Analog Macromodeling using Kernel Methods

Joel Phillips
Cadence Berkeley Labs
Cadence Design Systems
San Jose, CA 95134, U.S.A.
jrp@cadence.com

João Afonso Arlindo Oliveira L. Miguel Silveira
INESC ID/IST, Cadence European Labs
Technical University of Lisbon
Lisbon, Portugal
{jpa,aml,lms}@inesc-id.pt

ABSTRACT

In this paper we explore the potential of using a general class of functional representation techniques, kernel-based regression, in the nonlinear model reduction problem. The kernel-based viewpoint provides a convenient computational framework for regression, unifying and extending the previously proposed polynomial and piecewise-linear reduction methods. Furthermore, as many familiar methods for linear system manipulation can be leveraged in a nonlinear context, kernels provide insight into how new, more powerful, nonlinear modeling strategies can be constructed. We present an SVD-like technique for automatic compression of nonlinear models that allows systematic identification of model redundancies and rigorous control of approximation error.

1. INTRODUCTION

In the past decade, the need to reduce the time and risk required for implementation of analog/mixed-signal (AMS) designs has become increasingly evident. This is particularly true because of the complex designs arising in the communications area. As a result, interest has grown, as witnessed by the emergence of the analog modeling languages Verilog-AMS and VHDL-AMS, in more structured design methodologies that utilize behavioral modeling in the specification, design, and verification stages.

A particularly vexing problem is the verification of the operation of analog sub-components when integrated into a larger design such as a wireless communications system. Macromodels are a potential solution to this problem: they can greatly accelerate simulation-based verification, are simpler to manipulate, as they offer isolation from the details of the lower-level implementation, and offer some degree of IP-protection, as they hide many of the implementation details. The problem with macromodeling strategies is, first, assuring that they can be made sufficiently accurate, and, second, the large effort required to create them by hand. Both concerns motivate the development of mathematically rigorous automatic macromodeling algorithms.

There has been considerable success, mostly in the context of interconnect and package modeling, devoted to macromodeling of linear, time-invariant passive components via model reduction algorithms for large-scale linear systems [1, 2, 3, 4]. Recently, extensions of these techniques were proposed for time-varying and weakly nonlinear systems [5, 6, 7, 8]. The trajectory algorithms of [9, 10] are attempts to extend these Krylov-subspace based approaches to more strongly nonlinear models.

ICCAD'03, November 11-13, 2003, San Jose, California, USA.

In this paper we consider projection-based methods for model reduction of nonlinear systems. We will not directly address the choice of the projection itself. Instead we will focus on an issue intrinsic to nonlinear modeling: efficient representation of nonlinearities whose structure is not known a-priori. Necessarily, the fundamental difficulty becomes the manipulation of nonlinear functions that induce features in high dimensional spaces[11].

Our intent is to demonstrate the utility of kernel-based representation techniques [12, 13] in manipulating such high-dimensional data. Kernel methods, perhaps the most recently popular example of which are support vector machines (SVMs), provide several potential advances in analog macromodeling. First, they provide a common, systematic and computationally convenient framework for bringing to bear a wider class of mathematical models to the reduction problem. As an example, we will show how to formulate an approach that includes both the polynomial and piecewise-linear techniques as special cases. Second, they bring potentially large increases in computational efficiency. For example, the polynomial based methods require manipulating a set of functions whose number grows exponentially with degree and model order. We will show how to manipulate the same polynomial basis functions at a cost that is at worst cubic in the number of samples in a chosen set. When the samples are well chosen, dramatic gains in efficiency can be obtained. Third, kernel methods, equipped with appropriate techniques of statistical inference, can lead to increased mathematical rigour in the model construction process. For example, in the piecewise-linear technique of [9] there are parameters such as the number of linearization points, the location of the linearization points, the size of the linearization regions, and the way of combining the linear models, that are chosen in a more or less ad-hoc manner. As we shall see, in the framework of kernel methods, these problems can be tackled in a systematic and well understood manner, since it is possible to develop techniques for exposing and exploiting redundancy in *nonlinear* functions.

The tangible results of these advantages will be shown to be smaller models, a more systematic way of choosing parameters in the modeling process, and stronger guarantees on the approximation error.

2. REDUCTION BACKGROUND

2.1 Two-Stage Projection Methods

Projection-based methods for reducing linear systems of the form

$$E\frac{dx}{dt} = Ax + Bu, \quad y = Cx + Du \tag{1}$$

where $u(t)$ represents system inputs, $y(t)$ system outputs, $x(t)$ system state, are now highly developed. Reduction is performed by

drawing an approximate state vector $x^{(r)} = Vz$ from a lower dimensional subspace defined by the column span of V. With an orthogonal projection of the equations (1), the reduced model is of the same form,

$$E^{(r)}\frac{dz}{dt} = A^{(r)}z + B^{(r)}u, \quad y = C^{(r)}x + D^{(r)}u \tag{2}$$

with the reduced model matrices defined by

$$A^{(r)} \equiv V^T AV \quad B^{(r)} \equiv V^T B \quad E^{(r)} \equiv V^T EV \quad C^{(r)} \equiv CV, D^{(r)} = D. \tag{3}$$

Recently, there have been several techniques proposed in an attempt to generalize reduction techniques to nonlinear systems, which, for simplicity, we will consider in the somewhat restricted form

$$\frac{dx}{dt} = f(x) + Bu, \quad y = Cx + Du \tag{4}$$

where $f(x)$ is an arbitrary nonlinear function. The projection formula may be applied as in the linear case to obtain (with the additional constraint $V^TV = I$ imposed for convenience of notation) a reduced model [14, 15, 16, 17, 6]

$$\frac{dz}{dt} = V^T f(Vz) + V^T Bu, \quad y = CVz + Du. \tag{5}$$

This method of reduction requires two computational steps. First is the basis, or state-variable selection, computation, which boils down to choice of the projection matrix V. Second, and perhaps more important, is the computation of the projected model in the reduced space. For linear systems, projection computation (Eq. 3) is trivial, but projection computation is very difficult in general. No practical reduction algorithm can be developed without an efficient means of performing this computation in fairly general settings. Consider the cost of evaluating $V^T f(Vz)$. This function can in principle be evaluated explicitly in the "reduced" model, by reference to the original function $f(x)$, but the difficulty is that, while the reduced model is of smaller dimension than the original, without a more compact representation of $f^{(r)}(z) \equiv V^T f(Vz)$, no practical acceleration of computation is achieved.

