# Fast Methods for Extraction and Sparsification of Substrate Coupling

by

Joseph Daniel Kanapka

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

Doctor of Philosophy in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2002

Author ..........................................................
Department of Electrical Engineering and Computer Science
May 24, 2002

Certified by......................................................
Jacob White
Professor
Thesis Supervisor

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

# Fast Methods for Extraction and Sparsification of Substrate Coupling

by

Joseph Daniel Kanapka

Submitted to the Department of Electrical Engineering and Computer Science
on May 24, 2002, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Electrical Engineering and Computer Science

## Abstract

Substrate coupling effects have had an increasing impact on circuit performance in recent years. As a result, there is strong demand for substrate simulation tools. Past work has concentrated on fast substrate solvers that are applied once per contact to get the dense conductance matrix $G$. We develop a method of using any underlying substrate solver a near-constant number of times to obtain a sparse approximate representation $G \approx QG_{wt}Q'$ in a new basis. This method differs from previous matrix sparsification techniques in that it requires only a "black box" which can apply $G$ quickly; it doesn't need an analytical representation of the underlying kernel or access to individual entries of $G$. The change-of-basis matrix $Q$ is also sparse. For our largest example, with 10240 contacts, we obtained a $G_{wt}$ with 130 times fewer nonzeros than the dense $G$ (and $Q$ more than twice as sparse as $G_{wt}$), with 20 times fewer solves than the naive method, and fewer than 4 percent of the $QG_{wt}Q'$ entries had relative error more than 10% compared to the exact $G$.

Thesis Supervisor: Jacob White
Title: Professor

## Acknowledgments

The work of this thesis was a substantial undertaking for me and would not have been possible without the help and support of many people. My adviser, Jacob White, was an excellent source of research ideas, sounding board, and mentor. I credit his encouragement and faith in me when I started in the group, at a difficult time in my research career at MIT, for a large part of whatever success I have had.

Joel Phillips was, to a large extent, a second adviser on this project. Without forgetting to mention Joel's technical contributions, I would like to thank him for helpful discussions about this work in particular, and for being a mentor more generally. Mike Chou's thesis on substrate coupling extraction was both an inspiration for this work and one of its practical underpinnings, in the form of his QuickSub extraction code. His help, in allowing the use of his code as well as several pictures from his thesis, has been invaluable. Johannes Tausch was very helpful to me in the short time we were together in the group, and his work underlies our approach to substrate coupling sparsification. I thank Wolfgang Hackbusch for a helpful discussion. I would also like to thank my committee members, Alan Edelman and Terry Orlando, for their helpful comments and general forbearance. Also to be commended for putting up with me in the final months of the process are my roommate Amir Sufi and my officemate John Rockway.

Without financial support, I could not have completed this work. I would like to thank IBM for two years of fellowship support. I would like to thank Bhavna Agrawal and Khalid Rahmat from the EDA group for helpful discussions and for giving me the opportunity to give a seminar talk. In addition to the IBM support, I would like to acknowledge my other sources of support during the time I worked on this project: at the beginning, I was supported by an NDSEG fellowship, and for the last year, I've been supported by a research assistantship through MARCO, SRC, and NSF.

For being helpful in many ways, and just being generally great people to be around, I'd like to thank everyone else in Jacob's group in the time I was there: Igor Balk, Jaydeep Bardhan, Bjarne Büchmann, Luca Daniel, Suvranu De, Xin Hu, Jingfang

Huang, Tom Klemas, Dave Kring, Shihhsien Kuo, Junghoon Lee, Jing Li, Yehia Massoud, Deepak Ramaswamy, Michal Rewienski, Jürgen Singer, Ben Song, Anne Vithayathil, Frank Wang, Junfeng Wang, Xin Wang, Dave Willis, Wenjing Ye, and Zhenhai Zhu.

Thanks also go to Miguel Silveira and Sharad Kapur for helpful discussions, and to Keith Nabors for helping me out during my summer at Cadence.

