UNIVERSITY OF CINCINNATI

Date: Oct. 29th, 2004

I, Glenn August Wolfe, hereby submit this work as part of the requirements for the degree of:

Doctor of Philosophy

in:

Computer Engineering

It is entitled:

Performance Macro-Modeling Techniques for Fast Analog Circuit Synthesis

This work and its defense approved by:

Chair: Dr. Ranga Vemuri
        Dr. Harold Carter
        Dr. Carla Purdy
        Dr. Donald French
        Dr. Ali Minai
Performance Macro-Modeling Techniques for Fast Analog Circuit Synthesis

A dissertation submitted to the
Division of Research and Advanced Studies
at the University of Cincinnati
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in the Department of Electrical and Computer Engineering
and Computer Science of the College of Engineering

Glenn August Wolfe
B.S. Electrical Engineering, University of Cincinnati, 1999

Advisor and Committee Chair: Dr. Ranga R. Vemuri

October 2004
Abstract

The synthesis of large digital integrated circuits is ubiquitous, highly developed, and efficient. Designs exceeding hundreds of millions of transistors are common and relatively inexpensive from a consumer viewpoint. Mature and reliable CAD tools exist for nearly every aspect of the digital design process. In contrast, the design of analog integrated circuits is often left to human designers utilizing first order approximation, experience, and iteration. Though the physical size of an analog block is typically orders of magnitude smaller than the digital part of a mixed signal IC, the analog design time often dominates.

One of the main challenges in analog design is sizing. Given an analog topology and a set of high-level design constraints, the size of all components in the circuit must be appropriately determined while hopefully meeting all design constraints. There are many ways to approach the sizing problem, from first-order approximation and manual design, to global optimization engines utilizing the full power of SPICE simulation.

An analog topology is typically characterized by a set of performance parameters used to succinctly quantify the properties of the circuit. The behavior of the performance parameters is often coupled and highly nonlinear. The numerical circuit simulator SPICE is often used as a benchmark of comparison for evaluating the performance of analog circuits. However, the computational requirements of running SPICE limit its use when attempting to evaluate a circuit's performances during circuit synthesis. Stochastic combinatorial optimization methods such as simulated annealing and genetic algorithms require the computation of performance parameters for a large number of circuit sizing alternatives. It is therefore beneficial to reduce the time associated with generating performance estimates.

This work focuses on the development of accurate and efficient performance parameter macro-models for use in the synthesis of analog circuits. Once constructed the mathematical models may be used as substitutes for full SPICE simulation, providing efficient computation of performance parameter estimates. In this thesis, we explore various modeling architectures, develop and apply two unique sampling methodologies for adaptively improving model quality, and attempt to apply the sizing rules methodology in order to perform dimensional reduction and ensure proper operation of analog circuits.

In order to properly create an analog performance model, a training data set is needed to create the model, and an independent validation data set is needed to verify the accuracy of the model. The training and validation data sets are comprised of discretely sampled points in the design space. Various methods exist for generating these sample points. A static sampler does not take into account the shape of the function under scrutiny, whereas an adaptive sampler strives to reduce modeling error through strategic placement of sample points. Two unique adaptive sampling methodologies are developed in this work and are applied to various analog circuit performance metrics. The first adaptive sampler is based on piecewise-linear models while the second sampler uses more complex non-linear pseudo-cubic splines. It is shown experimentally that both adaptive samplers are capable of improving maximum modeling errors for various performance metrics and analog topologies. Strategic placement of costly sample points improves model quality while reducing the time needed to create the performance models. Adaptive sampling also alleviates human intervention during model construction, realizing an automatic framework for sampling and modeling performance parameters.

The sizing rules method and feasibility region modeling are analyzed and applied to analog performance macro-modeling in an attempt to automatically reduce the dimensionality of the design space, simplify performance parameter behavior, and ensure proper DC biasing of analog circuits. A feasibility region is a portion of the design space satisfying design space and electrical space inequality constraints generated by the sizing rules method. Experimental evidence indicates that the sizing rules method alone does not sufficiently constrain a circuit to facilitate the creation of accurate analog performance macro-models. Additional, manually derived design constraints are required to enable the development of accurate performance parameter models.
Acknowledgements

This work was supported in part by an Ohio Board of Regents PhD Enhancement Program Fellowship, the Defense Advanced Research Projects Agency and the Sensors Directorate of the Air Force Research Laboratory, U.S. Air Force, Wright-Patterson AFB, OH, under Contract F33615-01-C-1977, the Dayton Area Graduate Studies Institution (DAGSI) under the DAGSI/AFRL Research Program Project SN-UC-01-08, and the National Science Foundation under award number CCF-0429717.
Table of Contents

CHAPTER 1 Introduction ......................................................... 4
  1.1 Digital vs. Analog Circuit Synthesis .................................. 4
  1.2 Related Work in Analog Macro-Modeling ............................... 7
  1.3 Research Approach and Overview ....................................... 9
  1.4 Thesis Outline .............................................................. 14

CHAPTER 2 Macro-Modeling Overview .......................................... 16
  2.1 Response Surface Methodology .......................................... 16
  2.2 Piecewise Linear Modeling .............................................. 17
  2.3 Symbolic Modeling ........................................................ 18
  2.4 Posynomial Templates .................................................... 18
  2.5 Neural Networks .......................................................... 19
  2.6 Basis Functions with Global Support ................................... 20
  2.7 Projection Pursuit Regression .......................................... 21
  2.8 MARS - Multivariate Adaptive Regression Splines .................. 21
  2.9 Conclusions ............................................................... 22

CHAPTER 3 Multi-Dimensional Function Exploration ............................ 23
  3.1 The Curse of Dimensionality ............................................. 24
  3.2 Domain Constraints ...................................................... 25
    3.2.1 Box Constraints - The Unit Hypercube ........................... 25
    3.2.2 Polygonal Constraints .............................................. 26
    3.2.3 Implicitly Defined Constraints ................................... 26
  3.3 Static Sampling Methods ................................................. 26
    3.3.1 Gridded ............................................................... 26
    3.3.2 Pseudo-Random ..................................................... 27
    3.3.3 Quasi-Random and Design of Experiments ....................... 28
  3.4 Dynamic Sampling Methods - Adaptive Sampling ....................... 28
  3.5 Conclusions ............................................................... 29

CHAPTER 4 Analog Performance Metrics ....................................... 30
  4.1 SPICE ................................................................. 30
    4.1.1 Transistor Models .................................................. 31
    4.1.2 Analysis Types .................................................... 31
  4.2 The Operational Amplifier Circuit Class and Associated Performance Metrics ......................................................... 32
  4.3 Test Bench Design ....................................................... 36
  4.4 Practical Extraction of Performance Parameters ....................... 38
  4.5 Conclusions ............................................................... 39

CHAPTER 5 Modeling and Synthesis of Analog Circuits using Neural Network Models ................................................................. 40
8.2.2 On the imperative accuracy of performance macro-models ...................... 128
8.3 Summary ................................................................. 129

Bibliography ................................................................. 130
1 Introduction

Moore’s law, while as steadfast as the day it was conceived [41], [46], has not satiated our need for computational power. Even with steady and exponential increases in computing power, many numerical problems remain intractable and difficult enough to require pools of computing machines realizing only sub-optimal, approximate solutions. Circuit synthesis, either digital or analog, is one such task.

1.1 Digital vs. Analog Circuit Synthesis

Circuit synthesis is the process of determining, from high-level, abstract specifications, the low-level details achieving those specifications. In a digital synthesis setting, the high-level specification is given in a hardware description language (HDL) closely resembling modern programming languages. Strict and well defined levels of hierarchy exist for the representation of digital circuits. Each level abstracts away unnecessary details so that the design process can proceed efficiently. As a testament to the maturity and stability of digital circuit synthesis, single chip designs exceeding 100 million transistors are common and relatively inexpensive.

Mixed signal integrated circuits are important for many applications such as communications and signal processing. The same common CMOS technologies utilized for digital circuitry can also be used to realize many analog circuits such as op-amps, filters, and analog to digital converters. CMOS analog integrated circuit design is a well studied topic for which many books exist [2], [5], [74]. These textbooks typically explain CMOS analog design using simplified first-order transistor equations. The use of simple transistor models for the analysis of analog circuits allows designers to build intuition about circuit behavior without complicating the design procedure with higher-order transistor effects. The performance of circuits sized using first-order equations must typically be refined by iterative simulation and adjustment of component sizes due to inaccuracies in the first order models. This type of manual and iterative design methodology is responsible for the large gap between digital and analog design times. While the analog part of a mixed signal circuit is usually much smaller than the digital part, the design time for the analog circuitry often either equals or exceeds the digital part.
A clear and widely excepted synthesis flow for analog circuits does not exist. Over the past few years several companies have developed industry grade synthesis tools for analog integrated circuits [11], [7]. These tools however, lack the hierarchical structure of digital synthesis methodologies. Automatic creation of novel analog topologies is very difficult and research on the topic is rather limited. No commercial tools even attempt to tackle topology synthesis. The main task of modern analog synthesis tools is to size the components of a specific topology such that all performance specifications are met.

SPICE is a general purpose, time-domain simulation tool that has become the de facto standard for verification of analog integrated circuit designs. In order to simulate a circuit, the sizes of all components, as well as the magnitude of independent sources must be known. An input signal is applied and the response of the circuit observed. SPICE is inarguably one of the most important tools for analog designer’s, however, it is not without its limitations. For interactive analysis of small circuits (around a hundred components) execution times of SPICE are typically not more than just seconds, especially if transient simulations are not needed. This sounds reasonable until one considers automating the sizing of circuits; the use of globally convergent stochastic optimization engines such as genetic algorithms may require the simulation of tens of thousands of circuit sizing alternatives. Run times then become prohibitive in the worst case scenario, that is when SPICE is directly used to analyze circuit configurations during synthesis. A common attribute of many of the current commercially available synthesis tools is the use of SPICE in the loop to actively provide performance attributes of sized circuits. This often results in software requiring pools of computers working in parallel to efficiently size circuit topologies.

One method of dealing with the problem of long simulation times involves constructing comprehensive, accurate and efficient models for the performances of analog sub-blocks. Once constructed these models stand in for the prohibitively expensive SPICE simulations during circuit synthesis. The macro-models map the sizes of components in the analog topology to many important performance metrics. Although there is overhead in building the models, they may be reused many times to efficiently synthesize the topology. The models are also capable of capturing complex sub-micron effects, thereby allowing new technologies to be readily used, without burdening the designer with the problem of nonlinear and interdependent circuit performances.

This thesis discusses methods for the construction of accurate, and efficient nonlinear macro-models for analog circuit performance metrics. Although the techniques discussed in this thesis are generic, emphasis is placed on the analysis and modeling of operational amplifier perfor-
 Operational amplifiers are ubiquitous in analog systems with a large set of widely accepted performance metrics. A typical synthesis flow utilizing SPICE as a performance estimator is shown in Figure 1-1. Because SPICE can simulate the behavior of any circuit, the technology, topology and test bench files may be separate inputs, thereby making “SPICE in the loop” a very general sizing methodology. The modified sizing flow using performance models is shown in Figure 1-2. During model creation the technology, topology and test bench circuits are fully specified, causing this information to be intrinsic to the analog performance models. One of the drawbacks of macro-modeling is that an entirely new model must be constructed for each desired technology, topology and test bench combination. Another drawback is that performance models do not have perfect accuracy, whereas SPICE is considered highly accurate (within the limits of the transistor models). However, the
Macro-models are much faster (approximately 40,000 times faster using neural networks) than SPICE when estimating performances. Macro-modeling trades versatility and accuracy for speed. The problem of versatility may be partially solved by compiling a technology dependent library of performance models for common analog topologies. A designer can quickly size any characterized topology from the library in just seconds. The problem of creating accurate analog performance macro-models with a reasonable number of sample points is extensively addressed in this thesis. Many black-box modeling approaches and sampling methodologies are discussed. In addition, two novel methods are developed for adaptively sampling analog performance functions. It is shown experimentally that the adaptive sampling techniques improve model accuracy through strategic placement of sample points while requiring fewer computationally costly calls to SPICE.

1.2 Related Work in Analog Macro-Modeling

Knowledge based analog CAD methods take advantage of circuit specific information. This information must typically be manually formulated and is specialized for each type of circuit or topology. In [4], ARCHGEN, a state based system, was used to store circuit specific information. BLADES [83] is an expert system which uses a custom designed rule set in order to make design decisions during synthesis. Specific to the design of operational amplifiers, OPASYN [53] relies on decision trees and analytical circuit models designed specifically for each topology considered. Knowledge based systems work well but the effort of encoding the knowledge used is often very high and must be repeated for any new circuit topologies.

Simulation based approaches rely on a general circuit simulator such as SPICE to enable dynamic knowledge acquisition of circuit performance. MAELSTROM [54], DELIGHT.SPICE [65], and Anaconda [72], [73] use full SPICE simulation to evaluate circuit performance during optimization. ASTRX/OBLX [66], [67] uses simulated annealing and a simplified simulation method called AWE that is faster but less accurate than SPICE. SPICE is also used in [40] along with hierarchical decomposition and macro modeling in the synthesis of analog topologies. Due to the computational complexity of full circuit simulators such as SPICE, multiple parallel computers are often employed to reduce simulation times when exploring many designs using a combinatorial search algorithm. Although extraction of data for use with neural networks is expensive, it is a one time cost per topology. Once the models are developed, execution times are very fast, leading to a considerable reduction in synthesis time. However, when directly employing SPICE during synthesis, any topology can be readily handled. Neural network models require an extraction step which is specific to each topology.
Symbolic techniques discussed in [32], [33], [34], [79] use linearized circuit model equations to perform performance parameter estimation. Symbolic techniques can be automated but the equations can become very large and are usually restricted to purely linear formulations. The equations can be simplified by removing terms that have little effect on accuracy in order to reduce their prohibitive size. The generated equations can provide analog designers with valuable insight into the interaction of design trade-offs. Neural networks are “black-box” models and typically cannot offer any qualitative insight into the behavior of a system. However, they can be used to readily model hyper-dimensional and non-linear functionality.

Table lookup methods for function approximation can handle large nonlinearity, are fast and accurate but require storage of all points in the sampled data space. Neural networks offer a more succinct representation of a multi-dimensional input-output mapping while maintaining speed and similar accuracy. Neural networks also offer an intrinsic ability to interpolate function values not in the table [90].

Global optimization techniques [42], [44], [61] such as convex geometric programming require behavior models of circuits using posynomial functions. These methods typically require manual formulation of the posynomial constraints. However, in [15] posynomial models were constructed using data generated by SPICE. While it is possible to find the global optimum using convex geometric formulations, this optimum is still governed by the simplified posynomial function used to model the circuit.

Neural networks have been used widely in the RF and Microwave CAD. In [14] S-parameters for multi-port RF components were successfully modeled using neural networks. In [25] HSPICE was used to generate transient simulation data for a MOSFET. This data was used to train a recurrent neural network which could be substituted for the MOSFET model to produce transient waveforms. In [36] neural networks were used to reduce simulation time for large-signal device modeling.

Geometric programming and posynomial systems of equations offer the tantalizing promise of “optimal” solutions to the synthesis of large analog systems [5], [8]. However, the posynomial equations are usually manually created by an expert designer. In addition, the “optimal” solution to the posynomial models and constraints is only as accurate as the tailored equations. Efforts have been made to automate the creation of posynomial models based on SPICE simulation outputs [2], however, accuracy of the models suffers due to the constraining nature of the posynomial equation template. In [14] neural network models were used to map circuit component sizes into various op-
amp performance metrics for several different op-amp topologies. The neural network models were embedded in a genetic algorithm optimization loop which allowed the synthesis of op-amp transistor sizes given a set of user specified constraints. Reuse of simulation data from commercial sizing tools was considered for use in creating analog macro models by Liu et al. in [7]. Regression techniques were used to model subsets of a large set of simulation data to produce models with acceptable accuracies.

An adaptive sampling scheme using piecewise linear models was proposed in [15]. A least squares plane is fit to local areas of discrete data in order to detect regions of nonlinearity. These regions are sampled more densely than linear regions in order to improve maximum model error. This methodology was applied to two analog circuit topologies and modeling accuracy was compared to gridded sampling and uniform pseudo-random sampling. A major drawback to this method is the instability of the piecewise linear models used. Because of large errors near the boundaries of the performance manifolds, the function domain (a unit hypercube) was arbitrarily truncated by 0.1 units on each side to remove the regions with unacceptable error. For a five dimensional function, removing 0.1 units from each face of the hypercube leaves a sub-space only 33% the size of the original unit hypercube. This is a significant loss of viable design alternatives.

In [4] Gräb restricts the size and shape of the domain considered for modeling by building hierarchical constraints based on proper device operating regimes. It was demonstrated that box constraints are inferior to more complicated restrictions on device sizes when model accuracy is of concern. Their method is automated with an initial effort in judiciously deciding on hierarchical building blocks and constraints on proper device operation.

### 1.3 Research Approach and Overview

The initial premise of this work was the development of comprehensive analog macro-models for use in a synthesis system. Utilizing SPICE directly as a performance analyzer provides accuracy and generality, however the time needed to size an analog topology can take days even when utilizing a pool of parallel computers. Other techniques exist for analog performance macro-modeling but many rely on detailed analysis and manual extraction of design equations or make parametric assumptions about the underlying system. This work focuses on removing assumptions on the underlying topologies and alleviating as much manual effort as possible from the macro-modeling process. Treating analog topologies as black-box functions with observable inputs and outputs transforms the modeling process into a very general problem of non-parametric estimation of an unknown manifold. Since nothing is assumed about the shape of the performance functions,
an estimate of their shape must be constructed from discrete samples. Taking the sizes of components of a topology as inputs and real valued performance metrics as outputs, each analog topology can be characterized by various well known modeling methodologies capable of estimating high-dimensional nonlinear surfaces. Each modeling methodology has strengths and weaknesses, therefore an analog macro-modeling tool should offer the user several choices of models.

As an initial effort, some simplifications were made to the modeling problem. First, analog topologies were manually analyzed to determine basic matching constraints typical of structures such as differential pairs and current mirrors. Since matching reduces the total number of free variables in the circuit, this process is known as dimensional reduction of the input space. In addition, the widths of all transistors were fixed to some minimum feature size, thereby eliminating many of the free design variables. Using transistors with fixed lengths and variable widths is a common practice in analog design (although ideally we would like to “free-up” the lengths as true variables). Surprisingly many analog topologies containing as many as 33 transistors may be reduced to as few as 5 to 6 meaningful free variables. Intuitively it makes sense to have as few free input variables as possible when attempting to build models over a considerable range of component sizes, but as will be discussed later, the problem of modeling high-dimensional functions has deeper roots than just intuition. The phenomenon of inherent sparsity of points, while well known as the curse of dimensionality, has only recently emerged as an active research field. The validity of distance metrics such as the Euclidean norm has been called into question for many uses such as nearest neighbor search. Luckily these results imply that common norms used for distance measures begin to deteriorate around 10 to 15 dimensions. All models in this thesis are of five and six dimensions, however, some modeling methodologies used such as pseudo-cubic splines rely on the Euclidean norm thus calling into question the potential accuracy for high-dimensional models. It appears that there is no quick fix to the curse of dimensionality and it may come into play when attempting to model analog circuits with many free variables.

With the simplifications described above, six performance metrics for three op-amp topologies were successfully modeled using neural networks. Neural networks are well studied and provide a safe platform for modeling high-dimensional functions from discrete samples. All performance models were based on full gridded sampling over a normalized unit hypercube domain with components ranges of one order of magnitude (ex. 20\(\mu\)m-200\(\mu\)m). Gridded sampling usually produces high quality models, however, the number of sample points grows polynomially w.r.t grid density and exponentially w.r.t. dimensionality rendering gridded sampling impractical for high-dimensional functions. The neural performance models proved accurate and very fast (about 40,000
times faster than SPICE) and were used to replace SPICE altogether in an analog synthesis framework. A detailed setup of the synthesis system is provided along with tables of sized topologies under various performance constraints. Using the neural models as performance estimators op-amp topologies can be sized in just seconds. In addition the mathematical macro-models are very stable and easy to use compared to SPICE. During the DC root solving process SPICE may fail to converge and consequently fail to return a valid value for performance metrics. This phenomenon is documented and illustrated by an example in which an “invalid” design point is made valid by changing the size of one transistor by less than one percent. This inherent instability does not cause a problem for macro-models since the removal of a few of the training samples does not invalidate the model or severely hinder its accuracy.

With confidence that accurate analog performance models could be created and cast into a synthesis setting, focus switched to improving model quality through intelligent placement of costly sample points. Though two adaptive sampling methodologies are successfully developed in this thesis, the task was formidable with many false starts and difficulties. To the authors knowledge no adaptive sampling algorithms allowing unstructured placement of points for high-dimensional functions exist in the literature.

The first adaptive sampler is based on Delaunay piecewise linear (PL) models and works by detecting and sampling non-linear regions in the analog performance functions. Nonlinear regions are sampled with priority while linear or constant regions are sparsely sampled. The accuracy of the PL adaptive models was experimentally shown to be superior to pseudo-random and gridded sampling w.r.t. maximum modeling error for an op-amp topology and RF-mixer. In addition the component sizes were allowed to vary over a larger range of two-orders of magnitude as opposed to one (ex. 2\(\mu\)m-200\(\mu\)m, 0.1pF-10pF). There are, however, problems inherent to PL models in high-dimensions. First, the number of “pieces” comprising the model increases exponentially w.r.t. dimension. Second, large modeling errors can occur near the boundary of the function domain. For this reason the region of validity for the models was restricted to a hyper-cube with 0.8 units per side, centered within the original unit hypercube function domain.

The second adaptive sampling algorithm works directly in conjunction with Duchon pseudo-cubic splines (PCS). The basic premise is that if a spline is well defined, i.e. the function is sufficiently sampled, the removal of one point from the spline should not deform its shape too drastically. The sampling strategy is based on the relative magnitude of changes in the spline structure when points are added and deleted. The PCS adaptive sampler was applied to two artificial test functions as well as the performance metrics of an operational amplifier with components again
spanning two orders of magnitude (ex. 2µm-200µm, 0.1pF-10pF). The PCS adaptive models are shown to be superior to quasi-random and gridded sampling w.r.t. maximum modeling error. They also do not suffer from large modeling errors near the domain boundary as do piecewise linear models. However, there is a practical upper limit of approximately 5000 points that can be used to collectively assemble a pseudo-cubic spline model. This is due to the inherent instabilities involved in creating the spline as well as memory and computational requirements. This limit can be overcome via division of the domain into locally modeled sub-spaces, individually requiring less than 5000 points. Pseudo-cubic splines have no free parameters and are completely determined by the discrete sampled data, thus requiring no human interaction for proper construction, unlike neural network models. They are also independent of domain dimension with the understanding that performance may be affected by the curse of dimensionality.

As stated previously, the development of the two adaptive sampling methodologies required considerable effort. However, they remove much of the human interaction required to build an accurate model. For comparison, the parameters of a neural network model (number of neurons, depth of network, training time) must typically be defined and tuned by the user. In addition the training and validations sets must be generated and then used to train and validate the neural model. The adaptive samplers require no human interaction. They take as input an unknown function and strategically place sample points such that model quality is maximized. Both the piecewise linear model and pseudo-cubic model are exactly defined by the input data. In addition there is evidence that there is no need for a validation set of data since the errors resolved by the adaptive sampler are usually the largest errors occurring in the model. By monitoring the errors resolved by the adaptive sampler, insight may be gained as to the overall accuracy of the model without even testing the model using an independent data set.

An interesting aspect of both adaptive samplers is there inherent coupling to a respective modeling methodology. Each sampling algorithm must be used in conjunction with it’s corresponding modeling methodology to observe the benefits of designed sample point placement. The sample points chosen by the piecewise linear adaptive sampler will not prove useful when pseudo-cubic splines are utilized as the modeling methodology and vice-versa. This is due to the difference in the resulting shape of each model given a set of sample points. Clearly a piecewise linear model will favor an abundance of sample points when curvature of the function is the highest. However, pseudo-cubic splines, which are based on third order polynomial functions, can readily model non-linear functions with less sample points than needed for piecewise linear models.
While adaptive sampling alleviates much of the manual effort required for model creation, robust test circuits are imperative to ensure stability and convergence of an automated system for the extraction of performance metrics. Robust test bench design is currently a manual effort, however, analog topologies may usually be grouped into pin compatible classes for which a single test bench applies to many topologies. Not only must raw data be read from SPICE, it must be carefully analyzed in order to properly extract the true performance value. If significant measurement noise exists, estimators (such as neural networks) must be employed rather than pure interpolators (such as pseudo-cubic splines). A library of robust and carefully constructed test bench circuits is essential to automating the macro-modeling process.

In Chapter 5 and Chapter 6, the design parameters of analog topologies are manually constrained by fixing all transistor lengths and using matching conditions commonly employed in analog circuit design in an attempt to achieve proper biasing and good behavior of performance metrics. Detailed knowledge and experience in analog design is required to realize the matching constraints and to choose which design variables should be fixed, and which should be left as free parameters. The sizing rules method attempts to automatically constrain a circuit to ensure proper operation, good behavior of performance metrics, insensitivity to process variations and robust manufacturability [24][38][77][91]. The sizing rules method is examined to explore its potential application to analog performance macro-modeling. Ideally we would like to fully automate the process of selecting and constraining the design variables of a circuit to further automate the macro-modeling process.

The sizing rules methodology uses design space and electrical space constraints to define a feasibility region or what has also been termed the non-pathological [38] design space. In order to efficiently apply the sizing rules method to analog performance macro-modeling, it is necessary to model the feasible region to eliminate the need to execute SPICE when testing the feasibility of a design. Two methods for modeling the feasible space defined by the sizing rules method are developed. The performance parameters of two operational amplifier circuits are modeled in box-constrained, approximately feasible, and truly feasible design regions. Experimental evidence indicates that the sizing rules method alone is insufficient to properly constrain a circuit for the purposes of analog performance macro-modeling. It is necessary to impose additional constraints on the design variables of a circuit in order to facilitate accurate modeling of the performance metrics. When the design variables of a circuit are properly constrained, little, if any improvement in modeling accuracy is observed when the models are restricted to the feasible design region. Unfortunately, the need for manual analysis of analog circuits is not prevented by the inclusion of the sizing rules
method. However, numerous performance macro-modeling experiments in this thesis, utilizing a variety of sampling and modeling methodologies, support the fact that fast and accurate performance macro-models are useful to reduce synthesis times of analog circuits.

1.4 Thesis Outline

Chapter 2 surveys many popular methods for modeling discretely sampled multi-dimensional functions. In general there is no “best method” for black-box function modeling. Both parametric and nonparametric techniques are discussed and where applicable, methods utilized for modeling analog performances are mentioned.

In Chapter 3 various methods for the acquisition of sample points used in the modeling process are presented. The curse of dimensionality is introduced with several examples emphasizing the inherent sparsity of sample points in high-dimensional spaces. Static sampling methods such as gridded, random, and design of experiments techniques are useful for initial exploration of black-box functions, however, dynamic sampling methods intelligently place sample points in nonlinear regions in order to optimize model accuracy.

Analog performance metrics and SPICE simulation is discussed in Chapter 4. Special attention is paid to operational amplifiers with a detailed taxonomy of various performance metrics for this class of circuits. Robust test bench design is essential for the automatic extraction of analog performance parameters. An example test bench circuit is presented that “closes the loop” at DC yet allows AC signals to be processed in the desired open loop configuration. Lastly, practical aspects of performance parameter extraction are discussed. Specifically one-dimensional spline interpolates are used to interpolate discrete Bode plot data. In addition a reverse interpolation methodology is presented enabling the extraction of various performance metrics.

A detailed illustrative example in analog macro-modeling and synthesis using neural networks and a genetic algorithm optimization engine is presented in Chapter 5. Various performance metrics for three different operational amplifier topologies, ranging in size from 8 transistors up to 33 transistors, are successfully modeled using artificial neural networks. A genetic algorithm is used to synthesis one of the op-amp topologies under various performance constraints. The neural network performance models provide more the 4 orders of magnitude speedup in performance estimation. Synthesis utilizing the fast neural performance models takes only seconds. In addition the inductance of a layout aware spiral inductor is modeled to further convey the utility of the methodology.
Adaptive sampling is the process of judiciously sampling new points while considering the history of previous sample points such that model quality is maximized. Chapter 6 presents two novel algorithms for adaptively sampling analog performance metrics. The first adaptive sampler utilizes piecewise linear models and is shown through an analog modeling example to produce superior models with respect to traditional gridded and random sampling methods. The second adaptive sampling algorithm utilizes more complex nonlinear pseudo-cubic spline models. The adaptive sampler is applied to two artificial test functions as well as performance metrics for an operational amplifier. The adaptive sampler strives to reduce maximum modeling error and produces more accurate models than gridded and quasi-random sampling methods. The conversion gain of an RF mixer is also adaptively sampled and modeled to further support the methodology.

In Chapter 7 the sizing rules method and feasibility region modeling are incorporated into macro-model generation. Research suggests that designs constrained to the feasible design region defined by the sizing rules method are automatically constrained to ensure proper operation, good behavior of performance metrics, insensitivity to process variations and robust manufacturability [24][38][77][91]. Performance models for a single-stage op-amp are constructed using basic sizing rules over box-constrained, approximately feasible and truly feasible design regions. Experimental evidence suggests that basic sizing rules are insufficient to properly constraint a circuit for macro-modeling purposes. Additional manually derived constraints similar to those used in Chapter 5 and Chapter 6 are required for building accurate macro-models. The feasible design is modeled using nearest neighbor and power series methodologies. By building models inside the approximate feasible region, designs are robustly designed as defined by the sizing rules methodology.

Chapter 8 presents results and conclusions for the thesis. The pseudo-cubic spline adaptive sampler is reviewed and future extensions are discussed. The effects of variability in the manufacturing process and inaccuracies in transistor models on performance macro-modeling accuracy is also discussed.
### 2 Macro-Modeling Overview

Modeling of analog performances can be divided into two disciplines, knowledge based or parametric, and black box or nonparametric. In a parametric setting, knowledge of the circuit structure and components is utilized to simplify the modeling process. For example, in symbolic techniques, nonlinear devices are represented by linear sub-circuits with a fixed number of components. These linear sub-circuits which stand in for the complicated transistor device equations are governed by a set of numeric model parameters whose values determine the behavior of the model. If the prototype function, i.e. the fixed parametric assumptions, about the underlying system are incorrect, the estimate may lie far from the true function regardless of how many training samples are available. In a nonparametric setting, very little is assumed of the circuit and the behavior must be inferred by simulating the circuit at various points in the design space; from a set of discrete data points an appropriate model must be inferred.

Analog modeling may also be broadly categorized by observing the level of human interaction necessary during the model building process. Many methods require almost no interaction to build an accurate model while others require parameters to be tuned interactively. Since we are concerned with automating the synthesis of analog circuits using macro-models, we must be weary of the level of user interaction required to build acceptable models.

#### 2.1 Response Surface Methodology

Response surface methods (RSM) [8],[10],[51], involve designing experiments, executing the chosen experiments, and predicting new responses based on the results of the experiments. More often than not the optimization of some objective function is also involved; the prediction of new responses via the response surface model is used during optimization as a surrogate for costly simulation. Response surfaces in general are not coupled to a particular modeling methodology such as splines, radial basis functions or neural networks. RSM is a formalization of experimentation and the acquisition of knowledge in a structured framework.

**Design of Experiments** - Traditionally, independent variables in RSM are referred to as factors. Each factor influences the response variable, thus defining an underlying manifold. The
goal is to choose a set of factors which will maximize information obtained by executing the experiments. Favorable designs include full factorial (the corners of a hyper-cube), fractional factorial (only a subset of the hypercube corners), maximum entropy, latin hypercubes, pseudo-random (stochastic), quasi-random (deterministic); the list goes on. None of these “designs” consider the outcome of previous experiments, that is they ignore the shape of the manifold when picking new simulation points. They are all essentially ways to pick unbiased experiments capable of resolving features of the surface in an efficient manner. Full factorial designs are the most desirable, however, they are often not feasible when many factors are present.

