Computer Science and Artificial Intelligence Laboratory

Technical Report

MIT-CSAIL-TR-2007-032 June 5, 2007

An Analysis of Posynomial MOSFET Models Using Genetic Algorithms and Visualization

Lynne Rafik Salameh

massachusetts institute of technology, cambridge, ma 02139 usa — www.csail.mit.edu

An Analysis of Posynomial MOSFET Models Using Genetic Algorithms and Visualization

by

Lynne Rafik Salameh

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Masters of Engineering in Electrical Engineering and Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2007

Author .....
Department of Electrical Engineering and Computer Science
May 25, 2007

Certified by .....
Dr. Una-May O'Reilly
Principal Research Scientist
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

An Analysis of Posynomial MOSFET Models Using Genetic Algorithms and Visualization

by

Lynne Rafik Salameh

Submitted to the Department of Electrical Engineering and Computer Science
on May 25, 2007, in partial fulfillment of the
requirements for the degree of
Masters of Engineering in Electrical Engineering and Computer Science and Engineering

Abstract

Analog designers are interested in optimization tools which automate the process of circuit sizing. Geometric programming, which uses posynomial models of MOSFET parameters, represents one such tool. Genetic algorithms have been used to evolve posynomial models for geometric programs, with a reasonable mean error when modeling MOSFET parameters. By visualizing MOSFET data using two dimensional plots, this thesis investigates the behavior of various MOSFET small and large signal parameters and consequently proposes a lower bound on the maximum error, which a posynomial cannot improve upon. It then investigates various error metrics which can be used to balance the mean and maximum errors generated by posynomial MOSFET models. Finally, the thesis uses empirical data to verify the existence of the lower bound, and compares the maximum error from various parameters modeled by the genetic algorithm and by monomial fitting. It concludes that posynomial MOSFET models suffer from inherent inaccuracies. Additionally, although genetic algorithms improve on the maximum model error, the improvement, in general, does not vastly surpass results obtained through monomial fitting, which is a less computationally intensive method. Genetic algorithms are hence best used when modeling partially convex MOSFET parameters, such as r_0.

Thesis Supervisor: Dr. Una-May O'Reilly
Title: Principal Research Scientist

Acknowledgments

First and foremost, I would like to extend my sincerest thanks to my thesis supervisor, Dr. Una-May O'Reilly, whose unrelenting enthusiasm towards the project, and her continuous advice, support and feedback have propelled my research forward. Her excitement towards her research will always remain my source of inspiration, and she has been the main catalyst behind the ideas presented in this thesis, for which she retains my foremost gratitude.

Secondly, I'd like to thank Varun Aggarwal, fellow advisee, labmate, and Master's candidate, for his immense wells of valuable knowledge, and for his help in devising creative solutions around my research roadblocks. I also thank him for distracting me with long, engaging conversations about history and existentialism, whilst keeping me company during late lab hours. I would also like to thank Tony Eng and Prof. Chris Terman for allowing me to TA their respective classes, through which I've gained invaluable teaching experience and funded my graduate education. I would also like to thank Mar Hershenson for her insightful discussions regarding posynomial MOSFET modeling.

Finally, I would like to thank my parents, Rafik and Leila, without whom I would have never made it to MIT in the first place. Their faith, loving support, and patience were immense sources of encouragement for completing this research, and for surviving MIT's proverbial fire-hose.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Circuit Sizing Approaches

Although analog and mixed mode circuits constitute a fundamental component of electronic design, the stages of their design and verification continue to pose a significant bottleneck within the overall production process. The lack of automation in the design stages delays the release of the product for marketing. Therefore, fast and reliable Computer Aided Design (CAD) tools have become a pressing demand for analog designers.

Analog designers are tasked with taking large and small signal models of circuit components, and, in order to fulfill certain performance requirements, deducing the components input parameters in accordance with the output constraints. For example, given output performance measurements like gain g_m , unity gain bandwidth w_c , and phase margin ϕ, they must correspondingly produce the input MOSFET parameters, e.g. the length L, and width W and the input current I. The process of translating the performance measurements into component parameters is called circuit sizing. In a modern analog design process, designers can specify between 10 to 100 input parameters in order to achieve up to 20 output performance measurements.

Several automatic and manual methods for circuit sizing exist in practice. The manual method involves a designer using his or her accumulated knowledge of circuit behavior to iteratively adjust the component parameters such that they satisfy

a set of first order transistor models, and then test the accuracy of these models. Naturally, tests performed on the fabricated silicon circuit produce the largest accuracy, but since continuously fabricating test circuits is costly and not readily available during the design stages, simulating the circuit with SPICE yields a good approximation of circuit behavior. SPICE is a circuit simulator with highly complex physical device models: its results reliably confirm whether a circuit meets its performance specifications given its sizing parameters.

In practice, the mapping between the input component parameters and output performance measurements is multi-modal and misbehaved due to parameter coupling. Therefore adjusting one parameter to satisfy a certain performance constraint may result in a failure to satisfy another constraint, and hence trades off one improvement in performance with a degradation in performance relative to another variable. Therefore, the manual process is a very lengthy and tedious one, because, since it is impossible to satisfy all the required constraints on the first pass, the designer is forced to continuously readjust the parameters and try again.

In order to tackle the issue of parameter coupling, circuit sizing can by recast as an optimization problem, where a specific algorithm optimizes one or several objectives subject to a set of constraints. But before one can choose the most suitable optimization method for circuit sizing, it becomes imperative to evaluate several available optimization methods in terms of speed, ease of use, and accuracy.

Automated Equation-Based Approaches fall under one class of optimization techniques, and they use simplified transistor models in lieu of manual designer effort. The approaches analytically solve multiple symbolic equations relating the performance measurements and input parameters. Although faster, the approaches remain relatively less accurate, since the equations do not provide as good estimates of true circuit specifications as SPICE does. Since the equations are based upon certain assumptions and approximations in terms of transistor behavior, the inaccuracies become even more notable as technologies scale down.

On the other hand, Simulation Based Approaches deploy Black Box Optimization that simply uses SPICE combined with an adaptive search algorithm (e.g. simulated

annealing or genetic algorithms) to optimize one or more targets subject to multiple constraints. SPICE is computationally expensive so, although more accurate, Black Box Optimization typically takes a long time. More recently, Black Box Optimization has been sped up with more powerful computers and parallelization.

Finally, Equation Driven Global Optimization, such as Geometric Programming, expresses the structure of the multiple symbolic performance measurement equations in a form that can be almost instantly solved. In Geometric Programming, output performance parameter equations are approximately expressed using posynomials, which are polynomials restricted to only containing positive coefficients. From there, geometric programming takes the performance measurements and constraints expressed as a series of posynomial equations, and computes the global optimum for the objectives in a matter of seconds. Unfortunately, the process of expressing the performance measurements in posynomial form is often performed manually, resulting in posynomial models which do not accurately reflect transistor behavior and thus leading to the determination of faulty global optima. On a similar note, although the Geometric Program itself consumes a relatively small of time, the process of expressing transistor models as posynomials manually proves to be quite taxing.

Using empirical data gathered from several transistors, Genetic Algorithms can generate the posynomial MOSFET models required for the Geometric Program. Aggarwal et al.[1] used Genetic Algorithms to generate the posynomial MOSFET models and measure the mean fitness of these models. Because they depend on measured empirical data, for which a certain error metric is minimized, the generated posynomial models exhibit an improvement in accuracy. The following section will discuss Genetic Algorithms as an approach to circuit modeling in more detail.

1.2 Genetic Algorithms for Circuit Modeling

In the field of automated circuit design, Genetic Algorithms have been previously used as standalone methods for generating both the topology and components of a circuit. Koza et al.[2] has shown that, given the number of inputs and outputs of

the circuit, the set of available components, and a fitness measurement in terms of performance, Genetic Programming tended to perform satisfactorily in synthesizing 8 different analog circuits. Nevertheless, the computational cost for solving the sizing problem becomes less feasible for more complex circuits such as op-amps, which require more stringent constraints and therefore larger dimensionality. In fact, due to the need to simulate each new generation in SPICE for a large number of fitness evaluations, even simple filter circuits took 2 days to run on a parallel computing system with 64 80-MHz processors.

In order to improve computation speed, Grimbleby et al.[3] decoupled the circuit synthesis problem into two different stages: topology design, then component parameter selection. The two stage hybrid Genetic Algorithm uses evolutionary techniques to choose the topology of the first circuit, and then numerical optimizations to size the circuit components. Although the hybrid Genetic Algorithm exhibits a performance improvement in linear analysis, non-linear analysis continues to consume a significant chunk of computational power in the numerical optimization stages.

The above two simulation based techniques suffer from scalability problems, and often yield solutions that are suboptimal. They are therefore unfeasible for the commercial design of robust products. Geometric Programming provides a solution to these problems: since it uses simplified interior-point methods in place of the numerical optimization of the hybrid Genetic Algorithm problem it does not suffer from reduced performance and scalability problems. Hershenson et al. [4] and Mandal et al.[6] applied Geometric Programming on simple CMOS op-amp circuits and arrived at a global optimum in a matter of seconds. Unfortunately, Geometric Programming equations, as mentioned in Section 1.1, can be subject to inaccuracy in the hand-written equations. Along the same lines, the generation of the posynomial equations would benefit from automation and relaxing the need for topology specific knowledge, so that the Geometric Programming optimization can extend beyond simple op-amp circuits.

Finally, Genetic Algorithms were proposed as tools for reducing the inaccuracies found in the MOSFET posynomial equations, [1, 5]. Given certain performance vari-

able constraints, posynomial equations were evolved and then evaluated for fitness against a random sample from 70000 data points of measured empirical data. Each data point corresponds to a performance measurement for a TSMC 0.18μm NMOS MOSFET given its parameters L, W and I. The points were systematically generated by sweeping through the MOSFET's entire range within the saturation region. The new algorithm, which generated models optimized for mean squared error, generally showed a reduced mean squared error over models generated by hand written posynomial models, piecewise monomials and logarithmic regression. But, although the new method showed up to a 85% [1] performance increase over other models, on the other hand it exhibited large error for certain estimated component parameters.

1.3 Problem Statement

In retrospect, although a reduction in mean squared error seems like a substantial improvement in model accuracy, analog designers are more interested in reducing the maximum single data point error between the posynomial transistor models and the underlying physical ones. A reduction in maximum error will, as a result, ensure that solutions from the Geometric Program don't "fall out" when they are simulated later in SPICE. But, it is important to recognize that the mean and maximum errors exist in tradeoff. Therefore, the most useful model is one which balances both: its is a model that achieves a relatively small maximum error without greatly compromising the value of the mean error.

This thesis discusses various approaches aimed at achieving models with a smaller maximum error. It investigates and compares these approaches and determines which of them is best in terms of balancing both the mean and maximum error. It also examines output parameters with large maximum error and attempts to explain the limitation of the algorithm when applied to these parameters.

On the other hand, whilst evaluating the accuracy of various posynomial models

[1] The evolved posynomial models for parameter g_{ds} showed an 85% improvement in mean error over piecewise monomial fitting

on different output parameters, a new question rose to the surface: are we in any sense constrained by only using a posynomial form to express the models? In other words, does the posynomial formulation possess some inherent limitation that restricts model accuracy? Despite the accepted use of posynomial models coupled with Geometric Programming for solving MOSFET optimization problems, there remains a need to examine the suitability of posynomial models with regards to the nature of the MOSFET data. In accordance with the special requirements enforced by the Geometric Program, we have so far been concerned with generating better posynomials. But the question we should ask ourselves is whether or not MOSFET parameters are well approximated by posynomials models in the first place.

In order to answer the new questions posed, the consequent chapters will, in addition to evaluating models that reduce the maximum error, investigate the nature of the MOSFET data and how well we can use posynomial models to emulate MOSFET behavior. They will present a visualization method for viewing five dimensional MOSFET data, and provide insights on the behavior of a selection of performance parameters using the proposed method. Given the nature of the data, the chapters will propose a theoretical bound on the maximum error arising from posynomial models. They will consequently attempt to verify that such a bound exists. Furthermore, the last chapter will discuss future work in terms of proposed optimization models to be investigated.

Chapter 2

Understanding MOSFET Behavior

2.1 Data Aggregation

In our system formulation, the Genetic Algorithm, (GA), seeks one primary goal: to evolve viable posynomial MOSFET models, mapping the input MOSFET parameters (e.g. L, W, I and V_{ds}) into the output small and large signal performance parameters (e.g. the transconductance g_m). Of course, the algorithm requires empirical training data for calculating the error generated by a particular model, which in turn acts as a selection metric for propagating current models into the next generation of the genetic algorithm.

In order to evaluate the performance of different models generated by the genetic algorithm, a better insight into the nature of the training data is required. Only by exploring the behavior of the output parameters as response to the input parameters for real, physical mosfets, can we derive some intuition about how close our evolved models come to emulating the underlying physical ones.

Using typical TMSC 0.18μm n-doped and p-doped MOSFETs, fabricated with the 0.18μm Logic Salicide (1P6M, 1.8V/3.3V) process, four input parameters, L, W V_{gs}, and V_{ds} were used in a SPICE simulation of the MOSFETs. The simulation swept across the input parameters, beginning at their lower bound and reaching their upper bound in increments of a fixed step size, as shown in Table 2.1 . For some parameters, namely L and W, a logarithmic scale was used. During the simulation,

Parameter	Lower Bound	Upper Bound	Number of Steps	Logarithmic Scale	Units
L	1.8×10^{-7}	1.8×10^{-6}	5	Yes	m
W	8.95×10^{-7}	2.0×10^{-5}	5	Yes	m
V_{gs}	0.2	1.98	10	No	V
V_{ds}	0.0	1.98	8	No	V

Table 2.1: Parameter ranges and sampling used for SPICE simulation

Input Parameters A: L, W, V_{gs}, V_{ds}		**Input Parameters B:** L, W, I, V_{ds}	
Parameter	Units	Parameter	Units
g_m	S	g_m	S
g_{ds}	S	g_{ds}	S
ρ	Ω	ρ	Ω
C_{db}	F	C_{db}	F
C_{gs}	F	C_{gs}	F
C_{gd}	F	C_{gd}	F
V_{dSAT}	V	V_{dSAT}	V
V_{eff}	V	V_{eff}	V
V_T	V	V_T	V
I	A	V_{gs}	V

Table 2.2: Performance output parameters measured using SPICE

a set of output parameters were measured for each combination of input parameters. These were consequently filtered to ensure that the MOSFET was operating in the saturation region, yielding about 900 data points. Given the output parameters, two sets of input to output mappings can be examined, and used to generate the MOSFET models using the Genetic Algorithm. On one hand, we can use the parameters L, W, V_{gs} and V_{ds} as input parameters. On the other hand, we can use L, W, V_{ds} and I. Keep in mind that the current I constitutes a measured parameter from the SPICE simulation. By changing the dependency between the input and output parameters, we can generate two different MOSFET models, and evaluate their performance. Table 2.2 shows the two sets of input parameters, and the output parameters that are associated with them. Figure 2-1 shows a diagram of the small signal representation of a MOSFET, illustrating the significance of the small signal output parameters shown in Table 2.2.

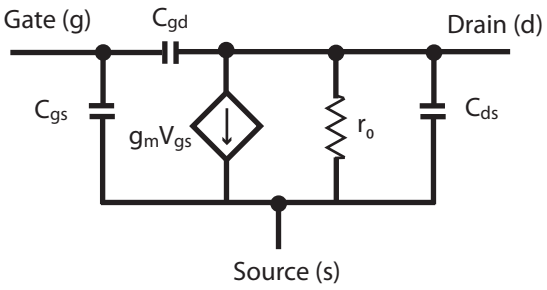


Figure 2-1: Small signal MOSFET model

2.2 Visualization

Since the genetic algorithm will evolve models dependent upon four input parameters, any data visualization technique aimed at understanding the data distribution will require plotting the output parameter as a function of the four proposed input parameters, thus dealing with a five dimensional array. By fixing two of the input parameters, the problem can be reduced to a three dimensional plot. But, in reality, two dimensional plots are probably the easiest to understand. Therefore, we decided to flatten out the third dimension. To obtain a two dimensional plot, we initially slice along two dimensions by fixing two of the input parameters for the current plot. Then, we then sweep through values within the third dimension's range, and for each value, plot the output parameter against the final input parameter. The method, as a result, produces multiple superimposed curves corresponding to different values of the third input parameter, for each plot of the output parameter against one of the remaining input parameters. If we use a Matlab GUI tool, we can also add an animation feature, where instead of fixing the second parameter, we produce several plots corresponding to a different value of the second parameter, and animate across them.

The interactive Matlab GUI tool, shown in Figure 2-2, allows for an ease of manipulation of the data, where the user can specify where to make the slices along any different dimension, resulting in a simplified two dimensional plot. Before running

the visualization tool, the user can easily select which dimension will be plotted on the x-axis, and which dimensions will be toggled by any of the three pulldown menus. On launching the tool, the user can then use the pulldown menus to fix certain dimensions, while changing others. The tool also features the use of color for sweeping through values corresponding to the third dimension. As can be seen in Figure 2-2, the curves traverse from dark to light blue as the value of the third input parameter increases. Therefore, we can easily use the tool to obtain a spectrum of plots, and to inspect the MOSFET data from all possible angles.

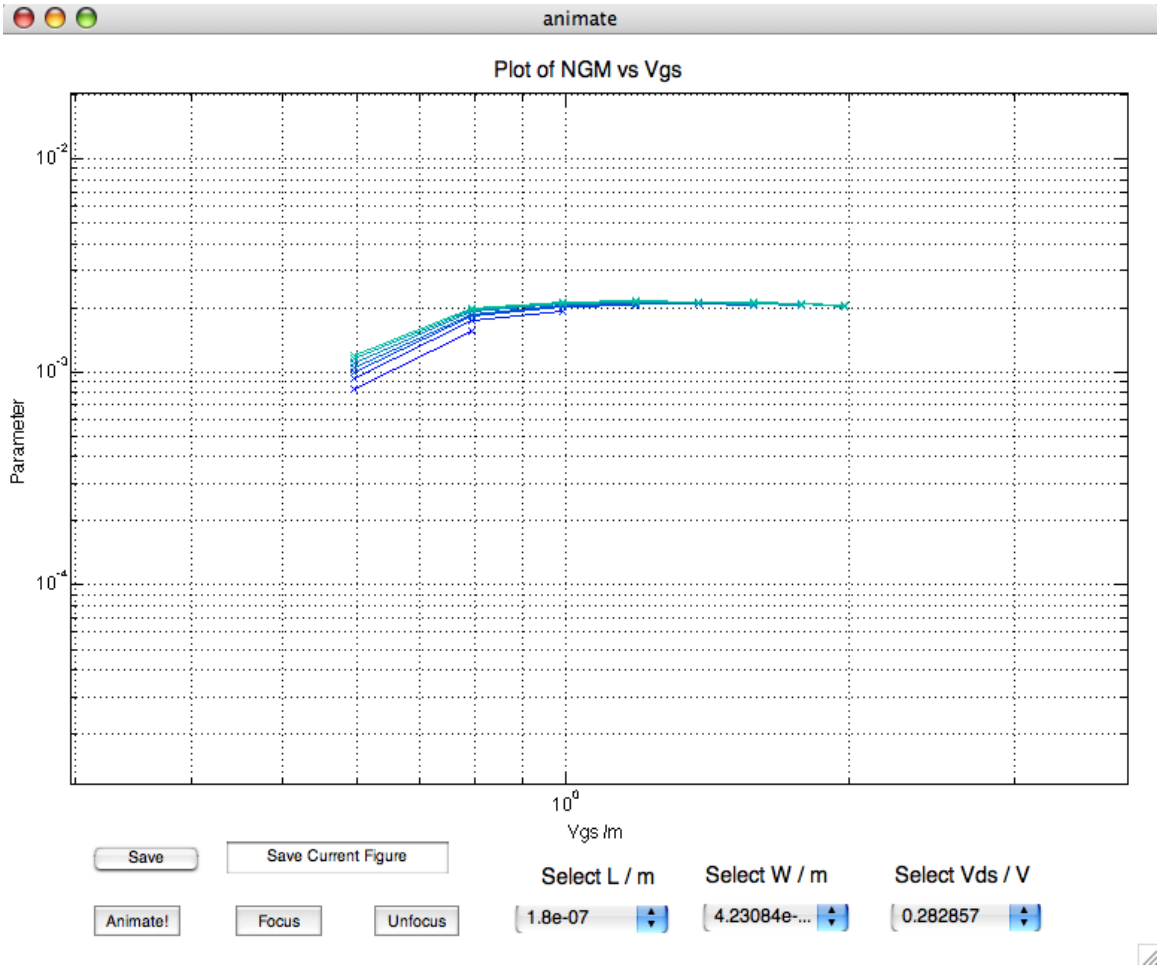


Figure 2-2: Screen shot of Visualization Tool

2.3 Case Study of MOSFET Parameters

In the following discussion, we will be performing case studies on a subset of the parameters shown in Table 2.2, considering data from n-doped MOSFETs. We will be examining the the characteristics of the output parameters g_m, g_{ds}, r_0 and C_{gs} across the four dimensions specified by the input parameters B. The case studies aim to fuel future arguments regarding posynomial models and their use for fitting MOSFET data. All plots in the following were generated using the visualization tool described earlier, and plotted on a logarithmic scale for both the x and y axes [1].

2.3.1 Characteristics of the Transconductance g_m

Perhaps the best method for understanding the behavior of g_m in relation to input parameters L, W, V_{gs} and V_{ds} requires plotting g_m against each input parameter in turn, whilst fixing two of the other parameters and sweeping through the last. The specifics involving which two parameters are fixed and which is swept through are of little consequence, as long as we plot the same parameter on the x-axis. Figures 2-3 and 2-4 show some plots of g_m against each of these four parameters on the x-axis, which in a sense summarize g_m's behavior across the four dimensions. From the small signal MOSFET model, the relationship between the output and input parameters is given by:

$$g_m \cong \mu_n C_{ox} \frac{W}{L}(V_{gs} - V_T) = \sqrt{2\frac{W}{L}\mu_n C_{ox} I} \tag{2.1}$$

Where C_{ox} and μ_n are constants respectively denoting the gate oxide capacitance and the electron mobility.

The relation above indicates that g_m should be linear in L, W and V_{gs} in logarithmic space. But the three plots of Figure 2-3(a) outline g_m's concave response with respect to V_{gs}. The discrepancy results from the underlying inaccuracy of the square-law, from which equation 2.1 was derived. On the other hand, the plots of Figure 2-3(b) show g_m against W for different slices of L and V_{gs}, and, in compliance

[1] Since posynomials are log-convex, we use a logarithmic scale for insights into how well posynomials will model the MOSFET parameters.

with the first order equation, the plots are linear in logarithmic space.