I am grateful to Dave Grenda, Hugh Nicholson, Alex and Shelina Mallozzi, and Leo Trasande for being great friends during a time with many ups and downs for me. Stas Jarecki and Rina Panigrahy, old classmates from the theory group, have also been great friends and very supportive. I'd like to give a general thanks to all the people who have made my musical hobby possible, and to Eric Drucker and Filbert Hong in particular. Music has been a source of joy in good times and solace in difficult times, in the emotional roller-coaster of thesis research. Finally, all my life I have had the love and support of my family, and for that I am very thankful. Thanks Mom, Dad, and Bob!

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Motivation

Integrated circuits are typically built on a layered silicon substrate. In recent years, coupling through the substrate has come to have a substantial effect on overall circuit performance, for a variety of reasons including shrinking feature sizes and the trend toward integrating analog and digital blocks of circuitry on the same chip [1, 2, 3]. Switching noise from the digital block injects current into the substrate, which can then affect the sensitive circuitry of the analog block.

Many techniques have been developed for extracting substrate models [2, 4, 5, 6, 7, 8, 9, 10]. Some use coarse, inaccurate models to achieve efficiency: for example, treating the substrate as a single circuit node to which the contacts are connected by resistors. This would mean there is no dependence of current response at contact B to a voltage at contact A on the distance between the contacts, which isn't even approximately true for realistic contact layouts.

The others require a dense conductance matrix to be extracted (because the resistive substrate effectively links every contact to every other contact by a resistor). (In some cases, the inverse of the conductance matrix is extracted instead.) They develop efficient substrate solvers which can solve for contact currents given contact voltages (or, in some cases, for voltages given currents). There are two basic problems. First, the extraction of the dense matrix is very costly, requiring one solve on the discretized substrate per substrate contact, or $n$ solves for $n$ contacts. Second, if the goal is to include a model of the substrate in a circuit simulator, including the dense block corresponding to the conductance matrix is undesirable. This is true both in memory cost and in time: realistically, an analog block may have tens of thousands of contacts, leading to a conductance matrix $G$ with hundreds of millions or billions of entries. Even just applying $G$ to a vector may be computationally expensive.

In this work, we develop techniques for extracting a representation of $G$ quickly, assuming only a substrate solver which, given contact voltages, returns the contact currents, by reducing the number of solves required from $n$ to $O(\log n)$. In addition, the representation will be a "sparsified" representation of $G$, in the sense that it will typically require only $O(n \log n)$ operations to apply our representation to a vector, compared to the $n^2$ operations for applying the dense matrix-vector product. It is not immediately obvious how obtaining such a representation can improve circuit simulator efficiency, but the recent work of [11] provides important progress in this direction.

## 1.2 The problem

Our model of the substrate is simple: it is just a layered block of resistive (Ohm's law) material. Each layer has its own conductivity. The contacts are on the top surface. This is illustrated in Figure 1-1. Each contact is assumed to be a perfect conductor, so that the voltage on any contact is a constant.

Given voltages on the $n$ contacts, one can solve for the $n$ contact currents. (Each contact current is the current density integrated over the contact. Unlike voltage, current density is *not* uniform on a contact.) How this is formulated and solved numerically is the subject of Chapter 2. For us, the solver is viewed as a black-box: given a vector of the $n$ contact voltages, it returns a vector of the $n$ currents.

However, this black box is likely to be costly to apply, since its cost is related to the number of discretization points in whatever formulation is used, which can be many times greater than the number of contacts. The naive approach to obtaining

