Faster, Parametric Trajectory-based Macromodels Via Localized Linear Reductions

Saurabh K Tiwary
Cadence Berkeley Labs
Berkeley, CA USA
stiwary@cadence.com

Rob A Rutenbar
Carnegie Mellon University
Pittsburgh, PA USA
rutenbar@ece.cmu.edu

***Abstract*— Trajectory-based methods offer an attractive methodology for automated, on-demand generation of macromodels for custom circuits. These models are generated by sampling the state trajectory of a circuit as it simulates in the time domain, and building macromodels by reducing and interpolating among the linearizations created at a suitably spaced subset of the time points visited during training simulations. However, a weak point in conventional trajectory models is the reliance on a single, global reduction matrix for the state space. We develop a new, faster method that generates and weaves together a larger set of smaller localized linearizations for the trajectory samples. The method not only improves speedups to 30X over SPICE, but as a side benefit also provides a platform for parametric small-signal simulation of circuits with variational device/process parameters, at a speedup of roughly 200X over SPICE.**

I. Introduction

Macromodels are simplified circuits which capture just the essential behaviors of some target circuit, and are fast enough to support full system simulation. Trajectory-based methods [8] [15] [14] [5] [4] [3] are an increasingly popular methodology for automatic macromodel construction. These methods enjoy several attractive properties: they can be built from simulation data during standard transient circuit simulation; they usefully exploit powerful linear model order reduction, interpolation, and datamining ideas for numerical accuracy and efficiency; they have recently been extended to highly scalable, and parametric forms.

Trajectory methods sample the state trajectory of a circuit as it is simulated in the time domain, and build a macromodel by reducing the linearizations created at an appropriately chosen subset of the time points visited during training simulations, and then interpolating among them. Interpolation combines these reduced linearizations to predict the dynamic behavior of the circuit at any new point in the state-space not visited during prior training. Trajectory methods can build macromodels automatically ("on demand" in language of [17]) using trusted simulation data.

One significant weakness of conventional trajectory methods, however, is their reliance on a single, essentially global reduction matrix for projecting from the intractable, large, full state space of the original circuit, down to a smaller, more efficient state space for interpolation. There is no reason to believe that a single projection matrix is likely to be optimal across a large, high dimensional state space. However, we cannot simply build an arbitrary set of local projection matrices if we still plan to approximate the circuit response at each timestep by interpolating from among a set of local linearizations at previously visited trajectory pionts. In other words, we trivially combine the essential data we need from the trajectory samples, if each uses a different local linearization. The result is that our global reduction is less efficient, yields a larger subspace than is likely needed, and some efficiency is lost.

In this paper, we suggest a practical strategy for building and combining local linearizations in a trajectory-based engine. The key idea, which exploits the scalable algorithm introduced in [17], is to build the *local* reductions for groups of k points that are actually used to interpolate the state space equation at a new point in the state space. The interpolation computation is thus, handed off from one *local* group to another as the circuit evolves through its state space. A somewhat surprising side benefit of this strategy is that the methodology provides a natural platform for efficient small-signal analysis of parametric circuits, i.e., ac analysis of circuits with variable device or process parameters. The *scalable* architecture ideas and *local* models provide us with the opportunity to store and interpolate the small signal state space equation for the circuit at their DC bias points for different device and parameter values. The small signal models thus generated capture the variational behavior of the circuit over large ranges of device and process parameters. This capability provides extremely fast and efficient design space exploration opportunities.

This paper is organizes as follows. Sec. 2 gives some basic background on trajectory methods, and the features missing in prior efforts. Sec. 3, 4 and 5 describes the key innovations in our new algorithm: compact *local* reduced order models; extensions for parameter space modeling and support for circuit loading effects. Sec. 6 shows experimental results for a range of circuits and their trajectory/parameter macromodels; our largest models achieve up to $\sim$30X speedups for transient and $\sim$200X speedups for AC simulation. Finally, Sec. 7 offers concluding remarks followed by acknowledgements in Sec. 8.

ICCAD'06, November 5-9, 2006, San Jose, CA

II. Background

A. Trajectory-Based Models

Circuit simulators represent a circuit as a set of nonlinear differential equations. In state-space form, these can be written as:

$$\left.\begin{aligned}\frac{dg(x)}{dt} &= f(x) + B(x)u \\ y &= C^T x\end{aligned}\right\} \quad (1)$$

The twin difficulties of simulating complex circuits are: (a) the nonlinearities associated with each transistor require many expensive model evaluations as the circuit moves through its state-space, and (b) the order (N) of the resulting equations is large. Together, these make large circuits with complex models that are slow to simulate. We can use trajectory methods to attack these problems in two ways. First, we replace expensive nonlinear model evaluations with simpler lookups and interpolation. Since evaluating $f(.)$ and $g(.)$ is expensive, we approximate these as a simple first-order Taylor series, expanded around some state x_0, for example:

$$\left.\begin{aligned}f(x) &\simeq f(x_0) + A_0(x - x_0) \\ g(x) &\simeq g(x_0) + G_0(x - x_0)\end{aligned}\right\} \quad (2)$$