Two techniques have recently been proposed that can reduce the evaluation cost of $f^{(r)}(z)$. Several authors [5, 6, 7, 8, 18] have proposed using multi-dimensional polynomial representations. In this approach, the nonlinear function is expanded in polynomial series,

$$f(x) = A_1x^{(1)} + A_2x^{(2)} + A_3x^{(3)} + \cdots \tag{6}$$

with $x^{(1)} \equiv x, \quad x^{(2)} \equiv x \otimes x, \quad x^{(3)} \equiv x \otimes x \otimes x,$ etc, and the $A_k \in R^{n \times n^k}$ are the multi-dimensional (tensor) polynomials coefficients of the expansion. Under the projection operation, the polynomial terms may be compressed to lie in $R^{q \times q^k}$ by application of the rule

$$A_k^{(r)} = V^T A_k (V \otimes V \otimes \cdots V). \tag{7}$$

The order-m coefficient, with q states preserved, is a factor of $(q/n)^m$ smaller than the original. Even so, because the final coefficients contain q^m terms, if m is the order of polynomial approximations, these methods are in practice only suitable for fairly weak nonlinearities, those that can be represented by a second- or third- order polynomial expansion.

Another drawback of these methods is that the coefficients A_k can be difficult to obtain. Obtaining the coefficients by Taylor series, for example, requires analytical manipulations that are labor-intensive, and not practical to apply when the function $f(x)$ is given by a "black-box", or when a very large number of complex functions is involved, as is the case in modern semiconductor device models.

The limitation of the polynomial-based techniques was one motivation behind development of the trajectory-piecewise-linear algorithm [9], which has been shown capable of treating somewhat stronger nonlinearities. In this approach, the function f is approximated by a linear combination of affine models,

$$\hat{f}(x) = \sum_k w_k(x)\left[f(x_k) + A_k(x - x_k)\right]. \tag{8}$$

After projection, the reduced $f^{(r)}(z)$ then has the similar form

$$\hat{f}^{(r)}(z) = \sum_k w_k(z)\left[V^T(f(x_k) - A_k x_x) + V^T A_k Vz\right]. \tag{9}$$

Next we will discuss further extension of the class of of nonlinear function representations suitable for reduction applications. For the remainder of the paper, we will assume that an acceptable technique for choosing the basis V is available. For example, the basis matrix may be chosen using the time-series analysis techniques [14, 15, 17] or local linearizations [9, 16]. We will concern ourselves with the second part of the model reduction challenge, the reduced-model regression problem, finding an efficient way to compute and represent $f^{(r)}(z) = V^T f(Vz)$ with an approximate $\hat{f}^{(r)}(z)$. We will also show some interesting interactions between regression and reduction via the projector V.

2.2 Kernel Methods

For the moment, let us leave aside the question of representing $f^{(r)}(z)$, and consider representing a scalar function $g(x) : \mathbb{R}^n \to \mathbb{R}$ of multi-variable argument. Generalization to vector functions can be done entry-wise. An interesting class of approximations are those of the form

$$\hat{g}(x) = \sum_{k=1}^{N} a_k K(x, x_k) \tag{10}$$

where $x \in \mathbb{R}^n$ is the evaluation point, $K(x,y) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is the *kernel*, and the N vectors $x_k \in \mathbb{R}^n$ we will denote the "support vectors."

The most basic kernel methods [13] use simple functional forms for the kernels, and pick the coefficients based on local properties of the function. For example, tabulating the function $g(x)$ at some sample points x_k, and using the radial Gaussian kernel, $K(x,y) = \exp(-\beta||x-y||^2)$, will lead to an approximation of $g(x)$ based on a weighted local average,

$$\hat{g}(x) = \sum_k g(x_k)K(x, x_k) \tag{11}$$

Better results are often obtained by renormalizing the kernel so the weights sum to unity at each point, which is equivalent to using the so-called Nadaraya-Watson (NW) kernel [13],

$$K'(x,y) = \frac{\sum_k K(x,x_k)K(y,x_k)}{(\sum_k K(x,x_k))(\sum_k K(y,x_k))}. \tag{12}$$

The similarity to the piecewise-linear approximation in Eqn. (9) should be apparent, and in fact in [9], the exponentially-weighted radial functions with normalization as in Eqn (12) were used to obtain the weights $w_k(z)$.

Now consider the general regression problem, representing the function $g(x)$ in the basis of M functions h_k, $k = 1, \ldots, M$,

$$\hat{g}(x) = \sum_{k=1}^{M} \beta_k h_k(x) \tag{13}$$

which can be written

$$\hat{g}(x) = [h_1(x) \;\; h_2(x) \;\; \cdots h_M(x)]\,\beta \tag{14}$$

where $\beta = [\beta_1; \cdots; \beta_M]$ is the vector of the expansion coefficients from (13). Instead of using local-averaging arguments, the coefficients β_k are chosen to minimize a *loss function*, $L(\beta) = V(e) + Q(\beta)$, where $V(e)$ is some measure of the approximation error $e = \hat{g} - g$ and Q is a regularization term chosen (roughly speaking) to prevent "over-fitting" of the data (i.e. to penalize choosing an approximant that is richer or more "complicated" than necessary).