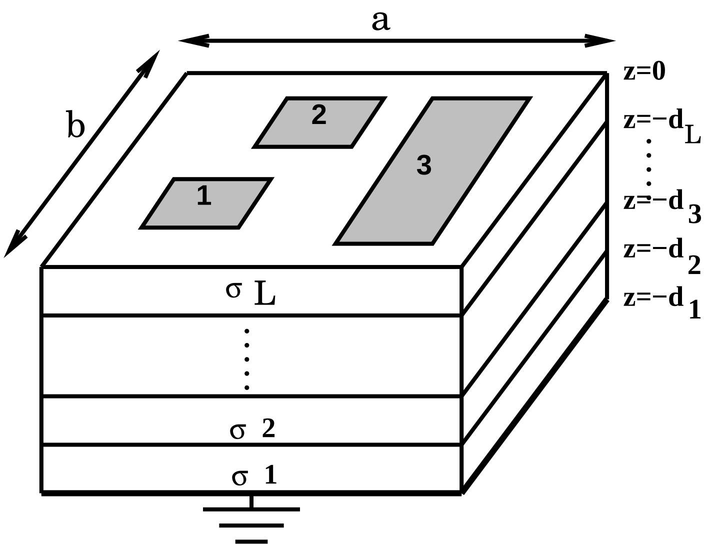

Figure 1-1: 3D layered substrate profile. Figure courtesy of Mike Chou.

the conductance matrix $G$ requires $n$ solves: we use the fact that $Ge_i = G(:,i)$, where $e_i$ is the $i^{\text{th}}$ standard basis vector, representing 1 volt on contact $i$ and 0 volts on all other contacts. Thus applying the black-box solver to the voltage function which is 1 volt on contact $i$ and 0 on all others, for each $i$, obtains $G$ with $n$ black-box calls. In this work we reduce this to $O(\log n)$ for reasonably regular contact layouts.

In addition, our algorithms obtain a representation of $G$ which can be applied in $O(n \log n)$ operations. Note that this $n$ is the number of contacts, and not the much larger number of discretization points used in the black-box solver.

## 1.3 Sparsification techniques

Many techniques have been developed which fall into the category of matrix sparsification. The term sparsification comes from the fact that applying a sparse matrix to a vector is much cheaper than applying a dense matrix of the same size, but the term is used more generally to mean any technique for applying a particular matrix $A$ to a vector that is reasonably accurate and more efficient than the standard $O(n^2)$ matrix-vector product. Such techniques obviously can only work for specific types of matrices, since just reading in the entries of a general $A$ costs $n^2$ operations.

Since our goal in this work is to get a good sparsification of $G$ efficiently, we should ask why previous sparsification techniques do not meet our needs.

Multipole methods [12, 13, 14] (along with the older Barnes-Hut treecode [15]) are perhaps the most famous of all sparsification techniques. However, the technique of [12] only applies to matrices which come from the $1/r$ kernel, or polynomials in $1/r$. This includes, for example, the panel potential-from-panel charge matrices used in capacitance calculation [16]. A kernel has been derived for the substrate problem, but it is not of $1/r$ form. In addition, it takes panel currents to panel potentials. Our conductance matrix $G$ instead takes *contact* potentials to contact currents.

Precorrected-FFT methods [17] can be applied to a broader range of problems because they only require translational invariance of the kernel; that is, the kernel $K(x, y, z; x', y', z')$ should be a function of $x - x'$, $y - y'$, and $z - z'$ only. Because of the finite thickness and sidewalls of the substrate, this is not satisfied even for the substrate coupling kernel, and certainly not for the conductance matrix.

Multiresolution methods based on the singular value decomposition, such as IES$^3$ [18], originally developed for capacitance computations, might seem to have more promise. Other techniques also based on using the SVD at many levels include [19, 20, 21]. However, they require constant-time access to the individual entries of the matrix being sparsified to be efficient. This is not a problem for $1/r$ matrices, but there is no easy way to get access to individual entries of the conductance matrix $G$. As mentioned earlier, all we assume is a black-box solver which can give currents on *all* the contacts given voltages on all contacts, in time proportional to the number of discretization points.

Assuming only access to a black-box solver is also useful because solvers which include more realistic substrate features, such as indentations in the substrate surface, can be included with no modifications to our algorithms.

