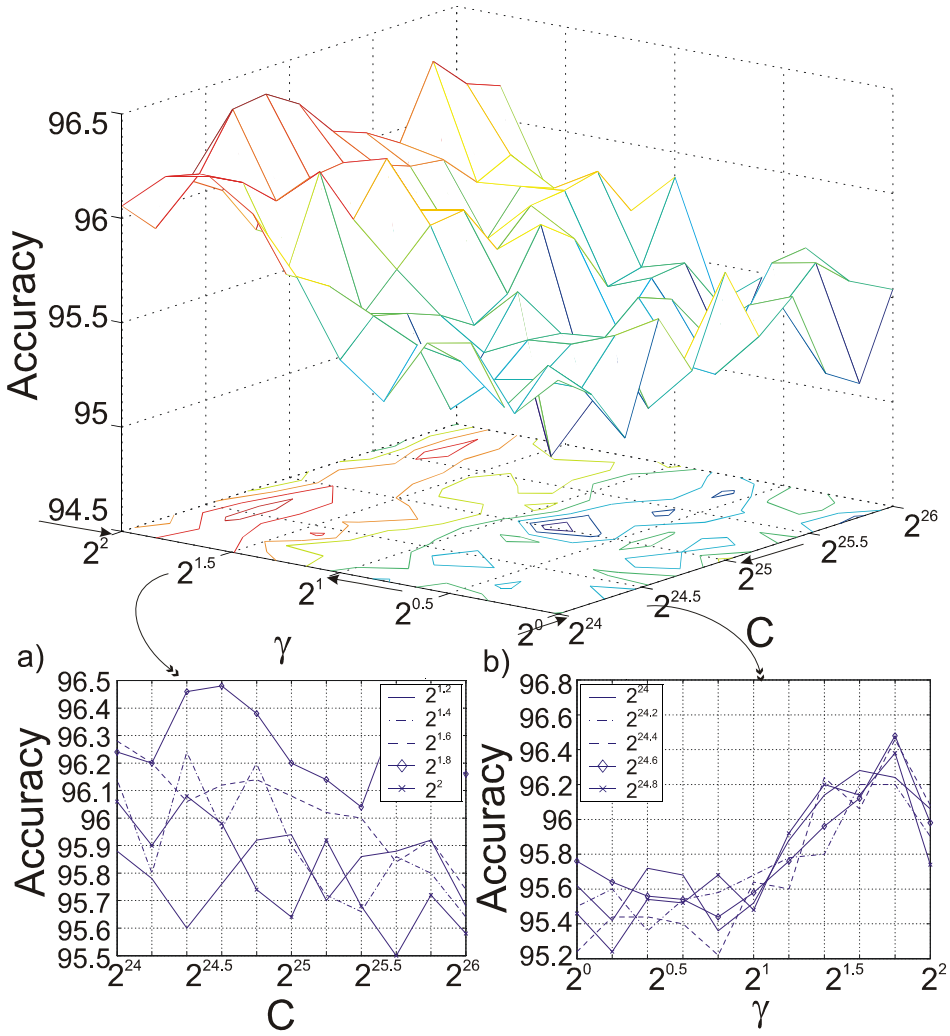


FIGURE 4.9: 3D prediction accuracy diagram with contours showing the RGS searching the optimal area labelled in figure 4.8.

.

the CGS and it is used at the central values for these two parameters' scan in the RGS.

The RGS accuracy surface around the optimum C-γ pair is shown in figure 4.9. A better pair of C and γ with higher accuracy can be found at $C = 25.4 \times 10^6$ and $\gamma = 3.48$ with about 2% prediction accuracy improvement. As commonly seen in supervised learning, the undertrained and overtrained models with lower accuracies than the optimal have been observed in the experiments as shown at the bottom in figure 4.8. The ranges have been labelled in the subplots.

Figure 4.10 shows a sample performance space of the performance parameter R projected onto a two-dimensional space, which is formed by normalised a_1 and a_2

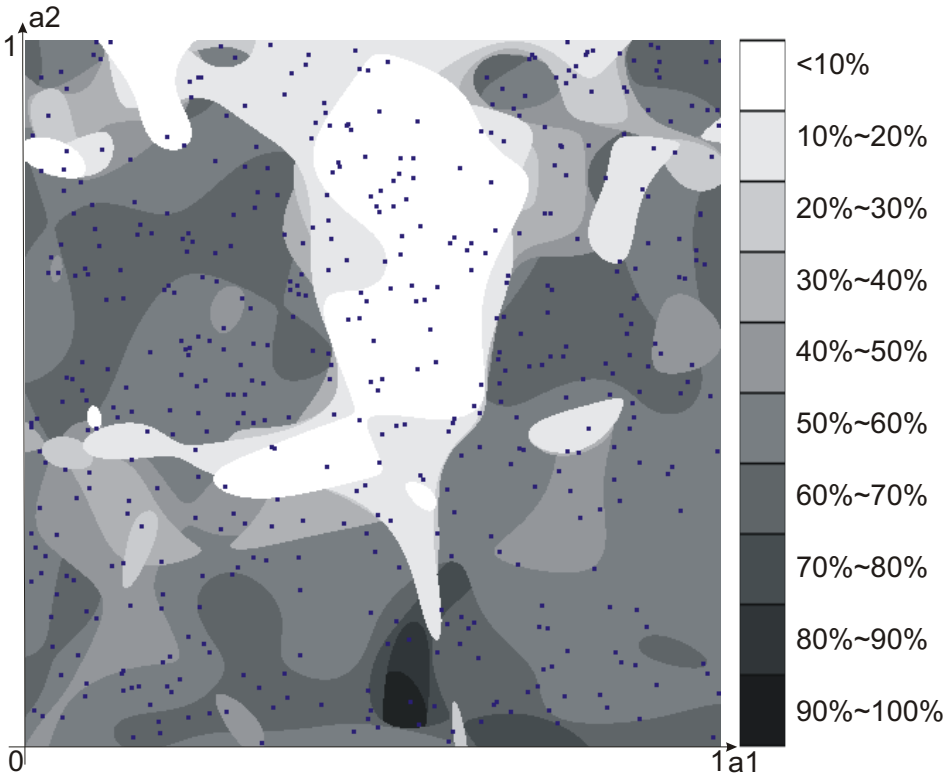


FIGURE 4.10: Graded sub-spaces showing the divided performance space of parameter $R(a_1, a_2)$.

in $[0, 1]$. Every grade of R corresponds to one grey level in the figure. Support vector points are observed on the boundaries between different grades. The figure shows clearly the main advantage of the linearly graded models where relationships between the design and performance parameters provides more potential for accurate modelling of analogue systems than those of the "good-bad" approaches.

4.4 Concluding remarks

The development of the linearly graded AMS performance model construction methodology is to extend the classical 'good-bad' binary classification models so that designers can have more insight into the behaviour of the design and be more capable of exploring the design space. This methodology has been introduced in this chapter. The algorithms utilised for the automated model construction process have been illustrated. This includes an SVM training parameter determination algorithm as well as the BDG algorithm for training data grading. Apart from the conceptual examples, the proposed methodology has been illustrated by an

analogue filter example. The SVM training accuracy contours using the CGS and RGS training parameter determination algorithm have been shown for the design. An example graded performance space has also been included to show how the new methodology can provide better design space exploration than the classical method. A more complex demonstration using the 2^{nd} order SDM can be found in chapter 7.

Chapter 5

Computational cost aware automatic generation of SVM regression performance models

This chapter includes the introduction of the new AMS performance modelling methodology using SVM regression technique and describes an efficient, computational cost aware algorithm for SVM training parameter determination. The computational cost of the training parameter determination process is analysed in terms of CPU time in section 5.1. The analysis is based on the grid search algorithm introduced in the previous chapter. Influence of different training parameters on the computational cost is presented separately. Based on observations in a number of experiments, an abstract computational cost model of the grid search algorithm and the corresponding accuracy model are proposed. A measure of the effectiveness in SVM training parameter determination is introduced and a new, computational-cost aware training parameter determination algorithm is developed and presented in section 5.2. In section 5.3, the SVM regression performance model construction process with the new algorithm is explained in detail.

5.1 Computational cost analysis of SVM model construction

Since the introduction of the SVM technique to AMS performance modelling, the huge computational cost related with performance model construction has been ignored by the state-of-the-art researches [45–47]. This is mainly because of the lack of automated implementation of the training parameter determination and the need to use an empirical process for this important task. However, this research has already developed an automated SVM performance model construction approach[49]. The grid search algorithm has been applied but is extremely computationally expensive, e.g. for a 2^{nd} order SDM design, the whole performance model construction process can take more than 100 hours. Thus, to employ the SVM method for practical AMS designs, it is of great importance to study the computational cost of the grid search algorithm, and very essential to develop new algorithms for highly efficient training parameter determination approach. This technique is key to the development of the knowledge based SVM optimisation methodology presented here.

The computational cost analysis outlined below has been carried out using LibSVM, a popular SVM trainer [46, 47, 49, 118, 119]. The grid search algorithm is used to generate the computational cost contours because it is virtually an exhaustive searching method. In all the experiments, the RBF kernel has been selected. So there are two training parameters to be determined: C and γ. As introduced in chapter 3, C is the SRM optimisation trade-off parameter between the accuracy and model generality; γ is the RBF kernel parameter that determines the nonlinearity of the kernel function.

5.1.1 C and γ in solving the SVM problems

For each training parameter C-γ pair, LibSVM solves the SVM optimisation equation 3.39 on a selected subset, called the *working set*, of the training data set in the feature space using Lagrangian optimisation. This solving process is composed of two phases. Firstly, before the iteration starts, design samples in the training data set are mapped to the feature space by involving the kernel function. Then, a working set is iteratively selected from the training data set based on the gradients

at the mapped sample points in the feature space. The Lagrangian coefficients of the working set, which lie in the range [0,C], are found by solving the optimisation problem defined by equations 3.39 and its subjects. So it is clear that γ is involved only at the kernel mapping while C is in the iterative optimisation solving.

Parameter C has influence on the computational cost as it determines the width of the intervals where the Lagrangian coefficients lie. The working set selection is dependent on the gradients of the training samples in the feature space which in turn determines the convergence speed of the solving process. γ, as the parameter that defines the nonlinearity of the kernel function, can influence the computational cost by affecting the working set selection. The characteristics of the influences are the main concerns for the development of new algorithms.

5.1.2 Influence of C on computational cost

The influence of parameter C on the computational cost, as shown in figure 5.1, can vary significantly in the scan of C. It can be extremely expensive, especially when C is large and γ is optimum in the grid search. The data set is obtained from an online database [124].

Not only the data set shown in figure 5.1 has a clear feature in the scanning of C, the observations of the experiments in chapter 7 all support the following generalised description that the computational cost can be characterised by two ranges on the dependence of C: insensitive and sensitive ranges. As shown in figure 5.2, there is a breaking point at C_0, where the computational cost curve becomes sensitive to the change of C. The computational cost remains at constant t_0 in the insensitive range. While in the sensitive range, the computational cost shows a strong linear dependence on $log_2 C$.

The following empirical expression is proposed for a general description of the influence of C:

$$T = logt = \begin{cases} t_0 & 0 < C < C_0 \\ t_0 + k[log(C) - logC_0] & C >= C_0 \end{cases} \tag{5.1}$$

where t is the computational cost, T is its log scale, t_0 is the constant computational cost in the insensitive range, k is the slope of the linear dependence of

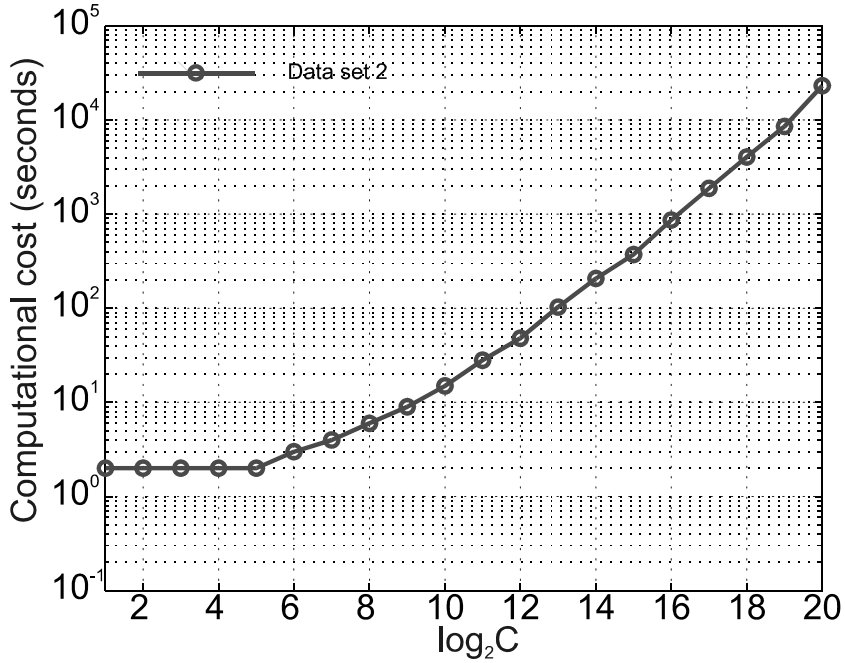


FIGURE 5.1: The computational cost influenced by parameter C in an SVM training parameter determination experiment using the data set from a database [124].

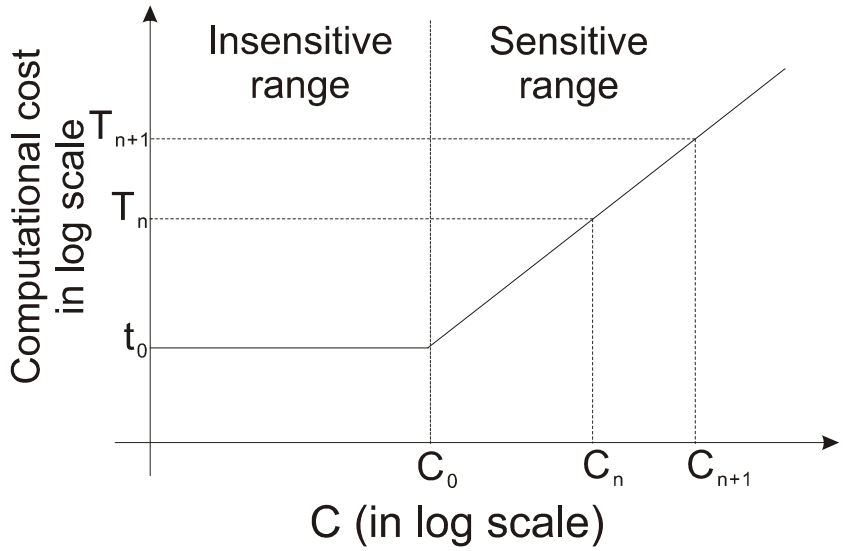


FIGURE 5.2: Computational cost model in the scan of parameter C.

computational cost on C in the sensitive range and C_0 is the breaking point value of C. Using this equation, the relationship of the T_{n+1} and T_n on points C_{n+1} and C_n in the grid search with fixed γ can be derived as the following:

$$\frac{T_{n+1}}{T_n} = \frac{t_0 + k[log(C_{n+1}) - logC_0]}{t_0 + k[log(C_n) - logC_0]} \tag{5.2}$$

Assuming the increment factor of C in the grid search is r, i.e. $C_{n+1} = rC_n$, equation 5.2 thus becomes:

$$\frac{T_{n+1}}{T_n} = \frac{t_0 + k(log(r) + log(\frac{C_n}{C_0}))}{t_0 + k \cdot log\frac{C_n}{C_0}} \tag{5.3}$$

The equation can be revised as:

$$\frac{T_{n+1}}{T_n} = 1 + \frac{klog(r)}{t_0 + klog\left(\frac{C_n}{C_0}\right)} \tag{5.4}$$

So the relationship between the two computational costs in normal time scale is:

$$t_{n+1} = t_n^d \quad and \quad d = \left(1 + \frac{klog(r)}{t_0 + klog\frac{C_n}{C_0}}\right) \tag{5.5}$$

where d is the exponent of the computational cost factor. This equation can be used to deduce the following equation:

$$t_{n+1} = t_0^d \quad and \quad d = \prod_{i=0}^{n}\left(1 + \frac{klog(r)}{t_0 + klog\frac{C_i}{C_0}}\right) \tag{5.6}$$

Equation 5.6 indicates that the exponents of previous Cs accumulate when the current C value is scanned in the grid search. As the scanning range of C varies widely, the accumulation creates a significant expansion of the computational cost. It was observed when analysing the case studies described in 7 that when C is large, computational cost becomes extremely high, up to several hours of CPU time for just one point of C. In many cases, this means that the difference between the insensitive range constant t_0 and the high computational cost can be up to three orders of magnitude. So it is important to consider the computational cost as a critical factor in the development of the SVM training parameter determination algorithm.

5.1.3 Influence of γ on computational cost

In the analysis of the computational cost influenced by parameter γ, again, an interesting pattern shows up. As in figure 5.3, generally, the computational cost in the scan of γ has a sectioned character.

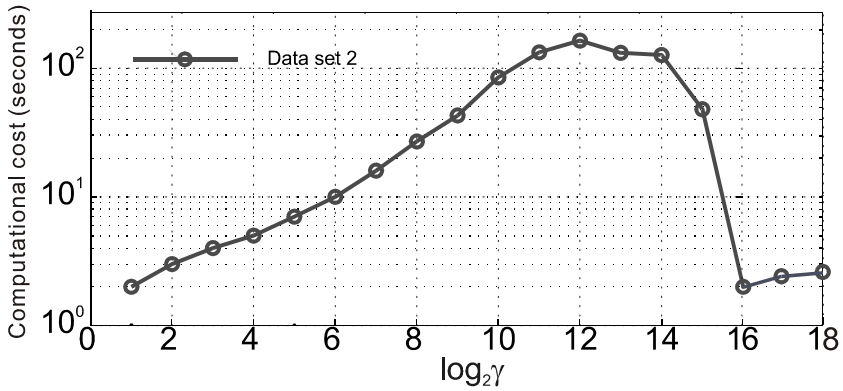


FIGURE 5.3: The computational cost influenced by parameter γ in the same SVM training parameter determination experiment as previously. The data set is still the one from the database [124].

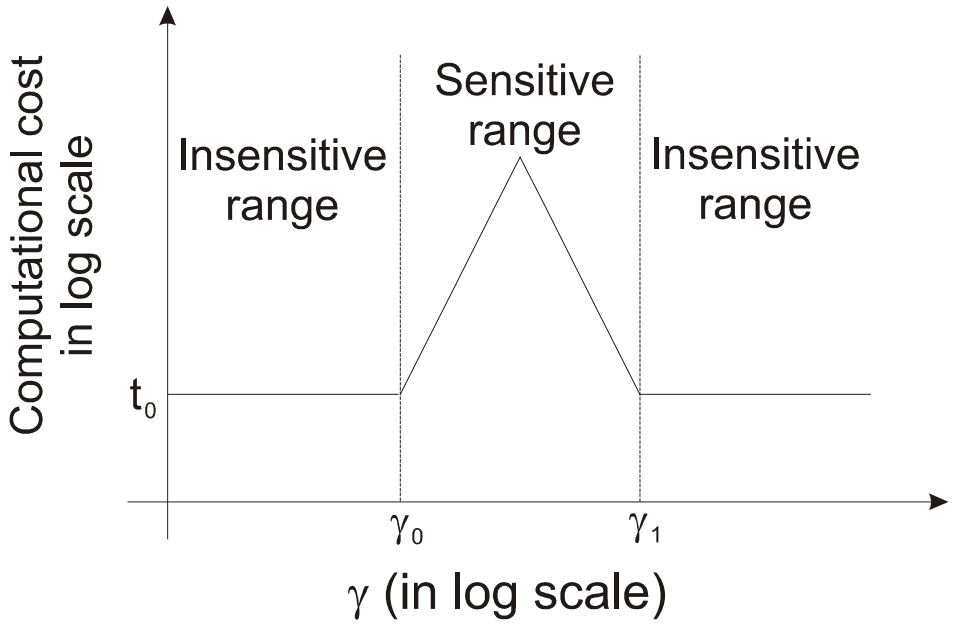


FIGURE 5.4: Computational cost model in the scanning of parameter γ.

Based on the observations in the experiments, the character of the computational cost can be generalised as an abstract nonlinear relationship to γ when C is fixed. As shown in figure 5.4, three ranges present, including two insensitive ranges when $\gamma < \gamma_0$ and $\gamma > \gamma_1$. The nearly constant t_0 is not necessarily the same for the two

insensitive ranges. Between these two, when $\gamma_0 < \gamma < \gamma_1$, is the sensitive range where a peak computational cost presents which usually corresponds to accuracy optimal or near optimal values of γ.

Although it is difficult to express the exact relationship in an explicit form, the influence of γ on the computational cost can be significant, especially at some points in the sensitive range. The magnitude can be as much as nearly up to three orders larger than the insensitive ranges as shown in the case studies.

5.1.4 Computational cost and prediction accuracy

SVM method belongs to the supervised learning category [35] that creates a function from training data. The model's prediction accuracy and its generality is a well-known trade-off in the training process determined by the training parameters. The training error can always be reduced by constructing more and more specific models for the specific training data set; however, when using testing samples to verify the models, the more specific the model is the more error it will generate because of its low generality on unknown data. This phenomenon is illustrated in figure 5.5 and can be used on both C and γ for the explanation of the prediction accuracy curves as shown in section 5.1.4.1.

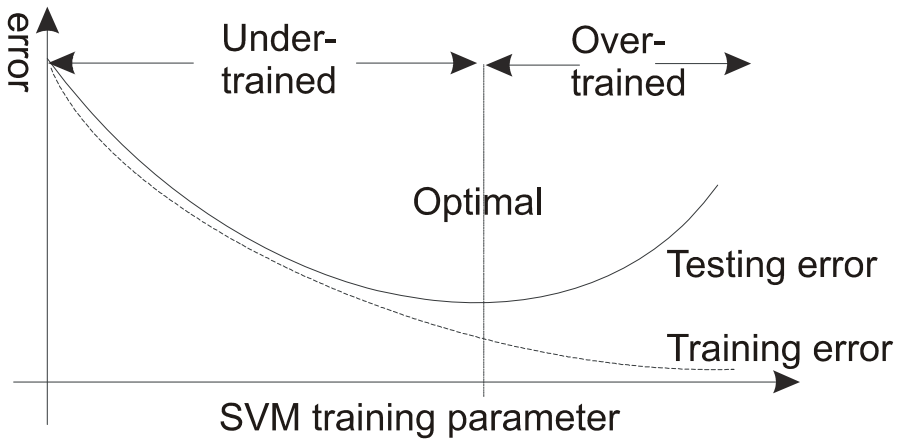


FIGURE 5.5: Under and over training in supervised learning [86].

Because of this, the prediction accuracy in a two-dimensional space formed by C and γ in the grid search thus has an optimal range corresponding to the optimal ranges of C and γ accordingly. Based on the observations in the case studies and additional experiments in chapter 7, this can be illustrated as the model shown in

figure 5.6 b). The optimum region has been highlighted by the grey shade. The prediction accuracies around the optimal area correspond to low accuracy models.

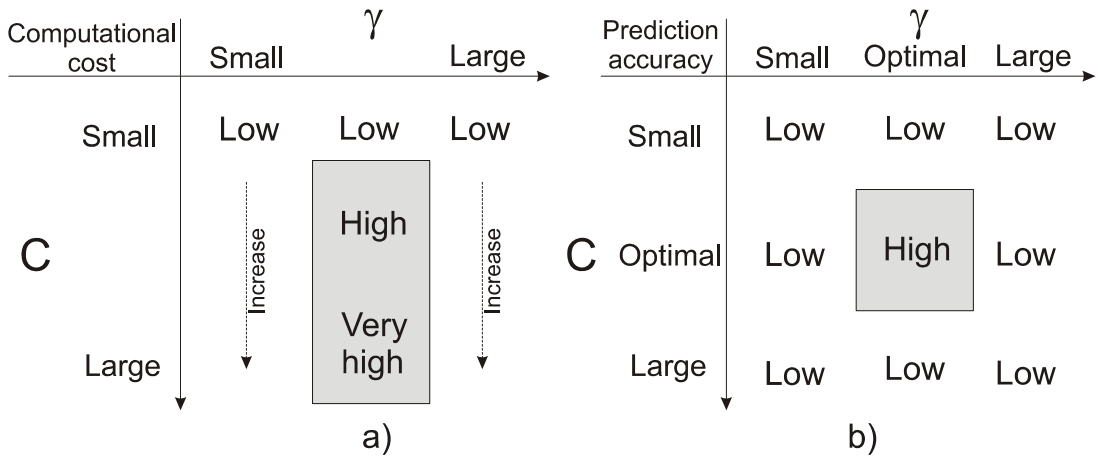


FIGURE 5.6: a) empirical computational cost model of C-γ grid-search process; b) the corresponding accuracy model.

In the same two-dimensional space, the computational cost has different characteristics. First of all, when C is small (smaller than the breaking point in figure 5.2), the computational cost is low. Secondly, high computational cost areas start to present at the C-γ values that correspond to the sensitive range of γ when C is larger than the breaking point. The influence of γ is combined with the high computational cost range of C and makes the computational cost at these areas very high. The computationally expensive areas have been highlighted by the grey rectangle in figure 5.6 a). For practical values of C and γ, the areas outside the grey rectangle can be considered as low computational cost.

The significance of the models in figure 5.6 is that they indicate the high computational cost area in the grid search is not necessarily overlapped to the high prediction accuracy area as observed in the case studies and additional experiments. In most of the cases, the most computational-cost-expensive areas have no overlapping to the high accuracy area thus waste resources and could not contribute to find the optimal C-γ values for high accuracy models. Therefore, the standard grid search algorithm is not the most economical solution for SVM training parameter determination. In addition, because the computational cost in the grid search is so high that unless special care is taken in the algorithm to avoid the 'very high' computational cost regions (figure 5.6 a)), finding the optimal C-γ pair for SVM training can hardly be effective.

5.1.4.1 Computational cost analysis in the case studies

In this section, the analysis of the computational cost in the case studies is carried out to verify the proposed modelling technique. Details of the design of these case studies can be found in chapter 7. In the 2^{nd} order SDM case study, the influence of C on the computational cost of training parameter determination and prediction accuracy for performance parameter SNR is shown in figure 5.7. As shown in figure 5.7 a), the insensitive and sensitive ranges clearly present and are separated by the breaking point at $log_2C_0 = 3$. The linear relationship between the computational cost and log_2C is as described in the proposed model. Also, as figure 5.7 c) shows, the influence of parameter γ on the computational

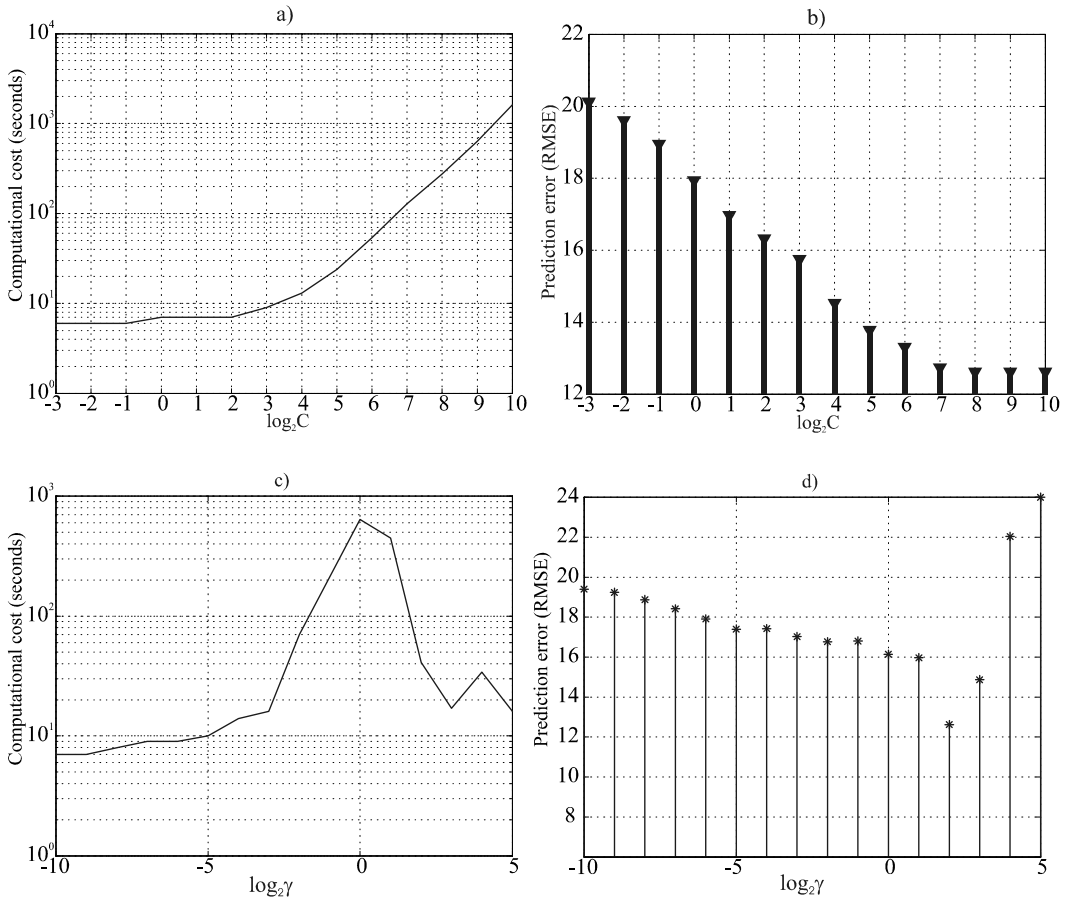


FIGURE 5.7: Computational cost influenced by the training parameters C and γ for the SNR of the 2^{nd} order SDM. a) the influence of C on the computational cost when γ=0.5; b) the corresponding prediction accuracy plot; c) and d) are the computational cost and prediction accuracy diagrams when C is a constant while γ is scanned.

cost presents a highly nonlinear feature and the sensitive range is very noticeable

between $\gamma_0 = log_2 - 3$ and $\gamma_1 = log_2 3$. More importantly, the computational cost varies by almost three orders of magnitude in the scan of C and γ. This indicates how expensive the computational cost will be when the grid search covers high computational cost C and γ pairs. For the prediction accuracy, both of the subplots b) and d) in figure 5.7 follow the same pattern, that in the grid search, the cross validation errors decrease in the under-trained range then increase in the over-trained range.

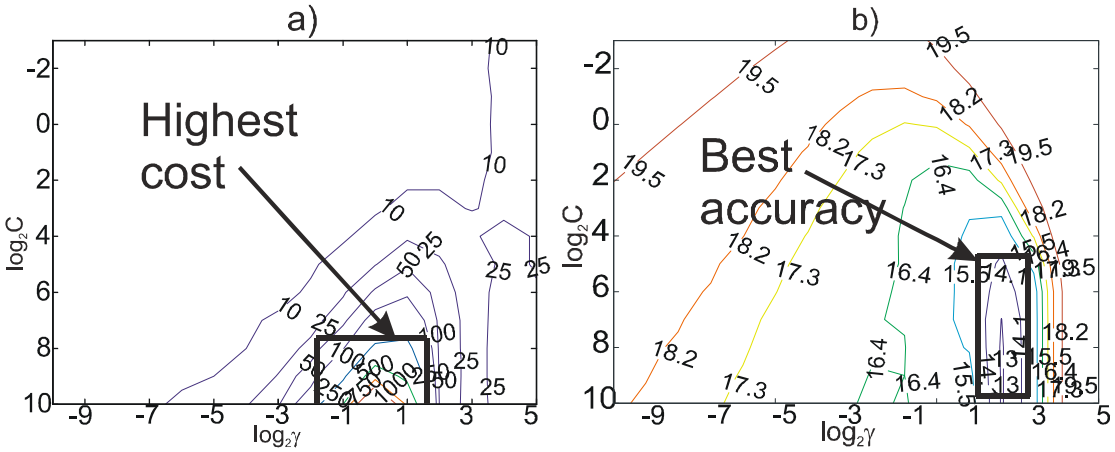


FIGURE 5.8: a) Computational cost contours (in seconds) of C-γ grid search process for the SNR of the 2^{nd} SDM, b) corresponding accuracy performance of the models (RMS error in dB).

The combined computational cost contours of the complete grid search process in the 2^{nd} order SDM case study are shown in figure 5.8 a). The high computational cost area concentrates in a small region that corresponds to the values in the sensitive ranges of C and γ. Although there are not many pairs of C and γ in this region, each pair needs such a long time that the computational cost for all the pairs in this region can take over 90% of the overall computational cost for the whole grid search. The corresponding prediction accuracy diagram is in figure 5.8 b). It is obvious that the high computational cost region and high accuracy region are only partially overlapped, and some of the high computational cost areas are outside the high accuracy C-γ region, which means that a lot of computational resources have been wasted. This also indicates that highly accurate model construction can be achieved without scanning the whole high computational cost area. It is especially unnecessary to scan the very high computational cost C-γ pairs corresponding to the over trained performance models that are not helpful to find optimal values.

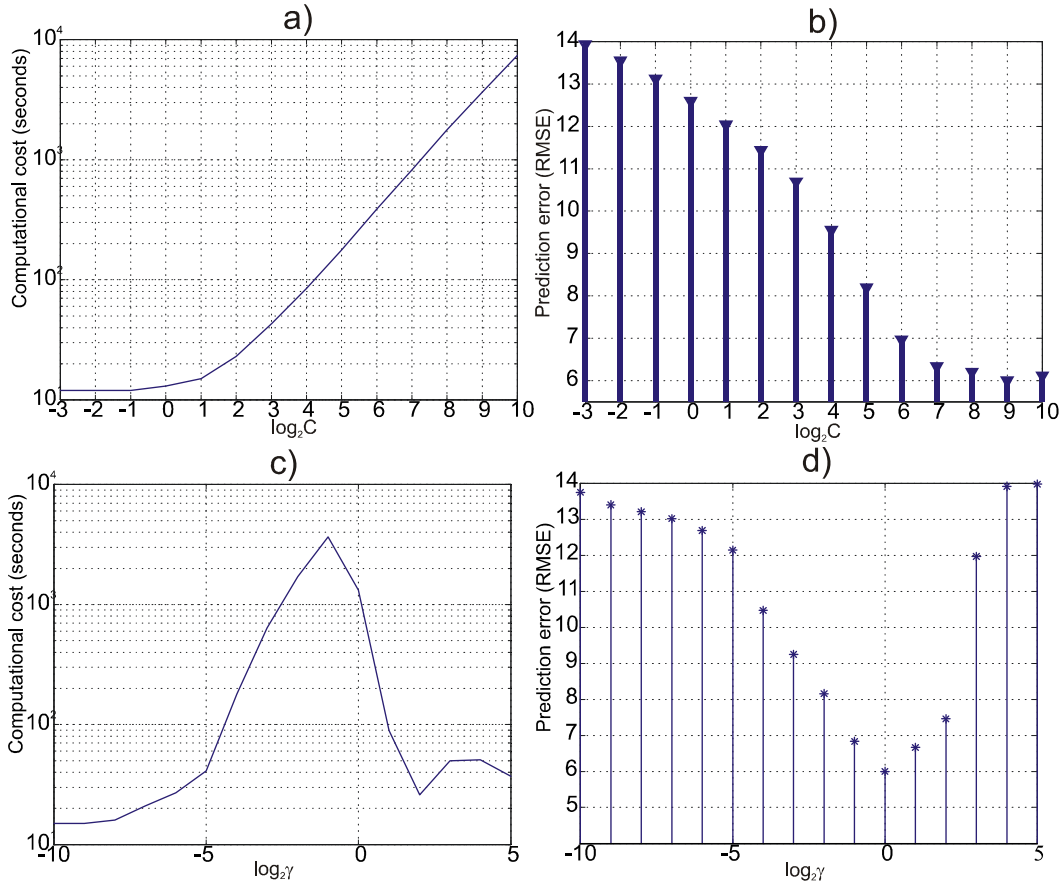


FIGURE 5.9: Computational cost influenced by the training parameters C and γ for the Q factor of the the Colpitts RF filter. a) influence of C on the computational cost when γ=0.5; b) the corresponding prediction accuracy plot; c) and d) are the computational cost and prediction accuracy diagrams showing the influence of γ when C is a constant.