The problem when representing multi-dimensional nonlinear functions is that the number of basis functions M in Eqn. (13) can be very large. For example, consider the multi-dimensional polynomials of degree up to d in a space of dimension n. One choice of basis functions h_k are the monomials $h_k(x) = x_1^{q_1} x_2^{q_2} x_3^{q_3} \cdots x_n^{q_n}$ of degree $q = q_1 + q_2 + \cdots q_n$ up to $q \leq d$. The number of monomials of a given degree grows combinatorically (there are $(d + n-1)!/(d!(n-1)!)$ monomials of degree d; the redundant tensor-product representations used in [6] are larger by more than a factor of 2^d). The space spanned by the basis functions h_k, sometimes called feature space, is thus very high dimensional. Thus any reasonably accurate approximant, even if "low" order, must rely on the *potential* usage of a fairly large number of basis functions.

The attractiveness of kernel methods is that, for suitably chosen basis functions, we can represent functions with series of the form in (13), but the results, and all intermediate computations, are of the form in (10), *which only involve N degrees of freedom.* In other words, the representation exists in the input space but all computations and evaluations can be conducted in the feature space provided by the kernels. When $M \gg N$ this represents substantial savings. Kernel methods attain efficiency by expressing all algorithmic operations in terms of inner products in this feature space, that is, sums of the form

$$< \Phi(x), \Phi(y) >= \sum_{k=1}^{M} h_k(x) h_k(y) \tag{15}$$

where $\Phi : x \mapsto (h_1(x), \cdots, h_M(x))$, is a map from the input space to the feature space, Thus the key to kernel methods is the connection between useful basis functions, and a kernel function that can be easily evaluated. For example, inner products in the space of inhomogeneous polynomials can be evaluated by using the kernel

$$K(x,y) = (1+ < x,y >)^d. \tag{16}$$

Likewise, kernel functions for bases of orthogonal polynomials can be obtained via the Christoffel-Darboux formula. Generally, for any set of functions that have a closed-form summation rule, we can find a kernel that reproduces the same function space. Splines and Fourier series are notable examples.

Conversely, we may choose to construct the (positive-definite) kernel, and let the basis be thus implicitly defined. A simple way to construct complex kernels is by taking sums, or coordinate-wise products, of simpler kernels. For example, taking the kernel to be a product of Gaussian radial basis function (RBF) kernels and polynomial kernels leads to a representation very similar to that used in [9] – more general, in fact, since it leads to a piecewise-polynomial, rather than just piecewise-linear, basis.

The connection between representations of the form in Eqns. (13) and (10) is not necessarily obvious, so we now show a restricted example [13] demonstrating the connection between kernels and approximation in the basis of $h_k(x)$.

Suppose we are given a set of N measurements $y_i = g(x_i), i = 1, \ldots, N$. To fit this data with a function $\hat{y} = \hat{g}(x)$, of the form (13), consider obtaining the expansion coefficients from minimizing the penalized least-squares error,

$$\min_{\beta} L(\beta) = \min_{\beta} \left\{ \lambda \|\beta\|^2 + \|\hat{y}(\beta) - y\|^2 \right\} \tag{17}$$

for some regularization parameter λ and $y = [y_1; \cdots; y_N]$. Defining the matrix $H_{jk} = h_k(x_j)$,

$$H = \begin{bmatrix} h_1 & h_2 & \cdots & h_M \end{bmatrix}, \tag{18}$$

the fit function values, $\hat{y}$, are given by $\hat{y} = H\beta$. This minimization problem can be solved in closed form, via the equations

$$-H^T(y - H\beta) + \lambda\beta = 0. \tag{19}$$

Now it will be convenient to substitute $\beta = H^T\alpha$. In this case, we have

$$\left[H^T H H^T + \lambda H^T \right] \alpha = H^T y \tag{20}$$

which after multiplying from the left by H becomes

$$(HH^T) \left[HH^T + \lambda I \right] \alpha = (HH^T) y \tag{21}$$

which can be solved by using

$$\left[HH^T + \lambda I \right] \alpha = y \tag{22}$$

to obtain the coefficients α. With this representation, defining

$$h = [h_1(x) \;\; h_2(x) \;\; \cdots h_M(x)] \tag{23}$$

the function has the representation

$$\hat{g}(x) = hH^T\alpha. \tag{24}$$

This has the form

$$\hat{g}(x) = \sum_{k=1}^{N} \sum_{m=1}^{M} h_m(x) h_m(x_k) \alpha_k \tag{25}$$

By defining the kernel

$$K(x,y) = \sum_{m=1}^{M} h_m(x) h_m(y) \tag{26}$$

this can be written more conveniently as

$$\hat{g}(x) = \sum_{k=1}^{N} \alpha_k K(x, x_k). \tag{27}$$

Note that the i, j entries of the matrix HH^T are given by $K(x_i, x_j)$, thus all operations can be performed by only reference to the kernel function.

The framework we just established can easily be extended to accomodate more generic objective functions than the one introduced in Eqn. (17). This will allow us to cast other procedures under the same framework, in particular, regression via Support Vector Machines (SVMs) [12]. Typically SVM's try to fit $y = g(x)$ by minimizing an objective function such as

$$L(\beta) = V(e) + Q(\beta) = C\frac{1}{N}\sum_{i=1}^{N} \|y_i - g(x_i)\|_\varepsilon + \frac{1}{2}\|\beta\|^2 \tag{28}$$

where $V(e)$ is known as the regularized risk functional, in essence a measure of the regression error ($\|.\|_\varepsilon$ is a particular measure to be discussed) and C is a constant determining the tradeoff between the error and the complexity penalizer $\|\beta\|^2$. Eqn. (28) is thought to encapsulate the main ideas behind statistical learning theory, namely that in order to obtain a good model for the data available, both

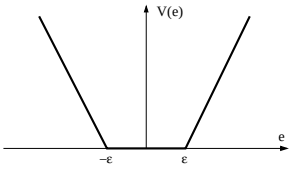

Figure 1: ε-insensitive penalty function $V(e)$.

training error and model complexity must be simultaneously controlled. SVMs typically choose the loss function $L(x)$ to be convex. This decision enables one to use the power of convex optimization, take advantage of the existence of fast, efficient solvers, and guarantee that the solution obtained is a unique, global optimum.

