Highly Accurate Fast Methods for Extraction and Sparsification of Substrate Coupling Based on Low-Rank Approximation

Joe Kanapka
Massachusetts Institute of Technology
Cambridge, MA 02139
kanapka@rle-vlsi.mit.edu

Jacob White
Massachusetts Institute of Technology
Cambridge, MA 02139
white@rle-vlsi.mit.edu

ABSTRACT

More aggressive design practices have created renewed interest in techniques for analyzing substrate coupling problems. Most previous work has focused primarily on faster techniques for extracting coupling resistances, but has offered little help for reducing the resulting resistance matrix, whose number of nonzero entries grows quadratically with the number of contacts. Wavelet-like methods have been applied to sparsifying the resistance matrix representing the substrate coupling, but the accuracy of the method is very sensitive to the particulars of the contact layout. In this paper we show that for the substrate problem it is possible to improve considerably on the wavelet-like methods by making use of the algorithmic structure common to the fast multipole and wavelet-like algorithms, but making judicious use of low-rank approximations. The approach, motivated by the hierarchical SVD algorithm, can achieve more than an order of magnitude better accuracy for commensurate sparsity, or can achieve much better sparsity at commensurate accuracy, when compared to the wavelet-like algorithm.

1. INTRODUCTION

Designers of analog blocks that are incorporated in mixed signal integrated circuits are making more extensive use of layout techniques to block signal interference from the substrate. In order to assess the effectiveness of these techniques, not only must extraction tools be developed to accurately compute substrate coupling[1, 2, 3, 4, 5], but the extracted model must be sufficiently efficient that it can be included in a circuit-level simulation of the analog block. The difficulty is that a complicated analog block might have more than 10,000 contacts to the substrate, and a naive representation of the substrate coupling conductance matrix would have more than 100 million entries.

It is possible to reduce the number of coupling conductances by "thresholding", that is by removing all the coupling conductances smaller than a fixed threshold. In a large substrate, one might expect that geometrically distant contacts would have small coupling, but this is often not the case. If the substrate is made of a thin top layer of relatively low conductivity and lower layers of higher conductivity, a situation desirable for latchup suppression, distant contacts can still have significant coupling. For these problems, another approach is needed to find an efficient representation of the coupling.

The problem of how to find a sparse representation of a dense matrix, like the coupling conductance matrix, has received considerable attention in the last decade. There are methods that exploit analytic properties of the matrix entries, like the fast multipole algorithms developed for the fast integral equation solvers[6], and methods that exploit a "near-convolutional" structure to the matrix, like the precorrected-FFT methods[7]. The conductance matrix has neither of those properties, because the matrix entries depend globally on the distribution of contacts. For this reason, general multiresolution, or wavelet-like, methods were applied to the substrate sparsification problem[8, 9].

In this paper we show that for the substrate problem it is possible to improve considerably on the wavelet-like methods by making use of the algorithmic structure common to the fast multipole and wavelet-like algorithms, but making judicious use of low rank approximations. The approach, motivated by the hierarchical SVD algorithm [10], can achieve more than an order of magnitude better accuracy for commensurate sparsity, or can achieve much better sparsity at commensurate accuracy.

Developing an efficient extraction algorithm based on low-rank approximation is significantly more difficult for the substrate coupling problem than for potential-from-charge problems. The added difficulty arises in the substrate coupling problem because one must rely on substrate analysis algorithms (i.e. finite-difference or fast integral equation methods) that can compute contact currents due to an entire set of contact voltages, but cannot efficiently compute the individual sensitivity of a single contact current to a single contact voltage.

In the next section we begin with a motivating example, and then in Section 3 we present the low-rank algorithm. Results on several examples are presented in Section 5 and conclusions are given in Section 6.

2. SOME INTUITION

In our setting, the substrate model is purely resistive (with layers of different conductivities), and the voltage (v) to current (i) relation is therefore linear, denoted by $Gv = i$ where G is called the *conductance matrix*. The goal is to obtain an efficient representation of G; the tool we have is a solver which, by discretizing the layout geometry, gives Gv for any particular v.