**Execution of Experiments** - Experiments are assumed to be prohibitively expensive. Often the experiments may be clinical studies, chemical or industrial experiments, or time consuming computer simulation. The outcome of the experiments may or may not contain noise.

**Model Building** - By far the most popular modeling methodology in RSM is linear and quadratic regression. While linear and quadratic regression are not the most accurate regression schemes available, they are simple, fast and often quite valid when modeling only a small neighborhood in the design space. Many other modeling methodologies have been used successfully in RSM such as splines, statistical methods, neural networks, etc. When noise is present in the samples, an estimator with smoothing capabilities such as neural networks or polynomial regression must be employed.

**Optimization/Synthesis** - Often the models derived from experimental data are used as substitutes for simulation. Gradient descent methods may be used to optimize an objective function within some small neighborhood. Once the optimization engine travels outside the valid region for a given model, the process of design of experiments, execution and model building is reiterated in the new neighborhood of interest.

### 2.2 Piecewise Linear Modeling

A viable application for piecewise linear models [57], [75] is modeling device physics to reduce computational cost (as opposed to evaluation of a complex nonlinear model) and to improve the convergence properties of the Newton-Rhapson method used to solve for the DC operating point of a circuit. High dimensional piecewise linear models using unstructured data can be constructed by tesselation of the sample points. The tesselation consists of a union of many sub-spaces known as simplexes. Inside each simplex, the function is modeled as a linear hyper-plane. Unfortunately the number of simplexes in a tessellation grows exponentially with both dimension and the number of sample points [29]. Despite this drawback, piecewise linear models based on the Delaunay cri-
terion were used to realize an adaptive sampling algorithm for black box functions [86] along with specific examples in analog circuit performance modeling.

2.3 Symbolic Modeling

Symbolic techniques [32], [33], [34], [79] use linearized device equations to construct algebraic expressions for many analog performance parameters. The number of terms in expanded symbolic expressions increases exponentially with the size of the circuit. Symbolic equations in fully expanded form are limited to circuits with at most 15 transistors. The equations can be simplified by removing terms that have little effect on accuracy in order to reduce their prohibitive size. Approximate symbolic equations consisting of only dominant terms allow designers to gain intuitive insight into the behavior of the circuit. When approximate symbolic equations are utilized the number of allowable transistors increases to 25.

Symbolic methods using hierarchical decomposition divide a circuit into loosely coupled sub-circuits which are analyzed independently and later combined to form a nested symbolic expression with much fewer terms than the expanded version. The computational complexity of hierarchical decomposition techniques increases linearly w.r.t. circuit size and therefore allows the analysis of large analog circuits at the sacrifice of human interpretability. Although most work in symbolic analysis restricts the devices in a circuit to purely linear approximations, work has been done allowing soft and hard nonlinearity in the device equations [63].

Symbolic modeling captures the behavior of a circuit over a large range of frequencies as opposed to performance manifold modeling techniques where the behavior of the circuit is modeled only at a specific frequency. For example, when modeling open loop gain, the entire transfer function is captured by symbolic equations, whereas only the open loop gain at DC is usually modeled using nonparametric methods.

2.4 Posynomial Templates

Geometric programming and posynomial systems of equations offer the tantalizing promise of “optimal” solutions to the synthesis of large analog systems [44], [61]. However, the posynomial equations are usually manually created by an expert designer. In addition, the “optimal” solution to the posynomial models and constraints is only as accurate as the tailored equations. Despite these drawbacks posynomial equations have been successfully used to synthesize analog circuits such as operational amplifiers.
Efforts have been made to automate the creation of posynomial models based on SPICE simulation outputs [15]. The independent variables were taken as bias voltages and bias currents of the circuit. A nominal value for these voltages and currents is chosen and a hypercube proportional to the nominal point of size 0.01 and 0.1 is chosen for the modeling region. A somewhat nonstandard “quality of fit” is defined to assess the accuracy of the posynomial models for points not used to create the models. The accuracy results are given in percentage, however the error metric is somewhat misleading. Normally percentage normalizes the error by the value of the actual result, however, the authors choose to normalize the error by the total maximum range of output values. In addition the quality of fit metric is the average of modeling error. This does not give sufficient information about the worst case error occurring in the models. All reported “quality of fit” metrics were less than 10% for all of the performance metrics over the largest hypercube, i.e. 0.1 times the nominal design voltages. The relationship between bias voltages/currents and actual device sizes is not reported, making it difficult to compare the results in [15] to previous modeling approaches that vary the device sizes directly. Some work [88] has reported performance models with valid regions spanning over two orders of magnitude. It is apparent that the posynomial models have a limited range of validity, especially when a hypercube of approximately 0.1 on each side is compared to a hypercube of two orders of magnitude on each side for other works.

2.5 Neural Networks

Neural networks have become one of the most popular platforms for function learning because they can readily model hyper-dimensional and non-linear functionality. Neural networks are typically trained with a discrete set of data points called the training data set. A second set of data points not present in the training data set is traditionally used to validate the neural network model. Neural networks are black-box models and typically cannot offer any qualitative insight into the behavior of a system. Training of neural models is not a fully automatic process. The topology of the network must usually be manually tuned in order to balance bias and variance, although some automatic methodologies exist, such as Bayesian regulation and cross validation for automatically tuning network size.

In [87] neural network models were used to map circuit component sizes into various op-amp performance metrics for several different op-amp topologies. The neural network models were embedded in a genetic algorithm optimization loop which allowed the synthesis of op-amp transistor sizes given a set of user specified constraints. Neural networks have received much attention in
the RF area of analog electronics as well with concrete and plentiful results justifying the book by Zhang and Gupta on the topic[89].

2.6 Basis Functions with Global Support

Tensor product splines require that data points are on a regular grid and that none of the points are missing. This presents a problem when modeling multi-dimensional functions since the number of sample points increases exponentially w.r.t. grid density and dimension. Splines using global support functions alleviate this problem and allow data points to be scattered. A unique basis function with one free parameter is allocated to each sample point and the surface is approximated by the weighted sum of all basis functions. Since the shape of every basis function affects the height of the surface globally, all scaling coefficients for the basis functions must be precisely balanced in order to reproduce the surface exactly. This requires the solution to a dense system of linear equations which requires $O(n^3)$ computational complexity where $n$ is the number of sampled points.

Thin plate splines are perhaps the most popular form of spline interpolation allowing scattered data. They use a quadratic basis function and a supporting kernel polynomial whose degree increases with increasing dimension. This presents an inconvenience for high dimensional situations since a high order polynomial must be constructed. Pseudo-cubic splines are closely related to thin plate splines but use a cubic basis function and require only a linear support polynomial [23]. This alleviates the problem of designing a polynomial function of arbitrary dimension and arbitrary order.

A major advantage to pseudo-cubic splines is that they can be solved for exactly and do not require interactive adjustment of parameters. Due to the computational complexity of dense matrix inversion, there is a maximum of about 5000 points when constructing a spline interpolate. This problem has been addressed for the two dimensional case using thin plate splines by employing finite element methods to reduce the computational complexity required to solve for the spline, however, these methods do not directly generalize to high dimensional situations. Another method for reducing the computational complexity of building the spline is to divide the data points into regions and then solve for splines containing fewer data points over each region individually. Pseudo-cubic splines were used in conjunction with adaptive sampling to successfully model various analog performance metrics in [88].
2.7 Projection Pursuit Regression

Projection pursuit regression (PPR) and feed-forward neural networks (FNN) are closely related. Both methods first project the inputs using a linear transformation, and then nonlinearily transform these projected vectors using an activation function. The final output is a linear combination of the nonlinear activation functions. All of the activation functions in a neural network are typically identical (the tangent and sigmoid functions being popular choices) while the activation functions for PPR are derived directly from the data. The free parameters in a PPR model are gradually estimated and additional free parameters are iteratively added, as opposed to FNN where all parameters are optimized simultaneously. As with neural networks, it is touted that PPR models are capable of approximating any continuous function to any degree of accuracy. In practical situations with finite sample size the bias-variance trade-off is a good stopping criterion for model complexity.

Although PPR can provide results on par with other regressors such as neural networks, there are no exact guidelines for construction [76]. The choice in number of activation functions to use is left to the user and optimization routines must be used to find the appropriate weight vectors for recombining the support functions.

2.8 MARS - Multivariate Adaptive Regression Splines

Multi-resolution methods such as wavelets typically derive high-dimensional basis functions via tensor products from a one-dimensional case. This causes an exponential increase in the number of feasible high-dimensional basis functions. The goal is to find the optimal set of basis functions from the very large set of potential functions that may be combined in order to reproduce the data points accurately and smoothly. So far this problem has not been solved exactly. MARS attempts to build a model by searching the space of potential basis functions and adding candidate basis functions according to some goodness of fit functional. Once a large number of candidate basis functions have been chosen (i.e. a very complex model for the data has been found), the model is strategically pruned such that the well known bias variance trade-off is optimized.

MARS is an estimator, that is it cannot exactly fit the training data. For this reason it is better suited for estimation of functions with random noise components. However, some research has supported that MARS can find fitting solutions superior to that of neural network estimation. My limited experience with MARS, using generally default settings, resulted in inferior models compared to both pseudo-cubic splines and neural networks.
2.9 Conclusions

There are many modeling approaches useful for modeling analog performance metrics. Symbolic simulation is very well studied and contains many analog modeling examples in the literature. It can be fully automated and is applicable to large circuits when hierarchical decomposition is employed. The use of neural networks for circuit modeling is also abundant. The ease of use and computational efficiency of neural network models is attractive, although some interaction is usually required to build an accurate and unbiased neural estimator. Somewhat new to applications in analog modeling, splines utilizing global basis functions offer exact interpolation of data points and their construction is easily automated, however, there is a theoretical limit to the number of points used to build the splines. The sheer number of different modeling methodologies attests to the fact that no one method is superior. Multiple modeling methods should be made available to the user of an analog modeling tool.
3 Multi-Dimensional Function Exploration

We will now consider surfaces described by a function mapping a $d$-dimensional space to a one dimensional real number. A function is said to be $C^k$ continuous if the $k$-th derivative of the function is without discontinuities. For the case of $k=0$, we are only guaranteed that the zeroth derivative of the function is continuous, in other words the function itself is continuous but may have cusps and sharp bends resulting in discontinuous higher order derivatives. A function with a one-dimensional domain is simply a curve in two dimensions; in a two dimensional domain the function is a surface existing in three dimensions; for any domain with dimension $d$ greater than two the function is a hyper-surface living in $d+1$ dimensions.

This chapter discusses the exploration of unknown functions when the only means of “looking at” the function is to repeatedly sample discrete points on the surface of the function. Gradient and higher order derivatives are only available using finite difference formulas and require the evaluation of multiple samples near the point at which the derivative is of interest. The end goal of this sampling process is to acquire sufficient knowledge about the underlying function such that an accurate model may be built that is capable of standing in for the true function. A core assumption here is that the function is worthy of being modeled, that is, evaluation of the function is costly and substitution with a model will yield computational savings. One such situation explored in this thesis occurs in analog integrated circuit macro-modeling. Circuit simulation via SPICE maps the component sizes of the circuit into a variety of performance metrics. Each metric is described by a real valued number at each simulation point in the domain space thus describing a high dimensional surface.

There are a variety of methods for sampling an unknown surface. Probably the most popular choice is uniform gridded sampling in which a fixed sampling interval is chosen in each dimension. While gridded sampling is suitable for low-dimensional functions, the exponential dependence of the number of sample points on both dimension and sample density invalidates its use for most higher dimensional functions. This exponential increase in sample points is strong motivation for the use of unstructured methods, i.e. random sampling and design of experiments.
concepts such as \textit{latin hypercubes}, \textit{maximum entropy designs}, etc. All of these methods strive to create an unbiased set of sample points, that is they attempt to pick sample points that will not miss features of the surface due to structure in the placement of the sample points. For example, gridded sampling is orthogonally biased and may miss “diagonal” features of the function.

While it is important to sample the function in an unbiased fashion for initial exploration, all the methods discussed thus far sample independently of the shape of the function. Intuitively, a higher density of sample points should be placed in regions where the function is the most nonlinear, while sampling should be sparse in linear or constant regions. A novel contribution of this thesis is the development of two such methods for \textit{adaptively sampling} a function. These methods strive to enhance model accuracy by strategically placing sample points while considering both the past sampling history as well as the current shape of the function. Employing these adaptive sampling strategies improves modeling accuracy and allows models to be built with less sample points, thus fewer costly simulation runs.

This chapter begins with a discussion of high dimensional functions. An intuitive as well as mathematically structured knowledge of high dimensional functions is necessary to understand the difficulty involved in approximating functions from discretely sampled points. The “curse of dimensionality” is presented and the implications discussed. Constraints on the size and shape of the function domain such as box constraints and simplicial complexes are considered along with popular sampling methodologies used to explore the function. Finally novel techniques for adaptively sampling high-dimensional functions are presented.

\subsection{3.1 The Curse of Dimensionality}

Friedman and Stuetzle give a thought experiment illuminating the problem of inherent sparsity of points and nearest neighbor search in high dimension [30]. They consider a ten-dimensional unit hypercube with each dimension subdivided into ten equally spaced intervals yielding a grid of $10^{10}$ constituent sub-hypercubes. A set of uniformly and randomly distributed points are scattered inside the unit hypercube. Given one point in the space and a hypercubal search neighborhood around this point of 0.1 units on each side, the probability of finding a neighboring point is $0.1^{10}$ multiplied by the total number of sample points. In contrast, if the search is instead constrained to 10\% of all sample points, the resulting search area is a hyper-cube of approximately 0.8 units on a side. In other words, constraining a search for nearest neighbors to a fixed volume has the effect that the probability of finding a neighboring point is very small, while constraining the search area to encompass a fixed portion of the samples results in a vast neighborhood. This implies that sample
points in high-dimensional domains are inherently sparse. Another way to phrase this phenomenon is to realize that of all ten dimensions any one of them may cause the points to be distant. In order for two points to be close w.r.t. the Euclidean norm, the distance between the points must be small in all ten dimensions, an unlikely chance that only diminishes with increasing dimension.

Recently there has been a flurry of research addressing the problem of sparsity in high-dimensional space, i.e. the so called *curse of dimensionality*. In [9] it is observed that, under the Euclidean norm, as the number of dimensions increase, the distances from a query point to its nearest and farthest neighboring points is nearly indistinguishable. Essentially all points in the space are separated by approximately the same distance. They indicate that this effect takes place with as few as 15 dimensions and imply that the nearest neighbor criterion is essentially unreliable and contains little meaning in higher dimensions.

Since the Euclidean norm inherently causes sparsity of points and an invalid, fuzzy distance measure for high-dimensional spaces, Aggarwal argues against the Euclidean norm and in favor of the manhattan distance measure [1]. They further explore the notion of fractional norm measures and show that this typically unorthodox measure of distance outperforms the traditional norms used in low-dimensions spaces. They note that caution should be used in choosing an appropriate norm when analyzing high-dimensional spaces. It is not quite clear where current research addressing the curse of dimensionality will lead. However, attention is quickly being drawn to misuse of the Euclidean norm in many of the data mining, image recognition, and various other applications requiring nearest neighbor search in high-dimension.

### 3.2 Domain Constraints

The modeling problem must be simplified by constraining the function domain to a valid region of interest. This section discusses three types of constraints that are used in subsequent experiments in this thesis.

#### 3.2.1 Box Constraints - The Unit Hypercube

The simplest type of domain constraints are one-dimensional inequalities usually realized by a vector of upper and lower bounds for each design variable. Each variable can then be scaled, either linearly or nonlinearly, into the interval of [0,1] resulting in the well known unit hypercube [87]. While box constraints are simple and natural for an analog designer, they can result in circuits with highly nonlinear behavior. This concept of pathological circuits is discussed in Chapter 7 and arises primarily from violation of good design practices. Despite the drawback of the potential vio-
lation of important design practices, accurate analog macro-models are successfully constructed for a variety of performance metrics in Chapter 5 and Chapter 6.

3.2.2 Polygonal Constraints

More advanced constraints may be enforced on a domain using linear or nonlinear equality and inequalities. Equality constraints must be differentiated from equality constraints since the dimension of the domain is reduced by one for each equality constraint. Transistor matching in a differential amplifier is an example of dimensional reduction using equality constraints. Inequality constraints do not reduce the domain dimension, but instead prune away some part of it. When all inequality constraints are linear and the domain space is taken as the intersection of sub-spaces defined by inequalities, the domain is reduced to a convex polytope. Non-convex domains can occur when the inequalities are nonlinear or if the domain is a union of convex polytopes.

3.2.3 Implicitly Defined Constraints

Consider the multi-dimensional mapping \( y = f(x) \) where \( x \in \mathbb{R}^m \), \( y \in \mathbb{R}^n \). The function \( f \) is a black-box function and has no explicit inverse. Inequality constraints enforced on \( x \) implicitly define constraints on \( y \) and vice versa. Since \( x \) is the vector of independent variables, constraints posed on the domain of \( x \) do not cause a problem because we are free to choose any \( x \) we wish. However, constraints posed on the domain of \( y \) can only be ensured by executing the function \( f(x) \) and subsequently checking the constraints on \( y \). In the case that evaluation of \( f \) is computationally expensive, building an image of the constraints on \( y \) by sampling points in \( x \) becomes costly and difficult.

3.3 Static Sampling Methods

The sampling methods discussed in this section are considered static because they place sample points independently of the sampling history and current state of the model.

3.3.1 Gridded

While gridded sampling often produces very good models, it is only practical when the domain is box constrained. In addition the number of sample points increases exponentially as \( n = k^d \) where \( k \) is the grid density and \( d \) is the domain dimension. For functions of even moderate dimension, gridded sampling quickly becomes an unacceptable solution. Even so, gridded sampling often produces high quality models, and therefore will often be used in this thesis as a mark of comparison to other sampling methodologies.
3.3.2 Pseudo-Random

Pseudo-random sampling is typically produced using the familiar random number generators available in nearly every computer language today. Pseudo-random generators require a seed, a number on which to base all future generated numbers. Once the seed is chosen, the generation of numbers is deterministic. However, even a small change in the seed results in a completely different sequence of random numbers. The distribution of numbers produced is usually uniform over the interval \([0,1]\), although more specialized generators are available for producing Gaussian, or normal distributions.

Random sampling typically produces sub-optimal models when compared with gridded sampling. This may be attributed to clumping of sample points in localized areas as well as a reduced number of points on the boundary when compared to gridded sampling. The boundary problem can be easily explained with a short example. Consider a five dimensional hypercube sampled with five points in each dimension for a total of \(5^5 = 3125\) sample points. The number of sample points on the boundary of the domain is \(5^5 - 3^5 = 2882\); nearly all of the sample points for gridded sampling are placed on the boundary. Random sampling on the other hand can only place sample points near the boundary and never on it.

![Figure 3-1. Two dimensions sampled with a pseudo-random generator. Note the “clumping” of points in some areas and the sparsity in others.](image)
3.3.3 Quasi-Random and Design of Experiments

The goal of a quasi-random sampler is to avoid clumping of sample points and to strive for a more uniformly distributed set that is still random in nature. Unlike pseudo-random generators which are deterministic yet can be seeded to produce a completely new sequence of numbers, quasi-random generators lack the concept of a seed and will always produce the same sequence of numbers. As the quasi-random sequence proceeds, the density of sample points increases while maintaining a similar distance between neighboring points. An example of a quasi-random sequence known as the Halton sequence is shown in Figure 3-2.

Quasi-random sampling often produces slightly better models than pseudo-random sampling [86] but still suffers from fitting problems near the boundary of the function domain. Gridded sampling nearly always outperforms any of the random sampling methods. A Halton sequence is used in to produce analog circuit macro-models in this thesis in Chapter 6.

Design of experiment methods [31] blur with that of quasi-random generators. Essentially, there are many ways to create sample points that are unbiased with respect to some measure. One example, maximum entropy designs, rely on concepts from Shannon’s entropy theorem to ensure that points are placed forming minimum energy (maximum entropy). Regardless, design of experiment methods can usually only be expected to build an adequate model for very small neighborhoods. They are useful for constructing local linear or quadratic approximations of a function but do not typically provide a good framework for comprehensive and global model building.

3.4 Dynamic Sampling Methods - Adaptive Sampling

All previously discussed sampling methods are ignorant of the behavior of the function being sampled; all functions are sampled without regard for the shape of the surface. There are two ways model accuracy may be improved\(^1\). We may change the very nature of the model, for example,

\(^1\) Assuming here that dimensional reduction and variable transformations have been performed to a satisfactory degree.
moving from a piecewise linear to thin plate spline model, or control the placement of sample points such that model quality is maximized in each specific modeling situation. We define adaptive sampling as the process of judiciously sampling new points while considering the history of previous sample points and the nature of the function such that model quality is maximized.

Two unique methods cohering to the criterion of adaptive sampling are fully developed in Chapter 6. One method utilizes a Delaunay piecewise linear tessellation of the function while the second method uses a more advanced nonlinear Duchon pseudo-cubic spline as the modeling platform. Both adaptive sampling algorithms are capable of enhancing modeling accuracy while requiring fewer costly queries to the function being modeled. The algorithms are capable of consistently outperforming other static sampling methods discussed such as gridded, pseudo-random and quasi-random techniques. The adaptive samplers are also fully automatic and require no human interaction when sampling and building an appropriate model.

3.5 Conclusions

Analog performance metrics can be realized as hyper-surfaces with component sizes as independent variables. These functions are discretely sampled in order to explore the shape and behavior of the function such that an efficient macro-model may be constructed. As dimension increases, sample points become inherently sparse; this phenomenon is known as the curse of dimensionality and makes operations such as nearest neighbor search difficult. Static sampling methods such as gridded and random sampling ignore the shape of the function, placing sample points independently of function behavior. Intelligent placement of sample points may be used to maximize model quality. Two novel adaptive sampling techniques are discussed in Chapter 6.
Most low to medium frequency analog integrated circuits are verified in detail using the circuit simulation tool SPICE. This simulator has three primary analysis types, 1) DC bias point analysis, 2) small signal AC analysis, and 3) large signal transient analysis. In addition SPICE is able to analyze other attributes of a circuit such as noise, however, we will focus almost exclusively on the three primary types of analysis when building performance parameter macro-models. SPICE has been in use for over 20 years and is often used as a benchmark for comparing the relative accuracy of alternative schemes for approximating the performance of analog circuits.

This chapter gives details on executing SPICE to effectively extract performance metrics. Although the methodologies are generic in that all circuits are treated as black boxes, emphasis is placed on operational amplifiers and their associated performance metrics. The theoretical definitions for popular performance metrics are discussed as well as the practicalities associated with extraction from true SPICE data. Robust test bench design is also discussed and several possible test bench circuits are presented.

4.1 SPICE

SPICE, the de facto standard for circuit analysis, is a powerful simulation tool for verifying both digital and analog IC designs. While there are many commercial variants of the SPICE simulator touting advanced features such as graphical interfaces, the basic publicly available SPICE2G.6 from U.C. Berkeley [92] is often powerful and robust enough to handle most analog simulation settings. Based on SPICE2G.6, but now supported by “The NG-Spice Project” [93], the free and publicly available variant “ngspice” version 14 was used for all circuit simulations in this thesis unless otherwise specified. The word SPICE will be used to refer to usage of the circuit simulator “ngspice”.

4.1.1 Transistor Models

Without loss of generality, all circuit topologies in this thesis contain only CMOS transistors as well as linear resistors, capacitors and inductors. In order to simulate an integrated circuit in SPICE, appropriate transistor models must be supplied. These models are typically physically based with parameters extracted from experimental data. MOSIS [95], a company for integrated circuit development is popular with universities and government agencies. They offer a variety of manufacturing processes with transistor feature sizes down to the sub-micron range of 0.13µm. Various SPICE transistor models are also available for download from the MOSIS site, perhaps the most well known in order of increasing potential accuracy, the LEVEL 1, LEVEL 3, and BSIM3v3 models. The Berkeley BSIM3v3 transistor models [94] have become the industry standard for accurate sub-micron integrated circuit simulation. BSIM3v3 device models obtained from the MOSIS web site were used exclusively for all simulation based experiments in this thesis. Allen and Holberg [2] offer a good review of the previously mentioned MOS models.

4.1.2 Analysis Types

The first step in circuit simulation is to find the DC bias point, i.e. the exact node voltages and device currents resulting from the application of static supply voltages. While the operating point of a circuit in itself may be useful for checking transistor saturation conditions or other voltage/current based constraints, AC and transient analysis also require a DC operating point. The node voltages serve as a linearization point for nonlinear device equations in AC analysis, and as initial conditions for differential equations when performing transient analysis.

Even under the assumptions that a circuit contains no floating nodes or shorted loops (netlist problems), and that the DC bias point is inherently stable and unique, SPICE is not guaranteed to converge to the correct DC bias point [68]. DC analysis requires the solution to a general system of nonlinear equations for which there is no known globally convergent algorithm. However, we will later see that robust test bench design can aid in ensuring proper circuit biasing, alleviating many of the problems of DC non-convergence.

During AC analysis, all nonlinear device equations are linearized about the DC operating point, resulting in a circuit containing only linear components, i.e. the well known linear time invariant or LTI system. Although SPICE does not allow the symbolic computation of transfer functions,
any number of discrete points may be numerically evaluated resulting in a sampled image of the true underlying Bode plot. Any node in the circuit may be specified as an input to, or output of, the LTI system. Many important performance metrics, such as open loop gain, unity gain frequency, common mode rejection, and power supply rejection can be extracted from AC SPICE data.

The most time consuming type of SPICE simulation is usually a transient analysis. Transient analysis requires the solution of time dependant nonlinear differential equations. Any node voltage or current may be monitored and extracted providing a discretely sampled waveform over a user specified simulation time and resolution. Transient simulations are necessary to expose the large signal behaviors of the circuit under test such as slew-rate, and settling time.

### 4.2 The Operational Amplifier Circuit Class and Associated Performance Metrics

Deciding on an exact set of performance metrics for a circuit class is a subjective matter. Consultation of IC data sheets can provide a nearly overwhelming number of potential performance parameters. Typically, these data sheets simplify the problem somewhat by posting assumptions on operating conditions such as load, frequency, and temperature. Companies often provide only several typical test conditions, with no method for interpolating between these few discrete examples. While a stand alone chip must be able to drive a variety of loads, there is much more control over loading conditions when designing an integrated circuit responsible for a small function in a larger system. For example, when designing a CMOS op-amp as part of an analog to digital converter, the designer knows that the load driven by the op-amp will be in some small range (perhaps 0.1pF to 1pF) and will be almost purely capacitive due to CMOS gate inputs.

It is not practical to represent a performance metric with a single real number at multiple frequency, temperature and load conditions; perhaps the problem of modeling performance parameters is simply too difficult for all temperatures, all loads and all frequencies. Some assumptions must be made about these conditions in order to simplify the problem to a mapping of many (the device sizes) to one (the performance metric). There exists no master list of performance metrics associated with a class of analog circuits. Metrics to include during design must be those that are important to the designer and can be extracted via simulation.

In this section the analog circuit class of a single ended operational amplifier will be presented. Operational amplifiers are possibly the most common building blocks in analog systems. They vary greatly in complexity from less than 8 transistors to 50 or more transistors and serve a
multitude of purposes from simple buffering to precise instrumental sensing. The test benches and performance metrics discussed are generic to the class and independent of the specific op-amp topology.

• **Input Offset Voltage** - $v_{os}$ - DC

Input offset voltage can be simulated using SPICE, however the value obtained from simulation will be meaningless due to the statistical nature of $v_{os}$. A meaningful value for $v_{os}$ cannot be obtained for a sized topology since it is affected by random perturbations in CMOS process parameters such as transistor size and threshold voltage [2], [3].

• **Open Loop Gain** - $A_0$ - AC

The magnitude of the ratio of output voltage to differential input voltage is the open loop gain of an op-amp. Typically open loop gain is large and constant from DC to some 3dB frequency where the gain begins to roll off. Open loop gain can be extracted by running an AC analysis at one point of sufficiently low frequency so that the measured open loop gain is equivalent to the DC open loop gain; this is the quantity of interest for macro-modeling. Unless intrinsically stabilized with special circuitry, input offset voltage usually makes it impossible to accurately simulate gain with the op-amp in an open loop test configuration [2]. Application of a common mode voltage results in the input offset voltage being amplified by the open loop gain, which typically results in the output node of the op-amp straying significantly from the desired quiescent point (i.e. zero volts for a dual supply system); the output of the op-amp may even saturate at either the positive or negative supply rails yielding significantly erroneous bias conditions. Although it is possible to test for and determine the exact input offset voltage through multiple DC analyses, there is a more elegant solution to correctly and constantly biasing an op-amp while retaining an open loop configuration. By using fictitious and arbitrarily large capacitors and inductors to enforce negative feedback at DC while allowing AC signals to pass in an open loop fashion, a proper DC bias can be obtained and the AC performance metrics successfully extracted.

This trick of “closing the loop” at DC and “opening the loop” for AC signals is not new and variations on this theme can be found in many CMOS analog design books [2]. A unique example of such a test bench circuit is presented in Section 4.3. The use of robust test circuits is imperative to ensure stability and convergence of an automated system for the extraction of performance metrics.

• **3dB Frequency** - $f_{3dB}$ - AC
As mentioned, the open loop gain of an op-amp remains constant over a large range of frequencies until the dominant pole of the op-amp becomes active. This dominant pole is a central characteristic among nearly every op-amp design and is often necessary to insure stability when the op-amp is placed in a closed loop configuration. The 3dB frequency is the point at which the open loop gain of the op-amp is attenuated from its nominal value (at DC) by a gain of -3dB. The method of reverse spline interpolation (Section 4.4) is used to obtain the frequency at which the 3dB attenuation of open loop gain occurs. If the magnitude of the open loop transfer characteristic crosses the 3dB threshold more than once the lowest frequency occurrence is recorded.

• **Unity Gain Frequency - UGF - AC**

  The unity gain frequency is the point at which the open loop gain of the op-amp deteriorates to unity; it is also the frequency at which phase margin and gain margin are measured. Very similar to 3dB frequency, the reverse spline interpolation method is employed to find a close approximation to the true unity gain frequency with the lowest frequency crossing taken as the unity gain frequency.

• **Phase Margin - PM - AC**

  Phase margin is a measurement of the inherent stability of an op-amp and is measured at the unity gain frequency. The phase of an output signal, due to an applied differential AC signal, always begins at zero for low frequency and then changes as frequency increases and the poles and zeros of the circuit become active. Causality ensures that the output signal can only be a delayed version of the input signal, and thus beyond the dominant pole the output signal progressively becomes more “out of phase” with the input signal. Of particular interest is the phase of the output signal at the unity gain frequency. If the phase of the output signal is delayed by more than one half cycle (180 degrees), a signal that is fed back negatively will no longer be “negative”, but will in fact have switched sign resulting in positive feedback and a potentially unstable circuit. A positive phase margin means that the phase of the output signal is not delayed by more than 180 degrees at the unity gain frequency implying that the op-amp is inherently stable. A negative phase margin means that the output signal is delayed by more than one half cycle and the op-amp may become unstable in closed loop configurations. Assuming that the phase plot from SPICE begins at zero degrees and has been properly “unwrapped” (see Section 4.4), phase margin is calculated by subtracting 180 degrees from the phase of the output signal at the unity gain frequency. Even though an op-amp with a phase margin greater than zero is inherently stable, the prudent designer usually requires a phase margin of at least 45 degrees to insure robust stability, i.e. a margin of stability. Typically phase
margin cannot be greater than 90 degrees, as long as the gain of the op-amp is greater than approximately 10dB.

• **Gain Margin - GM - AC**
  
  Gain margin is closely related to phase margin in the sense that it refers to a margin of stability. Phase margin is measured precisely at the frequency when the open loop gain falls to unity; gain margin is measured at the frequency of zero phase margin, i.e. the point at which negative feedback transitions to positive feedback. Any gain greater than unity at the point of zero phase margin will cause instability, and therefore the gain margin is measured relatively from this point. When expressed in dB, a positive gain margin occurs when the open loop gain is less than one, while a negative gain margin occurs when the open loop gain is greater than one at the frequency of zero phase margin. A robust design usually requires a gain margin of at least 10dB. Again the method of reverse interpolation in Section 4.4 must be employed in order to find the frequency at which zero phase margin occurs.