where A_0, G_0 are the Jacobians of f and g respectively at x_0. (One can also choose a second-order expansion, at additional complexity; see [5].) Second, we reduce the dimensionality of the overall system of equations. We approximate the real N-dimensional state-vector x with a much smaller vector z of order $q \ll N$. The idea is to approximate the original system by carefully selecting a reduced subspace wherein most of the dynamics of the system occur. This can be done via projection methods, by constructing a suitable reduction matrix V of size $N \times q$ whose columns define a basis in the reduced state-space. So, we approximate states x from the original state-space by reduced states z:

$$x \simeq \hat{x} = Vz \quad (3)$$

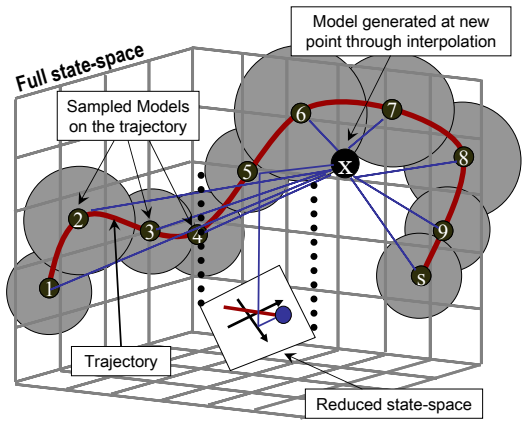

Fig. 1. Models generated at sampled points on the trajectory of a system in its state-space. State-space equation at a new point in state-space is generated through interpolation of the reduced order sampled models.

Reduction techniques combining Krylov [16] [11] [9] and TBR [8] methods can be employed here. The pseudo-code for computing the projection basis V for trajectory based models based on first order approximation of the original system is shown in Algorithm 1. [3] presents an extension of this algorithm for generating the projection basis V that captures the process variation information as well.

Algorithm 1 Algorithm for generation of projection basis V

1: Let x_i be the trajectory samples for i={1,2,...,s}
2: V_{agg}=[]
3: **for all** i **do**
4: Compute trajectory linearizations around x_i

$$G_i \frac{dx}{dt} = f(x_i) + A_i(x - x_i) + B_i u$$
$$y = C^T x$$

5: Construct $q1^{th}$ order orthogonal Krylov bases V_1 and V_2 using Arnoldi [10] or block Arnoldi [7]

$$span\{V_1\} = K_{q1}\{A_i^{-1}G_i, A_i^{-1}B_i\}$$
$$span\{V_2\} = K_{q1}\{A_i^{-1}G_i, A_i^{-1}(f(x_i) - A_i x_i)\}$$

6: $\tilde{V}$ = [V_1 V_2 x_i]
7: $\tilde{V}'$ = svd($\tilde{V}$)
8: V_i = Columns of $\tilde{V}'$ corresponding to singular val $> \epsilon_1$
9: V_{agg} = [V_{agg} $\tilde{V}'$]
10: **end for**
11: V'_{agg} = svd(V_{agg})
12: V = Columns of V'_{agg} corresponding to singular val $> \epsilon_2$

If we linearize at appropriately spaced points on the trajectory, and then reduce the order of these linearizations, we can approximate the dynamics at any new point in the space by interpolating among these saved linearized reduced order equations.

Suppose reduced linearizations have been generated at s points on the trajectory. Interpolation builds a single effective linearization for the new point x in state-space as a weighted sum of these s stored linearizations.

$$w_i(z) = \frac{(exp(z_i - z)^2)^{-k}}{\sum_{i=1}^{s}(exp(z_i - z)^2)^{-k}} \quad \text{for i=1,2,..,s} \quad (4)$$

where, $w_i(z)$ is the weight contribution of the i^{th} linearization on the trajectory, for the point z's state-space matrix in the reduced order state-space. We used a value of $k \sim 5$ in our implementation.

The weighting function [14] [18] (Equation 4) is based on the heuristic that the state-space equations for the points lying inside the spheres in Figure 1 are similar to the equation for the linearized point at the center of the respective spheres. Interpolation lets us approximate the overall state-space equation as shown in Eqn 5.

This formulation has been referred to as *piecewise linear trajectory-based model order reduction* [14]. The "piecewise linear" derives from the first-order Taylor expansion and the weighted linear combination form for the interpolation. The

$$\left.\begin{aligned} \frac{d}{dt}\Big(\big(\sum_{i=1}^{s} w_i(z)G_{ir}\big)z + \sum_{i=1}^{s} V^T\big(g(x_i) - G_i x_i\big)w_i(z)\Big) &= \big(\sum_{i=1}^{s} w_i(z)A_{ir}\big)z + \sum_{i=1}^{s} V^T(f(x_i) - A_i x_i)w_i(z) + Bu \\ y &= C_r^T z \end{aligned}\right\} \quad (5)$$

where,

$$G_{ir} = V^T G_i V, \quad A_{ir} = V^T A_i V, \quad C_r = C^T V \quad \& \quad \sum_{i=1}^{s} w_i(z) = 1.$$

strategy works well when we have enough training trajectories to create a sufficient density of "overlapping" linearizations, as represented by the spheres in Figure 1, and when the dynamics of the circuit can be approximated with much reduced versions of these linearizations. In several experiments to date, these seem to be viable assumptions.