It is quite common in SVM's to pick $V(e)$ to be a penalty which is zero inside some epsilon, linearly growing outside (see Fig. 1). This penalty term leads to what is known as ε-insensitivity. All samples whose error margin measured against the representation is smaller than or equal to ε are not penalized in the loss function. Referring to Eqn. (27) which represents the model in terms of the kernels, and noting that the loss function also attempts to minimize a norm-related complexity penalty, those α_k that are related to samples which are within that ε-insensitive bound, can thus be optimally chosen equal to zero. This leads to a sparsification of the representation in (27): only samples which do not properly satisfy the optimal error criteria are needed to generate the representation. These are termed **support vectors**.

3. SAMPLING-BASED REGRESSION

The above discussion shows how we can potentially use kernel techniques to represent $f^{(r)}(z)$ with an approximant, $\hat{f}^{(r)}(z)$ using a richer class of functions than has previously been possible. However, increased flexibility means an increased number of choices to be made in constructing an algorithm. We must choose a specific form for the kernels and we must choose parameters for the kernels. More critically, we must choose the sample points z_i, and we would like as few as possible, as the cost of the algorithm is directly related to the number of support vectors N. In this section we will consider the relation between the choice of support vectors and accuracy of the approximation.

3.1 Complete Sampling

Ultimately we wish to represent $\hat{f}(z)$ by the sum

$$\hat{f}^{(r)}(z) = \sum_{k=1}^{N} \alpha_k K(z, z_k) \tag{29}$$

for some α_k, z_k ($1 \leq k \leq N$). The most general, and straightforward, way of choosing the parameters α_k for the kernel expansion is by fitting to a set of sampled $\hat{f}(z_k)$ for some set of sampling points z_k. The fitting procedure may use the least-squares procedure detailed above, the ε-insensitive loss function of the support vector machine (SVM) (presented in Section 2.2), or other metrics as desired. For now we will treat the fitting procedure as a black box. Once have performed the projection, then fit the function $\hat{f}^{(r)}(z)$, we will have a reduced model. The first question is how to pick the vectors z_k. Lacking any prior knowledge about the function, a reasonable choice is to perform random sampling. This is in some sense the least informative choice; however, should it give good results, it indicates that the function under consideration is "reducible" in the absence of large amounts of specific prior knowledge. This is very desirable for an automated procedure.

So for the moment suppose that we have chosen N independent vectors z_k, perhaps at random, perhaps by design. What can we achieve with simple choices? First, it should be clear that by using the NW-type kernels (Eqn (12)), we can (by inspection) reproduce the method of [9], if the z_k we are given are the linearization points. The "linearization points" in [9] will become our "support vectors."

From considering the space induced by the polynomial kernel, it is also evident that polynomial methods can be readily replicated. The space of polynomials up to a given order is a finite-dimensional space, i.e., M in Eqns (15) and (26) is a finite integer. From (26) it is thus clear that the polynomial kernels have finite rank M. With a sufficient number of sampling points, if the rank of the kernel matrix achieves M, we can exactly match a multi-dimensional polynomial function of the appropriate order. The best-fit polynomial depends on its construction, whether by Taylor series, least-squares over a region, the region of operation to consider itself being another degree of freedom, or some other method. Each of these choices, which can be made implicitly via choice of the points z_k, leads to a a particular polynomial model, and that model can be computed implicitly via kernels. Thus, kernel methods with polynomial kernel encompass the polynomial methods of [5, 6, 7, 8]. In and of itself, this is a useful result, because of the computational simplicity of the kernel methods, that only samples of the multi-dimensional functions are required. In contrast, analytic methods of obtaining the polynomial coefficients A_k, such as Taylor series, are essentially impractical in many contexts, in particular, circuit simulation, because of the size and complexity of the model equations involved.

3.2 Incomplete Sampling: No Free Lunch

The point of introducing the kernel methods was to be able to manipulate complex functions – higher order polynomials, for example. Note also, that the functions $K(x, x_k)$, as functions of x, lie in a space spanned by the $h_k(x)$. For a set of x_k, appropriately chosen, the $K(x, x_k)$ will span the full (Hilbert) space spanned by the $h_k(x)$. For exact representations of all possible functions, we need generally as many sample points N as basis functions M. We must trade either accuracy – representing some components of the Hilbert space approximately – or comprehensiveness – hoping that some components will not appear in the functions we need to approximate. However, problem-specific knowledge may reduce the number of basis functions we need in a specific context.

Consider fitting using polynomials of order 1 (i.e., affine functions). Suppose it is known that the target function $g(x)$ is also affine, $g(x) = c^T x + a$ for some $c \in \mathbb{R}^n, a \in \mathbb{R}$. Given $n+1$ support vectors that are linearly independent, it is always possible to recover $g(x)$ exactly. The situation with $m < n+1$ vectors depends on the choice of vectors. If the column span of the set of vectors happens to include c, the function will be exactly recovered. But exact representation can be done with only two vectors, if they are chosen wisely. More generally, the accuracy will depend on the angle between the subspace spanned by the sample vectors and c. This argument generalizes to more complex basis functions, for example, polynomials. If the image in the feature space of the the chosen set of candidate support vectors spans a given basis function, the representation of that basis function in the function samples can be recognized and reproduced, in principle without error. In general, the minimum achievable fitting error will depend on the norm of the projection of the sampled function orthogonal to the space implied by the kernel and the sample point.

The analogy to the moment-matching properties of the linear Krylov-subspace projection methods should be clear. If the column span of the projection matrix V in Eqn. (3) spans a Krylov-space,

moments will be matched in the final solution of the state equations. Here, we may match, approximately or exactly, components of a more general space used to approximate $f(z)$.