• **Common Mode Rejection Ratio - CMRR - AC**
  
  The main purpose of an op-amp is to amplify differential signals while rejecting common mode signals. The magnitude of the ratio of differential gain to common mode gain is the common mode rejection ratio. We can no longer call upon a single SPICE simulation in order to calculate CMRR since we must apply both a differential and a common mode signal. SPICE allows only one active AC input source at a time. Similar to open loop gain, the common mode gain is measured in an open loop configuration, thus requiring the use of robust test bench circuits. Common mode gain is typically minimal at DC and remains constant over a large range of low frequencies. At higher frequencies the common mode gain increases and the open loop gain decreases resulting in degradation of the common mode rejection ability of the op-amp. For the purpose of macro-modeling, measurement of CMRR is taken at a single, sufficiently low frequency.

• **Power Supply Rejection Ratio - PSRR - AC**
  
  Transients on the power supply rails affect the output voltage of the op-amp. The ability of the op-amp to reject these signals in favor of differential voltage amplification is called power supply rejection ratio. Similarly to CMRR, the PSRR is normalized by the open loop gain of the op-amp and typically expressed in dB. PSRR is constant near low frequencies and begins to deteriorate, or to allow signals from the supply line to pass to the op-amp output as frequency increases. For the purpose of macro-modeling, measurement of PSRR is taken at a single, sufficiently low frequency.

• **Slew Rate - SR - TRAN**
Slewing of the op-amp output signal cannot be observed through simulation using AC analysis in SPICE. Since all nonlinear components in the circuit are linearized around a DC operating point during AC analysis, the circuit is simplified to a linear time invariant system and nonlinear transformations of the input cannot occur. Using a transient simulation, any arbitrary signal may be applied to the op-amp. One could apply a sine wave with sufficiently high frequency to induce slewing at the output, however, it is again prudent to utilize a robust test bench to more efficiently facilitate extraction of the performance metric. Time driven simulation becomes costly if high frequencies are present in the circuit and long simulation times are needed to observe the desired effect. Deliberately placing the op-amp in a positive feedback configuration, thus forcing it to act as a comparator with hysteresis, ensures two things; the output signal most definitely slews during transition from rail to rail, and the point at which slewing occurs is controllable. Some SPICE simulation engines have an adaptive system for placing sample points in high frequency regions, however, many do not, such as ngspace v.14. When SPICE does not facilitate adaptive time sampling, it may become difficult to ensure sufficient simulation resolution for proper performance parameter extraction while avoiding extremely long simulation times. An example test bench and practical extraction methodology is given in Chapter 5.

4.3 Test Bench Design

It is often difficult to perform simulations on an op-amp in an open loop configuration due to high open loop gain [1]. We have found this to be true while developing our own test bench circuits for performance parameter extraction. SPICE is sometimes unable to solve for the correct DC operating point of an op-amp in an open loop configuration. In real world applications, an op-amp is typically used in a closed loop configuration utilizing negative feedback. Negative feedback guarantees that the inherent input offset voltage \( v_{os} \) of the op-amp does not cause the output voltage to saturate for reasonable common mode input voltages. Due to \( v_{os} \), applying zero input to an op-amp in an open loop configuration typically results in the output being saturated at the positive or negative voltage rail or sufficiently far from a satisfactory quiescent voltage to significantly affect the linearized small signal transistor models [1]. These considerations led to the development of a robust test bench circuit which takes advantage of stable closed loop DC characteristics while retaining the simplicity of an open loop AC simulation. The proposed test bench circuit is illustrated in Figure 4-1, with suggested component values in Table 4-1. The test bench is used to simulate data sufficient for the extraction of six AC performance parameters of a CMOS op-amp topology (namely A0, PM, UGF, CMRR, positive and negative PSRR).
An informal analysis of the proposed test bench circuit is now presented. When solving for the DC operating point of a circuit using SPICE, all capacitors and inductors are considered ideal and therefore treated as open circuits and short circuits respectively. This allows the test bench circuit to stabilize the op-amp output voltage using artificially induced negative feedback via the two dependant sources $E_{fbn}$ and $E_{fbp}$, which both simply copy and amplify the op-amp output voltage back to the op-amp inputs. The output of the op-amp is forced very close to zero at DC with the inputs of the op-amp splitting the required input offset voltage necessary to maintain zero volts at the output. During an AC analysis, the inductors and capacitors in the test bench circuit do not affect the frequency response of the of the op-amp under test due to the extremely low frequency 3dB points induced by these components. For any useful frequency ranges, the common mode and differential mode AC input voltages ($v_{cm}$, $v_{dm}$) are passed unhindered to the input terminals of the op-amp.

![Figure 4-1](image)

Figure 4-1. Robust test bench circuit for single-ended operational amplifiers. At DC negative feedback avoids faulty biasing, and at AC the true open loop behavior of the op-amp is unhindered.

<table>
<thead>
<tr>
<th>Table 4-1</th>
<th>Recommended component sizes for robust test bench Circuit</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Edn$</td>
<td>$0.5 \times V_{dm}$</td>
</tr>
<tr>
<td>$Edp$</td>
<td>$0.5 \times V_{dm}$</td>
</tr>
<tr>
<td>$Ep1$</td>
<td>$0.5 \times V_{dc1}$</td>
</tr>
<tr>
<td>$Ep2$</td>
<td>$0.5 \times V_{dc2}$</td>
</tr>
<tr>
<td>$En1$</td>
<td>$0.5 \times V_{ac2}$</td>
</tr>
<tr>
<td>$En2$</td>
<td>$0.5 \times V_{dc2}$</td>
</tr>
<tr>
<td>$E_{fbp}$</td>
<td>$1e3 \times V_{out}$</td>
</tr>
<tr>
<td>$E_{fbn}$</td>
<td>$1e3 \times V_{out}$</td>
</tr>
<tr>
<td>$Cp$, $Cn$</td>
<td>$1e6 , F$</td>
</tr>
<tr>
<td>$Lp$, $Ln$</td>
<td>$1e6 , H$</td>
</tr>
<tr>
<td>$R_{Cp}$</td>
<td>$1e6 , \Omega$</td>
</tr>
<tr>
<td>$R_{Ln}$</td>
<td>$1e6 , \Omega$</td>
</tr>
<tr>
<td>$V_{cm}$</td>
<td>0V or 1V</td>
</tr>
<tr>
<td>$V_{dm}$</td>
<td>0V or 1V</td>
</tr>
<tr>
<td>$V_{p}$</td>
<td>0V or 1V</td>
</tr>
<tr>
<td>$V_{n}$</td>
<td>0V or 1V</td>
</tr>
</tbody>
</table>
4.4 Practical Extraction of Performance Parameters

The AC transfer function data from SPICE is typically sampled using logarithmic spacing with the number of points per decade specified by the user. The imaginary and real parts of a Bode plot are much better behaved than the resulting magnitude and phase plots. The magnitude plot is usually not that poorly shaped, however the phase plot wraps around between $-\pi$ and $+\pi$ due to the nature of the tangent function used to calculate phase. The slope of the phase plot is sometimes so steep that it may be difficult to decide if a transition between $\pm\pi$ should be unwrapped without additional sample points in the ambiguous region. For these reasons it makes sense to model the real and imaginary parts of the discretely sampled Bode plots using cubic spline interpolates. From the spline models one can compute the magnitude and phase at any desired frequency point and successfully unwrap the phase under any circumstance. In Figure 4-2 the real and imaginary data points (circles) of an op-amp open loop transfer function are shown along with a standard cubic spline interpolates.

![Figure 4-2](image)

Figure 4-2. Modeling of real and imaginary parts of Bode plot data implicitly models the magnitude and phase data. The real and imaginary plots are better behaved than the corresponding magnitude and phase plots making modeling inherently easier. There are two curves in each plot, the spline interpolate and the true underlying function, however, the approximation is nearly identical to the true function.
model (dashed line) and true underlying function (solid line). The function was sampled using ten points per decade. The corresponding magnitude and phase plots for the AC transfer function are shown in the bottom two plots. Again the true function (solid line) and interpolate calculated from the spline models on the real and imaginary data (dashed line) are plotted. The interpolate and true function are nearly identical and the phase plot is guaranteed to be correctly unwrapped since we may use the spline interpolates to query any frequency in the domain to resolve ambiguities in areas of the phase plot with a very large gradient.

The cubic spline interpolates described in the previous paragraph provide fast and accurate models of discretely sampled SPICE Bode plots which can be queried to obtain the magnitude and phase at any frequency. Sometimes it is necessary to reverse the role of the interpolate to obtain a frequency given a specific magnitude or phase. Since there is no guarantee that the magnitude and phase plots are monotonic, there may be multiple values for magnitude or phase at a given frequency. Since there is a relatively small number of one-dimensional discrete samples comprising the Bode plot, we are at liberty to exhaustively search for every pair of points that has one sample above and one sample below the magnitude or phase of interest. Once all such pairs of points are found, a bisection search is used on each interval to determine very precise frequencies corresponding to the queried magnitude or phase. This method of reverse interpolation must be applied in order to obtain phase margin, gain margin and unity gain frequency.

4.5 Conclusions

The circuit simulator SPICE and its three primary analysis types, DC, AC and TRANS may be used to extract a variety of performance metrics for analog circuits. Operational amplifiers are widely used in industry and therefore implementation of robust test benches for single output op-amps is a good starting point for analog performance extraction and modeling. Ensuring proper DC biasing using negative feedback allows the true open loop op-amp characteristics to be more easily extracted. Discrete transfer function data must be carefully modeled to guarantee accurate extraction of performance metrics without significant measurement error. In addition, reverse spline interpolation is necessary to query frequency from a corresponding gain or phase. In summary, robust test bench circuits and careful data extraction methods must be utilized to facilitate automatic extraction of analog circuit performance metrics. These efforts are circuit class specific and may be used to characterize any topology falling within a specific class. So while the initial design effort of test bench and extraction is largely a manual effort, the ability to reuse such efforts without additional modification justify the design time.
5 Modeling and Synthesis of Analog Circuits using Neural Network Models

An analog system is typically characterized by a set of performance parameters used to succinctly quantify the properties of the circuit. Given a fixed topology, circuit synthesis is the process of determining numerical values for all components in the circuit such that the circuit conforms to a set of performance constraints. Due to the high degree of nonlinearity and interdependence among design variables, manual design of an analog circuit is often reduced to a process of trial and error in which the solution space is searched in an ad hoc manner for a circuit satisfying all constraints. The numerical circuit simulator SPICE is often used as a benchmark of comparison to determine the relative accuracy of alternative schemes for evaluating the performance of analog circuits. However, the computational requirements of running SPICE limit its use when attempting to evaluate a circuit’s performance parameters during circuit synthesis. Stochastic combinatorial optimization methods such as simulated annealing and genetic algorithms require the computation of performance parameters for a large number of circuit sizing alternatives. It is therefore beneficial to reduce the time associated with generating performance estimates.

In this chapter neural network models are used to provide robust and accurate estimates of performance parameters for several CMOS operational amplifiers. Potential problems with using a circuit simulator for data generation are presented along with a detailed analysis of the experimental results. The utility of the neural models is demonstrated in a circuit sizing algorithm using genetic algorithm optimization.

A neural network of sufficient size can estimate functional mappings to an arbitrary precision given a finite discrete set of training data [45], [80]. Hyper-dimensional non-linear functions are readily modeled using neural networks. Neural networks can also easily incorporate knowledge of system behavior. Course functional models can be embedded in the network structure reducing the functional complexity that must be mapped by the network [84], [89]. This often results in a smaller network size and a reduction in training effort. Once trained with a particular functional
mapping, the evaluation time of a neural model is very fast. However, obtaining sufficient data and determining appropriate network size (number of hidden neurons, and number of layers) can be difficult and time consuming. Training algorithms such as Bayesian Regulation [18] can help reduce the interaction needed to determine an appropriate network size, however, full automation of data generation and training is difficult.

The neural network models used in this chapter were trained using data generated directly from SPICE, and correspondingly are often able to provide SPICE level accuracy. Because the evaluation time for the neural models is much less than that required by a full SPICE simulation, the models can be incorporated into a circuit synthesis algorithm used to optimize a fitness function based on performance parameter constraints.

5.1 Overview

Neural networks are typically trained with a discrete set of data points called the training data set. This represents the functionality of the underlying system being modeled. While it is not necessary to know the internal structure of a system in order to model it using neural networks, it is necessary to have examples corresponding to the behavior between the inputs and outputs. This set of example data is the training data set. A second set of discrete data points not present in the training data set is used to validate the neural model of the system.

The circuit simulator SPICE was used to create time-dependant and frequency-dependant data for several circuit topologies instantiating the op-amp to be characterized. The SPICE data from AC and transient simulations was used to calculate several standard performance parameters describing the functionality of operational amplifier circuits (see Section 5.3). By repeating the SPICE simulations for many combinations of op-amp transistor sizes, training and validation data sets were produced and used to create neural network models for the performance parameters.
A general iterative methodology used for neural network model development is shown in Figure 5-1. There is a trade-off between accuracy on the training data set, and good generalization on the validation set. The goal is to be “satisfied” with both accuracy and generalization. If the network has too many free parameters, it can overlearn the training data and generalization becomes poor in the validation set. Training time also plays a role in over-learning and generalization. However, by choosing a network of appropriate size with sufficient, but not too many free parameters, the potential for overtraining can be minimized.

After training, the neural model for one of the op-amp topologies is used to provide estimates of op-amp performance parameters during optimization. Op-amps under various performance constraints are then synthesized using a genetic algorithm. The output of the genetic algorithm is a sized op-amp circuit meeting constraint specifications. The performance parameters are extracted for the optimized circuit and compared to those provided by the neural network models inside the figure.
genetic algorithm. The block diagram in Figure 5-2 depicts the synthesis methodology using the neural network models.

5.2 Data Acquisition

Given a circuit topology and a set of design variables (transistors widths/lengths, capacitor and resistor values, etc.) we wish to find a mapping between the design variables and the associated performance metrics of the circuit. The design variables of the circuit will be referred to as the physical parameters. We introduce a smaller virtual parameter set \( (p_v) \) with cardinality less than or equal to the physical parameter set \( (p_p) \). A mathematical mapping associates the physical and virtual parameters:

\[
\begin{align*}
p^i_v &\in [0,1], \ i = 1, ..., n_v, \\
p^j_p &= f(p^1_v, ..., p^n_v), \ j = 1, ..., n_p
\end{align*}
\]

There are \( n_v \) independent virtual parameters each constrained between zero and one. Each of the \( n_p \) physical design parameters is an arbitrary function of one or more virtual parameters. The performance metrics of the circuit are related to the physical design parameters and the physical parameters are dependant on the virtual parameters. The problem of performance modeling is now reduced to finding a mapping between the proposed virtual parameter space and the corresponding performance metrics.

In order to create an accurate model for each of the performance parameters, the virtual parameter space is sampled in all \( n_v \) dimensions using \( k+1 \) equally spaced values for a user-specified \( k \). This data will be referred to as the \textit{training pattern set} and can be expressed mathematically:

\[
TP_{i,j} = g_j(p^1_v, ..., p^n_v)
\]

\[
p^1_v, ..., p^n_v \in \left\{ 0, \frac{1}{k}, \frac{2}{k}, ..., \frac{k}{k} \right\}
\]

\[
i \in \{ 1, ..., (k + 1)^{n_v} \}
\]

\[
j \in \{ 1, ..., n_p \}
\]

Here the variable \( i \) indexes the list of discrete enumerated points in the virtual parameter space and the variable \( j \) indexes the measured performance parameters of the circuit. The functional operator \( g \) performs a table lookup and is invalid for points not in the sample space.

For any model used to estimate performance parameters at points not in the training data set, we need a way to validate and measure the potential accuracy of these estimates. For this, we
define a validation pattern set. Again the virtual parameter space is sampled in all \( n_v \) dimensions, however, these sample points are maximally distant from sample points in the training data set:

\[
VP_{i,j} = g(p_v^1, ..., p_v^n)
\]

\[
p_v^1, ..., p_v^n \in \left[ \frac{1}{2k}, \frac{3}{2k}, ..., \frac{2k-1}{2k} \right]
\]

\[
i \in \{1, ..., k^{n_v}\}
\]

\[
j \in \{1, ..., n_p\}
\]

Again the variable \( i \) indexes the evaluated points in the virtual parameter space, the variable \( j \) indexes the measured performance parameters of the circuit, and \( g \) performs a table lookup. There are now two sets of data. The larger training data set that used to build a model of the performance parameters, and the smaller validation data set used to check the accuracy and generalization of the proposed model.

A single output two stage CMOS op-amp, depicted in Figure 5-3 will be used as an illustrative example in this paper. The circuit has 8 CMOS transistors, compensation and load capacitance and a reference bias current. For this illustrative example, the widths of all transistors were fixed at 4µm. The load capacitance (not depicted) was fixed to 10pF, and the width of transistor M6 was fixed to 10µm. The widths of the remaining seven transistors and the size of the compensation capacitor were taken as design parameters. It is clear that transistors M1 and M2 should be matched and that transistors M3 and M4 should also be matched in order to evenly split the bias current provided to the differential pair by transistor M5. It is also necessary to equate the DC bias currents through transistors M7 and M8 in order to minimize DC offset voltage at the output. Based on these biasing requirements and design considerations the design space can be reduced to five independent variables. Utilizing the virtual parameter framework described previously, it is necessary to define a mapping between the five virtual parameters and the undetermined component values in the circuit. The widths of M1=M2, M3=M4, M5, and M7, as well as the size of the compensation capacitor were defined as logarithmically scaled versions of the 5 virtual parameters. The width of transistor M8 can be solved for in terms of the other transistor widths in order to balance
its drain current with that of transistor M7. Logarithmic scaling causes equal increments in the virtual parameter space to be scaled relative to the size of the physical parameter value. If the physical parameter’s range is large, logarithmic scaling ensures that similar sampling frequency occurs in the range of small parameter values as well as in the range of large values. Linear scaling fails to appropriately sample the physical space when the parameters have a logarithmic nature (as is often the case for electrical circuit parameters). Devabhaktuni et al. discuss logarithmic parameter scaling in detail [20]. Logarithmic scaling of many electrical and physical circuit parameters is supported in [59]. The following equation was used to logarithmically map the virtual parameters into physical parameters:

\[ p_p = e^{(1-p_v)\ln(r_1) + p_v\ln(r_2)} \quad \text{where} \quad p_p \in [r_1, r_2] \quad (11) \]

The physical parameter \( p_p \) is bounded in a chosen range between \( r_1 \) and \( r_2 \), and recall that the virtual parameter \( p_v \) is bounded in the range of \([0, 1]\). Table 5-1 lists component sizes and ranges for the single-stage op-amp example.

Seven performance parameters were chosen to characterize the behavior of the op-amp, namely: 1) open loop gain, 2) unity gain frequency, 3) phase margin, 4) common mode rejection ratio, 5) power supply rejection ratio relative to \( v_{dd} \) only, 6) rising slew rate and 7) falling slew rate.

In order to create an accurate model for each of the seven performance parameters, the virtual parameter space was sampled for \( k=4 \) in Equation 4 and Equation 8 to produce training and validation sets of size 3125 and 1024.

<table>
<thead>
<tr>
<th>Virtual Parameter Values</th>
<th>Physical Parameter Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p_v^1 )</td>
<td>( M_1 = M_2 ) [20, 80] µm</td>
</tr>
<tr>
<td>( p_v^2 )</td>
<td>( M_3 = M_4 ) [20, 80] µm</td>
</tr>
<tr>
<td>( p_v^3 )</td>
<td>( M_5 ) [20, 80] µm</td>
</tr>
<tr>
<td>( p_v^4 )</td>
<td>( M_7 ) [50, 150] µm</td>
</tr>
<tr>
<td>( p_v^5 )</td>
<td>( C_C ) [2.10] pF</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Virtual Parameter Values</th>
<th>Physical Parameter Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>20.00 28.28 40.00 56.56 80.00</td>
</tr>
<tr>
<td>0.25</td>
<td>20.00 28.28 40.00 56.56 80.00</td>
</tr>
<tr>
<td>0.5</td>
<td>20.00 28.28 40.00 56.56 80.00</td>
</tr>
<tr>
<td>0.75</td>
<td>20.00 28.28 40.00 56.56 80.00</td>
</tr>
<tr>
<td>1.0</td>
<td>20.00 28.28 40.00 56.56 80.00</td>
</tr>
</tbody>
</table>

Table 5-1: Training Pattern Sample Space

NOTE: All transistor lengths fixed at 4µm. Width of \( M_8 = (2M_3M_7) / M_5 \). CL fixed at 10pF. Ibias fixed at 10µA.
samples respectively. The virtual parameter sample grid used for the training data was \{0, 0.25, 0.5, 0.75 and 1\} while the sample grid for the validation data was \{0.125, 0.375, 0.625 and 0.875\}. Figure 5-4 illustrates the nature of the gridded training and validation data sets. It was intended that SPICE be used to determine the performance parameters for all 4149 discrete sample points in the 5-dimensional virtual parameter space. However, for some of the circuit configurations the sample space cannot be correctly mapped into the physical parameter space due to convergence problems inherent in the circuit simulator. These faulty values will be discussed later. For now we will treat the sample space as complete, with the understanding that certain values may be excluded.

### 5.3 Measurement Oracle

In order to extract the seven performance parameters, it was necessary to construct a set of test benches which would provide sufficient data to facilitate the automatic extraction of performance parameters via post-processing of the SPICE output files. Three AC analyses and two transient analyses are sufficient to extract the seven performance parameters of the circuit. The following methodology is applicable to any single output op-amp circuit that can be instantiated into the test benches. Without loss of generality, the positive and negative supply rails for all test bench circuits were set at ±10 volts in the experiments. The circuit simulator Spice3, which is built
on Berkeley SPICE2G.6, was used for all experiments in this paper and will be referred to as simply SPICE.

It is difficult to directly simulate the open loop transfer characteristic of an op-amp due to high open loop gain and large potential DC offset [2]. For this reason closed loop topologies were adopted from which the desired performance characteristics could be extracted. Three of the performance parameters, open loop gain, unity gain frequency and phase margin, can be obtained by a single AC analysis of the op-amp in the well known inverting unity gain configuration depicted in Figure 5-5. Since in this topology, the positive input terminal of the op-amp is grounded, and the negative input terminal is a virtual ground, the common mode input voltage seen by the op-amp is negligible, i.e. no effective common mode signal will be seen at the output of the op-amp and thus common mode gain can be neglected. Frequency was swept between a very low value, and a very high value (1mHz to 10GHz for all experiments) to obtain a Bode plot of the output voltage of the op-amp. The expected closed loop gain (neglecting common mode gain) of the inverting unity gain configuration according to [78] is:

\[ A_F = \frac{A_0}{1 + \beta A_0} \leftrightarrow A_0 = \frac{A_F}{1 - \beta A_F} \]  

(12)

The circuit gain with feedback \( A_F \) is a function of op-amp open loop gain \( A_0 \) and feedback factor \( \beta \). The equation can be reversed to obtain the open loop gain from the other two quantities. Frequency dependence for the gains \( A_F \) and \( A_0 \) is implied while \( \beta \) is frequency independent. For the inverting unity gain configuration (\( \beta = 1 \)) the closed loop gain can be obtained from the Bode plot generated by SPICE at any choice of low frequency, and thus the open loop gain of the op-amp can be calculated from the above formula.

By definition, the unity gain frequency of an op-amp is the frequency at which the open loop gain drops to one. Using the equation for closed loop gain above, we can solve for the closed loop gain when the open loop gain falls to unity:

\[ A_F = \frac{A_0}{1 + \beta A_0} = \frac{1}{1 + (1)(1)} = 0.5 \]  

(13)

Therefore, on the Bode plot generated by SPICE for the closed loop configuration, it is necessary to find the lowest frequency at which the closed loop gain falls to 0.5. This frequency is the unity gain frequency of the op-amp.
The phase margin of an op-amp is the number of degrees remaining before the phase of the output signal reaches 180° when the loop gain $A_0$ falls to unity (if the feedback network is purely resistive). This also occurs for the frequency at which the closed loop gain of the circuit drops to 0.5. At this frequency the phase of the output signal can be determined from the SPICE generated Bode plot and thus the phase margin can also be calculated.

The fourth performance parameter, common mode rejection ratio, can be obtained by performing a second AC analysis with the op-amp in the non-inverting unity gain configuration as shown in Figure 5-6. For the inverting unity gain configuration, the common mode input voltage was approximately zero; however, for the non-inverting configuration the common mode voltage is approximately equal to the input voltage. Previously it was possible to neglect the effect of common mode gain due to a negligible common mode input signal. AC common mode input voltage, common mode gain and common mode rejection ratio are defined in [78] as:

$$\begin{align*}
v_{icm} &= \frac{v_{in+} + v_{in-}}{2} \\
A_{cm} &= \frac{v_o}{v_{icm}} = \frac{2v_o}{v_{in+} + v_{in-}} \\
CMRR &= \frac{|A_0|}{|A_{cm}|} = \frac{A_0(v_{in+} + v_{in-})}{2v_o} = \frac{A_0(v_{in+} + v_{in-})}{2v_o}
\end{align*}$$

(14) \hspace{1cm} (15) \hspace{1cm} (16)

The variables $v_{in+}$ and $v_{in-}$ are the positive and negative input terminals to the op-amp. The value computed for op-amp open loop gain $A_0$ for the inverting configuration is used here in the computation of CMRR. The input voltage of the unity gain buffer $v_{in}$ is set to 1 for the simulation and the output voltage $v_{out}$ can be the output voltage at any low frequency point in the SPICE generated Bode plot. In fact it is only necessary to generate one low frequency data point in the Bode plot for the non-inverting configuration since no high frequency information is used.

Power supply rejection ratio can be obtained by placing an AC source in series with the positive DC power supply, while grounding both inputs of the op-amp. A third AC SPICE analysis was used to obtain the transfer characteristic from the AC source in line with the power supply to the output of the op-amp. PSRR can then be directly obtained from the output voltage of the op-amp sampled at a low frequency in the generated Bode plot.
Slew rate is the most difficult parameter to accurately extract. For this measurement, the op-amp is configured as a comparator by creating a positive feedback loop from the output back to the $v_{in+}$ terminal through a suitable resistor network as shown in Figure 5-7. A piecewise linear triangle wave of low amplitude was presented as input to the circuit, resulting in a hard switching of the output voltage. Two transient simulations were run in SPICE, one to create a low to high transition and another to create a high to low transition. The difficulty in automating the extraction of slew rate was due to the somewhat subjective nature of the calculation. It was observed that the switching delay time and the shape of the waveform could vary significantly from one circuit configuration to the next. In order to preclude erroneous measurements of slew rate an alternative calculation scheme was developed involving error function minimization.

The output of the SPICE transient simulation consists of one time vector (non-uniformly sampled) ranging from a starting time $t_0$ to an ending at time $t_f$ and a corresponding voltage vector representing the switching output of the op-amp. A parameterized model was created that takes as input parameters: slew rate, zero voltage crossing time, positive and negative voltage rail values, and the time vector output from the SPICE simulation. The parameterized model provides as output the expected switching waveform conforming to the specified input parameters. This prototype waveform can then be compared to the actual experimental waveform through the use of an error function. The error function used was the sum squared error between each point in the experimental waveform and each prototypical point generated by the model. Standard numerical optimization was then used to minimize the error between the two waveforms relative to two of the prototype function parameters, namely slew rate and zero voltage crossing time (voltage rail values were fixed at ±5 volts to ensure that the prototype waveform “latched onto” the middle of the rising edge). The behavior and convergence of this algorithm is shown in Figure 5-8. The initial guess for the prototype waveform is taken to be a straight line connecting -5 volts at time $t = 0$ with +5 volts.
at the ending time of the simulation, here 50µs. As the optimization routine minimizes the error between the two functions, the prototype is gradually fitted to the experimental data. The last convergent waveform was changed by hand to have voltage rail values of ±6 volts purely to emphasize this waveform in the graph. The prototype waveform fits the data very accurately in the desired voltage range, and the slew rate of the experimental waveform is taken to be identical to the slew rate of the convergent prototype function.

We now consider the faulty or incomplete data occurring in both the training and validation data sets due to non-convergence of the simulation algorithms, and other unknown numerical stability problems. These faulty points should be removed from the training and validation sets. The gain of the inverting unity gain configuration should be very close to negative 1 for low frequency operation. However, for our illustrative example, it was observed that a portion of the Bode plots had a gain much less than the expected value. Erroneous closed loop gain values reported by SPICE ranged from \(-1\times10^{-3}\) to \(-1\times10^{-16}\) for the inverting amplifier topology. A subset of the failing points was hand selected and verified interactively. For many of these failing points correct results could be obtained by changing the width of a single transistor in the circuit by as little as 1%. For example, when running an AC simulation on the single stage op-amp topology for a particular set of transistor sizes, SPICE returned a closed loop gain of \(-2.973\times10^{-5}\). Transistor M3 was changed from size \(W=150\mu m\) to size \(W=150.5\mu m\). The AC analysis was repeated, this time with SPICE returning a low frequency closed loop gain of 0.999721, or an equivalent open loop gain of 3583V/V. Clearly the gain of the op-amp is not in actuality this sensitive to parametric deviation. This anomaly occurred for a portion of the AC analysis runs on the non-inverting configuration as well. For our example, 30 of the 3125 training patterns had to be discarded resulting in 3095 valid patterns, and 4 of the 1024 validation patterns were removed, leaving 1020 usable patterns. Although some of these issues concerning faulty data might have been alleviated by using another simulator, problems with DC convergence and numerical stability would exist in other simulators as well. Thus the problem of data validity in the training/validation data sets remains and erroneous values in the data sets must be identified and discarded.
5.4 Neural Network Training

A standard two layer feed forward neural network, illustrated in Figure 5-9 was used to model each of the seven op-amp performance parameters. The number of hidden layer neurons used for each network was experimentally varied for each performance parameter in order to obtain good generalization and accuracy on both the training and validation sets. See Table 5-2 for the specifics of network sizes used to model the op-amp performance parameters. The neural network toolbox in Matlab [18] was used to create, train and simulate all neural network models. A hyperbolic tangent sigmoid function was used as the activation function for all hidden layer neurons and a linear function was used for all output layer neurons. All networks are “fully connected”, that is, all input vectors are connected to all hidden layer neuron inputs, and all hidden layer neuron outputs are connected to all linear output neuron inputs. Each network in this case has only one output neuron, since a unique network was used to model each op-amp performance parameter. Therefore, each network performs a functional mapping from the five dimensional virtual parameter space to the single dimensional performance parameter space. Together, all seven networks can be viewed as a 5 dimensional to 7 dimensional mapping of real numbers.

The training pattern set $TP_{ij}$ (pruned down to 3095 patterns) was used to train the networks using the Levenberg-Marquardt [18] back propagation algorithm. Each network was trained for about 100 epochs. The validation pattern set $VP_{ij}$ (1020 patterns) was used to check the interpolation accuracy of the networks.