We begin by trying to give some intuition for why an approach,

0-7803-7247-6/01/$10.00 © 2001 IEEE

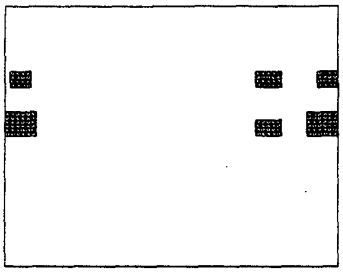

Figure 1: Simple example contact layout

such as ours, which uses information from applying G to form the new basis for the voltages and currents, may be more effective than one which simply relies on geometric information. We would like to find a basis function, nonzero on the two contacts on the left side in Figure 1, such that the current response at the four right-hand side contacts is very small. Geometric moment-matching techniques (the technique of [11, 8, 9]), applied in their simplest (order 0) form, require choosing the two voltages such that their integral over the 2 left-side contacts is 0. Since the area of the larger contact (call it contact 2) is 2.25 the area of the smaller contact (contact 1), this suggests $v = (-2.25\ 1)'$ (-2.25 volts on contact 1, 1 volt on contact 2) as a basis function.

Unfortunately, the basis function's response at the left-side four contacts is not very small. We computed G using the our solver for the contact layout of Figure 1 and for a two-layer substrate (thin high-resistivity layer on top of a low-resistivity layer). Forming the interaction matrix G_p giving current responses at contacts 3 through 6 from voltages at 1 and 2, we get (for this particular example)

$$G_p v = \begin{pmatrix} 5.80 & 11.06 \\ 4.25 & 8.12 \\ 8.74 & 16.82 \\ 5.77 & 11.12 \end{pmatrix} \begin{pmatrix} 2.25 \\ -1 \end{pmatrix} = \begin{pmatrix} 1.99 \\ 1.45 \\ 2.84 \\ 1.87 \end{pmatrix}.$$

However, looking at G_p we observe that the second column is very close to a multiple (1.92) of the first column—that is, the current responses at 3,4,5, and 6 to unit voltage on contact 2 are very close to 1.92 times the current response at 3,4,5, and 6 to unit voltage on contact 1. This suggests using $(-1.92\ 1)'$ as our basis function. In fact, doing this gives $G_p v = (-.08\ -.04\ .04\ .04)'$, a much smaller faraway response.

So, by using information obtained from applying G, we were able to find a much better basis function, one whose faraway response is close to zero and can be zeroed out without too much loss of accuracy. Our algorithm is a generalization of this idea: instead of just using information from geometry, use information from carefully chosen applications of G to form the new basis.

3. THE ALGORITHM

3.1 Overview

The algorithm, like others which have been proposed for this and similar problems [9, 10] is *multilevel*, and we now define this more precisely. The number of contacts is denoted by n. For simplicity assume the substrate surface is a unit square. The levels start from 0, and on level i there is a grid of $2^i \times 2^i$ squares, each with side-length 2^{-i}. The finest level $maxlev$ is chosen so that there are at most a small constant number of contacts per square. We denote by $\#(s)$ the number of contacts in a particular square s. The *parent* of a square s on level $l > 0$ is the square p on level $l-1$ containing s, and s is a *child* of p. In general, on a given level, the algorithm will deal with interactions between squares which are not too far apart (this will be defined more precisely later).

The algorithm is divided into two phases. First, a *multilevel row basis* representation is obtained by a process which ascends through the levels from coarsest to finest. The result is a representation of the coupling operator which is approximately $O(n \log n)$ in both storage cost and cost of applying the operator to a vector—that is, getting currents on all n contacts from voltages on all n contacts. This part of the algorithm gives, by itself, a very efficient and accurate way to apply G.

In order to further improve performance, in the second phase, we use the multilevel row vector representation obtained in the first phase to create a *transformed basis* Q for the voltages and currents, in which the coupling interaction is numerically sparse: that is, QGQ' is numerically sparse. Furthermore, we know enough *a priori* about the sparsity structure that only approximately $O(n \log n)$ entries of QGQ' need to be computed at all. This phase descends through the levels from finest to coarsest.

We describe more precisely the goals of each phase and how they are achieved in the next sections.

3.2 Coarse-to-fine sweep

The goal of the first phase is to form a *multilevel row basis* representation of G. More precisely, consider an interaction matrix G_{sI_s} which is applied to voltages in square s and gives currents in the *interactive squares* I_s of square s. In general, following Greengard [12], we use the notation G_{ab} to mean the operator which takes a length $\#(a)$ vector v of the region-a voltages and returns a length $\#(b)$ vector i of the region-b currents resulting from putting the voltages in v on the contacts in region a and zero voltage on all other contacts. That is, $G_{ab}v = i$ and G_{ab} has $\#(a)$ columns and $\#(b)$ rows.

The interactive squares of a level-l square s are the squares on level l which are separated from s by at least one square but whose parent squares are neighbors (adjacent or have a common corner). The *local* squares L_s of a level-l square s are s itself and its neighbors on level l. See Figure 2.

