Contents

Chapter 1

SYMBOLIC DENSITY MODELS OF ONE-IN-A-BILLION STATISTICAL TAILS VIA IMPORTANCE SAMPLING AND GENETIC PROGRAMMING

Trent McConaghy[1]
[1]*Solido Design Automation Inc., Canada*

Abstract

This paper explores the application of symbolic regression for building models of probability destributions in which the accuracy at the distributions' *tails* is critical. The problem is of importance to cutting-edge industrial integrated circuit design, such as designing SRAM memory components (bitcells, sense amps) where each component has extremely low probability of failure. A naive approach is infeasible because it would require billions of Monte Carlo circuit simulations. This paper demonstrates a flow that efficiently generates samples at the tails using *importance sampling*, then builds genetic programming symbolic regression models in a space that captures the tails – the normal quantile space. These symbolic density models allow the circuit designers to analyze the tradeoff between high-sigma yields and circuit performance. The flow is validated on two modern industrial problems: a bitcell circuit on a 45nm TSMC process, and a sense amp circuit on a 28nm TSMC process.

Keywords: symbolic regression, density estimation, importance sampling, Monte Carlo methods, memory, SRAM, integrated circuits, extreme-value statistics

1. Introduction

In many types of industrial designs, random factors during the manufacturing process affect the performance of the final product. This is certainly the case in modern integrated circuits, where shrinking transistors have led to large process variations and therefore large performance variations. This in turn hurts chip yields, affecting time-to-market and profitability of semiconductor vendors.

Memory chips are among the circuits most affected in modern semiconductor design; effective solutions are of critical importance to memory vendors. While statistical effects can be simulated in a flow that incorporates a circuit simulator (e.g. SPICE) with a Monte Carlo analysis, that's not enough for memory design. Memory building block like bitcells are replicated millions of times or more on a chip; this means that for overall yields to be reasonable (e.g. >90%), each bitcell must have yields with failure rates millions of times lower than the overall chip. That is, they need yields up to 99.9999998% (6 sigma[1]).

For effective analysis, the memory circuit designer needs to analyze the tradeoff between such high-sigma yields and circuit performance. Equivalently, he or she needs accurate models of the extreme tails of the distribution.

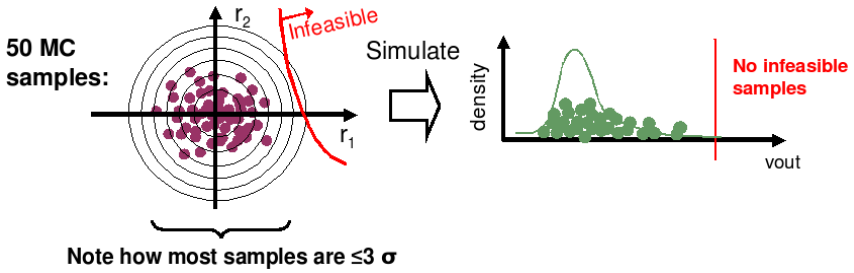

Figure 1-1. Monte Carlo sampling. Process point samples are drawn from a distribution (left), then simulated (middle), to get corresponding output values (right). A sample is "feasible" if all outputs' specifications are met. Here, a sample is feasible if $vout \leq vout_{thr}$, where $vout_{thr}$ is represented by the vertical bar in the output space (right), which maps to the nonlinear "infeasible" boundary in the process-variation space (left). Yield is the expected percentage of samples that are feasible. The challenge is: when the yield is extremely high, a small number of Monte Carlo samples will almost never have a failure, so yield cannot be accurately estimated.

This problem is challenging on several counts. Consider modeling a circuit where one in a million samples fail. Figure 1-1 shows the case if one draws a small number of Monte Carlo samples in process-variation space, and simulates them to get circuit performances (e.g. vout). No samples will even be close to failure, so any subsequent modeling on top of it would be useless. To get

[1]Sigma is another unit for yield: yield is the area under a Gaussian curve in the range -sigma to +sigma. Therefore 6 sigma is yield of 99.9999998%, or probability of failure 1.973e-9, which is about 2 in a billion.

just one failure, one would expect to run 1 million simulations, and 10 failures would take on average 10 million simulations. Figure 1-2 illustrates this case. One million simulations, even on a fast-simulating circuit and with a compute cluster, still typically takes a full day. So ten million is ten days, and 10 billion is 3 years.