In the second main case study, the Colpitts RF bandpass filter, the influence of parameter C on the computation cost of SVM training parameter determination is shown in figure 5.9 a). The figure is generated for the Q factor of the design. Although the constant computational cost in the insensitive range t_0 is different from the previous case study, the insensitive and sensitive ranges clearly present. The breaking point in this case is at $log_2 C = 1$. The slope of the linear relation is about 45 degrees and the computational cost finally reaches the point almost three orders of magnitude larger than that in the insensitive range. A similar amount of computational cost variation and the same pattern as described by the proposed computational cost model have been observed in the scan of parameter γ. The high computational cost figure, like the one in the previous case study, appears in the nonlinear sensitive range as seen in figure 5.9 c). The corresponding prediction accuracies in the search process of C and γ are shown in figure 5.9 b) and d). The

optimal values for C and γ are found between the under-trained and over-trained ranges.

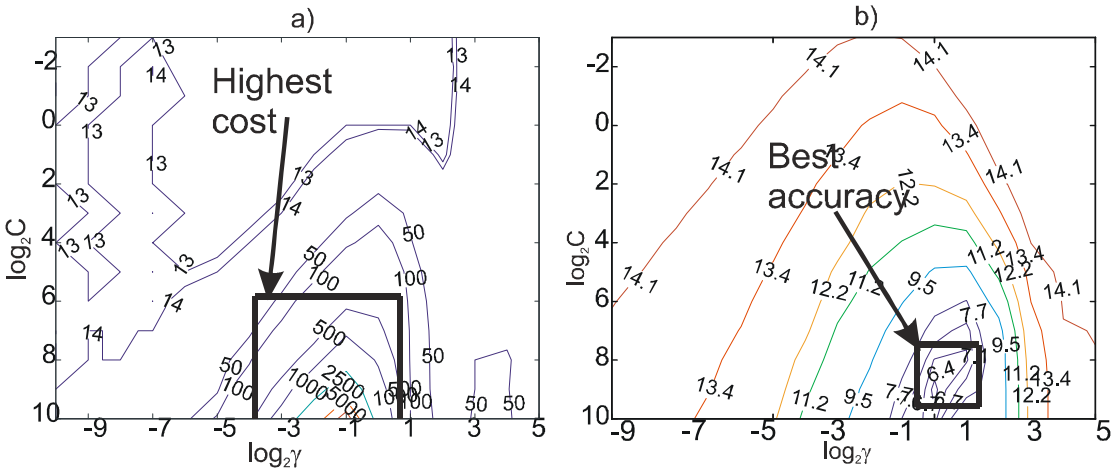


FIGURE 5.10: a) Computational cost contours (in seconds) of C-γ grid search process for the Q factor of the Colpitts RF filter; b) the corresponding accuracy performance of the models (RMS error).

The computational cost and prediction accuracy contours in the grid search are shown in figure 5.10 a) and b) separately. The areas that are of interest to the designer have been highlighted by solid boxes. The high accuracy area is inside the high computational cost area. However, it should be noted that the very high computational cost C-γ pairs at the bottom in both subplots correspond to over-trained models with low prediction accuracy. They should be avoided as they are not helpful for optimal performance model construction.

According to the experiments of these two case studies, it is clear that the standard grid search algorithm wastes computational resources as it unnecessarily scans areas of very high cost and sub-optimal accuracies. The computational cost in these regions is considerably high and it would therefore be very beneficial to achieve such savings here for more effective performance model construction.

5.2 Computational-cost aware SVM regression training parameter determination algorithm

The observations and analysis discussed in the previous section are critical in the development of the improved algorithm presented below. It is important to point

out that in the high computational cost areas, small accuracy improvements are achieved at high computational cost expense. For example, a small improvement of 0.2 in the RMS error of the RF filter case study's Q factor at log_2C=9 takes 1.5 hours of CPU time as shown in figure 5.9 b). This brings the question: is the accuracy improvement worth such a high computational expense? In addition, the very high computational cost region outside the high prediction accuracy region should be avoided because it only generates over-trained performance models and consumes computational resources but cannot contribute to finding the optimum solution.

Now, a measure of the accuracy improvement effectiveness is defined as the following:

$$E_{ipv} = \frac{\Delta A}{\Delta T} \tag{5.7}$$

where ΔA is the accuracy improvement and ΔT is the extra computational cost spent to achieve such an improvement at the current C-γ pair. Using the above effectiveness measure, a new heuristic gradient-based C-γ determination algorithm can be proposed. The algorithm uses the accuracy improvement effectiveness to dynamically modify parameters C and γ as follows:

$$C_{n+1} = k_C^{sgn(\Delta E_{ipv})} C_n \; n = 1, 2, \cdots \tag{5.8}$$

$$\gamma_{n+1} = k_\gamma^{sgn(\Delta E_{ipv})} \gamma_n \; n = 1, 2, \cdots \tag{5.9}$$

where n is the iteration index, k_C and k_γ are the stepsize refining factors for parameters C and γ correspondingly, and ΔE_{ipv} is the effectiveness change at the current iteration. The algorithm repeatedly involves the SVM trainer, calculates the accuracy improvements for equations 5.8 and 5.9, and terminates when E_{ipv} becomes higher than the pre-defined maximum effectiveness and a minimum required training accuracy has been achieved. The pseudo code is shown in algorithm 4. The initial C-γ point P_s and the refining factors k_C, k_γ are required before the algorithm starts. Firstly, m points of P_n are derived around the P_s using the initial refining factors. The related information about the surrounding points including the prediction accuracy A_n, the computational cost T_n, the accuracy improvement ΔA_n and the computational cost difference ΔT_n is calculated after

involving the SVM trainer on all the points. For each selected pair, if the prediction accuracy improvement $\Delta A_n < 0$, which means that the prediction accuracy of the performance model is improving, the point with maximum effectiveness E_{ipv} will be selected as the next point. At the same time, depending on the condition of $E_{ipv\_n} - E_{ipv\_n-1}$, the stepsize refining factors k_C and k_γ are either extended or halved for the calculation of the surrounding points in the next iteration. If the prediction accuracy degrades, i.e. $\Delta A_n > 0$, the search stays at the current point and reduces the stepsize refining factors for the next iteration. The number of surrounding points in this research is four. The expansion coefficient e for the refining factors is 2 and for degradation - it reciprocal value 1/2. At the end, the last current pair $P_{current}$ is outputted as the optimal values of C and γ.

5.3 SVM regression performance model construction

The flow chart of the SVM performance model construction system is presented in figure 5.11. This technique has been implemented in MATLAB. Unlike other similar systems [45–47], which need the designer's interaction to determine SVM training parameters, this approach utilises the algorithm proposed in the previous section for the task and is fully automatic, and therefore it is less labour intensive. The simulator can be a behavioural or a circuit-level kind, as required. WinSpice [120] has been used in the RF filter case study and SystemC [105] for the mixed-signal SDM example presented in chapter 7. The performance model construction process creates a training data set by running multiple simulations and, as a result, builds regression models for further usage in performance optimisations. A uniformly sampling scheme has been used to build training data sets. The number of samples is arbitrarily determined as the relationship between the design and performance spaces is unknown beforehand. However, the accuracy of the SVM regression performance model is verified at the testing stage after the training, and should the regression accuracy be found insufficient, the models can be rebuilt using a larger number of samples.

Each SVM regression model performs a regression functional mapping from the design space to a single performance parameter. Together, all the SVM regression models can be viewed as a direct mapping to the multi-dimensional performance

Algorithm 4 Computational cost aware SVM training parameter determination algorithm.

Require: initial point $P_s = [C_0, \gamma_0]$;
Require: k_C, k_γ;
$n = 1$;
calculate the points $P_n[1:m]$, around the current point;
for $i = 1$ to m **do**
    involve the SVM trainer;
    record $A_n(i)$, $T_n(i)$, $\Delta A_n(i)$, $\Delta T_n(i)$ for point $P_n(i)$;
end for
while $A_n > A_{min} || E_{ipv\_n} > E_{ipv\_min}$ **do**
    if $\Delta A_n < 0$ **then**
        find the point P_{temp} with maximum E_{ipv} in $P_n[1:m]$;
        $P_{next} = P_{temp}$;
        if $E_{ipv\_n} > E_{ipv\_n-1}$ **then**
            $k_C = 2 * k_C$;
            $k_\gamma = 2 * k_\gamma$;
        else
            $k_C = 2^{-1} k_C$;
            $k_\gamma = 2^{-1} k_\gamma$;
        end if
    else
        $P_{next} = P_{current}$;
        $k_C = 2^{-1} k_C$;
        $k_\gamma = 2^{-1} k_\gamma$;
    end if
    $n = n + 1$;
    $C_n = k_C C_{n-1}$;
    $\gamma_n = k_\gamma \gamma_{n-1}$;
    calculate the points $P_n[1:m]$, around the current point;
    for $i = 1$ to m **do**
        involve the SVM trainer;
        record $A_n(i)$, $T_n(i)$, $\Delta A_n(i)$, $\Delta T_n(i)$ for point $P_n(i)$;
    end for
end while
return $P_{current}$;

space. Once SVM regression models are constructed, they are stored as a knowledge database represented by the cylinder in figure 5.11, from which the performances of the design can be estimated at an arbitrary point in the design space without resorting to further simulations.

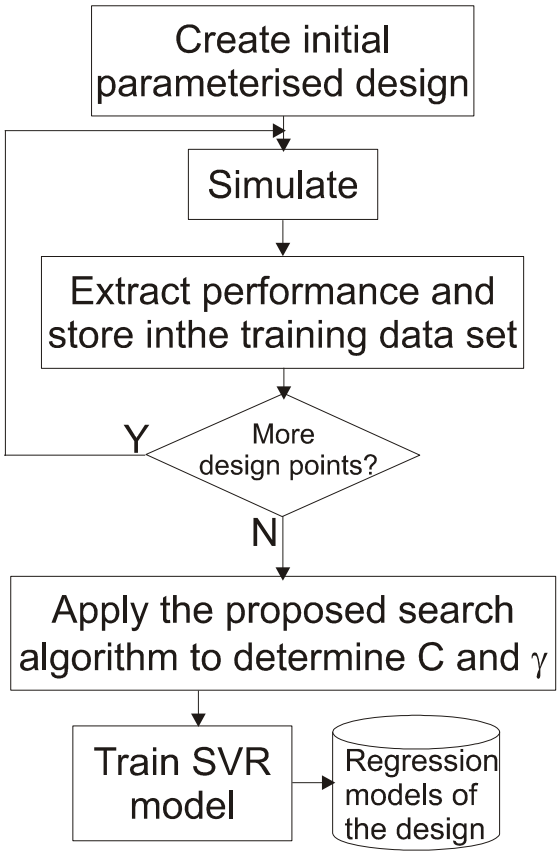


FIGURE 5.11: Flow chart of the SVM regression performance model construction.

5.4 Concluding remarks

This chapter has presented the SVM regression performance model construction methodology using a novel computational-cost aware algorithm for SVM regression training parameter determination. Firstly, a computational-cost analysis has been carried out based on the standard grid search algorithm. The influence of the training parameters C and γ on the computational cost has been illustrated. Based on a number of experimental tests, an empirical abstract computational cost and prediction accuracy model has been proposed. A new measure of the searching effectiveness in the SVM regression performance model construction is defined and utilised in the proposed SVM regression training parameter determination algorithm. Using this algorithm, an automatic SVM regression performance model construction methodology is proposed and presented. An effectiveness improvement by nearly one order in the model construction has been achieved as shown by the case studies.

Chapter 6

Knowledge based AMS performance optimisation

This chapter presents a new knowledge based AMS performance optimisation system which uses the SVM performance models constructed by the proposed methodologies in chapter 4 and 5. The automated generation of SVM performance models for AMS designs lends themselves naturally to optimisation applications. Two optimisation algorithms have been implemented. Section 6.1 introduces the structure of the optimisation system including its components and interfaces. The optimisation system using the GA for linearly graded SVM models is presented in sections 6.2.1. The other optimiser with pattern search using SVM regression performance models is introduced in section 6.3.1.

6.1 Structure of the optimisation system

The working environment of the AMS performance optimisation system is shown in figure 6.1. It has the classical structure of knowledge based optimisation systems, which is introduced in section 2.1.2. The system accepts an initial AMS design, which is parameterised, and performance constraints, which define the optimisation goals. The optimisation engine searches the design space and sends modified design parameter values to the performance estimator which predicts performance parameter using the SVM knowledge database. The knowledge database replaces

the simulations used in simulation-based optimisation systems. When the optimisation process finishes, a set of design parameters is generated for the optimal design. The generation of the knowledge database, i.e. the SVM performance models of the AMS design, is the application of the grid search algorithms in the previous two chapters. The algorithm for the trade-off between the computational cost and prediction accuracy in section 5.2 is utilised. Verification is carried out as an extra step to confirm the optimisation results by simulating the design with the optimised design set so that they can be compared with the predictions given by the performance estimator using the SVM performance models.

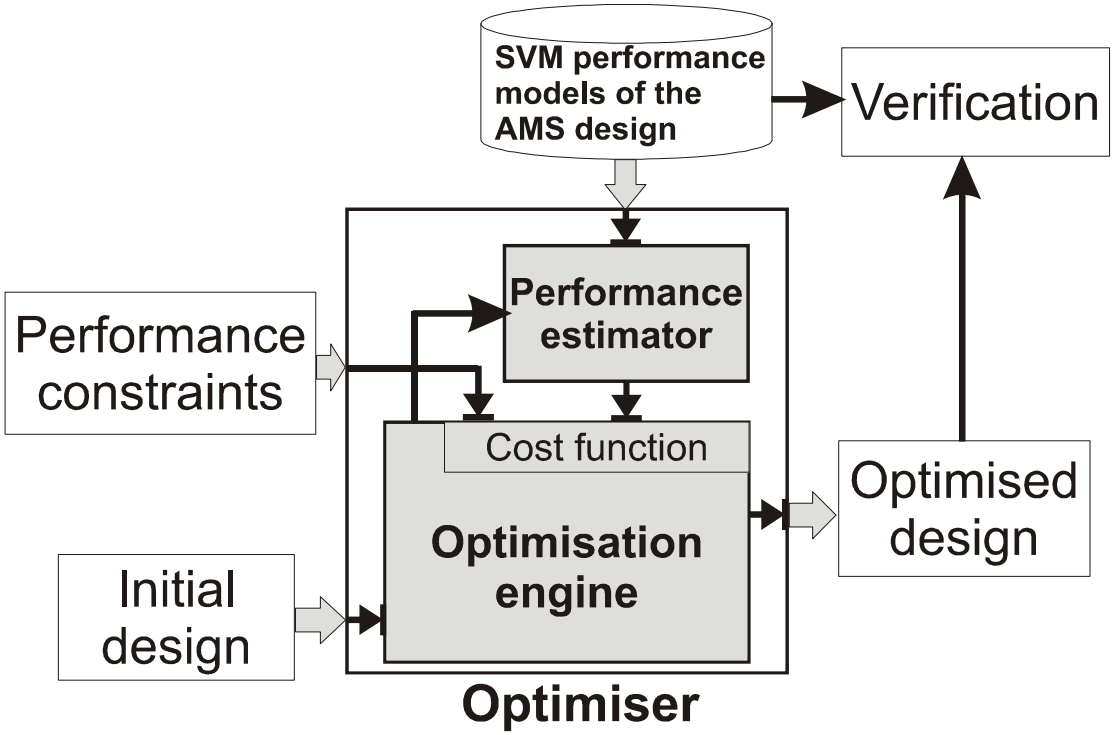


FIGURE 6.1: Working environment and structure of the AMS performance optimisation system.

6.1.1 Components of the system

The optimisation system is consisted of three components as shown in figure 6.1. The main component of the optimiser is the optimisation engine. Performance spaces of AMS designs have complex relationships with the design parameters so when they form the cost function, it presents complex characteristics in the design space. Figure 6.2 shows an example contour of the cost function $C = SNR + 50 * INT_1$ in a two dimensional design space which is a projection of the

whole design space onto the design parameters a_1 and b_1. These two parameters are the coefficients of the amplifiers in the feed forward path of the 2^{nd} order SDM design (see section 7.1 for details). Three optimal regions with local and global optima present and this requires optimisation algorithms able to escape a local optimal and to find the global optimum. The two algorithms used in the system, as shown in chapter 2, both satify this requirement.

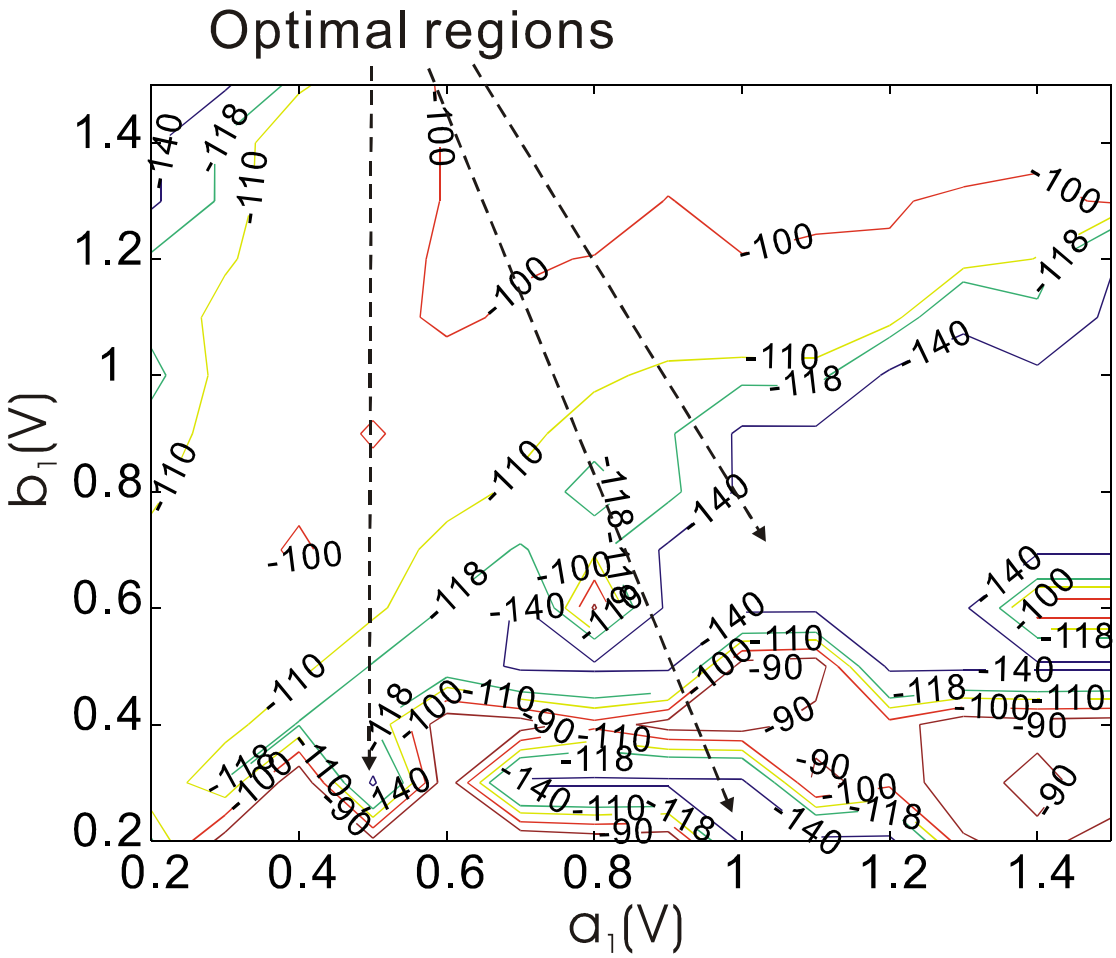


FIGURE 6.2: Contour of a cost function $C = SNR + 50 * INT_1$ formed by the performance parameters SNR and INT_1 in a projected design space of the design parameters a_1 and b_1 of the 2^{nd} order SDM.

The optimisation engine takes an initial design and performance constraints as the inputs. By including a cost function, the optimisation engine can be made independent from the system's working environment such that it can be easily replaced by other implementations with different optimisation algorithms by simply re-writing the cost function.

The cost function is formed by a combination of a linear summation and squared deviations of weighted performance parameters as the follows:

$$V_{cost} = \sum_{i=1}^{m} w_i P_i + \sum_{i=1}^{n} w_i (P_i - P_i^{spec})^2 \tag{6.1}$$

where w_i is the weight coefficient, P_i is the predicted performance value provided by the performance estimator and P_i^{spec} is the performance specification value. This provides enough capacity for both specific and open performance specifications such as 'Q factor is 100' and 'minimum chip area'. Scalar weights in the equation balance the competing objectives for various design targets.

The performance estimator needs to be customised according to the SVM performance models used in the knowledge database and the performance parameters of the design. The main flow is composed of two phases: call and run the SVM predictor and post-processing collected predictions. Two estimators have been implemented for the two SVM performance modelling methodologies separately. One is for the linearly graded performance models introduced in chapter 4; the other is for the SVM regression performance models in chapter 5.

6.1.2 Interfaces

There are three inputs and one output in the optimisation system. One of the inputs is the parameterised initial design with a vector of design parameters $D_{i\_initial}$ as the starting point and their corresponding bounds as the follows:

$$\begin{aligned} ini\_design &= [D_1, D_2, \cdots, D_m] \\ D_{i\_initial} &= c_i \bigcap D_i \in [B_{lb\_i}, B_{ub\_i}],\ i \in 1, \cdots, m \end{aligned} \tag{6.2}$$

where D_i is the i^{th} design parameter, c_i is the i^{th} parameter's initial value, B_{lb_i} and B_{ub_i} are the lower and upper bounds of the i^{th} design parameter. This information is used to configure the optimisation engine. The other input, a vector of the performance constraints, is used by the cost function to build an overall numerical evaluation of the AMS design. The performance constraints can have specific value, e.g P_i is exactly equal to V_c or a tendency such as P_i approaches to the min/max where $i \in 1, \cdots, n$ is the index. The third input is

the SVM performance models and they are involved by the performance estimator iteratively. The output of the optimisation is a design parameter vector that can be used to assign values to the generic parameters in the design netlist for simulations.

6.2 Optimisation system using the linearly graded SVM performance models

6.2.1 Genetic optimisation

When the knowledge database is the linearly graded SVM performance models, the optimiser uses a canonical GA as the optimisation engine. The population size is usually from 10 to 20 and it is suitable for the case studies used in this research. For problems with larger number of design parameters, a larger population size should be selected. Every design parameter is mapped to one gene so that one design vector corresponds to one chromosome of one individual. Pseudo code of the algorithm is shown in algorithm 5.

The algorithm requires an initial population $Pop[1 : n]$ with n individuals. An individual's chromosome is formed by the m design parameters $D[1 : m]$. In every generation, n chromosomes will be evaluated by the performance estimator using the SVM performance models. Performance figures are included in the cost function f_C to calculate the overall performance. The cost function values are sorted so that chromosomes with better performances have a higher chance to be selected by the Roulette Wheel algorithm. Firstly, r parents chromosomes are selected. Then for all the parents, if the crossover possibility $RAND$ is larger than the predefined possibility pc, crossover happens on parents M and D to generate their offspring. After all the offspring have been generated, each gene in every chromosome may mutate if mutation possibility $RAND$ is larger than the predefined mutation possibility pm. The mutated gene g replaces the old gene in the chromosome $D(o)$, where o is the position.

An one-point arithmetic crossover [121] operator is used in the crossovers which are the main variation mechanism during the evolution. The crossover position is

Algorithm 5 Algorithm of the GA optimisation engine.

Require: Initial population $Pop[1:n]$;
Require: every chromosome is $D[1:m]$;
while $i < g$ **do**
  for $j = 1$ to n **do**
    evaluate $Pop(j)$ performance using SVM models;
    evaluate $f_C(j)$ for $Pop(j)$;
  end for
  generate r parent chromosomes by Roulette Wheel selection algorithm;
  while $l < r$ **do**
    if $RAND > pc$ **then**
      select two parents M and D;
      generate their offspring;
      update the selected chromosome list r;
    end if
    $l = l + 1$;
  end while
  while $s < n$ **do**
    if $RAND > pm$ **then**
      while $o < m$ **do**
        generate a new gene g;
        $D(o) = g$;
      end while
    end if
  end while
  $i = i + 1$
end while
return $Pop[1:n]$

randomly determined by:

$$p = \lambda \cdot m \tag{6.3}$$

where p is the crossover position on the chromosome, $\lambda \in [0,1]$ is a random crossover ratio; m is the chromosome length. The design point with the best performance in the last generation is outputted as the optimal solution achieved in the design space exploration. MATLAB scripts have been implemented using data structures for Pop and chromosome definitions.

6.2.2 Performance estimator

The dedicated performance estimator, which uses linearly graded SVM performance models, is shown in algorithm 6. The main **while** loop can predict the values for all the performance parameters P_i, $i \in 1 \cdots, n$ for the given design set

Algorithm 6 Algorithm of the performance estimator using the linearly graded AMS SVM performance models.

Require: design sample $D[1:m]$;
while $i < n$ **do**
  SVM classification $O_C = SVMClassification(D[1:m])$;
  identify the class $MC_{Pi,j}$ the design set belongs to;
  if P_i is regressive **then**
    load SVM regression models $RC_{Pi,j}$ that corresponds to $MC_{Pi,j}$;
    SVM regression $O_R(i) = SVMRegression(D[1:m], RC_{Pi,j})$;
    store $O_R(i)$;
  else
    store $O_C(i)$;
  end if
end while
return O_R and O_C;

$D[1:m]$. Firstly, the classification model of the i^{th} performance parameter is carried out using function $SVMClassification()$. The classification output O_C is generated and the corresponding classification model $MC_{Pi,j}$ is located. If the current performance parameter needs a numerical prediction, a regression function $SVMRregression()$ will be involved with the regression model $RC_{Pi,j}$. For each performance parameter, either a classification prediction value $O_C(i)$ or a regression prediction value $O_R(i)$ is recorded and returned.

6.3 Optimisation system using the SVM regression performance models

6.3.1 Pattern search optimisation

The standard MATLAB pattern search algorithm has been employed as the optimisation engine in the optimiser using SVM regression performance models. As

introduced in section 2.2.2.3, the algorithm can obtain global optimum in a design space with local optimums.

The most significant feature of the algorithm is that it optimises designs without any information of derivatives of the performance space of the design. So it is safe to utilise the algorithm in this research as the characteristics of the relationship between the design and performance spaces are unknown.

The algorithm is explained in algorithm 7. The **while** loop is executed when current tolerance $t_{current}$ is larger than the specified tolerance τ. The algorithm uses a pattern to search n points around the current point $p_{current}$. The pattern vector $V[1:n]$ separates the new points $p_{next}[1:n]$ mesh size m away from $p_{current}$. The best point is selected by polling the cost function values on $p_{next}[1:n]$ and used as the new starting point for the next loop.

Algorithm 7 Pseudo code of the pattern search algorithm.

Require: initial point $p_{current} = p_{ini}$;
Require: initial mesh size $m = m_{ini}$;
while $t_{current} > \tau$ **do**
    modify pattern vector $V[1:n] = V[1:n] * m$;
    calculate new points $p_{next}[1:n] = p_{current} + V[1:n]$;
    evaluate the cost function values $c[1:n]$ at point $p_{next}[1:n]$;
    poll the cost function values $c[1:n]$;
    update the current point $p_{current}$ to p_i who has the largest cost function value;
    update mesh size m;
end while
return with $p_{current}$;

6.3.2 Performance estimator

The performance estimator is simpler using the SVM regression models than the one for the linearly graded SVM models. As shown in algorithm 8, the main iteration process maintains as in the performance estimator for the linearly graded SVM performance models. For each performance parameter $P_i, i \in 1, \cdots, n$, the corresponding regression model R_{Pi} is loaded first, then applied in the SVM regression prediction. The results O_R for all the n performance parameters P_i are collected and returned.

Algorithm 8 Algorithm of the performance estimator using the SVM regression performance models.

Require: design sample $D[1:m]$;

1: **while** $i < n$ **do**
2: load SVM regression models R_{Pi} that corresponds to P_i;
3: SVM regression prediction $O_R(i) = SVMRegression(D[1:m], R_{Pi})$;
4: store $O_R(i)$;
5: **end while**
6: **return** O_R;

6.4 Concluding remarks

A novel, knowledge based AMS performance optimisation system has been presented in this chapter. It is applicable to a wide range of AMS designs. The optimal solution can be obtained very fast. Although the preparatory effort for the knowledge database is high compared with simulation based systems, the reusability of system is enhanced because the models can be used in many optimisations. The interfaces, main working process and components of the optimiser have been introduced. Two optimisation algorithms - genetic algorithm and pattern search algorithm - have been applied in two systems for linearly graded SVM performance models and SVM regression performance models separately. In the case studies, the experiment using the GA can optimise the design in 10 to 20 minutes depending on the number of generations; for the three experiments using the pattern search finishes within 5 minutes. All of the experiments can achieve optimal designs with higher performances than manual designs. Details of the experiments are in the next chapter.

Chapter 7

Case studies

Firstly, this chapter contains two case studies for the general AMS performance modelling and optimisation methodologies introduced in chapter 4, 5 and 6. The first case study in section 7.1 is a 2^{nd} order SDM. Based on the switched capacitor implementation, the behavioural level design has integrated most circuit level imperfections that have influence on system's performances. This makes modelling the nonlinear relationships between the design and performance parameters even more challenging. The SDM has been developed in both MATLAB Simulink and SystemC (see appendix B). Using this example, both the linearly graded models and SVM regression performance models for the case study have been constructed and applied in two optimisation systems separately. In section 7.2, the second case study based on an RF bandpass Colpitts filter is presented. This example represents design challenges encountered in RF designs in high frequency applications that standard filter transfer function modelling method is invalid. An accurate physical model for on-chip silicon spiral planar inductors has been included in the circuit level simulation for data generation (see appendix C). SVM regression performance models of the case study have been constructed and used in optimisations.

Secondly, additional experiments to confirm the computational cost and accuracy models (chapter 5) of the standard grid search SVM training parameter determination algorithm are accomplished and presented in section 7.3. Five public data sets from different sources are randomly selected. The influence of C and γ on the computational cost and the two dimensional computational cost contours have

been listed, and they reaffirm the proposed computational cost and SVM training accuracy models.

7.1 2^{nd} order SDM: a mixed-signal example

This case study is about a mixed-signal SDM example modelled at the behavioural level. The construction of the linearly graded SVM model and SVM regression performance model are introduced in section 7.1.1 and 7.1.2 separately. The performance optimisation experiments and results are presented in section 7.1.2.2.

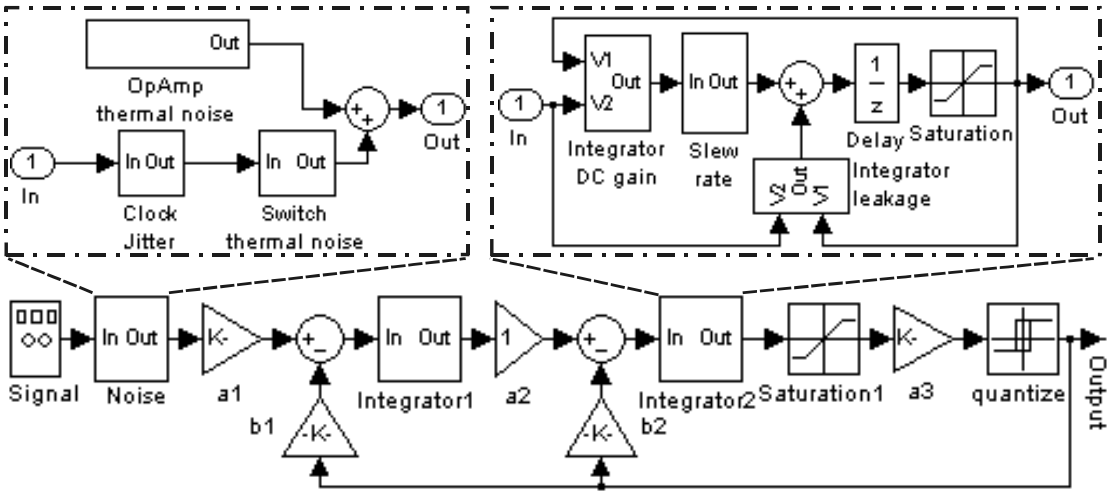


FIGURE 7.1: Signal flow graph of the non-ideal SDM using MATLAB Simulink modules.

7.1.1 Scenario 1: linearly graded SVM model construction

At the early stage of the research, the SDM design used for the linearly graded SVM model construction contains less imperfections than the models introduced in appendix B. The model includes the imperfections such as clock jitter, OpAmp and switch thermal noise, OpAmp finite and non-linear DC gain, SR and integrator leakage as well as quantiser hysteresis and offset. However, compared with the complete model in figure 7.1, there are two differences. The first one is that there is no saturation component after the second integrator stage. The second one is that the amplifier coefficient a_3 is not included. This is mainly because the hysteresis and offset of the quantiser are not considered as important and influencing [126] so

the effects of saturation and coefficient a_3 is absorbed by the quantiser immediately and does not propagate further. So the four amplifier coefficients on the feed forward and feedback pathes a_1, b_1, a_2 and b_2 are selected to construct the design space in the linearly graded SVM performance model construction.

The two most important performance parameters of SDM systems are the SNR and dynamic range (DR). Both are calculated using FFT on the simulation results. The peak SNR is selected as a performance parameter instead of the whole SNR curve as this is usually what designers are interested in. DR is defined as a ratio of the output power at the frequency of the input sinusoid with a full-scale input over the output power of a small input for which the SNR is 0dB. The third is the stability, which is a binary classification parameter, i.e. the SDM is either stable or unstable. An easy definition of the stability is when internal signals are detected exceeding a predefined threshold, the SDM is unstable. The fourth performance parameter is the dynamic signal range. This requirement represents a severe problem in circuit technologies such as CMOS VLSI, where the dynamic range of the technology itself is limited [122]. In the SDM, this parameter is controlled by the outputs of the two integrators (represented below as INT_1 and INT_2). So, in total, there are five performance parameters. The linearly graded SVM performance model of the SDM system is for the direct relationship between the four dimensional design space and the five dimensional performance space. 4725 design points are simulated as the training data set. As explained in section 5.3, the number of training points is verified at the testing stage when the models are constructed.