It is also important to note that since the channel length modulation is small, i.e. λ_n is small, the MOSFET's small signal gain g_m exhibits very little dependence on V_{ds}. Figure 2-4(a) shows plots of g_m against V_{ds} for different values of W, and, since they are approximately straight horizontal lines, they indicate that g_m doesn't change much with increasing values of V_{ds}. Nevertheless, the plots display some slight convexity, especially for lower values of L. Finally, g_m plotted against L for small values of V_{gs} shows the predicted linearity. But as we increase V_{gs}, the plots become more and more concave. Figure 2-4(b) shows the concave behavior of g_m with respect to L.

2.3.2 Characteristics of the Output Conductance g_{ds}

The small signal approximation of g_{ds} using the square-law is given by:

$$g_{ds} = \frac{W}{2L}\mu_n C_{ox}(V_{gs} - V_T)^2 \cdot \lambda_n \tag{2.2}$$

The equation indicates that g_{ds} should be linear with respect to L and W in logarithmic space, whereas it should be concave with respect to V_{gs}. After simulation, Figures 2-5 and 2-6 show the empirical behavior of the parameter g_{ds} with respect to the input parameters. Interestingly, the curvature of g_{ds} with respect to V_{gs} changes drastically as we traverse from low values of L to higher ones. As Figure 2-5(a) shows, g_{ds} ellicts some concavity for small values of L, but as L increases, g_{ds} begins to twist around an inflection point. The data is concave to the left side of the inflection point, and convex to the right side of it. Similarly, the data is largely dependent on V_{ds}, and as V_{ds} increases (and the shades of the lines grow lighter), the data's curvature becomes more and more pronounced. In fact, the data is nearly linear for smaller values of V_{ds}. As for g_{ds}'s dependence on W, it is linear in logarithmic space. g_{ds}'s response to L, on the other hand, although approximately linear, shows some non-linearity in the form of an inflection point, as seen in Figure 2-6(b). Finally, g_{ds} is convex with respect to V_{ds}, and the convex curves move upwards as we increase V_{gs}.

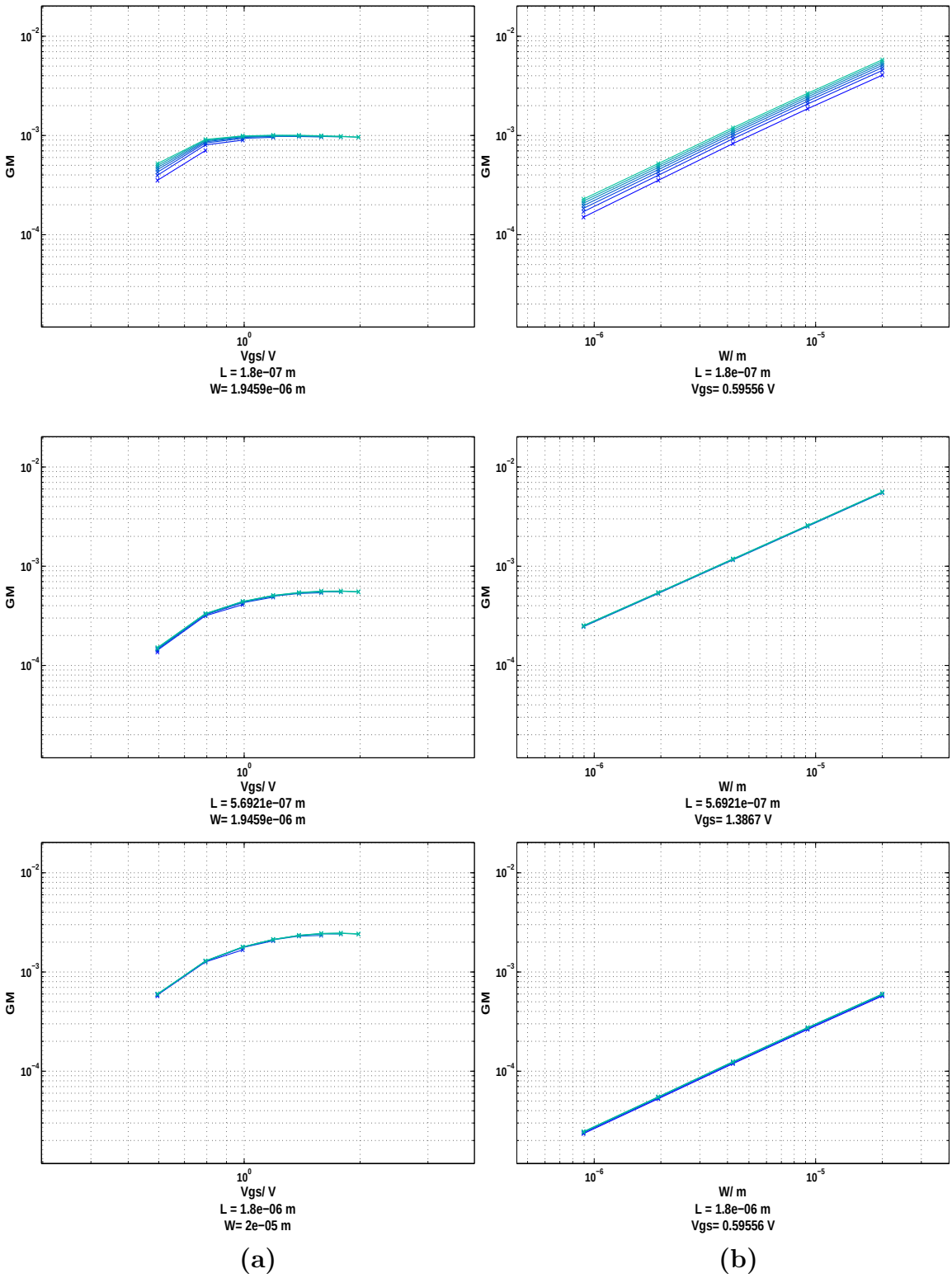


Figure 2-3: g_m plotted against **a)** V_{gs}, **b)** W, for an n-doped MOSFET, with parameter V_{ds} swept across range $[0.28V \quad 1.98V]$.

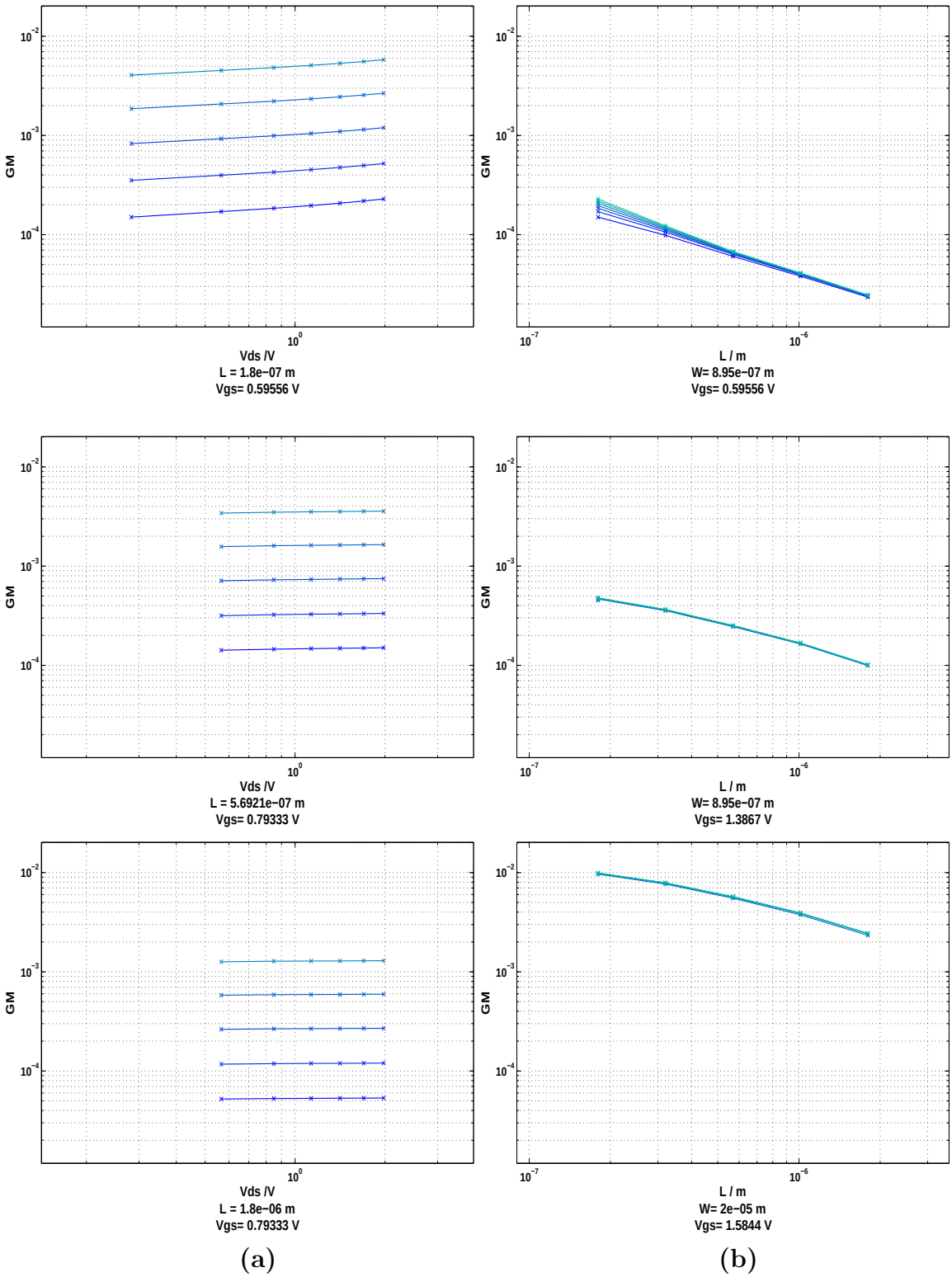


Figure 2-4: g_m from an n-doped MOSFET plotted against **a)** V_{ds} with W swept across range $[0.895\mu m \quad 20\mu m]$ **b)** L with V_{ds} swept across range $[0.28V \quad 1.98V]$.

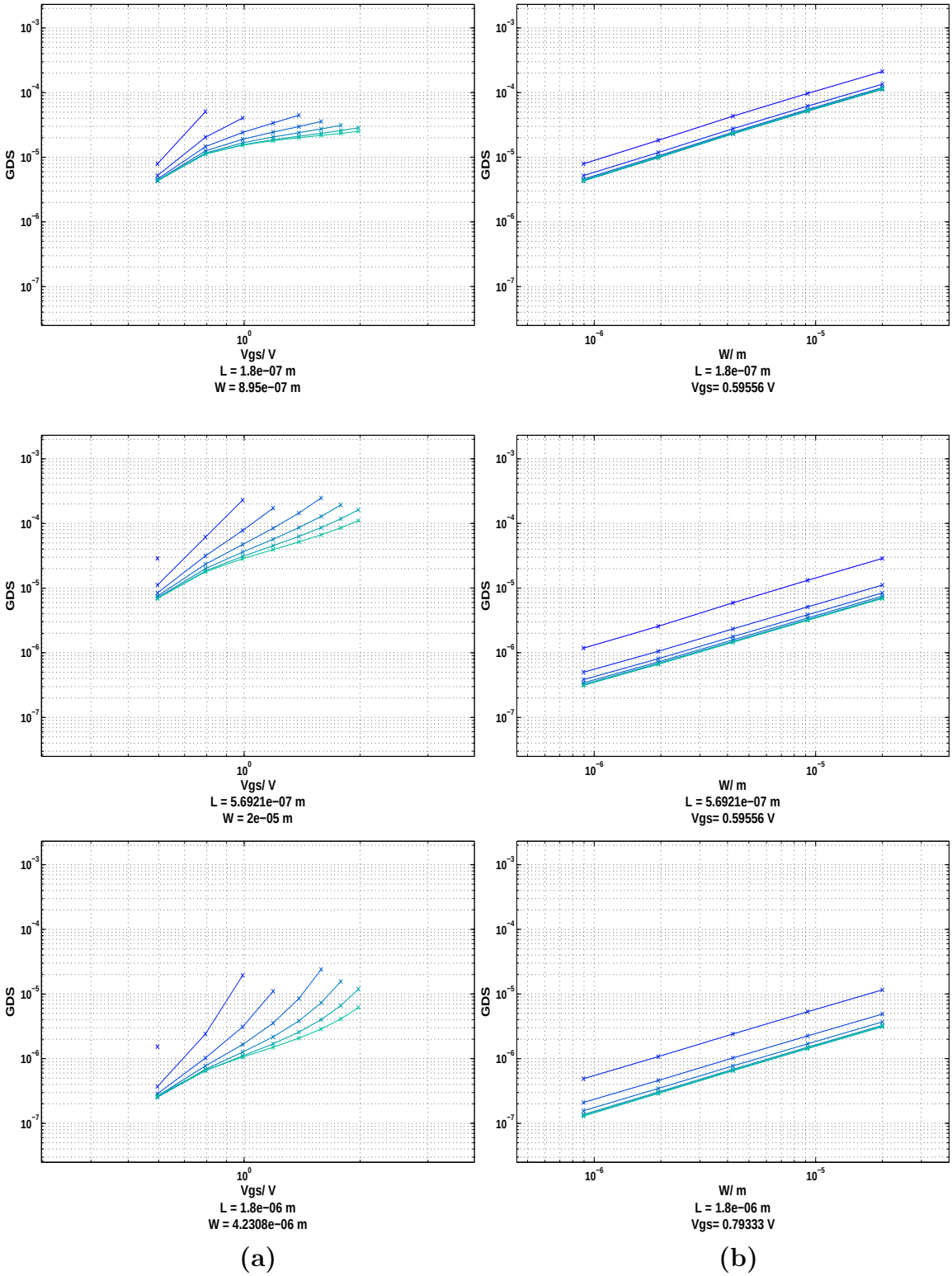


Figure 2-5: g_{ds} plotted against **a)** V_{gs} **b)** W, for an n-doped MOSFET, with parameter V_{ds} swept across range $[0.28V \quad 1.98V]$.

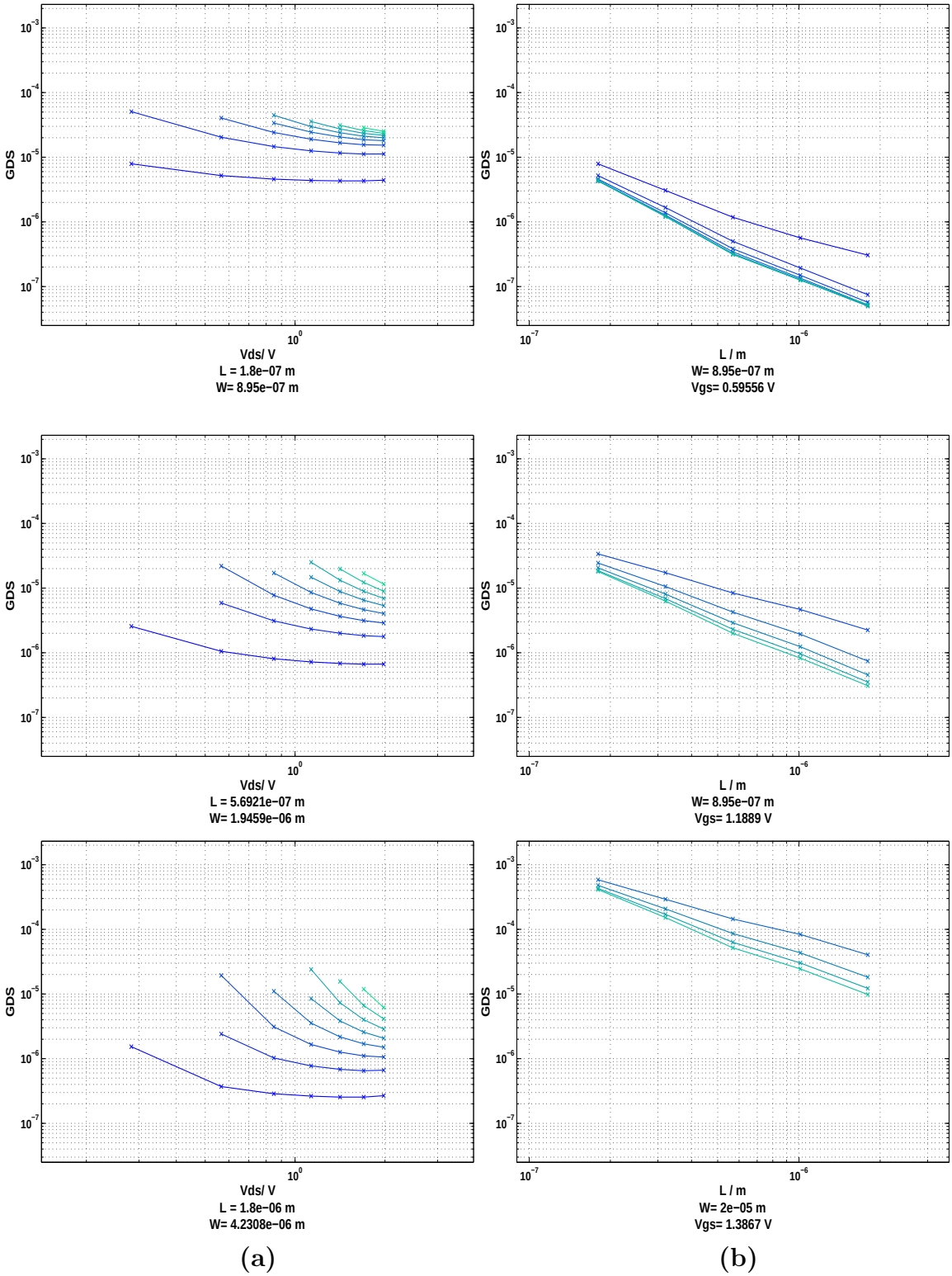


Figure 2-6: g_{ds} from an n-doped MOSFET plotted against **a)** V_{ds} with V_{gs} swept across range $[0.596V \quad 1.98V]$ **b)** L with V_{ds} swept across range $[0.28V \quad 1.98V]$.

2.3.3 Characteristics of the Output Resistance r_0

Since r_0 is given by the equation:

$$r_0 = \frac{1}{g_{ds}} \tag{2.3}$$

we expect plots of r_0 with respect to its input parameters to be the reflection of g_{ds} plots along the $y = x$ axis. Therefore, it is not surprising that input parameters, like V_{ds} for smaller values of L, which were concave for the parameter g_{ds}, become convex for the parameter r_0. Conversely data which is convex, such as g_{ds} relative to V_{ds}, becomes concave, as shown in Figures 2-7 and 2-8. Finally, for the dimensions that exhibit inflection points, i.e. L and V_{gs}, the curvature around the inflection point is inverted for r_0.

2.3.4 Characteristics of the Capacitance C_{gs}

The small signal capacitances, namely C_{gs}, C_{db} and C_{gd}, are linear, or approximately linear in logarithmic space. C_{gs} and C_{gd} are approximated by the following equations:

$$C_{gs} = \frac{2}{3} W L C_{ox} + W C_{ov} \tag{2.4}$$

$$C_{gd} = W C_{ov} \tag{2.5}$$

For example, when we plot C_{gs} for we obtain an approximately linear plot shown in Figures 2-9 and 2-10. C_{gs} is linear with respect to L and W as can be seen from the figures. It is also slightly concave with respect to the parameter V_{gs}. Finally, C_{gs} is very weakly dependent on V_{ds}, therefore the its plot is a horizontal line in logarithmic space.

2.3.5 Characteristics of the Current I

Some parameters exhibit large jumps within their range, where the data seems to be accumulated in two contiguous clusters. The output parameter I plotted against V_{gs} exhibits this property, as can be seen in Figure 2-11. For a fixed length L, the data

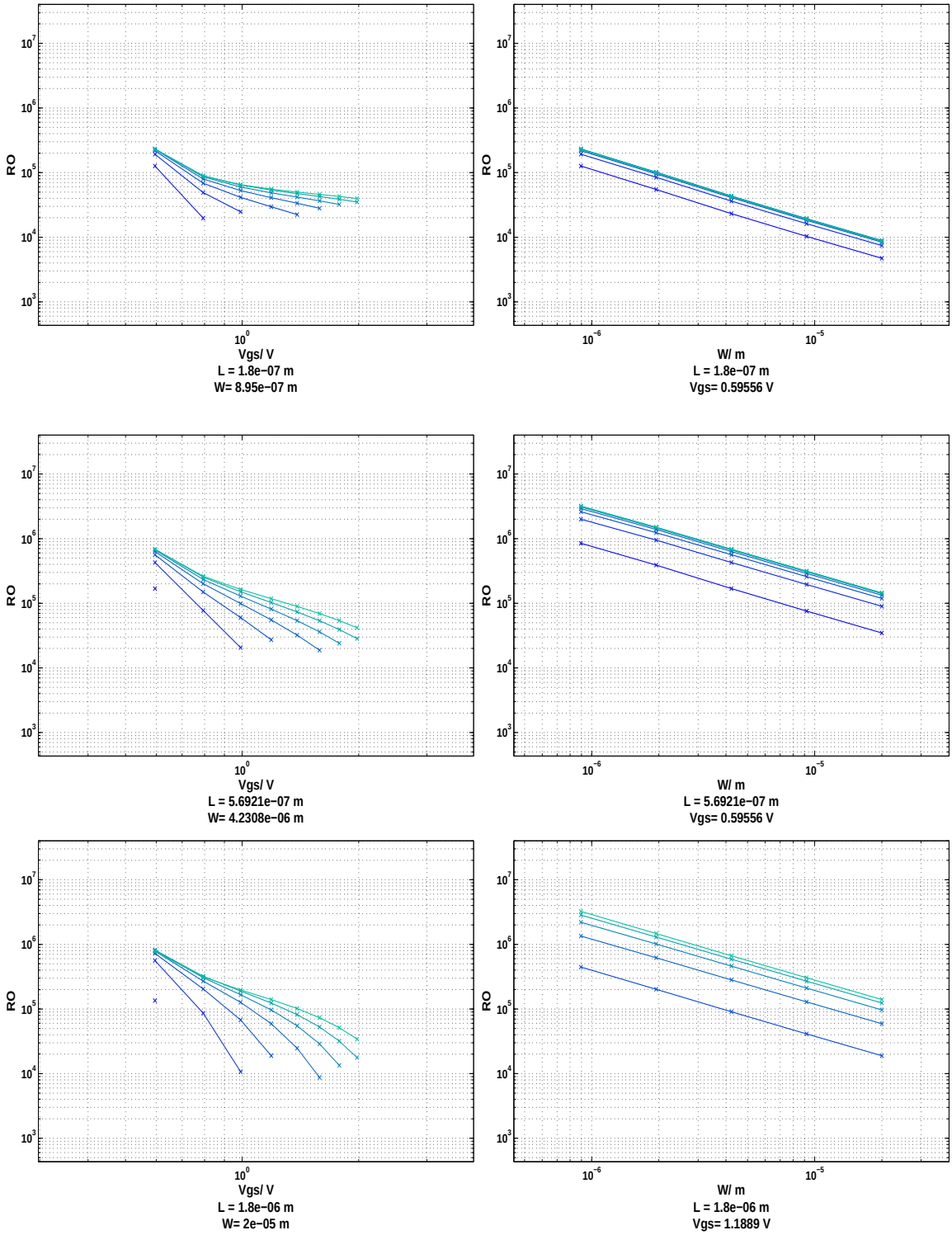


Figure 2-7: r_0 plotted against **a)** V_{gs} **b)** W, for an n-doped MOSFET, with parameter V_{ds} swept across range $[0.28V \quad 1.98V]$.