### Table 5-2

<table>
<thead>
<tr>
<th>Parameter</th>
<th># Hidden Neurons</th>
<th>Exec. Time ($\mu$s)</th>
<th>Range</th>
<th>Mean Error</th>
<th>Standard Dev.</th>
<th>Maximum Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open Loop Gain V/V (%)</td>
<td>8</td>
<td>5.4</td>
<td>642.80 - 100137</td>
<td>0.1309</td>
<td>0.0895</td>
<td>0.1196</td>
</tr>
<tr>
<td>Unity Gain Freq. MHz (%)</td>
<td>14</td>
<td>8.6</td>
<td>0.6523 - 13.722</td>
<td>0.2434</td>
<td>0.1713</td>
<td>0.2359</td>
</tr>
<tr>
<td>Phase Margin o</td>
<td>12</td>
<td>7.6</td>
<td>-40.901 - 120.66</td>
<td>-0.0026</td>
<td>0.0009</td>
<td>0.2934</td>
</tr>
<tr>
<td>CMRR dB</td>
<td>12</td>
<td>7.6</td>
<td>56.157 - 100.01</td>
<td>-0.0006</td>
<td>0.0007</td>
<td>0.3954</td>
</tr>
<tr>
<td>PSRR dB</td>
<td>10</td>
<td>6.5</td>
<td>113.00 - 174.60</td>
<td>0.0010</td>
<td>0.0040</td>
<td>0.1577</td>
</tr>
<tr>
<td>Rising Slew Rate V/µs (%)</td>
<td>13</td>
<td>8.1</td>
<td>1.9956 - 12.005</td>
<td>0.4142</td>
<td>0.5236</td>
<td>0.4144</td>
</tr>
<tr>
<td>Falling Slew Rate V/µs (%)</td>
<td>13</td>
<td>8.1</td>
<td>2.0483 - 50.025</td>
<td>0.2439</td>
<td>0.2127</td>
<td>0.2130</td>
</tr>
</tbody>
</table>

51
tion and generalization capabilities of the networks for points not present in the training data. Typically, training data presented to a neural network must be scaled so that all values have a similar range. A discussion on the importance and methodology of data scaling can be found in [20]. Since the input values to the networks are the discretely sampled values in the virtual parameter space and are bounded to an interval of [0,1], no data scaling was required for input vectors to the network. However, to ensure good performance, all network outputs were scaled to an interval of [0,1] either linearly or logarithmically. Phase margin, common mode rejection ratio and power supply rejection ratio were linearly scaled. The following formula, taken from [20] was used for linear scaling:

\[
\hat{x} = \tilde{x}_{\text{min}} + \frac{x - \tilde{x}_{\text{min}}}{\tilde{x}_{\text{max}} - \tilde{x}_{\text{min}}} (\tilde{x}_{\text{max}} - \hat{x}_{\text{min}}) 
\]

\[
x \in [\tilde{x}_{\text{min}}, \tilde{x}_{\text{max}}], \hat{x} \in [\hat{x}_{\text{min}}, \hat{x}_{\text{max}}] 
\]

The original data value \(x\) is linearly scaled to a new value \(\hat{x}\) in the chosen interval \([\tilde{x}_{\text{min}}, \tilde{x}_{\text{max}}]\). Both ranges of \(x\) and \(\hat{x}\) must be known prior to scaling. The remaining performance parameters, open loop gain, unity gain frequency and slew rate (rising/falling) were logarithmically scaled before presentation to the neural networks using:

\[
\hat{x} = \tilde{x}_{\text{min}} + \frac{\ln \left( \frac{x}{\tilde{x}_{\text{min}}} \right)}{\ln \left( \frac{\tilde{x}_{\text{max}}}{\tilde{x}_{\text{min}}} \right)} (\tilde{x}_{\text{max}} - \hat{x}_{\text{min}}) 
\]

Similar to linear scaling, the ranges of both variables must be known prior to logarithmic scaling.

Using SPICE, the generation of the 3125 points in the training data took 1 hour and 47 minutes on a Sun SunBlade100 machine with 256MB of RAM. This averages out to about 2 seconds of execution time per sample point in the input space (disregarding the time needed to post process the SPICE results). Execution times for one evaluation of each neural network estimator executed on the same computer are given in Table 5-2. Generating all seven performance estimates using the neural models takes around 51.9us, a speedup factor of about 40,000 when compared to directly using SPICE.

### 5.5 Circuit Synthesis Using a Genetic Algorithm

The neural network based macro models can be used in any combinatorial search algorithm for constraint-directed circuit sizing. Naturally, using the neural network model makes this process significantly faster than using a circuit simulator. To illustrate the use of the neural network models for circuit synthesis, our models were used inside of a genetic algorithm [85] to give quick and accu-
rate estimates of performance parameters to aid in the automatic sizing of transistors in the op-amp circuit. Although any other combinatorial optimization algorithm such as simulated annealing or tabu search could have been used, we chose the genetic algorithm since it is known to be robust in the presence of multiple constraints and is insensitive to the nature of the cost function. Genetic algorithms (GA) have been successfully used for effective analog synthesis by other researchers [37], [55], [55], [21], [54].

For our genetic algorithm, performance constraints were formulated using the fuzzy membership approach proposed in [26]. Each inequality constraint on the performance parameter set was represented by a membership function. The membership function returns 1 if the constraint is satisfied and returns a value less than 1 but greater than 0 if the constraint is violated. A fuzziness parameter controls how quickly the membership function falls to 0 for a violated constraint. Figure 5-10 illustrates the effect of the fuzziness parameter on the constraint membership function. The membership function depicted represents an inequality constraint that is satisfied when the normalized op-amp performance parameter is greater than 0.8. For fuzziness parameters close to 0, i.e. 0.1 and 0.01 in the illustration, the membership function falls to zero very fast for violation of the constraint. Increasing the fuzziness parameter allows a relaxation in constraint satisfaction. For fuzzy constraints, being almost satisfied is nearly as good as being completely satisfied. This is useful when attempting to satisfy difficult or unsatisfiable constraints. Instead of failing to meet one or more constraints while meeting others, it is possible to fail over all the constraints, but fail by about the same amount.

All membership functions representing inequality constraints were combined into a weighted sum where the weight for each function can be adjusted based on the relative importance between constraints. The GA was then used to maximize this weighted sum, and thus attempts to satisfy all performance parameter constraints. A normalized performance parameter can also be minimized or maximized by directly including it in the weighted sum of membership functions. For the case of circuit area minimization, no neural network model was needed since the actual transis-
tor sizes of the circuit were directly known (area was estimated by the sum of all transistor areas). In order to minimize area using the membership function scheme, it was necessary to create a function, which when maximized, is representative of area being minimized. Area could then be minimized by maximizing the representative function while adhering to the member function scheme. The following function was used in the GA to minimize circuit area:

\[ \tilde{A} = 1 - \frac{A - A_{\text{min}}}{A_{\text{max}} - A_{\text{min}}} \quad (20) \]

The circuit area \( A \) is bounded in a known range \([A_{\text{min}}, A_{\text{max}}]\). The representative function for area \( \tilde{A} \) reaches a maximum of 1 when circuit area is minimized and goes to 0 as circuit area is maximized.

The MIT GAlib genetic algorithm package [85] was used to implement a GAIncrementalGA scheme. In this specific implementation of the genetic algorithm, at each iterative step, only one new child genome is created by crossing over two parent genomes and then mutating the child genome with some probability. The child genome is introduced back into the current population by replacing a random genome currently in the population. Pseudo-code for the genetic search algorithm is illustrated in Figure 5-11.

A population size of 100 genomes was used in the GA optimization. Each genome was constructed by concatenating binary coded versions of the virtual parameter values. For each of the 5 virtual parameter values, 31 binary bits were used for encoding, resulting in a binary genome of length 155. For this experiment, the probability of mutation was set to 0.05 and the crossover probability as set to 0.9. The GA was run for 10,000 iterations on each constraint configuration and

![Figure 5-11](image_url). Pseudo-code describing the behavior of the GA used to perform circuit synthesis.
took only a few seconds of execution time using the neural network models. An example of the convergence properties of the GA out to 4,000 iterations is shown in Figure 5-12.

The GA was used to successfully size transistors in the op-amp circuit for various constraint configurations. The results of 8 different constraint sets are presented in Table 5-3 through Table 5-10. For each experiment, SPICE was used to validate the GA optimized circuit parameters. The neural network estimates of the parameters are very close to the actual performance parameter values determined using SPICE. A “nominal” set of satisfiable constraints were devised as well as a “difficult” set of unsatisfiable constraints. Table 5-3 through Table 5-7 all contain results for satisfiable constraints while Table 5-8 through Table 5-10 contain results using unsatisfiable constraints. For most of the experiments the inequality constraints posed were basically non-fuzzy, with the fuzziness factors set to 0.01. An interesting observation comes from comparing the results in Tables 5-8, 5-9 and 5-10, i.e. the difficult constraint configurations. When working with non-fuzzy constraints (Table 5-8) the GA “chose” to meet the constraints for unity gain frequency, rising slew-rate, and falling slew rate while failing to meet all other constraints. A similar situation occurs in Table 5-10. The GA meets the constraints for open loop gain, phase margin and common mode rejection ratio, but fails to meet the remaining constraints by a large margin. It is unknown which constraints will be met after synthesis for the difficult constraint set. When the constraints are fuzzy as in Table 5-9, the GA still fails to meet constraints, but fails more graciously. The constraints fail by about the same margin rather than a few satisfied constraints dominating the unsatisfied ones (the normalized neural outputs are useful for noticing this fact). Details for relative weights and fuzzy factors used in each constraint setup are given in Table 5-11.
Note: "Difficult Constraint" implies the constraint is set at 90% of the highest possible value.

<table>
<thead>
<tr>
<th>Parameter Constraint Spice</th>
<th>Neural Network Scalled Normalized</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open Loop Gain V/V &gt; 20000</td>
<td>40704</td>
</tr>
<tr>
<td>Unity Gain Freq. MHz &gt; 5e6</td>
<td>5070</td>
</tr>
<tr>
<td>Phase Margin ° &gt; 70</td>
<td>70.922</td>
</tr>
<tr>
<td>CMRR dB &gt; 80</td>
<td>92.102</td>
</tr>
<tr>
<td>PSRR dB &gt; 150</td>
<td>150.45</td>
</tr>
<tr>
<td>Rising Slew Rate V/µs &gt; 6.0e6</td>
<td>8.708</td>
</tr>
<tr>
<td>Falling Slew Rate V/µs &gt; 6.0e6</td>
<td>10.454</td>
</tr>
<tr>
<td>Area µm²</td>
<td>minimize 1552</td>
</tr>
</tbody>
</table>

Area µm² minimize 1505

<table>
<thead>
<tr>
<th>Parameter Constraint Spice</th>
<th>Neural Network Scalled Normalized</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open Loop Gain V/V &gt; 20000</td>
<td>62847</td>
</tr>
<tr>
<td>Unity Gain Freq. MHz &gt; 5e6</td>
<td>5049</td>
</tr>
<tr>
<td>Phase Margin ° &gt; 70</td>
<td>77.924</td>
</tr>
<tr>
<td>CMRR dB &gt; 80</td>
<td>95.965</td>
</tr>
<tr>
<td>PSRR dB &gt; 150</td>
<td>152.77</td>
</tr>
<tr>
<td>Rising Slew Rate V/µs &gt; 6.0e6</td>
<td>8.959</td>
</tr>
<tr>
<td>Falling Slew Rate V/µs &gt; 6.0e6</td>
<td>9.984</td>
</tr>
<tr>
<td>Area µm²</td>
<td>minimize 1701</td>
</tr>
</tbody>
</table>

Area µm² minimize 1501

<table>
<thead>
<tr>
<th>Parameter Constraint Spice</th>
<th>Neural Network Scalled Normalized</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open Loop Gain V/V &gt; 20000</td>
<td>38854</td>
</tr>
<tr>
<td>Unity Gain Freq. MHz &gt; 5e6</td>
<td>5272</td>
</tr>
<tr>
<td>Phase Margin ° &gt; 70</td>
<td>72.433</td>
</tr>
<tr>
<td>CMRR dB &gt; 80</td>
<td>91.788</td>
</tr>
<tr>
<td>PSRR dB &gt; 150</td>
<td>150.17</td>
</tr>
<tr>
<td>Rising Slew Rate V/µs &gt; 6.0e6</td>
<td>8.416</td>
</tr>
<tr>
<td>Falling Slew Rate V/µs &gt; 6.0e6</td>
<td>9.791</td>
</tr>
<tr>
<td>Area µm²</td>
<td>minimize 1505</td>
</tr>
</tbody>
</table>

Area µm² minimize 1501

<table>
<thead>
<tr>
<th>Parameter Constraint Spice</th>
<th>Neural Network Scalled Normalized</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open Loop Gain V/V &gt; 60443</td>
<td>50715</td>
</tr>
<tr>
<td>Unity Gain Freq. MHz &gt; 10.118</td>
<td>727</td>
</tr>
<tr>
<td>Phase Margin ° &gt; 104.50</td>
<td>62.741</td>
</tr>
<tr>
<td>CMRR dB &gt; 95.626</td>
<td>94.102</td>
</tr>
<tr>
<td>PSRR dB &gt; 168.44</td>
<td>149.90</td>
</tr>
<tr>
<td>Rising Slew Rate V/µs &gt; 10.033</td>
<td>10.950</td>
</tr>
<tr>
<td>Falling Slew Rate V/µs &gt; 36.489</td>
<td>13.433</td>
</tr>
<tr>
<td>Area µm²</td>
<td>minimize 1584</td>
</tr>
</tbody>
</table>

Area µm² minimize 1584

<table>
<thead>
<tr>
<th>Parameter Constraint Spice</th>
<th>Neural Network Scalled Normalized</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open Loop Gain V/V &gt; 60443</td>
<td>50599</td>
</tr>
<tr>
<td>Unity Gain Freq. MHz &gt; 10.118</td>
<td>727</td>
</tr>
<tr>
<td>Phase Margin ° &gt; 104.50</td>
<td>62.741</td>
</tr>
<tr>
<td>CMRR dB &gt; 95.626</td>
<td>94.102</td>
</tr>
<tr>
<td>PSRR dB &gt; 168.44</td>
<td>149.90</td>
</tr>
<tr>
<td>Rising Slew Rate V/µs &gt; 10.033</td>
<td>10.950</td>
</tr>
<tr>
<td>Falling Slew Rate V/µs &gt; 36.489</td>
<td>13.433</td>
</tr>
<tr>
<td>Area µm²</td>
<td>minimize 1584</td>
</tr>
</tbody>
</table>

Area µm² minimize 1584

Table 5-10

<table>
<thead>
<tr>
<th>Parameter Constraint Spice</th>
<th>Neural Network Scalled Normalized</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open Loop Gain V/V &gt; 5e6</td>
<td>5070</td>
</tr>
<tr>
<td>Unity Gain Freq. MHz &gt; 5e6</td>
<td>5049</td>
</tr>
<tr>
<td>Phase Margin ° &gt; 70</td>
<td>77.924</td>
</tr>
<tr>
<td>CMRR dB &gt; 80</td>
<td>95.965</td>
</tr>
<tr>
<td>PSRR dB &gt; 150</td>
<td>152.77</td>
</tr>
<tr>
<td>Rising Slew Rate V/µs &gt; 6.0e6</td>
<td>8.959</td>
</tr>
<tr>
<td>Falling Slew Rate V/µs &gt; 6.0e6</td>
<td>9.984</td>
</tr>
<tr>
<td>Area µm²</td>
<td>minimize 1701</td>
</tr>
</tbody>
</table>

Area µm² minimize 1501

<table>
<thead>
<tr>
<th>Parameter Constraint Spice</th>
<th>Neural Network Scalled Normalized</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open Loop Gain V/V &gt; 60443</td>
<td>50715</td>
</tr>
<tr>
<td>Unity Gain Freq. MHz &gt; 10.118</td>
<td>727</td>
</tr>
<tr>
<td>Phase Margin ° &gt; 104.50</td>
<td>62.741</td>
</tr>
<tr>
<td>CMRR dB &gt; 95.626</td>
<td>94.102</td>
</tr>
<tr>
<td>PSRR dB &gt; 168.44</td>
<td>149.90</td>
</tr>
<tr>
<td>Rising Slew Rate V/µs &gt; 10.033</td>
<td>10.950</td>
</tr>
<tr>
<td>Falling Slew Rate V/µs &gt; 36.489</td>
<td>13.433</td>
</tr>
<tr>
<td>Area µm²</td>
<td>minimize 1584</td>
</tr>
</tbody>
</table>

Area µm² minimize 1584

Table 5-11

<table>
<thead>
<tr>
<th>Parameter Constraint Spice</th>
<th>Neural Network Scalled Normalized</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open Loop Gain V/V &gt; 60443</td>
<td>50715</td>
</tr>
<tr>
<td>Unity Gain Freq. MHz &gt; 10.118</td>
<td>727</td>
</tr>
<tr>
<td>Phase Margin ° &gt; 104.50</td>
<td>62.741</td>
</tr>
<tr>
<td>CMRR dB &gt; 95.626</td>
<td>94.102</td>
</tr>
<tr>
<td>PSRR dB &gt; 168.44</td>
<td>149.90</td>
</tr>
<tr>
<td>Rising Slew Rate V/µs &gt; 10.033</td>
<td>10.950</td>
</tr>
<tr>
<td>Falling Slew Rate V/µs &gt; 36.489</td>
<td>13.433</td>
</tr>
<tr>
<td>Area µm²</td>
<td>minimize 1584</td>
</tr>
</tbody>
</table>

Area µm² minimize 1584

Note: "Difficult Constraint" implies the constraint is set at 90% of the highest possible value.
5.6 Additional Experimental Results for Neural Network Performance Models

Using the same methodology as described for the simple two stage op-amp in Figure 5-3, neural network performance models were generated for two additional analog cells, a cascode op-amp and a differential output op-amp. The cascode op-amp using 16 MOS transistors was borrowed from [2] and is shown in Figure 5-13. The differential output op-amp in Figure 5-14 was borrowed from [13] and contains 33 MOS transistors. For both circuits, DC bias and matched-transistor requirements were analyzed. This leads to a small number of free design variables with other design variables having absolutely or relatively fixed sizes. The number of free parameters for both circuits was reduced to 5 virtual parameters. The device sizing requirements for these two circuits are summarized in Table 5-12 and Table 5-13. Note that such reduction in the number of free variables is quite common in analog circuit design due to DC biasing and matching requirements.

Three sets of training and validation data were generated for each circuit by varying the sampling grid density in the virtual parameter space. The cascode op-amp was sampled for \( k = \{3, 4, 5\} \) and the differential op-amp for \( k = \{4, 5, 6\} \). Neural network models were then trained to map the virtual parameters of each circuit into the corresponding performance parameter metrics. The neural network model accuracies are listed in Tables 5-14 through 5-20. For each circuit all training and validation patterns were compiled into a *global test set* labeled \( GP_{i,j} \). The global test set is the union of all training and validation patterns for a given circuit and represents all the acquired knowledge harvested about the underlying relationships governing the 7 performance parameters. It is expected that the neural network models will be least accurate over the global test set. For low sampling densities of the virtual parameter space we see that while the neural networks perform well on
Figure 5-14. The differential operational amplifier with biasing circuitry.
the validation set, they do not necessarily perform well on the global test set. This is because the underlying function is not well represented by such a low number of sample points. As the density of the sample grid is increased more features of the function emerge in the training data and the overall network accuracy increases.

5.7 Conclusions

Neural network modeling of op-amp performance parameters proved to be an effective methodology for fast and accurate performance estimation. The neural models capture non-linear behavior and require no knowledge or equations describing the internal structure of the op-amp. The method is also generic and extensible. The same set of test bench circuits used to extract performance parameters for the op-amps in this paper could be used to extract parameters for other op-amp topologies. In order to create models for circuits with a fundamentally different functionality than an op-amp, a new set of SPICE test bench circuits need to be constructed for the extraction of performance parameters.

Generating layout-aware models is one of the challenging tasks of analog macro modeling. Layout-aware models are based on training data generated from simulations of circuit models extracted from layouts. We present a small example to illustrate the robustness of neural network based macro models in capturing layout-dependent parasitic effects. Figure 5-15 shows the layout of a CMOS spiral inductor. The inductor design parameters are the number of turns \( t \), wire spacing \( s \), and wire width \( w \). The inner diameter is assumed to be fixed. The design parameters of the inductor were swept over fixed ranges and the low-frequency lumped inductance was extracted using VPEC [48], [70] and FASTHENRY [71] to produce training and validation sets of data. The inductors are implemented in the MIT Lincoln Laboratory’s 0.18 micron FDSOI CMOS technology. Table 5-21 summarizes the accuracy of the neural network model for the inductance with layout parasitics.
A significant drawback to the method described is the large number of sample points required to accurately map the behavior of a circuit. Many sample points are needed to catch sharp peaks and valleys of the functional mappings. However, for fully enumerated data tables in hyper-dimensional spaces, the number of sample points in both the test and validation data sets grows exponentially w.r.t. the number of free parameters and polynomially w.r.t. the density of the sample grid. A non-uniform sampling of the input parameter space could reduce the number of sample points required if highly non-linear regions of behavior could be identified and sampled at a higher frequency. The use of better sampling methods to reduce the number of sample points in is discussed in Chapter 6.

Although a large number of simulations are required to capture the behavior of the performance characteristics of a circuit, the effort is justifiable when considering the reusability of the models. Generating model data for the training and validation sets for the simple op-amp required about 4000 sample points, each requiring approximately 2 seconds of SPICE execution time. The GA, using the neural network models, was executed for 10,000 steps on each of the 8 performance parameter constraint configurations. Performing the same experiment using SPICE to directly supply performance parameters would require about 8x10,000 2-second evaluations, i.e. about 44 hours. Whereas, the execution time using the neural networks was less than 10 seconds for each of the constraint configurations, or a little over 1 minute to synthesize all configurations. The neural network

---

**Table 5-14**

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Hidden Neurons</th>
<th>k</th>
<th>Dynamic Range of Training Data (V/V)</th>
<th>Mean Error (%)</th>
<th>Standard Dev. (%)</th>
<th>Maximum Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cascode Opamp</td>
<td>10</td>
<td>2</td>
<td>4.403 2682.7</td>
<td>0.395 0.64 0.774</td>
<td>0.264 0.379 1.133</td>
<td>1.101 1.209 9.919</td>
</tr>
<tr>
<td>(no. = 5)</td>
<td>11</td>
<td>3</td>
<td>4.403 2662.7</td>
<td>0.372 0.28 0.362</td>
<td>0.277 0.212 0.299</td>
<td>1.392 1.007 2.532</td>
</tr>
<tr>
<td>Differential Opamp (no. = 5)</td>
<td>10</td>
<td>3</td>
<td>1.51e4 1.04e4</td>
<td>0.925 6.87 3.95</td>
<td>0.76 15.41 7.80</td>
<td>4.72 55.1 69.2</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>4</td>
<td>1.51e4 1.02e4</td>
<td>3.076 7.92 4.69</td>
<td>2.27 12.81 7.21</td>
<td>12.22 51.7 63.2</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>5</td>
<td>1.51e4 1.04e4</td>
<td>2.019 4.74 2.97</td>
<td>1.86 9.67 5.70</td>
<td>10.78 47.2 47.3</td>
</tr>
</tbody>
</table>

**Table 5-15**

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Hidden Neurons</th>
<th>k</th>
<th>Dynamic Range of Training Data (MHz)</th>
<th>Mean Error (%)</th>
<th>Standard Dev. (%)</th>
<th>Maximum Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cascode Opamp</td>
<td>10</td>
<td>2</td>
<td>2.43e4 3.64e6</td>
<td>0.271 0.344 0.346</td>
<td>0.231 0.245 0.297</td>
<td>1.442 1.035 2.563</td>
</tr>
<tr>
<td>(no. = 5)</td>
<td>10</td>
<td>3</td>
<td>2.43e4 3.64e6</td>
<td>0.258 0.256 0.239</td>
<td>0.224 0.182 0.207</td>
<td>1.810 1.120 2.402</td>
</tr>
<tr>
<td>Differential Opamp (no. = 5)</td>
<td>10</td>
<td>3</td>
<td>2.22e5 3.44e6</td>
<td>0.25 0.18 0.22 0.19 0.14 0.18</td>
<td>1.08 0.61 1.08</td>
<td></td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>4</td>
<td>2.22e5 3.44e6</td>
<td>0.42 0.35 0.39</td>
<td>0.36 0.27 0.33</td>
<td>2.32 1.39 2.62</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>5</td>
<td>2.22e5 3.44e6</td>
<td>0.32 0.17 0.31</td>
<td>0.27 0.17 0.26</td>
<td>1.78 1.04 2.03</td>
</tr>
</tbody>
</table>

**Figure 5-15.** Integrated circuit layout of a four turn spiral inductor. The inductor is parametrized by the number of turns, the distance between wires, and the wire width.
### Table 5-16
Neural Network Accuracy

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Hidden Neurons</th>
<th>K</th>
<th>Dynamic Range of Training Data (°)</th>
<th>Mean Error (Degrees)</th>
<th>Standard Dev. (Degrees)</th>
<th>Maximum Error (Degrees)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Min</td>
<td>Max</td>
<td>TPI</td>
<td>VP</td>
</tr>
<tr>
<td>Cascode</td>
<td>10</td>
<td>2</td>
<td>-13.35</td>
<td>123.4</td>
<td>-5.32e-4</td>
<td>-0.018</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>3</td>
<td>-17.95</td>
<td>123.4</td>
<td>9.58e-3</td>
<td>0.031</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>4</td>
<td>-18.78</td>
<td>123.4</td>
<td>-3.02e-3</td>
<td>0.014</td>
</tr>
<tr>
<td>Differential</td>
<td>10</td>
<td>3</td>
<td>90.2</td>
<td>105.5</td>
<td>2.3e-4</td>
<td>-0.013</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>4</td>
<td>90.2</td>
<td>105.5</td>
<td>5.1e-3</td>
<td>4.45e-3</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>5</td>
<td>90.2</td>
<td>105.5</td>
<td>8.23e-5</td>
<td>-5.52e-3</td>
</tr>
</tbody>
</table>

### Table 5-17
Neural Network Accuracy

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Hidden Neurons</th>
<th>K</th>
<th>Dynamic Range of Training Data (dB)</th>
<th>Mean Error (dB)</th>
<th>Standard Dev. (dB)</th>
<th>Maximum Error (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Min</td>
<td>Max</td>
<td>TPI</td>
<td>VP</td>
</tr>
<tr>
<td>Cascode</td>
<td>8</td>
<td>2</td>
<td>11.53</td>
<td>68.50</td>
<td>1.56e-3</td>
<td>0.043</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>3</td>
<td>11.53</td>
<td>68.50</td>
<td>1.04e-3</td>
<td>4.92e-4</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>4</td>
<td>72.42</td>
<td>103.0</td>
<td>-4.90e-3</td>
<td>0.431</td>
</tr>
<tr>
<td>Differential</td>
<td>8</td>
<td>3</td>
<td>66.9</td>
<td>68.4</td>
<td>4.47e-6</td>
<td>-0.014</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>4</td>
<td>66.9</td>
<td>68.4</td>
<td>1.56e-3</td>
<td>-4.15e-3</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>5</td>
<td>66.9</td>
<td>68.4</td>
<td>1.91e-4</td>
<td>-1.69e-3</td>
</tr>
</tbody>
</table>

### Table 5-18
Neural Network Accuracy

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Hidden Neurons</th>
<th>K</th>
<th>Dynamic Range of Training Data (V/µs)</th>
<th>Mean Error (%)</th>
<th>Standard Dev. (%)</th>
<th>Maximum Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Min</td>
<td>Max</td>
<td>TPI</td>
<td>VP</td>
</tr>
<tr>
<td>Cascode</td>
<td>10</td>
<td>2</td>
<td>0.0577</td>
<td>201.79</td>
<td>1.15</td>
<td>1.772</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>3</td>
<td>0.0577</td>
<td>234.86</td>
<td>1.243</td>
<td>0.638</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>4</td>
<td>0.0577</td>
<td>201.79</td>
<td>0.627</td>
<td>0.365</td>
</tr>
<tr>
<td>Differential</td>
<td>8</td>
<td>3</td>
<td>66.9</td>
<td>68.4</td>
<td>6.8302</td>
<td>0.518</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>3</td>
<td>0.0667</td>
<td>150.32</td>
<td>0.788</td>
<td>0.999</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>4</td>
<td>0.0667</td>
<td>195.82</td>
<td>0.603</td>
<td>0.627</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>5</td>
<td>0.1510</td>
<td>6.8302</td>
<td>0.556</td>
<td>0.545</td>
</tr>
</tbody>
</table>
models provide a great deal of time savings in situations where a fixed topology must be reused and re-synthesized many times.

The neural network models are also robust [27]. Numerical instability in SPICE [50] and other circuit simulators can prohibit the acquisition of performance parameters for some of the circuit configurations in the sample space. The neural network models can give estimates of values that the simulator failed to provide. However, there are never any guarantees of absolute accuracy when approximating unknown functions. Use of a sufficiently large validation data set helps insure accuracy for most of the points in the input space.

We will now transition from performance models based on static sampling methods and introduce two novel adaptive sampling algorithms. It is not clear how an adaptive sampling algorithm might be implemented using neural network models since neural networks do not provide an exact interpolation of the data. Each time a new network is trained, or an old network is retrained, the shape of the function described by the neural model changes, complicating the issue of where to place additional sample points. The adaptive sampling algorithms presented require an exact and repeatable interpolation of the discrete data and therefore are not directly compatible with neural network theory.

<table>
<thead>
<tr>
<th>Performance Summary of Neural Network Models for Layout Aware Spiral Inductor</th>
<th>$TP_{ij}$</th>
<th>$VP_{ij}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic Range of Training Data (Henrys)</td>
<td>Max</td>
<td>3.044e-8</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>1.508e-11</td>
</tr>
<tr>
<td>Mean Error %</td>
<td></td>
<td>0.640</td>
</tr>
<tr>
<td>Standard Dev. %</td>
<td></td>
<td>0.551</td>
</tr>
<tr>
<td>Maximum Error %</td>
<td></td>
<td>5.808</td>
</tr>
</tbody>
</table>

Reported results for 1728 training data points and 840 validation points.
Adaptive sampling is the process of judiciously sampling new points while considering the history of previous sample points such that model quality is maximized. In this chapter we present two novel adaptive sampling algorithms useful in guiding simulation and data acquisition for analog circuit macro models. The first method uses piecewise linear models and works by detecting and sampling nonlinear regions in the function space. The second adaptive sampling methodology presented is based on nonlinear Duchon pseudo-cubic splines. Both sampling methods are used to guide the acquisition of various performance parameters for analog circuit topologies and are shown to reduce maximum modeling error by strategic placement of costly sample points when compared to gridded and random sampling strategies. Gridded sampling is impractical for high-dimensional functions due to the exponential increase in sample points with respect to dimension and grid density. In addition gridded sampling can only be applied to simple box-constrained function domains. Unstructured methods such as pseudo-random or quasi-random samplers are compatible with domains of any shape, however they are still ignorant of the shape of the function being modeled and place sample points irrespective of model quality.

6.1 Adaptive Sampling with Piecewise Linear Models

Intuitively, an adaptive sampling algorithm should place the highest density of sample points in regions where the function is the most nonlinear, while only sparsely sampling linear or constant regions. A novel adaptive sampling algorithm is presented that is useful for efficiently exploring multidimensional functions; the method is independent of domain dimension and works by detecting and sampling nonlinear regions in the function space. The adaptive sampler is coupled to piecewise linear models due to the nature of the non-linearity estimate used to guide placement of additional sample points. The second adaptive sampling algorithm is also inherently couple to a respective modeling methodology. A “good” set of sample points for one type of model may not be “good” for another, and therefore the sampler and modeling methodology are strictly coupled. A Delaunay linear interpolator is used to model the unstructured sample points and validate the effec-
tiveness of the adaptive sampling methodology in comparison to gridded and random sampling strategies. The adaptive sampler strives to reduce maximum error in the model and outperforms the other two sampling methodologies with respect to maximum error and standard deviation. As an illustrative example six performance metrics are modeled for a CMOS op-amp topology using gridded, random, and adaptive sampling methods. Additionally, the conversion gain of an RF mixer circuit is also presented to the adaptive sampler and model accuracy is compared with the other sampling methodologies.

6.1.1 Algorithm

We assume the real valued function to be explored, \( f \), describes a hyper-surface of dimension \( d+1 \) with \( d \) real independent variables. The adaptive sampling algorithm begins with an initial set of sample points \( X_0 \) for which the function \( f \) is evaluated.

\[
x \in \mathbb{R}^d
\]

\[
X_0 = \{x_1, x_2, \ldots x_n\}
\]

\[
Y_0 = \{f(x_1), f(x_2), \ldots f(x_n)\}
\]

Logical choices for points in \( X_0 \) are corners in the hyper-cube function domain, a star configuration around the geometric center, or simple uniform random samples.