7.1.1.1 Balanced data grading algorithm

Figure 7.2 shows the distribution of the design samples in the performance space with different grading vectors for each performance parameter. The left bars in the figures are the distributions achieved using the original grading vectors provided by a designer. They reveal the disadvantage of the original grading vectors that some of the grades have very few samples than other grades. The sparse distribution can generate over-fitted models easily thus increases the difficulty for regression model construction. The right bars in the subplots are the distributions achieved using the recalculated grading vectors for all the performance parameters using the BDG algorithm. A new set of grading vectors is generated and non-equilibria

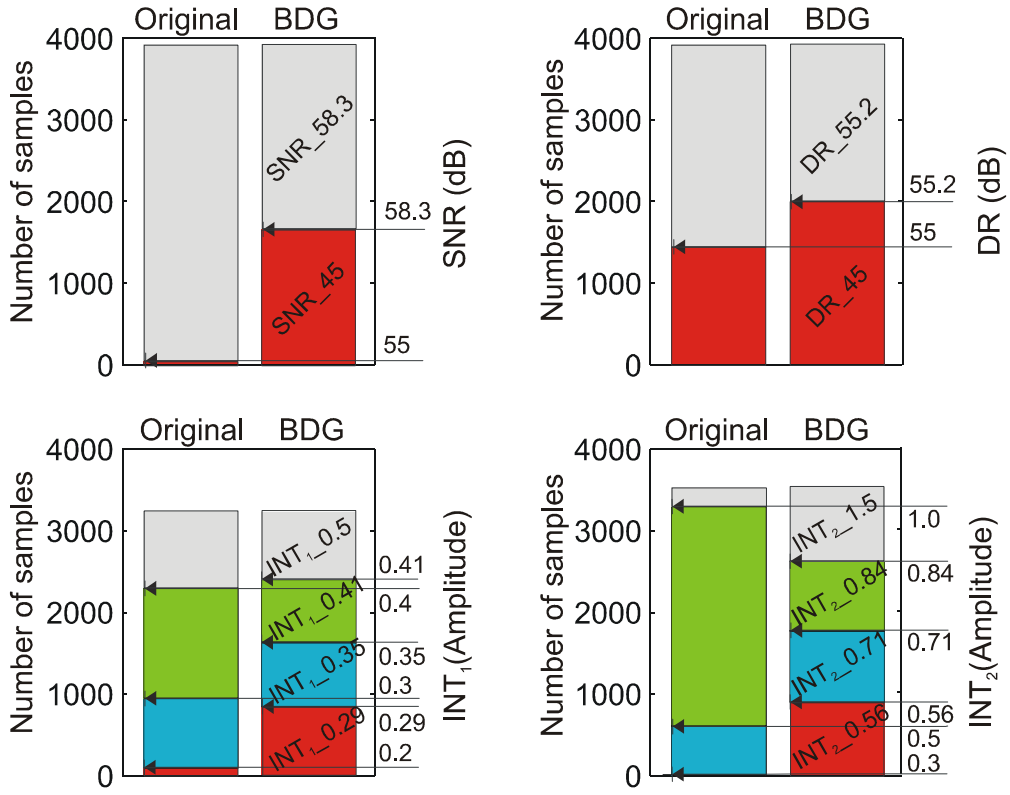


FIGURE 7.2: Distributions of the design samples of every performance parameter dimension in dual-y-axis plots. The left y axes are the number of samples and the right y axes are performance parameters' values.

are avoided. Each segment in the BDG bars in figure 7.2 corresponds to one class of the performance parameter and is labelled by the inclined text.

7.1.1.2 Model construction

Firstly, the SVM training parameter needs to be determined. The standard grid search algorithm is used here. Figure 7.3 a) shows the CGS accuracy contours of the stability with the scanning ranges of $C \in [2^{-3}, 2^{15}]$ and $\gamma \in [2^{-10}, 2^{5}]$. The expected optimal region is boxed by a thick frame and centered at $C = 2^{14}, \gamma = 2^{-3}$. This feature is observed in all the CGS phases of SVM classification experiments of all the performance parameters. Then the optimal region is searched again in the RGS phase. The accuracy contours in figure 7.3 b) show the result of RGS scanning for classification accuracy of the stability parameter. As in the figure, the accuracy contours are in the region of $C \in [2^{13}, 2^{15}]$ and $\gamma \in [2^{-5}, 2^{-3}]$, where a pair of C and γ with improved classification accuracy is found. The summary

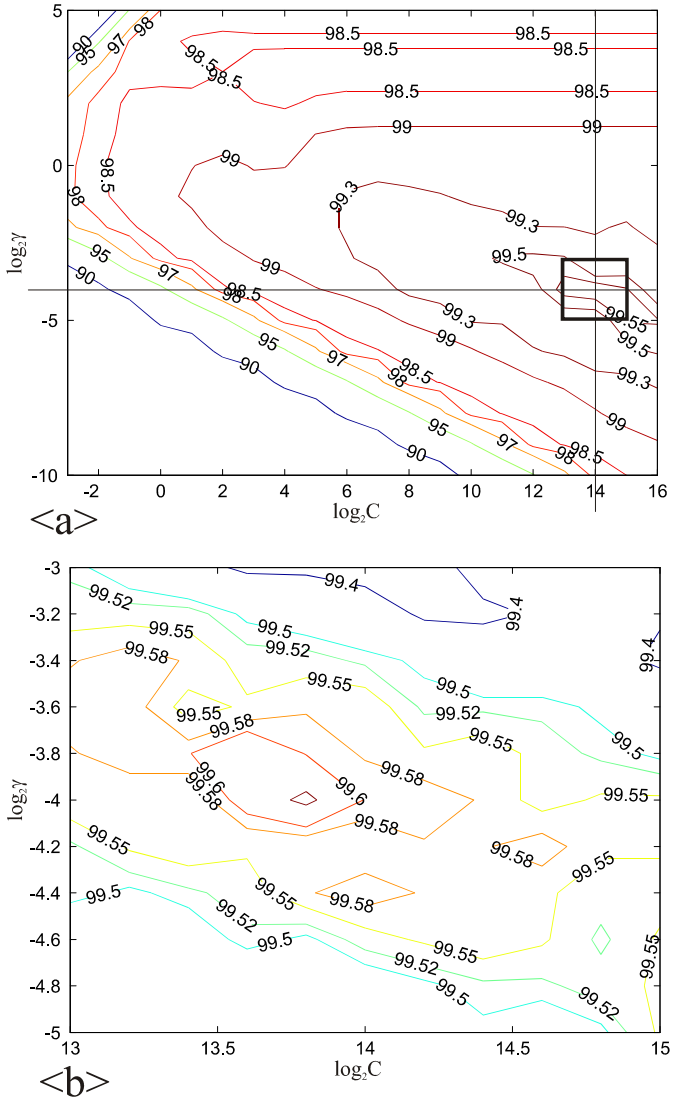
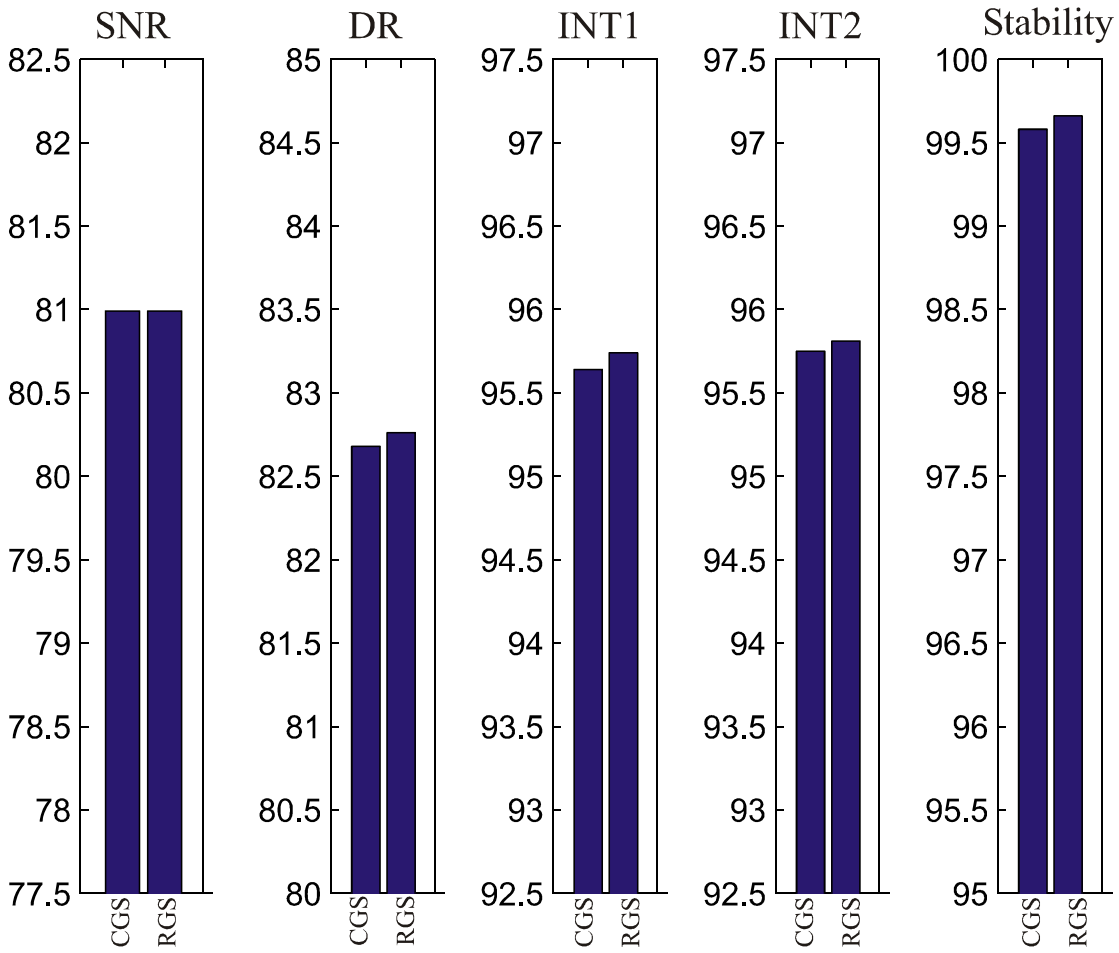


FIGURE 7.3: Classification accuracy contours of parameter stability. a) accuracy contours of the CGS phase; b) accuracy contours of the RGS phase.

of the CGS and RGS SVM classification results are compared in figure 7.4. It can be seen that RGS can find improved or at least equal SVM training parameters.

Figure 7.5 a) and b) shows the CGS and RGS mean squared error (MSE) contours for the SVM regression model of INT_1_0.35, which is a grade of INT_1. Similar to the classification accuracy contours, the plot shows the same feature that there is an optimal region confined in the CGS scanning, where it is the RGS phase that searches the region in detail for even better training parameters. Results of the CGS and RGS phase in the SVM regression model construction process are summarised and compared in figure 7.6.



	CGS	RGS
Total time	47:50:46	02:18:04

FIGURE 7.4: Comparisons of the classification accuracies of all the performance parameters in CGS and RGS phases. The total computational costs of the two phases are listed in the bottom table.

SVM training parameters found in the RGS phase improve the prediction accuracies for both classification and regression models. These training parameters have been used to train the models. Both the prediction accuracy and computational cost of the linearly graded models are compared to the full-space analysis approach, which treats the entire design space as a whole. The top plots in figure 7.7 show the MSE performance comparison of the SVM regression models. It is clear that, in most cases, the MSE performance of the linearly graded approach is better than that of the full-space analysis approach. The bottom table presents the summary of the computational cost. The significance is that it shows that the proposed methodology lead to more than 50% saving on the computational cost.

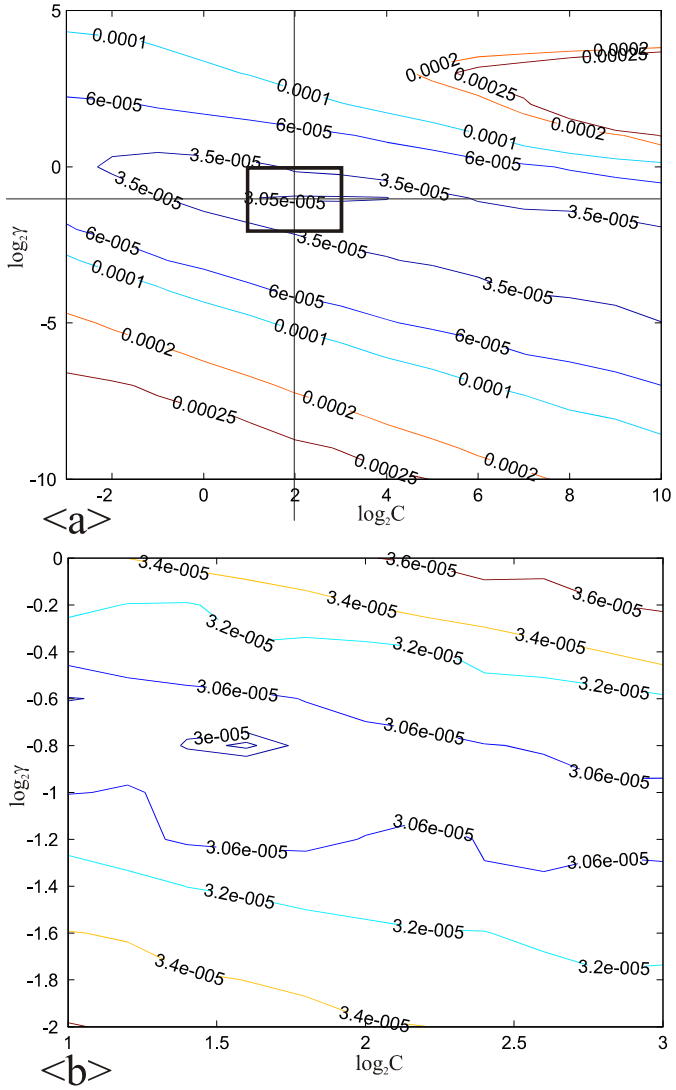
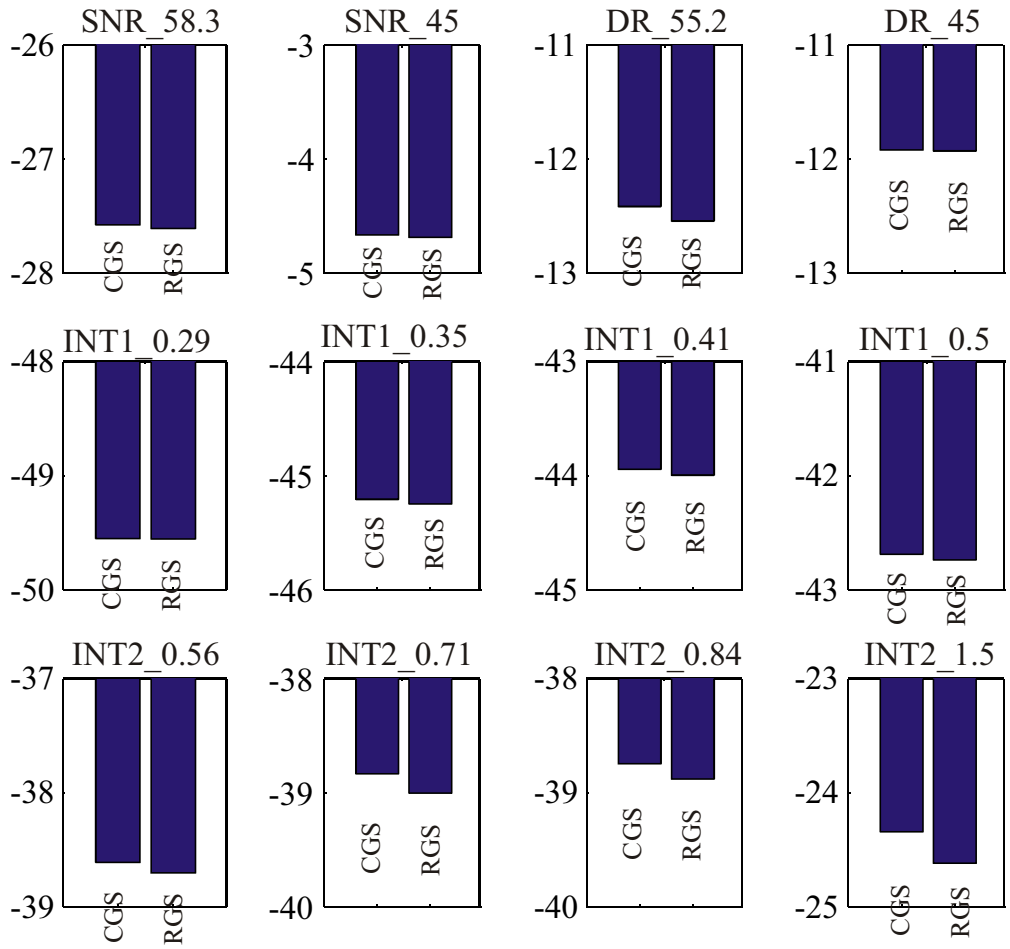


FIGURE 7.5: MSE contours of the SVM regression model for 0.35 grade of the performance parameter INT_1. a) the CGS MSE contours; b) the RGS MSE contours.

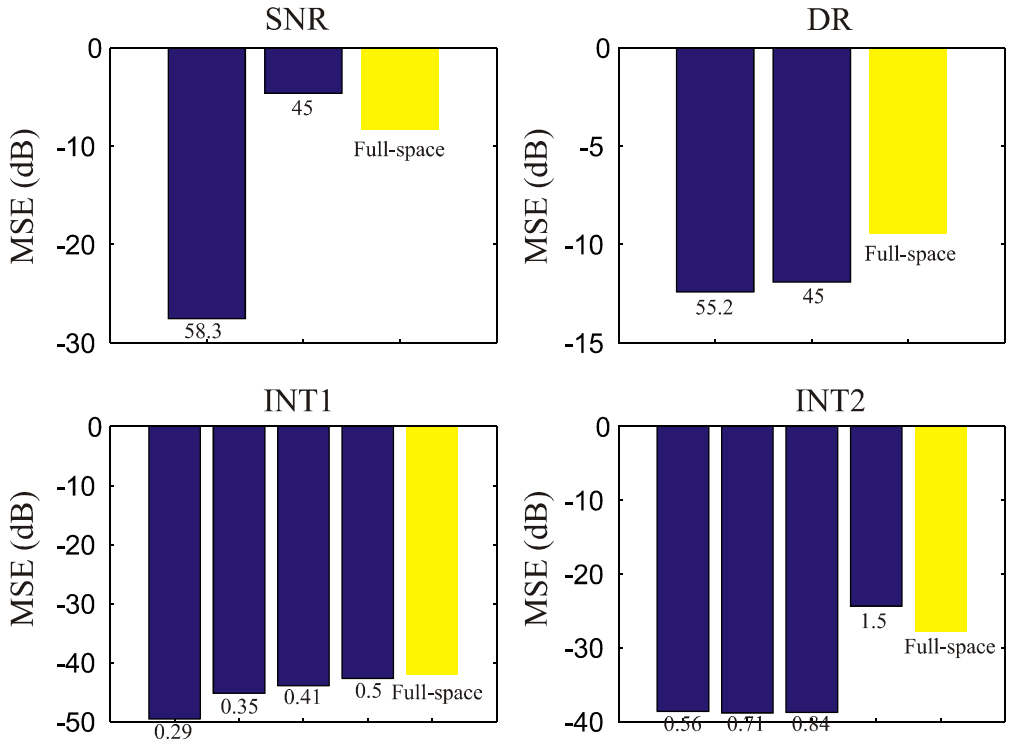
Two testing data sets have been generated to verify the classification and regression performance models. The generation of the data sets assures that all the testing data samples are unknown to the models so that the trained models must have good generality to preserve their accuracy. Testing data set 1 (T_1), with 1323 samples, spreads over large regions in the design space covering both the stable and unstable designs cases to test the overall quality of the models. Testing data set 2 (T_2), with 4725 samples, is centralised on a small stable region in the design sapce with mainly preferable design cases but different trade-off between the performance parameters. The testing procedure is to use the classification models to predict the classification of the testing data first then according to the predictions,



	CGS	RGS
Total time	49:24:57	66:52:47

FIGURE 7.6: Comparisons of the MSE (dB) accuracies of all the performance parameters' grades in the CGS and RGS phases. The model construction computational costs are listed in the bottom table.

corresponding regression models are used to give numerical estimations. The performance of the models in the testing phase is summarised in table 7.1. The overall prediction performance confirms that the training performance is well preserved by the models. Blank cells in the table mean that the corresponding models are not used either because the performance space covered by the testing data set does not include that region or because of misclassifications. Misclassifications happen during the classification testing. The exact influence of these errors on the prediction accuracy is yet unknown but according to the experiments, mistakes are in very reasonable range. The MSE performance of the regression testing shows that very accurate numerical predictions are calculated, so misclassifications do not degrade the performance of the method significantly.



Approach	Computational cost
Linearly graded	50:08:50 (classification) 49:27:57 (regression CGS)
Full-space analysis	190:44:11 (CGS only)

FIGURE 7.7: Comparisons of the prediction accuracies of the regression models using the linearly graded approach and the full-space analysis approach. The corresponding grading elements' values are labelled at the bottom of each bar.

7.1.1.3 Genetic algorithm optimisation

The configurations of the GA optimisation engine include the number of generations (100), population size (20), number of genes in a chromosome (1 chromosome is composed by 4 genes), the crossover rate (0.7) and the mutation rate (0.05). With these configurations, two experiments have been carried out. In the first experiment, the design to be optimised corresponds to a cost function that has a weighted combination of all the four regressive performance parameters. The running results of the GA optimisation engine are shown in figure 7.8. When the population is evaluated in the GA optimisation, the classification models of the stability parameter are used firstly to filter out the chromosomes that are predicted as unstable so that the stable chromosomes are left for further GA evolutions. The four regressive performance parameters SNR, DR, INT_1 and INT_2 are contained

TABLE 7.1: Testing results for the linearly graded SVM classification and regression performance models.

	Parameter	T_1	T_2
$C_{lassification}$ $A_{ccuracy}$	SNR	66.72%	78.73%
	DR	74.5%	62.7%
	INT1	87.7%	97.1%
	INT2	78%	99.6%
	stability	98.4%	100%
$R_{egression}$ $MSE_{(dB)}$	SNR_58.3	-5	-26
	SNR_45	-	-
	DR_55.2	-11	-6
	DR_45	-	-5
	INT1_0.29	-	-29
	INT1_0.35	-43	-47
	INT1_0.41	-37	-50
	INT1_0.5	-38	-
	INT2_0.56	-28	-20
	INT2_0.71	-28	-29
	INT2_0.84	-26	-
	INT2_1.5	-14	-

in the cost function to calculate an overall evaluation. The convergence curve of the predicted overall performance result of the four regressive parameters is shown in figure 7.8 a) with both of the best performances and the average of the whole population. The corresponding convergence curves of each regressive parameter are shown as b) to e). The convergence behaviour of the GA optimisation is consistent to the desired changing trend of all the performance parameters. In the second experiment, the design target is solely to maximise the SNR without considering any other performance parameters. So the cost function only includes the SNR.

Results from both experiments are compared with a manual design based on the method introduced in section B.2 and the results have been summarised in table 7.2. The computational cost of the optimisation using the proposed optimisation system with the GA configurations is less than 20 minutes. The performance figures in the two experiment columns in table 7.2 have been verified by SystemC and MATLAB simulations using the optimised design parameters. Experiment 1 shows that the overall performance of the SDM has been improved, especially the INT_1 and INT_2, although it is achieved in different degrees for the four parameters

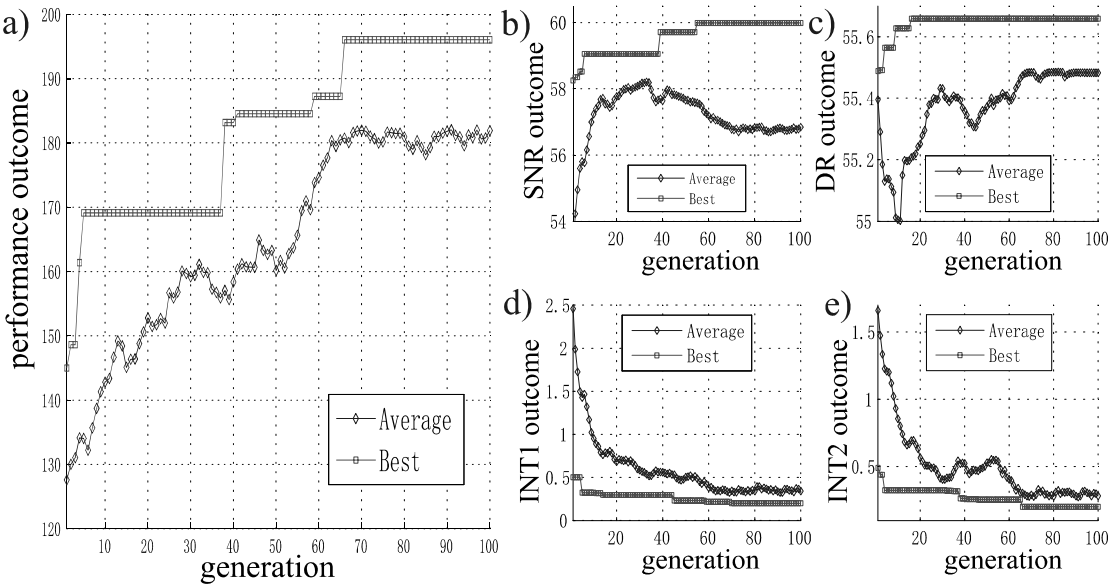


FIGURE 7.8: Typical convergence curves of the predicated performances with the GA optimization engine using a cost function including all the performance parameters.

separately. Experiment 2 shows that both of the SNR and INT_1 have outstanding

TABLE 7.2: Comparisons of the performance values achieved by the manual design and the optimisation in experiment 1 and 2.

Parameter	Manual	Experiment 1		Experiment 2	
		Values	$I_{improvements}$	Values	$I_{improvements}$
SNR (dB)	58.4	59.3	1.5%	62.2	6.1%
DR (dB)	55.2	55.3	0%	55.4	0.3%
INT_1 (amplitude)	0.34	0.26	30.8%	0.29	17.2%
INT_2 (amplitude)	0.65	0.33	97%	0.72	-9.7%
stability	Yes	Yes	-	Yes	-

improvements. The improved design can provide half more resolution bit (as indicated by equation 2.15) at the output making the ENOB equal to 10 in the optimised SDM as compared to 9 in the manual design.

7.1.2 Scenario 2: SVM regression model construction

At the latter stage of the research, the complete SDM model, as shown in figure 7.1, is used for the SVM regression model construction and optimisation. The

model of the design includes the saturation and amplifier coefficients a_3 even the influence of these components are not significant. The selected design parameters are the amplifier gains (a_1, a_2, a_3, b_1, b_2

A MATLAB script extracts the following performance parameters from SystemC time-domain simulations: the SNR, maximum input signal amplitude range ($ISAR$) at peak SNR, maximum dynamic range of integrator 1 and integrator 2 outputs INT_1 and INT_2 correspondingly as well as the SNR degradation ratio ($SNRDR$). Unlike in scenario 1, the DR is not selected because it is directly related to even equal to SNR [111]. As in the previous implementation, the SNR is calculated by means of an FFT analysis on time-domain waveforms and peak SNR has been selected rather than the whole SNR curve. The $SNRDR$ is measured as the average slope of the SNR curve obtained from multiple simulations using input signals of varying amplitude. It indicates how well the system can reserve the SNR performance in a certain ISAR for example from 0.5 to 0.8. In summary, the models to be constructed are about the relationship between the six dimensional design space and the five dimensional performance space.

7.1.2.1 Computational cost aware model construction

The computational cost using the standard grid search algorithm and the corresponding accuracy contours for this case study has been shown in section 5.1.4.1 in chapter 5. Here, the results of applying the computational cost aware algorithm for SVM training parameter determination are shown. To construct the SVM regression models, each design parameter range was uniformly sampled to provide 3125 samples in the design space. The SVM regression model construction process is time consuming, especially in this case, as some performance parameters need to be calculated from multiple simulation runs. The CPU time used by the standard

TABLE 7.3: Computational cost comparison of the standard grid search and the proposed method in the 2^{nd} order SDM case study.

Performance model construction approach	Computational cost
Standard grid search	6 days 1 hour
Proposed method	1 day 4 hours

grid search algorithm and the proposed computational-aware approach are listed in table 7.3. The proposed algorithm saves nearly 5 days of CPU time in this case. This is significant as this saving can make the proposed method more practical for applications.

TABLE 7.4: RMS errors of the SVM regression performance models constructed using the training parameters determined by the standard grid search and proposed methods and the corresponding C and γ values for the 2^{nd} order SDM.

	Performance prediction accuracy RMS error				
	SNR_{dB}	SNRDR	ISAR	INT_1	INT_2
Grid search	12.7	9	5.75E-3	3.66E-3	3.82E-3
Proposed method	12.9	9	6.32E-3	5.44E-3	4.11E-3
	C and γ				
Grid search	512, 4.6	1024, 6	1, 5.28	36.7, 1	8, 2.64
Proposed method	362, 4	1024, 5.6	1, 4	36.7, 1.52	90.5, 1

Table 7.4 summarises the performance prediction accuracies of the SVM regression models and the corresponding training parameters C and γ values. Firstly, it shows that there are no significant difference in the performance prediction accuracy between both algorithms. Secondly, the C and γ values found by the two approaches have similar values in most cases, except for parameter INT_2. However, both models estimates the SDM performance with comparable accuracies indicating the existence of semi-optimal solutions in the training parameter determination.

7.1.2.2 Performance optimisation using pattern search algorithm

Two cost functions with different weighted sums of performance characteristics were used in two performance optimisation experiments: Exp I and Exp II. Cost function I gives more weighting to frequency-related performance characteristics namely SNR and SNRDR, whereas cost function II gives preference to DC-related characteristics. Optimisation results are summarised in table 7.5. The first row in table 7.5 specifies the performance characteristics obtained using the standard

SDM design procedure [123] introduced in section B.2 so that they can be compared with the best designs found through the optimisation using the automatically generated SVM regression model based knowledge database.

TABLE 7.5: Summary of the standard and optimised design results for the 2^{nd} order SDM.

Parameters			SNR_{dB}	SNRDR	SR	INT_1	INT_2
Standard design using non-ideal SDM model			64.2	10.73	0.7	0.29	0.61
Grid search	Exp I	Optimised	84.1	0.01	0.74	1.48	1.51
		Verified	84.1	0.01	0.75	1.5	1.5
	Exp II	Optimised	69.9	7.47	0.83	0.2	0.3
		Verified	69.2	9.75	0.8	0.29	0.34
Proposed method	Exp I	Optimised	81.1	0.8	0.82	1.44	1.47
		Verified	84	0.2	0.8	1.5	1.5
	Exp II	Optimised	71.8	7.51	0.86	0.14	0.21
		Verified	68	10.9	0.85	0.2	0.24

The results in table 7.5 show that the standard grid search approach determines values for C and γ such that they provide slightly more accurate predictions than the proposed approach. However, the marginal prediction performance degradation is compensated by a huge computational cost saving. Optimisation is extremely fast for both of these two experiments. Exp I takes 82 seconds and exp II - 121 seconds of CPU time.

7.2 Colpitts filter: an RF example

This section is about the application of the SVM regression performance model construction on a Colpitts filter example (see appendix C for details). Firstly, the design and performance spaces are introduced. Then it is followed by section 7.2.2, which contains the construction of the SVM regression performance models. Section 7.2.3 presents the optimisation results using the knowledge based optimiser. Standard filter synthesis techniques cannot be used in this application [135] due to its non-linear nature, with significant losses and parasitics in the RF operating ranges, especially those in the spiral inductor and transistor. The goal in this case study is to model the relationship between the design and performance parameters

and to demonstrate how an optimal design can be extracted from this relationship according to a set of performance constraints.

7.2.1 Design and performance spaces

The design space is a six-dimensional hyperspace formed by the following parameters within their corresponding bounds: L_1, L_2, C_1, C_2, I_{bias} and W_{id} (width of the NMOS transistor, see figure 7.9). L_1 and L_2 are selected out of the five structural

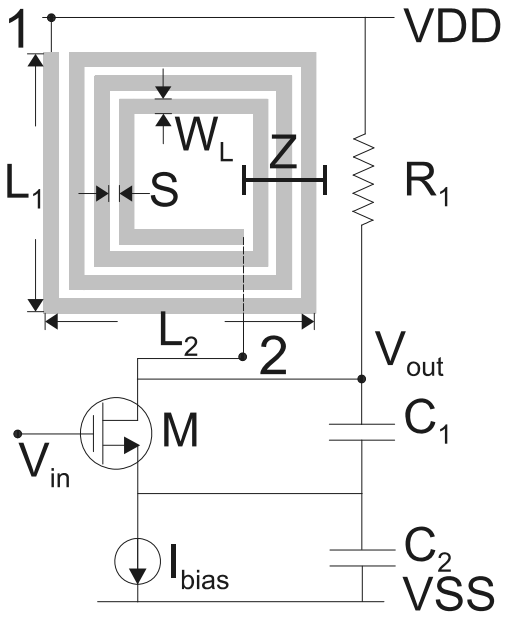


FIGURE 7.9: Schematic of the Colpitts RF bandpass filter with a highlighted on-chip planar spiral inductor.

parameters associated with the spiral inductor because they have the most significant influence on the inductor model while other three W_L, S and Z are of minor importance. I_{bias} and W_{id} can affect the effectiveness of the Q-enhancement. The two capacitors are related to the centre frequency performance parameter by the following ideal equation:

$$F_c = \frac{1}{2\pi}\sqrt{\frac{1}{L \cdot C_T}} \quad and \quad C_T = \frac{C_1 C_2}{C_1 + C_2} \tag{7.1}$$

where L is the total inductance of the inductor.

The performance space is formed by five performance parameters. The centre frequency (F_c) and the passband bandwidth (determined by the Q factor) are both

included in the performance space as the two key measures of the filter quality. As both on-chip capacitors and the inductor are area consuming, the area of the design has been included in the performance parameters and calculated as:

$$A_{total} = A_L + A_C + A_M \tag{7.2}$$

where A_{total} represents the total area of the design consisted of area of the inductor A_L, two capacitors A_C and the NMOS transistor A_M. A_L is the rectangular area formed by $L_1 \times L_2$; area of the capacitors are calculated by:

$$A_C = \frac{t_{ox} \cdot C}{\epsilon_o \epsilon_r} \tag{7.3}$$

where ϵ_o is the permittivity of free space, ϵ_r is the relative permittivity of oxide between the two plates formed by two polysilicon layers, C is the capacitance. The transistor area is roughly calculated as $A_M = W_{id} \cdot L_{en}$, where W_{id} and L_{en} are the width and length of the transistor separately. Another two performance parameters are the input and output noise power (I_{noise} and O_{noise} correspondingly) over the frequency range of $0.1GHz$ to $10GHz$.

7.2.2 SVM regression model construction

Firstly, each design parameter has been assigned to their bounds and those define the closed hyperspace to be explored. Table 7.6 lists the upper and lower boundaries ('U. bound' and 'L. bound' correspondingly) for the design parameters. The modelling construction process follows the introduction in chapter 5.

TABLE 7.6: Bounds of the design parameters in the Colpitts RF filter case study.

Parameter	L_1	L_2	C_1	C_2	W_{id}	I_{bias}
U. bound	500	500	7	35	300	10
L. bound	300	300	3	6	50	5
Unit	μm	μm	pF	pF	μm	mA

7.2.2.1 Computational cost analysis of the standard grid search algorithm

The characteristics of the computational cost using the standard grid search algorithm of this case study have been presented in section 5.1. The experimental results are outlined here. To construct SVM regression models, design parameter intervals were uniformly sampled to provide 4096 samples to form the training data set in the design space. Each is simulated and the corresponding five performance parameter values are extracted. A set of design parameters and performance parameters are combined to form one training sample. The CPU time consumed

TABLE 7.7: Computational cost comparison in the Colpitts RF filter case study.