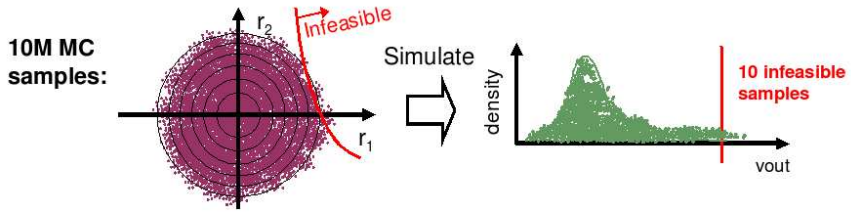

Figure 1-2. Monte Carlo sampling with many samples can capture the tails, but is computationally expensive.

A normal Monte Carlo run draws process points directly from the process variation distribution. As seen, far too many samples are needed in order to get (rare-event) failures in the design. A key insight is that we do not need to draw samples directly from the distribution. Instead, we can create samples that are infeasible more often, so that decent information is available at the tails. Importance Sampling (IS)(Hesterberg, 1988) is a well-known approach for rare event simulation, where the sampling distribution is shifted towards rare infeasible samples, as Figure 1-3 illustrates. Each sample has a weight that relates its density on the sampling pdf to the true pdf.

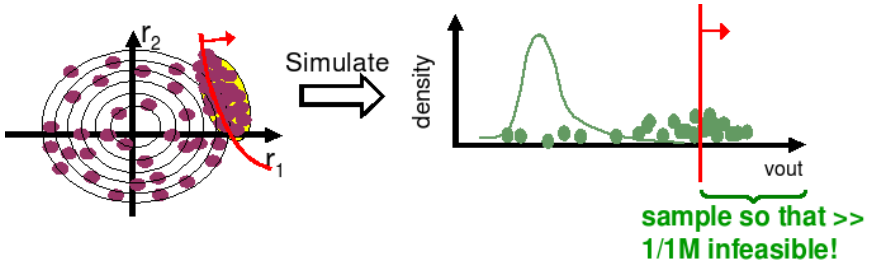

Figure 1-3. Importance sampling takes more samples at the tails of the distribution. But how do we build (symbolic) models from that data?

Given that we can efficiently take samples at the tails, how can we construct symbolic density models? This paper explores a practical flow using genetic programming (GP)(Koza, 1992). Specifically, GP symbolic regression models the distributions accurately even at the tails, by working from the importance-sampled data points, rescaled into the normal quantile space. Figure 1-4 right illustrates.

For reference, we show the naive Monte Carlo-based flow in Figure 1-4 left. As discussed, it is impractical because it requires too many simulations. Also, the high sample count is too computationally expensive for classical density estimation techniques such as kernel density estimation or expectation-maximization of gaussian mixture models (Hastie et al., 2001). Finally, the output model is not symbolic.