The basic idea of trajectory models can be easily extended to capture the small signal behavior of the circuit in its parameter space. Rewriting the state-space equation for the circuit around a DC operating point, we can get the small signal equation for the circuit.

$$sG_0x = A_0x + B_0u \quad (6)$$

$$sG_{0r}z = A_{0r}z + B_{0r}u \quad (7)$$

Projection techniques (Eqn. 3) can be used to represent Eqn. 6 in a compact form for fast evaluation of the frequency response of the circuit (Eqn. 7). Multiple reduced order small signal equations can be generated for different values of design and process parameter values. These can then be interpolated to predict the small signal characteristics of the circuit at any new point in the "parameter space" (Fig. 2). A similar idea is presented in [3] to generate parametrized models for transient simulation of a circuit.

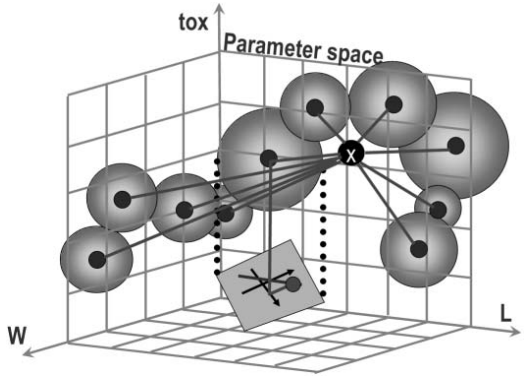

Fig. 2. Small signal models of the circuit can be sampled in the parameter space. Interpolation can then be used to predict the frequency characteristics of the circuit at any new point in the parameter space.

B. Extensions for Supporting Scalability

In recent extensions to this modeling methodology [17], a rigorous training strategy was used to generate trajectory points that sample as much of the reachable state-space of the circuit as possible. The resulting models would thus be robust across a wider variety of test inputs. Instead of interpolating all the trajectory linearizations, reduced order equations are generated at a new point in the state-space by interpolating only the k closest trajectory samples. This strategy works because the weighting function (Eqn 4) has been chosen in such a way that the resulting weight values are non-zero only for a few closest neighbors of the current point in the state-space. Efficient search schemes have been proposed in [17] to find the k nearest neighbors of a given point in a high dimensional state-space. The search time for these algorithms is nearly independent of the number of trajectory samples. Thus, robust macromodels with efficient model evaluation times can be generated for system level simulation of analog circuits.

C. Inadequacies in Trajectory Methods

While implementing these methods, we observed some practical inadequacies in the proposed modeling methodologies:

- *Simulation speed-ups*: Model order reduction (MOR) methods have been the tool of choice for generating macromodels for linear circuits [12] [13]. This is primarily because they yield reduced state-space equations whose order (q) is significantly smaller than the order of the original set of state-space equations (N). In other words, the reduction matrix V is of size N×q, where $q \ll N$. A similar idea is used for generating reduced order linearizations for trajectory based models. However, the reduction matrix V, in this particular case, is not the same as that for the linear case. Rather it is composed of a large number of individual reduction matrices corresponding to each linearization point; all combined into one through a biorthonormalization algorithm (Algorithm 1). The number of columns in matrix V_{agg} in step 11 of Algorithm 1 is of the order of $O(s)$. Performing singular value decomposition (SVD) of V_{agg} and picking columns corresponding to significant eigenvalues results in the reduction matrix V of size $N \times q$. It was observed through multiple experimentations that $q \leq N$. However, the value of q was not significantly smaller than that of N as is commonly observed while generating the reduction matrices for state-space equations for linear systems. Thus, the size of the interpolated reduced order state-space equations computed for predicting the dynamics of the non-linear system at a given point in state-space is not significantly different from that of the original state-space equation. Hence, simulation speed-ups reported in published literature using trajectory based models are smaller than those commonly observed while applying MOR techniques to linear systems.
- *Variational Models*: The idea of storing piecewise linearized samples of a non-linear circuit can be extended to predict the small signal behavior of the circuit across parameter variation – process as well as design parameters (Fig. 2). In constrast to approaches such [19] which build efficient nonlinear response surface from simulation samples, our interest is in extending the trajectory platform to generate

these samples, across both process and design variable perturbations, in the most accurate and efficient manner. The first such attempt in this direction was [3], which aims to model the complete non-linear characteristics of the circuit across process parameters. However, our recent experience with the localized reduction strategy suggests that the improvements obtainable by the global reduction strategy of [3] may be severely limited when generating models over large range of parameter variations. This is primarily because of merging together of large number of reduction matrices as described in the previous paragraph. The problem is quite common in circuits with complex nonlinearities e.g., transistor models like BSIM3.