Performance model construction approach	Computational cost
Standard grid search	9 days 6 hours
Proposed method	1 day 8 hours

by the standard grid search method and the proposed computational cost aware algorithm are shown in table 7.7. As it can be seen, the proposed algorithm saves almost 8 days (190 hours) of CPU time for SVM training parameter determination. Both performance model construction approaches have returned similar results in terms of performance prediction accuracy as show in table 7.8. The C-γ pair are also similar except the filter output noise O_{noise}, which is not sensitive to the trade-off parameter C.

7.2.3 Performance optimisation

The goal of the optimisation is to obtain a set of design parameters that can achieve a design with centre frequency of $1GHz$ and a Q factor of 50 (i.e the bandwidth of $20MHz$). Simultaneously, the total area, input and output noise are minimised. Using the SVM regression models of the Colpitts RF filter, an experiment has been carried out to optimise the design's performance using a cost function which is a weighted sum of the total silicon area, input and output noise and squared error of the centre frequency and the Q factor. A typical regression curve of the pattern search optimisation has been shown in figure 7.10. As the

TABLE 7.8: RMS error of the SVM regression performance models constructed using the training parameters determined by the grid search and proposed methods and the corresponding C and γ values for the Colpitts RF filter.

	Performance prediction accuracy RMS error				
	F_c	Q	A(μm^2)	I_{noise}	O_{noise}
Grid search	$0.06GHz$	6.1	719	0.28E-7V	2.31E-9V
Proposed method	$0.1GHz$	6.1	739	0.27E-7V	2.3E-9V
	C and γ				
Grid search	67108, 0.0625	512, 1	32768, 0.0625	1024, 0.0625	64, 0.125
Proposed method	65535, 0.0625	512, 1	32768, 0.0537	1024, 0.0372	362, 0.0884

design approaches the optimum, the regression mesh size decreases as indicated by the right subplot in figure 7.10.

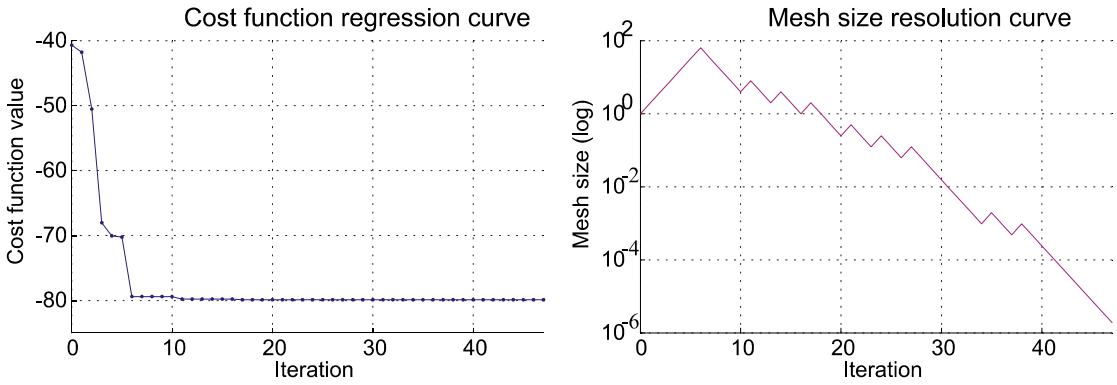


FIGURE 7.10: Typical cost function and mesh resolution regression curves in the pattern search optimisation

TABLE 7.9: Summary of the manual and optimised design results for the Colpitts RF bandpass filter.

Parameters		F_c	Q	A(μm^2)	I_{noise}	O_{noise}
Manual		$1.34GHz$	0.16	13500	1E-5V	5.8E-10V
Grid search	Optimised	$1.08GHz$	50	11500	1E-7V	6.8E-9V
	Verified	$1.01GHz$	48.2	12076	0.9E-7V	6.1E-9V
Proposed method	Optimised	$1.05GHz$	50	12180	1E-7V	7E-9V
	Verified	$0.99GHz$	49.6	12650	1E-7V	5.9E-9V

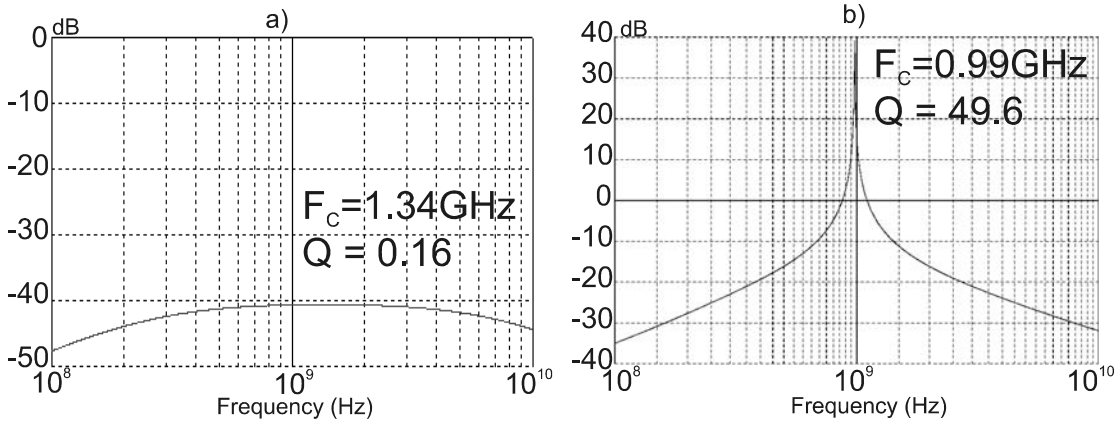


FIGURE 7.11: a) Manual design using design parameters derived from ideal model [65], b) optimised design derived by the proposed algorithm.

The optimisation results in table 7.9 show that the performance figures are significantly improved than those from a manual design obtained using ideal formulas. In figure 7.11, the results shown are from SPICE simulation using the design parameters derived from the optimisation. The optimisation is extremely fast, using only 269 seconds of CPU time, as the associated performance evaluation involves no simulations. To provide additional verification of the knowledge database accuracy, the best design has been fully simulated using WinSPICE. As shown in table 7.9, the performance figures calculated from the knowledge database are in excellent agreement with those obtained from full simulation-based verification.

7.3 Addtional experiments

The empirical expression on the computational cost is derived based on not only the two case studies in this chapter but also more experiments. Five public data sets from various open databases have been randomly selected and analysed using the standard grid search training parameter determination algorithm. Details of the data sets are summarised in table 7.10. These five data sets are obtained from various problems. The number of samples in each data set is different. Data set 1, 2, 3 and 4 have similar or less training data samples than the training data set used in the case studies while data set 5 is much larger. The features (one feature is equal to one design parameter) are similar or larger than the case studies. All the data sets and further relevant information can be found on the LibSVM website [116].

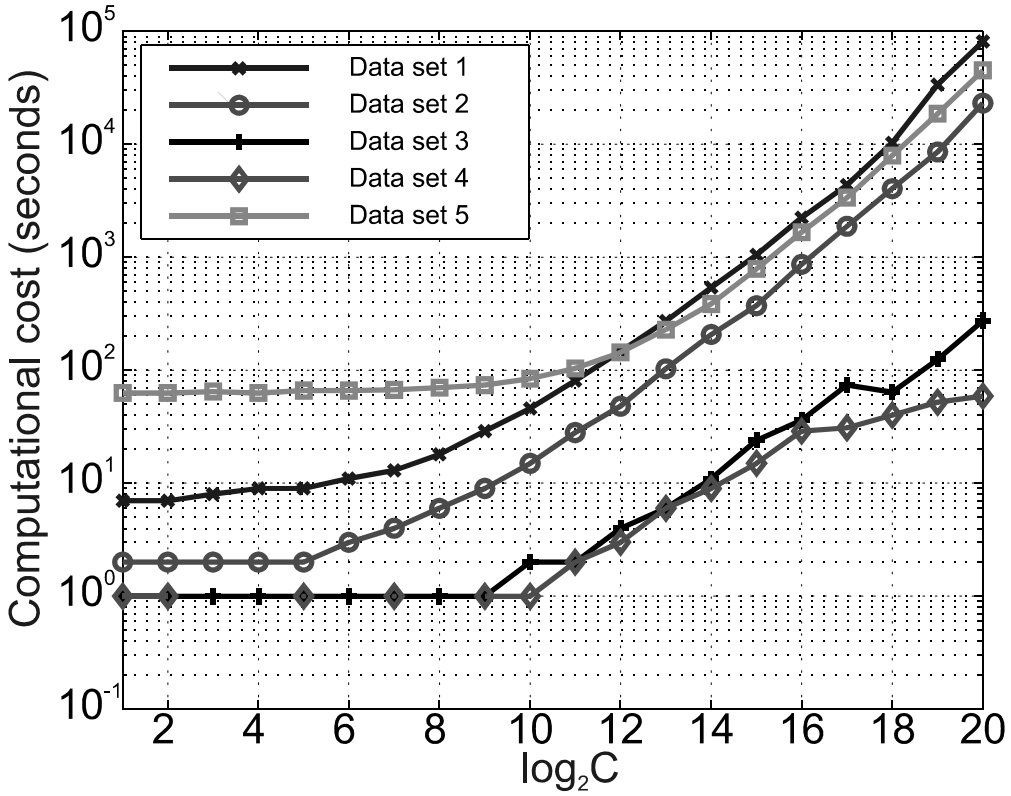


FIGURE 7.12: Influence of C on the computational cost of SVM regression training with different data sets.

TABLE 7.10: Information of the sample data sets.

Data set	# of samples	# of features	Description
1	3107	6	Election data including spatial coordinates on 3,107 US counties from CMU StatLib
2	1385	6	[124]
3	392	7	City-cycle fuel consumption in miles per gallon from UCI
4	506	13	Housing values in suburbs of Boston from UCI
5	8192	12	Predict a computer system activity from system performance measures from Delve of Toronto Uni

Running the SVM regression training on all the five data sets has confirmed the significant influence of parameter C on the computational cost. According to figure 7.12, the insensitive and sensitive ranges for the influence of C are observed in all the five data sets. The linear relationship between the computational cost and

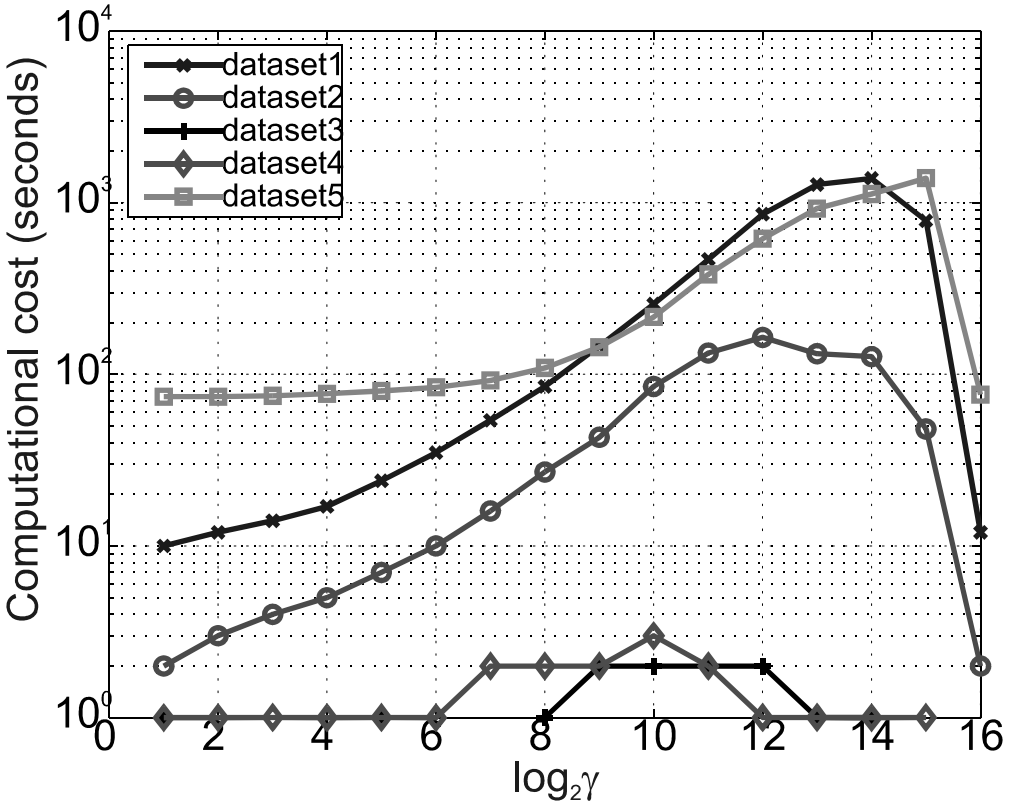


FIGURE 7.13: Influence of γ on the computational cost of SVM regression training with different data sets.

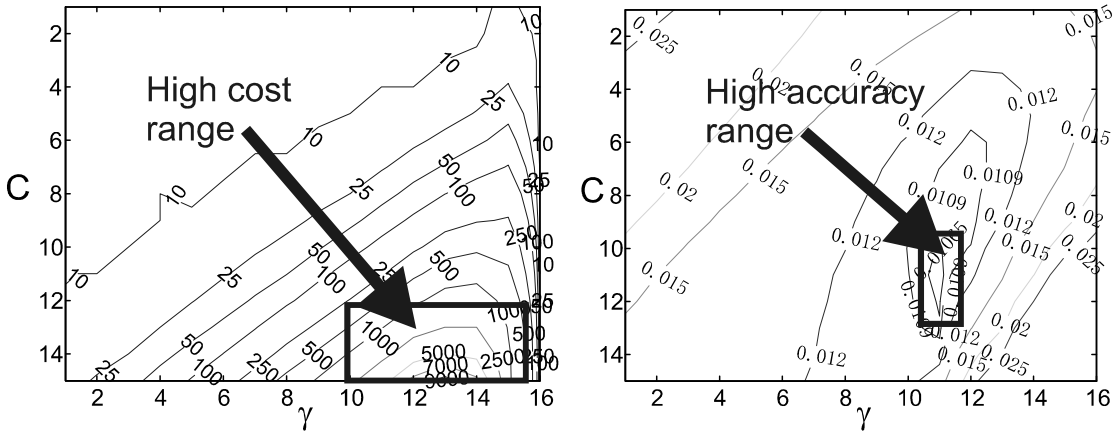


FIGURE 7.14: Computational cost (in seconds) and prediction contours using the standard grid search for SVM training of data set 1.

$log_2 C$ can be well described by equation 5.1. The computational cost within each data set can vary from tens to tens of thousands seconds for different values of C. So unless special attention is paid to the computational cost, any SVM training algorithm is likely to take much more CPU time than necessary.

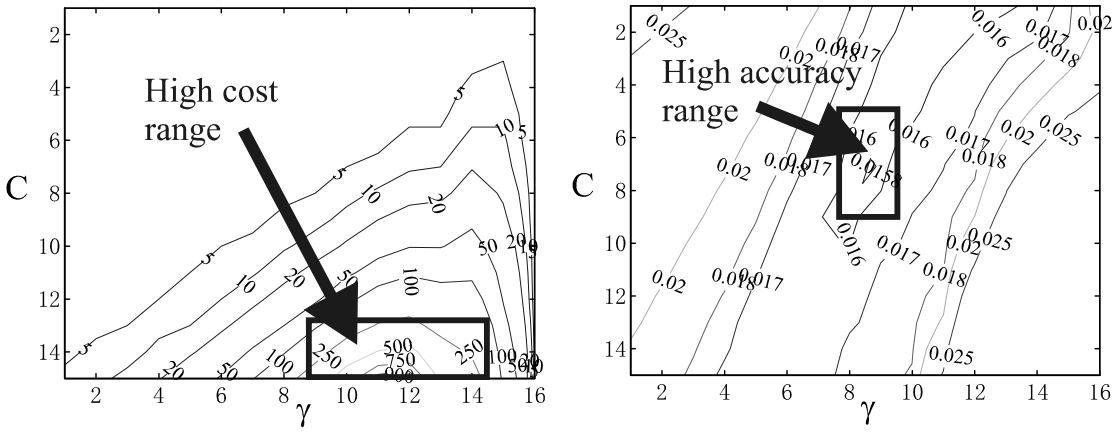


FIGURE 7.15: Computational cost (in seconds) and prediction contours using the standard grid search for SVM training of data set 2.

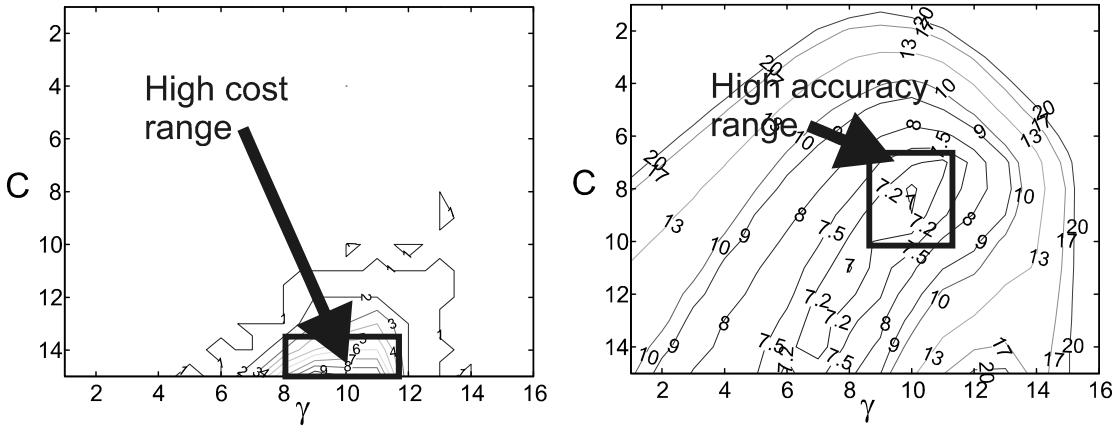


FIGURE 7.16: Computational cost (in seconds) and prediction contours using the standard grid search for SVM training of data set 3.

The corresponding non-linear influence of the RBF kernel parameter γ on the computational cost in the grid search training is shown in figure 7.13. The nonlinear relationship is clear. Even for data set 2 and 3 that have very small magnitude of computational cost, the sensitive range presents.

The computational cost and accuracy contours in the C-γ space of the grid search training algorithm using the five public data sets are shown in figures 7.14-7.18. The high computational cost and high prediction accuracy regions both present for each of the data set. However, the locations of these two regions are not necessarily the same or even overlap. For data set 2, 3 and 4 (figure 7.15, 7.16, 7.17), the high computational cost regions could not contribute the optimal model construction and only waste resources. The contours of data set 1 and 5 are similar to the case studies that some of the high computational cost regions that correspond to over

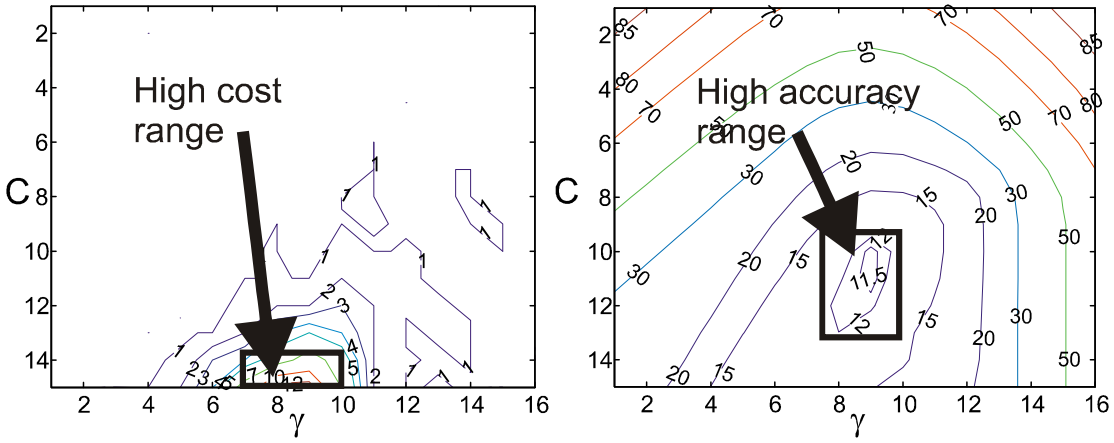


FIGURE 7.17: Computational cost (in seconds) and prediction contours using the standard grid search for SVM training of data set 4.

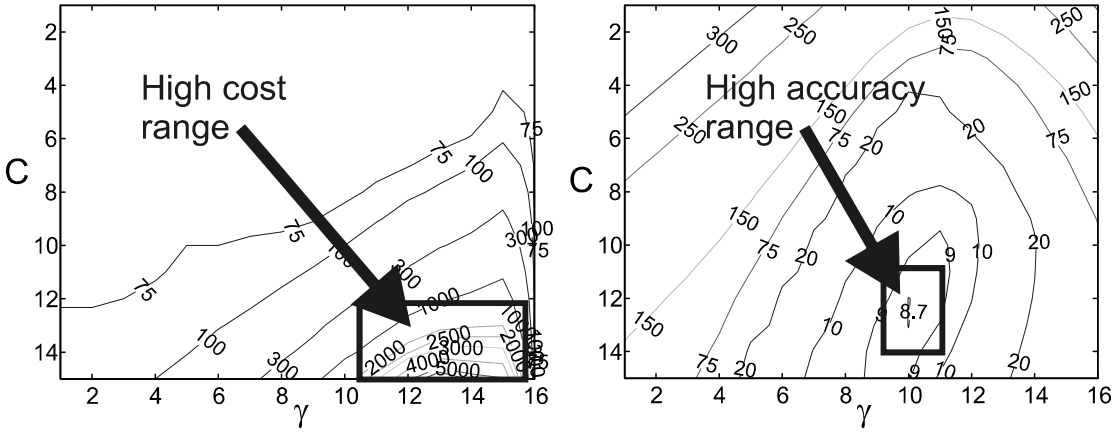


FIGURE 7.18: Computational cost (in seconds) and prediction contours using the standard grid search for SVM training of data set 5.

trained models have been wasted and should be avoided. The experiments support the conjecture that firstly, the computational cost and prediction accuracy in the grid-search algorithm follow the same pattern as described by the proposed models presented in chapter 5; secondly, the trade-off between the computational cost and prediction accuracy is essential for effective SVM regression training parameter determination.

7.4 Concluding remarks

This chapter presents the two case studies to verify the proposed AMS performance modelling and optimisation methodologies. The first case study is a 2^{nd}

order SDM that represents complex AMS designs that require effective performance modelling methods to explore the design space. The linearly graded and SVM regression SVM performance modelling methods are both applied for the performance model construction. Performance optimisation experiments show that optimised designs can be derived very quickly using the proposed performance modelling methodologies in the knowledge-based optimisation system. The second case study is a Colpitts RF analogue bandpass filter. This example represents RF designs that need highly expertise design knowledge and the standard filter modelling techniques are not applicable. The SVM regression models of the design have been constructed and performance optimisation experiments can been carried out to obtain optimal designs extremely quickly. With the verification by simulations, the models are shown to possess good prediction accuracy.

The other matter that has been demonstrated is the SVM training parameter determination using the computational cost aware algorithm that is developed on the base of analyzing the computational cost and prediction accuracy of the standard grid search algorithm. Apart from the case studies results, additional experiments have been carried out to provide more support for the empirical models. Results show excellent agreement with the proposed empirical expressions and models introduced in chapter 5.

Chapter 8

Conclusion and future work

This thesis presents a novel general performance modelling methodology applicable to a wide class of AMS designs. The modelling methodology is based on SVM classification and regression. It can help to develop a synthesis system for arbitrary AMS applications that may involve complex relationships between multi dimensional design and performance spaces. The proposed method needs minimum human intervention as it is fully automatic even for the SVM training parameter determination task, which normally is done manually as reported in most published applications of SVM techniques for AMS performance modelling. The contributions of the research are:

- A linearly graded SVM performance modelling methodology.
- An automated SVM regression performance model construction for AMS designs and a computational-cost aware SVM training parameter determination algorithm based on a critical analysis of the standard grid search algorithm.
- A knowledge-based performance optimisation system using SVM linearly graded and regression performance models.
- The proposed performance modelling and optimisation methodologies have been verified using two complex AMS designs with practical imperfections integrated.

The purpose of developing both the linearly graded and SVM regression performance modelling methodologies is to explore different possibilities of applying SVM techniques to AMS performance modelling problems. The linearly graded performance modelling method extends the classical 'good-bad' models and provides a graded design space exploration approach. The construction of SVM regression models is simpler compared to the linearly graded approach. This simplifies the development of the optimisation system. Because SVM techniques are applicable to arbitrary design modelling, the performance modelling methodologies developed in this research have a wide range of applications. Both methods have achieved highly accurate modelling results as demonstrated in the case studies.

The new computational-cost aware SVM training parameter determination algorithm has been developed based on the analysis of the standard grid search algorithm and has been demonstrated to provide almost an order of magnitude speed up for the task. It has been integrated into the the SVM regression model construction process.

The proposed performance modelling methodologies lend themselves naturally to effective performance optimisation of AMS designs. The proposed optimisation system uses automatically generated knowledge database rather than simulations and is therefore extremely fast.

The techniques described in chapter 4, 5 and 6 have been verified using three case studies of complex and highly nonlinear analogue and AMS systems. They are described using different HDLs including VHDL-AMS, MATLAB Simulink and SystemC. The case studies cover both system level and circuit level designs. Two main case studies, the signal delta modulator and silicon RF filter, comprise complex imperfection effects and are known to provide significant challenges and design difficulties in modern AMS and RF designs.

There are important points to make for future work. The most challenging problem in AMS synthesis designs is the difficulty introduced by large design space dimension. The difficulty in performance modelling expands significantly in terms of computational cost and modelling accuracy when the number of design parameters increases. One suggestion for future research is to explore the quality of the proposed performance modelling methodologies for larger scale problems. Complex AMS systems with detailed circuit level components such as phase locked loops can be good candidates for verification. A possible solution may lie on

the combination of both the linearly graded method and the computational cost aware SVM training parameter determination algorithm. Another research issue is to study further the characteristics of SVMs in the AMS performance modelling applications. Firstly, there may exist a rigid mathematical justification for the computational cost model of the standard grid search algorithm; secondly, the use of other kernel functions may be explored in performance model construction. The latter issue has been preliminarily studied in this work and some kernel functions have been compared with the RBF kernel using the SDM case study. Results have been published [49]. However, this can be taken further in a future research.

Appendix A

Journal paper submitted to IET CDS Proceeding

Computational-cost aware SVM-based performance modelling and optimisation for automated analogue and mixed-signal system design

Xianqiang Ren and Tom J Kazmierski
School of Electronics and Computer Science
University of Southampton, UK
Email: xr03r, tjk@ecs.soton.ac.uk

***Abstract*— This paper presents a novel approach to efficient performance modelling and optimisation which can be applied to both high-level and circuit-level analogue and mixed-signal (AMS) system synthesis. Support vector machine (SVM) regression is used to construct automatically general AMS performance models which lend themselves naturally to pattern-search optimisation by exploring the design space. Based on the trade-off between the computational cost and prediction accuracy, a new search algorithm has been proposed to determine accurate regression model parameters almost an order of magnitude faster than standard grid search thus saving days of CPU time in practical applications. Two case studies of complex analogue and mixed-signal systems have been presented to demonstrate that, once a support-vector knowledge data base has been created, the proposed approach can provide accurate and extremely fast performance estimation.**

I. Introduction

Knowledge-based analogue and mixed-signal (AMS) design automation systems, that have design expertise stored in a database for use as the intelligence source in designs, have been studied for over twenty years [1]. The main problem that knowledge-based synthesis systems face is their specialised nature due to the fact that specific and unique performance characteristics are required almost for every type of analogue systems. Traditionally knowledge-based synthesis systems have been applied only to a narrow class of circuits mostly op-amps [2] or voltage and current references [3]. In recent years attempts have been made to generalise knowledge-based approaches and make them applicable to wide classes of AMS designs [4]. Some landmark methodologies such as fuzzy logic [5] and neural network methods [6] have been applied in the research of this subject. Such systems automatically construct learning rules or neural networks to represent and reproduce the behaviour of target systems. However, these approaches are time consuming, labour intensive and also require a high degree of specialised knowledge. Support vector machines (SVMs) have already been suggested as an efficient means to model analogue performance, although restricted to the classical "good-bad" classification model construction [7], [8] and performance space regression [9]. SVM models tend to be more compact than corresponding neural network models because optimal support-vector regression functions include only a subset of the whole training data set, namely the support vector set [10]–[12]. It has been reported that SVM based approaches are able to find global optima in cases where neural networks converge to a local one [12]. Also, SVM methods seem to perform better in terms of training and prediction errors than simple linear least-square regression and posynomial models [9].

This paper presents two contributions. Firstly, we study the computational cost of constructing SVM models and propose a very efficient algorithm to determine SVM training parameters. The proposed algorithm is almost an order of magnitude faster than widely used Hsu's standard grid search method [13] and can save days of CPU time when applied to practical problems. Secondly, we extend the application of SVMs to automatic generation of a general AMS system performance optimiser. The methodology is fully automatic, does not require specialised knowledge, and regression databases are built from simulations without a need for manual adjustment. Two case studies, one a mixed-signal system and the other an analogue RF circuit, have been optimised using the proposed techniques. Results show that, once the knowledge data-base has been created, optimisations can be accomplished extremely quickly within a few minutes of CPU time. In contrast, most state-of-the-art publications [7], [8], [9] use manual and "try-it-out" methods for SVM training parameters determination which often lead to non-optimal results. In an earlier work [14] we demonstrated how the standard grid search method [13] can be applied to AMS performance modelling. However, the automated model construction process presented there is extremely time consuming where the process of generating an SVM regression knowledge database for a mixed-signal sigma-delta modulator took more than 5 days. The improved SVM training strategy proposed in this paper reduces the required CPU time for a similar sigma-delta system to about 30 hours and is therefore significant.

II. Computational cost of SVR performance model construction

A. Support vector regression

SVMs [10] were proposed originally in the context of machine learning for classification problems on large sets of data which have complex and unknown relationships between variables. The SVM approach can be used both for

classification and regression estimation [10]. Support vector regression (SVR) works as follows: suppose l observations in an n-dimensional input space $\Re^n$ are given. Let vector x denote one observation and y denote the corresponding real numbered output value generated by the unknown relationship to be modelled. $x_i, i \in [1, \cdots, l]$, is the i^{th} sample in the training data set.

In ϵ-SVR [10], the goal is to construct a regression function $f(x)$ that can generate y_i with at most deviation ϵ from the simulation results y_i^{sim} for all the training data samples. The structured risk minimisation (SRM) [11] strategy is used in the SVM training to construct the models. In a nutshell, the SRM builds the optimum regression function by minimising the prediction error and maximising the distance between the nearest samples and the regression function. In contrast to this strategy, traditional neural network technology uses empirical risk minimisation (ERM) [11] that minimises the approximation error only. An SRM-based training can be presented as the following constrained optimisation problem:

$$minimise \qquad \frac{1}{2}||\omega||^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*) \tag{1}$$

$$subject \;\; to \qquad \begin{cases} y_i - < \omega, x_i > -b \leq \epsilon + \xi_i \\ < \omega, x_i > +b - y_i \leq \epsilon + \xi_i^* \end{cases} \tag{2}$$

where $\omega \in \Re^n$ is the norm vector of the regression function f, b is the bias vector of the regression function, ξ_i and ξ_i^* are the error terms introduced to cope with the mis-predicted samples by the regression function, the constant C is the trade-off parameter that determines the weight of the error terms in the optimisation problem, operator $<,>$ is the dot product. The training error minimisation is included in equation (1) maximisation of the separation distance is achieved by minimising the Euclidean norm $||\omega||^2$.

Support vector regression maps the x_i vectors into the *feature space* via a function Φ. This mapping is usually implemented in an implicit form since it is the following kernel function:

$$K(x_i, x_j) \;=\; < \Phi(x_i), \Phi(x_j) > \;\; i, j \in l \tag{3}$$

that needs to be calculated not the Φ function itself. Several kernel functions have been proposed [10], [11], [13]. We have selected the radial basis function (RBF) kernel for its capability to handle nonlinear relationships and easy tuning:

$$K(x_i, x) = e^{-\gamma||x_i - x||^2}, \;\; \gamma > 0 \tag{4}$$

The optimal regression function found by an SVR includes only a subset of the whole training data set called the support vector (SV) set. In this way, SVM models are usually smaller than the corresponding neural network models, which use all the training data points.

B. Computational cost analysis

The computational cost analysis outlined below has been carried out using LibSVM [13], a popular SVM trainer [7], [8], [14]–[16] and a training parameter determination algorithm e.g the standard grid search algorithm. In the search process, for each training parameter vector, in the case of the RBF kernel the C-γ pair, the SVM trainer iteratively solves equation (1) on a selected subset, called the *working set*, of the training data set in the feature space using Lagrangian optimisation. Before an iteration starts, samples in the training data set are mapped to the feature space by the kernel function in equation (3). Then, a working set is selected from the training set based on the gradients at the mapped sample points in the feature space. The Lagrangian coefficients of the working set, which lie in the range [0,C], are found by solving the optimisation problem defined by equations (1) and (2). As we show below, parameter C influences the computational cost because it determines the width of the intervals where the Lagrangian coefficients lie. As the kernel is involved at the mapping stage, its nonlinearity determines the gradients of the training samples in the feature space. So it can be deduced from this that kernel parameters affect the convergence speed of the solving process by influencing the working set selection.

Experiments show that parameter C affects the computational cost of training parameter determination very significantly, by up to three orders of magnitude as demonstrated in figures 5, 9 a) and 11. The computational cost increases monotonically with the increment of C. The RBF kernel parameter γ has a nonlinear influence on the computational cost as indicated in figures 5, 9 c) and 12. In our case studies the regions where γ provides optimal mapping also coincide with high computational cost especially when C is larger than the optimum value. The computational cost as a function of C and γ in two-dimensional contour diagrams for all the experiments have been shown in subplots a) of figures 6, 10, 13, 14, 15, 16 and 17. Corresponding prediction accuracy contours are also shown in the above figures. It is clear that in many cases it should be possible to find the best C-γ pair by avoiding the high computational cost regions. A new computational-cost aware algorithm is proposed in the next section as an alterative to the standard grid search.

C. Computational cost aware SVR performance model construction

The observations discussed in the previous section are critical in the development of the improved algorithm presented below. In the high computational cost area, where the optimum C-γ pair usually lies, small accuracy improvements are achieved at high computational cost expense. For example, a small improvement of 0.2 in the RMS error of the RF filter Q factor at $log_2 C$=9 takes 1.5 hours of CPU time as shown in figure 5 b). The very high computational cost range outside the high prediction accuracy range should be avoided because it only consumes computational resources but cannot contribute to finding the optimum solution.

Let us now define a measure of the accuracy improvement effectiveness:

$$E_{ipv} = \frac{\Delta A}{\Delta T} \tag{5}$$

where ΔA is the accuracy improvement, ΔT is the extra computational cost spent to achieve the improvement at the current C-γ pair. Using the above effectiveness measure, a new heuristic gradient-based C-γ determination algorithm can be proposed. The algorithm uses the accuracy improvement effectiveness to dynamically modify parameters C and γ as follows:

$$C_{n+1} = k_C^{sgn(\Delta E_{ipv})} C_n \; n = 1, 2, \cdots \tag{6}$$

$$\gamma_{n+1} = k_\gamma^{sgn(\Delta Eipv)} \gamma_n \; n = 1, 2, \cdots \tag{7}$$

where n is the iteration index, k_C and k_γ are the stepsize refining factors for parameters C and γ correspondingly, and ΔE_{ipv} is the effectiveness change at the current iteration. The algorithm repeatedly involves the SVM trainer, calculates the accuracy improvements for equations 6, 7 and terminates when E_{ipv} becomes higher than the pre-defined maximum effectiveness and a minimum required training accuracy has been achieved.