Thus, the regression problem boils down to picking sample vectors whose image will represent the image of the test vectors well. Kernel methods help with the mechanics of the representation, as well as the problem of regularization that we have not discussed in detail, but the central problem, or assumption, is really the selection of appropriate sampling vectors. It may be that simple selections (e.g., random) cover the space "well enough" for 5% or 10% accuracy. Or maybe not. Now we consider ways to restrict the choice of sample points in order to improve the accuracy/cost tradeoff.

4. BOTTOM-UP KERNEL PROJECTION

For a certain class of kernels, inner product kernels, an interesting result is available that allows us to *automatically* obtain a kernel representation of the reduced $\hat{f}(z)$, if we have a kernel representation of the original $f(x)$. This result is a generalization of the formulas (3) for linear functions, and (7) for polynomial functions. This technique may be useful since in many applications, circuits in particular, the nonlinearities in the original $f(x)$ live in a much lower dimensional space, and there are relatively few distinct types of them. It is possible, for example, to fit kernel models to each of the different types of semiconductor devices in a circuit, models which would need to fit over only two or three dimensions. Then the nonlinear $\hat{f}(z)$, which lives in a high-dimensional space, can be automatically constructed.

4.1 Projection Formulas

Suppose the kernel of interest is an inner product kernel, $K(x,y) = K(< x,y >)$, and suppose that the kernel representation for the "unreduced" system is available as

$$\hat{f}(x) = \sum_{k=1}^{N} a_k K(x, x_i). \tag{30}$$

Now consider performing the standard projection-based model reduction, with reduced state variable z, and $x = Vz$. For any component j of $\hat{f}$, we have

$$\begin{aligned} \hat{f}_j(Vz) &= \sum_{k=1}^{N} a_k K(Vz, x_k) = \sum_{k=1}^{N} a_{k,j} K(< Vz, x_k >) \\ &= \sum_{k=1}^{N} a_{k,j} K(< z, V^T x_k >) = \sum_{k=1}^{N} a_{k,j} K(< z, z_k >) \end{aligned} \tag{31}$$

where $z_k \equiv V^T x_k$. Thus, the substitution leads *automatically* to a kernel expansion in the space of reduced dimension! To finish, we consider the reduced vector function $\hat{f}^{(r)}(z) = V^T \hat{f}(Vz)$. In this case, with v_i the ith column of V,

$$\hat{f}_i^{(r)}(z) = v_i^T \hat{f}(Vz) = \sum_{j=1}^{n} v_i(j)\, \hat{f}_j(Vz) \tag{32}$$

$$= \sum_{j=1}^{n} v_i(j) \sum_{k=1}^{N} a_{k,j} K(< z, z_k >) \tag{33}$$

$$= \sum_{k=1}^{N} (\sum_{j=1}^{n} v_i(j) a_{k,j}) K(< z, z_k >) \tag{34}$$

$$= \sum_{k=1}^{N} \hat{a}_{k,i} K(< z, z_k >) \tag{35}$$

$$\tag{36}$$

so that the vector expansion of $f(x)$, in a kernel representation, is automatically reduced to a kernel representation of the vector function $\hat{f}^{(r)}(z)$.

4.2 Reduced Set Expansions

There is one drawback to the above procedure – the same number of support vectors, N, are required in both sums, and N could be large in the original space. To make the technique practical, an automatic way of compressing the kernel expansion is needed. Several techniques are possible, all falling under the heading of "reduced set" methods [12]. These methods attempt to construct new kernel expansions that represent the function described by the original kernel expansion, but with fewer support vectors. Full treatment of reduced set methods is beyond the scope of this paper; we will briefly describe one method based on the singular value decomposition (SVD). This technique illustrates one way in which familiar techniques from linear modeling can be efficiently applied in the nonlinear context.

Suppose the kernel matrix $K(z_i, z_j), i, j \in 1...N$ has rank $r < N$. This means that the columns of K are not independent; some of them can be expressed in terms of the others. If a subset of size r is selected (see [19] for a robust computational procedure), the coefficients of a reduced set expansion

$$\tilde{f}(z) = \sum_{k=1}^{r} \beta_k K(z, \tilde{z}_k) \tag{37}$$

may be found by minimizing the norm $||\tilde{f}(z) - \hat{f}(z)||$, which can be accomplished by computing [12]

$$\beta = K_2^{-1} K_1 \alpha, \tag{38}$$

where the entries of K_1, K_2 are given by $K_2(i,j) = K(\tilde{z}_i, \tilde{z}_j)$ and $K_1(i,j) = K(\tilde{z}_i, z_j)$.

It is generically plausible that some compression is possible, simply because the size of feature space of functions in lower dimensions is intrinsically smaller. However, as will be shown empirically in Section 6, dramatic reductions in complexity can be obtained by exposing structure that is specific to an individual problem, and not predictable based on generic arguments. In fact, the reduced set methods give us a precise tool to reason about such structure. For example, in [9], there is no formal way to choose the linearization points. How many are needed, and what is the error based on neglecting a given point? Likewise, in the polynomial methods, there is no clear way to select particular (i.e., sparse) combinations of polynomials that may dominate the reduced model structure. Reduced set methods, in the kernel context, can answer both questions, *with consideration to the nonlinear structure of the problem under investigation*. In [9], an SVD technique was used to compress the basis formed from a union of Krylov vectors. This exposes *linear* relations among components of the state space. By manipulating the nonlinear kernel, the above procedure can expose relations among components of the *feature space*, which are nonlinear transforms of the state-space components. It is thus posible to compress nonlinear functions such as $\hat{f}(z)$ in a rigorous way, i.e. with errors (measured in a feature-space norm) controlled by the singular values of the kernel matrix.