Earlier approaches based on trajectory based models [14] [8] [5] [3] did not face this problem because of naive training strategies (model trained across a few waveforms at most) and simplistic circuit examples. As an illustrative example,

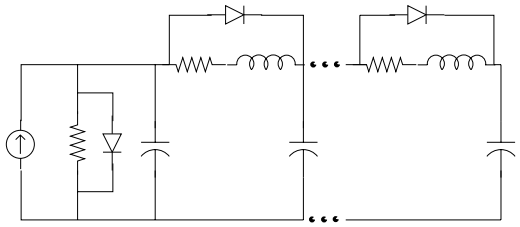

Fig. 3. Schematic of a nonlinear transmission line circuit similar to the ones used in [14] [3].

we present here a nonlinear transmission line circuit (Figure 3) similar to the ones used in [14] [8] [3]. All the resistor (R), capacitors (C) and and inductors (L) are linear elements. The diode is a nonlinear element represented by Equation 8.

$$i_d(v) = I_d(e^{\alpha v} - 1) \tag{8}$$

Here, $\alpha = 1/v_t$, where v_t is the threshold voltage. The number of repeated blocks, composed of R, L, C and nonlinear diode elements, that we used was 400 resulting in a state space equation of order 799 (N). After training across a single training waveform, the reduced order state space equation turned out to be of order 31 (q). When we generated a reduced order basis across variations in the parameter α for the same training waveform (using the algorithms described in [3]), the resulting reduced order state space turned out to be of order 33. We varied the parameter α across *4 orders of magnitude* during generation of this reduced basis. This extremely small increase in the dimensionality of the reduced state space is primarily because of the architecture of the circuit example. The circuit is structured in such a way that the voltages at the capacitive nodes always have a strong correlation between them. This prevents excitation of the circuit to different ("new") regions of its state space even with wide variations in the nonlinear diode parameter. On the other hand, if we take a complex analog circuit with BSIM3 [1] transistor models (Figure 6) as an example, the reduced order state space for a particular single training input was 9 compared to the original state space of order 24. However, when the model was trained across threshold voltage as the process parameter, the final state space equation turned out to be of order 21. This was for only a 30% variation in threshold voltage. Thus, we can see that the biorthogonalization algorithm for generating reduced order models across parameter variations [3] leads to inefficient models in the case of real circuit examples.

- *Loading Effects*: An important inadequacy of the trajectory based models that has been incompletely addressed to date is that of capturing loading effects. We macromodel so that circuit-level models can be inserted in system-level contexts and simulated. Previous approaches demonstrated the feasibility of building trajectory models for an individual circuit. Some prior methods [17] [6] have addressed the issue of inserting such models back into the simulator in a system context. A method to capture the circuit behavior in presence of different sets of output loading environment has been presented in [6]. In the proposed method, this is done by training the model across a range of loading conditions. The generated trajectory points are then merged into a single trajectory model for the circuit. The generated models, however, do not load the other circuits they are connected to. Also, the model accuracy is strongly dependent on the type of loading conditions in which the model was initially trained. No systematic approach for capturing the input and output loading effects of the circuit being macromodeled has been suggested.

We address these problems in the following section.

III. Local Reduced Order Models

One drawback of current trajectory based models is the limited simulation speed up that has been achieved to date. Simulation speed-ups up to $\sim 10X$ have been reported. We believe that there is a significant scope for improvement out here. The primary reason for the small simulation speed-up is the very small reduction in the order of the reduced state-space equation as has been discussed in section II-C. Merging a large number of reduction matrices corresponding to individual circuit linearizations gives us a common reduced order basis which may be applied across all the trajectory points. Earlier implementations [14] [4] [8] [15] of trajectory schemes interpolated all the circuit linearizations to predict the dynamics of the system at a new point in the state-space. This introduced a constraint that the reduced order equations at *all* the linearization points had to be generated using the *same* reduction basis (V_{common}). Using a different basis for different linearization points would make their interpolation expensive, if not impossible. However, recent improvements in the modeling methodology [17] (section II-B) that support robust training, thereby generating a large number of trajectory points, do not use all the linearizations for interpolation. Rather, they interpolate only a few ($k \sim 5$) nearest neighbors to predict the dynamics at the current point in state-space.

Since we are interpolating a chosen set of a few nearest points each time, we only need to make sure that the reduced order linearizations at these *few* points have been generated from the *same* reduced basis. This provides us with an excellent opportunity to use a much more compact *local* reduction matrix of size $N \times q'$ ($q' \ll N$) for the k nearest linearizations.

The reduction matrix needs to be composed of individual Krylov subspaces for only these k trajectory points and not for all the points in the model. In other words, the *for* loop in Algorithm 1 needs to run for $i = \{1, 2, ..., k\}$ instead of $i = \{1, 2, ..., s\}$ (where s is the number of trajectory samples in the models, typically on the order of 10,000-100,000). We call this smaller reduction matrix V_{local}.

Generating Krylov subspaces and reduced order state-space equations from the original set of equations of order N is an expensive process which cannot be performed while evaluating the models. Therefore, we need to compute these reduced order basis and the corresponding set of reduced linearizations during the model generation stage itself. However, this poses a new challenge for us. We need to compute reduced order bases for all different k nearest neighbors that we could encounter during the model simulation process as the model moves in its the state-space. In order to guarantee that we do have the reduced order equations for the k nearest trajectory points using the same reduction matrix at each point of the simulation trajectory, we need to compute a reduction matrix for a large number of groups of trajectory points such that: *for any point in the state-space, there exist its k nearest trajectory samples that belong to the same group.*

An obvious way of ensuring this would be to make $\binom{s}{k}$ groups of trajectory samples and build a reduction basis for each one of them. Using conservative values for the number of trajectory samples and nearest neighbors required for interpolation ($s = 1000$, $k = 5$), tells us that the number of groups that we would have to handle would be $\sim 10^{13}$. This brute force approach would obviously create groups of points that are spatially located far from each other and hence would never be used. In other words, they will not be the k nearest neighbors for *any* given point in the state-space.