Pseudo code for the adaptive sampling algorithm is shown in Figure 6-1 and is explained here in detail. We wish to detect regions of non-linearity and sample those regions appropriately using only the collected information contained in the sets \( X_0 \) and \( Y_0 \).
The function domain $\Omega$ is a hyper-cube of dimension $d$. During each iteration of the adaptive sampler, $n_i$ random points in $\Omega$ are considered for sampling. For each candidate point $x$ in $X_i$, we find the $k$-nearest neighbor points $X_k$ in $\Omega$ with respect to the Euclidean norm (subroutine \texttt{KNN}). The $k$-nearest neighbors problem is well studied with many algorithmic variations of the solution [64]. For simplicity, we used an exhaustive nearest neighbor search which has complexity $O(dn_i)$, in $d$-dimensional space with $n$ points. A hyper-plane $P$ is fitted to the nearest neighbor points (including "height" information $Y_k$) in the least squares sense [19]. With $k=2(d+1)$ we have twice the number of points necessary to precisely define a hyper-plane. (Alternative values greater than $2(d+1)$ could also be used for $k$. The dependence of algorithm behavior on $k$ needs to be further explored.) For non-linear regions of the function, the nearest neighbor points will not precisely intersect the least squares plane. This is the basis for the nonlinearity estimate $\xi$. The sum of the distances between the nearest neighbor points and the least squares plane is the estimate of non-linearity of function $f$ at point $x$. We store the non-linearity estimates in a list $\xi_i$ for each candidate point in $X_i$ and choose to sample the $n_s$ points with the largest $\xi$. The newly sampled points are assimilated into the lists $X_0$, $Y_0$, and iteration continues until a stopping criterion is met, such as a maximum number of samples, time limit, etc.

There is computational overhead in considering points for sampling, so one must use reasonable values for the $n_i$ and $n_s$ parameters based on the evaluation time of the underlying function. Long evaluation times for $f$ will justify large numbers of considered points, and small numbers of sampled points. We currently have no mathematical framework for determining $n_i$ and $n_s$. Nominal values of $n_i = 500$ and $n_s = 20$ were used for all experiments in this chapter. Large $n_i$ values imply a greedy algorithm while smaller $n_i$ implies a random or exploratory nature. For the case of $n_i = n_s = 1$ the algorithm reduces to purely random sampling.

\begin{figure}[h]
\centering
\begin{algorithmic}
\State $\Omega \leftarrow$ domain of black box function $f$
\State $d \leftarrow$ dimension of $\Omega$
\State $k \leftarrow 2(d+1)$
\State $n_i \leftarrow$ number of candidate sample points per iteration
\State $n_s \leftarrow$ number of best candidate points sampled
\State $X_0 \leftarrow$ initial set of points (i.e. corners, star, random)
\State $Y_0 \leftarrow f(X_0)$
\While{(stopping criterion not met)}
\State $X_i \leftarrow n_i$ uniform random points in $\Omega$
\State $\xi_i \leftarrow$ empty set
\ForEach{$x \in X_i$}
\State $(X_k, Y_k) \leftarrow \text{KNN}(X_0, Y_0, x, k)$
\State $P \leftarrow \text{LeastSquaresPlane}(X_k, Y_k)$
\State $D_k \leftarrow \text{PointToPlaneDist}(P, X_k, Y_k)$
\State $\xi_i \leftarrow \xi_i \cup \text{Sum}(D_k)$
\EndFor
\State $X_s \leftarrow \text{MostNonlinearPoints}(X_i, \xi_i, n_s)$
\State $Y_s \leftarrow f(X_s)$
\State $X_0 \leftarrow X_0 \cup X_s$
\State $Y_0 \leftarrow Y_0 \cup Y_s$
\EndWhile
\end{algorithmic}
\caption{Pseudo code for the piecewise linear adaptive sampling algorithm.}
\end{figure}
6.1.2 Example – 2D Chirp Function

The adaptive sampling algorithm was applied to a non-linear chirp function in two dimensions given by the following formula.

\[ g(x) = \cos(2\pi(f_0 + x(f_1 - f_0))) \]  \hspace{1cm} (24)

\[ h(x_1, x_2) = g(x_1)g(x_2) \]  \hspace{1cm} (25)

The frequency of the single variable cosine function \( g(x) \) is swept linearly from starting frequency \( f_0 \) at \( x = 0 \) to ending frequency \( f_1 \) at \( x = 1 \). The 2D chirp function \( h(x_1, x_2) \), illustrated in Figure 6-2, is taken as the product of two single variable chirps swept from 0.5Hz to 2Hz.

The test function was sampled 1000 times using both adaptive sampling and random sampling. Figure 6-2 shows these sampling results superimposed on a contour plot of the test function. The adaptive sampler clearly realizes a higher density of sample points at the peaks and saddle points of the test function where non-linearity of the function is highest.

6.1.3 Piecewise Linear Modeling

In order to evaluate the effectiveness of the adaptive sampling algorithm, piecewise linear models were constructed based on gridded, random and adaptive sampling. Piecewise linear models have been successfully used for behavioral and device modeling [56], [12]. We now discuss the details of the sampling and validation process.

Since we wish to compare gridded sampling to unstructured sampling, measurements of model accuracy were dictated by the discrete nature of gridded sampling. The number of grid points in each dimension was varied from three to six, resulting in sample point cardinalities of \( \{3^4, 4^4, 5^4, 6^4\} = \{81, 256, 625, 1296\} \) for four dimensional performance metrics and \( \{3^5, 4^5, 5^5, 6^5\} = \{243, 1024, 3125, 7776\} \) for five dimensional ones. Both random and adaptive samplings were initialized with corner and star points in the domain hyper-cube, and were then filled out to the necessary cardinality for comparison with gridded sampling.

![Figure 6-2. Left: 2D “chirp” test function; Middle: Contour plot of test function with random sampling; Right: Contour plot with adaptive sampling. Note the concentration of sample points at high frequency peaks and saddle points.](image)
6.1.4 Delaunay Tessellation

A popular method for creating piecewise linear models for a 3D surface is to subdivide the 2D domain space into disjoint triangles, such that the union of these triangles gives back the original domain space. The vertices of the triangles are sample points and no two triangles are allowed to overlap. The triangles are “lifted” to a height determined by evaluation of the unknown function at the triangle vertices. This triangulation can take on many different forms, but by far the most popular constraint is the Delaunay criterion [69]. A Delaunay triangulation is achieved when the circumsphere of each triangle contains no other sample point in the space. This concept is extensible to higher dimensions in which the triangles become simplexes (sets of \(d+1\) fully connected points living in \(d\)-dimensional space), with circumsphere containment checking becoming a hyper-sphere containment check. A 2D parabolic function, sampled and triangulated using the Delaunay criterion, is shown in Figure 6-3. The triangulation serves as a continuous, piecewise linear model of the parabolic function.

A Delaunay tessellation directly supports linear interpolation by projecting unknown points onto the linear piecewise-continuous facets describing the high-dimensional function [39]. Fortune [29] gives an excellent overview of Delaunay tessellations. All Delaunay tessellations in this paper were constructed using the publicly available Quickhull program written by Barber at the University of Minnesota [6]. Access to this program is also available in Matlab v.6 through the `delaunayn()` function call.

6.1.5 Validation Data and Domain Truncation

While a Delaunay based linear interpolator is practical and accurate over much of the function domain, the interpolation accuracy can suffer artificially at the boundaries of the hyper-cube. This is caused by poorly shaped simplexes spanning large areas of the domain (for instance, from one corner of the hyper-cube to another). For this reason the data sets used for validating the three sampling methodologies were constrained to a truncated hyper-cube smaller than the original function domain. By removing 0.1 units from all sides of the unit hyper-cube, we were able to eliminate
artificially high interpolation errors in the Delaunay linear model due to skewed or poorly shaped simplices. In this paper a validation set of 10,000 uniform random points sampled in each dimension within the interval [0.1, 0.9] was used to validate the accuracy of the gridded, random, and adaptive sampling methodologies for all extracted performance parameters.

6.1.6 Analog Macro-Modeling Example - Single-Stage Op-Amp

The op-amp topology was borrowed from [87] and is illustrated in Figure 6-4. The author’s prescribed methodology for transistor matching and dimensional reduction of the input space was followed resulting in a circuit with 5 or less design variables. Six performance parameters for the op-amp were sampled and modeled using the adaptive, gridded and random sampling strategies, namely: 1) open loop gain, 2) unity gain frequency, 3) phase margin, 4) common mode rejection ratio, 5) positive power supply rejection ratio and 6) negative power supply rejection ratio. Open loop gain, common mode rejection ratio, and positive/negative power supply rejection ratios depend only on transistor sizes and have no dependence on capacitive input parameters at low frequencies. For this reason these performance parameters have only four input dimensions. Phase margin and unity gain frequency do depend on variations in the compensation capacitance and thus were modeled using five independent variables. Logarithmic scaling of the components (transistor sizes and compensation capacitance) was also performed as discussed in [87]. Ranges for the physical device sizes are listed in Table 6-1. Transistor models used in this paper for simulation were TSMC 0.18µm BSIM3v3.1 with ±1.8V supply. All circuit simulations were performed on UNIX SunBlade 100 workstations with 256MB RAM using ngspice version 14, from U. C. Berkeley. Typical run times were 4 to 5 hours to sample a 5D op-amp performance metric for 7776 points.

Six performance parameters for a single stage CMOS operational amplifier were sampled and modeled using the adaptive, gridded and random sampling strategies, namely: 1) open loop gain, 2) unity gain frequency, 3) phase margin, 4) common mode rejection ration, 5) positive power supply rejection ratio, and 6) negative power supply rejection ratio. Open loop gain, common mode rejection ratio, and positive/negative power supply rejection ratios depend only on transistor sizes and have no dependence on capacitive input parameters at low frequencies. For this reason these performance parameters have only four input dimensions. Phase margin and unity gain frequency do depend on variations in the compensation capacitance and thus were modeled using five independent variables. Logarithmic scaling of the components (transistor sizes and compensation capacitance) was also performed as discussed in [87]. Ranges for the physical device sizes are listed in Table 6-1. Transistor models used in this paper for simulation were TSMC 0.18µm BSIM3v3.1 with ±1.8V supply. All circuit simulations were performed on UNIX SunBlade 100 workstations with 256MB RAM using ngspice version 14, from U. C. Berkeley. Typical run times were 4 to 5 hours to sample a 5D op-amp performance metric for 7776 points.
supply rejection ratio and 6) negative power supply rejection ratio. The adaptive sampling algorithm consistently outperformed gridded and random sampling methodologies with respect to maximum error using the piecewise linear Delaunay model. This trend is well represented in the error plots in Figure 6-6 using the validation set of 10,000 random points on [0.1, 0.9]. Detailed error metrics are summarized in Table 6-2.

### 6.1.7 RF Mixer/Down-converter

To further convey the utility of the adaptive sampling algorithm, the conversion gain of an RF mixer topology was sampled and modeled using the same methodology as the single stage op-amp example. The mixer circuit was borrowed from [49] and is shown in Figure 6-5. Device matching constraints and ranges are listed in Table 6-3. Typical run times for transient analysis of the mixer circuit were 5 to 6 hours for 7776 points.

---

#### Table 6-2

<table>
<thead>
<tr>
<th>Performance Parameter</th>
<th>Data Range</th>
<th>Sampling Points</th>
<th>Adaptive Sampling</th>
<th>Grid Sampling</th>
<th>Random Sampling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open Loop Gain</td>
<td>[61.13, 89.39] dB</td>
<td>81</td>
<td>Mean 0.251</td>
<td>Mean 0.262</td>
<td>Mean 0.265</td>
</tr>
<tr>
<td>Absolute Error</td>
<td>[61.13, 89.39] dB</td>
<td>256</td>
<td>Std 0.537</td>
<td>Std 0.852</td>
<td>Std 0.596</td>
</tr>
<tr>
<td></td>
<td>[61.13, 89.39] dB</td>
<td>625</td>
<td>Max 2.997</td>
<td>Max 3.896</td>
<td>Max 2.624</td>
</tr>
<tr>
<td></td>
<td>[61.13, 89.39] dB</td>
<td>1296</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unity Gain Frequency</td>
<td>[3.37e3, 1.37e8] Hz</td>
<td>243</td>
<td>Mean 14.48</td>
<td>Mean 26.28</td>
<td>Mean 15.86</td>
</tr>
<tr>
<td>Phase Margin Absolute Error</td>
<td>[-126.9, 89.5] degrees</td>
<td>3125</td>
<td>Mean 31.49</td>
<td>Mean 31.49</td>
<td>Mean 114.90</td>
</tr>
<tr>
<td>CMRR</td>
<td>[42.38, 103.82] dB</td>
<td>7776</td>
<td>Std 42.79</td>
<td>Std 42.79</td>
<td>Std 42.79</td>
</tr>
<tr>
<td>Positive PSRR Absolute Error</td>
<td>[36.56, 117.83] dB</td>
<td>81</td>
<td>Mean 2.051</td>
<td>Mean 2.051</td>
<td>Mean 2.051</td>
</tr>
<tr>
<td>Absolute Error</td>
<td>[36.56, 117.83] dB</td>
<td>256</td>
<td>Std 0.473</td>
<td>Std 0.473</td>
<td>Std 0.473</td>
</tr>
<tr>
<td>Negative PSRR Absolute Error</td>
<td>[63.14, 96.08] dB</td>
<td>625</td>
<td>Mean -0.028</td>
<td>Mean -0.028</td>
<td>Mean -0.028</td>
</tr>
<tr>
<td>Absolute Error</td>
<td>[63.14, 96.08] dB</td>
<td>1296</td>
<td>Std 0.012</td>
<td>Std 0.012</td>
<td>Std 0.012</td>
</tr>
</tbody>
</table>

---

69
Two signals were applied to the RF mixer, 1) a 0.1 V 900MHz radio frequency input sine wave between terminals RF+ and RF-, and 2) a 1V 1GHz local oscillator sine wave at terminals LO+ and LO-. The expected output signal is a 100MHz intermediate frequency signal at the IF+ and IF- terminals. In order to extract conversion gain for the RF mixer, a transient simulation was run for 3 periods of the output signal, i.e. 30 ns. The “fourier” command in SPICE was used to extract the magnitude of the output signal at 100MHz. The conversion gain is then the output signal magnitude at 100MHz divided by the input signal magnitude at 900MHz. In Table 6-4 we see that the adaptive sampling algorithm provided enhanced accuracy when compared to gridded and random sampling for mean error, maximum error, and standard deviation.

6.1.8 Conclusions

Computationally, the piecewise linear based adaptive sampler is easily justifiable. Measured run times increased less than 5% when using the adaptive sampler even with un-optimized, exhaustive nearest neighbor search (which is the most expensive aspect of the adaptive sampler). From Table 6-2 and Table 6-4 we see that mean error was quite good for all the performance parameter models. The piecewise linear based adaptive sampler successfully and consistently achieves lower maximum error as well while sometimes exhibiting a slightly higher mean error compared to the other two sampling methodologies. By placing a higher density of points in nonlinear regions, there are fewer points available to accurately describe the function in linear or nearly linear regions.
Figure 6-6. Number of samples vs. maximum error for the six piecewise linear performance models using gridded, random and adaptive strategies with a validation set of 10,000 random points.
Linear interpolation based on Delaunay tessellation suffers from the so-called curse of dimensionality. The number of simplexes increases exponentially with respect to dimension \([39]\). Thus the time used to triangulate the space and the space used to store the triangulation increase exponentially with respect to dimension. However, other options exist such as the local modeling technique used by Gross in \([39]\). Rather than triangulating the entire space, nearest neighbor search can be used in conjunction with a small local Delaunay triangulation to perform linear interpolation.

### 6.2 Adaptive Sampling with Duchon Pseudo-Cubic Splines

In this section a novel algorithm is presented for adaptively sampling multi-dimensional black box functions. This algorithm is coupled to a modeling methodology known as pseudo-cubic splines \([23]\) which readily interpolate multi-dimensional scattered data points. Pseudo-cubic splines are attractive because they exactly define the discrete sample data and can be solved for by solving a linear system of equations, as opposed to neural networks which only approximate the data. Our methodology is used to guide the exploration of various performance parameters through selective and adaptive acquisition of discrete data points via the circuit simulator SPICE. The performance parameters are also sampled using traditional gridded sampling as well as a quasi-random space filling sampling method known as a Halton sequence \([52]\). Quasi-random numbers are able to fill a space more uniformly than uncorrelated pseudo-random numbers and are useful for “off-grid” but thorough function exploration. Performance models are constructed via pseudo-cubic splines using data from all three sampling methodologies and relative accuracies are compared using a large random validation set. It is shown that intelligent choice of sample point placement enhances model accuracy and that comparable models can be built with fewer sample points using an adaptive scheme.

#### 6.2.1 Duchon Splines on Unstructured Data

Multi-dimensional splines are traditionally computed using tensor products of one-dimensional splines. They require full rectangular gridded data for proper construction and evaluation. While this is practical for closed-form equations with fast evaluation and a guaranteed numerical solution for all grid points, problems arise when incorporating circuit simulators such as SPICE. Due to numerical instabilities when solving the circuit equations, it may be impossible to acquire performance data for all grid points in the domain space \([87]\), rendering tensor product splines unusable.
The spline equations studied by Duchon [23] are a viable alternative to tensor product splines. They allow the use of scattered data points, provide exact interpolation (as opposed to approximation as is the case with neural networks and smoothing splines), and can be used for any dimension desired. One such class derived by Duchon is the well known thin-plate spline which minimizes the energy of a clamped (via the discrete data points) elastic plate [28]. We will later discuss a problem encountered when using thin plate splines, however, let us first discuss the class of splines used in this paper known as pseudo-cubic splines which have the form:

\[ Z_i = \sum_{j=1}^{k} d_j \phi(x_i - x_j) + P^m(x_i) \quad (26) \]

\[ \phi = ||x||_2^3 \quad (27) \]

The height \((z_i)\) of the \(N\)-dimensional point to interpolate \((x_i)\) is a weighted \((d_j)\) summation of basis functions \((\phi)\) applied to the difference between the unknown point and all \(k\) number of sampled points \((x_j's)\) currently defining the spline, plus a polynomial of degree \(m=1\). The basis function is the Euclidean norm cubed, i.e. the cubed distance between the points \(x_i\) and \(x_j\). Unlike neural network related radial basis functions, the basis function used here has global support. It is possible to uniquely solve for the basis function coefficients and the first order polynomial coefficients given a set of sample points by solving a dense system of linear equations. These coefficients along with the location of the sampled data points completely define the spline interpolate. Turk [82] discusses the details of setting up and solving such a linear system.

In theory it should be possible to build a Duchon-pseudo cubic spline model from any data set with unique sample points regardless of the distribution of the points, however, in practice this is not the case. Pseudo-cubic splines require that no two samples are “too close” to one another to ensure that the dense linear system of equations used to solve for the spline coefficients is non-singular. Pseudo-random points have a tendency to “clump” and have the potential to generate two or more points in the data set that are very near to one another. We have never had a problem solving for a Duchon pseudo-cubic spline when the sampled points are from a space filling technique such as the Halton random generator. In contrast, matrix inversion problems almost always occur when the sampled points are from a pseudo-random number generator. The exception to this rule is the pseudo-cubic spline based adaptive sampling algorithm developed in this section. Pseudo-random points may be used as the candidate point set in this setting. The adaptive sampler will not choose to sample a point from the candidate set that will cause the spline to become unstable. In practice
care must be taken to choose a sample set that the does not cause instability of matrix the matrix inversion required to solve for the pseudo-cubic spline models.

As mentioned earlier, thin-plate splines may also be used to interpolate multi-dimensional scattered data. The mathematical form of thin-plate splines is very similar to that of pseudo-cubic splines except for a difference in basis function and polynomial order [23]. For pseudo-cubic splines the order of the polynomial is strictly \( m = 1 \), but for thin-plate splines the polynomial degree must be \( m > \frac{D}{2} - 1 \) where \( D \) is the dimension of the function domain. Note that for three or four dimensional functions a first order polynomial is sufficient, but for higher dimensions the polynomial order must increase at half the rate of the domain dimension. This is somewhat of an inconvenience in that the number of terms in a multi-dimensional quadratic (cubic, quartic, quintic, etc.) is cumbersome. In addition, the terms in the polynomial add to the size of the linear system that must be solved. For these reasons all splines used in this paper are pseudo-cubic. For qualitative purposes, Figure 6-7 illustrates the approximation to an unknown function in one dimension using pseudo-cubic and thin-plate splines as well as a piecewise-continuous linear interpolate. Note that the pseudo-cubic and thin-plate splines produce similar approximations; however, the pseudo-cubic spline is considered the “smoother” interpolate. The thin-plate spline typically resides between the low-order piecewise linear interpolate and the pseudo-cubic spline.

The computational complexity of solving a set of dense linear equations is \( O(n^3) \) for \( n \) equations. In addition, solutions for dense systems become unstable as the number of equations increase. These considerations limit the practical number of points that can be used to create a spline to a few thousand points [82]. We have successfully and consistently solved for Duchon splines using 5000 sample points on a Sunblade 1000 machine with 2GB of RAM in under 55 minutes. This may seem rather limiting, however, local modeling approaches exist to overcome this problem by subdividing the domain into sub-regions which are individually modeled and later recombined while retaining \( C^2 \) continuity [28]. An alternative local modeling approach is used in this paper involving nearest neighbor search. For a given neighborhood a small local spline is quickly solved for around a point.
of interest. This unfortunately results in a discontinuous approximation of the underlying function across neighborhoods; however, strict continuity is not crucial to our methodology.

### 6.2.2 Adaptive Sampler Algorithm

Our adaptive sampling algorithm works directly in conjunction with Duchon pseudo-cubic splines. The basic premise is that if a spline is well defined, i.e. the function is sufficiently sampled, the removal of one point from the spline should not deform its shape too drastically. We base our sampling strategy on the relative magnitudes of changes in the spline structure when points are added and deleted.

Perhaps the easiest way to convey the idea behind our adaptive sampler is through an illustrative example. Figure 6-8 shows a 1-dimensional pseudo-cubic spline function $S_0$ (solid line) defined by a set $X_0$ of 8 discrete values at $x = \{0, 0.1, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$ with heights $Z_0$ for which the precise values are not important (the hollow circles as well as the darkened point at $x = 0.1$ all comprise the original spline $S_0$). Notice that the spline has more sample points to the right of $x = 0.5$, but approximately the same functional complexity over the entire domain (qualitatively speaking). It seems obvious that adding a new sample point somewhere in the interval $[0.1, 0.5]$ would prove to be the most promising way to refine the spline approximation. However, this needs to be quantified numerically in order to automate such a process.
We consider two new points not used to build the spline, namely \(x_1 = 0.05\) and \(x_2 = 0.35\). We call these candidate points and wish to determine which of the two to sample (in general more then two candidate points may be considered simultaneously). First we interpolate the height values \(z_1\) and \(z_2\) of the candidate points using the original spline function \(S_0\). The candidate points are designated by asterisks in Figure 6-8. We create two artificial data sets \(\{X_1, Z_1\}\) and \(\{X_2, Z_2\}\) by removing one point from the original set \(\{X_0, Z_0\}\) and adding back \((x_1, z_1)\) and \((x_2, z_2)\) respectively. For this example the darkened point at \(x = 0.1\), was chosen for deletion from the original spline. Two new splines are created, \(S_1\) and \(S_2\), using the augmented data sets. Both new splines exactly intersect their respective candidate point but miss the deleted point. We note that \(S_2\) deviates largely from the original spline while \(S_1\) more closely resembles \(S_0\). By comparing the infinity norms of the difference between \(S_0\) and the two mock splines we can detect the point causing the largest deviation from the original spline, namely \(x_2\), which intuitively seems to be the better of the two points to sample. Note that the point (at \(x = 0.1\)) deleted from the set \(X_0\) is only temporarily removed. After choosing the candidate point indicating the largest error, it is sampled and added to the original spline, thus increasing the number of discrete points comprising the spline. In general, the point temporarily deleted from the original spline is always taken as the one closest to the candidate point, allowing the algorithm to extend to higher dimensions. This process facilitates adaptive sampling of an unknown function by analyzing sets of randomly generated candidate points within \(\Omega\), always choosing the next sample point to be the one causing the largest deviation from \(S_0\).

Pseudo-code is shown in Figure 6-9 for the \(N\)-dimensional version of the adaptive sampling algorithm. This algorithm is nearly identical to the one dimensional case described previously. Lines 1-5 of the pseudo-code declare user defined parameters of the algorithm. Based on the evaluation time of the black box function \(f\), the user would choose values maximizing the aggressiveness of the algorithm while trying to minimize the overhead of the adaptive sampler itself. We have

\[
\begin{align*}
\Omega & \leftarrow \text{domain of black box function } f \\
k & \leftarrow \text{number of neighbors used in local spline augmentation} \\
n_i & \leftarrow \text{number of candidate sample points per iteration} \\
n & \leftarrow \text{number of pts used in inf. norm approximation} \\
X_0 & \leftarrow \text{initial set of points (i.e. corners, star, random)} \\
Y_0 & \leftarrow f(X_0) \\
\text{While (stopping criterion not met)} & \\
X_i & \leftarrow n_i \text{ uniform random points in } \Omega \\
\xi & \leftarrow \text{empty set} \\
\text{foreach } x_i \in X_i & \\
(X_i, Y_i) & \leftarrow \text{KNN}(X_0, Y_0, x_i, k) \\
S_i & \leftarrow \text{NewDuchonSpline}(X_i, Y_i) \\
(x_i, y_i) & \leftarrow \text{KNN}(X_0, Y_0, x_i, 1) \\
X_1 & \leftarrow (X_i \cap \text{comp}(x_i)) \cup x_i \\
Y_1 & \leftarrow \text{SplineEval}(S_i, X_1) \\
S_j & \leftarrow \text{NewDuchonSpline}(X_j, Y_j) \\
\xi & \leftarrow \xi \cup \text{ApproxInfNorm}(S_0, S_1, x_i, n_v) \\
\text{end foreach} & \\
X_s & \leftarrow \text{ChoosePointWithLargestNorm}(X_i, \xi) \\
Y_s & \leftarrow f(X_s) \\
X_0 & \leftarrow X_0 \cup X_s \\
Y_0 & \leftarrow Y_0 \cup Y_s \\
\text{end while} & 
\end{align*}
\]

Figure 6-9. Pseudo-code for the adaptive sampling algorithm based on Duchon pseudo-cubic splines.
no strict method to determine the algorithm parameters, however, the behavior of the sampler seems to be insensitive to minor variations in these numbers.

The functions in boldface type in Figure 6-9 need further explanation. The $\text{KNN}(X, Y, x, k)$ function searches for the $k$-number of points in $X$ that are closest to point $x$, and returns those points along with their associated height values in $Y$. For simplicity an exhaustive search was used even though more efficient methods exist for nearest neighbor search. $\text{NewDuchonSpline}$ simply creates a new spline interpolate using the $k$-nearest points by solving the dense system of linear equations mentioned previously. Note that on line 13 the $k$ argument to $\text{KNN}$ has been replaced by the number 1. This causes it to retrieve only one neighbor of $x_i$, i.e. the closest point $x_k$. Subsequently, this closest point is removed from $X_k$ via an intersection with a set complement in line 14. In addition, $x_i$ is added to this reduced set to produce the artificial data sets $X_1$, $Y_1$, for which all points lie on the surface defined by the original spline $S_0$.

We now discuss the $\text{ApproxInfNorm}(S_0, S_1, x_i, n_v)$ function which is the only aspect of the algorithm that is non-trivial to understand when generalizing to multi-dimensional functions. In the one dimensional case we were uniquely bounded by the extreme points of the original spline and needed only to search for the maximum difference between two splines over a small fixed interval. For multi-dimensional functions we approximate the infinity norm in the following manner. For a candidate point $x_i$ and its $k$-nearest neighbor points $X_k$ we define radius $r$ as the average Euclidean distance between $x_i$ and all its $k$-neighbors. We then generate $n_v$ random points uniformly inside a hyper-sphere (with dimension equal to the domain dimension) of radius $r$ centered around point $x_i$ while rejecting points not inside $\Omega$. These $n_v$ random points are evaluated using both splines $S_0$ and $S_1$ and the maximum difference over this discrete set is taken as the approximate infinity norm. There may be other ways to calculate this norm; however, we have had good results using this methodology. It is also fairly simple and computationally feasible.

The rest of the algorithm is straightforward. From the set of candidate points in $X_i$ we always choose the point causing the largest deviation from $S_0$. This point is sampled and incorporated into the discrete set of unstructured points currently describing the unknown function.

### 6.2.3 Test Functions

The first test function is the one dimensional “chirp” function shown in Figure 6-10, calculated using:

$$ y = \cos\left(\frac{1}{2} \pi (f_i - f_0) x^2 + f_0 x \right) $$

(28)
The frequency of the cosine sweeps quadratically from a starting frequency of $f_0 = 0.5\text{Hz}$ to an ending frequency $f_1 = 100\text{Hz}$. The function was sampled with 400 points using the adaptive sampler as well as uniform sampling (Halton sampling is nearly identical to uniform sampling in the one-dimensional case). The adaptive sampler was seeded with 10 uniform sample points and the algorithm parameters $k$, $n_i$, and $n_v$ were set to 10, 20, and 20 respectively. Note that we are well above the Nyquist sampling frequency when using 400 uniformly spaced sample points and therefore expect to get an accurate portrait of the sampled test function even at high frequencies. However, at low frequencies many sample points are unnecessarily used to sample the waveform. These wasted points could be used to improve the accuracy of the spline interpolate if distributed approximately proportionally to the quadratic increase in frequency of the chirp function. Referring to Figure 6-11 we see that this is precisely what the adaptive sampler accomplishes. The histogram shows the sample point density achieved by the two sampling methodologies. A least squares quadratic has been fit to the adaptive sample point density to emphasize the utility of this methodology. Pseudo-cubic splines were constructed using the uniform and adaptive sample points and validated against a set of 5000 random samples. The maximum absolute error between the true chirp function and splines constructed using the adaptively sampled points and uniform points was 0.00603 and 0.0791 respectively, a factor of improvement of 13.1.
The second test function is the nonlinear two dimensional function in Figure 6-12. It is constructed from a sum of weighted sigmoid functions. The function was sampled using adaptive sampling, gridded sampling and Halton sampling strategies for 2000 points. The corners of the function domain were also added to the Halton sample set. The adaptive sampler was seeded with corner points as well as the first 20 points from the Halton sequence. The parameters of the adaptive sampler were $k=20$, $n_i=50$, $n_v=25$. Referring to Figure 6-12, the adaptive sampler clearly realizes a high density of points around nonlinear parts of the function. The maximum absolute errors obtained using a validation set of 5000 random points for the adaptive, gridded, and Halton methodologies were respectively 0.00235, 0.0126, and 0.0683; factors of improvement of 5.37 and 29.06.
6.2.4 Analog Macro-Modeling Example - Two Stage CMOS Op-Amp

An operational amplifier circuit is presented for which adaptive, gridded and Halton sampling methodologies are applied. Various performance parameters are modeled using the Duchon pseudo-cubic splines and accuracies are compared between the three different sampling methods.

Six AC performance parameters were modeled for the operational amplifier topology in Figure 6-4, namely 1) open loop gain ($A_0$), 2) unity gain frequency (UGF), 3) phase margin (PM), 4) common mode rejection ratio (CMRR) and 5,6) positive and negative power supply rejection ratios (PSRRp, PSRRn). For low frequency performance metrics $A_0$, CMRR, PSRRp, and PSRRn, there are four free input variables, while UGF and PM have five free variables due to their dependence on the coupling capacitor $C_C$. For reasons discussed in [87], component sizes are logarithmically scaled and normalized to the interval [0, 1]. Transistor models used for simulation were TSMC 0.18m with 1.8V supply. Two domain sizes were analyzed in separate experiments using the same op-amp topology. Figure 6-14 and Figure 6-15 contain error plots for the three modeling methodologies. Note that in both figures the graphs are subdivided into a left and right area. The left areas contain plots of the circuit with a “small” domain and the right areas contain error plots for the “large” function domain. Table 6-5 gives exact values for the large and small domains, but in short, the small domain extends one order of magnitude and the large domain extends over two orders of magnitude for all free circuit parameters.