III. AMS PERFORMANCE OPTIMISATION USING SVR MODELS

The performance model construction technique outlined above has been implemented in MATLAB. The flow chart of the entire model construction system is presented in figure 1. Unlike other similar systems [7]–[9], which need the designer's interaction to determine SVR training parameters, this approach is fully automatic and therefore less labour intensive. The simulator can be behavioural one or circuit-level as required. We have used WinSpice [17] in the RF filter case study and SystemC [18] for the mixed-signal sigma-delta example presented in section IV. The performance model construction process creates a training data set by running multiple simulations and, as a result, builds regression models for further usage in performance optimisations. A uniformly sampling scheme has been used to build the design space. The number of samples is arbitrarily determined as the relationship between the design and performance spaces is unknown beforehand. However, the accuracy of the SVR model is verified at the testing stage after the training and should the regression accuracy be found insufficient, SVR models can be rebuilt using a larger number of samples.

Each SVR model performs a regression functional mapping from the design space to a single-dimensional performance parameter space. Together, all the SVR models can be viewed as a multi-dimensional mapping to the performance space. Once SVM regression models are constructed, they form a knowledge database represented by the cylinder in figure 1, from which the performance can be estimated at an arbitrary point in the design space without resorting to further simulations.

The SVR models are the intelligence source from which the behaviour of the design can be predicted extremely quickly. Even for very complex problem, once a regression knowledge database is constructed, performance figures for any design points in the design space can be obtained at negligible computational cost. Therefore, performance optimisation can be carried out very efficiently. The optimiser takes an initial design as the starting point in the optimisation process and a set of performance constraints. An arbitrary cost function can be employed. In our case studies the cost function has the form of linear combination of weighted performance characteristics and squared deviations:

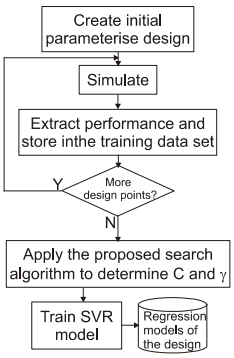


Fig. 1. Flow chart of the SVM regression model construction.

$$V_{cost} = \sum_{i=1}^{m} w_i P_i + \sum_{i=1}^{n} w_i (P_i - P_i^{spec})^2 \tag{8}$$

where w_i is the weight coefficient, P_i is the predicted performance value and P_i^{spec} is the performance specification value. The combination of performance characteristics and their squared deviations can explore the performance space for both open constraints such as "area as small as possible" and specific constraints such as "Gain is 65dB". Scalar weights w_i balance the competing objectives.

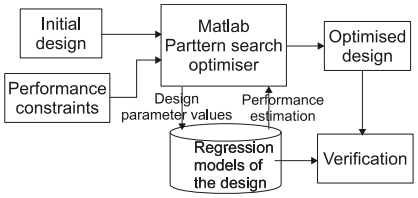


Fig. 2. Performance optimisation design flow using SVM regression models.

Given required performance criteria, the best design point can be found using e.g. a mixture of heuristic and gradient-based optimisation. The optimal design that provides a minimum cost function value is then verified by simulations. This methodology is represented by the chart in figure 2. In the case studies, the standard MATLAB pattern search algorithm has been used as the optimisation method.

IV. Case studies

Two case studies have been analysed and their performance optimised to validate the proposed methodology. Each includes a specification of the design and performance spaces, the SVR model construction analysis and performance optimisation experiments. The experiments were carried out on a Pentium IV 2.8 GHz Windows PC.

A. Colpitts radio-frequency filter

The performance of a second-order RF bandpass filter based on a Colpitts oscillator with a very low Q factor has been optimised using the technique proposed above. Colpitts-type LC oscillators, like the one shown in figure 3, are used in integrated circuit bandpass filter designs [19], [20]. The MOS transistor provides the positive feedback mechanism for Q-enhancement to compensate inductor losses. Standard filter synthesis techniques cannot be used in this application [19] due to its non-linear nature, with significant losses and parasitics in the RF operating ranges, especially those in the spiral inductor and transistor. The goal in this case study is to model the relationship between the design and performance parameters and to demonstrate how an optimal design can be extracted from this relationship according to a set of performance constraints.

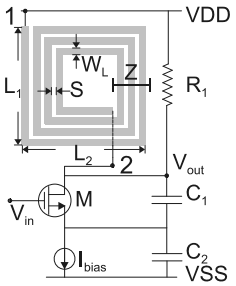


Fig. 3. Schematic of the Colpitts bandpass filter with a highlighted on-chip spiral inductor structure.

1) Inductor model: On-chip planar rectangular spiral inductors can be modelled by five structural parameters, as illustrated in figure 3. They include the length of the first and second wire segments (L_1 and L_2 separately), wire width (W_L), space between wires (S) and number of turns (Z). Two metal layers are needed by the planar spiral inductor, where metal 1 is used as the spiral wire and metal 2 as the underpass to connect the inside terminal of the inductor with external connections. The connections of the outside and inside terminals of the inductor have been labelled as '1' and '2' in the figure. The spiral inductor can be accurately modelled using the lumped π-model shown in figure 4 a) [21]. In the model, L_{total} represents the actual total inductance; R_s is the series resistance associated with the metal wires between the connection point 1 and 2; C_s indicates the capacitive coupling of the overlap between the spiral and the underpass; C_{ox} is the oxide capacitance between the spiral and the silicon substrate; the capacitance and resistance of the substrate are modelled by the network formed by C_{si} and R_{si}. The model provides accurate prediction of the inductor behaviour over a wide range of operating frequencies, layout dimensions and process parameters [21]. A set of scalable CMOS design rules based

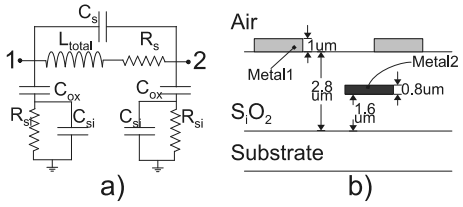


Fig. 4. a) the lumped π model of the on-chip spiral inductor; b) cross-section view of the spiral inductor.

on the TSMC $0.35\mu m$ process [22] with two polysilicon and four metal layers has been employed to obtain an industrial level accuracy. A cross-section view of the inductor is shown in figure 4 b). A MATLAB program has been developed based on Greenhouse's method [23] for automatic calculation of the inductance. The total inductance includes the self-inductance and mutual inductance between parallel wire segments. The method offers superior accuracy over many empirical formulas [21].

2) Design and performance space: The design space is a six-dimensional hyperspace formed by the following parameters within their corresponding intervals: L_1, L_2, C_1, C_2, I_{bias} and W_{id} (width of the NMOS transistor). L_1 and L_2 are selected out of the five structural parameters associated with the spiral inductor because they have the most significant influence on the inductor model while other three W_L, S and Z are of minor importance. The performance space is formed by five performance parameters. The centre frequency (F_c) and the passband bandwidth (determined by the Q factor) are both included in the performance space as the two key measures of the filter quality. As both on-chip capacitors and the inductor are area consuming, the area of the design has been included in the performance parameters and calculated as:

$$A_{total} = A_L + A_C + A_M \tag{9}$$

where A_{total} represents the total area of the design consisted of area of the inductor A_L, two capacitors A_C and the NMOS transistor A_M. A_L is the rectangular area formed by $L_1 \times L_2$; area of the capacitors are calculated by:

$$A_C = \frac{t_{ox} \cdot C}{\epsilon_o \epsilon_r} \tag{10}$$

where ϵ_o is the permittivity of free space, ϵ_r is the relative permittivity of oxide between the two plates formed by two polysilicon layers, C is the capacitance. The transistor area is roughly calculated as $A_M = W_{id} \cdot L_{en}$, where W_{id} and L_{en} are

the width and length of the transistor separately. Another two performance parameters are the input and output noise power (I_{noise} and O_{noise} correspondingly) over the frequency range of $0.1GHz$ to $10GHz$.

3) Grid search computational cost: The influence of parameter C on the computation cost of training parameter determination and prediction accuracy is shown in figure 5. In this example, the computational cost can vary by up to

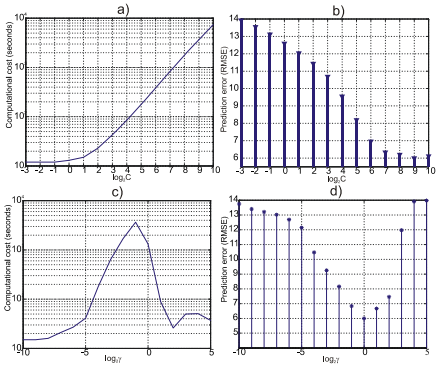


Fig. 5. Influence of C on the computational cost when γ=0.5 in a) and prediction accuracy in b) for the the Q factor of the Colpitts RF filter; c) and d) are the computational cost and prediction accuracy diagrams showing the influence of γ when C is a constant.

three orders of magnitude for both C and γ. The computational cost contours for the complete grid search process are shown in figure 6 a). The highest computational cost area has been highlighted by a solid-line rectangle. The corresponding

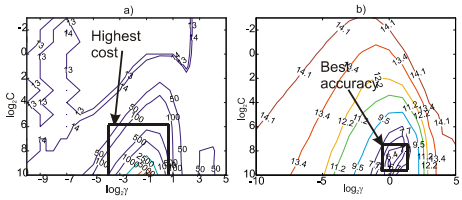


Fig. 6. a) Computational cost contours (in seconds) of C-γ grid search process for the Q factor of the Colpitts RF filter, b) Corresponding accuracy performance of the models (RMS error) for the Q factor.

prediction accuracy contours are in figure 6 b). The optimum C-γ area has been highlighted. It is clear that the standard grid search algorithm wastes computational resources as it is unnecessarily scanning areas of very high cost and sub-optimal accuracies.

4) SVR model construction: Firstly, each design parameter has been assigned to an interval and this defines the closed hyperspace to be explored. Table I shows the upper and lower boundaries ('U. bound' and 'L. bound' correspondingly) for the design parameters.

TABLE I
DESIGN PARAMETER BOUNDS.

Parameter	L_1	L_2	C_1	C_2	W_{id}	I_{bias}
U. bound	500	500	7	35	300	10
L. bound	300	300	3	6	50	5
Unit	μm	μm	pF	pF	μm	mA

To construct SVR models, design parameter intervals were uniformly sampled to provide 4096 samples to form the training data set in the design space. The CPU times consumed

TABLE II
COMPUTATIONAL COST COMPARISON IN THE COLPITTS FILTER CASE STUDY.

Performance model construction approach	Computational cost
Standard grid search	9 days 6 hours
Proposed method	1 day 8 hours

by the standard grid search method and the proposed algorithm are shown in table II. As it can be seen, the proposed algorithm saves almost 8 days (190 hours) of CPU time. Both performance model construction approaches have returned similar results in terms of performance prediction accuracy as show in table III. The C-γ pair are also similar except the filter output noise O_{noise}, which is not sensitive to the trade-off parameter C.

TABLE III
SVR PERFORMANCE MODEL PREDICTION RMS ERROR AND THE CORRESPONDING C AND γ VALUES FOR THE COLPITTS RF FILTER.

	Performance prediction accuracy RMS error				
	F_c	Q	A(μm^2)	I_{noise}	O_{noise}
Grid search	$0.06GHz$	6.1	719	0.28E-7V	2.31E-9V
Proposed method	$0.1GHz$	6.1	739	0.27E-7V	2.3E-9V
	C and γ				
Grid search	67108, 0.0625	512, 1	32768, 0.0625	1024, 0.0625	64, 0.125
Proposed method	65535, 0.0625	512, 1	32768, 0.0537	1024, 0.0372	362, 0.0884

5) Performance optimisation: The goal of the optimisation is to obtain a set of design parameters that can achieve a design with centre frequency of $1GHz$ and a Q factor of 50 (i.e the bandwidth of $20MHz$). Simultaneously, the total area, input and output noise are minimised. Using the SVR models of the Colpitts filter, an experiment has been carried out to optimise the design's performance using a cost function which is a weighted sum of the total silicon area, input and output noise and squared error of the centre frequency and the Q

factor. The optimisation results in table IV show significantly

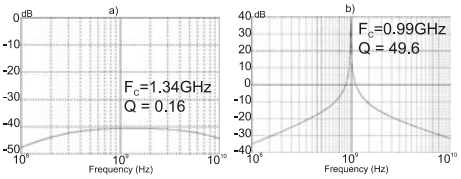


Fig. 7. a) Manual design using design parameters derived from ideal model [20], b) Optimised design derived by the proposed algorithm.

TABLE IV
SUMMARY OF THE MANUAL AND OPTIMISED DESIGN RESULTS FOR THE COLPITTS FILTER.

Parameters		F_c	Q	A(μm^2)	I_{noise}	O_{noise}
Manual		$1.34GHz$	0.16	13500	1E-5V	5.8E-10V
Grid search	Optimised	$1.08GHz$	50	11500	1E-7V	6.8E-9V
	Verified	$1.01GHz$	48.2	12076	0.9E-7V	6.1E-9V
Proposed method	Optimised	$1.05GHz$	50	12180	1E-7V	7E-9V
	Verified	$0.99GHz$	49.6	12650	1E-7V	5.9E-9V

improved performance figures than those for the manual design obtained using ideal formulas. Simulation results are shown in figure 7. The optimisation is extremely fast, using only 269 seconds of CPU time, as the associated performance evaluation involves no simulations. To provide additional verification of the knowledge database accuracy, the best design has been fully simulated using WinSPICE. As shown in table IV, the performance figures calculated from the knowledge database are in excellent agreement with those obtained from full simulation-based verification.

B. 2^{nd} order sigma-delta modulator

Here the proposed algorithm has been used to optimise the performance of a mixed-signal 2nd order sigma-delta modulator (SDM). A circuit-level model, based on a typical 2nd order SDM [24] but with added imperfections, has been developed in SystemC for time-domain simulations. The imperfections of the SystemC SDM model include, as shown in figure 8: the clock jitter, thermal noise, opamp non-linear DC transfer characteristic, opamp unity-gain bandwidth, slew rate, quantizer offset, quantizer hysteresis and integrator DC saturation. The imperfections not only degrade the system's performance but also require complex numerical simulations and results post-processing for accurate performance prediction.

1) SVR model construction: The selected design parameters are the amplifier gains (a_1, a_2, a_3, b_1, b_2 in figure 8) and the input amplitude, assuming sinusoidal excitation. A MATLAB script extracts the following performance parameters from SystemC time-domain simulations: the signal-to-noise ratio (SNR), maximum input signal amplitude range (SR) at peak SNR, maximum dynamic range of integrator 1 and integrator 2 outputs INT_1 and INT_2 correspondingly as well as the

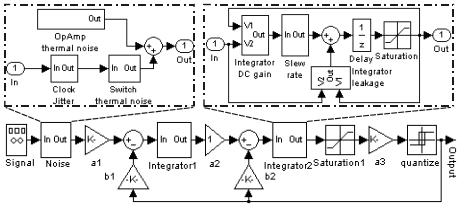


Fig. 8. 2^{nd} order SDM with imperfections.

SNR degradation ratio (SNRDR). The SNR is calculated by means of an FFT analysis on time-domain waveforms and the SNRDR is measured as the average slope of the SNR curve obtained from multiple simulations using input signals of varying amplitude.

To construct the SVR model each design parameter range was uniformly sampled to provide 3125 samples in the design space. The SVR model construction process is time consuming, especially in this case, as some performance parameters need to be calculated from multiple simulation runs. The

TABLE V
COMPUTATIONAL COST COMPARISON IN THE 2^{nd} ORDER SDM CASE STUDY.

Performance model construction approach	Computational cost
Standard grid search	6 days 1 hour
Proposed method	1 day 4 hours

CPU times used by the standard grid search algorithm and the proposed approach are listed in table V. The proposed algorithm saves nearly 5 days of CPU time in this case. Table VI shows that there are no significant difference in the performance prediction accuracy between both algorithms.

TABLE VI
SVR PERFORMANCE MODEL PREDICTION RMS ERROR AND THE CORRESPONDING C AND γ VALUES FOR THE 2^{nd} ORDER SDM.

	Performance prediction accuracy RMS error				
	SNR_{dB}	SNRDR	SR	INT_1	INT_2
Grid search	12.7	9	5.75E-3	3.66E-3	3.82E-3
Proposed method	12.9	9	6.32E-3	5.44E-3	4.11E-3
	C and γ				
Grid search	512, 4.6	1024, 6	1, 5.28	36.7, 1	8, 2.64
Proposed method	362, 4	1024, 5.6	1, 4	36.7, 1.52	90.5, 1

2) Grid search computational cost: The influence of C on the computational cost of training parameter determination and prediction accuracy is shown in figure 9. Here again, the computational cost varies by almost three orders of magnitude

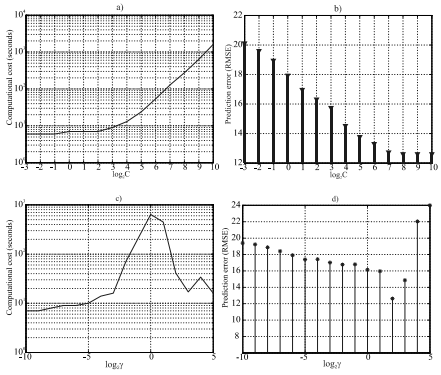


Fig. 9. The influence of C on the computational cost when γ=0.5 in a) and prediction accuracy in b) for the the SNR of the 2^{nd} order SDM. c) and d) are the computational cost and prediction accuracy diagrams when C is a constant while γ is scanned.

with the changes of C and γ. The combined computational cost contours of the complete grid search process are shown in figure 10 a). The corresponding prediction accuracy diagram

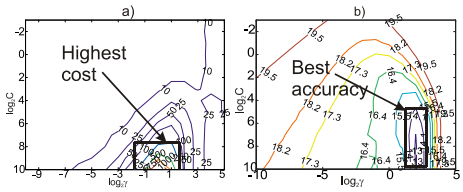


Fig. 10. a) Computational cost contours (in seconds) of C-γ grid search process for the SNR of the 2^{nd} SDM, b) Corresponding accuracy performance of the models (RMS error in dB).

is in figure 10 b).

3) Performance optimisation: Two cost functions with different weighted sums of performance characteristics were used in two performance optimisation experiments: Exp I and Exp II. Cost function I gives more weighting to frequency-related performance characteristics namely SNR and SNRDR, whereas cost function II gives preference to DC-related characteristics. Optimisation results are summarised in table VII. The first row in table VII specifies the performance characteristics obtained using a standard SDM design procedure [25] so that they can be compared with the best designs found through the optimisation using the automatically generated SVR-based knowledge database.

The results in table VII show that the standard grid search approach determines values for C and γ such that they provide slightly more accurate predictions than the new search approach. However, the marginal prediction performance degradation is compensated by a huge computational cost saving. Optimisation is extremely fast for both of these two experiments. Exp I takes 82 seconds and exp II - 121 seconds of CPU time.

TABLE VII
SUMMARY OF THE STANDARD AND OPTIMISED DESIGN RESULTS FOR THE 2^{nd} ORDER SDM.

Parameters			SNR_{dB}	SNRDR	SR	INT_1	INT_2
Standard design using non-ideal SDM model			64.2	10.73	0.7	0.29	0.61
Grid search	Exp I	Optimised	84.1	0.01	0.74	1.48	1.51
		Verified	84.1	0.01	0.75	1.5	1.5
	Exp II	Optimised	69.9	7.47	0.83	0.2	0.3
		Verified	69.2	9.75	0.8	0.29	0.34
Proposed method	Exp I	Optimised	81.1	0.8	0.82	1.44	1.47
		Verified	84	0.2	0.8	1.5	1.5
	Exp II	Optimised	71.8	7.51	0.86	0.14	0.21
		Verified	68	10.9	0.85	0.2	0.24

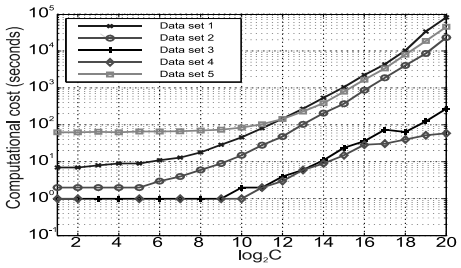


Fig. 11. Influence of C on the computational cost of SVR training with different testing data sets.

V. CONCLUSION

The main contribution of this paper is the computational cost analysis of the standard grid search algorithm for automatic SVM training. A significant dependence of up to three orders of magnitude of the computational cost on the trade-off parameter C has been observed and confirmed by experiments. The influence of the RBF kernel function parameter γ on the computational cost is also significant, up to two orders of magnitude as demonstrated by experiments. Based on the computational cost analysis, a new SVR training parameter determination algorithm has been proposed and implemented. The algorithm provides up to 7 times faster parameter determination compared with the standard and widely used grid search approach. This difference is very significant as it can save days of CPU time when applied to practical problems. The second contribution is a methodology for automatic construction of a knowledge-based performance optimiser for arbitrary analogue or mixed-signal systems. A general AMS design performance optimiser has been developed to implement the proposed methodology and the training parameter

determination algorithm. The significance of these contributions is that knowledge databases for analogue or mixed-signal system performance optimisation can be created efficiently in a fully automatic way, without a need for manual intervention or adjustment. The performance optimisation methodology presented here relies on regression databases automatically constructed from simulations.

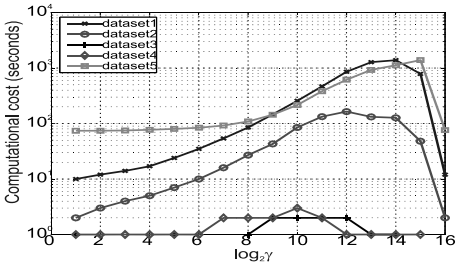


Fig. 12. Influence of γ on the computational cost of SVR training with different testing data sets.

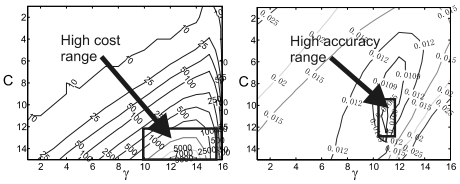


Fig. 13. Computational cost and prediction contours in the grid search of dataset 1.

VI. Appendix

To further confirm our computational cost analysis, five public data sets from various open databases have been randomly selected and analysed. Details are summarised in table VIII. All the data sets and further relevant information can be found on the LibSVM website [13].

TABLE VIII
Information of the sample data sets.

Data set	# of samples	# of features	Description
1	3107	6	Election data including spatial coordinates on 3,107 US counties from CMU StatLib
2	1385	6	[26]
3	392	7	City-cycle fuel consumption in miles per gallon from UCI
4	506	13	Housing values in suburbs of Boston from UCI
5	8192	12	Predict a computer system activity from system performance measures from Delve of Toronto Uni

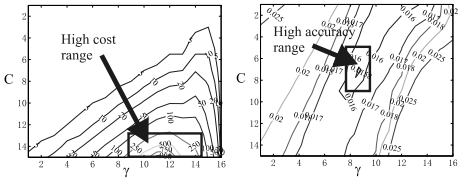


Fig. 14. Computational cost and prediction contours in the grid search of dataset 2.

Running the SVR training on all the five data sets has confirmed the significant influence of parameter C on the computational cost, as shown in figure 11. The computational

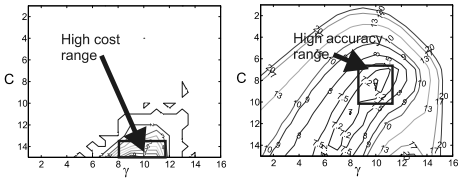


Fig. 15. Computational cost and prediction contours in the grid search of dataset 3.

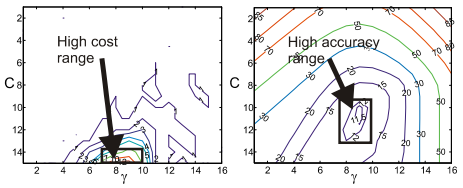


Fig. 16. Computational cost and prediction contours in the grid search of dataset 4.

cost within each data set can vary from tens to tens of thousands seconds for different values of C. So unless special attention is paid to the computational cost dependency on the parameter C, any SVM training algorithm is likely to take much more CPU time than necessary. The corresponding non-linear influence of the RBF kernel parameter γ on the computational cost in the grid search training is shown in figure 12.

The computational cost and accuracy contours of the grid search training algorithm using the five public data sets are shown in figures 13-17. They all follow the same pattern and further confirm the computational cost analysis in section II-B and support out conjecture that the trade-off between the computational cost and prediction accuracy is essential for effective SVR training parameter determination.

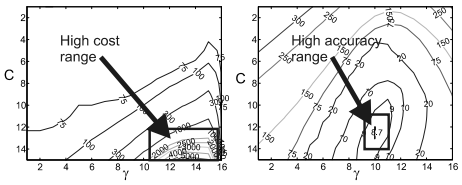


Fig. 17. Computational cost and prediction contours in the grid search of dataset 5.

REFERENCES

[1] G. G. E. Gielen and R. A. Rutenbar, "Computer-aided design of analog and mixed-signal integrated circuits," *Proceedings of the IEEE*, vol. 88, no. 12, pp. 1825–54, Dec 2000.

[2] F. El-Turky and E. Perry, "BLADES: an artificial intelligence approach to analog circuit design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 8, Issue 6, pp. 680 – 692, June 1989.

[3] M. Degrauwe, O. Nys, E. Dijkstra, J. Rijmenants, S. Bitz, B. Goffart, E. Vittoz, S. Cserveny, C. Meixenberger, G. V. der Stappen, and H. Oguey, "IDAC: an interactive design tool for analog CMOS circuits," *IEEE Journal of Solid-State Circuits*, vol. 22, Issue 6, pp. 1106 – 1116, Dec 1987.

[4] B. Antao and A. Brodersen, "ARCHGEN: Automated synthesis of analog systems," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 3, Issue 2, pp. 231 – 244, June 1995.

[5] A. Torralba, J. Chavez, and L. Franquelo, "FASY: a fuzzy-logic based tool for analog synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15 Issue 7, pp. 705 – 715, July 1996.

[6] G. Wolfe and R. Vemuri, "Extraction and use of neural network models in automated synthesis of operational amplifiers," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, Issue 2, pp. 198 – 212, Feb 2003.

[7] F. D. Bernardinis, M. I. Jordan, and A. S. Vincentelli, "Support vector machines for analog circuit performance representation," *Proceedings of Design Automation Conference*, pp. 964 – 969, Jun 2-7 2003.

[8] M. Ding and R. Vemuri, "A combined feasibility and performance macromodel for analog circuits," *Proceedings of Design Automation Conference*, pp. 63 – 68, June 13-17 2005.

[9] T. Kiely and G. Gielen, "Performance modeling of analog integrated circuits using least-squares support vector machines," *Proceedings of Design, Automation and Test in Europe Conference and Exhibition*, vol. 1, pp. 448 – 453, Feb 16-20 2004.

[10] V. N. Vapnik, *The Nature of statistical learning theory*. Springer-Verlag New York Inc, 2001.

[11] B. Scholkopf, "Statistical learning and kernel methods," Microsoft reasearch limited, Tech. Rep., Feb 2000. [Online]. Available: http://research.microsoft.com/ bsc

[12] A. Smola and B. Scholkopf, "A tutorial on support vector regression," NeuroCOLT2 technical report series, NC2-TR-1998-30, Tech. Rep., Oct 1998.

[13] C.-C. Chang and C.-J. Lin, *LibSVM: a library for support vector machines*, 2001, software available at http://www.csie.ntu.edu.tw/ cjlin /libsvm.

[14] X. Ren and T. Kazmierski, "Behavioral-level performance modeling of analog and mixed-signal systems using support vector machines," *Proceedings of Behavioral Modeling and Simulation Workshop*, pp. 28 – 33, Sept 2006.

[15] R.-E. Fan, P.-H. Chen, and C.-J. Lin, "Working set selection using second order information for training SVM," *Journal of Machine Learning Research 6*, pp. 1889–1918, Nov 2005.

[16] P.-H. Chen, R.-E. Fan, and C.-J. Lin;, "A study on SMO-type decomposition methods for support vector machines," *IEEE Transactions on Neural Networks*, vol. 17, Issue 4, pp. 893 – 908, July 2006.

[17] M. Smith, *WinSpice3 User's manual*, June 10 2004, software available at http://www.ousetech.co.uk/winspice2/.

[18] IEEE, "IEEE standard SystemC language reference manual," March 31 2006.

[19] D. Li and Y. Tsividis, "Active LC filters on silicon," *IEE Proceedings of Circuits, Devices and Systems*, vol. 147, Issue 1, pp. 49 – 56, Feb 2000.

[20] F. A. Hamid, "Architectural synthesis of analogue filters from behavioural VHDL-AMS descriptions," Ph.D. dissertation, School of Electronics and computer science, University of Southampton, 2004.

[21] C. Yue and S. Wong, "Physical modeling of spiral inductors on silicon," *IEEE Transactions on Electron Devices*, vol. 47, Issue 3, pp. 560 – 568, March 2000.

[22] MOSIS, "MOSIS scalable CMOS (SCMOS)," The MOSIS Service, Tech. Rep., Oct 4 2004. [Online]. Available: http://www.mosis.org/Technical/Designrules/scmos

[23] H. Greenhouse, "Design of planar rectangular microelectronic inductors," *IEEE Transactions on Parts, Hybrids, and Packaging*, vol. 10, Issue 2, pp. 101 – 109, Jun 1974.

[24] H. Zare-Hoseini, I. Kale, and O. Shoaei, "Modeling of switched-capacitor delta-sigma modulators in SIMULINK," *IEEE Transactions on Instrumentation and Measurement*, vol. 54, Issue 4, pp. 1646 – 1654, Aug 2005.

[25] M. Safi-Harb and G. Roberts, "Low power delta-sigma modulator for ADSL applications in a low-voltage CMOS technology," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 52, Issue 10, pp. 2075 – 2089, Oct 2005.

[26] G. W. Flake and S. Lawrence, "Efficient SVM regression training with SMO," *Machine Learning*, vol. 46, pp. 271–290, 2002.

Appendix B

Development of the 2^{nd} order non-ideal SDM model

In this appendix, the development of the behavioural-level mixed-signal SDM design, which has been used as the case study in chapter 7, has been explained in detail. Most significant imperfections have been integrated in the design. The ideal (section B.1 and B.2) and non-ideal SDM models (section B.3 and B.4) have been implemented in both MATLAB Simulink and SystemC. Comparisons between these models are presented to validate the accuracy of the models.

B.1 Theoretical analysis

SDM represents complex AMS designs that advanced performance modelling and optimisation techniques are highly required. As a long-existing and important technique in ADC design, theoretical analysis of the system provides some design guidelines especially when ideal SDMs are analysed. For ideal SDMs, different parameters such as the number of loops L (order of an SDM), OSR etc determine the general design parameters at the system level. A set of design parameters are summarised in table B.1.

TABLE B.1: Summary of system level design parameters of SDMs.

Parameter	Description
M	sine wave peak amplitude
n	SDM order
L	number of loops
B_q	successive quantisation level difference
f_s	sampling frequency
f_b	signal frequency
OSR	oversampling ration $OSR = f_s/f_b$

For ideal SDMs, noise uniquely comes from the quantiser. The SNR of the system then can be expressed as:

$$SNR = \frac{\sigma_i^2}{q_{rms}^2} \tag{B.1}$$

where σ_i^2 is the power of the input sine wave and q_{rms} is the RMS value of the quantisation noise. Considering equation 2.11 and 2.12 and approximating $\sin(\pi f/f_s)$ to $\pi f/f_s$ when the input signal frequency is low, the following equation can be derived to estimate the RMS noise in the signal band:

$$\frac{n_0^2}{e_{rms}^2} = \frac{\pi^{2L}}{2L+1} \cdot \frac{1}{OSR^{2L+1}} \tag{B.2}$$

where n_0^2/e_{rms}^2 is the ratio of the in-band noise over the whole RMS power of the quantisation noise and thus indicates the noise level in the signal band. This figure is influenced by the number of loops L and the OSR. As shown in figure B.1, with the increment of the OSR and L, this figure decreases which means the in-band noise is shaped and pushed more to the high frequency range so the in-band noise level becomes low.

Quantitatively, the peak SNR, which is assumed to be directly related or even equal to DR, is related with n, B_q and OSR by the following equation [123]:

$$SNR_{peak}(z) = \frac{3\pi}{2}(2^{B_q} - 1)^2(2n+1)(\frac{OSR}{\pi})^{2n+1} \tag{B.3}$$

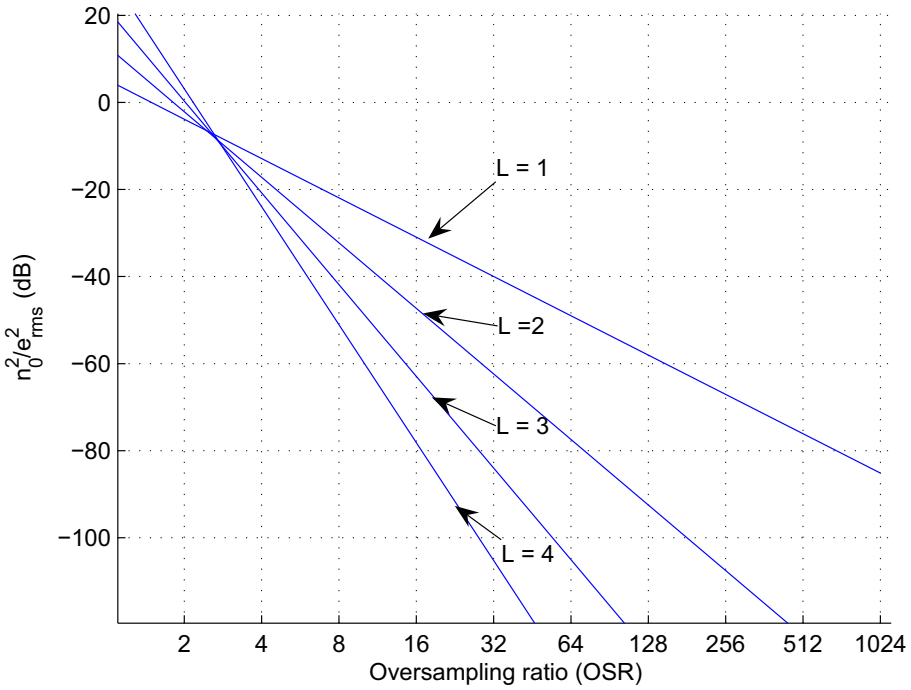


FIGURE B.1: Relationship of n_0^2/e_{rms}^2 and OSR with number of loops L.

Usually, B_q is fixed in many designs, so the relationship between the SNR_{peak}, n and OSR can be graphically illustrated in figure B.2.

These equations provide basic guidelines to decide important performance parameters in SDM designs. For instance, if a design needs the SNR to be above 80dB, according to figure B.2, the 1^{st} order designs cannot be used theoretically unless very high OSR is used. Designs that can fulfil the specification can be with an order equal or more than two if the OSR is less than 64. It has to be noted that considering the degradations in practical designs because of the imperfections of the fabrication process, 2^{nd} order structures, which just hit the point of 80dB, should not be used because they are not possible to reserve their perfect performance. So solutions should come from structures with more than two feedback loops or with higher OSR. The theoretical analysis is helpful as a starting point for further analysis. However, degradations always happen in practical designs on certain degrees and they are arduous to be explicitly expressed.