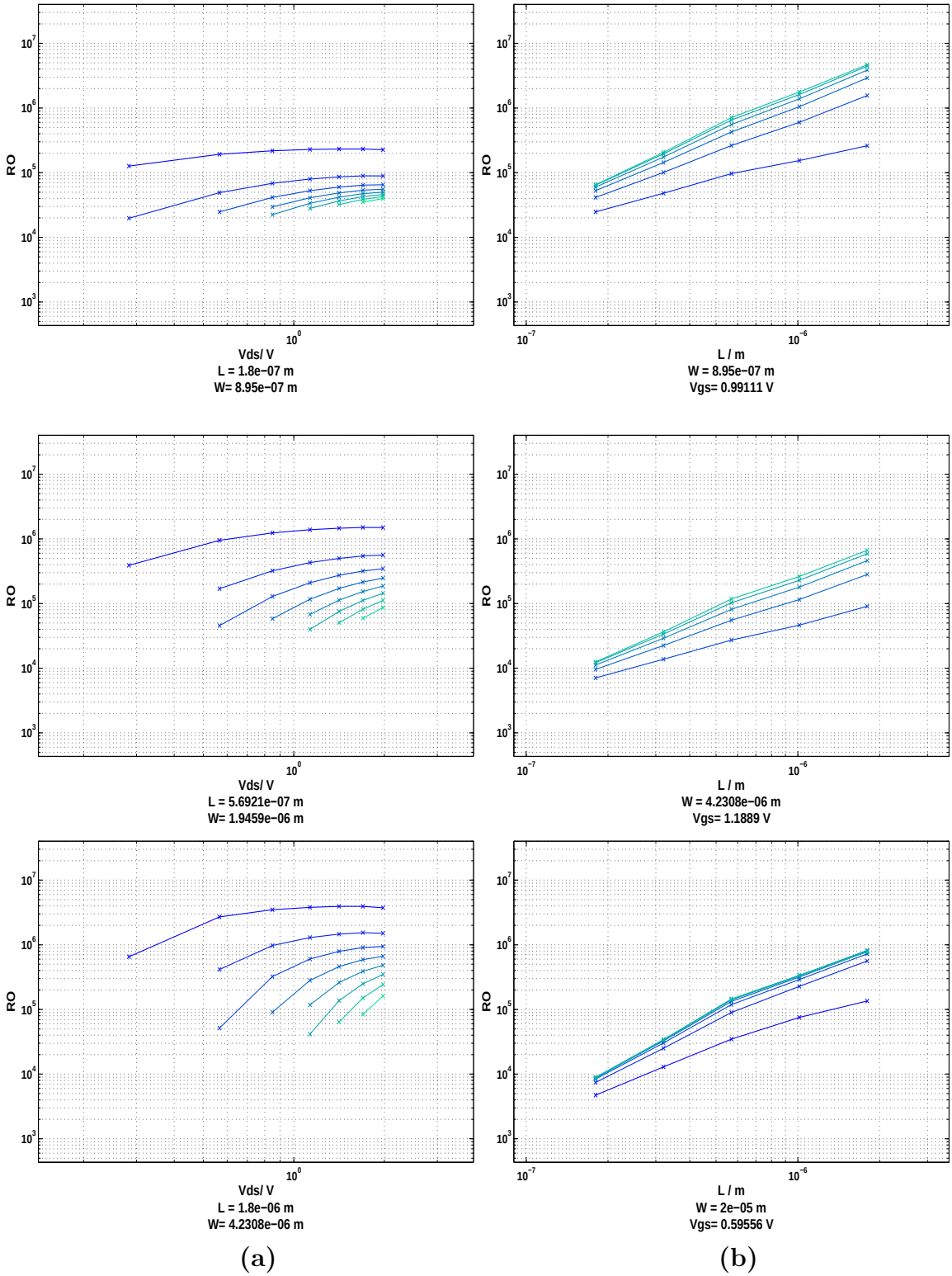


Figure 2-8: r_0 from an n-doped MOSFET plotted against **a)** V_{ds} with V_{gs} swept across range $[0.596V \quad 1.98V]$ **b)** L with V_{ds} swept across range $[0.28V \quad 1.98V]$.

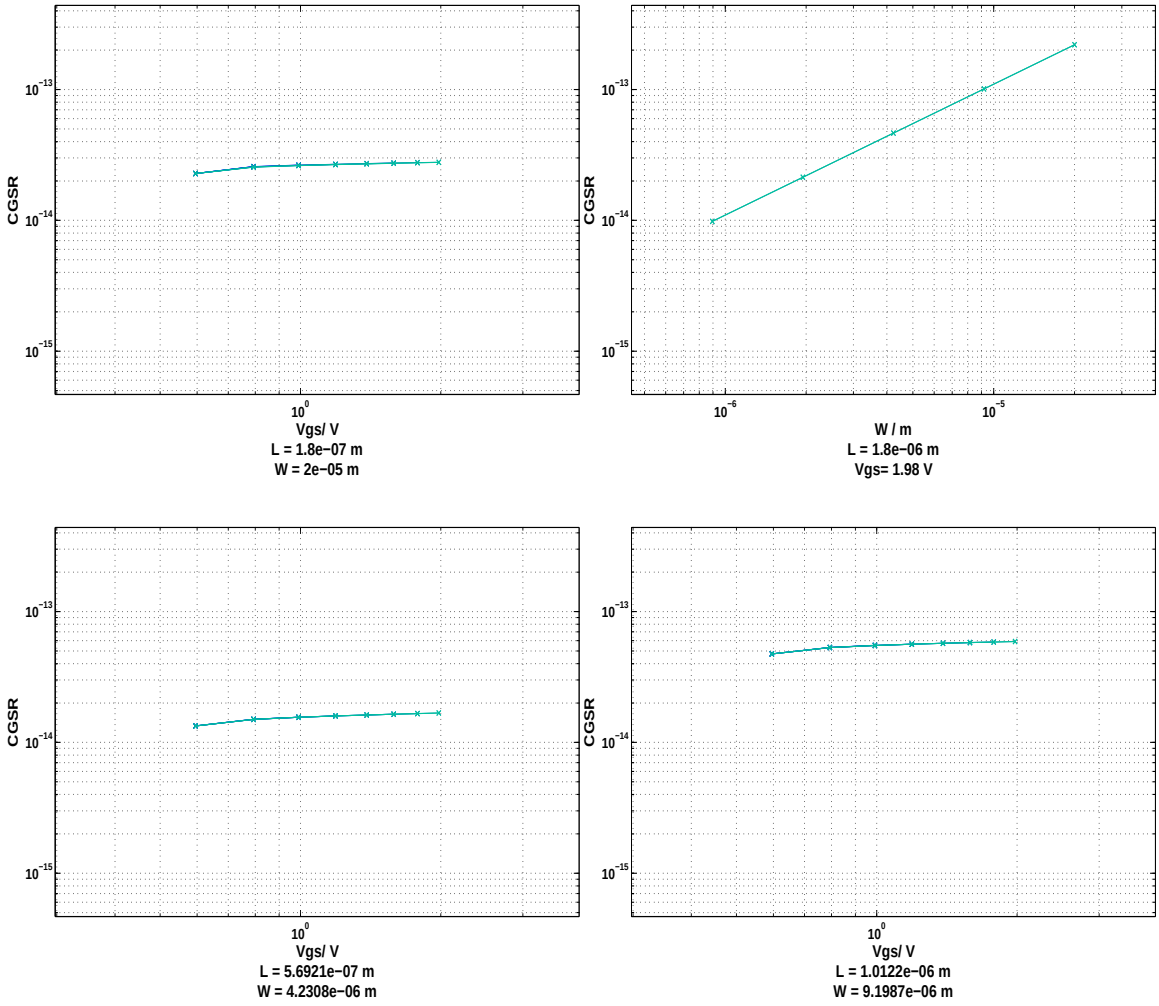


Figure 2-9: C_{gs} from an n-doped MOSFET, plotted against V_{gs} and W, with V_{gs} swept across range $[0.28V \quad 1.98V]$.

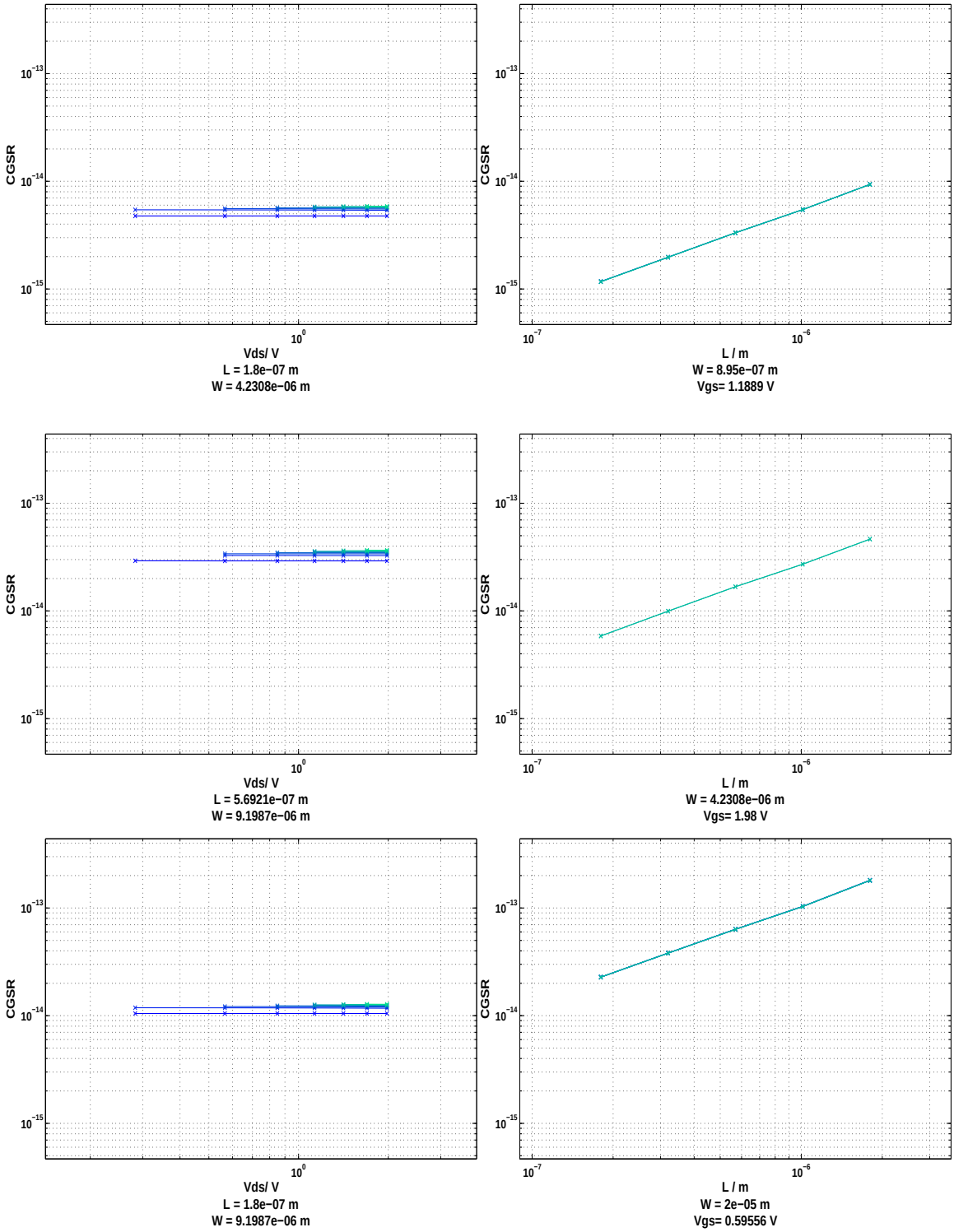


Figure 2-10: C_{gs} from an n-doped MOSFET plotted against **a)** V_{ds} with V_{gs} swept across range $[0.596V\ 1.98V]$ **b)** L with V_{ds} swept across range $[0.28V\ 1.98V]$.

begins with convex curvature, and as we increase W, new convex data appears at a higher range of I. In a sense, the current I is therefore split into two concave ranges, and we toggle between them by increasing the value of W.

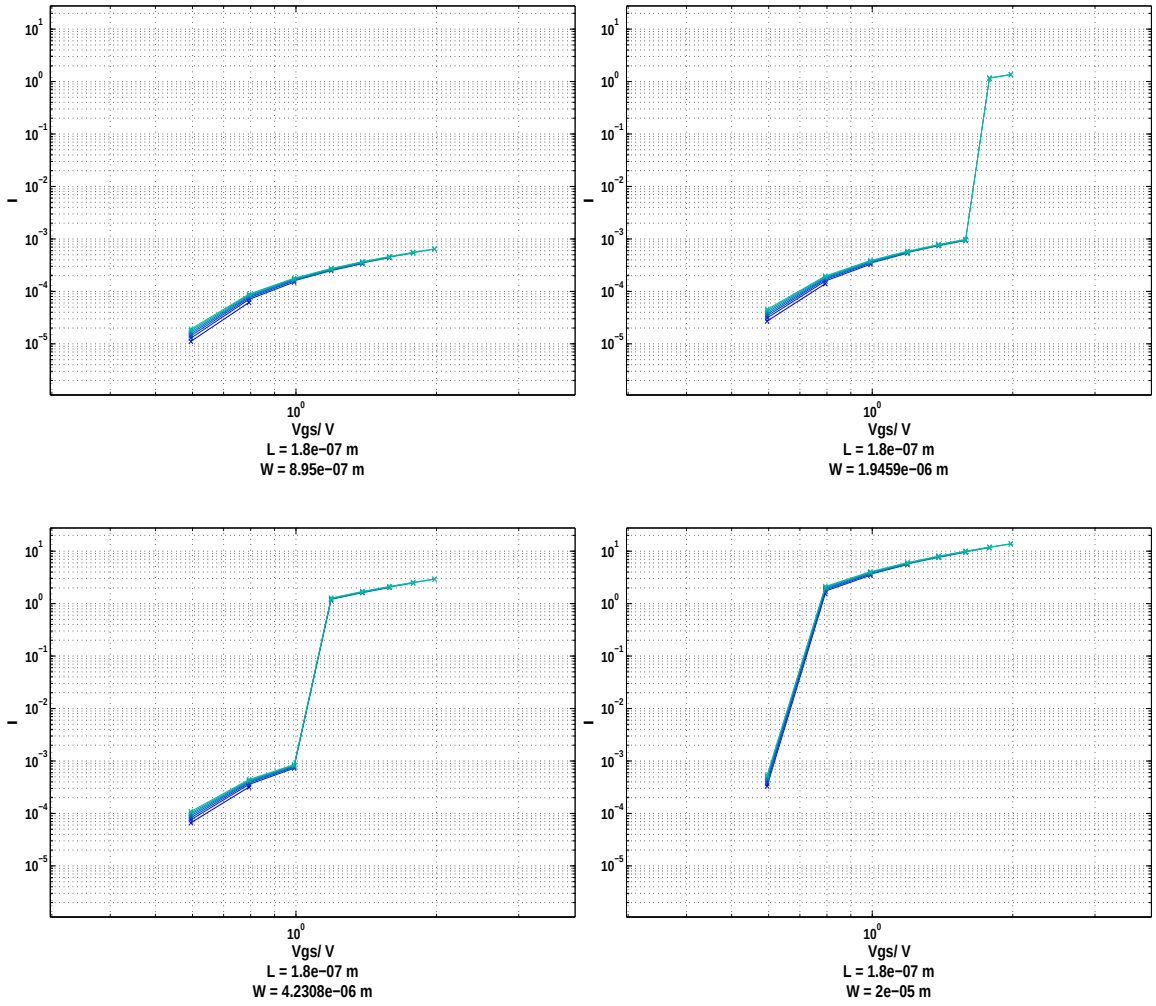


Figure 2-11: I from an n-doped MOSFET, plotted against V_{gs} for various values of W and L, and with parameter V_{ds} swept across range $[0.28V \quad 1.98V]$.

2.4 Posynomial Models

Returning to our original problem of circuit sizing, a geometric program is required to compute globally optimal values for the input parameters, such that they satisfy specific constraints on the output parameter values. A geometric program will find

the global optima in a matter of seconds by minimizing a target function subject to a group of constraints with a special form, shown below:

$$\begin{aligned} &minimize && f_0(x) \\ &subject\,to && f_i(x) \leq 1, \quad i = 1, \ldots, m \\ &&& g_i(x) = 1, \quad i = 1, \ldots, p \end{aligned} \tag{2.6}$$

Where f_0 and f_i are posynomial functions, g_i are monomial functions, and x are the optimization variables. A monomial is a function of the form:

$$g(x) = cx_1^{a_1}x_2^{a_2}\cdots x_n^{a_n} \quad \text{where } c > 0 \quad \text{and } a \in \Re \tag{2.7}$$

Where $x_1 \ldots x_n$ are n real positive variables.

In other words, a monomial function is required to have positive coefficients, but its exponents can take either positive or negative values. A posynomial, on the other hand, is a sum of monomial terms, i.e.:

$$f(x) = \sum_{k=1}^{K} c_k x_1^{a_{1k}} x_2^{a_{2k}} \cdots x_b^{a_{nk}} \quad \text{where } c_k > 0 \quad \text{and } a_k \in \Re \tag{2.8}$$

K is the maximum number of terms constituting the posynomial. According to this formulation, a monomial function is in fact a posynomial function with one term. Evidently, in order to obtain a well formed geometric program that optimizes the input parameters, it is necessary to form MOSFET models that are in fact posynomial functions. Therefore, we need to find posynomial functions of the input parameters that map to output parameters according to the following equations:

$$\begin{aligned} f(L, W, I, V_{ds}) \quad &= \quad \sum_{k=1}^{K} c_k L^{a_{1k}} \cdot W^{a_{2k}} \cdot I^{a_{3k}} \cdot V_{ds}^{a_{4k}} \\ \text{where } c_k > 0 \quad &\text{and} \quad a_k \in \Re \end{aligned} \tag{2.9}$$

where $f(x)$ is an output parameter dependent on a posynomial relation involving

input parameters A. Simiarly,

$$f(L, W, V_{gs}, V_{ds}) = \sum_{k=1}^{K} c_k L^{a_{1k}} \cdot W^{a_{2k}} \cdot V_{gs}^{a_{3k}} \cdot V_{ds}^{a_{4k}} \quad (2.10)$$
$$\text{where } c_k > 0 \quad \text{and} \quad a_k \in \Re$$

where the output parameter $f(x)$ depends on input parameters B.

2.5 Theoretical Bounds on Error

Given that our MOSFET models are posynomial functions, we have now reached a point where we can make predictions about how closely these models follow the empirical MOSFET data. But before we continue with our discussion, we must first ask ourselves: what are the curvature properties of a posynomial function on a logarithmic plot? The answer depends on the number of posynomial terms. Posynomials with one term, i.e. monomials, are linear in log-log space. Otherwise, any posynomial with $K > 1$, exhibits convex curvature on a log-log plot, as has been visualized graphically by Aggarwal[10].

Since our output parameters were orignally classified according to their curvatures in section 2.2, we can go through each different case and make general predictions about how well our posynomial models should perform, in terms of the maximum error as a measure of performance.

2.5.1 Concave Data

On enlisting a posynomial to fit data that is inherently concave in logarithmic space, the best fit, in terms of maximum error, is a monomial. We can deduce that monomials are the best fit for concave data by considering a two dimensional example, and extending the conclusions we derive to multiple dimensions. We begin as follows: A two dimensional function, concave in the interval defined by the domain $[x_1\ x_2]$ as shown in Figure 2-12, can be described as a function that lies above any line connecting two points on its curve, within the range defined by the interval. In other

words:

$$\begin{aligned} f(\lambda x_1 + (1-\lambda)x_2) &\geq \lambda f(x_1) + (1-\lambda) f(x_2) \\ \text{where } 0 \leq &\lambda \leq 1 \\ \text{and } x_1 \leq &x \leq x_2 \end{aligned} \tag{2.11}$$

Consider a linear fit of this concave interval, denoted by $\overrightarrow{ab}$, as shown in Figure 2-12. In general, the maximum absolute error between the concave function and its fitness line $\overrightarrow{ab}$ is defined as follows:

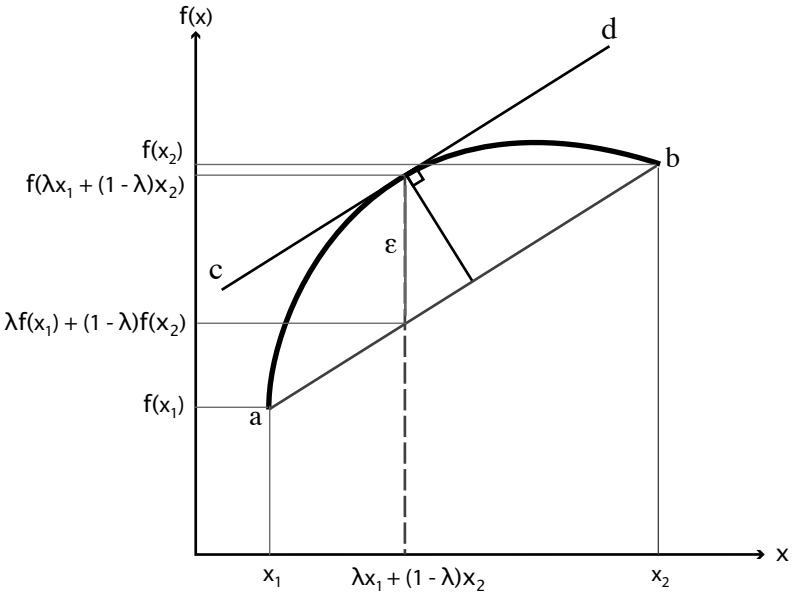


Figure 2-12: A concave function, and a linear fit of the function plotted on a log-log scale. The fit is a line anchored at the function's domain endpoints. The maximum error ε is generated by the fit.

$$\varepsilon_{max} = \max |f(x) - \tilde{f}(x)| \qquad \forall\, x \in [x_1\ x_2] \tag{2.12}$$

where $\tilde{f}(x)$ represents the model approximating the behavior of $f(x)$.