The important point is that the interaction matrix is numerically very low-rank. Specifically, there is a small number (denoted c) of rows, each of which is a linear combination of rows of G_{sI_s}, such that these rows form a basis (to a close approximation) for the row space of G_{sI_s}. (In our examples, we've found that choosing $c = 6$ rows is a close enough approximation for very good accuracy.) Write down these few rows in a matrix V_s' (c rows, $\#(s)$ columns). Any vector v of voltages in square s which is orthogonal to V_s' (i.e. $V_s'v = 0$) will have $G_{sI_s}v \approx 0$, because all the rows of G_{sI_s} are approximate linear combinations of rows of V_s'. Thus, we can get an accurate representation of G_{sI_s} by projecting v onto the rows of V_s' and knowing the responses to the c voltage vectors given by rows of V_s'—in matrix notation the responses are $G_{sI_s}V_s$. If we construct V_s' so that its rows are orthonormal, we get a compact representation of our approximation:

$$G_{sI_s} \approx (G_{sI_s}V_s)V_s'.$$

Note that using this representation gives a dramatic efficiency improvement over using the dense G_{sI_s} if the number of contacts in s is large: computing $G_{sI_s}v$ directly requires $\#(s) \cdot \#(I_s)$ multiply-add operations, whereas applying $(G_{sI_s}V_s)V_s'$ requires only $2c\#(I_s)$ multiply-adds. (We will refine this idea to improve

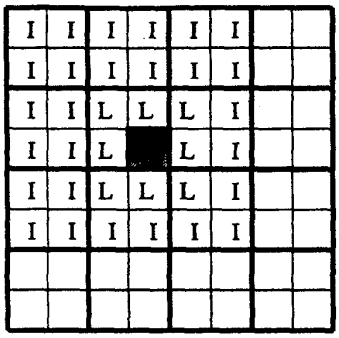

Figure 2: Interactive (labeled I) and local (labeled L) squares of shaded square: next-coarser level squares shown with bold lines

accuracy later.) So the goal is to form the *row basis* representation of G_{sI_s} for every square and its interactive squares at every level, consisting of V_s' (row basis) and $G_{sI_s}V_s$ (responses to row basis).

Coarsest level

For this phase, the coarsest level (which is the one we start with), is chosen to be $l = 2$. (The reason is that using our definition, for squares on levels less than 2, there are no interactive squares.) There are 16 squares on this level. We fix a square s, and show how to form the row basis V_s' for G_{sI_s}. One approach is to take a truncated singular value decomposition (taking only the largest c singular values, or taking only singular values bigger than a threshold ϵ)

$$G_{sI_s} \approx U\Sigma V'$$

where the number of columns of U equals the number of rows of V equals the number of singular values kept. Then set $V_s' = V'$. Now by writing $G_{sI_s} \approx (U\Sigma)V_s'$, it's clear that V_s' is an approximate row basis for G_{sI_s}.

The problem with this approach is that it requires obtaining the whole dense G_{sI_s}, which would require a number of black-box calls equal to the number of contacts in the square, for every square. We want to keep the number of black-box calls proportional to $\log n$. In order to achieve this, we use a version of *sampling*, similar to [10]. The idea is to get a few sample rows of G_{sI_s} and get the row basis from the svd of the sample rows (actually, for us a sample row is a linear combination of rows of G_{sI_s}). Specifically, if we pick a small number (say $2c$) of column vectors, each with one entry for each contact in I_s, and put them in a matrix of sample vectors S_s, the matrix of sample rows is $S_s'G_{sI_s}$. Notice that by symmetry of G and transpose properties,

$$S_s'G_{sI_s} = (G_{I_s s}S_s)'.$$

Thus it suffices to obtain $G_{I_s s}S_s$, which can be done with $2c$ black-box solver calls (one for each column of S_s). In fact, sample vectors can be shared among different squares on a given level, by choosing each sample vector to have support in exactly one square. Then for a square s, the sample vectors used are those with support in the interactive squares of s. In our implementation, 1 or 2 sample vectors per level per square works well.

Then we take the truncated svd $S_s'G_{sI_s} = U\Sigma V'$ and set $V_s' = V'$. V_s' is our row basis, and with $\leq c$ additional black-box calls, we obtain $G_{sI_s}V_s$. (In fact, we actually have the responses to V_s everywhere, and in particular at the interactive and local squares together. We denote the interactive and local squares together by P_s. We have $G_{sP_s}V_s$, which will be important on the finer levels.)