Note that the adaptive sampler is not as effective for the op-amp when modeling the small domain space. In fact, gridded sampling is hard to beat, however, adaptive sampling clearly wins out over both gridded and quasi-random sampling when the large domain space is considered. For all plots in Figure 6-14 and Figure 6-15 a validation set of 10,000 uniform random points was used to test the accuracy of the spline models. In each of the plots the upper family of curves is maximum absolute modeling error while the bottom family of curves is standard deviation of the signed errors. In addition, the mean value of the errors was also analyzed and compared to the standard deviation. In all cases the mean of the signed errors was at least ten times smaller than the standard deviation implying an unbiased estimator. Typical execution times required to adaptively sample 5000 points of an op-amp performance parameter were 8-9 hours on a Sunblade 1000 machine with 2GB of RAM. The specific but illustrative case of open loop gain required 6.3 hours for SPICE simulations and 2.4 hours for adaptive sampler.

Table 6-5
Component Ranges of Operational Amplifier

<table>
<thead>
<tr>
<th>Independent Parameters</th>
<th>M1 = M2</th>
<th>M3 = M4</th>
<th>M5</th>
<th>M7</th>
<th>$C_C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small Range</td>
<td>[2, 20]</td>
<td>[2, 20]</td>
<td>[2, 20]</td>
<td>[10,100]</td>
<td>[0.1,1]</td>
</tr>
<tr>
<td>Large Range</td>
<td>[2, 200]</td>
<td>[2, 200]</td>
<td>[2, 200]</td>
<td>[10,1000]</td>
<td>[0.1,10]</td>
</tr>
</tbody>
</table>

NOTE: All transistor lengths fixed at 2µm. Width of $M_8 = (2^3 M_3/M_7) / M_5$. $C_L$ fixed at 10pF. $I_{bias}$ fixed at 1µA.
overhead. Algorithm parameters were set to $k=50$, $n_i=50$, and $n_v=25$ when sampling analog performance parameters. The corners of the hyper-cube domain were added to the Halton sequence for spline model creation. The adaptive sampler was seeded with 100 points which included corner data as well as points from the Halton sequence. Although the computational requirement of the adaptive sampler is significant compared to SPICE, we believe this could be reduced by developing a more efficient implementation (all code was written in Matlab for this paper).

6.2.5 Conclusions

The pseudo-cubic spline based adaptive sampler proved useful for enhancing the accuracy of all of the performance parameter models when the range of the component values was large, performing most notably for UGF, PM, CMRR, and PSRRn. The discrepancy in performance of the adaptive sampler between the large and small domains may be attributed to the fact that gridded sampling is not a bad choice for sampling methodology and gives satisfactory results as long as the function is not too complex. In the case that the grid density is sufficient to capture all features of the function, then gridded sampling produces exceptional models. However, in the case of the large function domain, it may be that the function is complex enough that gridded sampling misses some important features in the performance metric. However the adaptive sampler is able to resolve features that gridded sampling could not, thereby winning out over both gridded and pseudo-random strategies for the larger function domain.

The adaptive sampler is heuristic by nature and is intrinsically coupled to pseudo-cubic splines. There is a practical upper limit of approximately 5000 points that can be used to collectively assemble a spline model. This is due to the inherent instabilities involved in creating the spline as well as memory and computational requirements. This limit can be overcome via division of the domain into locally modeled sub-spaces, individually requiring less than 5000 points. Pseudo-cubic splines have no free parameters and are completely determined by the discrete sampled data, thus requiring no human interaction for proper construction.

We emphasize the inherent problems associated with gridded sampling. The number of grid points grows exponential with both domain dimension and the number of grid points in each dimension. It is necessary to decide \textit{a priori} the density of the grid with which to sample, regardless of the modeling accuracy that may arise due to that choice. In addition, control of the number of sample points is very coarse allowing only greatly increasing, discrete jumps in grid density. Our adaptive sampler intelligently samples the design space, thus allowing the user more flexibility in choosing limits on number of sample points and/or time investment in model building. The other alternative
Figure 6-14. Performance of the pseudo-cubic spline based adaptive sampler for three of the six performance metrics over both a small domain (1 order of magnitude change in each variable) and large domain (2 order of magnitude change).
Figure 6-15. Performance of the pseudo-cubic spline based adaptive sampler for the three remaining performance metrics over both a small domain (1 order of magnitude change in each variable) and large domain (2 order of magnitude change).
sampling methodology presented, the Halton sequence, also retains this flexibility. However, a Halton sequence is deterministic (as is gridded sampling) and does not take into account the sample history or the structure of the data. Compared to Halton sampling our adaptive sampling algorithm even more clearly stands out, as shown for both the large and small domains in Figure 6-14 and Figure 6-15.

Another advantage to using the adaptive sampler is the concept of self-validation. Suppose a model is built using gridded sampling. This model must be validated using another set of sample points not used to create the model. The adaptive sampler is self-validating in the sense that it is constantly striving to find and sample areas in the model with the largest error. By recording a history of errors resolved in the spline during the sampling process, we can essentially gain confidence in our model by assuming that the errors discovered by the sampling methodology are indicative of the worst case errors still resident in the spline model.

In summary, the adaptive sampler accomplishes equal or better accuracy with respect to maximum absolute error over the large domain using significantly fewer sample points when compared to gridded sampling for four of the performance parameters. Specifically:

- UGF with 1000 adaptive points beats gridded at 5000
- PM with 1000 adaptive points beats gridded at 4096
- CMRR with 1000 adaptive points beats gridded at 4096
- PSRRn with 2000 adaptive points beats gridded at 4096

For the remaining two performance metrics, accuracies for all three sampling methods are similar with only a slight benefit apparent when using the adaptive sampler.

### 6.3 Summary of Adaptive Sampling Methodologies

While adaptive sampling is attractive due to a consistent decrease in maximum model error when compared to random and gridded sampling, there are other motivations for its use. Gridded sampling is impractical for high dimensional domains due to the exponential number of points needed to fill out the space. This is a strong motivation for unstructured sampling methods; however, purely random sampling seems ad hoc with the potential to miss important nonlinear features. The proposed adaptive sampling algorithms are viable alternatives supporting unstructured data while striving for model accuracy and intelligent placement of costly sample points.
7 Analog Performance Modeling with Feasibility Regions

The previous chapters of this thesis have established that fast and accurate analog performance macro-models can be constructed over a discretely sampled box-constrained domain. Analog performance macro-models can greatly reduce the time needed to synthesize a circuit by replacing the simulator SPICE by fast mathematical functions which directly compute circuit performances from device dimensions. Once a valid model is generated, it can be used as a performance estimator in synthesis settings normally requiring SPICE for the evaluation of performance metrics. In addition performance macro-models do not suffer from numerical instabilities and convergence problems inherent to SPICE. A macro-model always functions correctly and can be used by someone with no detailed knowledge of circuit simulation. The benefits of macro-modeling come at the cost of time spent developing the models and a reduction in the accuracy of performance estimates.

Many multi-dimensional modeling methodologies, such as neural networks, radial basis functions and projection pursuit regression, require manual and often iterative tuning of parameters to achieve accurate models. To alleviate some of the user interaction required for analog performance macro-model construction, two adaptive sampling algorithms were developed in this thesis which can be used to autonomously sample and model multi-dimensional functions. Of the two adaptive samplers, we believe that the pseudo-cubic spline based sampler of Section 6.2 to be superior due to its inherent nonlinear modeling capabilities and independence from domain dimensionality. Even if a function is not adaptively sampled and simply sampled with random points, pseudo-cubic splines perform well and require no user interaction during creation. Pseudo-cubic splines and the associated adaptive sampling algorithm alleviate much of the user interaction required to build accurate models of sampled analog performance metrics.

So far in this thesis, the design parameters of analog topologies have been manually constrained by fixing all transistor lengths and using matching conditions commonly employed in
analog circuit design in an attempt to achieve proper biasing and good behavior of performance metrics. Detailed knowledge and experience in analog design is required to realize the matching constraints and to choose which design variables should be fixed, and which should be left as free parameters. The sizing rules method attempts to automatically constrain a circuit to ensure proper operation, good behavior of performance metrics, insensitivity to process variations and robust manufacturability [24][38][77][91]. The premise of this chapter is to explore the properties of the sizing rules method and its potential application to analog performance macro-modeling. Ideally we would like to fully automate the process of selecting and constraining the design variables of a circuit to further automate the macro-modeling process.

The sizing rules methodology uses design space and electrical space constraints to define a feasibility region or what has also been termed the non-pathological [38] design space. In order to efficiently apply the sizing rules method to analog performance macro-modeling, it is necessary to model the feasible region to eliminate the need to execute SPICE when testing the feasibility of a design. In this chapter we discuss two methods for modeling the feasible space defined by the sizing rules method. The performance parameters of two operational amplifier circuits are modeled in box-constrained, approximately feasible, and truly feasible design regions. Experimental evidence indicates that the sizing rules method alone is insufficient to properly constrain a circuit for the purposes of analog performance macro-modeling. It is necessary to impose additional constraints on the design variables of a circuit in order to facilitate accurate modeling of the performance metrics. When the design variables of a circuit are properly constrained, little, if any improvement in modeling accuracy is observed when the models are restricted to the feasible design region. Unfortunately, the need for manual analysis of analog circuits is not prevented by the inclusion of the sizing rules method.

7.1 Related Work

We define the device sizes of a circuit as the design space \( \mathbf{d} \) and the resulting DC voltages and currents of configuration \( \mathbf{d} \) as the electrical space \( \mathbf{e} \). There exists a mapping \( \mathbf{e} = f(\mathbf{d}) \); that is, given a circuit with specified components we may use function \( f \) (in practice a SPICE DC analysis) to find all DC voltages and currents in the circuit. In addition we define the circuit performances resulting from configuration \( \mathbf{d} \) as the performance space \( \mathbf{p} \). Again there is a mapping \( \mathbf{p} = g(\mathbf{d}) \); for each point in the design space there exists a set of performance metrics. Inversion of the function \( g \) is known as analog circuit synthesis, that is device sizes are determined from a set of required performances. Analog circuit synthesis is a difficult problem because the transformations \( f \) and \( g \) are
nonlinear, computationally expensive, and are not usually available in a closed form. In order to invert multi-dimensional nonlinear functions such as $f$ and $g$, a global optimizer such as a genetic algorithm must be employed which is capable of escaping local minimums. However, global optimization techniques can require the computation of many points (easily thousands) in the design space in order to converge, resulting in long synthesis times [87].

In [59] it was shown that, under special circumstances\textsuperscript{1}, the process of finding voltages and currents from component sizes can be reversed, in other words it is possible to find $d = f^{-1}(e)$ with some restrictions on the space $d$. This method is called “operating point driven sizing”. Although $d$ maps nicely to $e$ given a fully specified circuit, the converse is not true when the voltages and currents are fully specified. The problem is that the number of free parameters in $d$ is greater than the number of free parameters in $e$, thus multiple solutions exist for a given point in $e$. The authors fix this problem using several approaches and call this process dimensional reduction. The reduction of variables in $d$ is accomplished by fixing some transistor lengths, realizing the ratio of width to length of a transistor as a single variable, and matching transistors that are part of clearly symmetric structures such as differential amplifiers. An automatic method for dimensional reduction is not discussed in [59].

Both methods, the component driven biasing method (SPICE), and the operating point driven sizing method, suffer from a common problem; neither supports constraints in both spaces $d$ and $e$ simultaneously. Given a point in the design space $d$, we have no control over the resulting point in the electrical space $e$ and vice versa. Posting constraints on device dimensions (such as minimum length or width) as well as constraints on voltages and currents (for example, all transistors remain in saturation) is necessary to ensure that the circuit is physically realizable and that it is well behaved [38]. This idea is developed in a series of papers discussed in the next section.

7.1.1 The Sizing Rules Method

The concept of “pathological” circuits and the enforcement of electrical space constraints is first addressed by Eckmüller in [24]. It is argued that while a sized circuit may meet all high level performance constraints (gain, slew rate, PSRR, etc.), it may possess unwanted behavioral attributes and may be overly sensitive to device size variations. They realized that an analog CMOS circuit may be decomposed hierarchically into a finite set of fundamental building blocks regardless

\textsuperscript{1} A problem arises when transistor parameters are dependant on both device size and voltages/currents, as in the case of internal drain and source resistances. The resistances cannot be known until the device is sized, and the device cannot be sized until it is biased.
of the internal topology of the circuit. Each basic building block can then be analyzed individually and in detail to determine a set of “functional constraints” applicable regardless of the topology. These functional constraints ensure that each individual, hierarchically derived sub-block remains well behaved, and therefore the overall circuit remains well behaved since it is a combination of constrained sub-blocks. The arguments posed by the authors are not fully developed in this first paper; however, they do give some short examples of functional constraints that may be useful for ensuring well-behaved circuits. For example they identify three “regions of operation” for a CMOS transistor and imply that transistors should not switch between predefined roles of saturation, or linear operating regime. They also identify slightly higher level components and mention, in a loose sense, functional constraints that may be imposed such as “minimization of threshold voltage mismatch”, “minimization of channel length modulation mismatch”, and “transconductance factor mismatch”, among others. It is fairly clear in this first paper that the hierarchical characterization is still mostly hand derived and exact details for deriving such functional constraints are not explicitly given. As a design example, a folded cascode op-amp is sized without regard for internal functional constraints. Although the designed op-amp meets all performance parameter constraints, it violates 9 of 41 functional constraints. Specifically, three transistors are incorrectly operating in the linear region, gate-source voltages of current mirror transistors are too small resulting in a high sensitivity mismatch, and the drain potentials of two transistors mirroring a similar current are unreasonably mismatched. The op-amp was then resized by hand to meet all performance and functional constraints. A table in the paper outlines the improvements of the circuit meeting both performance and functional constraints. In summary we see the concept of constraints imposed on both the design space \( \mathbf{d} \) as well as the electrical space \( \mathbf{e} \) in order to avoid poor circuit behavior. It is not enough to treat the circuit as an observable black box. The internal structure of the box (DC voltages and currents) must also be analyzed, however, this can be accomplished hierarchically and independently of the specific circuit topology.
A second paper by Zizala [91] expands the ideas in [24] relying on the concept of functional constraints and in addition shows through examples that it is easier to model performances of non-pathological circuits as opposed to allowing pathological designs to be present in the model. The concepts of hierarchical decomposition and the application of functional constraints are used throughout the paper, however these concepts are not further developed. The authors do give functional constraints for a simple current mirror, Figure 7-1 and Table 7-1, giving insight into the types of constraints necessary to ensure well-behaved circuits. All functional constraints (those posed in the electrical space $e$) are strictly inequalities, no equality constraints are allowed. Each potentially nonlinear inequality constraint defines a sub-space in $e$. The intersection of all such sub-spaces forms the feasibility region for non-pathological circuits. This sub-space cannot be easily mapped back to the design space $d$, in other words feasible designs can only be detected by running a DC SPICE simulation and checking the functional constraints explicitly. However, the authors present a method for linearizing the functional constraints as well as a formula for mapping these linear approximate constraints back to the design space. Since the approximate linear constraints are only valid around one quiescent point in both the design and electrical spaces the linearized constraints can fail to detect pathological designs. As an example a folded cascode op-amp was analyzed resulting in 18 constraints on device sizes, 59 functional constraints and 9 free parameters in the design space. The functional constraints in $e$ were linearized as described above and cast into the design space $d$. The authors test the classification accuracy of the linearized constraint set by simulating a random selection of points in the design space. The linearized constraints statistically misclassified points inside the true feasibility region 15% of the time while misclassifying points outside the true feasibility region about 10% of the time. Given any point in the box-constrained design space, the point can be classified as pathological or non-pathological without running a DC SPICE simulation by evaluating the linearized functional constraints, granted with some misclassification present. The non-pathological design space is reported to be 70% smaller than the original box-constrained design space. The authors then

![Figure 7-1. Simple CMOS current mirror.](image)

<table>
<thead>
<tr>
<th>Table 7-1</th>
<th>Some possible constraints imposed on the simple two transistor CMOS current mirror</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constraint Type</td>
<td>Constraint Equation</td>
</tr>
<tr>
<td>Saturation</td>
<td>$V_{DS1} &gt; V_{GS1} - V_{T1}$</td>
</tr>
<tr>
<td>Saturation</td>
<td>$V_{DS2} &gt; V_{GS2} - V_{T2}$</td>
</tr>
<tr>
<td>Length Matching</td>
<td>$L_1 = L_2$</td>
</tr>
<tr>
<td>Width Matching</td>
<td>$W_2 = K*W_1$</td>
</tr>
<tr>
<td>Minimum Width</td>
<td>$W_1 &gt; W_{min}$</td>
</tr>
<tr>
<td>Minimum Length</td>
<td>$L_1 &gt; L_{min}$</td>
</tr>
<tr>
<td>Minimum Gate Voltage</td>
<td>$V_{G1} - V_{T1} &gt; V_{Gmin}$</td>
</tr>
<tr>
<td>Minimum Gate Voltage</td>
<td>$V_{G2} - V_{T2} &gt; V_{Gmin}$</td>
</tr>
<tr>
<td>Minimum Change in $V_{DS}$</td>
<td>$</td>
</tr>
</tbody>
</table>

![Table 7-1](image)
apply both a linear and quadratic regression model to performance metrics over the original box-constrained domain and the smaller non-pathological domain. Modeling accuracy is approximately one order of magnitude better for both the linear and quadratic regressors when constrained to the non-pathological domain.

The third paper by Schwencker [77] once again draws upon the concepts of constraints in both the design space $d$ as well as electrical space $e$. A limited example of possible constraints for a current mirror is given in the paper. Equality and inequality constraints are allowed in the design space; each equality constraint reduces the number of free parameters by one. Only inequality constraints are allowed in the electrical space. Synthesis of analog cells is formulated as a cost driven, constrained optimization problem. The authors outline in detail a trust region based algorithm utilizing concepts from characteristic boundary curves capable of solving the synthesis optimization problem. The algorithm relies upon gradient calculations via the numerical forward difference formula and is guaranteed to converge to a local minimum. As a practical example, a folded-cascode amplifier (the same one as in [24] and [91]) is analyzed yielding 165 inequality constraints, 35 equality constraints and 9 free variables. The circuit is sized by hand to obtain an initial nominal design then synthesized twice, once considering structural (electrical) constraints and once neglecting structural constraints. The synthesis time for constrained optimization was significantly less than the unconstrained case and the authors show through Monte Carlo analysis that CMRR is hypersensitive to parametric deviations in the unconstrained case. We note that the number of constraints on the same folded cascode op-amp has progressed from 41 in the first and second papers to 165 constraints\(^1\) in this third paper. Clearly the authors are building a progression of valid and pertinent hierarchical and generic constraints that may be imposed on analog topologies.

The paper “The Sizing Rules Method for Analog Integrated Circuit Design” by Gräb et al. [38] realizes what the previous three papers have been hinting at; the potential for an automatic methodology for constraining CMOS analog circuits such that they are well behaved and contain a minimum of free variables. Implementation details are finally given making it feasible for others to utilize their methodology. The library of analog sub-blocks developed by the authors is comprised of four levels. Level 0 recognizes single transistors operating as either a voltage controlled resistor (linear region) or a voltage controlled current source (saturation region). Level 1 contains seven sub-circuits composed of transistor pairs, namely a simple current mirror, level shifter, voltage ref-

\(^1\) Not all of the 165 inequality constraints mentioned in the paper are purely electrical. However, the authors did not discuss a detailed taxonomy of the constraints. A guess still implies that the number of electrical constraints imposed on the circuit is growing in size with experience and insight.
erence, current mirror load, differential pair, voltage reference, and flip-flop. The authors claim that Level 1 can be considered complete. When building the Level 1 library, they enumerated all 206 possible combinations of transistor pairs, discarding those not used in analog CMOS design. Level 2 consists of four sub-circuits, the level shifter bank, current mirror bank and four transistor current mirror. Lastly Level 3 contains only one sub-circuit, the differential stage. Once defined, all sub-circuits in the hierarchy were analyzed to determine a suitable set of low-level functional constraints ensuring robust design practices as well as non-pathological behavior. Interestingly only 30 such applicable constraints were found necessary by the authors for the entire hierarchical sizing rules library. Thankfully, all of the applicable constraints for all components in Levels 0 through 3 are enumerated and explained by the authors making it possible for us to replicate, use and analyze their work. This also sheds light on the many constraints previously mentioned, but not outlined in the first three papers. The authors note that the library is not complete above Level 1, but indicate that many of the sub-circuits arising in common analog design are present, and if necessary the library may be upgraded with additional sub-circuits. An algorithm is given for sub-block identification for any analog topology, thereby making it possible to automatically generate all necessary functional constraints for a generic topology. As an example the sizing rules methodology is applied to three analog topologies and the resulting hierarchical decomposition is given graphically for each circuit. The authors briefly mention applications for which the sizing rules method might be useful. They give an example in which a circuit is sized with and without sizing rules and once again argue that designs are more robust and less sensitive to parametric deviations when sizing rules are obeyed. Response surface modeling is also mentioned with three arguments supporting sizing rules, namely:

1) the feasible modeling region is well-defined (although not explicitly)
2) the function domain is reduced in size due to the sizing rules constraints and therefore modeling of pathological regions is avoided
3) the behavior of the circuit in the non-pathological regions is nearly linear and therefore more easily modeled.

The sizing rules method [38] has been found useful by other researchers. In [47] the sizing rules method was incorporated into a genetic algorithm based synthesis tool using SPICE as the performance estimator. The authors comment that the sizing rules methodology is important for analog design automation resulting in designs which are more tolerant of device mismatch and operating point (power supply) variations.

In [77] and [47] circuits were synthesized with SPICE acting as the performance estimator. So far in this thesis it has been shown that analog macro-models can be very useful in reducing the computational expense of performance parameter estimation. In fact, if the performance models are
fast enough (ex. neural networks) virtually any pertinent algorithm may be utilized for synthesis since the evaluation of the performance models is very fast. The cost of developing an accurate and efficient analog macro-model is justifiable when extensive reuse of the model is desirable. Reuse of library cells is common practice in industry, especially in digital semi-custom standard cell designs. Considerable time is spent developing and characterizing a comprehensive library of technology specific cells so that they may be utilized in many designs.

As previously mentioned, it is necessary to simultaneously enforce constraints in both the design space $d$ and the electrical space $e$ in order to reap the benefits of the sizing rules methodology. However, we are burdened by the fact that it is costly to simulate a design in order to find its corresponding point in the electrical space. Typically, we can only expect a user to establish simple box constraints on the design space, that is the maximum and minimum desirable lengths and widths of all transistors are given. However, it has been argued that simple box constraints lead to pathological designs, complex performance parameter behavior, and sensitivity to operating conditions and geometric variations [24][38][91][77].

### 7.2 Analog Performance Estimation with Feasibility Region Modeling

This section discusses the idea of feasibility regions in an analog circuit design space and describes two methodologies for modeling a feasibility region as well as implementation details of the sizing rules method [38]. The sizing rules methodology requires that inequality constraints on electrical parameters (voltages and currents) be enforced to ensure that the circuit is well behaved. For example consider the saturation constraints imposed on a single transistor:

$$v_{ds} - (v_{gs} - V_{th}) - V_{sat, min} \geq 0$$  \hspace{1cm} (29)

$$v_{ds} \geq 0$$  \hspace{1cm} (30)

$$v_{gs} - V_{th} \geq 0$$  \hspace{1cm} (31)

The gate-source ($v_{gs}$), drain-source ($v_{ds}$) and transistor threshold ($V_{th}$) voltages are determined by SPICE DC simulation whereas the minimum saturation voltage ($v_{sat,min}$) is a constant to ensure that the transistor robustly stays in saturation. The threshold voltage of a transistor is not a constant when using BSIM3v3 transistor models since its value is dependant on device dimensions and therefore must be extracted from a SPICE simulation.
The feasible design space is the region in which all design space and voltage space inequality constraints imposed by the sizing rules method are satisfied, however the size and shape of this region are not explicitly known. Two methodologies for approximating the feasible design space are discussed in Section 7.2.1 and Section 7.2.2. The idea being that if the feasibility region approximator is sufficiently accurate, analog performance macro-models may be restricted to the approximate feasible region in order to reduce the complexity of the behavior of performance metrics. The feasibility region model can then be used to test whether a point is feasible, thus precluding the need for a SPICE DC simulation.

Figure 7-2 illustrates the concept of a feasible region \( \bar{f} \) and an approximate feasible region \( \hat{\Omega} \) in design domain \( \Omega \). Four situations can occur when attempting to classify a point using the feasibility classifier \( \hat{f} \): 1) the point is feasible and classified as feasible (correctly positive CP), 2) the point is infeasible and classified as infeasible (correctly negative CN), 3) the point is feasible but classified as infeasible (falsely negative FN) and 4) the point is infeasible but classified as feasible (falsely positive FP). Given a test set of \( N \) randomly distributed points in \( \Omega \) for which the true classification as well as the approximated classification are known, the following metrics can be used to rate the accuracy of a feasibility estimator:

\[
\Phi_1 = \frac{CP}{CP + FN} = \frac{\# \text{ of correct positives}}{\# \text{ of correct positives} + \# \text{ of false negatives}}
\]

\[
\Phi_2 = \frac{CP}{CP + FP} = \frac{\# \text{ of correct positives}}{\# \text{ of correct positives} + \# \text{ of false positives}}
\]

Both \( \Phi_1 \) and \( \Phi_2 \) are bounded in the range \([0,1]\) and the classifier is said to be a perfect classifier w.r.t. the test set if both metrics evaluate to 1. Both metrics are necessary since the first metric is driven to 1 by eliminating all false negatives (hyper-generalization, ex. \( \hat{\Omega} = \Omega \)) and the second metric can be driven to one by eliminating all false positives (hyper-specification, ex. setting \( \hat{f} \) to any CP point).
It is useful to compare the accuracy of a classifier with that of a random one. In other words, how will a classifier fair if it simply guesses the classification of a point. The expected relative size of the true feasible region w.r.t. domain $\Omega$ is $(CP + FN)/N$ and the expected size of the approximate feasible region is $(CP + FP)/N$. Let $\hat{p} = (CP + FN)/N$ be the probability that a point chosen randomly in $\Omega$ is feasible, and let $\hat{p} = (CP + FP)/N$ be the probability that the random classifier will guess the point to be feasible. A CP occurs when the point is feasible and the classifier guesses it to be feasible, i.e. $p \cdot \hat{p}$. A FN occurs when the point is feasible and the classifier guesses it to be infeasible, i.e. $p \cdot (1 - \hat{p})$. A FP occurs when the point is infeasible and the classifier guesses it to be feasible, i.e. $(1 - p) \cdot \hat{p}$. By variable substitution into Equation 32 and Equation 33 we have:

$$\Phi_1 = \frac{CP}{CP + FN} = \frac{p \cdot \hat{p}}{p \cdot \hat{p} + p \cdot (1 - \hat{p})} = \hat{p}$$

$$\Phi_2 = \frac{CP}{CP + FP} = \frac{p \cdot \hat{p}}{p \cdot \hat{p} + (1 - p) \cdot \hat{p}} = p$$

Thus a random classifier which guesses a point to be feasible as often as the approximate feasible classifier $\hat{f}$ will achieve accuracy ratings of $\hat{p}$ and $p$. Our classifier has to do at least this well and thus should have accuracy ratings $\Phi_1 > \hat{p}$ and $\Phi_2 > p$.

### 7.2.1 Approximation of the Feasible Design Space Using Taylor Series Expansion

In [91] a method was developed for linearizing electrical inequality constraints in the design space, i.e. a fast model was produced that is capable of distinguishing between pathological and non-pathological designs without running a SPICE analysis. The method is in fact a specification of a Taylor series approximation. A multi-dimensional Taylor series expansion may be calculated to any order, however, by far the most popular formulations are first and second order approximations due to the large number of function evaluations for higher order estimates. A second order multi-dimensional Taylor series is calculated by the following formula:

$$\hat{f}(x_0 + \delta x) \cong f(x_0) + \delta x^T \nabla f + \frac{1}{2} \delta x^T H \delta x + R$$
The function \( f \) is approximated around a nominal point \( x_0 \) using gradient (\( \nabla f \)) and Hessian (\( H \)) information which may be calculated using finite difference approximation if the first and second order derivatives of \( f \) are not explicitly known. The remainder (\( R \)), which is all higher order terms of the series is dropped for a second order approximation. Figure 7-3 illustrates the linearization of a one-dimensional quadratic function \( f = -x^2 + 2.4x - 0.44 \) about a nominal point \( x_0 = 0.6 \). The feasible region is defined as all values of the design variable \( x \) for which the quadratic inequality constraint is greater than zero, i.e. \( x > 0.2 \). After linearization about point \( x_0 \) the approximate feasible region becomes all values of \( x \) for which the linearized constraint is greater than zero, i.e. \( x > 0.067 \). For an \( n \)-dimensional function, a first order Taylor series produces an \( n \)-dimensional plane that is tangent to the function being approximated at the nominal point \( x_0 \). A second order Taylor series produces a quadratic function with first and second-order derivatives equal to the first and second-order derivatives of the approximated function at \( x_0 \). In the case of the quadratic constraint in Figure 7-3, a second order Taylor series would result in an exact fit to the quadratic. It is clear that the accuracy of the linearized constraint is dependant on the linearization point \( x_0 \).

The voltage space inequality constraints posed by the sizing rules method can be approximated using a Taylor series. As described in [91], a feasible design must first be found, about which the voltage space inequality constraints may then be approximated. In this thesis we will restrict the Taylor series to a second order approximation. The gradient and Hessian information is obtained by perturbing the design variables about the nominal feasible design using the well known forward difference formulas [58] for first and second order derivatives of an unknown function. The approximate feasibility region is then the region of the design space for which all of the Taylor series approximated constraints evaluate to a positive real number, just as in the example illustrated in Figure 7-3. Specific results for two op-amp topologies are presented in Section 7.3.1 and Section 7.3.2.

There are several implementation details for this methodology that should be discussed in the setting of analog feasibility modeling. First a nominal feasible design point must be found. This was accomplished in an automatic fashion by performing a DC SPICE simulation for random points.
in the box-constrained design space. In subsequent experiments in this thesis, 5000 pseudo-random sample points are used for two purposes, 1) to find a good nominal point around which to construct a power series classifier, 2) to construct a nearest neighbor feasibility classifier (as discussed in the next section). The nominal feasible design used for the power series classifier was taken as the center of gravity of all feasible design points in the initial 5000 point data set. If the feasible design is reasonably large, a much smaller set could be used to find the nominal feasible design point. Since there were 5000 points available for the nearest neighbor classifier, averaging all feasible points (i.e. finding the middle, the center of gravity) seems a good choice for a nominal point. As stated previously, the accuracy of a power series classifier is dependant on the linearization point, however it is not straightforward how to go about finding the best point around which to approximate the constraints. This limits the potential accuracy of this methodology, and calls into question its overall applicability. Experimental evidence later in this chapter indicates that other classifiers (such as the nearest neighbor classifier) can outperform the power series method for feasibility region approximation.

The Hessian and gradient information of the voltage space inequality constraints must be calculated using finite differences. An appropriate step size must be chosen that gives accurate estimates of the first and second order derivatives. The step size was manually decreased until very little change was noticed in the derivative information. The step chosen for experiments in this thesis was 0.001*abs(d), where d is a design variable of the circuit.

7.2.2 Approximation of the Feasible Design Space Via Nearest Neighbor Search

Nearest neighbor search is a straightforward way to classify a queried point in a multi-dimensional space. It requires a predefined set of classified test points and a metric for distance in the multi-dimensional space such as the Euclidean norm. The distance from all test points to the query point is calculated and the query point is assigned the same classification as the closest point in the test set. Nearest neighbor search is simple to implement and computationally efficient as long as the number of test points is not too large. The data sets in this thesis are fairly small (~5000 points) making the penalty for exhaustive nearest neighbor search insignificant w.r.t. SPICE simulation times.

In order to apply nearest neighbor search to feasibility region approximation, the design space of an analog circuit is randomly sampled and all voltage space inequality constraints posed by the sizing rules method are evaluated for each point. If all constraints are satisfied for a point it
is classified as feasible, otherwise it is classified an infeasible design. The data set used to classify
unknown queried points will be referred to as the feasibility classifier data set. The accuracy of the
nearest neighbor classifier is sensitive to the number of points in the test set and is expected to
improve with an increasing number of test points.