For the line joining the end points of the concave curve, we can assume that the maximum error occurs at some value of λ, and that the error can be expressed as:

$$\varepsilon = f(\lambda x_1 + (1-\lambda)x_2) - \lambda f(x_1) - (1-\lambda)f(x_2) \tag{2.13}$$

The absolute value disappears from the equation because, for a line joining the end points x_1 and x_2, the inequality 2.11 holds, and the difference between the two terms $f(x)$ and $\tilde{f}(x)$ is positive.

If we now take the derivative of the error ε with respect to λ, and set the result to zero, we can determine the value of λ which achieves the maximum error.

$$\frac{d\varepsilon}{d\lambda} = (x_1 - x_2) \cdot \frac{df(\lambda x_1 + (1-\lambda)x_2)}{d(\lambda x_1 + (1-\lambda)x_2)} - f(x_1) + f(x_2) = 0$$

$$\frac{df(\lambda x_1 + (1-\lambda)x_2)}{d(\lambda x_1 + (1-\lambda)x_2)} = \frac{f(x_1) - f(x_2)}{x_1 - x_2} \tag{2.14}$$

Note that the right hand side of Equation 2.14 actually corresponds to the slope of the fitness line $\overrightarrow{ab}$. In other words, the maximum error occurs at a value of λ where the derivative of the concave function at $x = \lambda x_1 + (1-\lambda)x_2$ equals the slope of the fitness line $\overrightarrow{ab}$. Therefore, the maximum error occurs at the point where a line $\overrightarrow{cd}$ parallel to the original fitness line $\overrightarrow{ab}$ is tangent to the concave curve. As a result, the maximum error is just the vertical difference between $f(\lambda x_1 + (1-\lambda)x_2)$ at the point of tangency and $\tilde{f}(\lambda x_1 + (1-\lambda)x_2)$ on the line, as shown in Figure 2-12.

Now that we can locate the data point with the maximum error between a fitness line and a concave function, our next step is to minimize this error. Suppose we shift and tilt the original fitness line upwards in an attempt to decrease the error, without worrying about maintaining the slope of the original line. Figure 2-13 shows that in the new scenario, the maximum error occurs at one of three points: either the point where a line parallel to the fitness line is tangent to the curve, or at one of the two endpoints. The errors are, once again, calculated by taking the vertical difference between the three aforementioned points on curve and their counterparts on fitness line, and the maximum error is the largest of the three errors, ε_1, ε_2 and

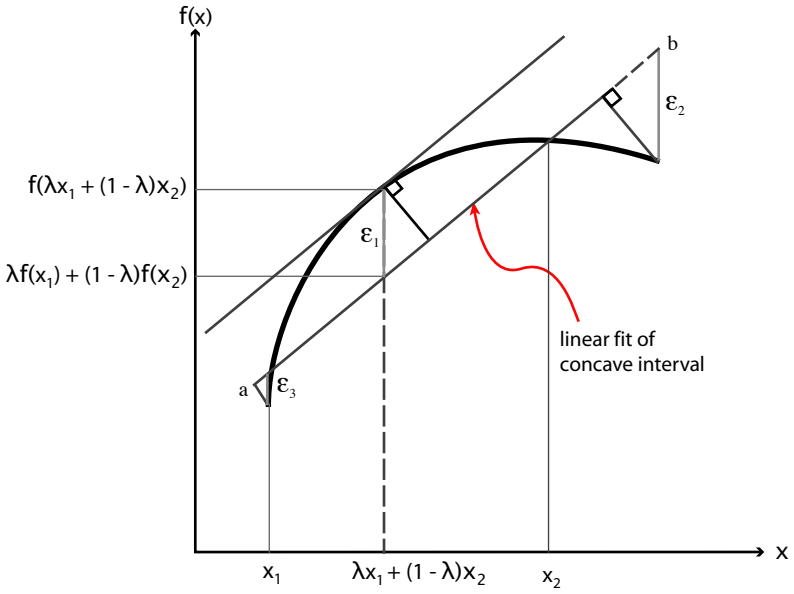


Figure 2-13: A concave function fitted with a new line, and the resulting errors ε_1, ε_2 and ε_3 generated by that fit.

ε_3. Figure 2-13 shows the largest error, ε_2, occurring at one of the end points.

Once again, we can minimize the error even further. We can pivot the fitness line on its left endpoint, and, as before, tilt it upwards in an attempt to achieve an equilibrium between errors ε_1 and ε_2. Figure 2-14 shows the result: an increase in ε_1 accompanied with a decrease in ε_2. The process can be repeated until the two errors are equal, producing the smallest maximum error.

On the other hand, we can accomplish a decrease in error by simply translating the original fitness line $\overrightarrow{ab}$ upwards along the perpendicular to the curve's tangent, rather than tilting it. In this case, we obtain Figure 2-15, which shows the smallest maximum error one can achieve using a fitness line parallel to $\overrightarrow{ab}$. As can be seen in the figure, the ε's are all equal in magnitude, hence producing an optimal maximum error. In fact, if we assume that the concave function is monotonic, we can argue that any line used to fit a concave interval will produce a maximum error equivalent to the error produced by a line parallel to a line through the endpoints of the concave interval.

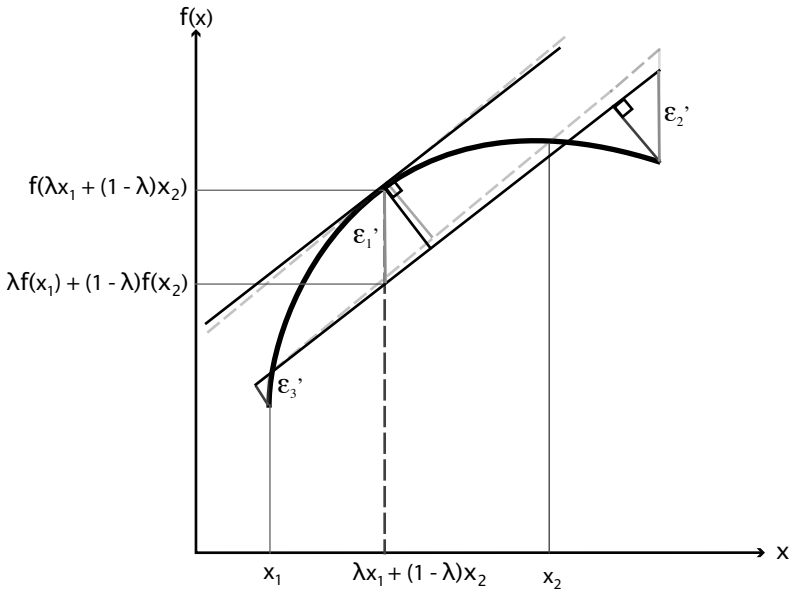


Figure 2-14: If we tilt our fitness line upwards, whilst pivoting on the leftmost endpoint, we achieve a smaller maximum error. The original fitness line and its associated errors are shown in faint grey.

Finally, if we attempt to improve the fitness obtained above by using a convex function rather than a line, Figure 2-16 shows the result. In order to maintain the original lower bound on the maximum error we had previously achieved by using a line, we would place the convex function such that we maintain the central error value, ε_2. The function, shown in Figure 2-16 would produce larger endpoint errors than its linear counterpart. Even if we reduce the curvature of the convex function, we will always do worse than a linear function. Therefore, a line is the only convex function producing the smallest maximum error when fitting a convex curve.

As a result of the previous discussion, we can conclude that monomials, which are linear in logarithmic space, form the best fit for concave data in logarithmic space.

But, one might argue, although our entire argument has been grounded in two dimensions, how does the discussion scale to multiple dimensions? Increasing the dimensionality of the input data and the posynomial model increases the degrees of freedom one can manipulate in order to decrease the maximum error. Regardless, if

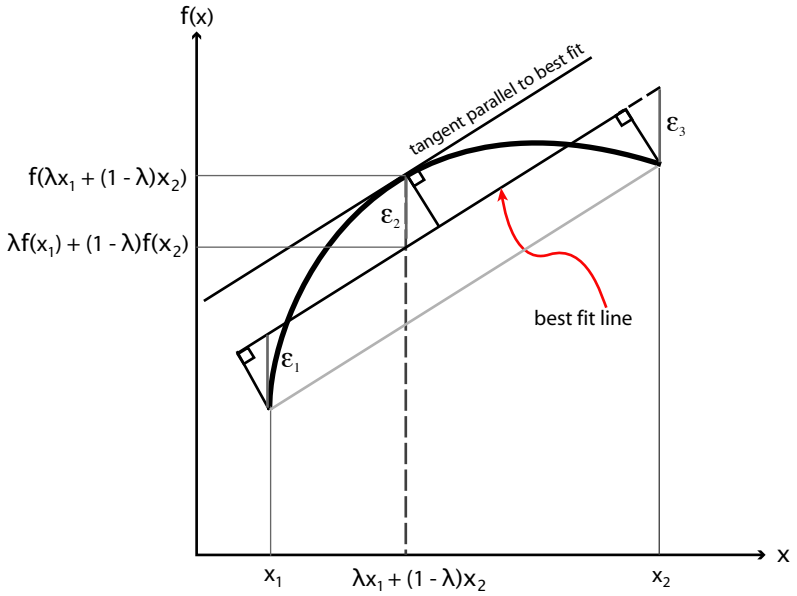


Figure 2-15: If we tilt our fitness line upwards, whilst pivoting on the leftmost endpoint, we achieve a smaller maximum error. The original fitness line and its associated errors are shown in faint grey.

the data is concave across all of its dimensions in logarithmic space, we predict that once again a hyperplane, i.e. a monomial, constitutes the best convex fit. Otherwise, data concave across some of its dimensions and not others would require a fitness curve that is more difficult to describe. Nevertheless, for data that exhibits some concavity, we predict the existence of a lower bound on the maximum error we can achieve. The lower bound arises because convex posynomials fit concave data rather poorly.

2.5.2 Data Containing an Inflection Point

In a manner similar to our previous discussion, let us begin by examining the two dimensional scenario. Figure 2-17 shows a function that contains an inflection point p, such that it is convex to the right of p, and concave to the left of it. Let us, for the sake of argument, split the function into two segments: concave and convex. From Section 2.5.1, we know that the best fit for the concave segment is in fact a line with

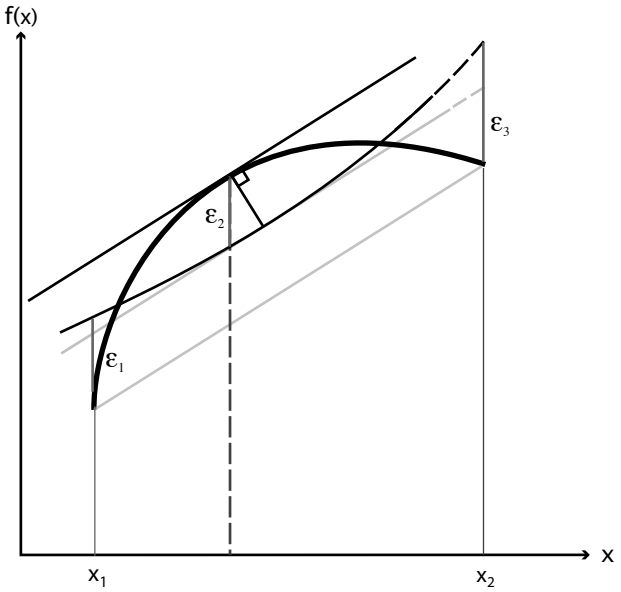


Figure 2-16: If we tilt our fitness line upwards, whilst pivoting on the leftmost endpoint, we achieve a smaller maximum error. The original fitness line and its associated errors are shown in faint grey.

the smallest maximum error within the segment itself. Similarly, the best convex fit for the convex curve to the right of p should naturally be curve itself over the specified interval. Figure 2-17 shows the resulting piecewise convex curve superimposed on top of the original function we are trying to fit. But, since posynomial models are monotonic and differentiable, the piecewise convex curve cannot be represented as a posynomial in logarithmic space. Nevertheless, it acts as a weak lower bound on the maximum error we can expect to achieve from any posynomial attempting to fit data with one or more inflection points.

Consequently, one can imagine a convex function in logarithmic space which smoothes out the discontinuity at the inflection point, producing a well formed posynomial that minimizes the maximum error, as shown in Figure 2-18. Conversely, the best fitness curve in terms of maximum error could be a line, as also seen in the figure. Regardless of what form the best fitness curve assumes, we predict that it will never do any better than the weak minimum bound proposed. A similar discussion

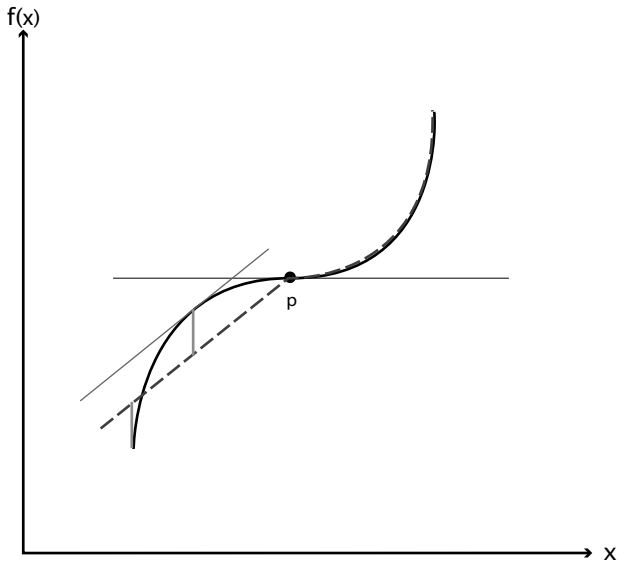


Figure 2-17: The best theoretical fits of a concave-convex function in log-log space.

can be used for data which is convex to the left of the inflection point, and concave to the right of it. For higher dimensionality, we still expect the lower bound to exist, because we cannot fit the concave sections of the data very well.

2.5.3 Linear Data

Obviously, a monomial in logarithmic space fits a line in two dimensional logarithmic space perfectly. As for higher dimensions, if the data is perfectly linear across all dimensions, we expect a monomial hyperplane in logarithmic space to constitute the best fit. Otherwise, we expect data which is linear in logarithmic space across some of its dimensions to exhibit a smaller maximum error than data that displays other forms of curvature across the same dimensions.

2.5.4 Convex Data

Finally, for data which is convex for low and high dimensions, we expect the best fit to be a posynomial, i.e. a curve which is convex in logarithmic space. Although all

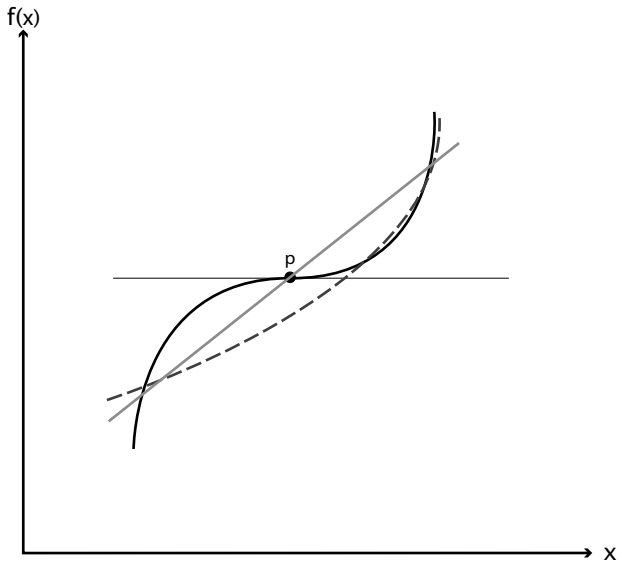


Figure 2-18: A monomial and posynomial fit of a concave-convex function in log-log space

posynomials are convex in logarithmic space, not all convex functions are posynomials. As such, we cannot specify a lower bound on the maximum error, since, depending on the form of the empirical data, the posynomial retains the potential to fit the data perfectly.

2.5.5 Final Remarks

Now that we have made general predictions about the nature of the posynomial models that will be used to fit the data, the next step is to determine whether the Genetic Algorithm does in fact find the best of these models. Once the Genetic Algorithm evolves a set of posynomial models to fit the data, we can once again use our visualization tool to acquire some intuition about the generated models in relation to the data, and its fitness.

Chapter 3

System Design

A circuit sizing system used to generate posynomial models consists of three main stages: evolution, optimization and finally validation, as shown in Figure 3-1. The evolutionary stage uses a Genetic Algorithm, (GA), combined with a convex optimization method such as QP or LP, to derive a posynomial model that emulates the empirical MOSFET data with a high degree of accuracy. These models can be reused for any circuit topology, provided it is consistent with the fabrication process that generated the models in the first place. The optimization stage then converts the posynomial for the consequent geometric program into a convex optimization problem, and then computes the global optima. Finally validation simulates the mosfet parameters in SPICE, finds the performance measurements, and calculates the error between the objective performance measurements and the simulated ones.

Combining the three stages, our overall circuit sizing system adheres to the following flow:

1. Initialization: the evolutionary stage during which the GA generates posynomial MOSFET models to populate a reusable "library".

2. Optimization: given the posynomial model library and a specific target topology, we iterate through the following steps:

   (a) Formulate the large and small signal MOSFET circuit equations in posynomial form given both the topology and the model library.

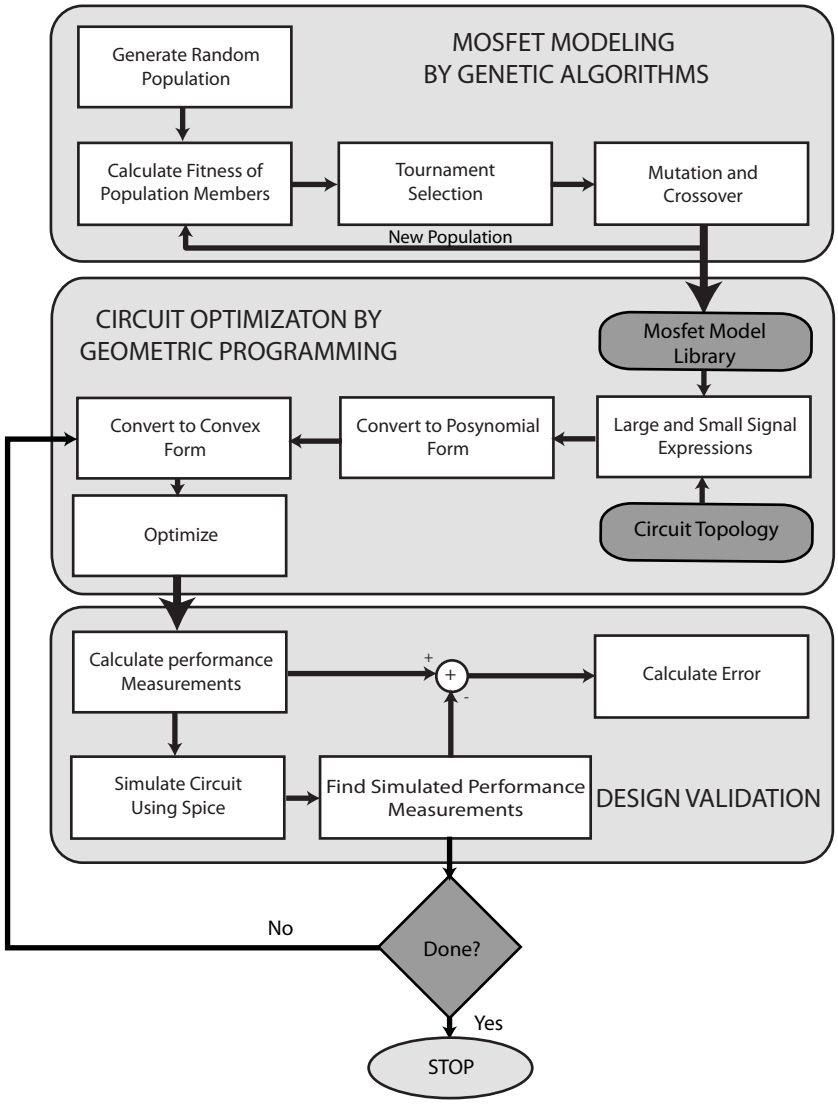


Figure 3-1: System Design

(b) Specify a set of objectives and constraints derived from the above circuit equations, to be optimized by the Geometric Program.

(c) Use the Geometric Program to solve for the optima.

(d) Validate the results in SPICE, and determine the error between the calculated and simulated parameters.

(e) Is the resulting validation error acceptable? If not, modify the Geometric Program objectives and constraints, and tighten the parameter range. Then iterate with the adjusted equations for a new set of optimal solutions. Otherwise, the flow is complete.

With the overall flow in mind, the following sections delve into the details of each separate stage, placing the most emphasis on the evolutionary stage.

3.1 Evolutionary Stage

The evolutionary stage uses a Genetic Algorithm [11] to generate posynomial models of the output parameters given either input parameters A or B. In order to evolve the the models effectively, the system uses a specific genotype representation characteristic to each different individual in the population. The genotype naturally encodes the exponents of each separate monomial term, such that we obtain the overall posynomial by summing together the terms. Although it encodes the exponents, the genotype has no knowledge of the terms' coefficients. The coefficients are calculated later using regression through Quadratic or Linear programming depending on which type of error our models are attempting to minimize, as seen in [10].

Finally, our genotype representation retains the ability to encode a variable number of terms to form the final posynomial. Hence, it includes a choice variable that equals one when a term should be used in the final posynomial representation. Otherwise, when the choice variable is zero, the exponents for the term are ignored. They are overlooked during both the phenotype generation and the coefficient regression stages. The genotype to phenotype mapping used in the algorithm is shown in

Figure 3-2.

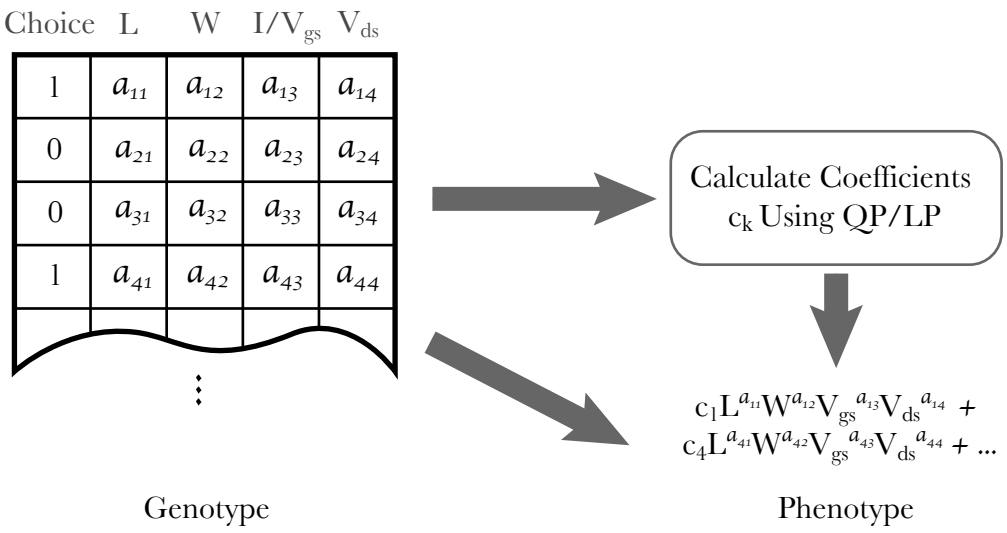


Figure 3-2: Genotype representation and its mapping to the phenotype.