Finer levels

The goal for the finer levels is the same: to obtain a small row basis V_s for each square s to represent the interaction of square s with the interactive squares I_s of s. (Notice that even as the number of squares multiplies on the finer levels, the number of interactive squares per square is bounded by a small constant—this is part of the reason for the low-rank algorithm's efficiency in deriving the representation.) The same approach of choosing 1 or 2 sample vectors per square and sharing these among the source squares to form G_{sI_s} for every s that was described for the coarsest level is used here.

However, the difference from the coarsest level is in how the sample vector and row basis responses are calculated in each square. If this were done in the obvious way, by calling the black-box once per sample vector and row-basis vector in each square, the resulting algorithm would be very inefficient, because the number of squares on the finest level is $O(n)$, so $O(n)$ solves would be needed.

To reduce the number of solves, we combine solves as described in [9]. For a square s on level l, a vector v_s (length $\#(s)$) of the voltages in that square can be expressed as a sum of two vectors in the parent square p as follows. First extend v_s to a vector v of voltages in p (length $\#(p)$) in the natural way, that is by copying the voltages in v_s to the entries corresponding to square s in v and putting zeros in the entries corresponding to the other three children of p. Then:

$$v = V_pV_p'v + (I - V_pV_p')v \tag{1}$$

Now

$$G_{pP_s}v = (G_{pP_s}V_p)V_p'v + G_{pP_s}(I - V_pV_p')v.$$

The reason we need to show how to obtain the current responses in P_s, which contains the local squares as well as the interactive squares of s, is that the first term on the right relies on having $G_{pP_s}V_p$ from the parent level, which we do, since P_s is contained in P_p. I_s is not contained in I_p, so the algorithm wouldn't work if we substituted I_s for P_s above.

The first term on the right is computed from the next-coarser level $(l-1)$ row-basis representation. The second term is G_{pP_s} applied to $(I - V_pV_p')v$, and $(I - V_pV_p')v$ is orthogonal to the level $l-1$ row-basis in square p. Thus we expect that $G(I - V_pV_p')v$ will be nearly 0 outside local squares of p. As described in [9], it is possible to combine black-box applications of G for many vectors into one solve under these circumstances, and to keep the number of solves per level constant. In this way we compute the second term.

Adding the two terms we obtain the responses to sample vectors with support in square s at the interactive squares $I(s)$. We use the same technique to obtain responses to the row basis obtained on level l at the interactive and local squares $P(s)$.

Finest level

At this point, we have represented all of G corresponding to squares which are interactive (this is a symmetric relation) at some level. The only part of G that remains to be computed is the interactions of local squares on the finest level. In fact, for each finest-level square s we already have the local responses to the row-basis vectors V_s. Denote by U_s a matrix the columns of which form a basis for the vectors in s which are orthogonal to V_s. (If V_s has a columns then U_s has $\#(s) - a$ columns.) The combine-solves technique is used to obtain local responses to the U_s. Since any vector v with support in square s can be written as a sum of a vec-

tor in V_s and a vector orthogonal to V_s (that is, in U_s), this gives the complete local interaction for the finest level.

3.3 Applying the operator

On each level, for each square s, we now have a low-rank representation of the current response in the interactive squares due to voltages in s given by $(G_{sI_s}V_s)V_s'$. It can be applied quickly (linear in the number of contacts in I_s). However, we gain a substantial improvement in accuracy by refining this technique. For the refinement, we consider the interaction between two interactive squares on a given level, denoted by G_{sd} where s is the voltage source square and d is the square where the current is being measured.

Since G_{sd} just restricts the contacts at which current is measured, compared to G_{sI_s}, we know that because $G_{sI_s} \approx (G_{sI_s}V_s)V_s'$, it's also true that $G_{sd} \approx (G_{sd}V_s)V_s'$. (We have $G_{sd}V_s$ because it's just a subset of the rows of $G_{sI_s}V_s$.)

We claim that

$$G_{sd} \approx V_d V_d' G_{sd}$$

as well, and we use this to refine the approximation. Note that we have V_d since we have a row basis for every square at every level. To justify the claim, observe that V_d is an approximate row basis for $G_{ds} = G_{sd}'$ by symmetry, and thus it's an approximate column basis of G_{sd}. So, writing

$$V_d V_d' G_{sd} v = (V_d V_d')(G_{sd} v),$$

we see that $V_d V_d' G_{sd}$ is the projection of the exact $G_{sd}v$ onto the (approximate) column space of G_{sd}. But $G_{sd}v$ is a linear combination of the columns of G_{sd} anyway, so the projection is accurate to the extent that the columns of V_d span the columns of G_{sd}. The approximation for G_{sd} is obtained as follows:

$$\begin{aligned} G_{sd} &= G_{sd}V_sV_s' + G_{sd}(I - V_sV_s') \\ &\approx G_{sd}V_sV_s' + V_dV_d'G_{sd}(I - V_sV_s') \\ &= (G_{sd}V_s)V_s' + V_d(G_{ds}V_d)'(I - V_sV_s') \end{aligned}$$