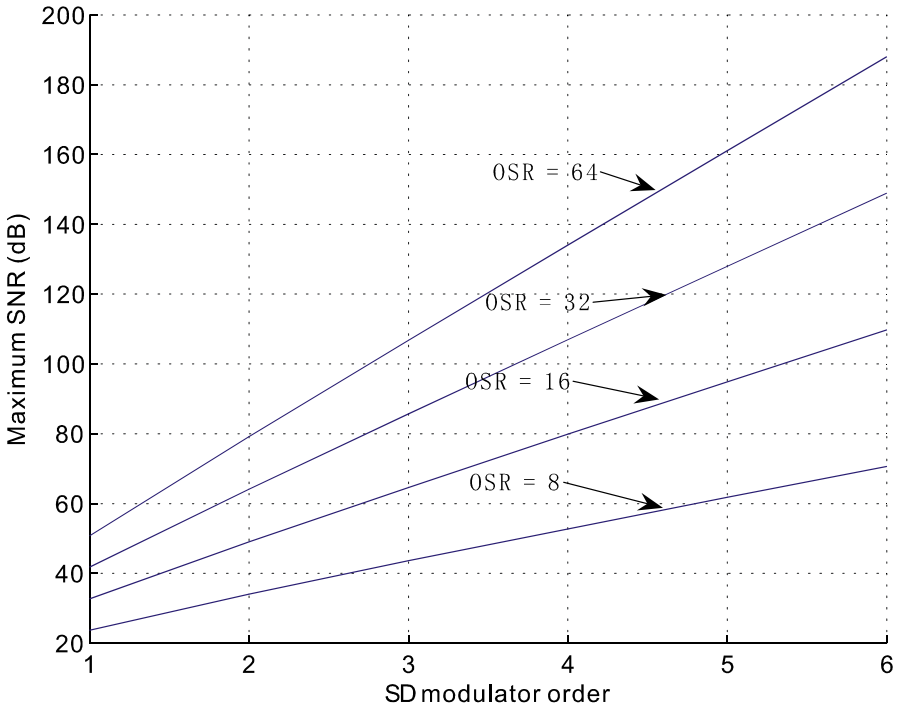


FIGURE B.2: Theoretical SNR as a function of SDM order n and OSR shown in equation B.3.

B.2 Standard manual design process for SDMs

The standard manual design process for SDMs is based on ideal transfer function analysis. Results of the design process are considered as the standard manual analysis output and used to compare with the optimised non-ideal designs so that the degradation of the imperfections can be clearly seen. Assuming the design is based on a 2^{nd} order low-pass SDM with $OSR = 64$. By cascading two 1^{st} order stages, the SDM can be built using the components in the signal flow graph shown in figure B.3. Every integrator (as highlighted in the dashed box) plus the beforehand subtractor form one stage. In z-domain, an integrator is expressed as $1/(z-1)$, which represents the summation of the input and the accumulation of its previous values. The quantiser can be linearised as an adder that adds the the quantiser input and quantisation noise.

The performances of SDMs are determined by their transfer functions, which are further controlled by the amplifier coefficients $(a_1, b_1, \cdots)$ on the feed forward and feedback paths. A straight forward derivation of the relationships on the labelled nodes, M, N, P, Y, V in figure B.3, can link the distributed coefficients to the

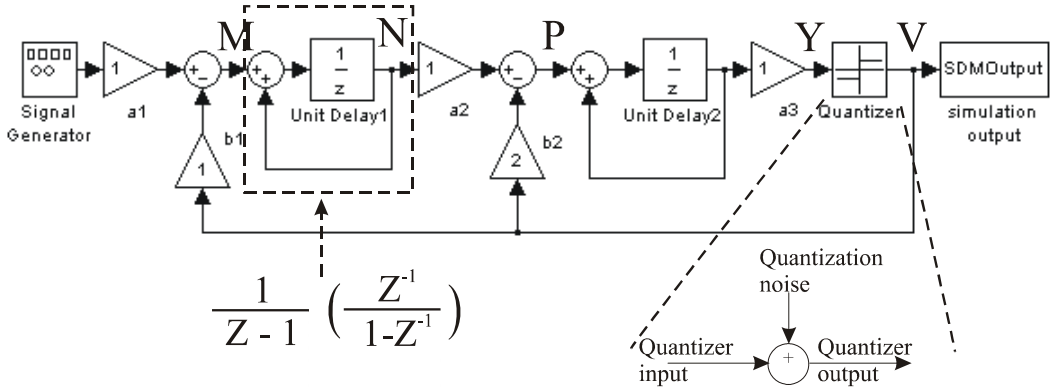


FIGURE B.3: Signal flow graph of a 2^{nd} order ideal SDM with the integrator boxed and a linearised quantiser.

transfer functions as the following:

$$\begin{cases} M(n) = a_1 \cdot U(n) - b_1 \cdot V(n) \\ N(n) = M(n) + N(n-1) \\ P(n) = a_2 \cdot N(n) - b_2 \cdot V(n) \\ Y(n) = a_3 \cdot P(n) + Y(n-1) \\ V(n) = Y(n) + E(n) \end{cases} \tag{B.4}$$

With the substitution of integrators by its z-domain transform "$1/(z-1)$" and iterative substituting of the signals from the left side to the right side in the signal flow graph, the equation that describes the transfer feature of the system is expressed as:

$$V = \frac{a_1 a_2 a_3}{z^2 + (a_3 b_2 - 2)z + 1 - a_3 b_2 + a_2 a_3 b_1} U + \frac{(z-1)^2}{(z-1)^2 + a_3 b_2 (z-1) + a_2 a_3 b_1} E \tag{B.5}$$

As an ideal SDM, the poles and zeros of the STF and NTF are the same as in the following standard equation:

$$V = \frac{1}{z^2} U + \frac{(z-1)^2}{z^2} E = z^{-2} U + (1 - z^{-2}) E \tag{B.6}$$

where the output V of the SDM is a linear superimposition of the delayed input U and a shaped quantisation noise E. By equalising the coefficients of U and E in equation B.5 and B.6, a set of constraints, which the coefficients of the amplifiers

need to satisfy simultaneously, can be generated:

$$\begin{cases} a_1 a_2 a_3 = 1 \\ a_3 b_2 - 2 = 0 \\ 1 - a_3 b_2 + a_2 a_3 b_1 = 0 \end{cases} \Rightarrow \begin{cases} a_1 a_2 a_3 = 1 \\ a_2 a_3 b_1 = 1 \\ a_1 = b_1 \\ a_3 b_2 = 2 \end{cases} \tag{B.7}$$

With these four equations, it is not possible to solve the five variables unless one of the variables is pre-assigned. For instance, let $a_1 = 1$, then one set of solution can be derived: $a_2 = 1, a_3 = 1, b_1 = 1, b_2 = 2$. It can be seen that the key step to get the constraints for the variable solving is to map the derived transfer function to the ideal ones. So if the poles and zeros of the STF and NTF need to be designed specifically for specified transfer feature of the modulator, equation B.5 needs to be mapped to the required transfer function so that a set of constraints can be generated and used to calculate the values of the coefficients.

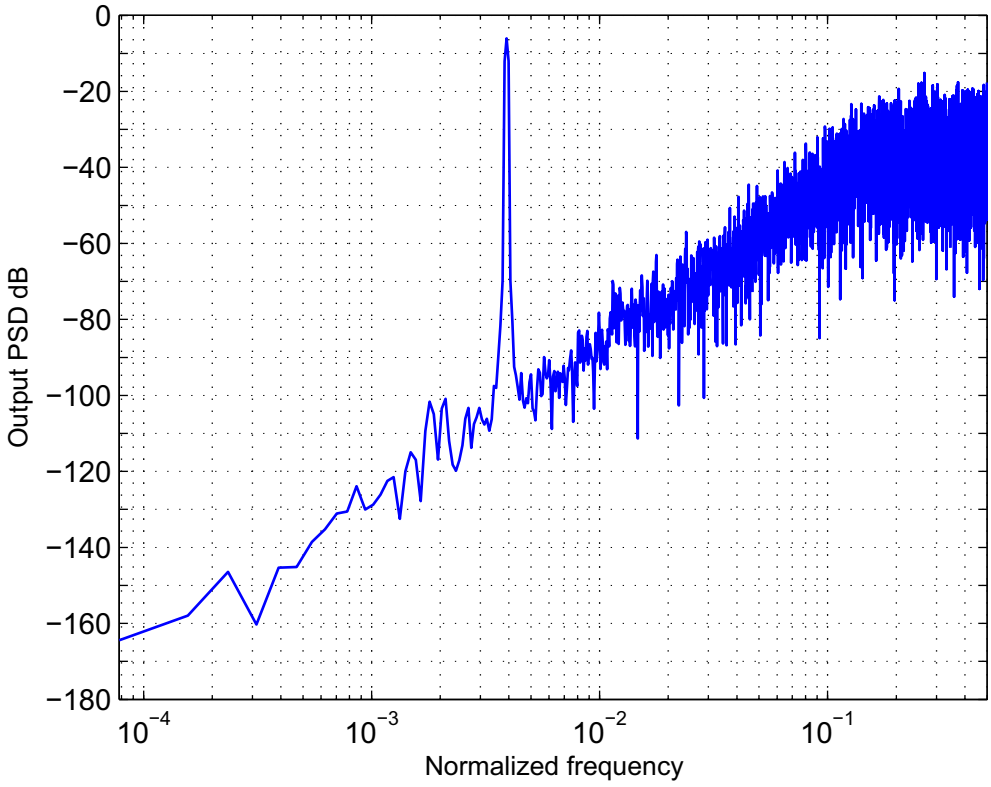


FIGURE B.4: FFT analysis of the output of the ideal 2^{nd} order SDM shown in figure B.3.

Assigning the derived coefficients to the amplifiers, the system is then ready for simulations. It is more meaningful to see the performance of the system in the frequency domain rather than the time domain. The well-known FFT analysis is employed to see the frequency response of the system with the derived coefficients. The results are shown in figure B.4. The spike shows the presence of the signal clearly. The slope of the FFT result shows that noise is pushed to high frequency band as illustrated in section 2.4.1.

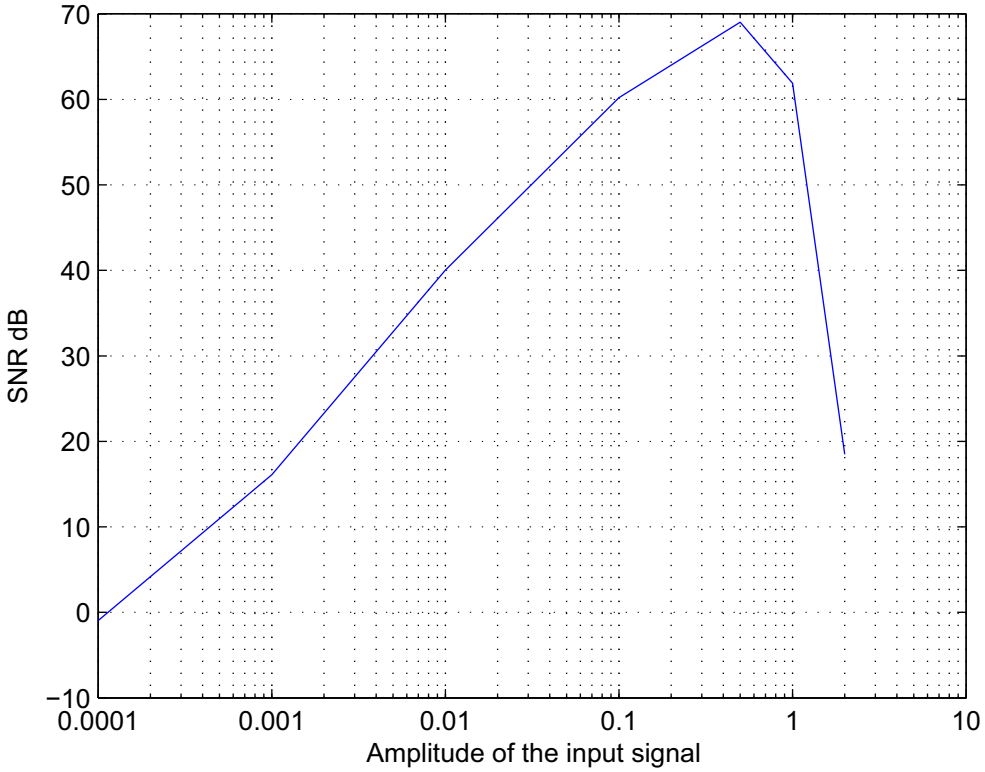
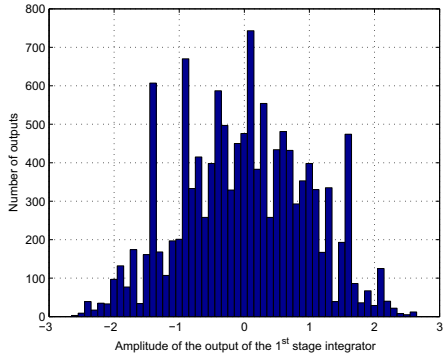


FIGURE B.5: SNR analysis of the ideal 2^{nd} order SDM shown in figure B.3.

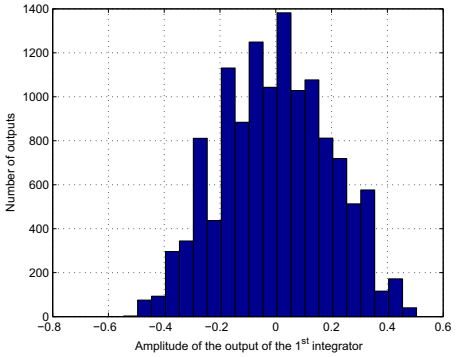
In ideal SDMs, the presentation of the signal dependents on how strong the signal is. When it becomes too small, the signal may not be possibly distinguished from the quantisation noise. Parameter SNR is used to measure the strength of the signal compared to the noise. In this design, figure B.5 shows the relationship between the SNR and the input signal's magnitude. It can be seen that when input signal's amplitude is approximately 0.0001, the $SNR \simeq 0dB$. That is the smallest amplitude for a meaningful signal.

Although, one of the coefficients should be pre-assigned to solve the design variables, the assignment of this coefficient is usually not arbitrary but dependent

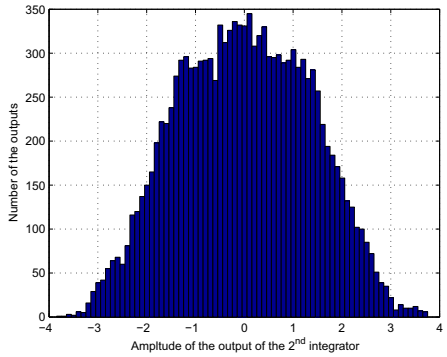
mainly on two considerations [123]. For ease of implementation and minimum hardware usage, it is desirable to choose the digital coefficients equal to 0, ± 1, or multiples of 2, which corresponds to one bit shift in the digital domain. Another critical consideration in choosing the coefficients is the achievable output swing for the integrators in the circuit implementation. The goal of the scaling is to increase the signal dynamic range performance. As shown in figure B.6, if the technology's dynamic range is limited, for instance, a CMOS technology with dynamic range of 0.5V, the assignment of the design parameters needs to reduce the signals' amplitude within the limit so that saturation can be mostly eliminated.



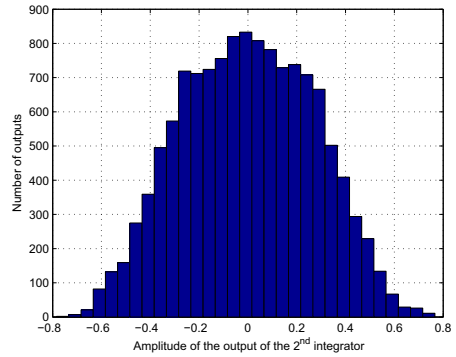
(a) unscaled signal statistics of the 1^{st} integrator



(b) scaled signal statistics of the 1^{st} integrator



(c) unscaled signal statistics of the 2^{nd} integrator



(d) scaled signal statistics of the 2^{nd} integrator

FIGURE B.6: Histogram of the outputs of the 1^{st} and 2^{nd} integrator before and after signal scaling.

B.3 Modelling imperfections in non-ideal SDM designs with MATLAB Simulink

Non-ideal effects are not avoidable in real designs and they degrade the performance of the system. It is beneficial if the influence of the non-ideal effects is modelled and the performance of the SDM with non-ideal effects is effectively estimated at system-level. This is because accurately estimated system level designs can avoid some redesign loop back when they could not pass low level specifications. The lower the level that the design needs to be redesigned is, the more costly the refinement process will be. The modelling of the non-ideal effects dependents on the implementation techniques. In continuous time SDMs, the non-ideal effects include clock jitter, OpAmp non-linear effects and excess loop delay [125]. If sampled data techniques such as switched capacitor (SC) or switched current are used, the advantages initially come from the fabrication technology, i.e. standard digital fabrication technologies can be used for mixed-signal implementation. SC designs are more preferable than continuous time SDM because they can be more efficiently realised in standard CMOS technology [122]. The design parameters are highly controllable thus the influence of the non-ideal effects can be rejected in some degree. The analyses outlined below are based on the SC implementation.

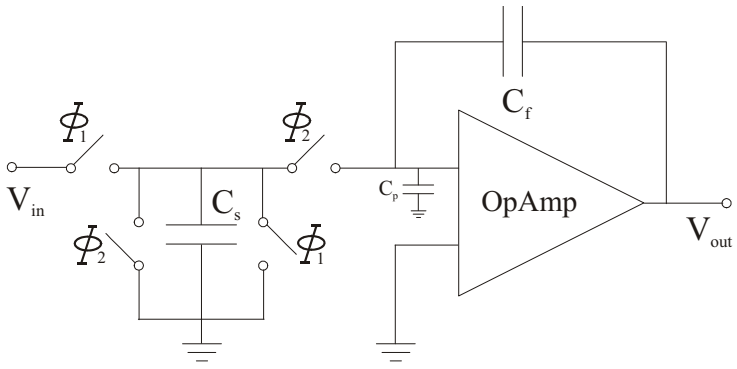


FIGURE B.7: 1^{st} order classical SC integrator module with the OpAmp parasitic capacitor C_p.

The non-ideal effects considered here include clock jitter, switch thermal noise, OpAmp and quantiser non-ideal effects. Detailed model of each effect has been developed and compared in both Matlab Simulink and SystemC. System-level modelling of these non-ideal effects can save computational cost and achieve circuit level accuracy as circuit level SDM simulation is very computationally costly [111].

Detailed behavioural modelling and simulation can make the examples ready to be used in the performance modelling and optimisation.

An SC integrator is shown in figure B.7. It is a basic component to build the SDM system. The analysis of the non-ideal effects is based on this unit. The module contains an SC resistor (C_s), a feedback capacitor C_f, an OpAmp and a parasitic capacitor C_p. The non-overlapping two working phases Φ_1 and Φ_2 form one cycle that is equal to one clock cycle of the system's sampling clock.

B.3.1 Slew rate and OpAmp unity-gain bandwidth

Slew rate (SR) and OpAmp unity-gain bandwidth (UBW) are the two distinct parameters about the settling behaviour of the OpAmps [126]. The effects of these two parameters are related to each other. When the OpAmp operates within the SR-limited region, it functions as a non-linear amplifier; while the OpAmp operates within the UBW-limited region, it works linearly.

To model the SR, the approach used here is to interpret the effect as a non-linear gain which is influenced by the output of the integrator by feedback [127]:

$$V_o(nT_s) = V_o(nT_s - T_s) + \frac{C_s}{C_f} V_s(1 - e^{-t/\tau}); \quad nT_s - \frac{T_s}{2} < t < nT_s \tag{B.8}$$

where V_s is the voltage on the sampling capacitor C_s, τ is the time constant of the integrator and n is the clock index and T_s is the sampling clock period. For any integrating phase (Φ_2 in figure B.7 is on), the starting value of V_s is the voltage on the capacitor when the charging phase (Φ_1 in figure B.7 is on) is over.

$$V_s(nT_s - \frac{T_s}{2}) = V_{in}(nT_s - \frac{T_s}{2}) \tag{B.9}$$

Then the voltage on the capacitor V_s decreases exponentially from the starting voltage. The slope of equation B.8 reaches its maximum when the differentiation over time is equal to 0:

$$\frac{dV_o(t)}{dt}|_{max} = \frac{C_s}{C_f}\frac{V_s}{\tau} \tag{B.10}$$

When this maximum slope is compared to the SR of the SC integrator, two situations can happen: either SR is larger than the maximum slope, the integrator works as indicated by equation B.8; or SR is smaller than the maximum, the integrator works under SR limitation for t_0 time, the output voltage would be:

$$V_o(nT_s) = V_o(nT_s - T_s) + \frac{C_s}{C_f}V_s + (SRt_0 - \frac{C_s}{C_f}V_s)e^{-(t-t_0)/\tau} \tag{B.11}$$

where SRt_0 is the value reached on node V_o at the time point. It has a valid value until SR limitation ends. The value of t_0 can be calculated by inspecting that the slope of equation B.11 should be equal to SR. Then t_0 can be substituted and equation B.11 becomes:

$$V_o(nT_s) = V_o(nT_s - T_s) + \frac{C_s}{C_f}V_s - sgn(V_s)SR\tau e^{-(\frac{T_s}{2\tau} - \frac{C_s|V_s|}{C_f SR\tau} + 1)} \tag{B.12}$$

where *sgn* represents the signum function that takes ± 1 only. This function together with the absolute operation on V_s can cover both the positive and negative of the output.

The implementation of the SR effect is shown in figure B.8. Calculation of the new output needs both the previous output and the current sampled input, thus a delay unit is placed after the zero-order hold component. The input value is compared with the slope to check whether the SR limitation is triggered or not. The last switch component is used to select which source the output should take.

Figure B.9 shows the overall output of the SR module with a ramp excitation. Within the non-SR-limitation region, output approximates input well; when SR limitation happens, the non-linear gain drives the output voltage away from the input as indicated in the figure.

B.3.2 Clock jitter

The working mechanism of SC circuits depends on the transfer of the charge during each clock phase. Clock jitter disturbs the charge transmissions and makes them to be non-equal in each clock phase, thus noise is added to the sampling process. However, as the sampling process is independent to the structure of the SDM, the

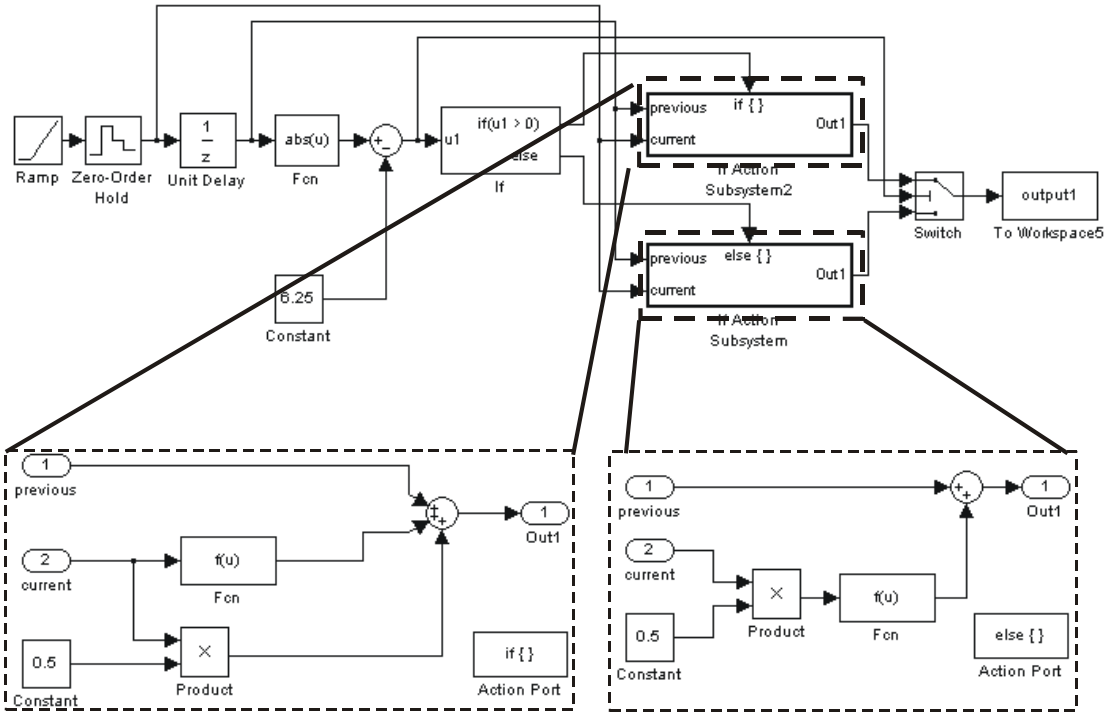


FIGURE B.8: MATLAB Simulink models for the SR modelling. The bottom two sub-systems show the detail implementations when the SR limitation is applied and not.

effect of clock jitter is considered as a superimposed noise to the input signal only [128]. The magnitude of the function is related to both the statistical property of the jitter and the input signal of the modulator. With an analogue input of V_{in} and an error of the time instant δ, the error sampled input resulted from the clock jitter is expressed as:

$$V_{in}(t+\delta) - V_{in}(t) \simeq \delta \frac{dV_{in}(t)}{dt} \quad subject\ to \quad \delta << T_s \tag{B.13}$$

where T_s is the sampling clock period, δ is a Gaussian random parameter with a standard derivation $\Delta\tau = \delta$ and is used to model the statistical property of the jitter, the differentiation of the input signal is the input dependent part of the jitter's model. The signal flow graph shown in figure B.10 is the MATLAB Simulink model used to simulate the superimposed clock jitter.

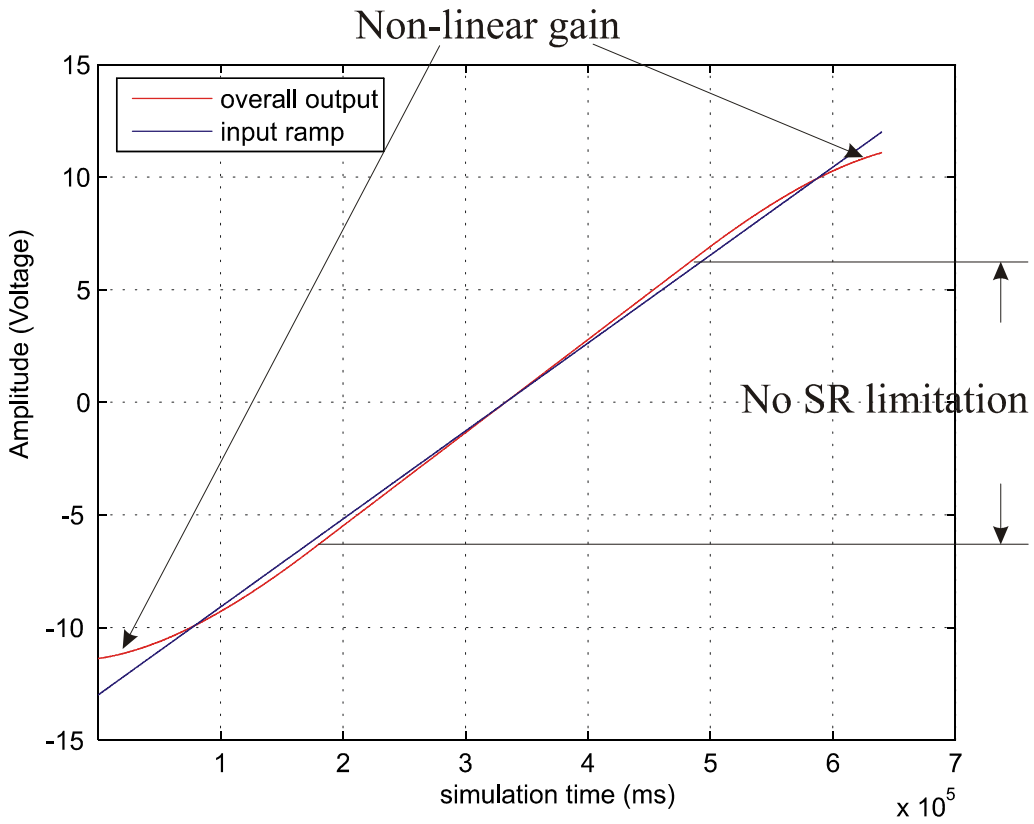


FIGURE B.9: Output of the SR module with a ramp input.

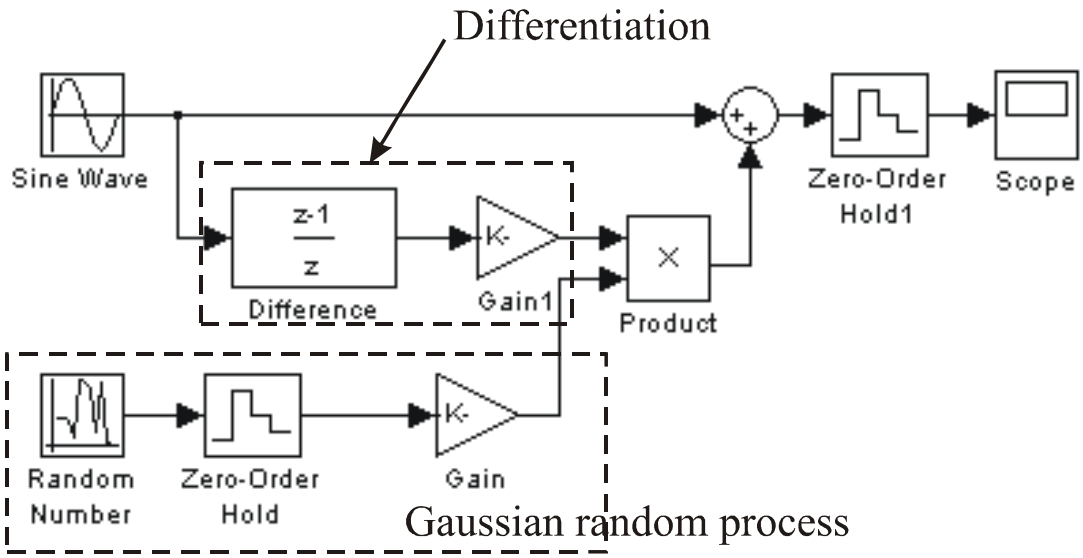


FIGURE B.10: MATLAB Simulink model for the clock jitter modelling.

B.3.3 Switch thermal noise

Switch thermal noise is defined as the thermal noise of the switch resistance sampled by capacitor C_s, which is in series with switch Φ. It has a white spectrum and wide band, limited only by the time constant of the SC integrator or the bandwidth of the OpAmp [128]. The sampling capacitor C_s is equal to finite resistance R_{on} and the total noise power can be found by evaluating the integral:

$$e_T^2 = \int_o^\infty \frac{4kTR_{on}}{1+(2\pi f R_{on}C_s)^2} df = \frac{kT}{C_s} \tag{B.14}$$

where k is the Blotzmann's constant and T is the absolute temperature at working condition. The above thermal nose voltage is superimposed onto the input signal:

$$y(t) = [x(t) + \sqrt{\frac{kT}{C_s}} n(t)]b \tag{B.15}$$

where $n(t)$ is a Gaussian random function, $b = C_s/C_f$ is the ratio that associated with the gain g of the SC integrator, whose transfer function can be expressed as the following:

$$H(z) = g\frac{z^{-1}}{1-\alpha z^{-1}} \tag{B.16}$$

Equation B.15 can be modelled by the signal flow graph shown in figure B.11.

B.3.4 OpAmp thermal imperfections

OpAmp imperfections contain thermal noise, flicker $(1/f)$ noise and DC offset etc. The reason that flicker and DC offset noise are neglected is because these two are cancelled by design techniques in low-pass SDM and they are not important in bandpass SDM designs [128]. The OpAmp thermal noise is modelled by the signal flow graph shown in figure B.12. The constant in the figure is the RMS value of the noise in voltage, which should come from transistor level simulation.

Figure B.13 shows the input signal with superimposition of the clock jitter, switch thermal noise and OpAmp thermal noise. As shown in the enlarged window, the input has fluctuation on the amplitude indicating the presence of the noise sources.

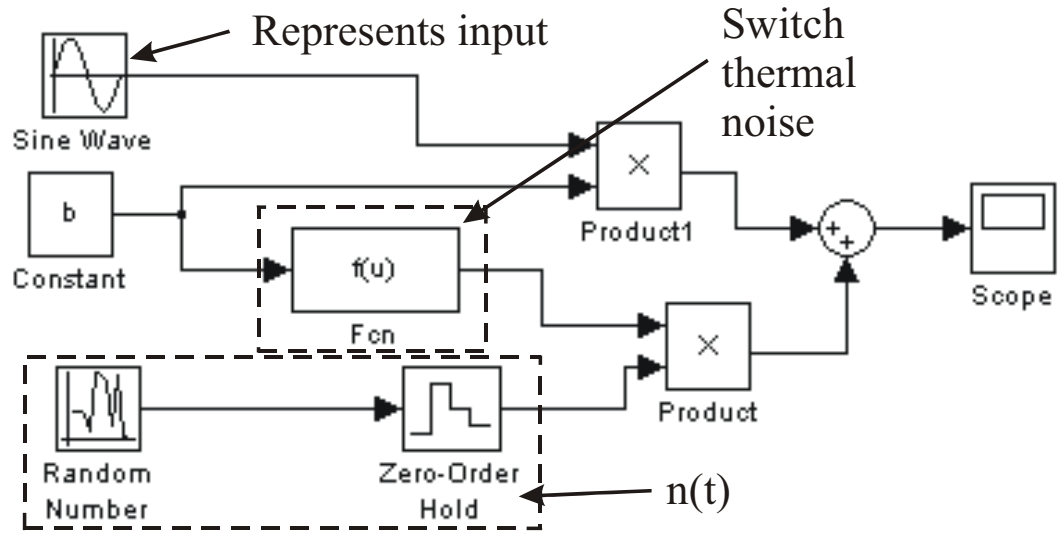


FIGURE B.11: MATLAB Simulink model for the switch thermal noise modelling.

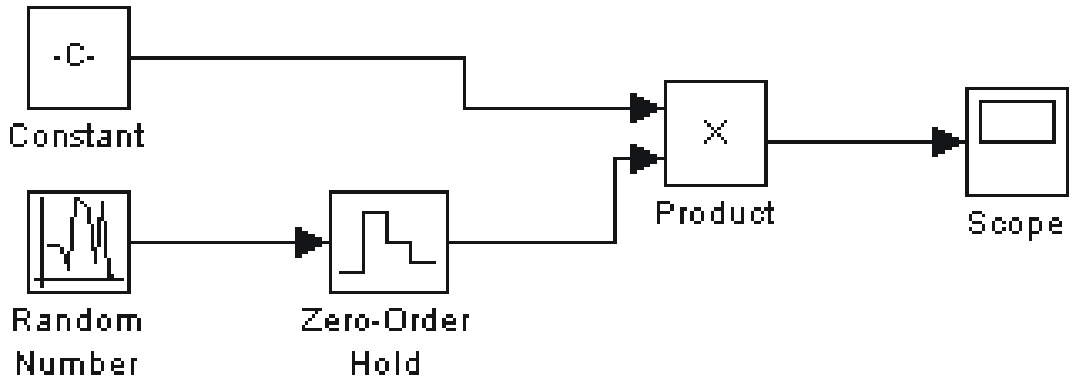


FIGURE B.12: MATLAB Simulink model for the OpAmp thermal noise modelling.