As shown by the figure, the posynomial function relating the output performance parameters to the input parameters forms the phenotype of each individual. The genotype, on the other hand, is represented by a matrix with a fixed number of terms, with each row corresponding to a single term within the posynomial. The first column of the matrix represents the choice variable, and, since the choice variable allows us to have posynomials of varying length, the total number of terms in the posynomial can lie anywhere between 1 and *maxTerms*. The remaining four columns of the matrix each correspond to one of the four input parameters, with each cell containing the exponent. In order to calculate the phenotype, the coefficients of each term are calculated first by minimizing the error generated by a target fitness function, and then both the coefficients and exponents are combined together to form the phenotype.

The evolutionary stage begins by generating a random population, containing individuals with exponents randomly selected between $[minExponent, maxExponent]$. The choice variables are randomly initialized as well. The stage then calculates the fitness of each individual by computing the error between the individual's posynomial

model and the 900 empirical data points generated through SPICE simulations. But, before we calculate the fitness, the individual phenotypes need to be determined. Hence, we first optimize for the posynomials' coefficients, and then combine them with the exponents and input parameters to form the final posynomial representation. The QP or LP stages can produce coefficients that are zero valued, since their objectives are set to compute the optimal coefficients for each row. Therefore, the coefficient optimization stage performs a variant of feature selection.

After calculating the fitness of each individual, tournament selection determines which individuals are considered fit enough for propagation. The algorithm performs N tournaments with replacement, where each tournament randomly chooses s individuals from the tournament pool. The individual with the best fitness in the current tournament is selected for propagation and copied. All individuals are returned to the pool thereafter. Since we are selecting the elitist from every tournament, after N iterations there should be on average s copies of the best individual overall. Similarly, no copies of the worst individual should appear in the next generation.

After tournament selection has chosen fit parents for propagation, the algorithm generates the offspring by subjecting the parents to two variation operators. The first, probably more vital operator, is the crossover operator. The algorithm applies uniform crossover between individuals, where the crossover can only occur between terms, i.e. rows in the genotype matrix. In fact, the recombination process operates under the assumption that each term in the genotype, i.e. each monomial contributing to the overall posynomial, forms a building block [12]. The monomials, after all, represent low order components of high fitness that will, in theory, yield a higher fitness when added to other building blocks, thus generating a high fitness posynomial. Therefore each parent individual undergoes recombination with a probability of $p_{crossover}$, where the algorithm picks out another parent individual at random, and then uniformly chooses which rows of the offspring are copied from the first parent, and which rows from the second.

Once crossover is complete, the algorithm applies a mutation operator to perturb each exponent within the genotype matrix. We mutate the exponent by adding it

to a real valued number, drawn from a normal distribution with zero mean and a variance of λ. As the number of generations increases, the algorithm decreases the variance λ adaptively, in order to achieve two distinct variation stages: explorative and exploitative. During the explorative stage, the GA will be attempting to discover new building blocks of high fitness that will factor into the final solution, therefore the mutation variance λ is relatively large. As for the exploitative stage, since λ is small, the mutation injects random noise into the system in order to tweak exponents slightly in the hopes of improving individuals which have already attained a relatively high degree of fitness.

The algorithm makes a distinction between two different types of mutation applied on the parents to achieve the progeny. Rows, i.e. monomial terms, deemed unnecessary by Linear or Quadratic program are mutated with a higher probability than terms that have non-zero coefficients. Complying with our notion of building blocks, a coefficient of zero indicates that the corresponding exponents are not building blocks after all, and would be useless in finding a high fitness solution. Therefore, terms with zero coefficients are treated separately, and mutated with probability p_{zero_term}, while terms with non-zero coefficients are mutated with probability $p_{non\_zero\_term}$.

After the algorithm computes all the subsequent progeny, the new generation becomes ready to undergo another iteration of the GA. Evolution runs to completion (the maximum number of generations is attained), and from there the algorithm evaluates the fitness of all the individuals in the final generation, and chooses the best individual with the maximum fitness as the best posynomial model for the data.

It is important to note our trials were not intended to examine how well evolved posynomial models generalize to more than the 900 input data points, upon which we trained the GA. Rather, we are interested in determining how well the models perform on data they have been actively trained upon, in order to determine whether there exists a lower bound on the error generated by posynomial models of MOSFET data. Similarly, in order to compare different error metrics used to compute coefficients for each term within the posynomial, generalization becomes a secondary concern. Therefore, our entire discussion will focus primarily upon error arising from training

and testing on the 900 empirical data points, which were generated as described in Section 2.1.

As can be inferred from the problem statement in Section 1.3, we will be mainly concerned with the evolutionary stage of our system in order to answer the questions we have posed. Nevertheless, the consequent two sections will tackle the remaining two stages of the overall system for completeness.

3.2 Circuit Optimization Stage

Once we have arrived at a satisfactory set of posynomial models of the MOSFETs using the evolutionary stage, they constitute a library that can be reused for any circuit being designed with the same fabrication process. Given the model library, and a specific circuit topology, the optimization stage applies a geometric program to determine the optimal input parameters. The optimization stage creates a geometric program of objectives and constraints expressing small and large signal specifications derived using our MOSFET models. It then converts the geometric program to a convex optimization problem by taking the logarithm of the input variables, objectives and constraints in what it called the log-log transformation. Therefore the input variable x_i is replaced by $y_i = log(x_i)$, where $x_i = e^{y_i}$, and the geometric program is transformed to the following convex optimization problem:

$$\begin{aligned} &\textit{minimize} && \log f_0(x) \\ &\textit{subject to} && \log f_i(x) \leq 0, \quad i = 1, \ldots, m \\ &&& \log g_i(x) = 0, \quad i = 1, \ldots, p \end{aligned} \tag{3.1}$$

The constraints restricting the x's to positive values are implicit in the transformation. By taking the logarithm of a posynomial function, monomials become linear, and posynomials convex, facilitating convex optimization. The Geometric Program then solves the convex optimization problem to produce the globally optimum input parameter objectives.

3.3 Validation Stage

Once the circuit sizes have been optimized for the performance objectives via the Geometric Program, the validation stage determines how well the design truly meets imposed specifications using SPICE. First, the performance measurements given the optimal input parameters are calculated using analytical small and large signal equations. These are the objective performance measurements. Meanwhile, we use SPICE to simulate the resulting MOSFET with the given optimized input parameters to produce the simulated performance measurements. The validations stage then calculates the error between the objective and simulated performance measurements in order to evaluate the performance of our models. If the error is acceptable, then the process is complete. Otherwise, the algorithm returns to the optimizations stage for another pass, after appropriately modifying the objectives and constraints for the Geometric Program.

3.4 Monomial Fitting and Bisection

The GA proposed, although costly in terms of computational power, produces reusable models with greater accuracy, which should decrease the number of parameters that fail the validation stage. Yet, the fact that the GA is computationally intensive provokes a new question: is the GA better than existing, less computationally intensive methods in terms of accuracy? Since the GA's goal requires generating posynomial models that minimize the maximum error, we need to select some form of benchmark against which we can compare the performance of the GA's models and deduce the answer to our question. The posynomial models generated by the GA will be compared with a more straightforward monomial fitting (MF) algorithm, proposed by Boyd [8]. A monomial fit of a function $f(x_i)$ is represented by:

$$\tilde{f}(x_i) = c x_1^{a_1} x_2^{a_2} \cdots x_n^{a_n} \text{ where } c > 0 \tag{3.2}$$

For the benchmark, we are going to minimize the maximum absolute relative error, given by:

$$\varepsilon_{rlae} = \max_{i=1,\cdots,N} \frac{|f(x_i) - \tilde{f}(x_i)|}{f(x_i)} \tag{3.3}$$

In order to find the values of c and $a_1, \cdots, a_n$ from Equation 3.2, which minimize the above relation, we should first transform the equation into a linear program and then solve a regularized version of the initial problem. According to Boyd, we should first choose a value t representing the target maximum error, such that:

$$\frac{|f(x_i) - \tilde{f}(x_i)|}{f(x_i)} \leq t, \quad i = 1, \cdots, N \tag{3.4}$$

We can then determine whether the inequality 3.4 is feasible given the current value of t by choice of c and $a_1, \cdots, a_n$, such that $0 < t < 1$. But in order to do that, the inequality should be converted to:

$$f(x_i) \cdot (1 - t) \leq \tilde{f}(x_i) \leq f(x_i) \cdot (1 + t), \quad i = 1, \cdots, N \tag{3.5}$$

Then by taking the logarithm of both sides, such that $y_i = \log x_i$, we obtain:

$$\log\left(f(x_i) \cdot (1 - t)\right) \leq \log c + a_1 y_1 + \cdots + a_n y_n \leq \log\left(f(x_i) \cdot (1 + t)\right) \tag{3.6}$$

The equations 3.6 can be solved for c and the a_i's using linear programming, and we can use bisection to determine the value of t such that the inequalities remain feasible. Bisection, as seen in [7], operates by assuming the problem is feasible, and starting within an interval $[l\ u]$ within which we know the optimal maximum error t should exist. We then choose a t at the midpoint of this range, such that $t = \frac{l+u}{2}$. Given the new t, we solve the complex feasibility problem, and if the problem is unfeasible, then the optimal t should exist in the upper half of the interval. If the problem is feasible on the other hand, then t should exist within the lower half of the interval. Bisection therefore updates the interval and iterates once again, until the width of the interval containing the optimal value of t falls beneath an acceptable threshold. Finally, we can account for the cases where our error t is large and hence

the upper bound $l > 1$. In this case, the left hand side of inequality 3.6 disappears and we have a simpler optimization problem.

In order to penalize large exponents, we added a regularization term to the objective of our original Linear Program in the form of:

$$\mu \sum_{i=1,\cdots,N} a_i \tag{3.7}$$

The regularization term attempts to minimize the sum of all the exponents, which in turn minimizes the values of each different exponent subject to the aforementioned constraints. The regularization ensures that the monomial is composed of exponents within a reasonable range.

As demonstrated, monomial fitting using bisection constitutes a straightforward rather light-weight model that can be used to emulate MOSFET data. We can therefore evaluate the performance of GA generated posynomial models against the benchmark provided by monomial fitting which attempts to minimize the maximum error. If we obtain a notable improvement in maximum error using posynomial models generated by the GA system described in this chapter, then the more time consuming algorithm will be more favorable than mononomial fitting since it yields more accurate results.

Chapter 4

Error Metrics

In the previous chapter, we discussed how the GA in the evolutionary stage evaluated the fitness of each individual in order to propagate the elitist onto the next generation. We also discussed the QP/LP optimization used for obtaining the posynomial coefficients for each term. The current chapter discusses the error metrics used for evaluating both the fitness of individuals, and as a minimization target for the optimization problems used to determine the values of the coefficients.

4.1 Maximum Relative Error

As stated previously, analog designers are interested in minimizing the maximum error between the posynomial MOSFET models, and the actual empirical data obtained through simulating the MOSFETs themselves. Although already shown in section 3.4, the relative error is given by the following equation:

$$\varepsilon_{rlae} = \max_{i=1,\cdots,N} \frac{|f(x_i) - \tilde{f}(x_i)|}{f(x_i)} \tag{4.1}$$

Since the output parameters span over several orders of magnitude, it becomes imperative that we use relative error in order to normalize the values of the output parameters. Therefore, in order to determine the coefficients for the posynomial phenotype of each individual, we would use a Linear Program (LP) that minimizes the

above error to solve for the coefficients, subject to the constraint that the coefficients need to be greater than 0. After calculating the coefficients, we would report the maximum relative error to the selection algorithm in the GA, and based on the error, the best individual of each tournament is selected.

4.2 Mean Relative Error

Although analog designers are interested in reducing the overall maximum error, the mean error between the posynomial models and the empirical MOSFET data remains a vital concern. Therefore, another error metric which could be used as a fitness measurement for tournament selection, and also as a method for selecting posynomial coefficients is the relative mean squared error, which is given by the following equation:

$$\varepsilon_{rlse} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\frac{f(x_i) - \tilde{f}(x_i)}{f(x_i)}\right)^2} \tag{4.2}$$

In this case, we use Quadratic Programming, i.e. QP in order to find optimal posynomial coefficients given the exponents, which minimize the relative mean squared error, subject to the constraint that all the coefficients need to be positive. Once we determine the coefficients, the selection process determines the fitness of individuals based on the relative mean squared error, and chooses the best fitness individuals accordingly.

4.3 Adaptive Error

Rather than attempting to minimize either relative mean squared error, or relative maximum error between the models and the empirical data, we would like to attempt to control the tradeoff between obtaining a small mean error or maximum error. Therefore, we will attempt to minimize the error adaptively, such that during earlier generations, we place more emphasis on minimizing the mean error, whereas in later generations we focus on minimizing the maximum error. We achieve the aforemen-

tioned goal by introducing a new adaptive fitness measurement used for the selection of individuals during the tournaments. The new adaptive fitness measurement is given by the following:

$$\varepsilon_{adapt} = w_a \varepsilon_{rlse} + w_b \varepsilon_{rlae} \tag{4.3}$$

Here, the weights w_a and w_b are given by:

$$w_b = \frac{g_{no} - 1}{w_f} \qquad w_a = 1 - w_b \tag{4.4}$$

Where g_{no} denotes the current generation number during the GA iteration, and w_f denotes the weight factor, a constant which controls how fast we switch emphasis from mean error to maximum error during the execution of the algorithm. For a large w_f, the transition of focus between mean to maximum error occurs more slowly over the generations. By reporting an adaptive measure of the fitness, using both the mean and maximum error as components of the overall fitness values, the GA should in theory filter out individuals that exhibit a low fitness in mean and/or maximum errors, and retain individuals which have relatively low maximum and mean errors.

By altering the fitness measurement to incorporate both mean and maximum errors, there exists three different permutations of which coefficients to use for the posynomial models, and which fitness measurement to report given two different convex optimization algorithms for the coefficients.

1. In the first case, we can use both a QP which minimizes mean error, and an LP which minimizes the maximum error to find the coefficients for the calculation of ε_{rlse} and ε_{rlae} respectively. We can then calculate the adaptive error based on the two different values of error obtained above, and obtain ε_{adapt_qlp}. One pitfall for this method would be to assume that the error reported to the GA corresponds to one specific model, whereas it corresponds to two different models with two different coefficients for the same exponents in the genotype. Therefore, although we choose an error metric based on both, we decided to choose the coefficients determined by the QP as the model to adopt after the final iteration of the GA.

2. Rather than use two different optimization algorithms to obtain the values of the coefficients, we only use a QP aimed at minimizing the mean error ε_{rlse}, and then calculate both the maximum and mean errors produced by the resulting choice of coefficients. The error ε_{adapt_qp} is obtained by adaptively adding the mean and maximum errors obtained from using the coefficients determined by the QP, and is then reported to the GA for individual selection.

3. Finally, we can use an LP which minimizes the maximum error, ε_{rlae}, to obtain the coefficients for the posynomial phenotype, and then use the coefficients to calculate our measures for mean and maximum error. The error measurement ε_{adapt_lp} encompasses both the mean and maximum errors calculated as a result of the LP optimized coefficients, and is consequently reported to the GA so that it can perform individual selection accordingly.

Now that we have obtained several methods for determining the coefficients and measuring the fitness of the generated posynomial models, our next step is to evaluate the performance of these different methods, and determine which of them is best in terms of meeting some designer specifications on mean and maximum error.

Chapter 5

Experiments and Results

In light of the theoretical bounds we posed on the error obtained from posynomial modeling, and the various combinations of error metrics we can use as a measurement of fitness, the current chapter discusses the experimental setup we used to test the theories we postulated, and to compare the performance of the different types of models. We performed three collections of experiments, each concerned with tackling a specific question posited earlier. The first, Experiments A, compare the different types of error metrics we used as fitness measurements for propagating individuals in the GA. The comparison is performed across 4 of the output parameters we had previously performed case studies on in Chapter 2. As for the second collection of experiments, Experiments B, they collect data from the five output parameters g_m, g_{ds}, r_0, C_{gs} and I which have been modeled using a GA which optimizes for ε_{rlae}. The data is then plotted using the visualization tool described in section 2.2. Finally, Experiments C generate posynomial models using a GA which optimizes for ε_{rlae}. They gather results corresponding to all the output parameters across input parameters A and B, for n and p-doped MOSFETs. They then compare the values of ε_{rlae} obtained from the GA with those obtained from Boyd's monomial fitting.

5.1 Experimental Setup

For all collections of experiments, A through C, we used a particular setting of experimental constants, e.g. the number of generations, in order to facilitate the smooth execution of each run of the GA. Unless explicitly stated, the use of the constants described below was maintained across all of the experiments. For each set of experiments, we performed 5 different GA runs for each of the target output parameters. The GA iterated for 500 generations per run, with a population of 50 individuals. It randomly initialized the seed population. The choice variables were therefore randomly chosen so that the number of posynomial terms retained an average of 3 terms, and a maximum of 5 terms. The exponents were allowed to assume any real value between [−5 7].

Experimental Constant	Value
N_{pop}	50
s	6
$Runs$	5
$generations$	500
$minExponent$	-5
$maxExponent$	7
$maxTerms$	5
$avgTerms$	3
$p_{crossover}$	0.5
λ initial	2
λ rate	Halved every 35 generations
p_{zero_term}	0.7
$p_{non\_zero\_term}$	0.3
w_f	250

Table 5.1: Values of experimental constants used for the GA runs

Depending on the optimization applied to calculate the coefficients, and the error metric used to calculate the fitness measurement, the weight factor $w_f = 250$ was used to adaptively change the dependency of the fitness on mean or maximum error. Individual selection occurred within tournament sizes of 6 individuals, after which crossover was applied with probability $p_{crossover} = 0.5$, and mutation of each exponent was performed with probability $p_{zero\_term} = 0.7$ for terms with zero coefficients, and

$p_{non\_zero\_term} = 0.3$ for terms with non-zero coefficients. Finally, the mutation variance λ was initialized to 2, and was halved every 35 generations. Table 5.1 summarizes the experimental constants we used for the experiments.

In the case of the benchmark, monomial fitting, Table 5.2 shows the experimental constants used when running the bisection algorithm.

Experimental Constant	Value
μ	1
Lower Bound	0
Upper Bound	5
Threshold	0.001

Table 5.2: Values of experimental constants for monomial fitting using bisection

5.2 Experiments A: Error Metrics for Max-Mean Error Tradeoff

We performed five sets of contiguous experiments, each corresponding to a different error metric as described in Chapter 4. The five sets of experiments therefore used ε_{rlae}, ε_{rlse}, ε_{adapt_qlp}, ε_{adapt_qp}, and ε_{adapt_lp} respectively as a fitness measurements for propagating the best individuals onto future generations. In order to obtain statistically viable results, the GA was performed for 30 runs, using the 5 different error metrics, in order to model the output parameters g_m, g_{ds}, r_0, and C_{gs} using input parameters B. Therefore, we performed 5×30 runs per output parameter, each set of 30 pertaining to a different model based on the error metric. We then selected two individuals from each set of runs: the first is the best individual overall in terms of maximum error, the second has the best overall mean. After extracting these fit individuals, each of their mean and maximum errors were measured and recorded in order to produce the tradeoff plots shown in Figures 5-1 and 5-2. In the plots, RLAE denotes GA model which minimizes ε_{rlae}, RLSE the GA model minimizing ε_{rlse}. Adapt-QLP-RLAE and Adapt-QLP-RLSE are errors arising from

individuals with the two different sets of coefficients obtained from the GA which minimizes ε_{adapt_qlp}. The first, Adapt-QLP-RLAE denotes the error obtained when the coefficients are chosen as a result of the LP, while Adapt-QLP-RLSE chooses the coefficients from the QP. Finally, Adapt-QP and Adapt-LP denote the models with error metrics ε_{adapt_qp} and ε_{adapt_lp} respectively.

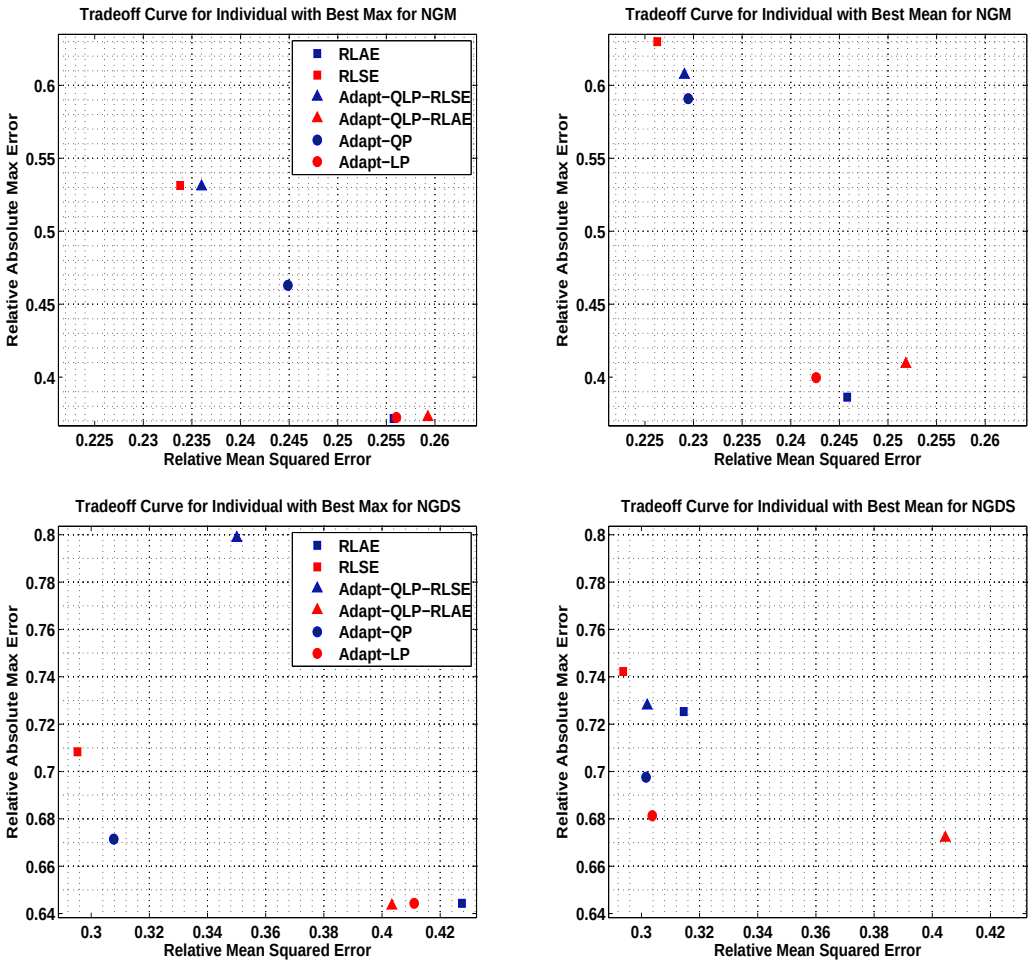


Figure 5-1: Tradeoff plots for parameters g_m, g_{ds}.