To make the problem tractable, we change the requirement for the groups slightly. We create groups of trajectory linearizations such that for each of these linearizations (x_i), its k' nearest neighbors lie in the same group. If any new point in the state-space (P) is closest to a trajectory linearization x_1, then the two points should share a few nearest neighbors between them. This assumption is not always true as can be seen in Figure 4 for highly skewed points.

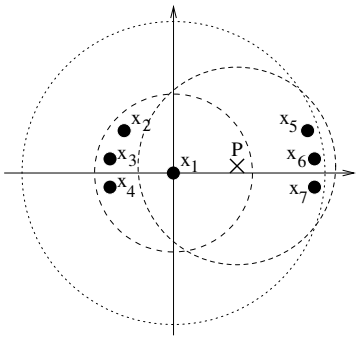

Fig. 4. Trajectory point x_1 and new point P may not share common nearest neighbors for $k' = k$ (dashed lines), even though x_1 is the nearest neighbor of P. However, making $k' > k$ (dotted line) increases the probability of sharing k common neighbors.

However, we could avoid this condition with high probability if we make $k' > k$. For a new point P in the state-space whose nearest neighbors do not belong to the same common group sharing the same reduction matrix, we could go back to the common reduction matrix (V_{common}) and the corresponding reduced order linearizations obtained by considering all the trajectory points together (Algorithm 1). The worst case average number of copies of reduced state-space equation that would need to be kept for each trajectory point corresponding to different reduced bases is $\sim k'$. This is not a significant penalty on model size because all these new reduced order state-space equations are of order q' where $q' \ll q \leq N$. However, in practice, we do not need to keep these many copies for each point. We observed that the new *local* models were approximately of the same size as the ones generated using a single reduction basis.

Algorithm: We start with the trajectory linearizations $\{x_1, .., x_s\}$ that we obtained during model training. We group them into disjoint clusters $\{c_1, .., c_l\}$ of trajectory points obtained through Gaussian Mixture Models (GMMs) [2]. Each of these gaussian clusters would have two types of trajectory samples as its member elements (Figure 5)–

1) Points with a very high probability of being a member of their particular cluster. These points typically lie near the core of the Gaussian distribution.
2) Points with very low probability of membership. These points populate the edge of the distribution and are very close to other GMM clusters.

For fringe points belonging to a cluster with very low membership probability, we compute their k' nearest neighbors and the respective GMM clusters they belong to. We repeat this process for all the fringe points of all the GMM clusters. We group all the fringe points and their k' nearest neighbors into new clusters $\{c_{l+1}, .., c_m\}$ such that in each such group, the member elements and their k' nearest neighbor belong to the *same* set of GMM clusters. The resulting final set of clusters $\{c_1, ..., c_l, c_{l+1}, .., c_m\}$ are the *local* groups for which we compute the local reduction matrices (V_{local}). These reduction matrices are then used to project the trajectory linearizations of their particular group down to a reduced order state space.

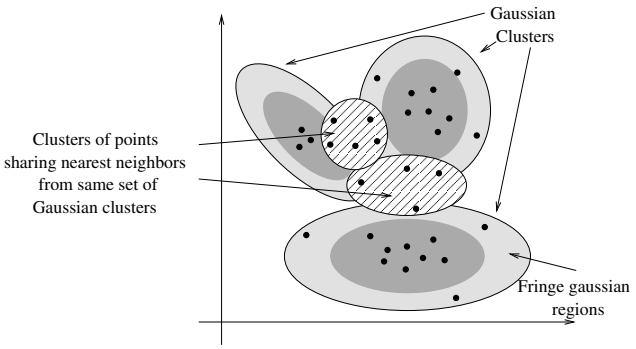

Fig. 5. Creation of clusters such that each linearization point and its k' nearest neighbors belonging to the same cluster.

Model Evaluation: The nearest neighbor search for finding the k nearest neighbors is performed in the reduced order state-space obtained using V_{common}. Once these neighbors are found, we move to a new reduced state-space using V_{local} corresponding to the common cluster that the nearest neighbors belong to. Model interpolation and evaluation is performed in the reduced state-space after which we return back to the common reduced space corresponding to V_{common} for the next nearest neighbor search using Eqn 3. In the rare case when we can not find a precomputed common cluster for the nearest neighbors, we use state-space equations corresponding to V_{common} for model evaluations.

Compact models: The local reduction matrices are computed for groups of trajectory linearizations of size $\sim k'$. Thus, the SVD computation in step 11 of Algorithm 1 is performed on matrix V_{agg} whose number of columns is of the order of $O(k')$ where k' is independent of the type of circuit being macromodeled. Thus, the size of the local reduced order models is almost independent of the order of original state-space equation (N) as well as the number of trajectory linearizations (s). Therefore, the models generated using the algorithm, would be smaller and faster for bigger circuit sizes.