7.2.3 Implementation of the Sizing Rules Method

The sizing rules method, as described in the paper by Gräb [38], was partially implemented
in Matlab 7 for this thesis. In the paper the authors describe 30 design space and electrical space
constraints applicable to a hierarchically defined library of analog building blocks, of which all but
two constraints were included in the implementation in this thesis. The first constraint removed is
the “Robust Geometric” (RG) constraint for a “Voltage Controlled Current Source” component
\( w \cdot I \geq A_{\text{min}} \). It is not necessary to explicitly implement this minimum area constraint since it is
implicitly enforced by the user defined box constraints required for the modeling methodology
developed herein. The second constraint removed from the list was the “Functional Geometric”
(FG) constraint for a “Simple Current Mirror” \( K = w_2/w_1 \). This constraint requires that the ratio
of transistor sizes in a two transistor current mirror be fixed to a constant number \( K \), which in turn
approximately fixes the ratio of currents through the two transistors to a constant. Removing this
constraint from the sizing rules library allows the ratio of current mirror transistors to be a free vari-
able and results in a richer analog design space. However, removing this constraint does cause a
problem for current mirror transistors that drive a differential pair, such as M3 and M4 in the two-
stage op-amp in Figure 7-4. These two transistors should always be matched since the current
through the differential pair transistors M1 and M2 should be equal. Care must be taken to manually
add an equality constraint such that the width of M3 equals the width of M4. All other sizing rules
constraints were implemented as described in [38].

We note that a sizing rules equality constraint reduces the dimension of the design space by
the elimination of one variable. Inequality constraints containing only design variables explicitly
reduce the size of the feasible region, while inequality constraints containing only voltage space
variables implicitly reduce the size of the feasible region. The only way to test whether a design is
inside the feasible region is to run a SPICE DC analysis and evaluate all inequality constraints.
7.3 Application of the Sizing Rules Method and Feasibility Region Modeling to the Single-Stage Op-amp

The single-stage op-amp in Figure 7-4 will be cast into two modeling experiments in this section differing in the rigor of the constraints enforced on the design variables. In the first experiment basic sizing rules are enforced resulting in a configuration with 9 free variables. We will show that sizing rules alone are not enough to ensure that a circuit can be modeled and that additional constraints on the design variables must be imposed in order to achieve good modeling accuracy of performance metrics. In the second experiment, hand derived constraints from previous chapters are applied as well as automatically generated sizing rules resulting in 5 free variables. Modeling accuracy of the performance metrics is greatly increased by more strictly constraining the circuit. While the automatically generated sizing rules prove useful for avoiding pathological designs, they do not significantly aid in increasing the modeling accuracy when the circuit is constrained to 5 free variables.

7.3.1 Experiment 1: Nine Free Variables

Experimental Setup.

For this experiment, sizing rules were automatically generated for the single-stage op-amp. An additional constraint was added to repair a deficiency imparted on the sizing rules method by the removal of the functional geometric constraint for the simple current mirror sub-circuit as described in Section 7.2.3. Specifically an equality constraint between the width of transistors M3 and M4 was added so that the currents through the differential pair transistors are symmetric. The width and length of transistor M6 was also fixed to 2µm since the only role of this transistor is to mirror a reference current to transistors M5 and M7. In addition, the bias current was fixed to 1µA and the load capacitance was fixed to 0.1pF. The sizing rules method failed to recognize transistor M8 as a “voltage controlled current source”, however this transistor should operate in saturation mode, thus “vccs” constraints were manually added. Even with these additional simplifications it is
shown that the performance metrics of the circuit are very difficult to model, even strictly inside the feasible region as defined by the sizing rules method.

The purpose of this experiment is to answer the following questions:

1) Does the sizing rules method sufficiently reduce the number of free variables in a circuit to facilitate macro-modeling of analog performance metrics?
2) Is there a significant decrease in the complexity of performance parameter behavior when a circuit is modeled inside the feasible region as defined by the sizing rule’s design space and voltage space constraints?
3) Does the sizing rules method avoid pathological designs?

**Data Acquisition.**

In order to answer these questions performance parameter macro-models were constructed over the box-constrained domain, approximate feasible region, and the true feasible region. Model quality is then compared over the different regions. Using the component ranges defined in Table 7-2, three data sets were generated in the 9-dimensional hyper-cube domain for the purposes of model generation, model validation, and approximation of the feasible region via nearest neighbor search.

The model generation data set consists of 5000 random points from the Halton generator, while the model validation and feasible region approximation data sets are each 5000 pseudo-random points with unique seeds used for the pseudo-random generator, i.e. they are non-intersecting. All performance metrics and DC bias information was simulated for the model generation and model validation point sets. Only DC bias information was simulated for the feasible region approximation data set since this is the only information needed to detect a feasible region. SPICE failed to converge during DC analysis for a subset of the simulated points. Since we cannot judge whether a point that fails DC analysis is inside or outside of the feasible region, these points were removed from the data sets resulting in 4657 model generation points, 4692 model validation points, and 4671 feasible region approximation points. Halton random points will be used exclusively for model generation data sets in this and subsequent

<table>
<thead>
<tr>
<th>Circuit Variables</th>
<th>Dependant Device Sizes</th>
<th>Constraint Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1_W = [2µm, 200µm]</td>
<td>Free Variable</td>
<td></td>
</tr>
<tr>
<td>M1_L = [2µm, 200µm]</td>
<td>Free Variable</td>
<td></td>
</tr>
<tr>
<td>M2_W = M1_W</td>
<td>Sizing Rule</td>
<td></td>
</tr>
<tr>
<td>M2_L = M1_L</td>
<td>Sizing Rule</td>
<td></td>
</tr>
<tr>
<td>M3_W = [2µm, 200µm]</td>
<td>Free Variable</td>
<td></td>
</tr>
<tr>
<td>M3_L = [2µm, 200µm]</td>
<td>Free Variable</td>
<td></td>
</tr>
<tr>
<td>M4_W = M3_W</td>
<td>Custom</td>
<td></td>
</tr>
<tr>
<td>M4_L = M3_L</td>
<td>Sizing Rule</td>
<td></td>
</tr>
<tr>
<td>M5_W = [2µm, 200µm]</td>
<td>Free Variable</td>
<td></td>
</tr>
<tr>
<td>M5_L = M6_L</td>
<td>Sizing Rule</td>
<td></td>
</tr>
<tr>
<td>M6_W = 2µm</td>
<td>Custom</td>
<td></td>
</tr>
<tr>
<td>M6_L = 2µm</td>
<td>Custom</td>
<td></td>
</tr>
<tr>
<td>M7_W = [2µm, 200µm]</td>
<td>Free Variable</td>
<td></td>
</tr>
<tr>
<td>M7_L = M6_L</td>
<td>Sizing Rule</td>
<td></td>
</tr>
<tr>
<td>M8_W = [2µm, 200µm]</td>
<td>Free Variable</td>
<td></td>
</tr>
<tr>
<td>M8_L = [2µm, 200µm]</td>
<td>Free Variable</td>
<td></td>
</tr>
<tr>
<td>Cc = [0.1pF, 1pF]</td>
<td>Free Variable</td>
<td></td>
</tr>
<tr>
<td>C = 0.1pF</td>
<td>Custom</td>
<td></td>
</tr>
<tr>
<td>Ibias = 1µA</td>
<td>Custom</td>
<td></td>
</tr>
</tbody>
</table>

1. ngspice version 14 was used for all simulations in this thesis as discussed in Chapter 4.
sections since Duchon pseudo-cubic splines will be employed as a modeling methodology. Section 6.2 explains why a Halton random data set is a good choice when utilizing pseudo-cubic spline models.

We now have training and validation data sets for the box-constrained domain as well as an independent set of data which can be used to approximate the feasible region. We also know exactly which points in all data sets are inside and outside the feasible region. A feasibility classifier was constructed using the feasibility approximation data set of 4671 points using both the power series and nearest neighbor classifier methods outlined in Section 7.2.1 and Section 7.2.2. The accuracy of these classifier methods is outlined in Table 7-4. It is apparent that the power series classifier does a very poor job of detecting the feasible region using both first and second order approximations. One may even be inclined to suspect software bugs in the author’s implementation of the power series classifier based on these numbers. This inclination will be absolved in future experiments in Section 7.3.2 and Section 7.4 for which the power series classifier fairs much better. The inherent inaccuracy of the power series feasibility classifier could be attributed to various possibilities. First, the accuracy is affected by the design point about which the constraints are approximated. A different point of linearization (quadraticization) may give a better approximation of the feasible region, however, there is no straightforward methodology for choosing a better point. Second, every constraint that is approximated has the potential to ruin the approximation by over constraining the feasible region. Once an approximated constraint trims some of the feasible region, this region can never be recovered. This is supported by the fact that the feasible region approximated by the power series method is very small. All experiments conducted imply that nearest neighbor feasibility region modeling is superior to power series feasibility region modeling. For this reason the nearest neighbor methodology was used to construct feasibility classifiers for all experiments.

The nearest neighbor feasibility classifier, utilizing the independent feasibility approximation data set of 4671 points, was used to find all points in the box-constrained training and validation data sets inside the approximate feasible region. After classification, training and validation sets of 2080 and 2125 points inside the approximate feasibility remain. The density of these data points inside the approximated feasible region is about the same as the data over the box-constrained domain. In addition, training and validation data sets over the true feasible region were constructed by simply filtering all infeasible designs from the box-constrained training and validation sets. The resulting truely feasible training and validation sets contain 1993 points and 1997 points respectively. Figure 7-5 illustrates the filtration of infeasible data using the sizing rules method. The plots
show the box-constrained pseudo-random point set and the corresponding feasible data points for the six performance metrics.

**Modeling Results.**

We now have three sets of training and validation data, i.e. the box-constrained, approximately feasible, and the truly feasible data sets. Duchon pseudo-cubic based performance macro-models were constructed from this data in order to compare the modeling accuracy in the three regions. The number of sample points used to construct the models was progressively increased in order to show the dynamic learning rate of the models with increasing sample size. It is unfair to compare the accuracy of models constructed over differing volumes of the hyper-cube domain since it should be inherently easier to model regions of decreasing size. For this reason a scale factor was used to penalize the modeling accuracies in the approximate and true feasible regions. Since the density of points in all three data sets is approximately the same (they are all random in nature), it is appropriate to scale the actual number of sample points used to build a model by the relative size of the region modeled. For example, the expected size of the approximate feasible region is the number of points detected to reside in this region divided by the number of random samples in the hyper-cube domain, i.e. 2080/4657 = 0.446 for the training set and 2125/4692 = 0.452 for the validation set. These two numbers are estimates of the relative size of the approximate feasible region based on two random and independent data sets and should therefore be consistent, which is indeed the case. Rather than “down-scale” the number of points in the hyper-cube domain, the number of sample points used to build the models in the approximate and true feasible regions were “up-scaled” by multiplication of the inverse of the expected size of

**Table 7-3**

<table>
<thead>
<tr>
<th>Sizing Component</th>
<th>Instanced Transistors</th>
<th>Level of Sizing Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>vccs M1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>vccs M2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>vccs M3</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>vccs M4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>vccs M5</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>vccs M6</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>vccs M7</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>vccs M8</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>cm M3, M4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>cm M6, M5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>cm M6, M7</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>dp M1, M2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>CMB M6, M5, M7</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Note: Transistor M8 was manually added to the sizing rules list since it was not automatically recognized as a “vccs”.

**Table 7-4**

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Input Transformation</th>
<th>Ns</th>
<th>CP</th>
<th>CN</th>
<th>FP</th>
<th>FN</th>
<th>( \Phi_1 (\hat{p}) )</th>
<th>( \Phi_2 (\hat{p}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN Linear</td>
<td>4657</td>
<td>1546</td>
<td>2107</td>
<td>557</td>
<td>447</td>
<td>0.776 (0.452)</td>
<td>0.735 (0.428)</td>
<td></td>
</tr>
<tr>
<td>NN Logarithmic</td>
<td>4657</td>
<td>1527</td>
<td>2120</td>
<td>544</td>
<td>466</td>
<td>0.766 (0.445)</td>
<td>0.737 (0.428)</td>
<td></td>
</tr>
<tr>
<td>PS (1\textsuperscript{st}) Linear</td>
<td>4657</td>
<td>2</td>
<td>2553</td>
<td>111</td>
<td>1991</td>
<td>0.001 (0.024)</td>
<td>0.018 (0.428)</td>
<td></td>
</tr>
<tr>
<td>PS (1\textsuperscript{st}) Logarithmic</td>
<td>4657</td>
<td>1</td>
<td>2513</td>
<td>151</td>
<td>1992</td>
<td>0.0005 (0.032)</td>
<td>0.007 (0.428)</td>
<td></td>
</tr>
<tr>
<td>PS (2\textsuperscript{nd}) Linear</td>
<td>4657</td>
<td>1</td>
<td>2661</td>
<td>3</td>
<td>1992</td>
<td>0.0005 (0.001)</td>
<td>0.25 (0.428)</td>
<td></td>
</tr>
<tr>
<td>PS (2\textsuperscript{nd}) Logarithmic</td>
<td>4657</td>
<td>1</td>
<td>2550</td>
<td>114</td>
<td>1992</td>
<td>0.0005 (0.025)</td>
<td>0.008 (0.428)</td>
<td></td>
</tr>
</tbody>
</table>

Nearest Neighbor Classifier (NN), First and Second Order Power Series Classifier (PS), # Samples (Ns), # Correct Positives (CP), # Correct Negatives (CN), # False Positives (FP), # False Negatives (FN), Classifier Accuracy \( \Phi_1, \Phi_2 \), Random Classifier Accuracy \( \hat{p}, p \).

101
the regions. In other words, 100 real sample points in the approximate feasible region is scaled by 1/0.45 = 2.22, appearing as 222 sample points. This scaling methodology equilibrates the models based on sample point density rather than absolute number of sample points. A similar computation was performed for the true feasible region.

Modeling results using Duchon pseudo-cubic splines for the 9-dimensional single stage op-amp over the box-constrained, approximate, and true feasible regions are shown in Figure 7-6. The upper family of curves in each plot is the maximum absolute modeling error and the lower family is the standard deviation of the signed errors. Indeed the modeling accuracy is improved for a majority of the performance metrics when constrained to the true feasible region. Most notably, the maximum modeling error for open-loop gain has been greatly reduced to a level below even the error standard deviation of the box-constrained and approximate feasible regions. However, even with the improvements in modeling accuracy imposed by the sizing rules constraints, the overall modeling accuracy is rather poor. Figure 7-7 further illuminates the inaccuracy of the performance models when restricted to the true feasible region. In these plots, an ideal model would yield a perfectly straight line from the upper right hand to the lower left hand corners of the plot (see Figure 7-13). Surely open loop gain is a usable model when constrained to the feasible region, but the other five models are grossly inaccurate. So far, all modeling examples in this thesis have been restricted to about 5 free variables. Here were are attempting to model functions with 8-dimensional (A0, CMRR, PSRRp, PSRRn) and 9-dimensional (UGF, PM) domains. One could argue that the Duchon splines are failing to appropriately model the higher dimensional functions, however, this doesn’t seem to be the case for open loop gain. The pseudo-cubic splines are doing an adequate job of modeling this performance parameter. This implies that open loop gain is well-behaved inside the feasible region, while the behavior of the other metrics is much more complex.

Another interesting result is that the error in the models over the box-constrained and approximate feasible regions is nearly the same. In other words, restricting the models to an approximate feasibility region does not significantly improve the quality of the models. But recall that the whole premise of analog performance macro-modeling is to completely eliminate the need to call a SPICE simulator for performance estimation during circuit synthesis. In order to test whether a design point is in the true feasible region SPICE must be called, otherwise we must rely on an approximation of the feasible region. To sum things up, this is a pretty grim example for macro-

1. Recall that the performance metrics A0, CMRR, and PSRRx are measured at low frequency and not dependent on capacitive variables while UGF and PM are measured at high frequencies, thus increasing the dimensionality through a dependence on the coupling capacitor Cc.
Figure 7-5. Filtration of infeasible points using the sizing rules method. Six performance metrics were sampled for the 9-dimensional single-stage op-amp configuration using 5000 pseudo-random box-constrained points (circles). 308 points suffered DC convergence problems. The remaining 4692 points were filtered using the sizing rules method resulting in 1997 points (x’s) inside the feasible domain.
Figure 7-6. Modeling accuracy of performance metrics for the 9-dimensional single-stage op-amp over the box-constrained domain (circles), nearest neighbor approximate feasible region (x’s), and true feasible region (triangles) using Duchon pseudo-cubic splines. The number of samples have been scaled by the size of the feasible region for the approximate and true feasible regions to reflect a similar density of points over all three domains. The upper family of curves is maximum absolute error, the bottom curves the signed standard deviation of the error. Models were created using Halton random samples and validated with pseudo-random samples.
Figure 7-7. Modeling accuracy for the 9-dimensional single-stage op-amp inside the feasible region as defined by the sizing rules method. The training data set and validation set consist of 1993 Halton random and 1997 pseudo-random points respectively. The performance metrics are modeled using Duchon pseudo-cubic splines.
modeling. While the sizing rules method did in fact define a feasible region for which the behavior of the performance metrics was simplified, the behavior was still too complex to model even when constricted solely to the true feasible region. The sizing rules method eliminated almost half of free variables in the circuit, but this is apparently not sufficient for the purpose of macro-modeling even for a small circuit example such as the single-stage op-amp. The feasibility region approximator performs marginally and does not provide a significant increase in accuracy for models restricted to the approximate feasible region. However, in the next example we will see that constraining the circuit design parameters beyond that accomplished by the sizing rules method results in much improved modeling accuracy and a more usable feasibility approximator.

7.3.2 Experiment 2 - Five Free Variables

Experimental Setup.

For this experiment, the design variables of the single-stage op-amp were similarly configured to that of previous experiments in this thesis. The reasoning behind the choices of design variable constraints will now be briefly outlined. It is common practice for an analog designer to fix the length of all transistors required to operate in saturation mode to a nominal minimum length [2]. This immediately eliminates nearly half of the free design parameters since all transistors in the single-stage op-amp should operate in saturation. The size of transistor M1 should equal M2, and the size of M3 should equal M4 to equalize the currents through the differential pair. Both M1=M2 and M3=M4 are left as free parameters. Transistor M6 can be fixed to some minimum nominal size since its job is to simply mirror the reference current Ibias, which can also be fixed. The width of transistors M5 and M7 control the current through the differential pair and output stage respectively and are also left as free parameters.

In order to minimize the DC offset voltage at the output node, transistor M8 should equal 2*M3*M7/M5. This is because the current through M4 = 0.5*Ibias*M5/M6. If M4 is acting as a current mirror with M3 the drain voltage of M4 must be equal to the drain/gate voltage of M3 since

<table>
<thead>
<tr>
<th>Circuit Variables</th>
<th>Dependant Device Sizes</th>
<th>Constraint Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1_W = [2µm, 200µm]</td>
<td>Free Variable</td>
<td></td>
</tr>
<tr>
<td>M1_L = 2µm</td>
<td>Custom</td>
<td></td>
</tr>
<tr>
<td>M2_W = M1_W</td>
<td>Sizing Rule</td>
<td></td>
</tr>
<tr>
<td>M2_L = M1_L</td>
<td>Sizing Rule</td>
<td></td>
</tr>
<tr>
<td>M3_W = [2µm, 200µm]</td>
<td>Free Variable</td>
<td></td>
</tr>
<tr>
<td>M3_L = 2µm</td>
<td>Custom</td>
<td></td>
</tr>
<tr>
<td>M4_W = M3_W</td>
<td>Custom</td>
<td></td>
</tr>
<tr>
<td>M4_L = M3_L</td>
<td>Sizing Rule</td>
<td></td>
</tr>
<tr>
<td>M5_W = [2µm, 200µm]</td>
<td>Free Variable</td>
<td></td>
</tr>
<tr>
<td>M5_L = M6_L</td>
<td>Sizing Rule</td>
<td></td>
</tr>
<tr>
<td>M6_W = 2µm</td>
<td>Custom</td>
<td></td>
</tr>
<tr>
<td>M6_L = 2µm</td>
<td>Custom</td>
<td></td>
</tr>
<tr>
<td>M7_W = [2µm, 200µm]</td>
<td>Free Variable</td>
<td></td>
</tr>
<tr>
<td>M7_L = M6_L</td>
<td>Sizing Rule</td>
<td></td>
</tr>
<tr>
<td>M8_W = 2<em>M3</em>M7/M5</td>
<td>Custom</td>
<td></td>
</tr>
<tr>
<td>M8_L = 2µm</td>
<td>Custom</td>
<td></td>
</tr>
<tr>
<td>Cc = [0.1pF, 1pF]</td>
<td>Free Variable</td>
<td></td>
</tr>
<tr>
<td>Cl = 0.1pF</td>
<td>Custom</td>
<td></td>
</tr>
<tr>
<td>Ibias = 1µA</td>
<td>Custom</td>
<td></td>
</tr>
</tbody>
</table>
the transistors are same size, have equal drain currents, and have the same gate to source voltages. Thus the gate voltage of M8 is equal to the drain voltage of M4, which is equal to the drain/gate voltage of M3. This causes M8 to mirror the current through transistors M3 and M4 by the ratio M8/M3. Putting this all together we have the current through M8 = 0.5*(Ibias*M5/M6)*M8/M3 and the current through M7 = Ibias*M7/M6. Equating the currents through M8 and M7 yields the necessary size of M8 = 2*M3*M7/M5. Lastly the compensation capacitor is left as a free variable since it controls the inherent stability of the op-amp. The load capacitor may be taken as a free or fixed variable, but usually the load a circuit must drive is known, so it is a safe assumption to fix the load capacitance in the interest of simplifying the modeling problem. The above arguments result in the 5-dimensional parametric configuration in Table 7-5 for the single-stage op-amp.

Data Acquisition.

Similar to the previous experiment in Section 7.3.1, we wish to compare the accuracy of performance parameter models in the box-constrained, approximate feasible, and true feasible design regions. Again three data sets were generated, this time in the 5-dimensional hyper-cube domain as defined by Table 7-5. The model generation data set consists of 5000 random points from the Halton generator, while the model validation and feasible region approximation data sets are each 5000 pseudo-random points. Using a nearest neighbor feasibility region approximator, an additional 5000 Halton series and 5000 pseudo-random points were generated inside the approximate feasible region. This can be accomplished by sequentially generating random points in the box-constrained domain and rejecting those points that are not classified as feasible by the nearest neighbor feasible region approximator. For the last data set, the infeasible points in the box-constrained Halton and pseudo-random sets were filtered to produce training and validation sets of 3403 points and 3398 points respectively. Thus the expected size of the true feasible region is ~3400/5000 = 68% of the box-constrained domain. The expected size of the approximate feasible region as defined by the nearest neighbor classifier is 67% of the box-constrained domain. The accuracy of the nearest neighbor and power series feasibility classifiers are summarized in Table 7-6. The power series classifier fared much better for this example as compared to the example in Section 7.3.1, but is still not as accurate as the nearest neighbor classifier.

Modeling Results.

Using the box-constrained, approximately feasible and truly feasible data sets described in the previous paragraph, Duchon pseudo-cubic models were constructed for varying sample sizes of the Halton data sets and error measurements taken using the full set of pseudo-random validation
points. The results are summarized in Figure 7-8. Again the sample sizes for the approximate and true feasible regions have been “up-scaled” by the respective expected sizes of these regions to reflect sample point density as described in Section 7.3.1. The overall modeling accuracies for the performance metrics are greatly improved over the previous 9-dimensional modeling example. However, except for CMRR, modeling accuracy in the true feasible region is not significantly improved over the box-constrained and approximate feasible regions. For this example sizing rules constraints as well as an additional set of custom design constraints were enforced. These additional constraints, as described previously, are based on a manual analysis of the circuit and cannot be generated using the sizing rules method. For the purposes of macro-modeling, these constraints are necessary to reduce the dimensionality of the input space as well as the complexity of the behavior of the performance metrics. The accuracy of the nearest neighbor feasibility classifier is also greatly improved. The percentage of the true feasible design space that is “chopped off” by the NN feasibility classifier is only \( (1 - \Phi_1) = 4.8\% \) while only \( (1 - \Phi_2) = 5.8\% \) of the NN approximate feasible region is incorrectly infeasible. While the automatically generated sizing rules don’t provide a significant improvement in modeling accuracy, they do properly define the feasible region and provide a method for ensuring the enforcement of voltage space constraints. The nearest neighbor feasible region approximator eliminates the need to run a DC SPICE simulation to check the feasibility of a point in the design space in a synthesis setting. Of course it is still a good idea to use SPICE to ensure that a design synthesized with performance and feasibility region models meets the performance criterion and is inside the true feasible region.

Referring to the accuracy plots of the performance metrics in Figure 7-9, the modeling accuracy of PSRRp appears to be nearly as bad as in the 9-dimensional modeling example. The exact source of error in this model is not known exactly, but possible culprits are discussed in Section 7.5.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Input Transformation</th>
<th>Ns</th>
<th>CP</th>
<th>CN</th>
<th>FP</th>
<th>FN</th>
<th>( \Phi_1 ) (( \hat{p} ))</th>
<th>( \Phi_2 ) (( p ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN Linear</td>
<td>5000</td>
<td>3239</td>
<td>1396</td>
<td>201</td>
<td>164</td>
<td>0.952 (0.688)</td>
<td>0.942 (0.680)</td>
<td></td>
</tr>
<tr>
<td>NN Logarithmic</td>
<td>5000</td>
<td>3264</td>
<td>1415</td>
<td>182</td>
<td>139</td>
<td>0.959 (0.689)</td>
<td>0.947 (0.680)</td>
<td></td>
</tr>
<tr>
<td>PS (1st) Linear</td>
<td>5000</td>
<td>2367</td>
<td>918</td>
<td>684</td>
<td>1031</td>
<td>0.697 (0.610)</td>
<td>0.776 (0.680)</td>
<td></td>
</tr>
<tr>
<td>PS (1st) Logarithmic</td>
<td>5000</td>
<td>2509</td>
<td>1088</td>
<td>514</td>
<td>889</td>
<td>0.738 (0.605)</td>
<td>0.830 (0.680)</td>
<td></td>
</tr>
<tr>
<td>PS (2nd) Linear</td>
<td>5000</td>
<td>182</td>
<td>1558</td>
<td>44</td>
<td>3216</td>
<td>0.054 (0.045)</td>
<td>0.805 (0.680)</td>
<td></td>
</tr>
<tr>
<td>PS (2nd) Logarithmic</td>
<td>5000</td>
<td>3069</td>
<td>392</td>
<td>1210</td>
<td>329</td>
<td>0.903 (0.856)</td>
<td>0.717 (0.680)</td>
<td></td>
</tr>
</tbody>
</table>

Nearest Neighbor Classifier (NN), First and Second Order Power Series Classifier (PS), # Samples (Ns), # Correct Positives (CP), # Correct Negatives (CN), # False Positives (FP), # False Negatives (FN), Classifier Accuracy (\( \Phi_1 \), \( \Phi_2 \)), Random Classifier Accuracy (\( \hat{p} \), \( p \)).
Figure 7-8. Modeling accuracy of Duchon pseudo-cubic spline models using Halton random samples over the box-constrained (circles), nearest neighbor approximate feasible region (x’s), and true feasible region (triangles) for the 5-dimensional single-stage op-amp. The number of samples have been scaled by the size of the feasible region for the approximate and true feasible regions to reflect a similar density of points over all three domains. The upper family of curves is maximum absolute error, the bottom curves the signed standard deviation of the error. Models were created using Halton random samples and validated with pseudo-random samples.
Figure 7-9. Modeling accuracy for 5-dimensional two-stage op-amp using Duchon pseudo-cubic spline models in the nearest neighbor feasible domain. Training set is 5000 halton random points, validation set is 5000 pseudo-random points.
Figure 7-10. Comparison of neural network (star), pseudo-cubic spline (circle), and adaptively sampled pseudo-cubic spline (triangle) modeling accuracy for six performance metrics of the two-stage operational amplifier. The function domain is constrained to the approximate feasible region defined by the nearest neighbor feasibility classifier.
Since we are interested in the accuracy of performance macro-models constrained to the approximate feasible region as defined by nearest neighbor search, additional modeling results were obtained in this region using neural network [87] and adaptively sampled pseudo-cubic spline models [88], enabling a direct comparison of the relative accuracy of various modeling methodologies. These results are summarized Figure 7-10. The plots show the accuracy of pseudo-cubic spline, neural network, and adaptively sampled pseudo-cubic spline models in the approximate feasibility region. The same approximate feasible region data that was used for the models in Figure 7-8 was also used for the pseudo-cubic spline and neural network models. Once again the 5000 Halton random data points was used as the training data set while the 5000 pseudo-random points were used as the validation set. In fact the pseudo-cubic spline models for the approximate feasible region are the same models used in Figure 7-8. The plots differ slightly in appearance because there is no need to “up-scale” the number of sample points since the modeling region is the same size for all three modeling methodologies, and thus the density of sample points in the approximate feasible region is equivalent.

The neural network models were created using the Matlab 7 neural network toolbox. All neural models are standard two-layer feed-forward networks with 30 hidden neurons and between 4 and 5 input neurons depending on the performance parameter modeled. Rather than tune the size of each network to maximize modeling accuracy, a training algorithm capable of automatically adjusting the number of free parameters in the neural networks was used. Bayesian regulation is such a training algorithm and is available in the Matlab neural network toolbox [18]. Each network was trained using the ‘trainbr’ (Bayesian regulation) option for a fixed number of 300 epochs. This training method proved effective for developing neural models with good accuracy with no human interaction.

The adaptive pseudo-cubic spline models were obtained by generating sequential pseudo-random numbers in the box-constrained domain and allowing only the points accepted by the nearest neighbor feasibility classifier to be considered for sampling. The free parameters that govern the adaptive sampler, i.e. the number of neighbors in the local spline \(k\), the number of candidate points \(n_c\), and the number of points used in the infinity norm approximation \(n_v\) were set to the same values as in [88]. The adaptive sampler was seeded with 100 pseudo-random points and allowed to generate 4900 additional sample points in the approximate feasible region. The models were validated with the same pseudo-random data set used to validate the non-adaptive and neural network models.
Enhanced neural model accuracy could no doubt be obtained by manually tuning the size of the networks, but even with the automated approach to training the networks, the neural models usually managed to achieve superior accuracy over even the adaptively sampled models. Neural networks are very compact models and are extremely computationally efficient. However, it is commonly known that the training methods used to create neural models may produce sub-optimal models due to the existence of local minimums in the error minimization process. This is experimentally evident for the results in Figure 7-10. The overall accuracy of the neural models is erratic from one sample point cardinality to another. This is because there is no guarantee on model accuracy and the training process may lead to a sub-optimal fit of the data based on the initial conditions of the free parameters in the neural network model.

### 7.4 Additional Modeling Results for the Cascode Op-amp

In this section the sizing rules method is applied to the cascode op-amp in Figure 7-11. In Section 7.3 results indicated that additional constraints on the design variables beyond those imposed by the sizing rules are required to obtain good modeling accuracy of performance metrics. For this reason the circuit design variables were constrained based on the configuration previously outlined in Chapter 5. These constraints are given in Table 7-8.

![Figure 7-11. Cascode op-amp circuit.](image)

Again we wish to explore the potential of the sizing rules method for simplifying performance parameter behavior in the approximate and true feasible regions. The experimental setup here is very similar to that of the 9-variable single stage op-amp in Section 7.3.1 Three sets of data were generated in the box-constrained domain, i.e. 5000 Halton points for model generation, 5000 pseudo-random points for model validation, and 5000 pseudo-random points feasibility region approximation. After filtering all points that failed DC convergence in SPICE, the cardinality of the Halton training, pseudo-random validation, and pseudo-random feasibility classifier data were reduced to 4908, 4905, and 4902 points respectively. A nearest neighbor
classifier based on the 4902 pseudo-random points was used to extract a 2112 point training data set and a 2171 point validation set for the approximate feasible region. Lastly, the points in the true feasible region were used to construct a training set of 2140 points and validation 2166 points.