By plotting the maximum absolute relative error against the relative mean squared error for the individuals, we can quantify the tradeoff demonstrated by each of the five different models in terms of maximum and mean errors. As seen across the four output parameters, model RLSE, which results from a GA attempting to minimize ε_{rlse}, naturally exhibits the largest maximum error. Because the model disregards

maximum error when it selects individuals for propagation onto the next generation, the maximum error of its final answer will be quite large, whereas the mean error is the smallest. We will therefore use the RLSE points as a benchmark for the smallest mean error we can obtain using our models.

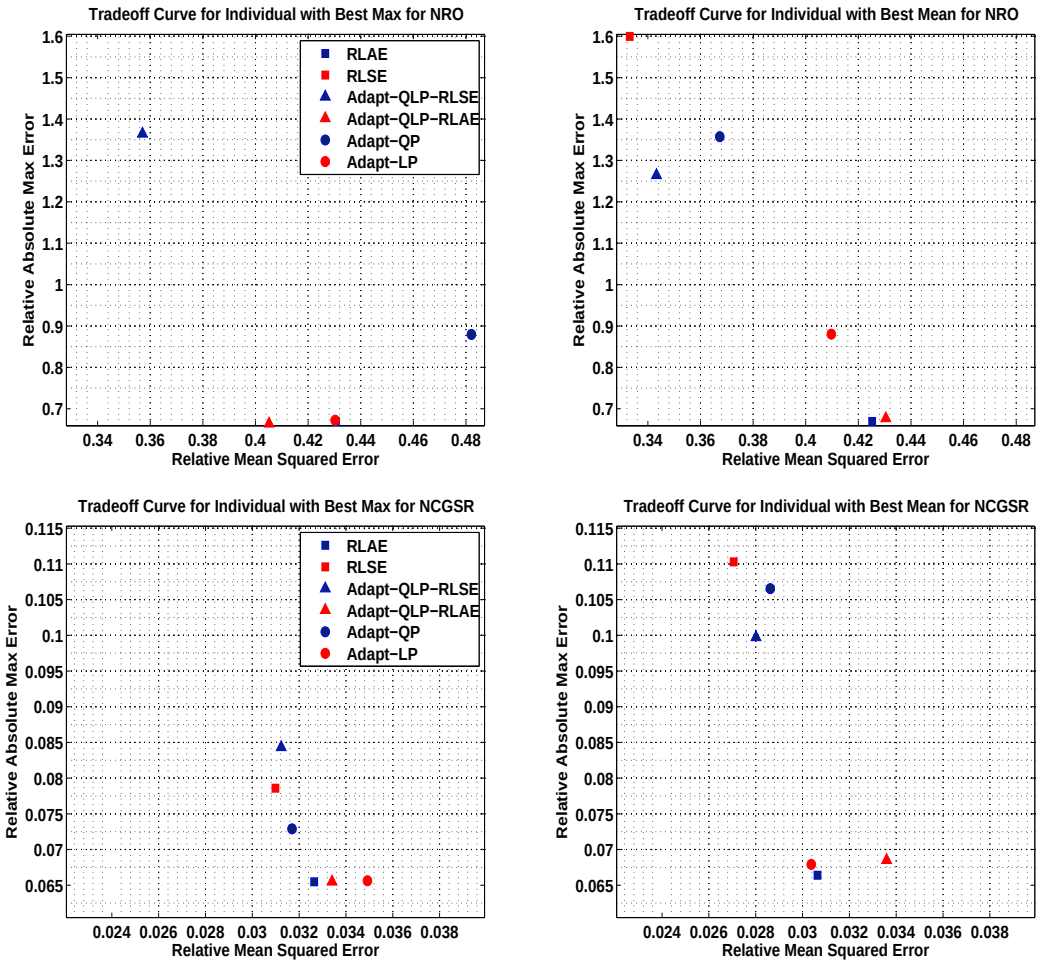


Figure 5-2: Tradeoff plots for parameters r_0 and C_{gs}.

We take our comparison one step further by extracting all the nondominated points that contribute to the Pareto optimal set [13] for each parameter, as shown in Table 5.3. The points are nondominated because, subject to a small threshold of leniency, there exists no other points with either a lower mean error or a lower maximum error. The resulting Pareto fronts are plotted in Figure 5-3, where the two fronts correspond to selecting the best individual based on the smallest maximum

Parameter	Optimal Set for Individual with Best Max			Optimal Set for Individual with Best Mean		
	Max Error	Mean Error	Algorithm	Max Error	Mean Error	Algorithm
g_m	0.3717	0.2557	RLAE	0.3862	0.2458	RLAE
	0.5313	0.2338	RLSE	0.6299	0.2263	RLSE
	0.3726	0.2593	Adapt-QLP-RLAE			
	0.3723	0.2560	Adapt-LP			
g_{ds}	0.7083	0.2953	RLSE	0.7422	0.2937	RLSE
	0.6433	0.4034	Adapt-QLP-RLAE	0.6720	0.4045	Adapt-QLP-RLAE
	0.6443	0.4111	Adapt-QP			
	0.6443	0.4275	RLAE			
r_0	1.3644	0.3570	Adapt-QLP-RLSE	0.6692	0.4253	RLAE
	0.6642	0.4052	Adapt-QLP-RLAE	1.5995	0.3331	RLSE
C_{gs}	0.0655	0.0326	RLAE	0.0664	0.0306	RLAE
	0.0786	0.0310	RLSE	0.1103	0.0271	RLSE
	0.0655	0.0334	Adapt-QLP-RLAE			
	0.0656	0.0349	Adapt-QP			

Table 5.3: Pareto optimal sets for the four different parameters, and the algorithms which generated them.

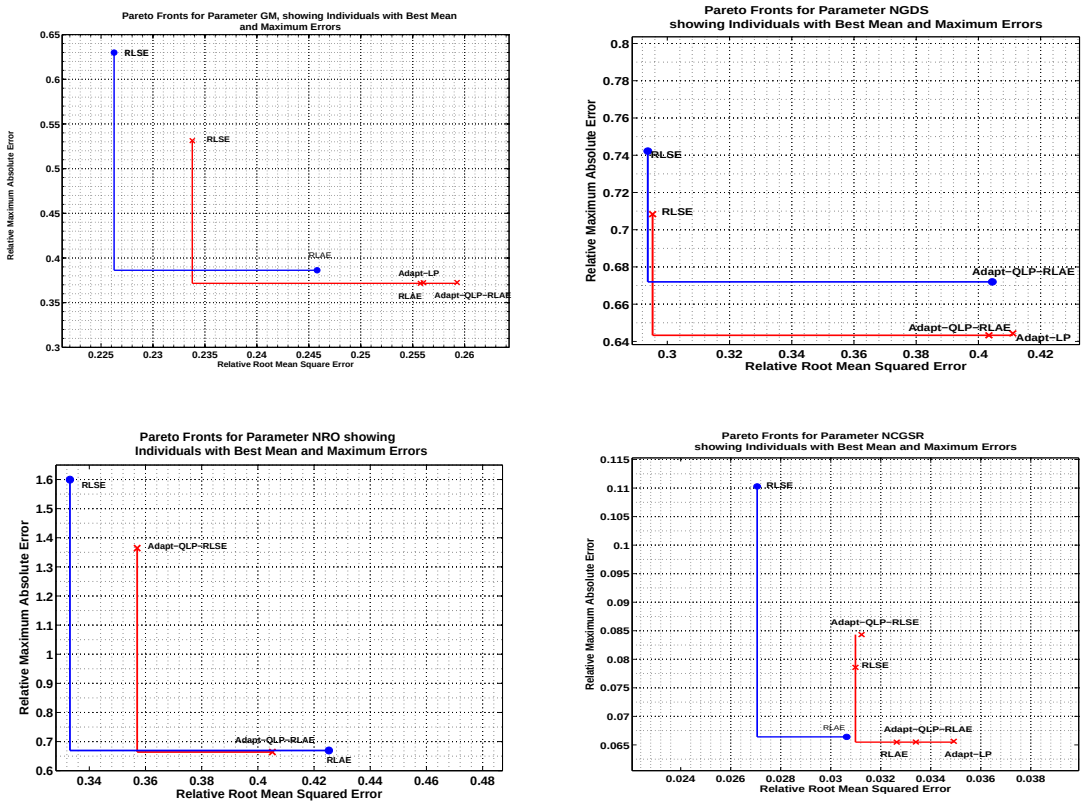


Figure 5-3: Pareto fronts for the parameters g_m, g_{ds}, r_0 and C_{gs}. Front for individual with best mean is shown in blue, while individual with best max is shown in red.

or mean errors. From the table and plots, we can deduce that model RLAE, corresponding to the GA which minimizes ε_{rlae}, in addition to producing the smallest maximum error with respect to all the other models, only compromises the mean error very slightly. In other words, if we choose model RLAE, the mean error it produces is about 5% larger than the smallest mean error we can obtain overall when using the RLSE model. In fact, the adaptive models do not seem to incur any substantial benefit in terms of improving the mean error whilst keeping the maximum error more or less constant. The Adapt-LP model is the only model which does slightly better on mean than the RLAE model, whilst maintaining the maximum error relatively the same. Nevertheless, the improvement in mean error does not prove substantial enough to warrant the use of a more complicated model over RLAE. Therefore, for the purposes of future discussion, the RLAE model, which minimizes ε_{rlae} was chosen for the GA, and the individual with the best maximum fitness was selected out of the GA runs.

5.3 Experiments B: RLAE Model Error for g_m, g_{ds}, r_0, C_{gs} and I

Now that we have chosen an error metric ε_{rlae} which minimizes the maximum error, whilst maintaining the mean error at a reasonable value, we can now revisit the parameters we had performed case studies upon in Chapter 2. To obtain the results for the following discussion, we ran a GA minimizing ε_{rlae} for five runs per output parameter obtained from an n-doped MOSFET, and then generated the values $\tilde{f}(x_i)$ from the best resulting model. We then used our visualization tool to plot the values of $\tilde{f}(x_i)$ predicted by the models on the same axes showing the simulated values of the output parameters, $f(x_i)$. The model generated data is shown in decreasing shades of red, while the empirical data is still shown in blue. We also incorporated the maximum absolute relative error for the data shown in the plot on the graphs, where the green line depicts the maximum overall error between the shown points across

the entire data range, whereas a black line depicts the maximum error for the current range designated by the plot.

5.3.1 Posynomial Model of the Transconductance g_m

Figures 5-4 and 5-5 show the model outputs for the parameter g_m. We can see that the posynomial model for g_m, which attempts to minimize the maximum error given by ε_{rlae}, exhibits curvature properties we had previously predicted in Section 2.5. The best posynomial fit for the concave dimensions of g_m are lines in logarithmic space, which can be seen for the dimensions V_{gs} and L. Similarly, the evolved posynomial generates a line to fit the linear dimension of g_m, W, in logarithmic space. Finally, the posynomial model fits the slightly convex dimension, V_{ds} with a line in logarithmic space as well. Therefore, the resulting posynomial is in fact a mononomial, producing a hyperplane in logarithmic space. One thing to note is that the dimensions are not independent of one another, and that the best fit lies across all the dimensions, resulting in a less than best fit if we slice along only one dimension. The maximum error overall occurs between two points on the plot of g_m versus L, where the models uses a linear fit to model a concave curve. In other words, the evolved posynomial model of g_m reinforces the notion of a lower bound on error, resulting from the fact that g_m is concave across two of its dimensions.

5.3.2 Posynomial Model of the Output Conductance g_{ds}

Figures 5-6 and 5-7 show the outputs of the GA evolved posynomial model superimposed over the original empirical data for the parameter g_{ds}. In a manner similar to the parameter g_m, the posynomial model was linear in logarithmic space with respect to the input paramter W. In other words the resulting posynomial modeled the linear dimension using a linear fit in logarithmic space as predicted. On the other hand, g_{ds} exhibits an inflection point with respect to V_{gs} and L. For both these input parameters, the posynomial model uses a convex fit in logarithmic space to emulate the empirical data. g_{ds} is slightly convex with respect to L, despite the presence of

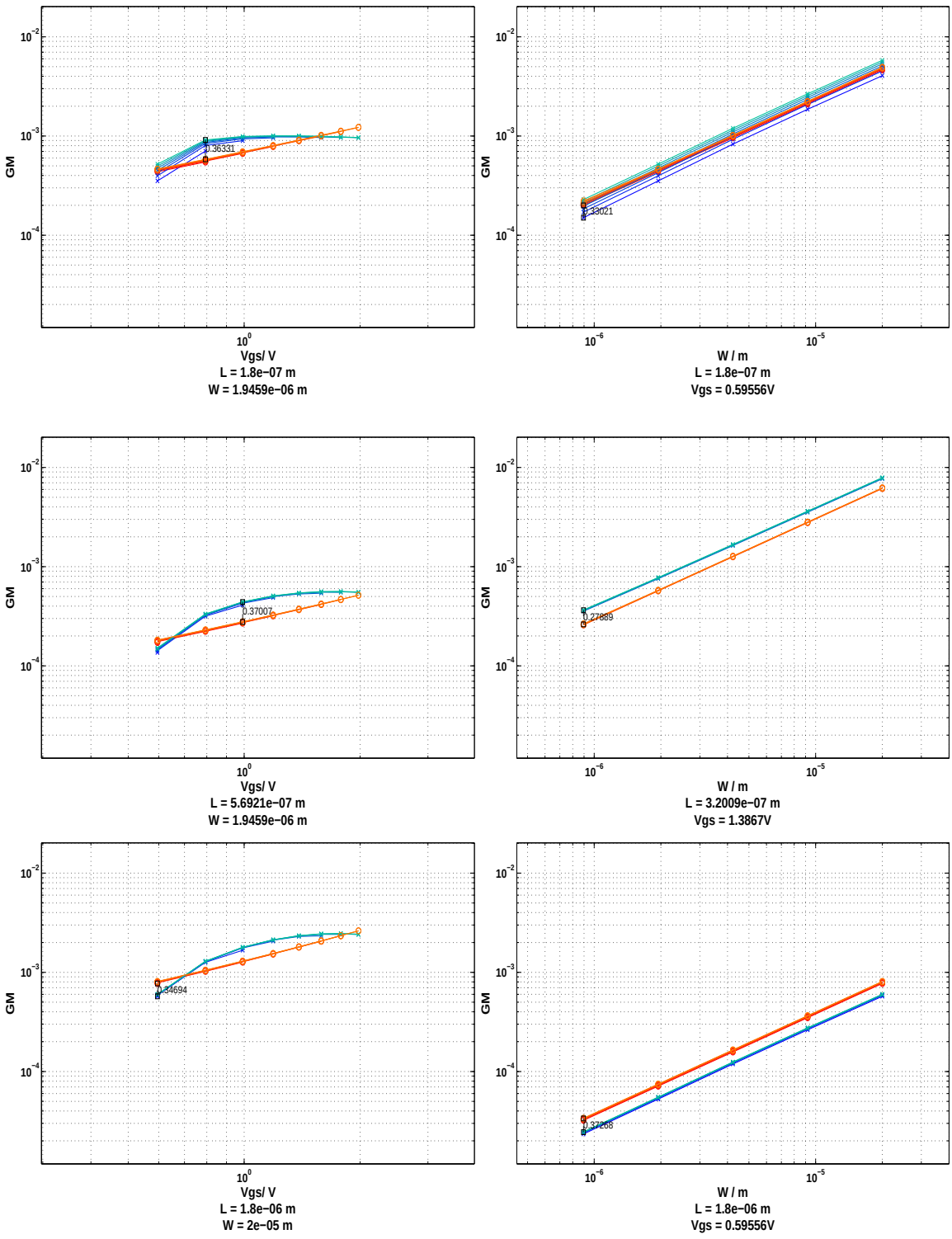


Figure 5-4: g_m plotted for an n-doped MOSFET, with input parameters L, W, I and V_{ds}. The first column shows g_m against V_{gs}, and second against W

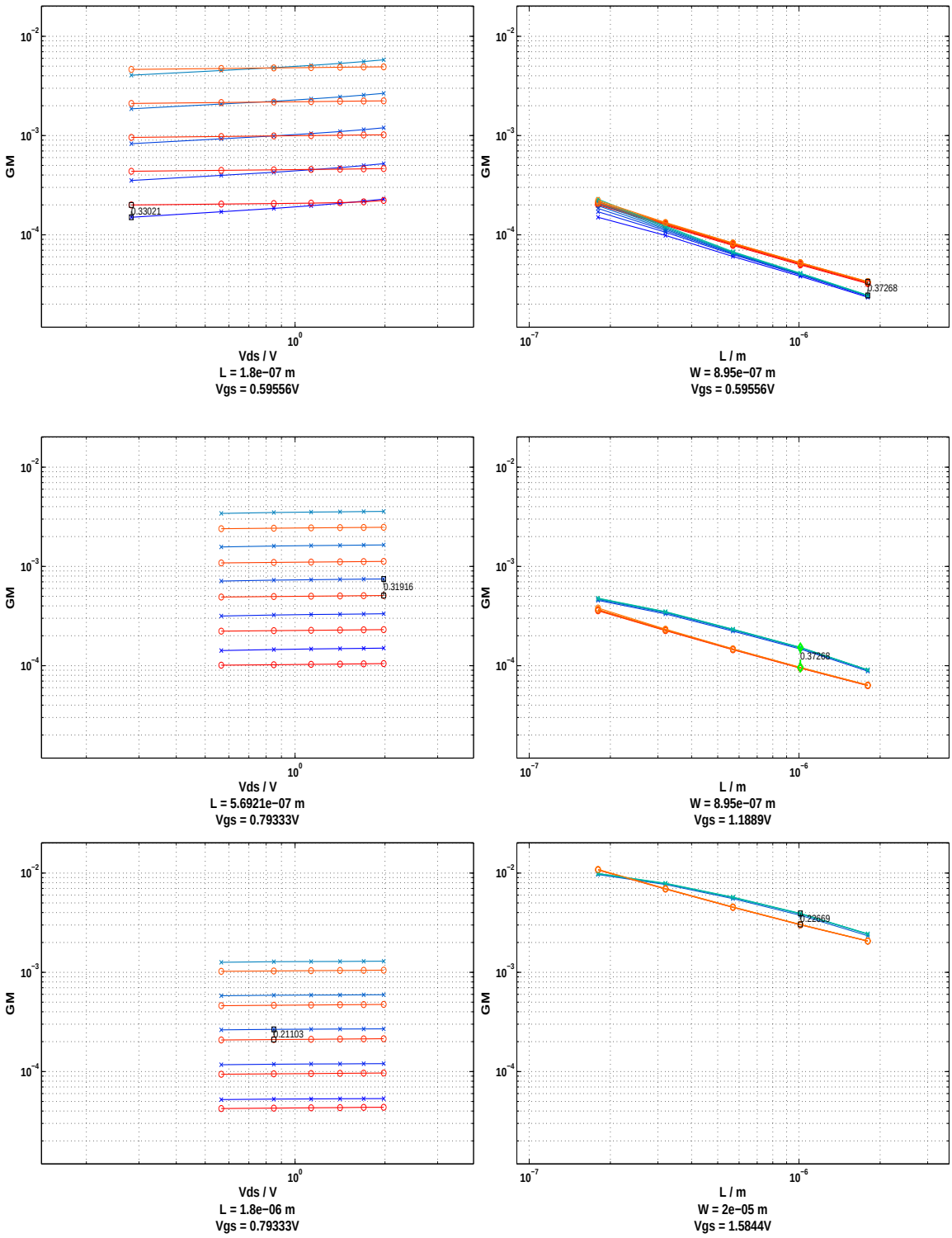


Figure 5-5: g_m plotted for an n-doped MOSFET, with input parameters L, W, I and V_{ds}. The first column shows g_m against V_{ds}, and second against L

the inflection point, and therefore the convex fit produces a smaller error along that particular dimension. As for the input parameter V_{ds}, the evolved posynomial model is convex relative to V_{ds} in order to fit the convex empirical data. Therefore, the results obtained for g_{ds} confirm the predictions regarding the curvature of posynomial models in order to fit data which in convex along some dimensions, and containing an inflection point along others. Similarly, the posynomial model for g_{ds} which minimizes ε_{rlae} demonstrates where the difficulty of fitting g_{ds} arises. Using a convex fit to model data with an inflection point produces relatively large error, as can be seen from the green error line in the figures.

5.3.3 Posynomial Model of the Output Resistance r_0

As discussed earlier, the plots or r_0 are in fact plots of g_{ds} reflected in the $x = y$ axis. Figures 5-8 and 5-9 show the plots of the posynomial model output superimposed with the empirical MOSFET data for the parameter r_0. In the case of r_0, the posynomial model fits the the concave data when r_0 is plotted against V_{ds} with a line in logarithmic space. Similarly, a line is used to fit both the linear dimension with respect to W, and the dimension L, which shows a slight inflection point, but is overall concave. Finally, the posynomial model is convex with respect to V_{gs}, in order to fit empirical data that contains an inflection point.