A similar observation applies to the decomposition step described in the previous subsection (Equation 1). This is analogous to the improvement in accuracy gained in wavelet techniques [9] by not dropping interactions between fast-decaying wavelet basis voltage functions and standard basis current basis functions, but instead dropping only wavelet-wavelet interactions.)

If we sum up the interactions between each square and its interactive squares on every level, this covers everything but the local interactions at the finest level, which we have a complete representation for (discussed at the end of Section 3.2.)

3.4 Fine-to-coarse sweep

In this part of the algorithm, the goal is to use the row-basis representation just obtained to obtain a representation which is wavelet-like in structure [11, 9]. That is, we obtain a sparse orthogonal change-of-basis matrix and a sparse approximate G_w which gives the conductance matrix in the new basis,

$$G = QG_wQ'.$$

This is a simpler representation to work with and has the advantage that further sparsity in G_w can be obtained by thresholding out small entries, trading off the better sparsity for decreased accuracy. It also makes comparisons to previous work [9] possible. Because we have the row basis representation to work with, no further calls to the black-box solver are needed in this phase.

On each level, starting at the finest, we construct *fast-decaying* and *slow-decaying* basis functions in each square. Denote by T_s and W_s the matrices whose columns are the fast- and slow-decaying basis functions respectively. That is, the current response to the fast-decaying basis functions in a square s should be close to 0 outside the local squares of s, and the whole basis (columns of $[T_s W_s]$) will be orthogonal.

The finest level is very easy: for a square s on the finest level, W_s consists of the row basis for s; i.e. $T_s = V_s$. The columns of T_s form a basis for the orthogonal space of W_s, i.e. $T_s = U_s$ in the notation of the "finest level" discussion of Section 3.2.

Coarser levels

For each parent square p on level l, the idea is to recombine slow-decaying basis functions from the level-$l+1$ children of p to form many fast-decaying and some slow-decaying basis functions in p. This is done using the singular value decomposition. Let X_p be the matrix whose columns are the columns of $w_{s_1}, w_{s_2}, w_{s_3}$, and w_{s_4} for each of the four children s_1, s_2, s_3, s_4 of p. Take the svd of $G_{pI_p}X_p$, and set W_p and T_p to sections of V as shown:

$$G_{pI_p}X_p = U\Sigma V' = U \begin{pmatrix} \Sigma_{\text{large}} & 0 \\ 0 & \Sigma_{\text{small}} \end{pmatrix} \begin{pmatrix} W_p' \\ T_p' \end{pmatrix}.$$

Choose the number of columns of W_p equal to the number of singular values in Σ_{large} (i.e., the number of large singular values according to whatever threshold we are using). Notice that if we multiply both sides by T_p, we obtain

$$G_{pI_p}(X_pT_p) = U(:, 1 : |\Sigma_{\text{small}}|)\Sigma_{\text{small}}T_p.$$

The right-hand side is close to zero (assuming the Σ_{small} are in fact very small), suggesting that the columns of X_pT_p are a very good "fast-decaying" basis on the parent level l.

We proceed through the levels, transforming the slow-decaying basis functions on a level into fast-decaying basis functions on the next-coarser level. At the end, the only slow-decaying basis functions left are at the coarsest level. Eventually all the basis functions together form our new orthogonal change-of-basis matrix Q.

We briefly sketch how the entries of G_w can be computed efficiently, given the new basis Q and the multilevel row-basis representation. The only interactions which need to be kept are those between fast-decaying (T_s) basis functions in squares which are local to each other. (For two basis functions on different levels, we take the conservative approach of defining "local" to mean that the finer-level square's ancestor on the coarser level is the same as or a neighbor of the coarser-level square. In fact, many of these interactions are very small, and are zeroed when a small threshold is applied). We also keep the top-level slow-decaying basis-function interactions with everything else. There are at most a small constant number of these.