B.3.5 OpAmp finite DC gain

The transfer function of an ideal integrator is:

$$H(z) = \frac{z^{-1}}{1 - z^{-1}} \tag{B.17}$$

Compared to equation B.16, it is a special case that the gain is unity and the feedback factor is 1. The presence of the differences is because of the finite OpAmp DC gain. The consequent effect is known as a "leakage" in the integrator [126, 128]. The precise transfer function can be rewritten with the considerations of the SC

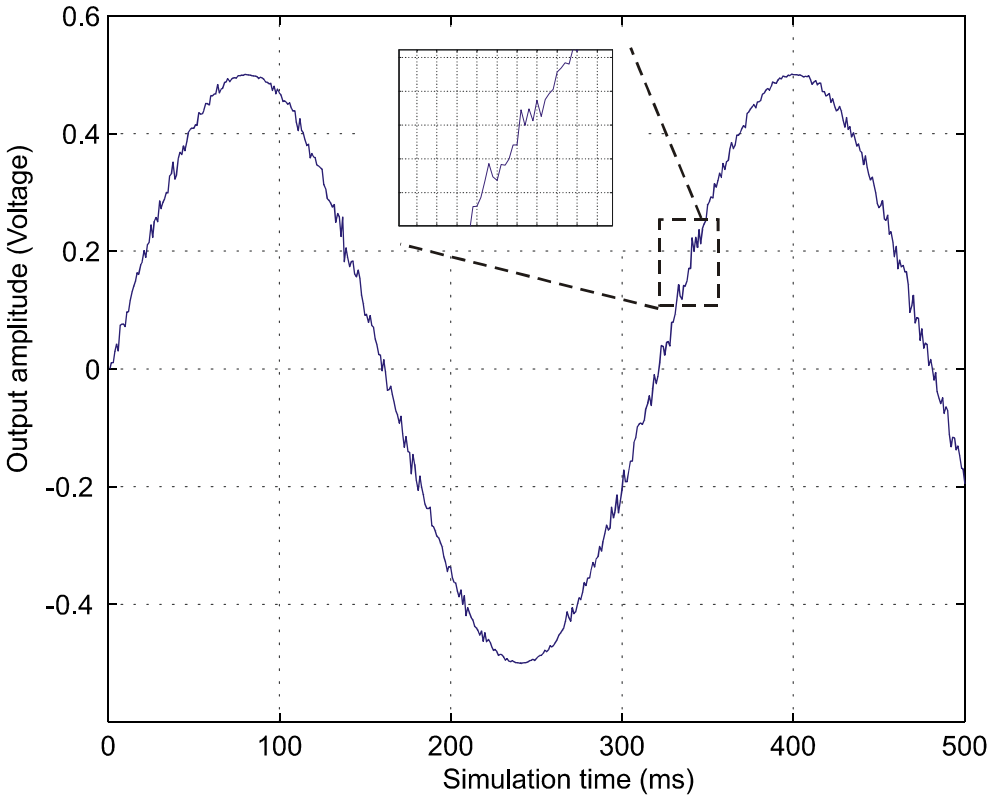


FIGURE B.13: Input sine wave with the superimposition of clock jitter, switch thermal noise and OpAmp thermal noise disturbances.

integrator implementation shown in figure B.7:

$$\begin{aligned} H(z) &= g_{real}\frac{z^{-1}}{1-\alpha z^{-1}} \\ &= r_s(1-\frac{1+r_s+r_p}{A_0})\frac{z^{-1}}{1-(1-\frac{r_s}{A_0})z^{-1}} \end{aligned} \tag{B.18}$$

where $r_s = C_s/C_f$, $r_p = C_p/C_f$ and A_0 is the OpAmp's finite DC gain. With the change of the integrator transfer function, the finite DC gain changes the ideal integrator pole from DC to other frequency defined by the circuit parameters.

B.3.6 OpAmp non-linear DC gain

As equation B.18 shows, even with the OpAmp finite DC gain, the relationship between the circuit parameters, the OpAmp DC gain and the integrator leakage is

linear. Finite DC gain of the integrator does not introduce distortion directly to the design [126, 127]. Distortion is introduced because the DC gain is non-linearly dependent on the input signal. Figure B.14 shows that because of the non-linear DC gain of the OpAmp, the output of the OpAmp drifts away from the ideal response.

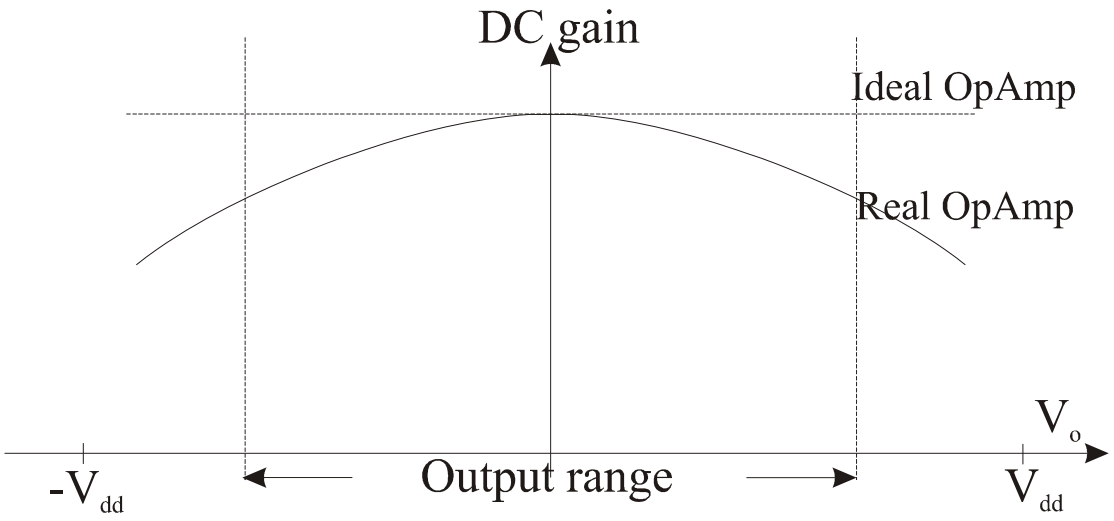


FIGURE B.14: Intuitive illustration of the non-linear DC gain versus output voltage with the rail-to-rail voltage of V_{dd}.

To estimate this effect, a biquadratic equation, like the following, is added to the integrator's gain and leakage:

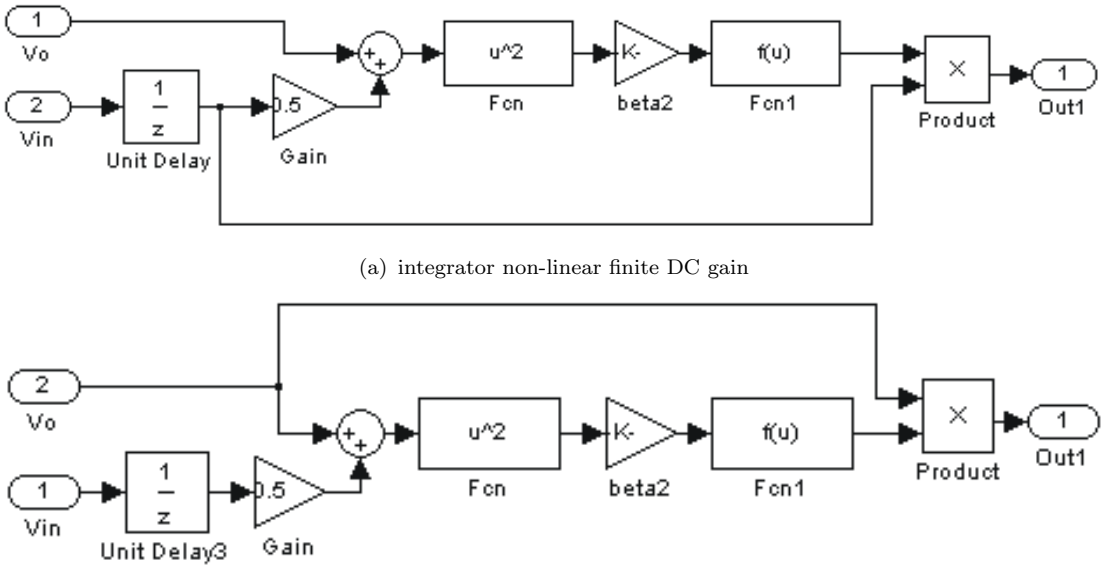
$$f(V_o) = 1 + \beta_1 V_o + \beta_2 V_o^2 \tag{B.19}$$

The coefficients of β_1 and β_2 can make the performance equal to ideal situation if they are set to 0. Adding the non-linear expression onto the integrator gain and leakage provides a new version of the integrator transfer equation:

$$g_{real} = r_s(1 - \frac{1 + r_s + r_p}{A_0} f(V_o)) \tag{B.20}$$

$$\alpha_{real} = 1 - \frac{r_s}{A_0} f_{V_o} \tag{B.21}$$

The imperfect g_{real} and α_{real} are modelled by the signal flow graph in figure B.15. Both of the modules are superimposed to the integrator block.



(a) integrator non-linear finite DC gain

(b) integrator non-linear leakage

FIGURE B.15: MATLAB Simulink modules to model the non-linear integrator DC gain (subplot a) and leakage (subplot b).

B.3.7 Quantiser non-ideal model

The quantiser in SDMs is literally a comparator. The response time of an ideal quantiser is zero for any input. However, a real quantiser needs some setup time to response the change of the inputs and generates correct output. Also, it features some offset voltage, which corresponds to the shift of the threshold of the quantiser. The two main non-ideal parameters used for the construction of a real quantiser model are hysteresis and offset [129], which are shown in figure B.16. This is modelled by a 'quantizer' component in MATLAB Simulink.

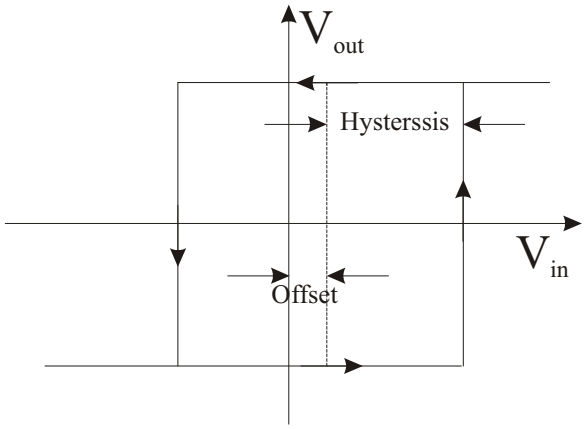
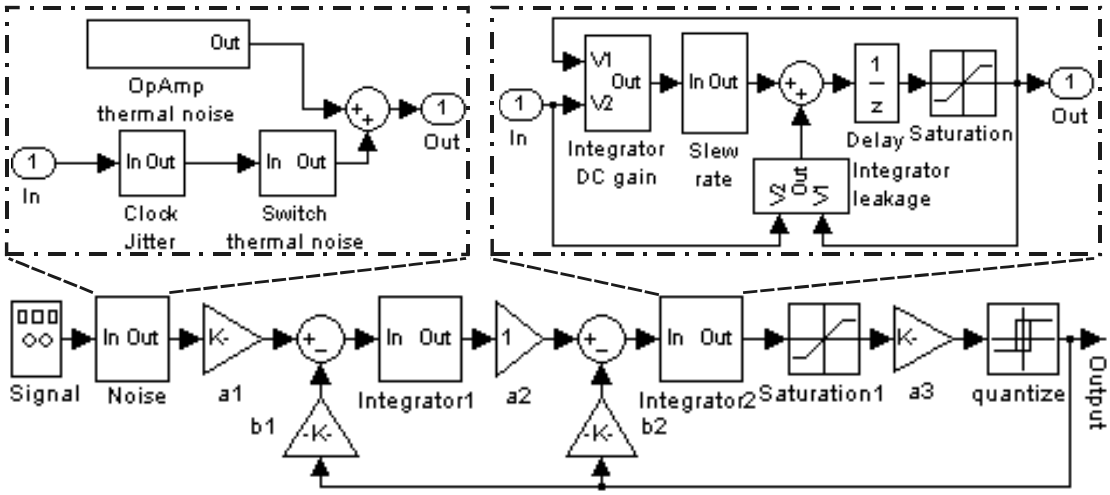
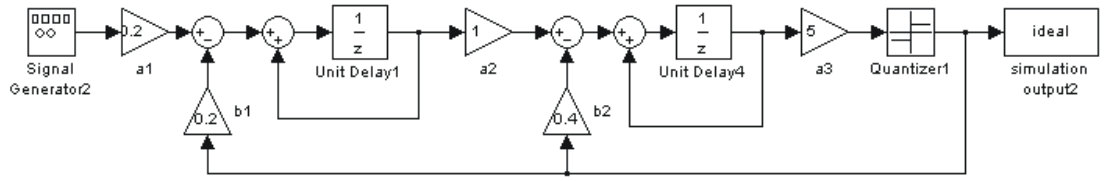


FIGURE B.16: A relay model of the quantiser with offset and hysteresis.

B.3.8 Overview of the non-ideal SDM model



(a) non-ideal 2^{nd} order SDM model



(b) ideal 2^{nd} order SDM model

FIGURE B.17: Comparison of the MATLAB Simulink modules for the non-ideal and ideal 2^{nd} order SDMs.

All the imperfections discussed above are integrated into one non-ideal 2^{nd} order SDM model and is compared with the ideal SDM model in figure B.17. The clock jitter, OpAmp thermal noise and switch thermal noise are superimposed onto the input directly because of their additive feature. For the integrator finite DC gain, leakage and non-linear DC gain, they are combined to model the behaviour of the non-ideal integrator. The saturation component is to model the power supply limitation applied on voltage signals. Finally, the model includes a non-ideal model of the quantiser even though noise from this component is not considered as important and influencing [126].

B.3.9 Simulation and analysis of the non-ideal model

To simulate the non-ideal SDM model, two sets of parameters are needed. The first set is the distributed amplifier coefficients for the transfer function analysis.

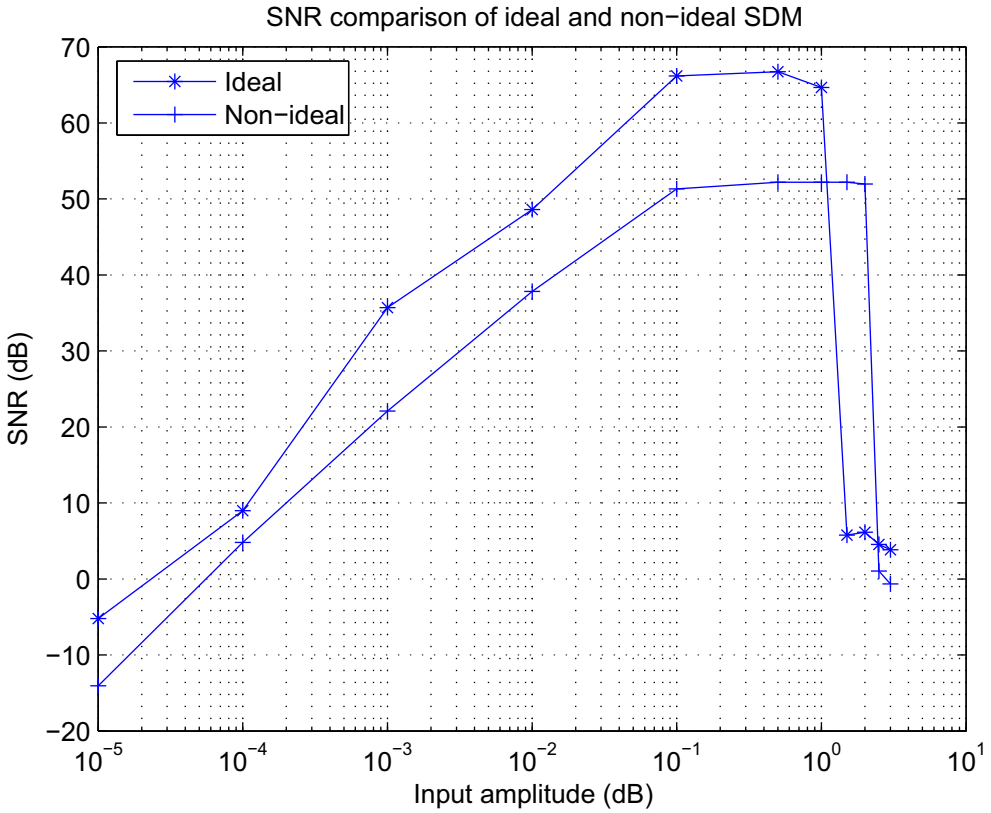


FIGURE B.18: SNR comparison of the ideal and non-ideal 2^{nd} order SDM obtained by FFT analysis on simulation results.

The other set of parameters is the circuit-level parameters to calculate the imperfections. The parameters should come from circuit level simulations [126] or measure of physical designs [128]. The accuracy of the performance modelling is dependent on this parameter set. By taking a study of design parameters measured in some physical designs [123, 130–133], the second set of parameters are assigned and listed in table B.2.

Figure B.18 shows the comparison of the SNR curve of the ideal and non-ideal 2^{nd} order SDM models using the same set of amplifier coefficients. The peak SNR of the non-ideal model is about 55 dB, which is about 15 dB lower than the ideal model result. This degradation on the SNR performance is significant. The models used here have been applied to predict real physical designs and have achieved good approximations to circuit level simulations [126]. It is observed that there is an evident shift of the non-ideal SNR curve in the figure. This is because the capacitor ratio constant b of integrator is 1/2 which modifies the gain of the integrator to half of unity, so the input range is extended.

TABLE B.2: Summary of circuit-level design parameters for non-ideal 2^{nd} order SDM modelling and simulation.

Parameter	Value	Description
M	1.0	sine wave peak amplitude
n	2	SDM order
L	2	number of loops
B_q	1.0v	successive quantisation level difference
f_s	12.8MHz	sampling frequency
f_b	50kHz	signal frequency
OSR	128	oversampling ration $OSR = f_s/f_b$
$\Delta\tau$	1ns	standard derivation of the clock jitter
C_s	2pF	sampling capacitor
T	300K	working temperature
k	1.38×10^{23} J/Kelvin	Boltzmann's constant
V_n	$30\mu V_{rms}$	input-referred OpAmp thermal noise
C_f	4pF	feedback capacitor
SR	$20V/\mu s$	OpAmp slew rate
A_0	75dB	OpAmp DC gain
V_{or}	3V	OpAmp output range
C_p	0.005pF	OpAmp input parasitic capacitance
V_{off}	30mV	quantiser offset voltage
V_{hys}	-50mV	quantiser hysteresis voltage

Figure B.19 shows the comparison of the FFT results of the non-ideal model and ideal model. Clearly, the power spectrum of the non-ideal SDM has larger in-band noise than the ideal one and a noise floor presents. Also, the in-band noise power spectrum is more flat because clock jitter noise, switch thermal noise and OpAmp thermal noise cover the whole frequency range from zero to half of the sampling frequency, when they are added, the entire noise level raises and degrades the effectiveness of the noise-shaping effect.

B.4 SystemC models

A SystemC model of the non-ideal 2^{nd} order SDM has been developed so that an executable parameterised design can be created and iteratively involved by the performance model construction program. A few points in the development of this model have been summarised below.

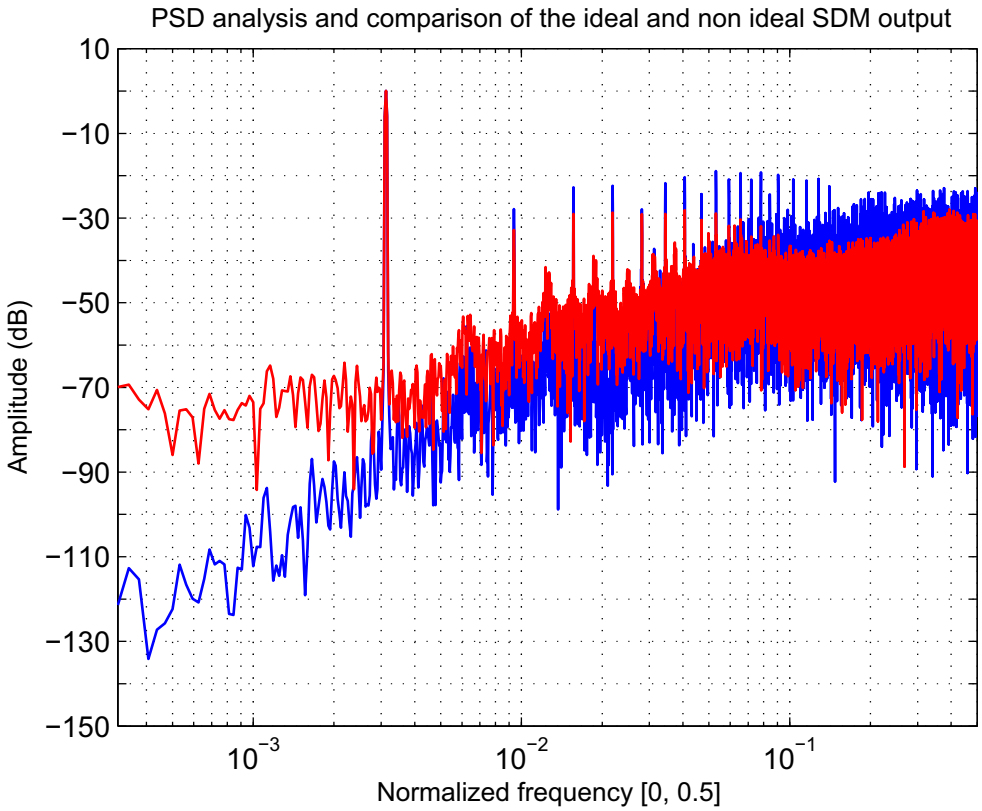


FIGURE B.19: Comparison of the power spectral density analyses of the non-ideal and ideal SDM outputs.

B.4.1 Why SystemC?

As mentioned in section 2.3, there have been a few new HDLs emerging for AMS modelling and simulation. Each of the languages has the capability of modelling the non-ideal 2^{nd} order SDM correctly. Compared to other HDLs, SystemC has its advantages over other languages.

VHDL-AMS is a popular AMS modelling language and has wide applications. As a standard HDL, it has all the language elements needed for the task. The problem encountered in this research is the lack of suitable VHDL-AMS simulators. The simulators that have been tried include SAMSA [117] and SystemVision from MentorGraphic. SAMSA is a MATLAB toolbox that has limited support of the language. It can not do simulation with VHDL-AMS alone but has to be assisted by separate simulation configuration files. More sadly, it does not support mixed model simulation, so it is for either pure digital or analogue designs. SystemVision is a commercial simulator with excellent graphic user interface and integrated

design environment. The limitation of the tool is listed in the user guide [134]. The weakest point of SystemVision is that it does not support text I/O operations, which makes the tool difficult to cooperate with other CAD tools and languages. So the post simulation data processing has to be done using VHDL-AMS along. This increases the development difficulty.

SystemC is flexible and powerful for digital system modelling. Without much effort, it can be easily fitted into sampled data system modelling as used for the SDM application. As the purpose of the research is not to develop dedicated simulation environment for SDMs, this general purpose language is selected.

B.4.2 Conversion of the MATLAB Simulink modules to SystemC

SystemC is a hardware-oriented HDL rather than the software-oriented computing language MATLAB.

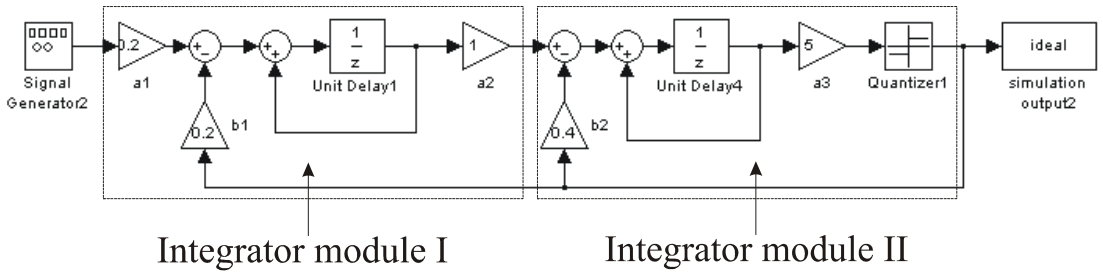


FIGURE B.20: SystemC module converted from MATLAB Simulink module.

Figure B.20 shows the structure of the SystemC module. Only two integrator modules are needed and they are triggered by the system clock. Other operations are combinational and have been integrated into the two modules respectively. The signal generator module can be separated because it is independent to the system and the only function is to provide one input signal.

B.4.3 Simulation results and comparison

Figure B.21 shows the comparison of the four FFT for the ideal and non-ideal SDM models using MATLAB Simulink and SystemC respectively. The results show good consistence of the two simulations. As the MATLAB model has been proved

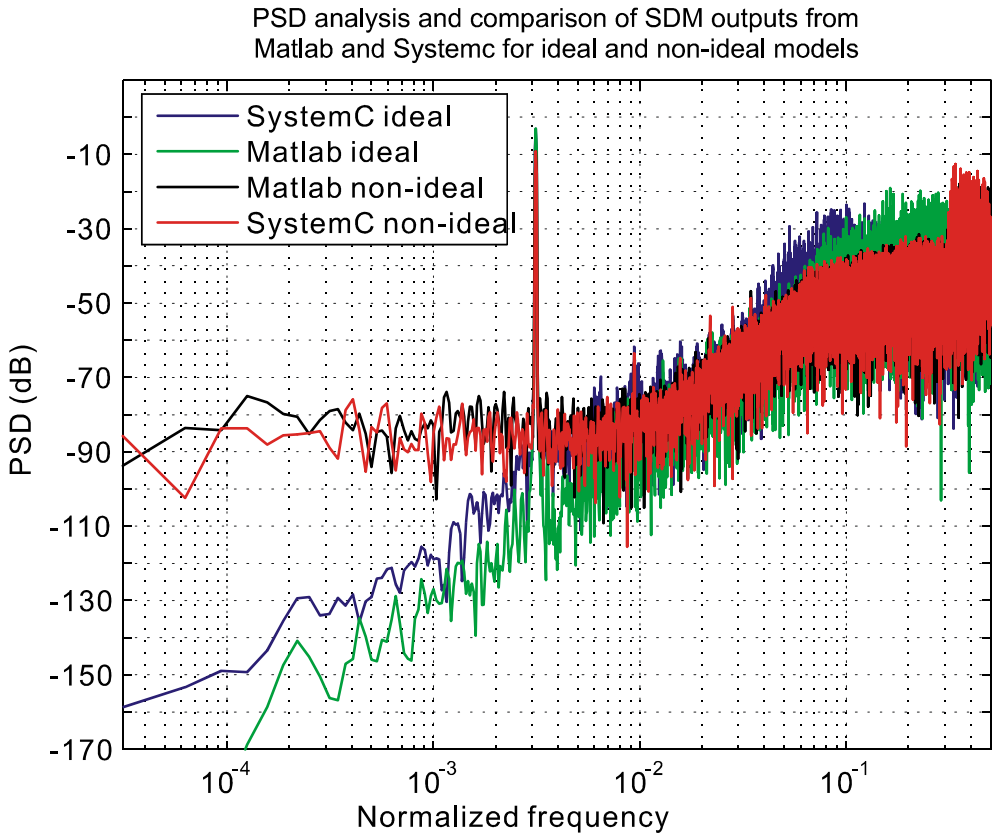


FIGURE B.21: FFT analysis and comparison of SDM outputs for both of the MATLAB Simulink and SystemC ideal and non-ideal models.

by both the circuit level simulation [126] and physical designs [128], SystemC model can be trusted and used to generate the data needed for further performance modelling and AMS optimisations.

Appendix C

Development of the RF Colpitts bandpass filer

This appendix presents the development of the RF Colpitts bandpass filter. Colpitts-type LC oscillators, like the one shown in figure C.1, are used in integrated circuit bandpass filter designs [65, 135]. The MOS FET provides the positive feedback mechanism for Q-enhancement to compensate inductor losses.

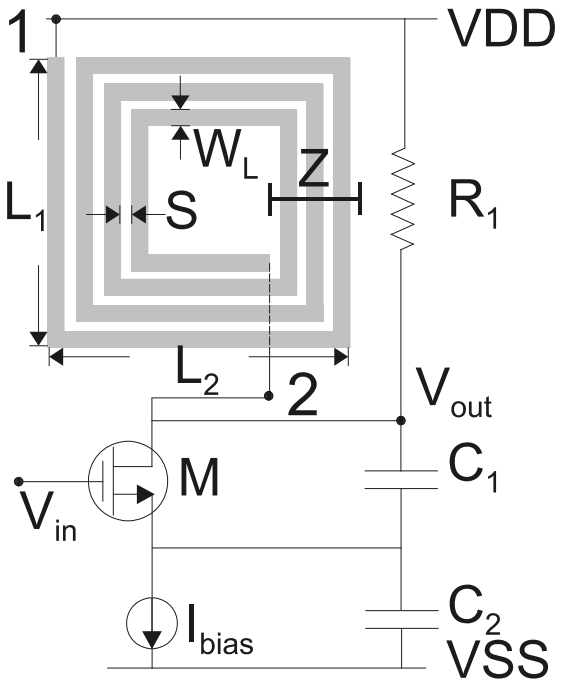


FIGURE C.1: Schematic of the Colpitts RF bandpass filter with a highlighted on-chip planar spiral inductor.

As the inductor in the design is the most challenging component to be implemented, the modelling methodology of the inductor and the calculation of the components' values are introduced in detail in section C.1, C.2 and C.3.

C.1 Inductor SPICE model

On-chip planar rectangular spiral inductors can be modelled by five structural parameters, as illustrated in figure C.1. They include the length of the first and second wire segments (L_1 and L_2 separately), wire width (W_L), space between wires (S) and number of turns (Z). Two metal layers are needed by the planar spiral inductor, where metal 1 is used as the spiral wire and metal 2 as the underpass to connect the inside terminal of the inductor with external connections. The connections of the outside and inside terminals of the inductor have been labelled as '1' and '2' in the figure. The spiral inductor can be accurately modelled by the lumped π-model shown in figure C.2 a) [136]. In the model, L_{total} represents the actual total inductance; R_s is the series resistance associated with the metal wires between the connection point 1 and 2; C_s indicates the capacitive coupling of the overlap between the spiral and the underpass; C_{ox} is the oxide capacitance between the spiral and the silicon substrate; the capacitance and resistance of the substrate are modelled by the network formed by C_{si} and R_{si}. The model provides accurate prediction of the inductor behaviour over a wide range of operating frequencies, layout dimensions and process parameters [136]. A set of scalable

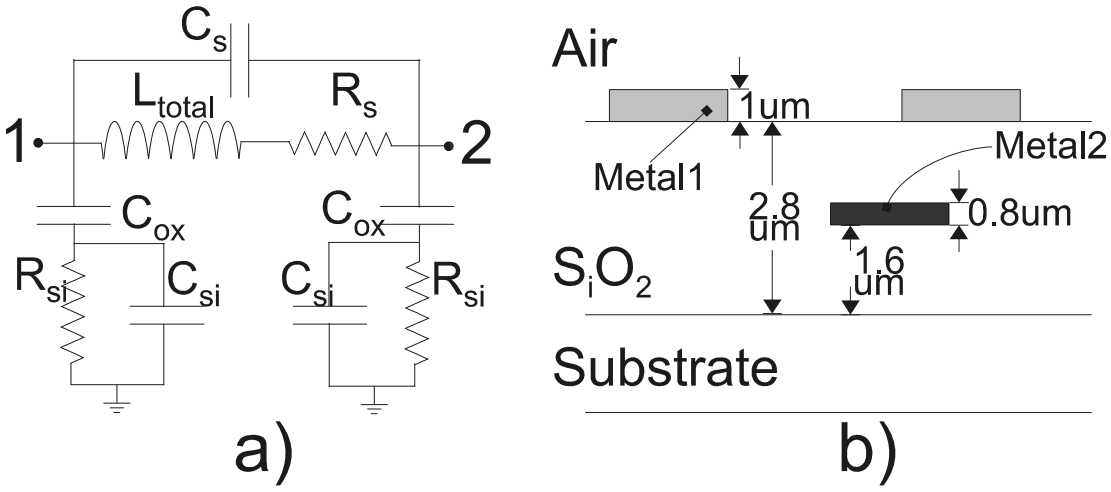


FIGURE C.2: a) the lumped π model of the on-chip spiral inductor; b) cross-section view of the dimension of the spiral inductor.

CMOS design rules based on the TSMC 0.35μm process [137] with two polysilicon and four metal layers has been employed to obtain industrial level accuracy. A cross-section view of the inductor is shown in figure C.2 b). A MATLAB program has been developed based on Greenhouse's method [138] for automatic calculation of the inductance. The total inductance includes the self-inductance and mutual inductance between parallel wire segments. The method offers superior accuracy over many empirical formulas [136].

C.2 Inductance calculation

To calculate the values of the components in the π-model, not only the structural parameters but also the parameters of the fabrication process are necessary. The total inductance includes self-inductance and mutual inductance between parallel wire segments.

$$L_{total} = L_0 + \sum M = L_0 + M_+ - M_- \tag{C.1}$$

where L_0 is the sum of self-inductance, M_+ and M_- are the sum of positive and negative mutual inductance separately. The self-inductance L_0 can be calculated using the following equation:

$$L_0 = 2 \cdot l \cdot (ln\frac{2l}{W_L + t} + 0.5 + \frac{W_L + t}{3l}) \tag{C.2}$$

where l is the wire length, W_L and t are the width and thickness of the wire. For mutual inductances, the signs depend on the current flow directions in two segments that are parallel. The values are functions of the length and separations as the following:

$$M = 2 \cdot l \cdot Q \tag{C.3}$$

where the mutual inductance parameter Q is calculated from equation:

$$Q = ln[\frac{l}{GMD} + \sqrt{1 + (\frac{l}{GMD})^2} - \sqrt{1 + (\frac{GMD}{l})^2} + \frac{GMD}{l}] \tag{C.4}$$

In this equation, l is the length of the wire, GMD is the geometric mean distance between two inductor segments. The exact value can be calculated from the

following equation:

$$ln(GMD) = ln(d) - \frac{W_L^2}{12d^2} - \frac{W_L^4}{60d^4} - \frac{W_L^6}{168d^6} - \frac{W_L^8}{360d^8} - \cdots \tag{C.5}$$

where W_L is the width of the inductor and d is the distance between the tracks' centres.

With a set of specifications on the structural parameters of the inductor, the MATLAB program calculates the lengths of the other segments of the inductor are derived using the following relationship:

$$l_{odd} = l_1 - (y-2) \cdot (W_L + S) \tag{C.6}$$

$$l_{even} = l_2 - (y-1) \cdot (W_L + S) \tag{C.7}$$

where $y \in (2, Z)$ is the loop index. Then, the next step is to calculate the self and mutual inductance using the equations introduced above. In general, for a parallel pair of inductor like shown in figure C.3, the equation is:

$$2M_{1,2} = (M_{2+p} + M_{2+q}) - (M_p + M_q) \tag{C.8}$$

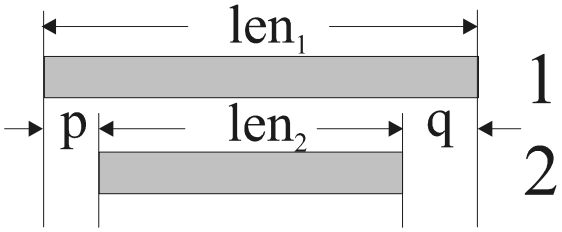


FIGURE C.3: Geometry of two inductors in parallel.

As the inductor segments are symmetric over the central axis, there are two cases for the parallel structure:

$$for \quad p = q, \; M_{1,2} = M_{2+p} - M_p \tag{C.9}$$

$$for \quad p = 0, \; 2M_{1,2} = (M_1 + M_2) - M_q \tag{C.10}$$

Considering the structure of the inductor, it is clear that equation C.9 deals with general situations and equation C.10 is applicable to the first segment only. For a Z-turn spiral inductor, the overall inductance involves $4Z$ self-inductance terms, $2Z(Z-1)$ positive mutual inductance terms and $2Z^2$ negative mutual inductance

terms. The above models for inductance calculations provide superior accuracy over wide range of rectangular planar on silicon than those empirical ones [136].