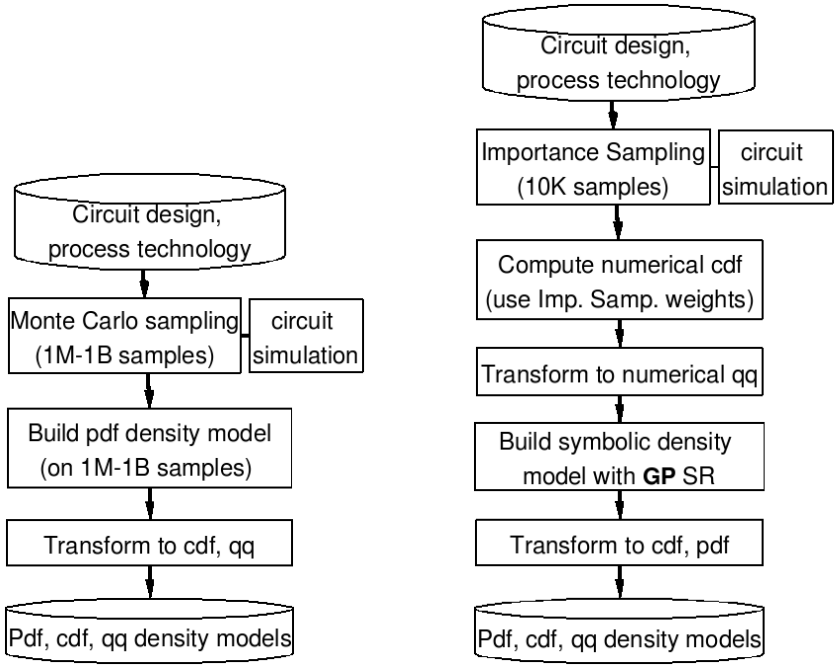

Figure 1-4. Left: Naive Monte Carlo based extraction of pdf, cdf, and nq density models needs too many samples for accuracy at the tails. Right: Proposed flow returns symbolic density models that are accurate even at the one-in-a-billion tails.

Working with importance-sampled data is not straightforward, because not every sample is equal; in fact some samples can have orders of magnitude more weight than others. This constraint renders most traditional density estimation techniques useless. Fortunately, we have a starting point: a numerical cdf (cumulative density function) can be computed from the raw importance-sampled data, where the y-values are dependent on the relative weights of samples.

One could consider a regression-style approach to build a density model in the cdf space, but that has problems. First, it underemphasizes the importance of the tails. Second, the search imposes difficult constraints: the cdf function must be monotonically increasing left-to-right, start at y=0.0 and end at y=1.0. To handle such constraints, one might use guaranteed-monotonic functions such as mixture-of-sigmoids (which is restrictive), or do numerical analysis of

chosen functions (which is expensive and provides no guarantees). One could also consider transforming the data into pdf space (probability densities, the derivative of cdf); but this is even more difficult because pdf models must be guaranteed to integrate to 1.0 across their whole input range. It is possible, but unpalatable: one must do numerical integration (which is computationally expensive), symbolic integration by linking to symbolic math software (which is complex), or use easy-to-analyze functional forms such as mixture of gaussians (which is restrictive). Further, doing regression on pdfs also underemphasizes the importance of the tails.

Rather than building models in cdf or pdf space, this paper proposes to build models in the *normal quantile* (nq) space, wherein raw cdf values are transformed based on the Gaussian function. A major benefit is that the closer a distribution is to Gaussian, the more linear the nq model is; this is quite unlike the highly nonlinear curves in cdf and pdf space. The only constraint for nq space is monotonicity, which is easy to meet by simply rejecting models that fail; because models are near-linear, most easily pass this constraint. Once models are built in nq space, they can be subsequently transformed to cdf and pdf space. The final challenge is how to make the regression-style nq models to be human-interpretable equations. This is the role of symbolic regression (SR) – the automated extraction of static whitebox models that map input variables to output variables. To ensure the models are human-interpretable, we use SR with canonical-form functions constrained by a grammar; that is, CAFFEINE (McConaghy and Gielen, 2006; McConaghy and Gielen, 2009). Figure 1-4 shows these steps of building symbolic density models in nq space, and transformation to cdf and pdf.

The rest of this paper is organized as follows. Section 2 reviews related work in GP. Section 3 describes the importance sampling approach, section 4 describes regression-style density modeling from importance samples, and section 5 describes symbolic regression-style density modeling. Section 6 gives the experimental setup, and section 7 presents experimental results. Section 8 concludes.

2. Related Work in GP

With a thorough search of the GP literature, we found just one set of work doing density estimation (Defoin Platel et al., 2007)[1]. However, that work could not model the crucial data – the tails – because it worked directly from Monte Carlo samples, and with just MC samples there are no samples at the high-sigma tails. One cannot model a region if one has no data for that region. In contrast,our approach explicitly samples at the tails, and avoids artificially

[1]While (Whigham, 2000) had "density model" in its title, it did not actually model pdfs/cdfs.

underemphasize the importance of the tails by modeling in normal quantile space.