## 1.4 Overview

In Chapter 2, we describe methods used for the underlying black-box substrate solver. Most of this is not new research, but is included for completeness and as a resource for workers starting out in this area. Chapter 3 gives a description of the wavelet-based sparsification algorithm. This is essentially an adaptation of the work of [22] to our problem, although there are some differences. In particular, the use of the combine-solves technique is new. The low-rank method of Chapter 4 is new, although one of the major underlying ideas, the use of the SVD to sparsify matrix sections corresponding to well-separated sets of contacts, has been well-known in the community [18]. The work of Chapter 4 was originally presented in [23].

# Chapter 2

# Substrate solvers

## 2.1 Overview

The algorithms we develop for extraction and sparsification of the substrate coupling conductance matrix are based on the idea of using a "black-box" *substrate solver*. This is simply a routine which, given voltages on the substrate contacts, returns currents on the contacts. There are two general approaches.

The first is a finite-difference scheme in which the entire three-dimensional substrate is discretized, Laplace's equation is solved in this volume to give voltages on all the nodes, and currents at the top-layer contact nodes are determined using Ohm's law and added for each contact to give the current at that contact. The second is an approach which uses variables on the top surface only, based on an analytic solution to Laplace's equation in a rectangular solid (with boundary and interface conditions determined from the conductivities of the substrate layers).

The basic advantage of the finite-difference approach is that it can deal with more general situations, such as irregular conductivity profiles and wells (indentations in the top substrate surface). However, in situations where the surface-variable approach is applicable, it is much more time and memory-efficient than finite difference, because the number of variables needed to discretize only the top surface is small compared to the number needed to discretize the whole volume as in the FD approach. In general one can have more confidence in the accuracy of the results as well.

Since our goal is a fast and accurate method for applying $G$, one might wonder why we don't simply use the "black-box" substrate solver, whether FD or surface-variable-based, instead of changing the basis of the conductance matrix. The reason is that the number of variables needed to discretize the substrate, or even the substrate contact surfaces, may be much larger than the number of contacts. Indeed, in [9], the one real example used has 478 contacts and 183905 panels associated with those contacts, or about 400 times as many. (Incidentally, this is the *only* example that the author is aware of in the substrate extraction literature which is substantial in size and comes from a real contact layout.)

We now turn to a discussion of the two methods, both of which we have used in our work.

## 2.2 Finite-difference solver

The finite-difference solver seems to have the advantage of simplicity. Poisson's equation

$$-\nabla \cdot (\sigma_i \nabla \phi(\mathbf{r})) = \rho(\mathbf{r}) \tag{2.1}$$

is discretized using finite-difference approximations for the derivatives. One standard way of doing this, which we develop below, results in a "grid of resistors" model of the substrate. The system of linear equations which arises has a number of variables (approximately) equal to the number of grid points for the 3-D grid. From the contact voltages, we set voltages at top-surface grid points which are within contact boundaries, and the system is solved to give voltages at all the grid points. There are several choices of algorithm for solving the system. From these voltages and using Ohm's law for the resistors terminating on contacts, the contact currents are computed. We now turn to each of these steps in more detail.

### 2.2.1 Discretization

How does discretizing Poisson's equation (2.1) lead to a grid of resistors? We examine this question in some detail in order to get a better understanding of the electromagnetic quantities involved and their interrelationships. Start with (2.1) in rectangular coordinates:

$$-\sigma\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}+\frac{\partial^2}{\partial z^2}\right)\phi(\mathbf{r})=\rho(\mathbf{r}). \tag{2.2}$$

We place a regular 3-D grid of points in the substrate. Finer grids give more accuracy, but at the cost of more memory and computation time. Because of the 3-D discretization, refinement by a factor of $k$ leads to a $k^3$ increase in memory requirements. As a practical matter, we found that memory requirements were the limiting factor in determining the size of problems we could attempt with the finite-difference approach.