The essential idea is to keep a data structure for each level containing the local responses to the T_s and W_s basis vectors in each square at that level. We have this already for the finest level, from the Phase 1 representation. On coarser levels, the interaction between a parent square p and its neighbors can be decomposed into the four interactions between each of its children $s_1 \ldots s_4$ and the neighbors of p. For each child s_i, this interaction can be decomposed into the interaction with squares local to that child (on the child level), and the interaction with interactive squares of that child (on the child level). (This is a consequence of the fact that the local squares on the parent level are the same as the interactive squares plus the local squares on the child level, as can be seen in Figure 2.) The local interactions we get from the data structure maintained on the child level, and the interactive square interactions can be obtained using the row bases of s_i and the interactive squares of s_i.

(Why not just use the Phase 1 row-basis representation directly? It can be shown for reasonably regular contact layouts that this leads to an $O(n \log^2 n)$ algorithm, whereas the approach just described leads to an $O(n \log n)$ algorithm.)

3.5 Algorithm summary

Phase 1: get multilevel row-basis representation
for lev:=2 **to** maxlev
  for each square s on level lev
    choose a random sample vector m_s (nonzero only in s)
  end
  get responses to sample vectors:
  if lev == 2
    for each square s on level lev
      Get response Gm_s to sample vector
      using black-box solver
    end
  else *get response to sample vectors using splitting method:*
    for each square s on level lev
      Decompose $m_s = r_s + o_s$ (m_s in span of
      parent level row-basis,
      o_s orthogonal to parent level row-basis)
    end
    Use combine-solves technique to get local and interactive (=local on parent level) responses $G_{sP_s}o_s$ to all o_s *with $O(1)$ black-box calls*
    Use parent-level row-basis responses to get local and interactive responses $G_{sP_s}r_s$ to all r_s with $O(n)$ work.
    for each square s on level lev
      get $G_{sP_s}m_s = G_{sP_s}o_s + G_{sP_s}r_s$
    end
  end

  Get row basis in each square s on level lev using sample vectors and svd

  Get responses to row basis: same method as getting responses to sample vectors
end
for each square s on finest level (lev=maxlev)
  get local responses to every standard basis function in s using combine-solves technique for orthogonal space of row basis V_s
end

Phase 2: get wavelet-structure basis Q
for lev:=maxlev **downto** 2
  Form T_s (fast decaying response), W_s (slow decaying response) for each square at level lev:
  if lev == maxlev
    for each square s at level maxlev
      set $W_s = V_s$, set T_s orthogonal to V_s
    end
  else
    for each square s at level lev
      use svd to get T_s, W_s from
      child $W_{s_1} \ldots W_{s_4}$
    end
  end
  For each square s on level lev,
  put vectors in T_s into Q (wavelet-structure basis)
end
Fill in G_w: form interactions of fast-decaying basis functions with each other and with coarsest-level slow-decaying basis functions

Example	Sparsity factor (low rank)	Sparsity factor (wavelets)	Max. rel. error (low rank)	Max. rel. error (wavelets)
1	2.8	2.5	1.0%	69%
1a	1.6	1.3	0.3%	18%
1b	6.8	6.1	1.9%	105%
2	2.7	2.5	0.7%	1.3%
3	2.4	2.3	0.9%	30%

Table 1: Accuracy achieved without thresholding

4. COMPUTATIONAL RESULTS

In presenting our results, the key point is that arbitrary sparsity can always be obtained by reducing accuracy—i.e., set a threshold t and drop entries of G with absolute value below t. Or, in our representation $G \approx QG_wQ'$, drop entries of G_w below t. By setting t large enough, any desired sparsity is obtained. Of course, when t is set very large the sparsified representation will be of very poor quality. Thus, to address algorithm performance, sparsity and accuracy must be considered together.

We will compare our results using the new low-rank algorithm to the geometric-moments wavelet algorithm presented in [9]. It is worth noting that both these algorithms obtain substantial sparsification on many examples and are clearly better than naive approaches such as thresholding out small entries in the original dense G. However, we believe the results here show a much better sparsity-accuracy tradeoff for the low-rank algorithm. First we mention that the number of solves needed for our method is slightly larger than for the wavelet method (essentially because we're doing extra solves for the sample vectors), but in none of our examples more than 20% larger. This is not a major issue for many applications, where the representation will be derived once and applied many times in a circuit simulator. The important point is that the operator derivation takes only a very efficient $O(\log n)$ solves for both methods.

To make the comparison, we will consider two ways of using the sparsification algorithms. First, we can apply the conservative assumptions described earlier to obtain some sparsity, but try to maintain high accuracy. In this case we measure the maximum *relative* error in all the n^2 entries of the dense matrix QG_tQ' (for medium sized examples where it is tractable to compute this). (Q is the orthogonal basis and G_t the sparsified operator, for whichever algorithm, geometric moment-matching or low-rank approximation, is being used.) Notice that this is a particularly harsh standard for small entries in G where small relative errors will require extremely small absolute errors, but the low-rank algorithm still delivers very good results. Table 1 summarizes the results. The *sparsity factor* of a sparse matrix is the ratio of the number of entries in a dense matrix of the same size to the number of nonzeros in the sparse matrix.