IV. Generating Variational Models

A rather useful side-effect of the mechanisms developed in the previous section is we create a very natural platform for variational macromodels – that is, models parametrized by device or process parameters. In this section, we show how to use the piecewise linear model order reduction approach to model a circuit's small signal characteristics across its parameter space. Multiple small signal matrix equations (Eqn. 6) are generated – one for each sampled point in the parameter space. These linear equations are then interpolated using a weighting function as given in Eqn. 9:

$$w_i(z) = \frac{(z_i - z)^2}{\sum_{i=1}^{s} (z_i - z)^2} \tag{9}$$

The weighting function is different from Eqn. 4. For the trajectory case, we had to ensure that the interpolated matrices were stable. This was done by skewing the weight function towards the nearest interpolated point. Since, all the trajectory points themselves had stable state space equations, the heuristic was that a synthetic state space equation formed by heavily weighting the closest stable matrix equation along with some other equations would also result in a state space equation. There is no such constraint for the small signal model which is why we choose a smoother interpolating function, thereby requiring fewer sampled models.

For regions of the parameter space that might have linear characteristics, pruning ideas similar to [17] are used for removing reduntant information. We visit each sampled point in the parameter space and use all other points except the current one to predict the circuit's characteristics at this point. If the error between the sampled and interpolated matrices is less than some fixed threshold (acceptable error), we discard that particular sample from our model. We observe better pruning results because of smoother weight function being used in this case (Eqn. 9). Using the projection basis V for each sampled model, local reduction matrices V_{local} are generated for groups of these sampled parameter space points. These overlapping local groups are generated by the same algorithm as described in Section III. For efficiency purposes, the pruning step can be done after the local model generation. Once the redundant points have been discarded, a final local model generation step can be applied on the remaining sampled points to create the final model for interpolation.

A practical application of the variational models can be in hierarchical synthesis of circuits. We can optimize for the parameter values of the macromodeled circuit so that the system level specifications are met. For such applications, interesting regions of the parameter space should be modeled. Performing a brute force sampling of the parameter space can make the model too large in terms of memory requirements and hence, inefficient for fast evaluation. Our modeling infrastructure can easily handle model of sizes of $\sim$10,000 sampled points. The number of parameters is not a concern since our modeling flow can easily handle > 24 dimensional searches [18].

V. Capturing Loading Effects

In our new modeling tool, we have also added the capability to capture the loading effects of the circuit. We start with the circuit linearization equations at a particular point in state-space obtained during model training. We select the KCL equation for the nodes of interest (input and output). A simplification that we introduce is to connect all other nodes of the circuit to ground. Thus, we get the following set of equation:

$$C\dot{x}_{in/out} + Gx_{in/out} = I \tag{10}$$

This equation is exactly similar to the KCL equation one would get if the input/output node were connected to one resistance, capacitance and current source. Thus, the C and G of Eqn. 10 become the equivalent input/output capacitance and conductance for the circuit. These values are functions of the state-variable x_i. Thus, we would get different input and output impedance values for different trajectory samples. The impedance value at a new point in the state-space, during model evaluation, is computed by interpolating the equivalent C and G values for the different trajectory samples using the same weighting scheme as given in Equation 4.

VI. Experimental Results

We have implemented a version of the piecewise model order reduction (PW-MOR) modeling framework built into Berkeley-SPICE3f5 ("SPICE3") that supports BSIM3 transistor models. Our infrastructure supports large training runs, automatic model extraction, model pruning and native simulation support for generated macromodel inside the SPICE simulator itself. A simulateable circuit netlist is all that is required from the user to generate the macromodel for a given circuit. The generated macromodel can be used in another SPICE netlist

as a new SPICE element ready to be simulated along with other circuit elements. In this section, we present some of the results obtained by using our new modeling tool for generating macromodels for analog circuits.

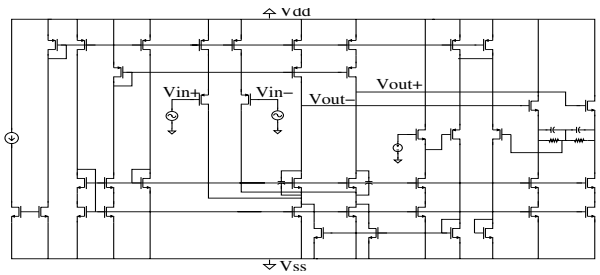

Fig. 6. Schematic of folded cascode opamp with common model feedback.

We generated a trajectory based macromodel for a 40 transistor folded cascode opamp circuit with common mode feedback stage (Figure 6). We trained the model using a varied range of input waveforms with different amplitudes and frequencies. The final macromodel consisted of $\sim$8000 trajectory points. The original state-space equation was of order 24(N). We got reduced set of equations of order 19(q) by using a single reduction matrix(V_{common}) to reduce all the sampled state-space linearization. Using the new localized model order reduction technique generated equations with order ranging from 7-11(q'). We used $k' = 30$ to generate clusters for computing the local models. Interpolation of state-space equations were done with 5 nearest neighbors in our experiments ($k = 5$). The macromodel without the local reduced order models was $\sim 9X$ faster to simulate than the transistor level circuit. After we included the localized reduction of state-space linearizations, the macromodel became $\sim 18.4X$ faster. Thus, we got an extra 2X simulation speed-up in the macromodel due to the introduction of the localized models. In another example, a filter circuit (Fig. 7) consisting of three copies of the opamp circuit was macromodeled as a single circuit block. Local reductions reduced the size of the state space equation from 70(N) to 12-20(q'). The macromodel simulated $\sim$30X faster than the transistor level circuit.