Consider a typical point $(x,y,z)$ in the grid with associated voltage $v(x,y,z)$. For notational convenience we call this point $r$, with associated voltage $v_r$. We refer to its neighbors $(x-h,y,z)$ and $(x+h,y,z)$ by $r_{x-}$ and $r_{x+}$, with associated voltages $v_r^{x-}$ and $v_r^{x+}$ respectively, and similarly for its neighbors in the $y$ and $z$ directions. For now assume that $(x,y,z)$ and its neighbors are all in one layer of the substrate having conductivity $\sigma_i$. Then, using a finite-difference approximation for the derivative results in

$$\begin{aligned}-\sigma_i\left(\frac{\frac{v_r^{x+}-v_r}{h}-\frac{v_r-v_r^{x-}}{h}}{h}+\frac{\frac{v_r^{y+}-v_r}{h}-\frac{v_r-v_r^{y-}}{h}}{h}+\frac{\frac{v_r^{z+}-v_r}{h}-\frac{v_r-v_r^{z-}}{h}}{h}\right) &= \rho \\ \sigma_i\frac{6v_r-v_r^{x+}-v_r^{x-}-v_r^{y+}-v_r^{y-}-v_r^{z+}-v_r^{z-}}{h^2} &= \rho_r. \qquad (2.3) \\ & \qquad (2.4)\end{aligned}$$

The quantity $\rho_r=\rho(x,y,z)$ is *current flux density.* This is a scalar quantity representing current injection per unit volume, that is, the current per unit volume injected into an infinitesimal ball around $(x,y,z)$. It is important to distinguish current flux density from current density, which is a vector quantity (units of current per unit area) representing the direction and strength of current at a given point. Assuming a purely resistive substrate, there can be no current flux into or out of any point in the substrate interior. This of course doesn't mean there's no current at that point, only that inflow of current must be balanced by outflow.

The only places with nonzero current flux density are the substrate contacts on the top surface, as well as a backplane contact when there is one (this is a single large contact covering the entire bottom surface of the substrate). In fact, having nonzero current flux density anywhere in the substrate is a consequence of the fact that we're considering the substrate and its contacts in isolation from the rest of the circuit. In the real circuit, of which the substrate is a part, the contacts connect to circuit elements such as transistors and the interconnect between them, and there is no current flux at the contacts when considered as part of the whole circuit. Since we've isolated the substrate, however, the currents from the rest of the circuit must be modeled by a nonzero current flux density at the contacts.

A further confusing aspect of the situation is that although current flux density is a per-unit-volume quantity (A/cm$^3$), in our model the substrate contacts have no thickness, so the total current is obtained by integrating over the contact area (not volume). This implies that the flux density is a multiple of a delta function, i.e.

$$\rho(x, y, z) = \delta(z)\rho_{\text{surface}}(x, y).$$

### Grid-of-resistors interpretation

Discretizing Poisson's equation as in (2.3) is equivalent to modeling the substrate with a grid of resistors as shown in Figure 2-1, where the resistance $R$ for each resistor entirely within layer $i$ is determined from the resistivity $1/\sigma_i$ by the familiar formula

$$R_i = \frac{L}{A} \cdot \frac{1}{\sigma_i} \tag{2.5}$$

where $L$ is the wire length and $A$ the area. (Layer boundaries are discussed later.) To see this, multiply both sides of (2.3) by $h^3$ where $h$ is the grid spacing, obtaining