3. Optimal Importance Sampling

Importance Sampling (IS) (Hesterberg, 1988) is a well-known approach for rare event simulation. In IS, samples are drawn from a sampling distribution $g(r)$ that is different than the true distribution $p(r)$. The sampling distribution has a greater bias towards the tails, e.g. towards the rare infeasible samples. To make statistical estimates (e.g. mean, yield) from importance sampled data, a weight w is assigned to each sample r: $w(r) = p(r)/g(r)$.

There are few theoretical constraints on the choice of sampling distribution, except that $g(r) > 0$ for all r that $p(r) > 0$. In practice, choosing g is more of a challenge when the random variables r have dimensionalty >10; we have 50-150 random variables. The challenge is to choose a $g(r)$ such that samples are infeasible often, yet the samples are sufficiently probable in $p(r)$ such that the weights $w(r)$ are not negligible. A pragmatic technique is to compute "centers", which are subsequently used as the means of Gaussian distributions for $g(r)$. Rather than heuristically choose centers as in (Kanj et al., 2006), we can cast the problem into an optimization problem:

$$\begin{aligned} \boldsymbol{r}^* = \quad & argmax\{p(\boldsymbol{r})\} \\ s.t. \quad & feasible(\boldsymbol{r}) = True \end{aligned} \tag{1.1}$$

where r is a random point in process variation space. r^*, the optimal r, is found by maximizing its density $p(r)$, subject to violating at least one performance value specification during simulation. If the true pdf has normal, independent and identically-distributed (NIID) random variables with mean=0 and no correlations, then maximizing density(r) is equivalent to minimizing ||r||.

Then, finding good centers amounts to solving the optimization problem, using an appropriate solver. We solve the optimization problem via (a) higher-stddev random sampling and simulation until a minimum number of infeasible samples are found, then (b) applying a small-population evolutionary programming algorithm (Yao et al., 1999), with SPICE simulation in the loop, to locally optimize the most-probable infeasible samples. Once the centers are found, $g(r)$ is defined as a mixture of (a) 25% samples from a mean=0 pdf with higher stddev, and (b) 75% samples are drawn from the centers with stddev=1.0. The sampling of Figure 1-3 follows this sampling. Importance sampling proceeds using this distribution until a stopping criteria is met; we stop when 10,000 samples have been taken. We dub this overall approach Optimal Importance Sampling.

The key output of importance sampling is a tuple for each sample i, where each tuple contains the process point r_i, its weight $w_i = w(r_i)$, and its SPICE-simulated performance value $m_i = simulate(r_i)$[1].

4. Density Modeling from Importance Samples

This section describes how density models can be computed from importance sampling data.

A typical (non-IS) density modeling problem is: given a set of performance values $\{m_1, m_2, \ldots, m_N\}$, compute a distribution $p(m)$ in terms of density, cumulative density, or normal quantiles (nq). This is usually treated as an unsupervised learning problem, and solved with e.g. kernel density estimation or Gaussian mixture models (Hastie et al., 2001).

Working with importance-sampled data is not straightforward, because not every sample is equal – they have weights. If we were to ignore the weights and compute the density, the density values in the tail regions would be far too large. However, we can apply a regression-style approach in nq space. Its steps are:

1 : Sort performance values in ascending order $m_1 <= m_2 <= \ldots <= m_N$, keeping corresponding weights aligned.

2 : Compute numerical cdfs as $cdf_i = \sum_{j=1}^{i} w_j$. These make the y-values in the numerical cdf, and the x-values are the corresponding m_i values.

3 : Compute numerical nq with the inverse-normal transformation $nq_i = erf^{-1}(2 * cdf_i - 1) * \sqrt{2}$, where $erf(x) = 2/\sqrt{\pi} \int_0^x e^{-t^2} dt$. These make the y-values in the numerical nq, and the x-values are the corresponding m_i values.

4 : Build a model $\widehat{nq}$ that maps $m \rightarrow nq$, i.e. a density model in nq space, using the training data $\{m_i, nq_i\} \forall i, i = 1..N$. This is performed with a 1-D regression method that minimizes the sum-squared prediction error across all possible input data, such as least-squares linear regression, quadratic regression, or more complex approaches.