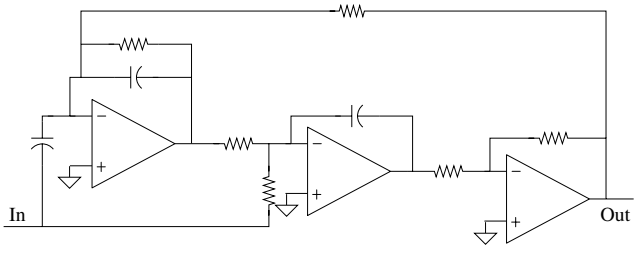

Fig. 7. Circuit schematic of the filter being macromodeled. Component opamps are the ones from Figure 6. The complete filter block was macromodeled as one single circuit.

In the experiment, we observed smaller macromodel simulation time due to faster interpolation and solution of the state-space equation due to smaller sizes of the localized reduced

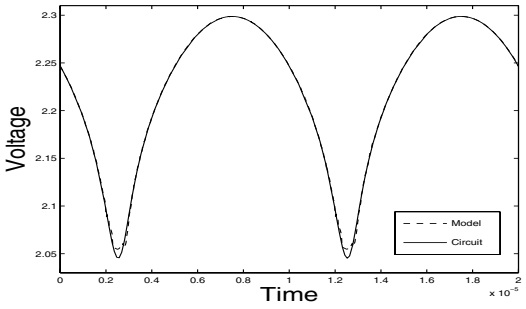

Fig. 8. Almost indistinguishable waveforms produced by the filter circuit of Figure 7 and its macromodel for a large signal input. Average error is less than 2%.

TABLE I
MACROMODELING DATA FOR DIFFERENT CIRCUITS

Circuit schematic	Size of full state space(N)	Size of reduced state space (q')	Simulation Speed-up
Opamp	24	7-11	18.4
Filter	70	12-20	29.7

order models. Since the nearest neighbor searches were still done in the common reduced order basis space, we observed an increase in the fraction of time the macromodel evaluator spent in nearest neighbor searches. The evaluator now spent $\sim$12% of its time in approximate nearest neighbor searches. Nearest neighbor search took $\sim$5% of the model evaluation time without the local models.

In another experiment, the small signal model of the opamp circuit (Fig. 6) was generated in the parameter space. Table II shows the range of the various parameters for which the model was generated that resulted in $\sim$800 sampled points. Local reduced order models were generated with final model sizes of 4-10. To test the accuracy of the generated model, gain and bandwidth of the circuit was computed at 500 randomly chosen points in the parameter space with the ranges listed in Table II. Average errors of $< 3\%$ were observed against ac simulation of SPICE. Figure 9 compares the frequency response of the circuit and model at a particular point in the parameter space. Pruning the sampled points reduced the size of the model from $\sim$800 to $\sim$300. Average model error increased to $\sim 5.5\%$. Simulation speed-ups in the range of 200-400X were observed. This large speed-up number is primarily because SPICE has to first do a DC convergence before running the AC analysis while our model performs simple interpolation. The model pruning step does not drastically change the simulation speed up number since our modeling scheme is fairly robust to model sizes [18]. However, pruning is very useful to contain the memory requirements of the model.

The final experiment was set-up to test the quality of the input-output models generated by our modeling methodology as discussed in section V. We generated the trajectory models for a stand alone 5-stage ring oscillator VCO circuit. The loading effects of the circuit were also captured during the model generation phase. We then inserted the model into a

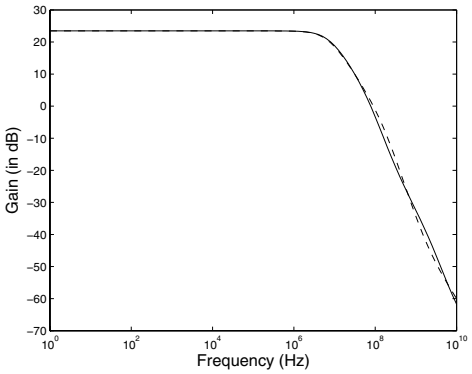

Fig. 9. Plot comparing the frequency response of model (solid) and circuit (dashed) under parameter variation at an untrained point in the parameter space.

TABLE II

PARAMETER RANGES FOR SMALL SIGNAL MODEL GENERATION

Parameter Name	Range of Values
tox	4.7-6.7nm
W-diff pair	90-160μm
Input DC bias	500-1500mV
Cascode cap	100-1500fF
W-current mirror	2-15μm

phase locked loop (PLL) of figure 10. All the other blocks of the PLL were composed of native SPICE elements. The PLL circuit that was used was a very simplistic one with unity frequency multiplication (input frequency = output frequency). Plot of waveforms (Figure 10) at $V_{control}$ of the VCO showing the frequency capture by the PLL shows a close match between the macromodel and circuit simulation outputs.