On the other hand, for many applications a less stringent standard is appropriate, and we can obtain a more efficient representation by thresholding out small entries of G_t. To get a handle on this we'll use as our measure of accuracy the percentage of entries in QG_tQ' which deviate by more than 10 percent from the corresponding entry in G. A threshold will be chosen which gives

Example	Sparsity of thresholded G_t (low rank rep.)	Entries Off by more than 10%	Wavelet sparsity (equiv. accuracy)	Wavelet QG_wQ' Entries Off by More than 10% (equiv. sparsity)
1	17.0	1%	2.5 (*)	85%
1a	9.6	2%	1.3 (*)	86%
1b	41.0	2%	6.1 (*)	82%
2	16.2	3%	8.5	20%
3	14.3	8%	5.5	92%

Table 2: Sparsity/accuracy tradeoff for low rank vs. wavelet representation: the (*) indicates that even with no thresholding the wavelet method didn't achieve the same accuracy as the low-rank method

the low-rank algorithm a 6x increase in sparsity factor from using it without thresholding. Then the wavelet algorithm will be compared to the low-rank algorithm in two ways: first, choose a threshold to obtain equivalent accuracy to the low-rank algorithm, and compare the sparsity achieved. Second, choose a threshold to obtain equivalent sparsity to the low-rank algorithm, and compare the accuracy of the two methods. Table 2 summarizes the results.

The new low-rank method is the clear winner in all the examples. In each case the maximum relative error for the very accurate (without thresholding) representations is worse (sometimes much worse) for the wavelet method. In the more efficient lower-accuracy representations, the low-rank method achieves more accuracy at the same sparsity as the wavelet-method, and more sparsity at the same accuracy as the wavelet method.

Example 1 (see Figure 3) represents a bad case for the wavelet method due to the proximity of large and small contacts throughout; this is where the greatest advantage is seen for the new method. For this example we also show a sparsity structure plot of G_t; they are similar in appearance for the other examples. Examples 1a and 1b a are the same except the grid of contacts is smaller (16 by 16=256 contacts) for Ex. 1a and larger (4096 contacts) for Ex. 1b—they are included mainly to show that the sparsity factor increases as we increase the problem size (9.6 for 256 contacts, 17 for 1024 contacts, 41 for 4096 contacts), suggesting the algorithm is much closer to $O(n)$ than $O(n^2)$, which is how it was designed. (Also, due to the extremely large size of the dense 4096 x 4096 matrix, the results for example 1b are based on a 10 percent sample of the columns of G.) Example 2 (see Figure 5) is a regular grid of contacts, and a good case for the wavelet algorithm (the fact that all the contacts are the same size makes the moment-matching representations more effective), but the low-rank algorithm still wins. Example 3 (see Figure 6) illustrates that our method can handle quite irregular layouts, with different sizes and shapes of contacts, including such features as guard rings. They are each about 1000 contacts in size.

5. CONCLUSIONS AND FUTURE WORK

We have shown a new method for efficiently extracting a sparse representation of the conductance matrix of substrate coupling. The results show very high accuracy for the low-rank algorithm compared to previously known methods, and also demonstrate a better sparsity/accuracy tradeoff.

Future work may include putting this model in a circuit simulator such as SPICE to try to simulate the substrate effectively. There are some improvements to the algorithm which are possible but not yet

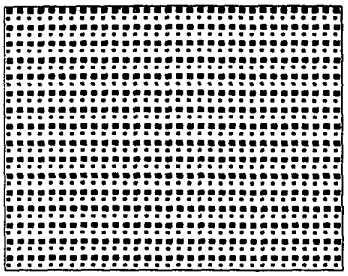

Figure 3: Example 1, alternating rows of large and small contacts (1024 contacts)— Example 1a is the same but with only 16x16=256 contacts