5 : The model $\widehat{cdf}$ is the normal transformation from nq to cdf: $\widehat{cdf} = (1 + erf(\widehat{nq}/\sqrt{2}))/2$. This transformation can be performed symbolically or numerically.

[1]For simplicity, we will focus on just one performance value at a time. This is a reasonable simplification for memory problems.

6 : The model $\widehat{pdf}$ is the derivative of the cdf model: $\widehat{pdf} = d\widehat{cdf}/dm$. This transformation can be performed symbolically (e.g. with automatic differentiation) or numerically (e.g. with finite element models).

5. Symbolic Density Modeling with CAFFEINE

The last section described how to make regression-based density models from importance-sampled data. To make *symbolic* density models, the key is to use symbolic regression (SR) in the last section's model-building step (step 4). We use CAFFEINE (McConaghy and Gielen, 2006; McConaghy and Gielen, 2009), but any almost GP-based SR system would do here since the problem is a simple 1-D mapping. CAFFEINE's advantage is that it uses a grammar to constrain its search to the space of human-interpretable expressions, which ensures human readability and implicitly prevents bloat. We used CAFFEINE off-the-shelf, without changing any parameters compared to (McConaghy and Gielen, 2009). Population size was 100, running for 100 generations. This includes the CAFFEINE's multi-objective search: minimize error, and minimize model complexity, to return a set of nondominated models.

To speed up runtime[1], prior to CAFFEINE, we pruned each training dataset down to 50 points in a two-step flow. In the first step, the samples were sorted and every n^{th} sample was taken, such that just 250 samples remained. In the second step, we applied the SMITS balancing procedure of (Vladislavleva, 2008) to prune down to 50 samples. In each iteration of the SMITS algorithm, the weights of all samples are computed and the lowest-weighted sample is removed. The weights are based on "local deviation from linearity": at each point, its $k = 8$ closest neighbors in input space (m) are selected, a linear model from m to nq is constructed; then the weight is the absolute distance from the point to the linear model (plane in 2-d space). This weighting procedure naturally focuses the samples towards those with high information content, i.e. at the nonlinear "bends" in the training data.

The multi-objective approach taken in CAFFEINE used nondominated-sorting layers (NSGA-II) (Deb et al., 2002). An issue with NSGA-II is that the final nondominated set can be over-represented in some regions. So, we apply bottom-up clustering (hierarchical agglomerative clustering) to the 2-D Pareto Front.

6. Experimental Setup

We use two test circuits, a bitcell and a sense amp, as shown in Figure 1-5. These are the two major building blocks in designing memory circuits.

[1]Note: the speedups are not *needed*, they just help to get results in real-time.

The bitcell has 6 devices, and the sense amp 12. Each circuit's device sizes were set to have "reasonable" values by a memory circuit designer, leading to "reasonable" performance values. For the bitcell, the circuit performance of interest is v_{out}, focusing accuracy in the specification region of $v_{out} > 17$ mV. For the sense amp, it is -22 mV $\leq v_{offset} \leq$ 22 mV.