5.3.4 Posynomial Model of the Capacitance C_{gs}

In contrast to other output parameters we have examined thus far, the parameter C_{gs} proves a relatively easy parameter to fit. Its linearity, or near linearity across all of its input dimensions, suggests that a mononomial must fit the data with a very small maximum error. On plotting the results of the posynomial model generated by the GA, as shown in Figures 5-10 and 5-11, we conclude that GA did in fact evolve a monomial which produces a hyperplane in logarithmic space to fit C_{gs}. The largest error arises in the regions where C_{gs} is slightly concave with respect to V_{gs}, which once again validates the notion of a lower bound on error when fitting non-convex

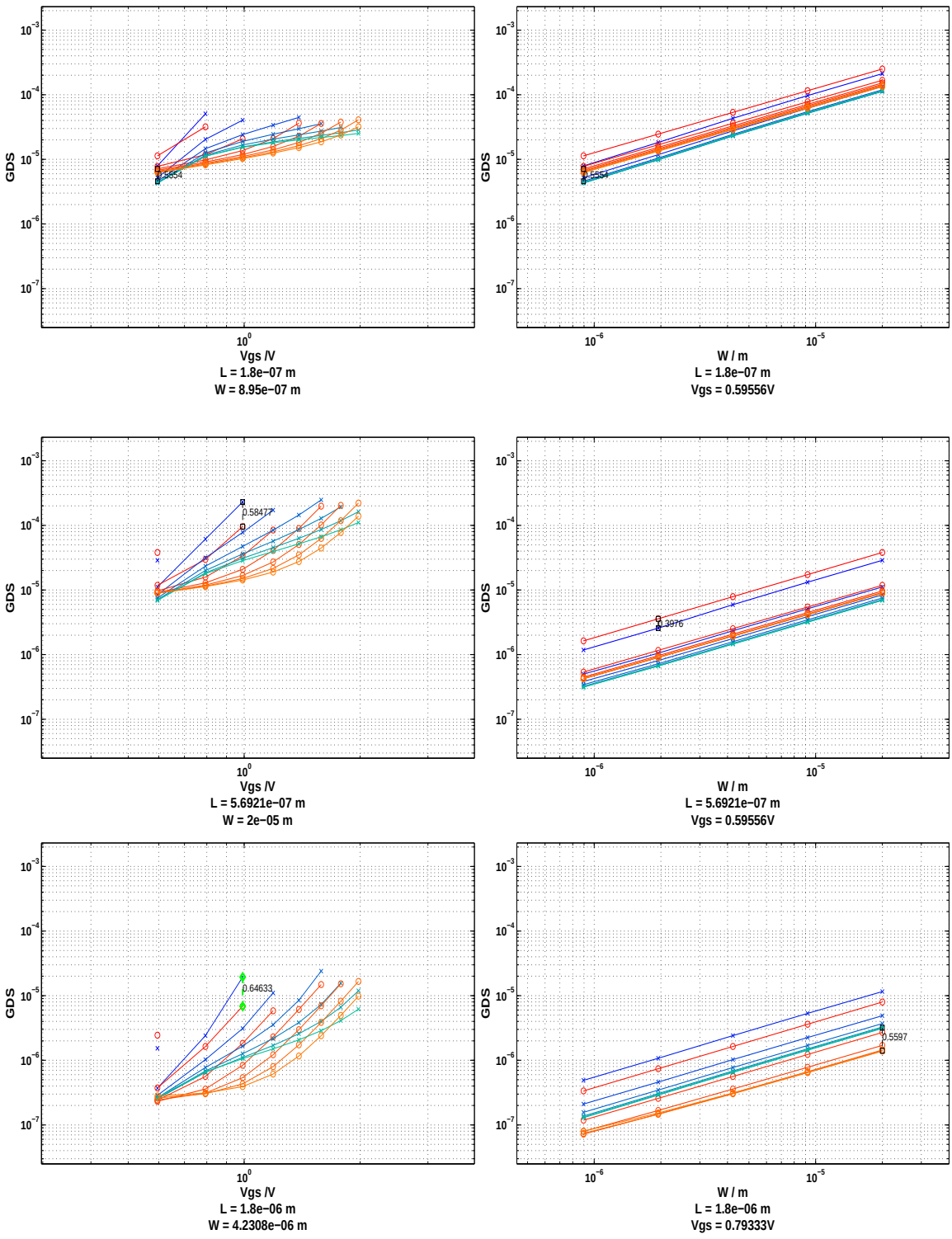


Figure 5-6: g_{ds} plotted for an n-doped MOSFET, with input parameters L, W, I and V_{ds}. The first column shows g_{ds} against V_{gs}, and second against W

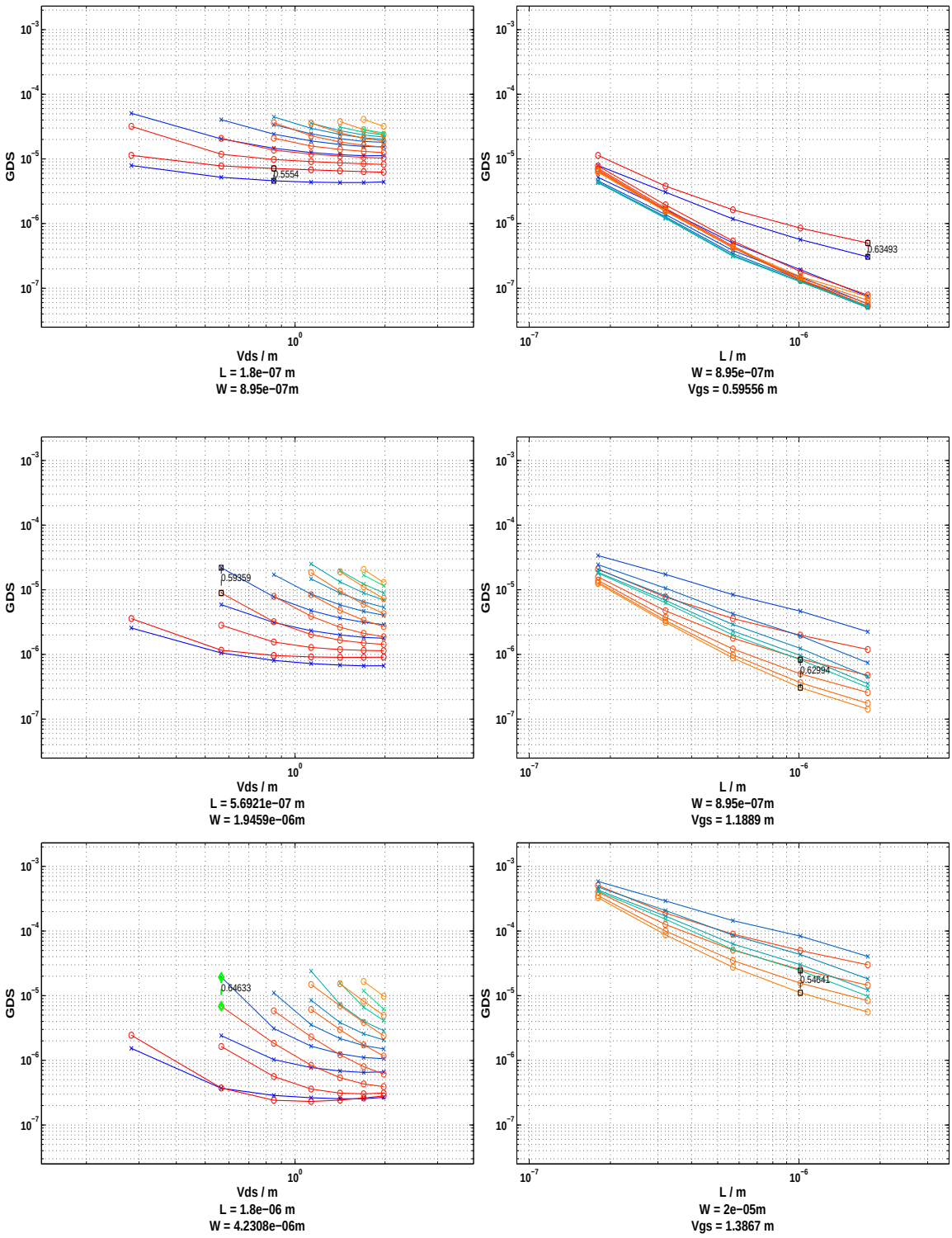


Figure 5-7: g_{ds} plotted for an n-doped MOSFET, with input parameters L, W, I and V_{ds}. The first column shows g_{ds} against V_{ds}, and second against L

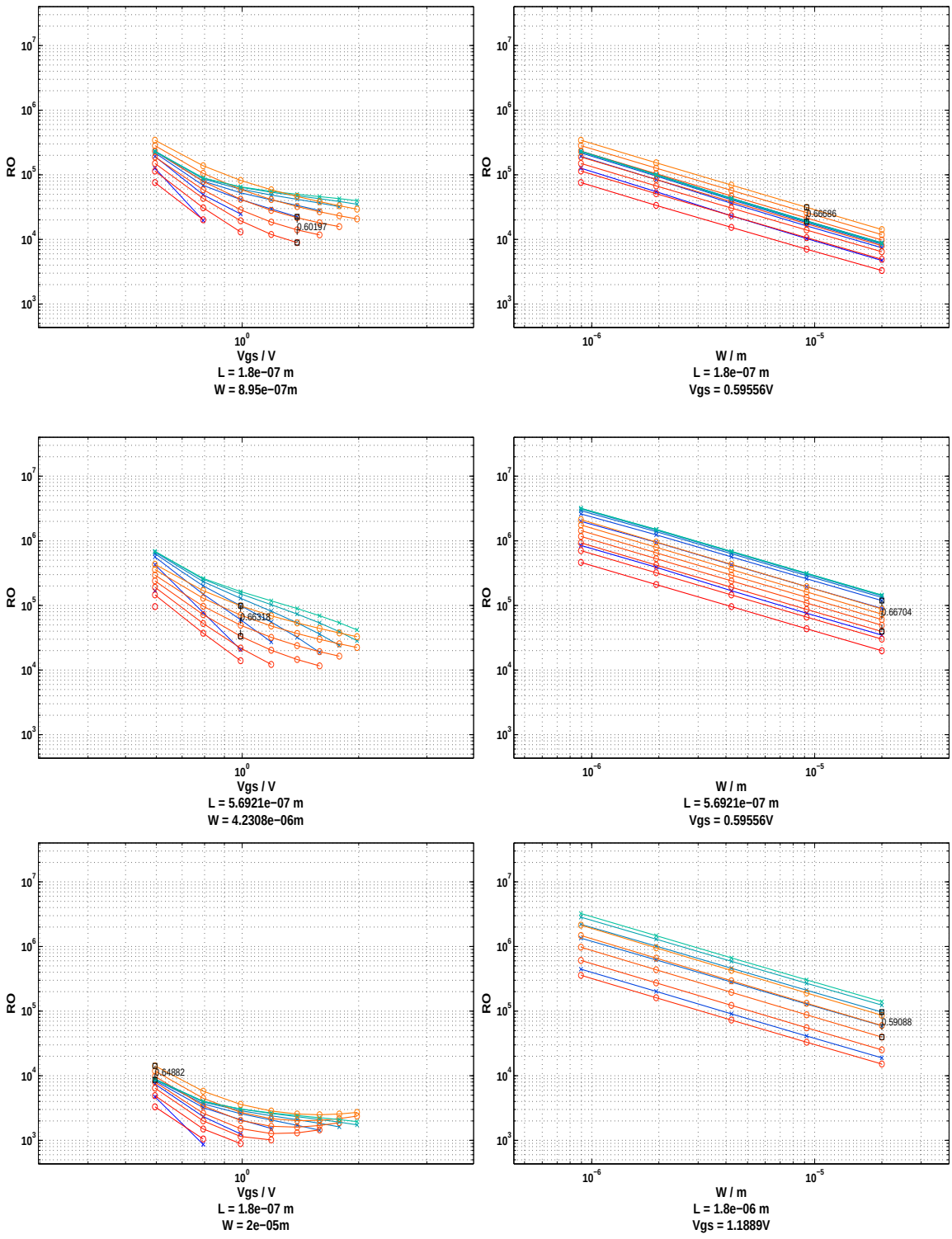


Figure 5-8: r_0 plotted for an n-doped MOSFET, with input parameters L, W, I and V_{ds}. The first column shows r_0 against V_{gs}, and second against W

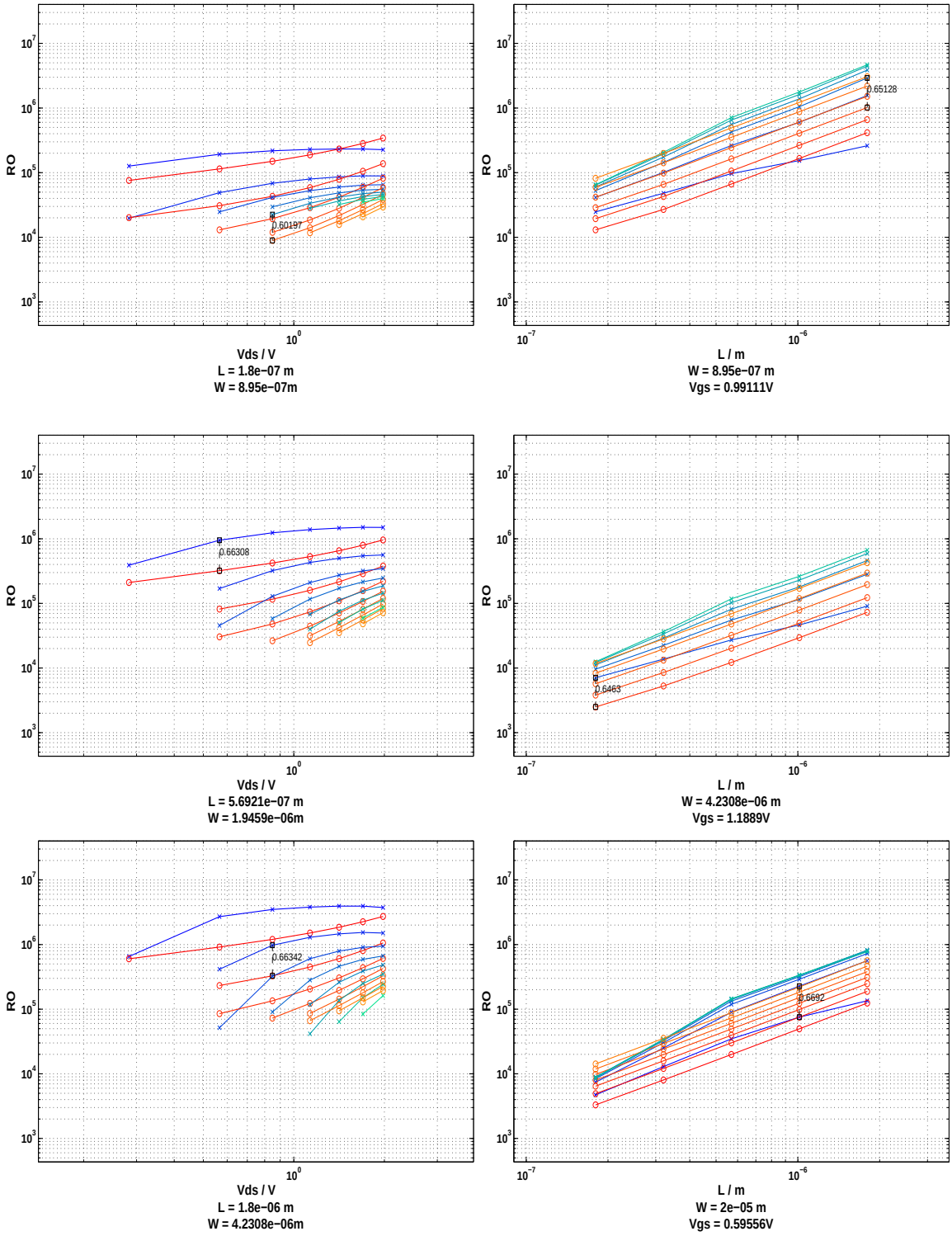


Figure 5-9: r_0 plotted for an n-doped MOSFET, with input parameters L, W, I and V_{ds}. The first column shows g_m against V_{ds}, and second against L.

data.

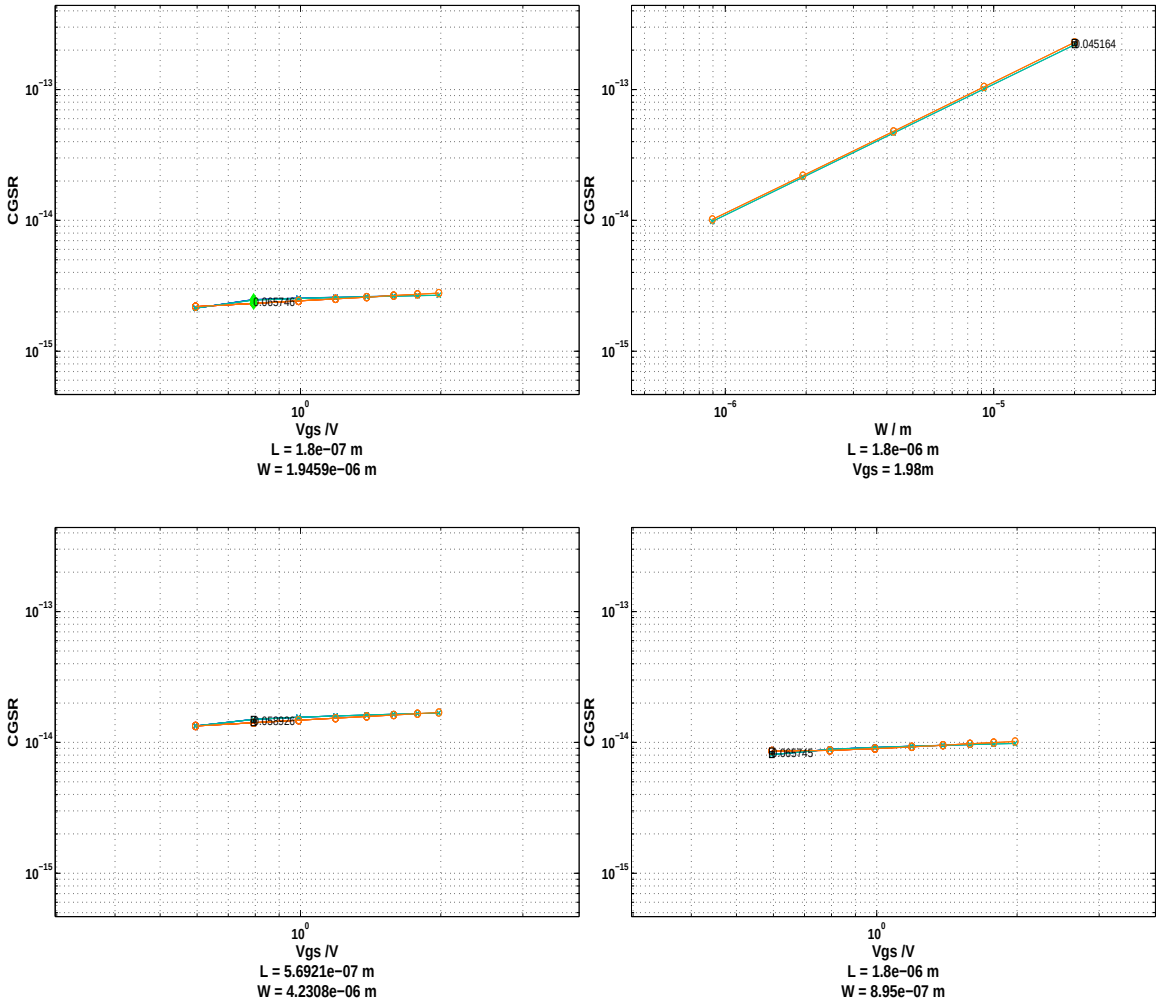


Figure 5-10: C_{gs} plotted for an n-doped MOSFET, with input parameters L, W, I and V_{ds}.

5.3.5 Posynomial Models of the Current I

Arguably, the current I is probably one of the most difficult parameters to fit. Recall that the current is concave with respect to V_{gs}, and exhibits a jump from one range to another when going from low V_{gs} to a higher V_{gs}. Figure 5-12 shows the plots of the current I against V_{gs} whilst fixing the other three dimensions, for both the empirical data and the posynomial model. As can be seen from the plots, a line in logarithmic space proves to be the best the posynomial model can do in terms of

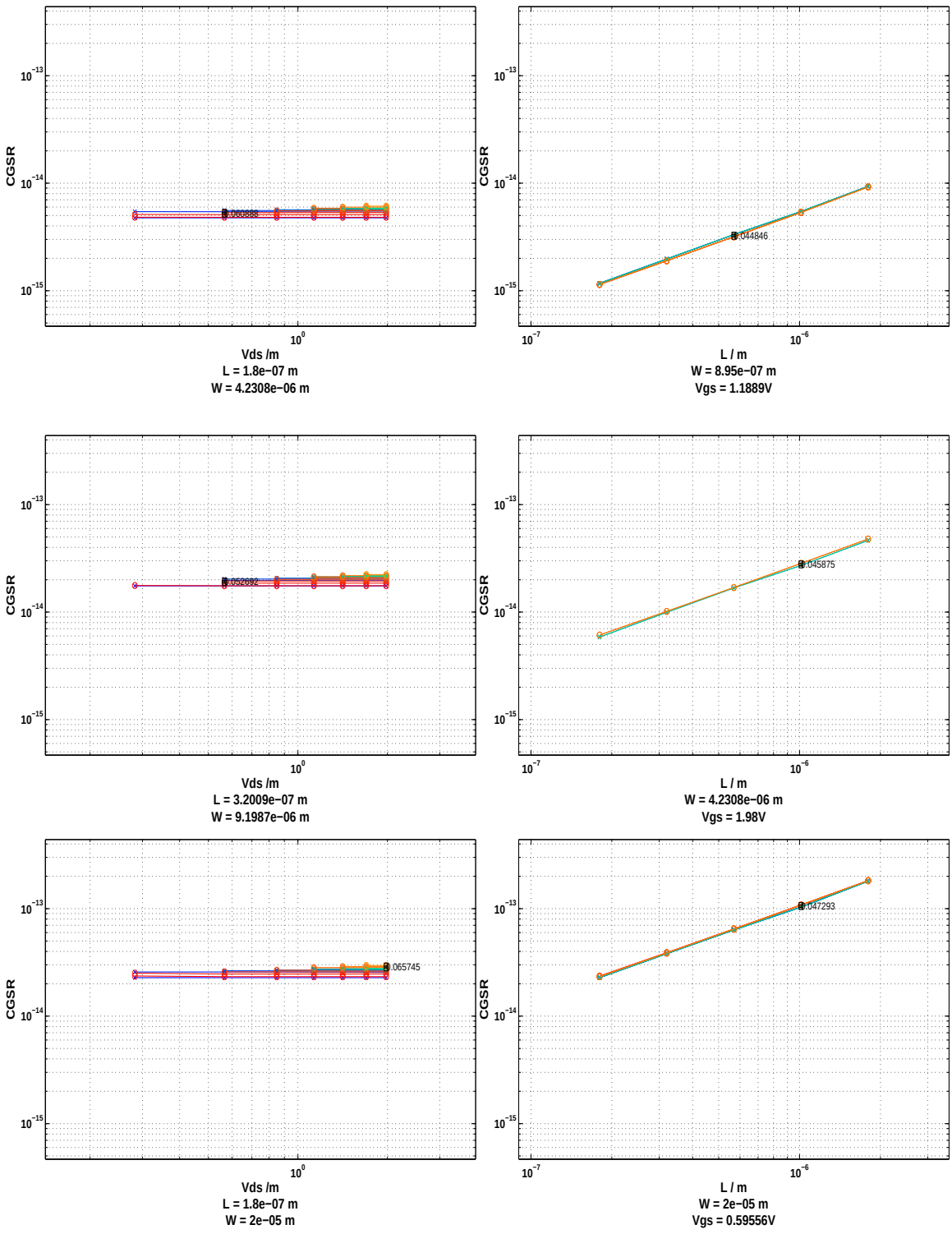


Figure 5-11: C_{gs} plotted for an n-doped MOSFET, with input parameters L, W, I and V_{ds}, where C_{gs} is plotted against V_{gs} and then W

fitting the concave, discontinuous data. As expected, the maximum error is quite large, about 100% in the plots shown.