VII. CONCLUSIONS

We have presented a tool for efficient and accurate modeling of analog circuits using trajectory based models. The tool supports three new ideas: smaller, localized reduced order models at the trajectory linearization points, small signal variational models in the parameter space and state dependent circuit loading characteristics. The *local* models give us $\sim 2X$ extra speed up over standard trajectory models. The same idea extended for ac variational models provides 200-400X speedup for computing small signal characteristics against SPICE, thereby, offering hierarchical design space exploration opportunities. The input and output impedances in the macromodel help us get better accuracy while simulating the model in a system level context. Our ongoing work focuses on both algorithmic and engineering implementation improvements to our infrastructure. We believe this work will be an excellent platform from which to propagate trajectory models into more widespread use.

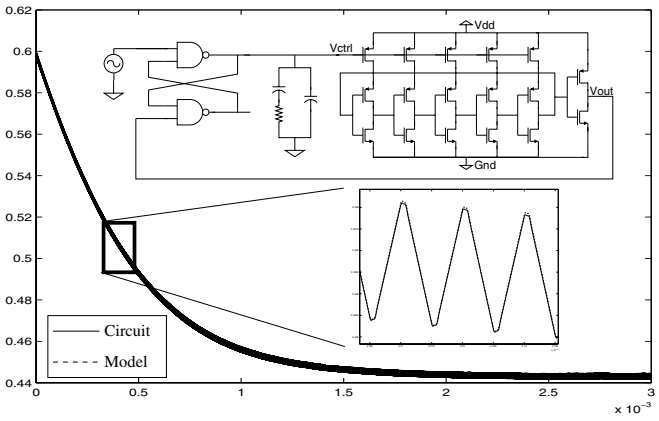

Fig. 10. Plot comparing the waveforms produced at V_{ctrl} of the VCO during frequency capture by the PLL simulated with modeled and circuit level VCO.

VIII. ACKNOWLEDGEMENTS

The authors would like to thank Joel Phillips (Cadence Berkeley Labs), Amith Singhee (CMU), Dmitry Vasilyev (MIT) and Paolo Nenzi (University of Rome) for their insightful suggestions and help. This work was supported by the Pittsburgh Digital Greenhouse.

REFERENCES

[1] BSIM3 transistor models. http://www-device.eecs.berkeley.edu/ bsim3.
[2] A.P.Dempster, N.Laird, and D.Rubin. Maximum-likelihood from incomplete data via EM algorithm. In *J. Royal Statistics Society B39*, 1977.
[3] B.Bond and L.Daniel. Parameterized model order reduction for nonlinear dynamical systems. In *ICCAD*, 2005.
[4] N. Dong and J.Roychowdhury. Piecewise polynomial nonlinear model reduction. In *DAC*, pages 484–489, 2003.
[5] N. Dong and J.Roychowdhury. Automated extraction of broadly applicable nonlinear analog macromodels from SPICE-level descriptions. In *CICC*, 2004.
[6] N. Dong and J.S.Roychowdhury. Automated nonlinear macromodelling of output buffers for high-speed digital applications. In *DAC*, pages 51–56, 2005.
[7] D.Ramaswamy. Automatic generation of macromodels for microelectromechanical systems (MEMS). In *Ph.D. dissertation, Massachusetts Inst. Technol., Cambridge, MA,*, 2001.
[8] D.Vasilyev, M.Rewienski, and J.White. A TBR-based trajectory piecewise-linear algorithm for generating accurate low-order models for non-linear analog circuits and mems. *DAC*, pages 490–495, 2003.
[9] E.Grimme. Krylov projection methods for model reduction. In *PhD Thesis, UIUC*, 1997.
[10] F.Wang and J.White. Automatic model order reduction of a microdevice using the arnoldi approach. In *Proc. Int. Mechanical Engineering Congr. and Exposition*, pages 527–530, 1998.
[11] K.Gallivan, E.Grimme, and P.V.Dooren. Asymptotic waveform evaluation via a lanczos method. In *Applied Mathematics Letters*, pages 75–80, 1994.
[12] L.T.Pillage and R.A.Rohrer. Asymptotic waveform evaluation. In *TCAD*, pages 352–366, 1990.
[13] A. Odabasioglu, M. Celik, and L. Pileggi. PRIMA: Passive reduced-order interconnect macromodeling algorithm. In *TCAD, Vol 17, No 8*, pages 645–654, 1998.
[14] M. Rewienski and J. White. A trajectory piecewise-linear approach to model order reduction and fast simulation of nonlinear circuits and micromachined devices. In *TCAD*, pages 155–170, 2003.
[15] M. J. Rewienski. A trajectory piecewise-linear approach to model order reduction of nonlinear dynamical systems. *PhD Dissertation, MIT*, 2003.
[16] R.W.Freund. Krylov-subspace methods for reduced order modeling in circuit simulation. *J. of Computational and Applied Mathematics*, pages 395–421, 2000.
[17] S.K.Tiwary and R.A.Rutenbar. Scalable trajectory methods for on-demand analog macromodel extraction. In *DAC*, pages 403–408, 2005.
[18] S. K. Tiwary. Scalable trajectory methods for on demand analog macromodel extraction. In *Phd Thesis (in preparation), CMU*, 2006.
[19] X.Li, J.Le, L.T.Pileggi, and A.Strojwas. Projection-based performance modeling for inter/intra-die variations. In *ICCAD*, 2005.