5. STATISTICAL TRAJECTORY SAMPLING

The second method to be considered we term trajectory-sampling. We adopt a similar strategy as used in [14, 15, 17] to select basis vectors, and [9] to pick samples. We choose a set of "test" inputs and take sample points from the evolution of the state equations of the original model under time-domain simulation. These points x_i

can be projected into the reduced space $z_i = V^T x_i$ to obtain sample points for $\hat{f}(z)$.

For efficient computation, it is important to choose as small a subset as possible, especially in least-squares fitting since all the sample points will be involved in the sum that defines the final function (other methods, in particular SVMs, may result in a sparser expansion). For purposes of generating the sample points themselves, we exploit the fact that often we have more simulation data than is needed to generate a good regressor. Therefore, we generate sample points by randomly selecting subsets from the union of stimulus trajectories. We fit each model, and if the fit is sufficiently accurate for all the subsets, proceed to the validation stage. As in the projection algorithm of Section 4, this procedure can also lead to a regressor with many support vectors. The reduced set regression procedure (Section 4.2) is thus also of interest for the trajectory-based algorithm.

We also must address the question of deciding whether or not the model obtained is accurate. To address these issues, we have adopted a two-stage sampling and cross-validation strategy. For each circuit to be modeled, we establish a set of inputs to be be used in stimulating the circuit, for purposes of generating data for the reduction and regression stage. We also form a set of inputs, of distinct form (different waveform shapes, amplitudes, frequencies) for use only as "test" signals to validate the regression/reduction.

6. COMPUTATIONAL EXPERIMENTS

The following set of experiments were performed to demonstrate various aspects of kernel methods in the nonlinear model reduction problem. In order to illustrate properties of the algorithm we have concentrated on the nonlinear delay line example [7, 9, 6] that has been a popular test case for reduction techniques. Unless otherwise noted, projection matrices V were obtained using an input-oriented principal components analysis[14, 15, 17] of simulation trajectories (this is analogous to projection onto the dominant controllable subspace in linear reduction). To choose the kernels and kernel parameters, we introduce an additional modeling stage. Kernel parameters are chosen by optimizing over a set of small, but distinct, circuits that contain similar nonlinear devices. We found this to be a tricky step, with poor choices leading to unacceptable results. We have found that careful attention to data scaling dramatically improves the accuracy of the procedures.

6.1 Piecewise Forms

Before beginning the reduction studies, in our first example, we compare the kernel-based regression schemes to function representations used in previous work on nonlinear model reduction. In order to best illustrate the differences in behavior between the various approximation schemes, we will first fit a simple, one-dimensional function composed of the difference of two shifted $\tanh(x)$ functions. Figure 2 shows the results of approximating the function using locally weighted linear approximations. Figure 2(a) uses the scheme of [9], exponentially weighted linear models obtained from the analytic derivative at each of the function sample points. Figure 2(b) uses penalized least-squares regression (Eq. 17), using a kernel constructed from products of Gaussian radial basis functions and linear polynomials (no derivative information was used in the fit), normalized as in (12).

The least-squares approximation is observed to be more accurate, as we would expect from using a more powerful fitting strategy, but also does not illustrate the large anomalies around changes in the function slope that are found in the locally-weighted scheme. This is due to two reasons. First is the more flexible, global, parameter adjustment, of least-squares. Local weighting was found

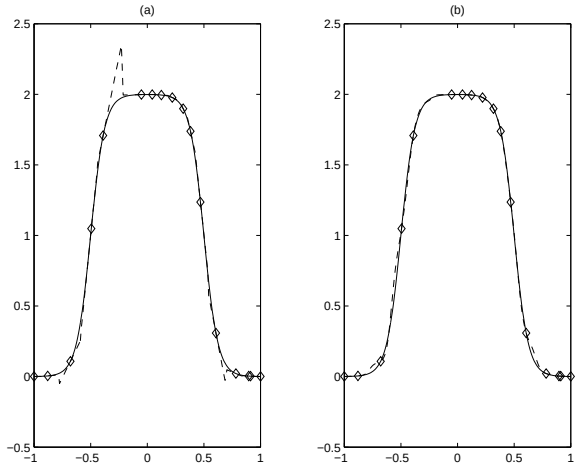

Figure 2: Piecewise-linear regressors. Solid line shows actual function, dashed line regressor, and diamonds sample points for regression. (a) Weighting-based regressor. (b) Regularized global-fit regressor.

to illustrate best overall behavior for relatively strong exponential decay, probably because it has no effective way to blend information from multiple samples in the function. Global least-squares, on the other hand, tends to choose parameters that allow it to utilize information from multiple sample points in constructing the regressor at intermediate points. Second, the global least squares strategy contains a regularization term that helps to control these "overfitting" effects. "Flatter", less complex functions are favored, all other things being equal. If the regularization term is removed, the results in Fig. 2(b) become somewhat more oscillatory.

6.2 Polynomial-Based Reduction

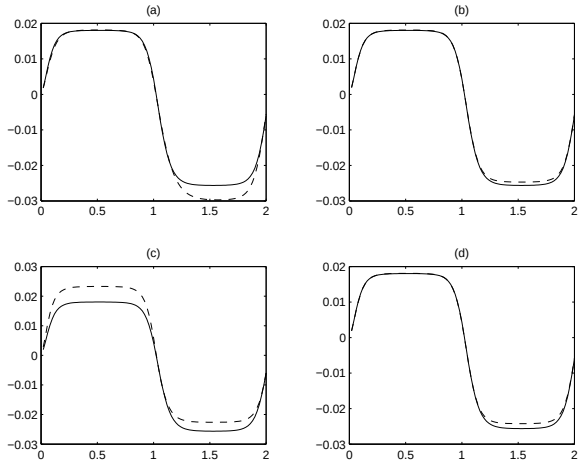

Figure 3: Polynomial-based model reduction using random function sampling. Solid line shows full model response, dashed line reduced model response. (a) $q = 2$. (b) $q = 3$. (c) $q = 4$, partial sampling. (d) $q = 4$, full sampling.