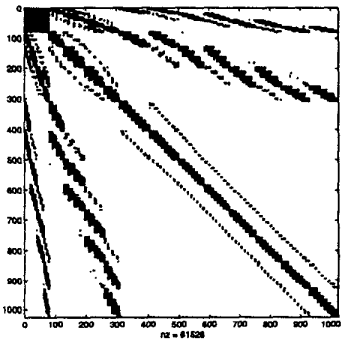

Figure 4: Example 1 sparsity structure plot for low-rank alg. G_w

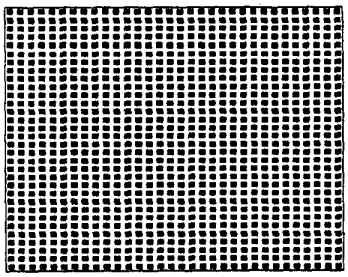

Figure 5: Example 2, regular grid (1024 contacts)

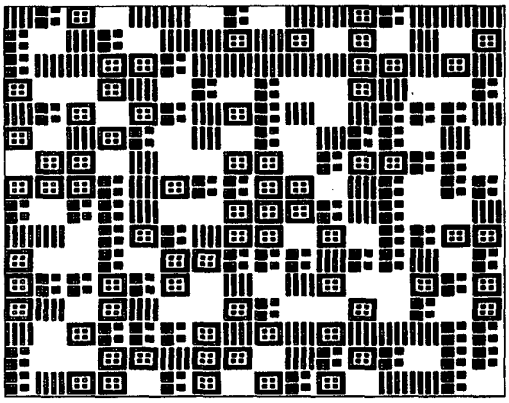

Figure 6: Example 3, with guardrings, long contacts, different-sized contacts (836 contacts)

implemented (for example, we believe the coarsest level can be set at level 1 instead of the current level 2). We would like to use real substrate layouts to test the algorithm further. Some form of automatic error estimation (outputting an error estimate with the new representation) would be desirable. It may also be interesting to see if any reasonable accuracy can be achieved with the new algorithm by reducing the rank-6 assumption for faraway squares even further, which would result in a very sparse representation.

6. ACKNOWLEDGMENTS

The authors would like to thank Joel Phillips for helpful discussions. This work was sponsored by the MARCO semiconductor industry consortium, and grants from Synopsys, Compaq and Intel. The first author was also supported by an IBM fellowship.

7. REFERENCES

[1] D. K. Su, M. Loinaz, S. Masui, and B. Wooley, "Experimental results and modeling techniques for substrate noise in mixed-signal integrated circuits," *IEEE Journal Solid-State Circuits*, vol. 28, no. 4, pp. 420–430, April 1993.

[2] F. J. R. Clement, E. Zysman, M. Kayal, and M. Declercq, "LAYIN: Toward a global solution for parasitic coupling modeling and visualization," in *Proc. IEEE Custom Integrated Circuit Conference*, May 1994, pp. 537–540.

[3] N. K. Verghese, T. Schmerbeck, and D. J. Allstot, *Simulation Techniques and Solutions for Mixed-Signal Coupling in ICs*, Kluwer Academic Publ., Boston, MA, 1995.

[4] T. Smedes, N. P. van der Meijs, and A. J. van Genderen, "Extraction of circuit models for substrate cross-talk," in *Proc. IEEE International Conference on Computer Aided Design*, November 1995, pp. 199–206.

[5] R. Gharpurey and R. Meyer, "Modeling and analysis of substrate coupling in integrated circuits," *IEEE Journal Solid-State Circuits*, vol. 31, no. 3, pp. 344–353, March 1996.

[6] K. Nabors, S. Kim, and J. White, "Fast capacitance extraction of general three-dimensional structure," *IEEE Trans. on Microwave Theory and Techniques*, vol. 40, no. 7, pp. 1496–1507, July 1992.

[7] J. R. Phillips and J. K. White, "A precorrected-FFT method for electrostatic analysis of complicated 3D structures," *IEEE Trans. CAD*, pp. 1059–1072, 1997.

[8] J. Tausch and J. White, "A multiscale method for fast capacitance extraction," in *Proceedings of the 36th Design Automation Conference*, New Orleans, LA, 1999, pp. 537–542.

[9] J. Phillips J. Kanapka and J. White, "Fast methods for extraction and sparsification of substrate coupling," in *Proc. of the 37th Design Automation Conference*, 2000, pp. 738–743.

[10] S. Kapur and D. Long, "IES^3: a fast integral equation solver for efficient 3-D extraction," in *Proc. IEEE International Conference on Computer Aided Design*, November 1997, pp. 448–455.

[11] B. Alpert, G. Beylkin, R. Coifman, and V. Rokhlin, "Wavelet-like bases for the fast solution of second-kind integral equations," *SIAM J. Sci. Comput.*, vol. 14, no. 1, pp. 159–184, 1993.

[12] L. Greengard, *The rapid evaluation of potential fields in particle systems*, MIT Press, Cambridge, 1988.