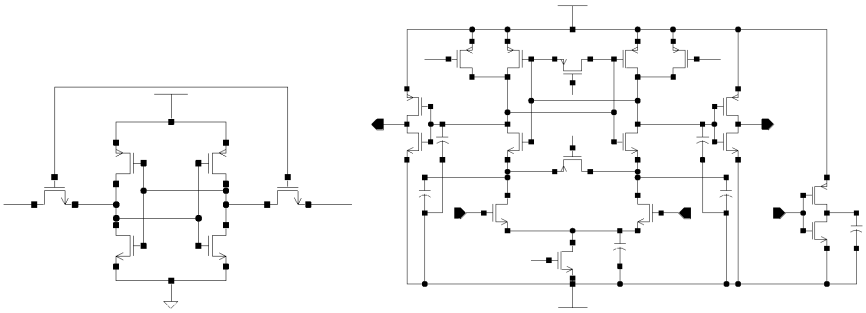

Figure 1-5. Circuit schematics. Left: bitcell. Right: sense amp

The variations in the circuit performance due to manufacturing imprecision can be modeled as a joint probability density function (jpdf). We use the well-known model (Drennan and McAndrew, 1999) where the random variables are "process variables" which model quantities like "substrate doping concentration". Variations in these quantities affect the electrical behavior of the circuit, and therefore its performances. In this model, there are about 10 normal independent identically-distributed (NIID) random variables per transistor. In total, the bitcell had 55 process variables, and the sense amp 125. In a Monte Carlo run, random points are drawn directly from the jpdf describing the process variations; and in importance sampling, random points are drawn from a sampling jpdf. Via simulation, the process-variation distribution maps to the performance distribution.

At each random point, we simulate the circuit at pre-specified environmental conditions. The bitcell's environmental conditions were temp=25degC, power supply voltage V_{dd}=1.0 V, and V_{cn}=0.0 V. The sense amp's environmental conditions were: load capacitance C_l=1e-15 F, temp=25degC, and V_{dd}=1.0 V. The simulator was was HSPICE^{TM}. The technology for the bitcell was TSMC 45nm CMOS, and for the sense amp 28nm CMOS.

7. Experimental Results

This section describes the results to validate the importance sampling plus SR flow on the bitcell and the sense amp.

First, we aimed to ensure that importance sampling could faithfully sample a broad range of the distribution. For "golden" results (results that we can use

as a reference for comparison), we ran 1M Monte Carlo samples for the bitcell and 100K for the sense amp. For IS results, we ran 10K importance samples. Then, we constructed a numerical nq plot for each set of results, using steps 2 and 3 in section 4 (in MC, all the weights are equal). The results are plotted in Figure 1-6. Note how the normal quantile values for the IS data line up nicely with the MC data, even in the tails (e.g. where normal quantile $<$-3 and $>$3). The discrete steps for the IS data are due to some samples having larger weights, which in turn influences the cdf and nq values more.

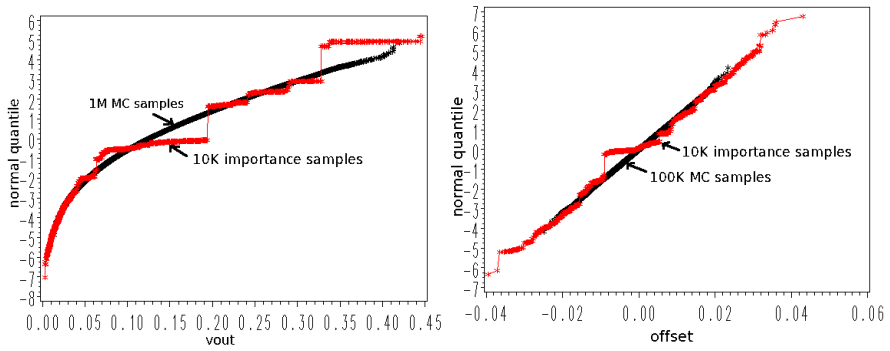

Figure 1-6. Importance Sampling results verified against "golden" Monte Carlo samples. Left: bitcell. Right: sense amp

These curves on their own have value to provide insight to the designer – information about the tails that we would not normally get with a limited sample MC run. In the sense amp, the linearity of the curves indicate that the sense amp's offset is Gaussian-distributed even into the tails, and that the mapping from process variables to offset is linear. In contrast, the bitcell's vout curve bends towards the bottom left, indicating a strong nonlinearity. Such information is valuable to gain insight into the nature of the tails of the distribution, and to understand the tradeoff between specifications and yield. From here on, we will focus on the bitcell results because, being nonlinear, they are the most interesting. (CAFFEINE solved the linear sense amp problem trivially.)

The next step was to build symbolic density models from the IS nq data, and to examine the results in nq, cdf, and pdf space. Given the $\{m,nq\}$ training data, the models were built according to section 5. Both steps of pruning took about 3 s total; CAFFEINE took about 30 s to run, and return its Pareto-optimal models. The models were pruned from 31 to 6 models. The Pareto-optimal models' output is plotted against the nq training data in Figure 1-7 top. The simplest CAFFEINE model was linear (note how its straight line misses the nonlinearity in the bottom left).