Next, using the nonlinear delay line example, of 30 starting states, we demonstrate implementation of polynomial-based model reduction using kernel techniques. Thirty random vectors in $\mathbb{R}^4$ were used for the sampling points to fit $\hat{f}(z)$. Figure 3 shows results of

a square-wave validation input applied to the reduced model obtained from nonlinear regression on the projected model of four states. From Figures 3(a) and 3(b), it is clear that the regression strategy effectively generated the polynomial models. Figure 3(c), a fourth-order model, however, demonstrates the difficulties with random sampling: the model is fairly representative of the original system, but it is actually less accurate than either of the lower order models. This is a result of under-sampling of the polynomial basis space. The space of polynomials up to fourth-order, in four dimensions, is dimension 70, and 30 random samples will not generally capture enough of the needed space. Using 80 independent random vectors, shown in Figure 3(d), recovers the expected behavior of a fourth-order model. These results were done with L_2 minimizing fits that produce a maximal set of support vectors; we found that using ν-SVMs [12] reduced the number of support vectors by typically 60%-70% on average.

6.3 Reduced-set selection from a trajectory

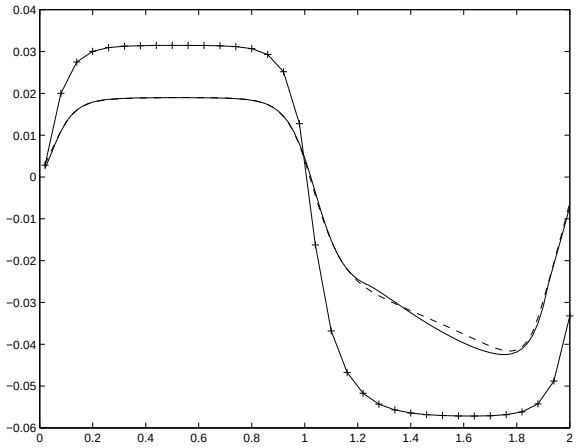

Figure 4: Seventh-order polynomial-based model reduction using sampling from simulation trajectories. Solid line shows full model response, dashed line shows reduced model response after reduced-set selection, solid-crossed line shows seventh order model response computed using 100 random samples (note large error).

In the next example, we consider obtaining the regressor by sampling from simulation trajectories. A set of sinusoidal inputs was applied to generate both the projection matrix and the sampling basis for regression. We used a slightly different configuration of the transmission line, driven harder, to exhibit more nonlinear behavior. To compensate, we took five states in the reduced model, and used a seventh-order polynomial kernel. The full feature space is potentially of dimension 792, and using the random vector technique on this example requires an excessively large number of samples to obtain good results. However, we were able to reliably obtain results such as shown Figure 4 using 80-100 samples from the stimulus trajectories, far too few to obtain accurate answers with random vector samples. This is still a large number of support vectors, particularly for a five-state model, so we applied the reduced-set technique of Section 4.2. By performing an SVD analysis on the kernel matrix (ala kernel-PCA [12]), and discarding singular values less in relative magnitude than 10^{-8}, we were able to reduce the number of support vectors to 17. No additional loss of accuracy relative to the 100-vector model was incurred. Thus, this example demonstrates the detailed analysis, and reduction, potential of the SVD and reduced-set techniques for nonlinear modeling. Explicit polynomial representations were not practical; the tensor-product forms have 80,000 entries in the reduced space, and potentially over 10^{10} entries were the full tensors in the dimension-30 space to be enumerated. More importantly, it illustrates emphatically that, at least for some circuits, reduction and approximation with some degree of statistical knowledge of behavior (or, as we shall see in the next example, structure) dramatically improves the efficiency of modeling. Using kernels, it is possible to efficiently encode, and manipulate, this structural, or statistical, knowledge in the kernel matrix. Note that random samples would not lead to a reduced set of support vectors, even if the full feature space of dimension 792 was sampled.

6.4 Kernel-Projection Example

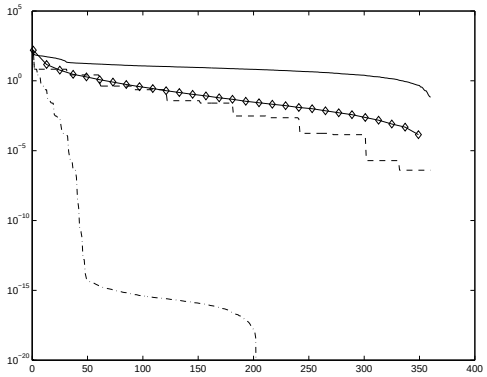

Figure 5: Singular values of kernel matrix in bottom-up projection example. Solid line: random vectors in original space. Diamonds: random vectors in reduced space. Dash line: kernel matrix for structured vectors in unreduced regression. Dash-dot line: kernel matrix for reduced support vectors.

Next we consider the bottom-up projection procedure of Section 4, again to form a five-state model of the delay line. To demonstrate use of a different kernel we used the sigmoid, $k(x,y) = \tanh(-\beta(< x,y >) + \theta)$, also often found in neural networks. The nonlinearities in the delay line have a particular form (two-terminal near-neighbor coupling) due to the connectivity of the delay line. Such structure is easily found in practical circuits. For the detailed model regression, we generated 12 support vectors per dimension, all multiples of the corresponding unit vector, spaced along the expected operation range (i.e. voltage swing) of the circuit, which can be shown sufficient to capture each of the detailed two-terminal nonlinearities. This generates 360 support vectors. By way of comparison, the space of 12th order polynomials in a thirty-state model has dimension over 10^{10}. Reduced-set selection was able to reduce the number of support vectors in the five-state projected model from 360 to 28 (a factor of 12.9X) and still maintain a model superior in accuracy to those previously shown. The singular value analysis upon which the support-vector reduction was based is shown in Figure 5.