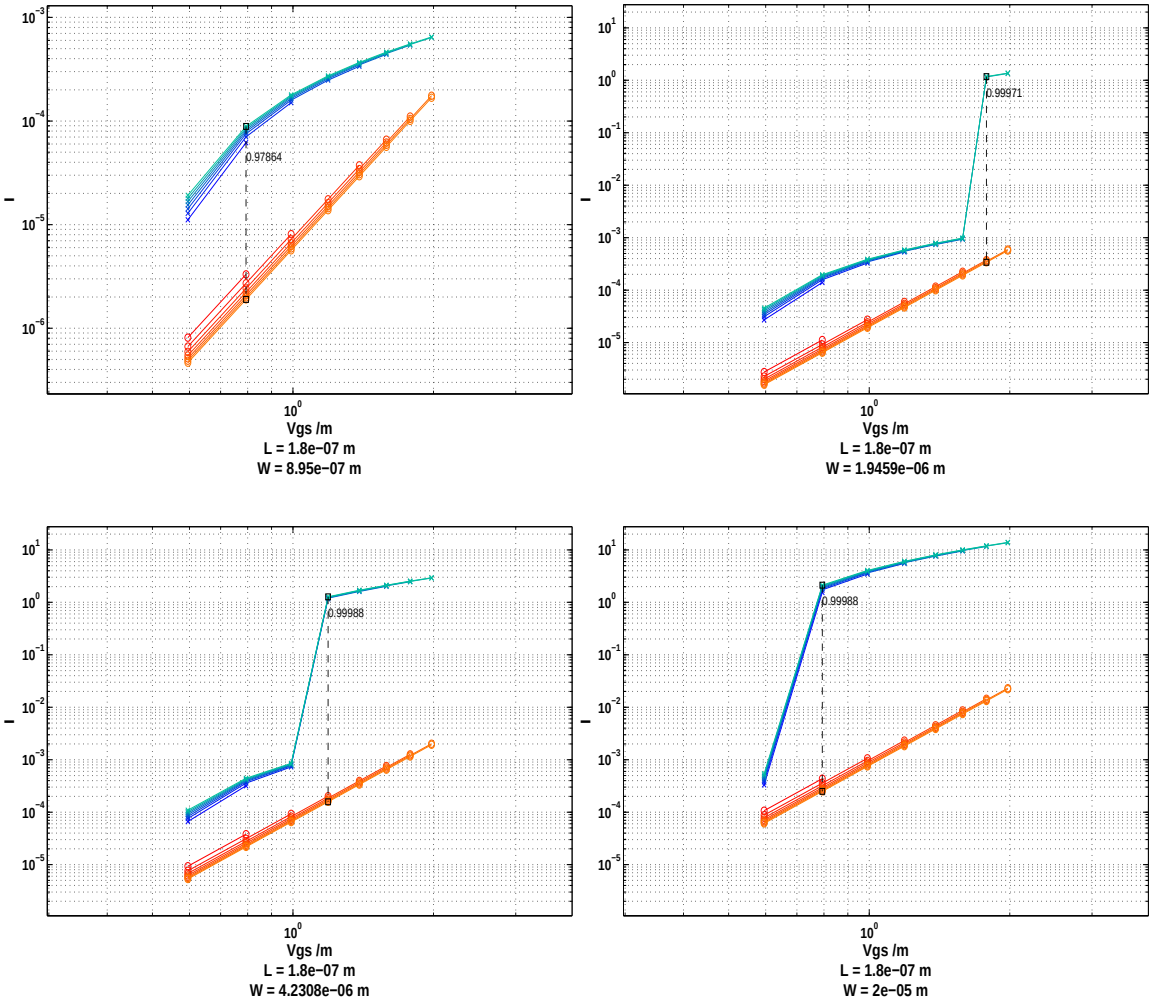


Figure 5-12: I plotted for an n-doped MOSFET, with input parameters L, W, I and V_{ds}. The first column shows C_{gs} against V_{ds}, while the second column shows C_{gs} against L

5.4 Experiments C: Comparison of GA with ε_{rlae} with Monomial Fitting

After we have attained some understanding regarding the behavior of our evolved posynomial models, their curvature, and the sources of error between the models

and the training data, we are now in the position to examine the overall maximum and mean errors across our entire data spectrum, and compare them with those derived from our benchmark algorithm, monomial fitting (MF). Posynomial models should theoretically perform as well as monomial fits for parameters that are either linear or concave across their various dimensions. On the other hand, posynomial models evolved through the use of the GA should perform better for parameters with non-linear, convex behavior across one or more of their dimensions. In order to test such a claim, we generated posynomials using model RLAE, for which the GA minimizes ε_{rlae}. We then selected the best overall individual in terms of maximum error, and calculated its percentage mean ($\%\varepsilon_{rlse}$) and maximum ($\%\varepsilon_{rlae}$) errors. The error percentages obtained from all the output parameters for input parameters A are shown in Table 5.4, whereas the results from input parameters B are shown in Table 5.5. Figures 5-13 and 5-14 plot the errors using a bar chart to emphasize any discrepancies between the GA and MF errors.

From the charts, we can determine when there exists a larger payoff from the use of GA evolved posynomial models, rather than monomial models from the less time consuming MF algorithm. The general trend though, indicates that GA evolved posynomials only perform slightly better than the MF models in terms of maximum error, ε_{rlae}. For example, in terms of input parameters B, and the output parameter g_m for an N-FET, the GA evolved posynomial only shows a 1.0% improvement in terms of maximum error over the MF produced monomial. Generally, the error improvement never exceeds the order of 5% for most of the MOSFET parameters.

The trend nevertheless, restricts itself to parameters that are either mostly linear in logarithmic space, or concave across some of their dimensions. For such parameters, a mononomial remains the best fit for the data regardless of whether we generate it using a GA or MF. The trend also includes parameters that are difficult to fit, such as g_{ds} with respect to parameters A, or I with respect to parameters B. Since these parameters exhibit a discontinuity in logarithmic space, once again, the best we can do is fit the data using a mononomial. Therefore, the both the GA and MF models produce maximum errors which are close to 100% for such parameters.

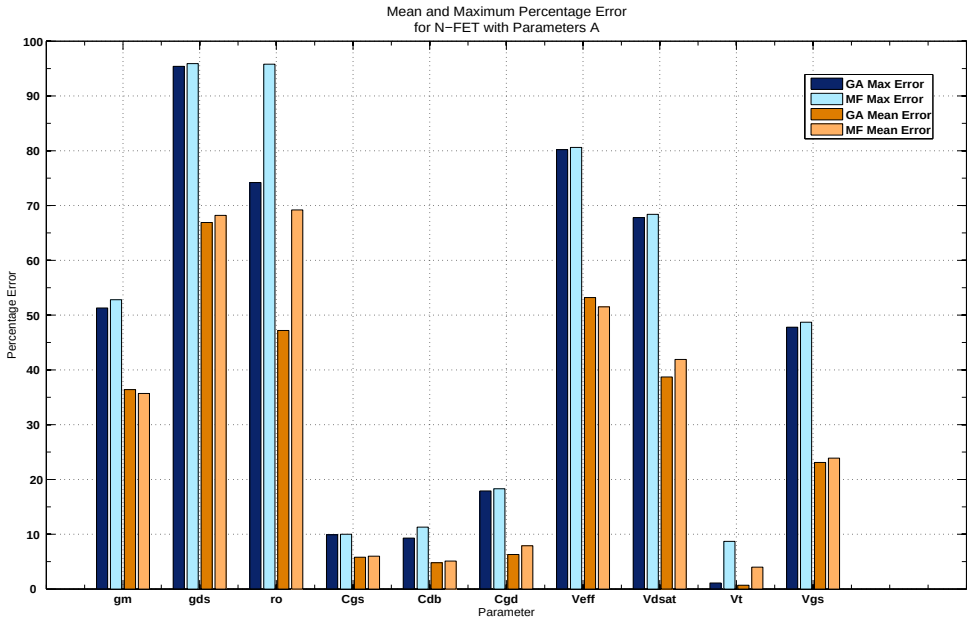
Parameter	Algorithm	N-FET		P-FET	
		$\%\varepsilon_{rlae}$	$\%\varepsilon_{rlse}$	$\%\varepsilon_{rlae}$	$\%\varepsilon_{rlse}$
g_m	GA	51.3	36.4	65.7	0.521
	MF	52.8	35.7	68.5	0.536
g_{ds}	GA	95.4	66.9	87.2	0.622
	MF	95.9	68.2	85.9	0.639
r_0	GA	74.2	47.2	48.3	0.299
	MF	95.8	69.2	85.9	0.501
C_{gs}	GA	9.9	5.8	29.4	0.213
	MF	10.0	6.0	30.1	0.253
C_{db}	GA	9.3	4.8	0.7	0.004
	MF	11.3	5.1	3.4	0.024
C_{gd}	GA	17.9	6.3	0.6	0.003
	MF	18.3	7.9	0.7	0.003
V_{eff}	GA	80.2	53.2	47.6	0.393
	MF	80.6	51.5	48.3	0.401
V_{dSAT}	GA	67.8	38.7	66.7	0.450
	MF	68.4	41.9	67.1	0.470
V_T	GA	1.1	0.7	0.6	0.003
	MF	8.7	4.0	02.1	0.014
V_{gs}	GA	47.8	23.1	63.1	0.432
	MF	48.7	23.9	71.7	0.538

Table 5.4: Percentage maximum and mean errors for the output parameters given input parameters A

On the other hand, recall that output parameters r_0 or g_{ds} for an n-doped MOSFET with respect to parameters B, and r_0 with respect to parameters A, are convex or contain an inflection point with respect to some of their input parameters. For these parameters, a GA evolved posynomial model produces an improvement in ε_{rlae} for up to 22%. Therefore, in cases of parameters g_{ds} and r_0, using the GA to evolve a posynomial model of the data produces models with significantly larger accuracy, and favors the GA over the MF algorithm. Nevertheless, MOSFET behavior has shown that most of the output parameters we are concerned with are not convex in logarithmic space.

Parameter	Algorithm	N-FET		P-FET	
		$\%_0\varepsilon_{rlae}$	$\%\ \varepsilon_{rlse}$	$\%\ \varepsilon_{rlae}$	$\%_0\varepsilon_{rlse}$
g_m	GA	37.2	25.6	67.1	49.4
	MF	38.2	26.9	67.3	49.0
g_{ds}	GA	64.3	39.4	72.9	53.2
	MF	82.8	40.1	72.8	53.7
r_0	GA	66.9	43.6	60.8	34.4
	MF	82.8	59.6	72.9	42.6
C_{gs}	GA	6.6	3.2	28.8	18.3
	MF	6.7	3.3	29.2	18.7
C_{db}	GA	6.0	2.9	0.6	0.4
	MF	11.2	4.7	3.4	2.4
C_{gd}	GA	9.7	5.6	0.5	0.3
	MF	16.2	8.5	0.6	0.3
V_{eff}	GA	40.8	24.8	97.7	97.2
	MF	41.1	25.6	100.0	99.4
V_{dSAT}	GA	20.6	13.7	49.6	34.4
	MF	20.7	13.9	49.7	34.9
V_T	GA	1.1	0.7	0.6	0.3
	MF	3.7	1.7	4.5	2.3
I	GA	99.9	63.5	100.0	78.3
	MF	99.9	60.9	100.0	78.5

Table 5.5: Percentage maximum and mean errors for the output parameters given input parameters B



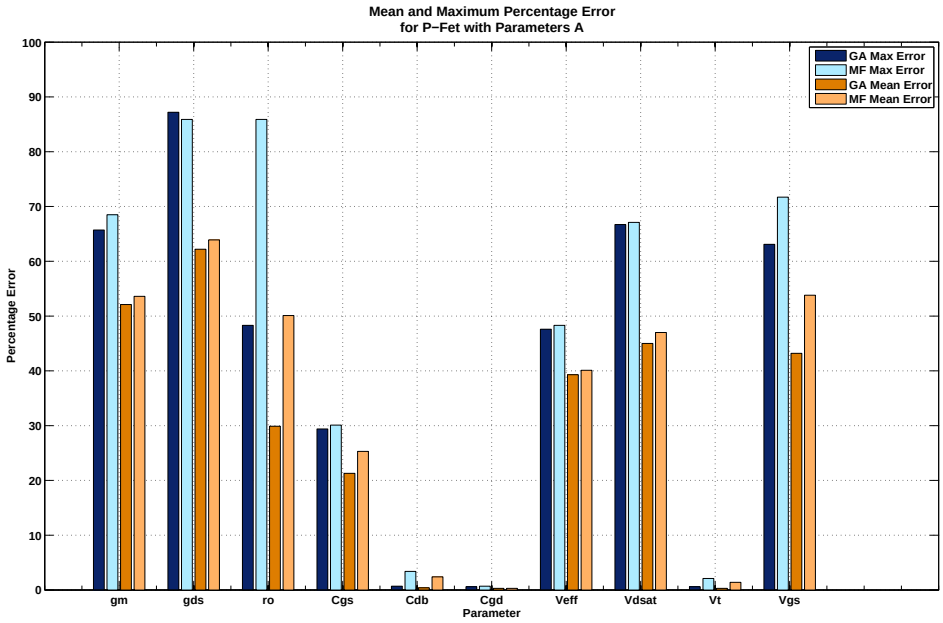


Figure 5-13: Bar chart of percentage mean and maximum error for n-doped and p-doped MOSFETs, using parameters A.

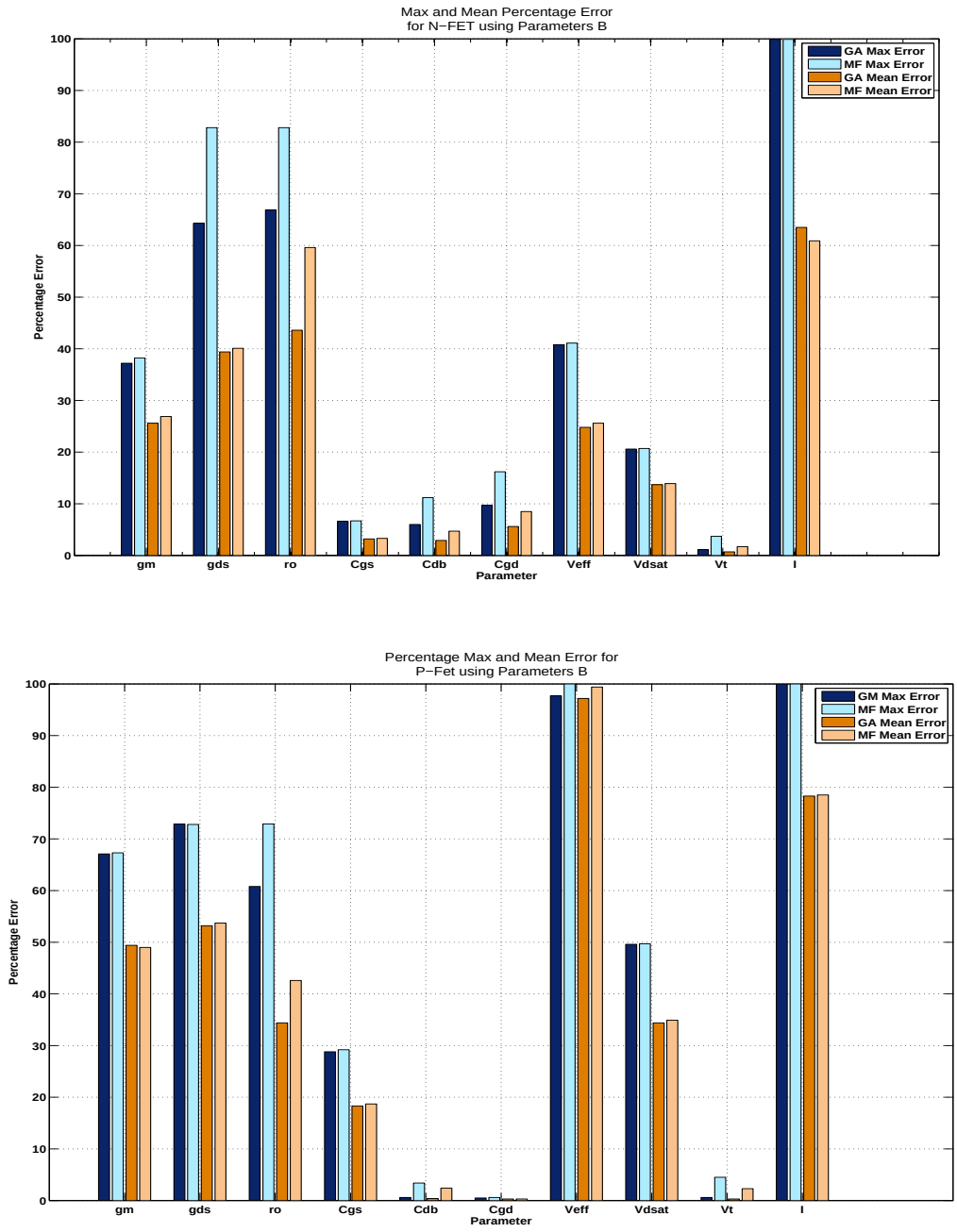


Figure 5-14: Bar chart of percentage mean and maximum error for n-doped and p-doped MOSFETs, using parameters B.

Chapter 6

Conclusions and Future Research

6.1 Conclusions

By empirically comparing various MOSFET models which attempt to minimize the maximum relative absolute error, we have shown that minimizing ε_{rlae} throughout the GA produces models with the smallest maximum error, without compromising the mean error too greatly. We have also shown that a GA with an adaptively adjusted error metric does not generate a significant improvement in mean error when compared to a GA model which purely minimizes the maximum error.

Our results have revealed that monomial fitting produces a maximum error nearly equivalent to that generated through the use of GA evolved posynomials for parameters that are either linear, or concave in logarithmic space. Nevertheless, the posynomials produced by the GA showed up to 20% improvement in error over monomial fitting when modeling parameters that showed an inherent convexity across some of their dimensions.

Using the visualization tool we have developed, we can clearly illustrate why MOSFET parameters are difficult to fit using posynomial models. On effectively plotting five-dimensional data, and providing insights on the behavior of MOSFET output parameters, we have posed a theoretical lower bound on the maximum error generated by posynomial models. After gathering empirical results for different MOSFET parameters, we have shown that the lower bound does exist for MOSFET posyno-

mial models, and is a result of an inherent concavity in MOSFET data. Therefore, Geometric Programming as a solution to the circuit sizing problem faces serious limitations in terms of model accuracy, which in turn questions the wisdom of its use for MOSFET parameter optimization.

6.2 Future Research

Throughout our investigation, we have been concerned with how well our evolved posynomial models perform on the data they were originally trained upon. Therefore, we have overlooked any questions pertaining to how well the models generalize to unseen data. Although the posynomial models retain an inherent limitation, the need to quantify how well the models generalize still remains.

On a different note, other types of models based on different optimization techniques for generating the posynomial coefficients and selecting individuals for propagation, require further investigation. We have begun assimilating a support vector machine (SVM) regressor into the GA's coefficient optimization stage, such that we have a new formulation for maximum error, which attempts to minimize the geometric margin. The SVM formulation introduces slack variables, which can be adjusted to control how stringently we attempt to fit data, thus allowing for certain points to be excluded from contributing to the final fit. An SVM, combined with a posynomial kernel, therefore constitutes a new area for exploration in the field of posynomial modeling.

Finally, since the unavoidable lower bound on error exists when modeling concave data, is there any way to work around this deterrence to geometric programming? The answer lies in specifying smaller ranges of workability, such that the output parameters are convex or linear within the specified ranges. The method should theoretically work well for output parameters that exhibit a large jump between lower and higher values, such as the current I modeled with respect to input parameters B. Branch and bound algorithms for finding ranges with the smallest errors may come in handy for such an approach, but one must keep in mind that such methods do

not scale very well when subjected to an increase in degrees of freedom, such as an increased number of input or output parameters.

Bibliography

[1] Varun Aggarwal and Una-May O'Reilly, "Simulation-based reusable posynomial models for mos transistor parameters", in *Design Automation and Test in Europe (DATE), 2007.*

[2] John R. Koza, Forrest H. Bennett III, David Andre, Martin Keane, and Frank Dunlap,"Automated Synthesis of Analog Electrical Circuits by Means of Genetic Programming", *IEEE Tracsactions on Evolutionary Computation*, 1(2): pp. 109-128, 1997.

[3] J.-B. Grimbleby, "Automatic Analogue Circuit Synthesis Using Genetic Algorithms", *IEEE Proceedings Circuits, Devices and Systems*, 147 (6): pp. 319-323, 2000.

[4] Maria del Mar Hershenson, Stephen P. Boyd, and Thomas H. Lee, "Optimal Design of a CMOS Op-Amp via Geometric Programming", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20(1): pp.1-21, 2001.

[5] Varun Aggarwal and Una-May O'Reilly, "Design of Posynomial Models for Mosfets: Symbolic Regression Using Genetic Algorithms", *Genetic Programming: Theory and Practice IV*: pp. 219-236, 2006.

[6] P. Mandal and V. Visvanathan, "CMOS Op-Amp Sizing Using a Geometric Programming Formulation", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20(1): pp. 22-38, 2001.

[7] Stephen Boyd and Lieven Vandenberghe, *Convex Optimization*, Cambridge University Press, United Kingdom, pp. 144-146, 2004.

[8] S. Boyd, S.-J. Kim, L. Vaudenberghe, and A. Hassibi. A Tutorial on Geometric Programming. In *Technical report, EE Department, Stanford University*, 2004.

[9] Franz Rothlauf, *Representations for Genetic and Evolutionary Algorithms*, Springer Berlin Heidelberg, the Netherlands, 2006.

[10] Varun Aggrawal, "Analog circuit Optimization Using Evolutionary Algorithms and Convex Optimization", M.S. Thesis, Massachusetts Institute of Technology, Cambridge, MA, United States, 2007.

[11] Melanie Mitchell, *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, USA, 1998.

[12] David E. Goldberg, *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[13] Carlos A Coello Coello, David A. Van Veldhuizen, and Gary B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*, Kluwer Academic Publishers, New York, NY, USA, 2002.