The other models had similar performance, and successfully captured the nonlinearity. The cdf and pdf curves for the lowest-error model were computed

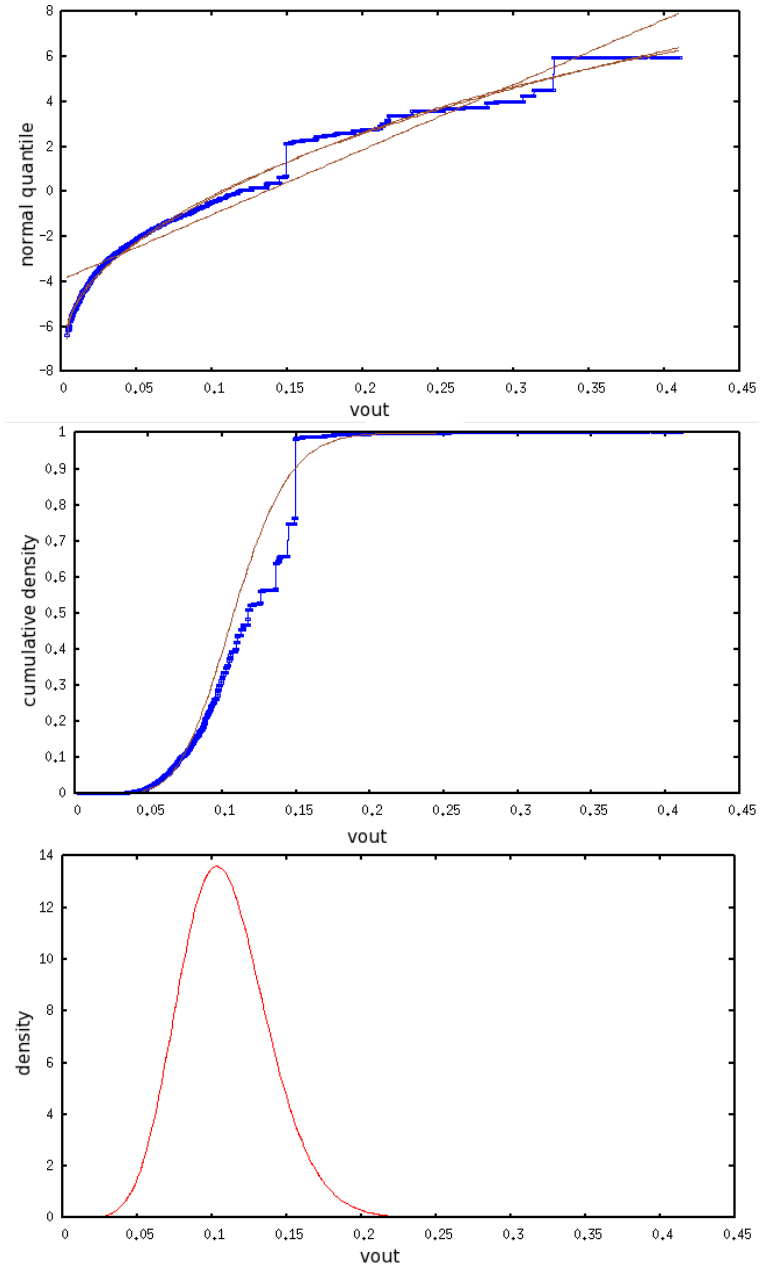

Figure 1-7. CAFFEINE results and training data in nq space (top), cdf space (middle), and pdf space (bottom). In the nq space, all nondominated models are shown. In the other spaces, just the lowest-error model is shown.

numerically according to the flow in section 4; they are shown in Figure 1-7 middle and bottom.

Notice how the cdf curve naturally starts at a value of exactly 0.0, and monotonically increases to finish at exactly 1.0. This would have been difficult or expensive to do if doing symbolic regression in cdf space, but was natural and simple by transforming from nq to cdf space. Also, with an "eyeball" check we can see that the pdf curve roughly integrates to 1.0. Once again, this would have been difficult or expensive if doing symbolic regression in the pdf space, but was natural and simple by transforming from cdf to pdf space.