From this figure it should be clear that the extreme reduction was *not* a property of the basis functions, was *not* due to the kernel alone, and it was *not* a result of the decreased size of the state space. The projected version of the structured regressor illustrates rapid decay of singular values, thus compressibility. Kernels using random support vectors, of similar scaling, do not illustrate this decay, either in the full or reduced state space. Likewise, the detailed model is not highly compressible, presumably because its potential

response is quite complex. Only when the behavior is restricted, such as in this case focusing on a state-space constructed by attention to a single input, with a kernel matrix that encodes structure of the nonlinearities, is the compressibility of the circuit exposed. *And kernel methods quantitatively expose the possibilities for compression.*

In summary, this example illustrates how kernel methods can exploit easily-obtainable information about nonlinear structure to improve the reduction process, and how, with such information, quantitative estimates for optimal model size, and achievable error, are obtainable.

6.5 Realistic Circuit

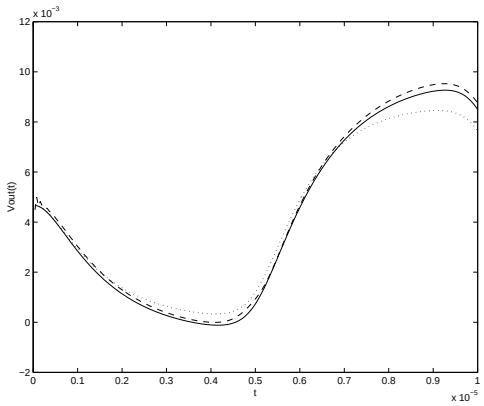

Figure 6: Response of UA741 amplifier. Solid line: full circuit. Dotted line: reduced model, unscaled regressors. Dashed line: reduced model, scaled regressors.

As a final example, we take the UA741 operational amplifier, configured as a small-gain non-inverting amplifier. Using sinusoidal stimulus inputs, we obtained reduced models based on polynomial and sigmoid kernel regression on the stimulus trajectories. Results are shown in Figure 6 for a pulse wave validation simulation. This example also illustrates that both a good choice of kernel parameters and careful data scaling is needed to obtain accurate results.

7. CONCLUSIONS

We have demonstrated how kernel methods (1) provide a convenient framework for more flexible approximation strategies in analog macromodeling, (2) are able to exploit statistical data, or structural information, to provide rigorous guidance about nonlinear reduction. These methods are not a panacea, however. With more complex functional representations comes the additional problem of parameter choice, increased sensitivity of the models to perturbations, and increased likelihood of generating models with non-physical artifacts. We believe that global stability and/or passivity of the nonlinear models is a particular concern.

8. REFERENCES

[1] Lawrence T. Pillage and Ronald A. Rohrer. Asymptotic Waveform Evaluation for Timing Analysis. *IEEE Transactions on Computer-Aided Design*, 9(4):352–366, April 1990.

[2] P. Feldmann and R. W. Freund. Efficient linear circuit analysis by Padé approximation via the Lanczos process. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 14:639–649, 1995.

[3] A. Odabasioglu, M. Celik, and L. T. Pileggi. PRIMA: passive reduced-order interconnect macromodeling algorithm. *IEEE Trans. Computer-Aided Design*, 17(8):645–654, August 1998.

[4] L. Miguel Silveira, Mattan Kamon, and Jacob K. White. Efficient reduced-order modeling of frequency-dependent coupling inductances associated with 3-d interconnect structures. In *Proceedings of the* 32^{nd} *Design Automation Conference*, pages 376–380, San Francisco, CA, June 1995.

[5] J. Roychowdhury. Reduced-order modeling of time-varying systems. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 46:1273–1288, 1999.

[6] Joel R. Phillips. Projection-based approaches for model reduction of weakly nonlinear, time-varying systems. *IEEE Trans. Computer-Aided Design*, 22:171–187, 2003.

[7] Y. Chen. Model order reduction for nonlinear systems. M.S. Thesis, Massachusetts Institute of Technology, September 1999.

[8] J. Chen and S. M. Kang. An algorithm for automatic model-order reduction of nonlinear MEMS devices. In *Proceedings of ISCAS 2000*, pages 445–448, 2000.

[9] M. Rewieński and J. White. A trajectory piecewise-linear approach to model order reduction and fast simulation of nonlinear circuits and micromachined devices. *IEEE Trans. Computer-Aided Design*, 22:155–170, 2003.

[10] N. Dong and J. Roychowdhury. Piecewise polynomial nonlinear model reduction. In 40^{th} *ACM/IEEE Design Automation Conference*, pages 484–489, Anaheim, CA, June 2003.

[11] J. R. Phillips. A statistical perspective on nonlinear model reduction. In *Proc. Behavioral Modeling and Simulation (BMAS) Workshop 2003*, October 2003.

[12] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA USA, December 2001.

[13] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Verlag, New York, 2001.

[14] E. S. Hung and S. D. Senturia. Generating efficient dynamical models for microelectromechanical systems from a few finite-element simulation runs. *IEEE J. Microelectromechanical systems*, 8:280–289, September 1999.

[15] S. Lall, J. E. Marsden, and S. Glavaski. A subspace approach to balanced truncation for model reduction of nonlinear control systems. *International Journal on Robust and Nonlinear Control*, 12(5):519–535, 2002.

[16] P. K. Gunupudi and M. S. Nakhla. Nonlinear circuit reduction of high-speed interconnect networks using congruent transformation techniques. *IEEE Trans. Advanced Packaging*, 24(3):317–325, 2001.

[17] J. Hahn and T.F. Edgar. Reduction of nonlinear models using balancing of empirical gramians and galerkin projections. In *Proceedings of the 2000 American Control Conference*, pages 2864–2868, Chicago, 2000.

[18] P. Li and L. T. Pileggi. NORM: Compact model order reduction of weakly nonlinear systems. In 40^{th} *ACM/IEEE Design Automation Conference*, pages 472–477, Anaheim, CA, June 2003.

[19] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, 1989.