C.3 Inductor model verification

The fabrication process parameters used for Q factor calculation have been summarised in table C.1.

TABLE C.1: Summary of the process parameters for on-silicon spiral inductor modelling.

Parameter	Value	Unit	Description
ϵ	8.854×10^{-18}	F/μm	permittivity of free space
ρ	2.82×10^{-2}	Ω-μm	resistivity of Aluminium
μ	1.257×10^{-12}	H/μm	permeability of Aluminium
C_{sub}	6×10^{-18}	F/μm^2	capacitance of unit area of substrate
G_{sub}	1.6×10^{-7}	S/μm^2	conductance of unit area of substrate
t	1	μm	thickness of the conductors
$t_{ox_M1_M2}$	0.4	μm	oxide thickness between metal1 and metal2
t_{ox}	2.8	μm	oxide thickness between inductor and substrate

The Q factor is calculated using the following equation:

$$Q = \frac{f_{maxG}}{|f_{-3dB} - f_{+3dB}|} \tag{C.11}$$

where f_{maxG} is the frequency where maximum gain G appears, f_{-3dB} and f_{+3dB} are the frequency points whose gain values are $3dB$ smaller than the maximum gain. Figure C.4 shows the influence of the inductor geometry on the Q factor. Compared to the measured results from fabrications [136, 139], the model can provide accurate modelling of the rectangular spiral on-silicon inductors.

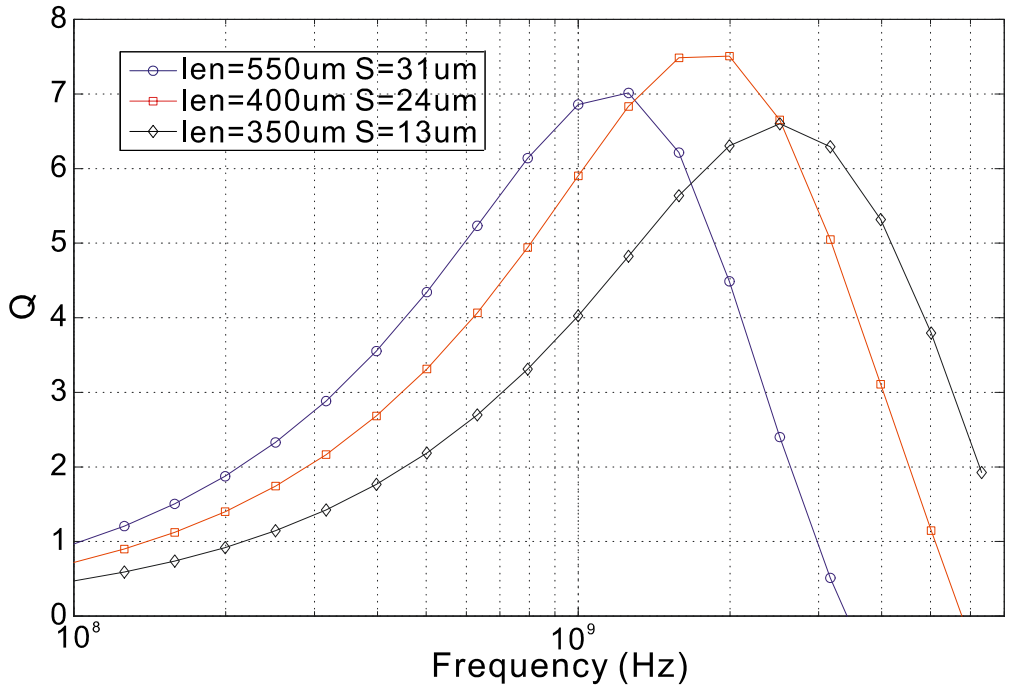


FIGURE C.4: Influence of the inductor geometry *len* and S on the Q factor.

C.4 Calculation of other components in the π model

The equations of other components in the π model reflect the studies of the high-order effects associated with the on-chip inductor. The series resistance is calculated from:

$$R_s = \rho \cdot l/(W_L * t_{eff}) \tag{C.12}$$

$$t_{eff} = \delta \cdot (1 - e^{(}\frac{-t}{\delta})) \tag{C.13}$$

$$\delta = \sqrt{\frac{\rho}{\pi \cdot \mu \cdot f}}; \tag{C.14}$$

where ρ is the resistivity of the metal wires, μ is the permeability, l, t and W_L are the length, thickness and width, δ is the skin depth of the eddy current, t_{eff} is the effective thickness. The series capacitance of the inductor in figure C.2 is:

$$C_s = n \cdot W_L^2 \cdot \frac{\epsilon_{ox}}{t_{ox\_M1\_M2}} \tag{C.15}$$

where n is the number of overlaps of the spiral metal 1 and the underpass metal 2, ϵ_{ox} is the oxide unit area capacitance, $t_{ox\_M1\_M2}$ is the oxide thickness between the spiral metal 1 and the underpass metal 2. The substrate network has three components. They are derived straight from the following three equations:

$$C_{ox} = 0.5 \cdot l \cdot W_L \cdot \frac{\epsilon_{ox}}{t_{ox}} \tag{C.16}$$

$$C_{si} = 0.5 \cdot l \cdot W_l \cdot C_{sub} \tag{C.17}$$

$$R_{si} = \frac{2}{l \cdot W_L \cdot G_{sub}} \tag{C.18}$$

where C_{sub} and G_{sub} are the capacitance and conductance per unit area of the substrate separately. All the calculations of the components have been implemented in MATLAB as functions to be utilised by the SPICE netlist creation scripts.

References

[1] G. Bois, S. Tahar, and M. Zaki, "Formal verification of analog and mixed signal designs: survey and comparison," IEEE North-East Workshop on Circuits and Systems, pp. 281 – 284, June 2006.

[2] G. Choi, J. Kim, H. Park, Y. Ahn, H. Park, J. Bae, I. Park, and D. Shin, "A 0.18-μm CMOS front-end processor for a Blu-Ray Disc recorder with an adaptive PRML," IEEE Journal of Solid-State Circuits, vol. Vol.40(1), pp. 342 – 350, Jan 2005.

[3] P. Kollig and S. Stan, "System-on-chip platform for high-speed DVD+R/RW video and data applications," IEEE International Conference on Consumer Electronics (ICCE), pp. 326 – 327, June 17-19 2003.

[4] S. Lee, J. Park, S. Kim, S. Ko, and S. Kim, "Implementation of H.264/AVC decoder for mobile video applications," Asia and South Pacific Conference on Design Automation, p. 2 pp, Jan 24-27 2006.

[5] D. Buss, B. Evans, J. Bellay, W. Krenik, B. Haroun, D. Leipold, K. Maggio, J. Yang, and T. Moise, "SOC CMOS technology for personal internet products," IEEE Transactions on Electron Devices, vol. Vol.50(3), pp. 546 – 556, March 2003.

[6] D. Buss, "Device issues in the integration of analog/RF functions in deep submicron digital CMOS," International Electron Devices Meeting (IEDM) Technical Digest, pp. 423 – 426, Dec 5-8 1999.

[7] R. A. Rutenbar, G. G. E. Gielen, and J. Roychowdhury, "Hierarchical modeling, optimization, and synthesis for system-level analog and RF designs," Proceedings of the IEEE, vol. Vol.95(3), pp. 640 – 669, March 2007.

[8] IEEE, "IEEE standard VHDL analog and mixed-signal extensions language reference manual," 1076.1 - 1999, Dec 1999.

[9] H. Al-Junaid and T. Kazmierski, "Analogue and mixed-signal extension to SystemC," IEE Proceedings-Circuits, Devices and Systems, pp. 682 – 690, Dec 9 2005.

[10] O. Kaser, "On squashing hierarchical designs [VLSI]," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. Vol.14(11), pp. 1398 – 1402, Nov 1995.

[11] M. Stan, A. Cabe, S. Ghosh, and Z. Qi, "Teaching top-down ASIC/SoC design vs bottom-up custom VLSI," IEEE International Conference on Microelectronic Systems Education (MSE), pp. 89 – 90, June 3-4 2007.

[12] G. Gielen and R. Rutenbar, "Computer-aided design of analog and mixed-signal integrated circuits," Proceedings of the IEEE, vol. Vol.88(12), pp. 1825 – 1854, Dec 2000.

[13] R. Rutenbar, "Design automation for analog: the next generation of tool challenges," IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 458 – 460, Nov 2006.

[14] IEEE, "Verilog-A language reference manual - analog extensions to Verilog HDL version 1.0," 1996.

[15] F. Pecheux, C. Lallement, and A. Vachoux, "VHDL-AMS and Verilog-AMS as alternative hardware description languages for efficient modeling of multidiscipline systems," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. Vol.24(2), pp. 204–225, Feb 2005.

[16] E. Ochotta, R. Rutenbar, and L. Carley, "Synthesis of high-performance analog circuits in ASTRX/OBLX," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. Vol.15(3), pp. 273 – 294, March 1996.

[17] G. Wolfe and R. Vemuri, "Extraction and use of neural network models in automated synthesis of operational amplifiers," Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, vol. Vol.22(2), pp. 198 – 212, Feb 2003.

[18] W. Nye, D. Riley, A. Sangiovanni-Vincentelli, and A. Tits, "DELIGHT.SPICE: an optimization-based system for the design of integrated circuits," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. Vol.7(4), pp. 501 – 519, Apr 1988.

[19] G. V. der Plas, G. Debyser, F. Leyn, K. Lampaert, J. Vandenbussche, G. Gielen, W. Sansen, P. Veselinovic, and D. Leenarts, "AMGIE-A synthesis environment for CMOS analog integrated circuits," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. Vol.20(9), pp. 1037 – 1058, Sep 2001.

[20] K. Francken, P. Vancorenland, and G. Gielen, "DAISY: a simulation-based high-level synthesis tool for $\Sigma\Delta$ modulators," IEEE/ACM International Conference on Computer Aided Design ICCAD, pp. 188 – 192, Nov 5-9 2000.

[21] T. Mukherjee, L. Carley, and R. Rutenbar, "Efficient handling of operating range and manufacturing line variations in analog cell synthesis," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. Vol.19(8), pp. 825 – 839, Aug 2000.

[22] S. Balkir, G. Diindar, and G. Alpaydin, "Evolution based synthesis of analog integrated circuits and systems," NASA/DoD Conference on Evolvable Hardware, pp. 26 – 29, June 24-26 2004.

[23] V. M. zu Bexten, C. Moraga, R. Klinke, W. Brockherde, and K. Hess, "ALSYN: flexible rule-based layout synthesis for analog IC's," IEEE Journal of Solid-State Circuits, vol. Vol.28(3), pp. 261 – 268, March 1993.

[24] T. Li and W. Sui, "Extending spice-like analog simulator with a time-domain full-wave field solver," IEEE MTT-S International Microwave Symposium Digest, vol. Vol.2, pp. 1023–1026, May 20-25 2001.

[25] B. Song, S. Kim, S. Kwack, M. Choi, and K. Kwack, "A simulation efficiency improvement method for simulation-based analog cell synthesis," IEEE Asia Pacific Conference on ASICs (AP-ASIC), pp. 225–228, Aug 23-25 1999.

[26] G. Gielen, H. Walscharts, and W. Sansen, "ISAAC: a symbolic simulator for analog integrated circuits," IEEE Journal of Solid-State Circuits, vol. Vol.24(6), pp. 1587 – 1597, Dec 1989.

[27] M. del Mar Hershenson, S. Boyd, and T. Lee, "GPCAD: a tool for CMOS op-amp synthesis," IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 296 – 303, Nov 8-12 1998.

[28] B. Ray, P. Chaudhuri, and P. Nandi, "Efficient synthesis of OTA network for linear analog functions," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. Vol.21(5), pp. 517 – 533, May 2002.

[29] C. Cuypers, N. Voo, M. Teplechuk, and J. Sewell, "General synthesis of complex analogue filters," Circuits, Devices and Systems, IEE Proceedings-, vol. Vol.152(1), pp. 7 – 15, Feb 4 2005.

[30] H. Aboushady, L. de Lamarre, N. Beilleau, and M. Louerat, "A mixed equation-based and simulation-based design methodology for continuous-time sigma-delta modulators," Midwest Symposium on Circuits and Systems (MWSCAS), vol. Vol.1, pp. oI – 109–12, July 25-28 2004.

[31] M. Vogels and G. Gielen, "Architectural selection of A/D converters," Design Automation Conference (DAC), pp. 974–977, June 2-6 2003.

[32] L. Carley, P. Fung, P. Donehue, and A. Biyabani, "Numerical optimization-based synthesis of pipelined A/D converters," IEEE International Symposium on Circuits and Systems (ISCAS), vol. Vol.5, pp. 2152–2155, May 3-6 1992.

[33] G. Gielen and J. Franca, "CAD tools for data converter design: an overview," IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, vol. Vol.43(2), pp. 77–89, Feb 1996.

[34] A. Odet-Allah and M. Hassoun, "An algorithm for symbolic and numeric architecture determination in a knowledge-based analog-to-digital converter synthesis environment using fuzzy membership functions," IEEE International Symposium on Circuits and Systems (ISCAS), vol. Vol.5, pp. 607–611, May 30 - June 2 1999.

[35] V. N. Vapnik, The Nature of statistical learning theory. Springer Verlag New York Inc, 2001. ISBN: 978-0387987804.

[36] S. Lu, W. Chen, and M. Li, "Fault pattern recognition of rolling bearing based on wavelet packet and support vector machine," World Congress on

Intelligent Control and Automation (WCICA), vol. Vol.2, pp. 5516 – 5520, June 21-23 2006.

[37] Z. Luo and Z. Shi, "On electronic equipment fault diagnosis using least squares wavelet support vector machines," World Congress on Intelligent Control and Automation (WCICA), vol. Vol.2, pp. 6193 – 6197, June 21-23 2006.

[38] H. Li, X. Zhu, and B. Shi, "Nonlinear identification based on least squares support vector machine," Control, Automation, Robotics and Vision Conference (ICARCV), vol. Vol.3, pp. 2331 – 2335, Dec 6-9 2004.

[39] A. Torralba, J. Chavez, and L. Franquelo, "FASY: a fuzzy-logic based tool for analog synthesis," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. Vol.15(7), pp. 705 – 715, July 1996.

[40] A. Sahu, B.; Dutta, "Automatic synthesis of CMOS operational amplifiers: a fuzzy optimization approach," Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 366 – 371, Jan 7-11 2002.

[41] B. Scholkopf, "Statistical learning and kernel methods," tech. rep., Microsoft reasearch limited, Feb 2000. Software available at `http:://research.microsoft.com/~bsc`.

[42] C. Hsu, C. Chang, and C. Lin, "A practical guide to support vector classification," 2001. National Taiwan University, `http://www.csie.ntu.edu.tw/cjlin/libsvm`.

[43] A. Smola and B. Scholkopf, "A tutorial on support vector regression," tech. rep., NeuroCOLT2 technical report series, NC2-TR-1998-30, Oct 1998.

[44] D. Sebald and J. Bucklew, "Support vector machine techniques for nonlinear equalization," Control, Automation, Robotics and Vision Conference (ICARCV), vol. Vol.48(11), pp. 3217 – 3226, Nov 2000.

[45] T. Kiely and G. Gielen, "Performance modeling of analog integrated circuits using least-squares support vector machines," Proceedings of Design, Automation and Test in Europe Conference and Exhibition, vol. Vol.1, pp. 448 – 453, Feb 16-20 2004.

[46] F. D. Bernardinis, M. I. Jordan, and A. sangiovanni Vincentelli, "Support vector machines for analog circuit performance representation," Design Automation Conference (DAC), pp. 964 – 969, Jun 2-7 2003.

[47] M. Ding and R. Vemuri, "A combined feasibility and performance macromodel for analog circuits," Design Automation Conference (DAC), pp. 63 – 68, June 13-17 2005.

[48] X. Ren and T. Kazmierski, "Linearly graded behavioural analogue performance models using support vector machines and VHDL-AMS," Forum on specification and design languages (FDL), ECSI, Sep 27-30 2005.

[49] X. Ren and T. Kazmierski, "Behavioral-level performance modeling of analog and mixed-signal systems using support vector machines," IEEE Proceedings of Behavioral Modeling and Simulation Workshop (BMAS), pp. 28 – 33, Sept 2006.

[50] X. Ren and T. Kazmierski, "Performance modelling and optimisation of RF circuits using support vector machines," IEEE International Mixed design conference (MIXDES), pp. 317 – 321, June 21-24 2007.

[51] W. Wolf, "Synthesis tools help teach systems concepts in VLSI design," IEEE Transactions on Education, vol. Vol.35(1), pp. 11–17, Feb 1992.

[52] J. Allen, "Performance-directed synthesis of VLSI systems," Proceedings of the IEEE, vol. Vol.78(2), pp. 336 – 355, Feb 1990.

[53] K. Kundert, H. Chang, D. Jefferies, G. Lamant, E. Malavasi, and F. Sendig, "Design of mixed-signal systems-on-a-chip," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. Vol.19(12), pp. 1561 – 1571, Dec 2000.

[54] J. Harlow, "Toward design technology in 2020: trends, issues, and challenges [VLSI design]," IEEE Computer Society Annual Symposium on VLSI, vol. Vol.20, pp. 3 – 4, Feb 2003.

[55] V. Grimblatt, "Synthesis - state of art," International Caribbean Conference on Devices, Circuits and Systems, pp. 327 – 332, April 2006.

[56] P. Oehler, C. Grimm, and K. Waldschmidt, "A methodology for system-level synthesis of mixed-signal applications," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. Vol.10(6), pp. 935 – 942, Dec 2002.

[57] R. Harjani and B. Vinnakota, "Analog circuit observer blocks," IEEE Transactions on Circuits and Systems II: Analog and Digital Signal, vol. Vol.44(3), pp. 154 – 163, March 1997.

[58] IEEE, "IEEE standard VHDL language reference manual language reference manual," 1076 - 1987, Feb, 21 1992.

[59] A. Doboli and R. Vemuri, "Behavioral modeling for high-level synthesis of analog and mixed-signal systems from VHDL-AMS," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. Vol.22(11), pp. 1504 – 1520, Nov 2003.

[60] T. Kazmierski and F. Hamid, "Analogue integrated circuit synthesis from VHDL-AMS behavioural specifications," International Conference on Microelectronics, vol. Vol.2, pp. 585–588, May 12-15 2002.

[61] M. Krasnicki, R. Phelps, R. Rutenbar, and L. Carley, "MAELSTROM: efficient simulation-based synthesis for custom analog cells," Design Automation Conference (DAC), pp. 945 – 950, June 21-25 1999.

[62] R. Tortosa, J. de la Rosa, A. Rodriguez-Vazquez, and F. Fernandez, "A direct synthesis method of cascaded continuous-time sigma-delta modulators," IEEE International Symposium on Circuits and Systems (ISCAS), vol. Vol.6, pp. 5585 – 5588, May 23-26 2005.

[63] K. Mashiko, "Challenge and opportunity in analog and RF electronics," International Conference On ASIC (ASICON), vol. Vol.1, pp. 1205 – 1209, Oct 24-27 2005.

[64] Z. Ciota, A. Napieralski, and J. Noullet, "Analogue realisation of integrated FIR filters," IEE Proceedings of Circuits, Devices and Systems, vol. Vol.143(5), pp. 274 – 281, Oct 1996.

[65] F. A. Hamid, Architectural synthesis of analogue filters from behavioural VHDL-AMS descriptions. PhD thesis, School of Electronics and computer science, University of Southampton, 2004.

[66] A. Younis and R. Massara, "Automated synthesis of switched-capacitor ladder filters within an analogue silicon compilation environment," IEE Proceedings of Circuits, Devices and Systems, vol. Vol.139(2), pp. 249 – 255, 1992.

[67] C.-M. Chang, B. Al-Hashimi, Y. Sun, and J. Ross, "New high-order filter structures using only single-ended-input OTAs and grounded capacitors," IEEE Transactions on Circuits and Systems II: Express Briefs, vol. Vol.51(9), pp. 458 – 463, Sept 2004.

[68] E. Drakakis, A. Payne, and C. Toumazou, ""log-domain state-space": a systematic transistor-level approach for log-domain filtering," IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, vol. Vol.46(3), pp. 290 – 305, March 1999.

[69] M. Wang, C.-I. H. Chen, and S. Radhakrishnan, "Low-power 4-b 2.5-GSPS pipelined flash analog-to-digital converter in 130-nm CMOS," IEEE Transactions on Instrumentation and Measurement, vol. Vol.56(3), pp. 1064 – 1073, June 2007.

[70] E. Culurciello and A. Andreou, "An 8-bit 800-μW1.23-MS/s successive approximation ADC in SOI CMOS," IEEE Transactions on Circuits and Systems II: Express Briefs, vol. Vol.53(9), pp. 858 – 861, Sept 2006.

[71] Q. Wu and A. Wang, "A 12 bits/200 MHz resolution/sampling/power-optimized ADC in 0.25 μm SiGe BiCMOS," IEEE International Symposium on Circuits and Systems (ISCAS), vol. Vol.6, pp. 6174 – 6177, May 23-26 2005.

[72] Y.-I. Park, S. Karthikeyan, W. Koe, Z. Jiang, and T. Tan, "A 16-bit, 5MHz multi-bit sigma-delta ADC using adaptively randomized DWA," IEEE Custom Integrated Circuits Conference, pp. 115 – 118, Sept 21-24 2003.

[73] H. Zhang and A. Doboli, "Fast time-domain symbolic simulation for synthesis of sigma-delta analog-digital converters," International Symposium on Circuits and Systems (ISCAS), vol. Vol.5, pp. 125 – 128, May 23-26 2004.

[74] J. Ruiz-Amaya, J. de la Rosa, M. Delgado-Restituto, and A. Rodriguez-Vazquez, "Behavioral modeling simulation and high-level synthesis of pipeline A/D converters," IEEE International Symposium on Circuits and Systems (ISCAS), vol. Vol.6, pp. 5609 – 5612, May 23-26 2005.

[75] Y.-T. Chien, L.-R. Huang, W.-T. Chen, G.-K. Ma, and T. Mukherjee, "SPEED: synthesis of high-performance large scale analog/mixed signal circuit," IEEE International Symposium on VLSI Design, Automation and Test (VLSI-TSA-DAT), pp. 112 – 115, April 27-29 2005.

[76] R. Saleh, B. Antao, and J. Singh, "Multilevel and mixed-domain simulation of analog circuits and systems," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. Vol.15(1), pp. 68 – 82, Jan 1996.

[77] X. Li, X. Zeng, D. Zhou, X. Ling, and W. Cai, "Behavioral modeling for analog system-level simulation by wavelet collocation method," IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, vol. Vol.50(6), pp. 299 – 314, June 2003.

[78] R. Arora, U. Dasgupta, D. Hocevar, and L. Goff, "OASYS: a tool for aiding in design of high performance linear circuits," IEEE International Symposium on Circuits and Systems (ISCAS), vol. Vol.3, pp. 1911–1914, May 1-3 1990.

[79] F. El-Turky and E. Perry, "BLADES: an artificial intelligence approach to analog circuit design," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. Vol.8(6), pp. 680 – 692, June 1989.

[80] M. Hassoun and L. Huelsman, "Symbolic circuit analysis: an overview," Midwest Symposium on Circuits and Systems, vol. Vol.1.2, pp. 957 – 960, Aug 13-16 1995.

[81] G. Gielen, H. Walscharts, and W. Sansen, "Analog circuit design optimization based on symbolic simulation and simulated annealing," IEEE Journal of Solid-State Circuits, vol. Vol.25(3), pp. 707 – 713, Jun 1990.

[82] Q. Yu and C. Sechen, "A unified approach to the approximate symbolic analysis of large analog integrated circuits," IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, vol. Vol.43(8), pp. 656 – 669, Aug 1996.

[83] B. Murmann, P. Nikaeen, D. Connelly, and R. Dutton, "Impact of scaling on analog performance and associated modeling needs," IEEE Transactions on Electron Devices, vol. Vol.53(9), pp. 2160–2167, Sep 2006.

[84] G. Alpaydin, S. Balkir, and G. Dundar, "An evolutionary approach to automatic synthesis of high-performance analog integrated circuits," IEEE Transaction on Evolutionary Computation, vol. Vol.7(3), pp. 240 – 252, June 2003.

[85] G. Alpaydin, S. Balkir, and G. Dundar, "An evolutionary approach to automatic synthesis of high-performance analog integrated circuits," IEEE Transactions on Evolutionary Computation, vol. Vol.7(3), pp. 240 – 252, June 2003.

[86] N. K. Bose, Neural network fundamentals with graphs, algorithms, and applications. McGraw-Hill series in electrical and computer engineering. Communications and signal processing, McGraw-Hill, 1996. ISBN: 0071140646 (pbk).

[87] R. J. Vaccaro, Digital control: a state-space approach. McGraw-Hill series in electrical and computer engineering. control theory, McGraw-Hill, 1995. ISBN: 0070667810.

[88] B. Antao and A. Brodersen, "ARCHGEN: Automated synthesis of analog systems," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. Vol.3(2), pp. 231 – 244, June 1995.

[89] S. Somanchi and M. Manwaring, "Analog synthesis from behavioral descriptions," IEEE International Symposium on Circuits and Systems (ISCAS), vol. Vol.3, pp. 2079 – 2082, May 1993.

[90] G. Klir, Fuzzy sets and fuzzy logic: theory and applications. Prentice Hall, 1995. ISBN: 0131011715.

[91] K. Ogata, Modern control engineering. Prentice Hall, 2nd ed., 1990. ISBN: 0135891280.

[92] J. Yuan, N. Farhat, and J. V. der Spiegel, "GBOPCAD: A synthesis tool for high-performance gain-boosted Opamp design," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. Vol.52(8), pp. 1535 – 1544, Aug 2005.

[93] J. Carnahan and R. Sinha, "Nature's algorithms [genetic algorithms]," IEEE Journal of Potentials, vol. Vol.20(2), pp. 21 – 24, Apr-May 2001.

[94] J. Grimbleby, "Automatic analogue circuit synthesis using genetic algorithms," IEE Proceedings of Circuits, Devices and Systems, vol. Vol.147(6), pp. 319 – 323, Dec 2000.

[95] T. Sriver and J. Chrissis, "Combined pattern search and ranking and selection for simulation optimization," Winter Simulation Conference, vol. Vol.1, Dec 5-8 2004.

[96] S. Ebadi, K. Forouraghi, and S. Sattarzadeh, "Optimum low sidelobe level phased array antenna design using pattern search algorithms," IEEE Antennas and Propagation Society International Symposium, vol. Vol.1B, pp. 770–773, July 3-8 2005.

[97] A. Honkela, "Speeding up cyclic update schemes by pattern searches," Conference on Neural Information Processing, vol. Vol.1, pp. 18–22, Nov 2002.

[98] A. Zaouche, I. Dayoub, and J. Rouvaen, "Blind equalization via the use of generalized pattern search optimization and zero forcing sectionally convec cost function," Information and communication technologies (ICTTA), vol. Vol.2, pp. 2303–2308, April 2006.

[99] V. Torzcon, "On the convergence of pattern search algorithm," SIAM Journal on optimization, vol. Vol.7(1), pp. 1–25, 1997.

[100] R. Lewis and V. Torzcon, "Pattern search algorithm for bound constrained minimization," SIAM Journal on optimization, vol. Vol.9(4), pp. 1082–1099, 1999.

[101] R. Lewis and V. Torzcon, "Pattern search algorithm for linearly constrained minimization," SIAM Journal on optimization, vol. Vol.10(3), pp. 917–941, 2000.

[102] E. Christen and K. Bakalar, "VHDL-AMS - a hardware description language for analog and mixed-signal applications," IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, vol. Vol.46(10), pp. 1263 – 1272, Oct 1999.

[103] T. Kazmierski, "A formal description of VHDL-AMS analogue systems," Design, Automation and Test in Europe (DATE), pp. 916 – 920, Feb 23-26 1998.

[104] G. Domenech-Asensi, T. Kazmierski, J. Ruiz-Marin, and R. Ruiz-Merino, "Architectural synthesis of high-level analogue VHDL-AMS descriptions using netlist extraction from parse trees," IEE Journal of Electronics Letters, vol. Vol.36(20), pp. 1680 – 1682, Sep 28 2000.

[105] IEEE, IEEE Standard SystemC Language Reference Manual. March 31 2006.

[106] H. Al-Junaid and T. Kazmierski, "SEAMS - a SystemC environment with analog and mixed-signal extensions," International Symposium on Circuits and Systems (ISCAS), vol. Vol.5, pp. 281 – 284, May 23-24 2004.

[107] A. Vachoux, C. Grimm, and K. Einwich, "Extending SystemC to support mixed discrete-continuous system modeling and simulation," IEEE International Symposium on Circuits and Systems (ISCAS), vol. Vol.5, pp. 5166 – 5169, May 23-26 2005.

[108] J. Bruce, "Meeting the analog world challenge: Nyquist-rate analog-to-digital converter architectures," IEEE Journal of Potentials, vol. Vol.17(5), pp. 36 – 39, Dec - Jan 1998 - 1999.

[109] P. Aziz, H. Sorensen, and J. vn der Spiegel, "An overview of sigma-delta converters," IEEE Signal Processing Magazine, vol. Vol13(1), pp. 61 – 84, Jan 1996.

[110] R. Stewart and E. Pfann, "Oversampling and sigma-delta strategies for data conversion," Electronics and Communication Engineering Journal, vol. Vol.10(1), pp. 36 – 47, Feb 1998.

[111] R. Schreier, Understanding delta-sigma data converters. Wiley-Interscience, 2005. ISBN: 0471465852.

[112] M. Pontil and A. Verri, "Support vector machines for 3D object recognition," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. Vol.20, pp. 637 – 646, June 1998.

[113] J. Huang, X. Shao, and H. Wechsler, "Face pose discrimination using support vector machines (SVM)," International Conference on Pattern Recognition, vol. Vol.1, pp. 154 – 156, Aug 16-20 1998.

[114] T. Onoda, H. Murata, G. Ratsch, and K.-R. Muller, Experimental analysis of support vector machines with different kernels based on non-intrusive monitoring data, vol. Vol.3, pp. 2186 – 2191. May 12-17 2002.

[115] A. Smola and B. Scholkopf, Kernel machine software, 2002. Report available at `http://www.kernel-machines.org/index.html`.

[116] C.-C. Chang and C.-J. Lin, LibSVM: a library for support vector machines, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[117] M. Zorzi, F. Franze, and N. Speciale, "Construction of VHDL-AMS simulator in MATLAB," International Workshop on Behavioral Modeling and Simulation (BMAS), pp. 113 – 117, Oct 7 - 8 2003.

[118] R.-E. Fan, P.-H. Chen, and C.-J. Lin, "Working set selection using second order information for training SVM," Journal of Machine Learning Research.

[119] P.-H. Chen, R.-E. Fan, and C.-J. Lin;, "A study on SMO-type decomposition methods for support vector machines," IEEE Transactions on Neural Networks, vol. Vol.17(4), pp. 893 – 908, July 2006.

[120] M. Smith, WinSpice3 User's manual, June 10 2004. Software available at `http://www.ousetech.co.uk/winspice2/`.

[121] F. Herrera, M. Lozano, and A. M. Sanchez, "A taxonomy for the crossover operator for real-coded genetic algorithms: an experimental study," International Journal of Intelligent Systems, vol. Vol.18, pp. 309–338, 2003.

[122] B. Boser and B. Wooley, "The design of sigma-delta modulation analog-to-digital converters," IEEE Journal of Solid-State Circuits, vol. Vol.33(6), pp. 1298 – 1308, Dec 1988.

[123] M. Safi-Harb and G. Roberts, "Low power delta-sigma modulator for ADSL applications in a low-voltage CMOS technology," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. Vol.52(10), pp. 2075 – 2089, Oct 2005.

[124] G. W. Flake and S. Lawrence, "Efficient SVM regression training with SMO," Machine Learning, vol. Vol.46, pp. 271–290, 2002.

[125] S. Rabii and B. A. Wooley, The Design of Low-Voltage, Low-Power Sigma-Delta Modulators. Springer, 1998. ISBN: 0792383613.

[126] H. Zare-Hoseini, I. Kale, and O. Shoaei, "Modeling of switched-capacitor delta-sigma modulators in SIMULINK," IEEE Transactions on Instrumentation and Measurement, vol. Vol.54(4), pp. 1646 – 1654, Aug 2005.

[127] F. Medeiro, B. Perez-Verdu, A. Rodriguez-Vazquez, and J. Huertas, "Modeling opamp-induced harmonic distortion for switched-capacitor sigma-delta modulator design," IEEE International Symposium on Circuits and Systems (ISCAS), vol. Vol.5, pp. 445 – 448, May 30 - June 2 1994.

[128] P. Malcovati, S. Brigati, F. Francesconi, F. Maloberti, P. Cusinato, and A. Baschirotto, "Behavioral modeling of switched-capacitor sigma-delta modulators," IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, vol. Vol.50(3), pp. 352 – 364, March 2003.

[129] R. Khoini-Poorfard and D. Johns, "On the effect of comparator hysteresis in interpolative al modulators," IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1148 – 1151, May 3 - 6 1993.

[130] T. Lee, W. Lin, and D.-L. Lee, "Design techniques for low-voltage micropower cmos switched-capacitor delta-sigma modulator," IEEE Asia-Pacific Conference on Circuits and Systems, vol. Vol.1, pp. 249 – 252, Dec 6 - 9 2004.

[131] J. Ruiz-Amaya, J. de la Rosa, F. Medeiro, F. Fernandez, R. del Rio, B. Perez-Verdu, and A. Rodriguez-Vazquez, "Matlab/simulink-based high-level synthesis of discrete-time and continuous-time σ δ modulators," Design, Automation and Test in Europe Conference and Exhibition (DATE), vol. Vol.3, pp. 150 – 155, Feb 16 - 20 2004.

[132] S. Rabii and B. Wooley, "A 1.8-V digital-audio sigma-delta modulator in 0.8-μm CMOS," IEEE Journal of Solid-State Circuits, vol. Vol.32(6), pp. 783 – 796, June 1997.

[133] S. Brigati, F. Francesconi, P. Malcovati, and F. Maloberti, "A fourth-order single-bit switched-capacitor σ δ modulator for distributed sensor applications," IEEE Transactions on Instrumentation and Measurement, vol. Vol.53(2), pp. 266 – 270, April 2004.

[134] M. Graphics, "SystemVision user's manual," www.mentor.com, January 2005.

[135] D. Li and Y. Tsividis, "Active LC filters on silicon," IEE Proceedings of Circuits, Devices and Systems, vol. Vol.147(1), pp. 49 – 56, Feb 2000.

[136] C. Yue and S. Wong, "Physical modeling of spiral inductors on silicon," IEEE Transactions on Electron Devices, vol. Vol.47(3), pp. 560 – 568, March 2000.

[137] MOSIS, "MOSIS scalable CMOS (SCMOS)," tech. rep., The MOSIS Service, Oct 4 2004. Document available at `http://www.mosis.org/Technical/Designrules/scmos`.

[138] H. Greenhouse, "Design of planar rectangular microelectronic inductors," IEEE Transactions on Parts, Hybrids, and Packaging, vol. Vol.10(2), pp. 101 – 109, Jun 1974.

[139] J. Burghartz, D. Edelstein, M. Soyuer, H. Ainspan, and K. Jenkins, "RF circuit design aspects of spiral inductors on silicon," IEEE Journal of Solid-State Circuits, vol. Vol.33(12), pp. 2028 – 2034, Dec 1998.