This data was used to construct Duchon pseudo-cubic splines for the box-constrained, approximate feasible, and true feasible regions. The number of points used to build the models was systematically varied to explore the improvement in performance macro-model accuracies with increasing sample size. The models were validated with the full set of pseudo-cubic points for each appropriate modeling region. These modeling results are shown in Figure 7-12. Again the sample sizes for the approximate and true feasible regions were scaled by one minus the expected size of the regions so that the density of sample points in these areas corresponds to the density of sample points in the box-constrained domain. Once again the accuracy of the performance models is only marginally, if at all, improved when the data points are restricted to the approximate and true feasible regions. From Figure 7-13 it is apparent that modeling accuracy for all performance metrics is very good, with the exception being CMRR. This deficiency in model accuracy is for CMRR is discussed in section Section 7.5. The accuracy of the nearest neighbor feasibility approximator is on par with that of the 5-dimensional single-stage op-amp.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Input Transformation</th>
<th>Ns</th>
<th>CP</th>
<th>CN</th>
<th>FP</th>
<th>FN</th>
<th>( \Phi_1 ) ( \hat{p} )</th>
<th>( \Phi_2 ) ( p )</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN</td>
<td>Linear</td>
<td>4908</td>
<td>1981</td>
<td>2617</td>
<td>151</td>
<td>159</td>
<td>0.926 (0.434)</td>
<td>0.929 (0.436)</td>
</tr>
<tr>
<td>NN</td>
<td>Logarithmic</td>
<td>4908</td>
<td>1997</td>
<td>2612</td>
<td>156</td>
<td>143</td>
<td>0.933 (0.439)</td>
<td>0.928 (0.436)</td>
</tr>
<tr>
<td>PS (1st)</td>
<td>Linear</td>
<td>4902</td>
<td>718</td>
<td>2584</td>
<td>119</td>
<td>1481</td>
<td>0.326 (0.171)</td>
<td>0.858 (0.436)</td>
</tr>
<tr>
<td>PS (1st)</td>
<td>Logarithmic</td>
<td>4902</td>
<td>705</td>
<td>2594</td>
<td>109</td>
<td>1494</td>
<td>0.321 (0.166)</td>
<td>0.866 (0.436)</td>
</tr>
<tr>
<td>PS (2nd)</td>
<td>Linear</td>
<td>4902</td>
<td>35</td>
<td>2693</td>
<td>10</td>
<td>2164</td>
<td>0.016 (0.009)</td>
<td>0.778 (0.436)</td>
</tr>
<tr>
<td>PS (2nd)</td>
<td>Logarithmic</td>
<td>4902</td>
<td>903</td>
<td>2563</td>
<td>140</td>
<td>1296</td>
<td>0.411 (0.213)</td>
<td>0.866 (0.436)</td>
</tr>
</tbody>
</table>

Nearest Neighbor Classifier (NN), First and Second Order Power Series Classifier (PS), # Samples (Ns), # Correct Positives (CP), # Correct Negatives (CN), # False Positives (FP), # False Negatives (FN), Classifier Accuracy (\( \Phi_1 \), \( \Phi_2 \)), Random Classifier Accuracy \( \hat{p} , p \).
### Table 1-8: Component ranges for 5-variable cascode op-amp circuit

<table>
<thead>
<tr>
<th>Circuit Variables</th>
<th>Dependant Device Sizes</th>
<th>Constraint Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1_W</td>
<td>[1(\mu)m, 100(\mu)m]</td>
<td>Free Variable</td>
</tr>
<tr>
<td>M1_L</td>
<td>1(\mu)m</td>
<td>Custom</td>
</tr>
<tr>
<td>M2_W</td>
<td>M1_W</td>
<td>Sizing Rule</td>
</tr>
<tr>
<td>M2_L</td>
<td>M1_L</td>
<td>Sizing Rule</td>
</tr>
<tr>
<td>M3_W</td>
<td>[1(\mu)m, 100(\mu)m]</td>
<td>Free Variable</td>
</tr>
<tr>
<td>M3_L</td>
<td>1(\mu)m</td>
<td>Custom</td>
</tr>
<tr>
<td>M4_W</td>
<td>M3_W</td>
<td>Custom</td>
</tr>
<tr>
<td>M4_L</td>
<td>M3_L</td>
<td>Sizing Rule</td>
</tr>
<tr>
<td>M5_W</td>
<td>[1(\mu)m, 100(\mu)m]</td>
<td>Free Variable</td>
</tr>
<tr>
<td>M5_L</td>
<td>M12_L</td>
<td>Sizing Rule</td>
</tr>
<tr>
<td>M6_W</td>
<td>M3_W</td>
<td>Custom</td>
</tr>
<tr>
<td>M6_L</td>
<td>M3_L</td>
<td>Sizing Rule</td>
</tr>
<tr>
<td>M7_W</td>
<td>M3_W</td>
<td>Custom</td>
</tr>
<tr>
<td>M7_L</td>
<td>M6_L</td>
<td>Sizing Rule</td>
</tr>
<tr>
<td>M8_W</td>
<td>M3_W</td>
<td>Custom</td>
</tr>
<tr>
<td>M8_L</td>
<td>M16_L</td>
<td>Sizing Rule</td>
</tr>
<tr>
<td>M9_W</td>
<td>M3_W</td>
<td>Custom</td>
</tr>
<tr>
<td>M9_L</td>
<td>M1_L</td>
<td>Sizing Rule</td>
</tr>
<tr>
<td>M10_W</td>
<td>10(\mu)m</td>
<td>Custom</td>
</tr>
<tr>
<td>M10_L</td>
<td>M4_L</td>
<td>Sizing Rule</td>
</tr>
<tr>
<td>M11_W</td>
<td>10(\mu)m</td>
<td>Custom</td>
</tr>
<tr>
<td>M11_L</td>
<td>1(\mu)m</td>
<td>Custom</td>
</tr>
<tr>
<td>M12_W</td>
<td>10(\mu)m</td>
<td>Custom</td>
</tr>
<tr>
<td>M12_L</td>
<td>1(\mu)m</td>
<td>Custom</td>
</tr>
<tr>
<td>M13_W</td>
<td>0.25(\mu)M3_W</td>
<td>Custom</td>
</tr>
<tr>
<td>M13_L</td>
<td>1(\mu)m</td>
<td>Custom</td>
</tr>
<tr>
<td>M14_W</td>
<td>0.5(\mu)M5_W*</td>
<td>Custom</td>
</tr>
<tr>
<td>M14_L</td>
<td>M6_W/M3_W</td>
<td>Sizing Rule</td>
</tr>
<tr>
<td>M15_W</td>
<td>0.25(\mu)M3_W</td>
<td>Custom</td>
</tr>
<tr>
<td>M15_L</td>
<td>M13_L</td>
<td>Sizing Rule</td>
</tr>
<tr>
<td>M16_W</td>
<td>0.25(\mu)M3_W</td>
<td>Custom</td>
</tr>
<tr>
<td>M16_L</td>
<td>1(\mu)m</td>
<td>Custom</td>
</tr>
<tr>
<td>CL</td>
<td>[1pF, 10pF]</td>
<td>Free Variable</td>
</tr>
<tr>
<td>Ibias</td>
<td>[2(\mu)A, 20(\mu)A]</td>
<td>Free Variable</td>
</tr>
</tbody>
</table>

### Table 1-9: Recognized sizing components for the cascode op-amp

<table>
<thead>
<tr>
<th>Sizing Component</th>
<th>Instanced Component</th>
<th>Level of Sizing Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>vccs</td>
<td>M1</td>
<td>0</td>
</tr>
<tr>
<td>vccs</td>
<td>M2</td>
<td>0</td>
</tr>
<tr>
<td>vccs</td>
<td>M3</td>
<td>0</td>
</tr>
<tr>
<td>vccs</td>
<td>M4</td>
<td>0</td>
</tr>
<tr>
<td>vccs</td>
<td>M5</td>
<td>0</td>
</tr>
<tr>
<td>vccs</td>
<td>M6</td>
<td>0</td>
</tr>
<tr>
<td>vccs</td>
<td>M7</td>
<td>0</td>
</tr>
<tr>
<td>vccs</td>
<td>M8</td>
<td>0</td>
</tr>
<tr>
<td>vccs</td>
<td>M9</td>
<td>0</td>
</tr>
<tr>
<td>vccs</td>
<td>M10</td>
<td>0</td>
</tr>
<tr>
<td>vccs</td>
<td>M11</td>
<td>0</td>
</tr>
<tr>
<td>vccs</td>
<td>M12</td>
<td>0</td>
</tr>
<tr>
<td>vccs</td>
<td>M13</td>
<td>0</td>
</tr>
<tr>
<td>vccs</td>
<td>M14</td>
<td>0</td>
</tr>
<tr>
<td>vccs</td>
<td>M15</td>
<td>0</td>
</tr>
<tr>
<td>vccs</td>
<td>M16</td>
<td>0</td>
</tr>
<tr>
<td>cm</td>
<td>M3, M6</td>
<td>1</td>
</tr>
<tr>
<td>cm</td>
<td>M4, M10</td>
<td>1</td>
</tr>
<tr>
<td>cm</td>
<td>M11, M9</td>
<td>1</td>
</tr>
<tr>
<td>cm</td>
<td>M12, M5</td>
<td>1</td>
</tr>
<tr>
<td>cm</td>
<td>M12, M14</td>
<td>1</td>
</tr>
<tr>
<td>cm</td>
<td>M13, M15</td>
<td>1</td>
</tr>
<tr>
<td>ls</td>
<td>M13, M7</td>
<td>1</td>
</tr>
<tr>
<td>ls</td>
<td>M16, M8</td>
<td>1</td>
</tr>
<tr>
<td>dp</td>
<td>M1, M2</td>
<td>1</td>
</tr>
<tr>
<td>CMB</td>
<td>M12, M5, M14</td>
<td>2</td>
</tr>
</tbody>
</table>
Figure 7-12. Comparison of modeling accuracy of over the box-constrained (circles), nearest neighbor approximate feasible region (x’s), and true feasible region (triangles) for the cascode op-amp using Duchon pseudo-cubic splines. The number of samples have been scaled by the size of the feasible region for the approximate and true feasible regions to reflect a similar density of points over all three domains. The upper family of curves is maximum absolute error, the bottom curves the signed standard deviation of the error. Models were created using Halton random samples and validated with pseudo-random samples.
Figure 7-13. Modeling accuracy in the nearest neighbor feasible region for the cascode op-amp using Duchon pseudo-cubic splines. Models were created using Halton random samples and validated with pseudo-random samples.
7.5 Anomalous Behavior of Positive Power Supply Rejection Ratio and Common Mode Rejection Ratio

The deficiencies in modeling accuracy of PSRRp for the single-stage op-amp and CMRR for the cascode op-amp are quite troubling. This section describes the steps that were taken in order to debug this problem. Though no hard evidence could be produced implicating the exact source of the problem, the process of elimination builds a strong circumstantial case implicating either instabilities in the BSIM3v3 transistor models, or some unknown caveat or bug in the SPICE\textsuperscript{1} simulator.

The sizing rules method produces a set of inequality constraints which must be satisfied for the circuit to be considered a feasible design. These constraints can always be written in the form \( c_j(d, e) \geq 0 \) where \( d \) and \( e \) are the design parameters (transistor widths, lengths, etc.) and electrical parameters (voltages and currents). The vector \( e \) is dependant on \( d \) and is found by running a SPICE DC analysis. Therefore the sizing rules constraints are really only dependant on the design parameters. When all constraints evaluate to a number greater than zero, the design point \( d \) is inside the feasible region. The boundary of the feasible region is defined by all points in the design space for which no constraints are less than zero and one or more constraints evaluate to exactly zero. Recall that these inequality constraints represent modal transitions of transistors from saturation to triode region as well as a host of other possible “bad” scenarios defined by the sizing rules method. Therefore the closer a design point is to the boundary of the feasible region, the more likely it is that the behavior of the circuits performance metrics will become erratic due the potential violation of a sizing rule constraint. Perhaps the large modeling errors in PSRRp and CMRR are due to this scenario? To test this hypothesis, the sizing rules inequality constraints for one hundred points in the

![](image)

**Figure 7-14.** Two dimensional projections of the anomalous behavior of PSRRp in a very small region of the design space. All designs in the anomalous region were confirmed to be in the feasible region defined by the sizing rules method.

---

1. ngspice version 14 was used for all simulations in this thesis as discussed in Chapter 4
design space were evaluated and examined. There proved to be no apparent relationship between the modeling error of a point and its “closeness” to the feasible region boundary for PSRRp or CMRR. This theory also does not make sense for the following reason. The modeling error for both PSRRp and CMRR is approximately the same over the box-constrained, approximate, and true feasible regions. If the modeling errors are due to transitions near the boundary of the feasible region, modeling accuracy should be worse for a model including both the interior and exterior of the feasible design space.

Another potential source of error could be a deficiency in the test bench circuit or extraction methodology of the performance data. A handful of points with large modeling errors were selected for a more thorough analysis for both PSRRp and CMRR. SPICE was interactively run for these points and the corresponding Bode plots manually examined. No problems were found to exist for the selected points. The Bode plots for all performance metrics looked normal and the automatically extracted values for PSRRp and CMRR corresponded exactly with those obtained manually. Thus the testbench circuit and post-processing of SPICE data appear not to be the problem.

SPICE is known to potentially have convergence issues when solving for the DC bias currents and voltages of a circuit [67]. Not only will SPICE simply fail to converge and report and convergence error, but it can also converge to incorrect voltages and currents without reporting a convergence error. This could potentially cause gross inaccuracies in a performance parameter model since two points in the design space that are arbitrarily close may have inconsistent biasing. This would incorrectly cause a sharp change in the value of the performance metrics. This problem can, however, be ruled out without further experimentation. The data for the six performance parameters of the cascode op-amp were created using identical sample points and a common test-bench circuit. If the error present in the PSRRp model is due to DC convergence issues, the other five performance metrics would be similarly affected at least to some degree. However, the accuracy of the performance parameter models for the other five metrics is impeccable.
Finally, the modeling errors for PSRRp and CMRR could be due to instabilities in the BSIM3v3 models. To test this hypothesis a point in the domain with a large modeling error for PSRRp was chosen for detailed analysis. This point will be referred to as the nominal design point. This point was confirmed to be well inside the feasible region. Several points around the nominal point were manually tested. It was observed that for very small perturbations of the design variables, the PSRRp values reported by SPICE varied by more than 50dB. In order to further explore this behavior, PSRRp was evaluated for the corner points of a very small hyper-cube domain around the nominal design point. We will call this region the micro-cube for lack of a better term. In addition 200 Halton random samples were evaluated inside the micro-cube in order to build a pseudo-cubic spline model of the behavior of PSRRp in the micro-cube around the nominal design point. By doing so, we have essentially zoomed in around the nominal point and built a detailed model of the region of interest. Specifically, each of the design variables was perturbed around the nominal point by a factor of one one-thousandth of the nominal design variables. The zoom factor was calculated as the volume of the original box-constrained domain divided by the volume of the micro-cube. The zoom factor was numerically calculated to be $1.1807 \times 10^{13}$. This is quite the virtual telescope to say the least. In order to visualize the behavior of PSRRp, two of the four free design variables must be fixed. The pseudo-cubic model describing the behavior of PSRRp in the micro-cube can then be sampled to build a three-dimensional response surface. Figure 7-14 shows two such cross-sections of the four-dimensional pseudo-cubic spline model for PSRRp in the micro-cube domain. Not only is the behavior erratic, but the range of the performance metric is very large for such a small region of the design space. For comparative purposes, a similar experiment was performed for PSRRn for the single-stage op-amp. Modeling accuracy for this performance metric was quite good and it is assumed to be well behaved. A cross-section of PSRRn in a micro-cube domain is shown in Figure 7-15. The behavior of PSRRn is very linear and does not span a large range of values. This is the expected response of a performance function in a very small region of the design space.

Multiple points with large PSRRp error were additionally analyzed to confirm the results of the micro-cube experiment. A similar behavior was observed for these points as well. To make
matters even worse, the points tested were nowhere near one another in the design space. This means that the problem is occurring in isolated and sporadic areas of the design space. The same problem was confirmed to be occurring for CMRR for the cascode op-amp.

Poor modeling accuracies for the PSRRp of the single-stage op-amp were also observed in Chapter 6 for the adaptive pseudo-cubic spline results. Table 6-5 on page 80 defines two ranges of component values defining a small domain, and a large domain, over which the performances were modeled. Referring to Figure 6-15 on page 83, we see that gross inaccuracies in the PSRRp model only occurred for the large domain configuration. This means that the problem is occurring only for large values of transistor widths and lengths. This is assuming that, if the problem exists in the small domain configuration, the ten thousand point validation set is sufficiently dense to resolve at least one grossly inaccurate point.

This is certainly a strange problem. We by no means claim that the micro-cube experiments confirm that the odd behavior of PSRRp and CMRR is due to the BSIM3v3 models. However, many of the common problems that may occur in macro-modeling were systematically ruled out in the previous paragraphs. The fact that this incidence occurred for two different performance metrics also adds to the complexity of the situation. It seems that the problem is either a bug or unknown caveat of the SPICE simulation engine, or a stability problem within the BSIM3v3 models. However, more work would need to be done to come to any definitive conclusions.

7.6 Conclusions

This chapter discussed the sizing rules method [38] and its applicability to analog performance macro-modeling. The sizing rules method defines design space and electrical space constraints for a hierarchical library of commonly used analog sub-circuits. The sub-circuits in the library are automatically matched to components in any general analog topology to generate a list of equality and inequality constraints in order to implicitly define a feasibility region. Research suggests that the feasibility region ensures proper operation of the circuit, good behavior of performance metrics, insensitivity to process variations and robust manufacturability [24][38][77][91]. The premise of analog performance macro-modeling is the elimination of SPICE from the synthesis loop during sizing of analog topologies. The feasible region, as defined by the sizing rules method, is not explicit in the design space and must be modeled in order to fully circumvent the use of SPICE simulation. Two methods were explored for modeling the feasible design region. Specifically, a linearization methodology [91] for constraints in the electrical space was generalized to a power series classifier. Nearest neighbor search was also explored as a viable alternative to power series estima-
tion of electrical space constraints. A series of experiments were presented in which the performance parameters of two op-amp topologies were modeled in a naive box-constrained domain, an approximate feasible region, and true feasible region to explore the benefits of the sizing rules method. In addition the accuracy of neural network performance models was compared to pseudo-cubic and adaptive pseudo-cubic models. Lastly, poor modeling accuracy due to anomalous behavior of two performance metrics was discussed.

The sizing rules method, while useful for semi-automatically defining the feasibility region, does not solve all the problems associated with the establishment of an appropriate set of design and electrical constraints. It fails to realize many types of basic topologies and even to simply constrain all transistors to either triode or saturation mode. For example, consider transistor M8 for the single-stage op-amp circuit in Figure 7-4. Though it indirectly acts as a current mirror as discussed in Section 7.3.2, it is not recognized by the sizing rules method as a current mirror transistor and must be constrained manually to saturation mode. The sizing rules library could be expanded to include additional specialized cases, however, this in itself is a fairly involved research problem. The sizing rules methodology does not appear to reduce the dimension of the design space or simplify performance parameter behavior sufficiently for performance macro-modeling. This isn’t necessarily a fault of the sizing rules methodology. Modeling of high dimensional nonlinear functions is a difficult problem, and it seems that even a simple circuit with nine free variables is out of reach at this time.

It is possible to model a variety of performance metrics when circuits are manually analyzed and constrained beyond the basic sizing rules technique. Modeling examples have been presented in this thesis for circuits with up to 33 transistors and as many as five free variables. The sizing rules method calls into question the validity of models defined on a box-constrained domain. It defines a framework for the analysis of analog circuits based on the internal voltages and currents arising from a sized circuit configuration. While the modeling accuracy is not greatly improved when designs are constrained to the feasible region, circuits inside the feasible region are more desirable from a manufacturing and robust design standpoint. Design points outside the feasible region should not be accepted as a valid design, even though the models are accurate for these points and the synthesized circuit may meet all performance requirements.

The feasibility of a design point cannot be known at this time without running a SPICE DC analysis, therefore the explicit shape and size of the feasible region is also unknown. In order to keep the computational cost of performance estimation low, it is imperative to completely eliminate the need to call any form of SPICE simulation. If the feasible design space is a significant portion
of the box-constrained domain, it is computationally feasible to generate macro-model sample points strictly inside the feasible region by running a SPICE DC analysis to impede the incorporation of infeasible points into a performance macro-model. For examples in this chapter the relative size of the feasible regions w.r.t. the box-constrained domain ranged between 44% to 68%. SPICE could be used to reject infeasible designs during the model building process, however, we cannot rely on SPICE during evaluation of a performance macro-model for feasible point identification. Therefore the actual feasible region is of little use since the models must be valid over not only the true feasible region, but an approximate feasible region not defined through SPICE simulation. Although two methods were discussed for estimation of the feasible region, results indicate that nearest neighbor search is more stable and accurate than the power series methodology. Feasibility region approximation is an active research area [22], and while more accurate classifiers may exist, nearest neighbor search was shown to perform fairly well for examples in this chapter. Once a valid approximate feasible region is defined, it makes sense to create performance models strictly within this region.

Problems can occur when attempting to model analog performance metrics. This was seen for positive power supply rejection ratio of the single-stage op-amp and common mode rejection ration of the cascode op-amp. Even after thorough analysis, it was not possible to determine the source of modeling error for these two performance metrics. Accurate macro-modeling relies on a stable simulation platform, proper DC biasing, robust test-bench circuits, proper extraction of performances from raw data, nonlinear multi-dimensional models, and complex BSIM3v3 transistor models. A bug or problem in any of these steps can result in unusable models. More work needs to be done in order to discover the source of error in the PSRRp and CMRR models, but evidence points to a stability issue in either the SPICE simulator or the underlying BSIM3v3 models.
8 Conclusions and Future Directions

8.1 Conclusions

In this thesis, the use of fast analog performance models for use in the synthesis of analog circuits has been studied. The models estimate performance metrics by efficiently mapping the device sizes in a topology directly into performance metrics and can be used as surrogates for SPICE simulation. Experimental evidence indicates that accurate models can be constructed for a variety of performance metrics such as open loop gain, unity gain frequency, phase margin, common mode rejection ratio, power supply rejection ratio, slew rate, conversion gain, and even layout aware inductance. Many of the experiments have focused on modeling performance metrics of operational amplifier topologies ranging in size from 8 transistors up to 33 transistors. The computational cost of performance estimation can be reduced by a factor of more than 40,000 using neural network models, resulting in much faster synthesis times.

The complexity of techniques for modeling performance metrics has progressively increased throughout this thesis. Initial experiments, using neural networks as a modeling platform, utilized a box-constrained function domain and gridded sampling to model various performance metrics of 3 different op-amp topologies. Constraints on design variables were manually determined in order to ensure proper biasing of the circuits and to reduce the dimensionality of the design space. A genetic algorithm was used to size an op-amp topology under various performance requirements using the neural network macro-models to provide performance estimates.

For gridded sampling the number of sample points increases polynomially w.r.t. grid density and exponentially w.r.t. dimensionality, limiting precise control over the number of samples used to construct and validate performance models. Unstructured sampling methods such as pseudo-random or quasi-random generators allow arbitrarily shaped function domains and offer fine control of sample point density, however they do not consider the history of previously sampled points or the shape of the function and are referred to as static sampling methods. Two unique adaptive sampling algorithms were developed in Chapter 6 that are capable of enhancing the accuracy
of performance macro-models using fewer sampled points than random and gridded sampling. We believe the pseudo-cubic spline based sampler to be superior to the piecewise-linear sampler due to its inherent ability to model nonlinear functions as well as its insensitivity to dimensionality. The piecewise-linear sampler suffers from an exponential increase in simplices (linear facets describing the function) with respect to the number of sample points and dimensionality. Additionally, the error of the piecewise linear models can be severe near the boundary of the domain where simplices are likely to be stretched over a large area of the domain. Experimental tests of analog performance macro-modeling using the adaptive sampling algorithms were conducted using a box-constrained domain as well as hand derived design constraints similar to those used in Chapter 5.

The sizing rules method [38] and feasibility region modeling were incorporated into the analog performance macro-modeling paradigm in Chapter 7. The sizing rules method implicitly defines a feasibility region in the design space which is reported to ensure proper operation of a circuit, good behavior of performance metrics, insensitivity to process variations and robust manufacturability [24][38][77][91]. Experiments indicate that the constraints generated by the sizing rules method do not reduce the dimensionality of the design space or simplify the behavior of performance metrics sufficiently for the purpose of analog performance macro-modeling. Additional, manually derived constraints on the design parameters of a circuit, similar to those used in Chapter 5 and Chapter 6, are necessary to simplify the behavior of performance metrics. With the addition of these constraints, the modeling accuracy of performance metrics can be much improved. However, this increase in accuracy is realized over the entire box-constrained domain space. The accuracy of performance macro-models does not seem to improve significantly when models are constrained to the feasible design region. The main strength of the sizing rules appears to be the enforcement of electrical space constraints that ensure robust design and correct DC biasing of analog topologies. We feel that the sizing rules method is a step in the right direction and that it is in fact bad practice to ignore the internal behavior of DC voltages and currents of a circuit. In previous chapters the circuit was treated as a black box with observable inputs and outputs. However, the sizing rules method argues against the black-box methodology in favor of a more detailed analysis of the inner workings of circuits. The sizing rules method can be considered a necessary, but not a sufficient condition for effective analog performance macro-modeling.

A SPICE DC analysis must be used to check the feasibility of a design point. However, analog macro-modeling requires the elimination of SPICE during performance estimation to be computationally efficient. For this reason two feasibility region approximation methods were analyzed which can classify a design point as feasible or infeasible without the need to run SPICE. The
method of linearization of voltage space constraints [91] did not prove to be as accurate as the nearest neighbor classifier methodology. The linearization method is sensitive to the design point around which the constraints are approximated, and often did not provide good classification accuracy of the feasible design space. Nearest neighbor search proved to be a viable method for the approximation of the feasible design region. This allows performance macro-models to be constructed within the approximate feasible design region as defined by nearest neighbor search. The feasibility model can then be packaged with the performance parameter macro-models to produce a unified model that can identify feasible designs as well as provide performance estimates.

8.2 Future Directions

8.2.1 A critique of the adaptive sampling algorithms

A main contribution of this thesis is the development of two adaptive sampling algorithms used to intelligently guide the placement of sampled design points in order to enhance the quality of performance parameter models. The sample points chosen by an adaptive sampler are intrinsically coupled to a specific modeling methodology. An optimal set of sample points for one modeling methodology will probably not be optimal for a different type of model. The shape of the underlying function certainly affects the required placement of sample points, however, the modeling methodology used to estimate the function also strongly biases sample point placement. For example, a piecewise linear model can easily model sharp and abrupt features using fewer sample points than Duchon pseudo-cubic splines. However, smooth and flowing features are more readily modeled by the spline models. This property can be seen in Figure 8-1, in which the test function was optimally sampled to reduce modeling error using both a piece-wise linear model and pseudo-cubic spline model. The two models need completely different sets of sample points in order to maximize model quality.

Figure 8-1. Density of optimally placed sample points for piecewise-linear and pseudo-cubic models.
This coupling between modeling methodology and sample point placement is evident in the adaptive sampling algorithms developed in Chapter 6. The piecewise-linear adaptive sampler operates by increasing sample density in areas with high curvature and nonlinearity. This methodology works because inaccuracies in the piecewise linear model will occur when a linear approximation will not sufficiently describe the function. Extending this concept to nonlinear pseudo-cubic spline models is non-trivial. The spline models are intrinsically nonlinear and do not necessarily need a high density of sample points to accurately model a nonlinear region. In the case of the piecewise-linear adaptive sampler we were able to guess a good metric for sample point placement, i.e. place samples in the most nonlinear regions. The coupling between the piecewise-linear methodology and sample point placement is only loosely defined and is based on simple linear primitives. The coupling between modeling methodology and sample point placement is much stronger for the pseudo-cubic spline based sampler. We can no longer reason through an appropriate metric for driving sample point placement. *The pseudo-cubic spline needs additional sample points in areas where it needs additional sample points.* This sentence may seem redundant or confusing but it actually precisely describes the problem. There is no external metric for guiding the placement of sample points, the model itself must be used as the guide. This concept was realized algorithmically by systematic addition and deletion of sample points in the spline in order to detect inconsistencies, or poorly defined regions in the spline model. The idea is that if a spline is well supported by the sample points, a small change in these sample points should not drastically affect the shape of the spline model.

The pseudo-cubic spline base adaptive sampling algorithm developed in Section 6.2, while valid for functions of any dimension, performs remarkably well for the 1-dimensional test case (the “chirp” function), but performance of the sampler seems to degrade with increasing dimensionality. This effect needs to be further explored in order to determine the source of degradation in performance. Likely candidates are several trade-offs that were made in order to make the algorithm computationally feasible. For example, only a finite set of candidate points are considered at one time for sampling. In addition a small localized spline model is used to approximate the effects of changes made to the comprehensive spline model comprised of the complete sample point set. The effects of perturbations in the spline model are only approximately detected using an ad-hoc implementation of an infinity norm operator. Any or all of these compromises could be affect the performance of the adaptive sampler in higher-dimensional settings. The stellar performance of the sampler for the 1-dimensional test case indicates that the underlying algorithm has merit, but may be falling short in higher dimensions due to implementation issues. This raises the question of
whether the sampling algorithm could be cast into a more rigorous mathematical framework. While this could be an interesting research topic, the mathematical concepts used by Duchon [23] to develop thin-plate and pseudo-cubic splines are likely beyond the comprehension of many computer and electrical engineers. While the benefits of such a sampling algorithm could be specifically applied to analog performance macro-modeling, it could be an interesting addition to the repertoire of multi-dimensional modeling methodologies in general.

8.2.2 On the imperative accuracy of performance macro-models

Throughout this thesis we have been striving to build performance macro-models that fit, as closely as possible, the performance values reported by SPICE. However, the accuracy of SPICE is limited by the accuracy of the underlying transistor models. In addition, the sized circuit cannot be exactly reproduced due to process variations in integrated circuit manufacturing. So how accurate do performance macro-models really need to be? The transistor models in this thesis were obtained from MOSIS [95], a low-volume prototype company. They offer BSIM3v3 models that are extracted from real world tests of specific die lots for a variety of process technologies. Therefore the transistor models are representative of average process variations for a set of specific process conditions. Transistor models from differing die lots will yield variations in the performance values reported by SPICE. This thesis does not address the variability of performance values due to process variations and transistor model inaccuracies. However, this is an important question for performance macro-modeling since it dictates the overall accuracy required of the performance models, and therefore the effort required to build suitable models. Consider for example a performance parameter such as open loop gain. If a model is developed with a maximum error of 0.1dB, but process variations and transistor model inaccuracies result in variability of open loop gain by 1dB, the detailed accuracy of the model is in vain. The concept of inherent and unresolvable error due to process and transistor model variance is probably best studied in a statistical framework.

We have also seen strange behavior of performance metrics such as PSRRp for the single-stage op-amp and CMRR for the cascode op-amp. The inaccuracies occurring for these performance metrics was not due to issues with the macro-modeling process, but were instead inherent to SPICE. Some analog synthesis tools [11] use SPICE in the loop to estimate the performance parameters of analog designs. This calls into question the methods used to deal with potentially erratic performance parameter behavior by an analog synthesis tools directly using SPICE as a performance estimator. Additional research needs to be performed to identify the sources of error occur-
ring in PSRRp and CMRR before we can expect to build accurate performance models for these metrics.

8.3 Summary

We feel that analog macro-modeling is not mature enough at this time for a general modeling tool to be developed. The sizing rules method is a good effort by researchers to automatically constrain a circuit to ensure proper operation, good behavior of performance metrics, insensitivity to process variations and robust manufacturability [24][38][77][91]. It does not, however, provide a framework rigorous enough to automate the process of dimensional reduction and simplification of performance parameter behavior for the purposes of analog performance macro-modeling. When performance macro-models are properly constructed, they can provide significant computation savings during the synthesis of analog circuits. However, there still exist caveats that often prevent the acquisition of accurate performance information. We must also rely on manual analysis of circuits to establish an appropriately constrained design space. Thus a fully automatic methodology for the development of accurate and efficient analog macro-models is still an open research problem.
Bibliography


[92] UC Berkely Design Technology Warehouse, http://www-cad.eecs.berkeley.edu


[94] BSIM Homepage - Official Site, http://www-device.eecs.berkeley.edu/~bsim3