The results in the three spaces are complementary: the nq illuminates the tails the most, the cdf allows one to quickly assess the tradeoff between yield (equivalent to the cdf value or 1-cdf value, depending on the spec), and the pdf gives intuition about where the samples tend to focus.

The output symbolic models have high value to the designer, as he can manually manipulate the equation. For high-sigma memory design, the nq space is actually the most natural, so the CAFFEINE models can be used directly. Table 1-1 shows some of CAFFEINE-output models. Note how readily-interpretable they are, while accurately capturing the mapping from v_{out} to normal quantile nq.

Table 1-1. CAFFEINE-generated models mapping v_{out} to normal quantile nq, in order of decreasing error and increasing complexity.

Train error (%)	nq **Expression**
26.3	$-3.903 + 28.90 * v_{out}$
11.42	$-7.358 - 3.736 * v_{out} + 23.89 * \sqrt{v_{out}}$
11.01	$368.1 - 46.482 * v_{out} + 170.959 * \sqrt{v_{out}}$ $+41.0435 * log10(1.07285e-7 * \sqrt{v_{out}})$ $-0.002472/\sqrt{max(0, 1.562e-10/\sqrt{v_{out}})}$

8. Conclusion

This paper described a new challenge for GP-based symbolic regression: building density models that are accurate at the extreme tails of the distribution. This challenge matters for the real-world problem of variation-aware memory circuit design. This paper discussed how a naive approach using plain Monte Carlo would fail in handling such rare-event tails. Then, this paper described a flow to extract extreme-value symbolic density models: optimal importance sampling, symbolic regression in the normal quantile space, and transformation to cdf and pdf space. The flow was validated on two industrial problems: a bitcell circuit on a 45nm TSMC process, and a sense amp circuit on a 28nm TSMC process.

9. Acknowledgment

Funding for the reported research results is acknowledged from Solido Design Automation Inc.

References

Deb, Kalyanmoy, Pratap, Amrit, Agarwal, Sameer, and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6:182–197.

Defoin Platel, Michael, Verel, Sébastien, Clergue, Manuel, and Chami, Malik (2007). Density estimation with genetic programming for inverse problem solving. In et al., Marc Ebner, editor, *Proc. European Conference on Genetic Programming*, volume 4445 of *Lecture Notes in Computer Science*, pages 45–54. Springer.

Drennan, P. and McAndrew, C. (1999). A comprehensive mosfet mismatch model. In *Proc. International Electron Devices Meeting*.

Hastie, T., Tibshirani, R., and Friedman, J.H. (2001). *The Elements of Statistical Learning*. Springer.

Hesterberg, T.C. (1988). *Advances in importance sampling*. PhD thesis, Statistics Department, Stanford University.

Kanj, R., Joshi, R.V., and Nassif, S.R. (2006). Mixture importance sampling and its application to the analysis of sram designs in the presence of rare failure events. In *Proc. Design Automation Conference*, pages 69–72.

Koza, John R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA.

McConaghy, T. and Gielen, G.G.E. (2006). Canonical form functions as a simple means for genetic programming to evolve human-interpretable functions. In *Proc. Genetic and Evolutionary Computation Conference*, pages 855–862.

McConaghy, T. and Gielen, G.G.E. (2009). Template-free symbolic performance modeling of analog circuits via canonical form functions and genetic programming. *IEEE Trans. Comput.-Aided Design of Integr. Circuits and Systems*, 28(8):1162–1175.

Vladislavleva, E. (2008). *Model-based Problem Solving through Symbolic Regression via Pareto Genetic Programming*. PhD thesis, Tilburg University.

Whigham, P. A. (2000). Induction of a marsupial density model using genetic programming and spatial relationships. *Ecological Modelling*, 131(2-3):299–317.

Yao, Xin, Member, Senior, Liu, Yong, Member, Student, and Lin, Guangming (1999). Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation*, 3:82–102.

Index