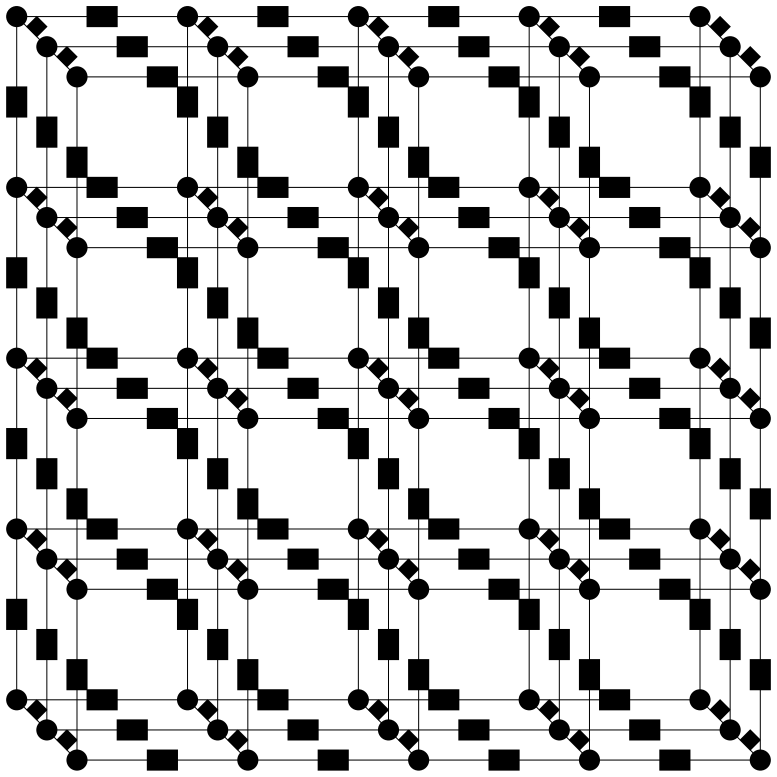

Figure 2-1: 3-D grid of resistors (resistors shown as conductances)

$$\sigma_i h(6v_r - v_r^{x+} - v_r^{x-} - v_r^{y+} - v_r^{y-} - v_r^{z+} - v_r^{z-}) = \rho_r h^3 = i_r. \tag{2.6}$$

Since $\rho_r$ is current flux density in A/cm$^3$, $\rho_r h^3$ is the current $i_r$ leaving an $h$-sided cube whose center is the node (grid point) $r$. So (2.6) relates voltages at a node and its neighbors to the node current (net current out of the node). We can derive the same equation from (2.5) and Ohm's law. The total current out of a node is

$$\begin{aligned} \frac{\sigma_i A}{L}\Big((v_r - v_r^{x+}) + (v_r - v_r^{x-}) + (v_r - v_r^{y+}) \\ +(v_r - v_r^{y-}) + (v_r - v_r^{z+}) + (v_r - v_r^{z-})\Big) &= i_r \\ \sigma_i h(6v_r - v_r^{x+} - v_r^{x-} - v_r^{y+} - v_r^{y-} - v_r^{z+} - v_r^{z-}) &= i_r. \end{aligned} \tag{2.7}$$

Equations (2.7) and (2.6) are the same.

### Layer boundaries

Having derived the equation for a node whose neighbors are all in its layer, we now consider the case of a node having a neighbor in a different layer. We assume that the node doesn't fall on a layer boundary; in fact, for convenience we have always placed the layer boundaries halfway between constant-$z$ planes of nodes in our experiments.

The resistors in the grid which cross layer boundaries are the only ones we need to consider. Such a resistor will be in the $z$-direction and have a fraction $p$ of its resistance in layer $i-1$ and a fraction $1-p$ in layer $i$ when the layer boundary is a fraction $p$ of the distance between the two planes of grid points, one in layer $i-1$ and the other in layer $i$, which are closest to the layer boundary.

If the "standard" (length $h$) resistor with resistance $R$ is reduced to a fraction $p$ of its original length, the resistance is now $Rp$. Since the length-$h$ resistor in layer $i$ has conductance $\sigma_i h$, we have

$$\begin{aligned}
\text{length } (p) \text{ layer } (i-1) \text{ resistance} &= \frac{p}{\sigma_{i-1}h} \\
\text{length } (1-p) \text{ layer } (i) \text{ resistance} &= \frac{1-p}{\sigma_i h} \\
\text{total resistance} &= \frac{p}{\sigma_{i-1}h} + \frac{1-p}{\sigma_i h} \\
\text{conductance} &= \frac{h}{\frac{p}{\sigma_{i-1}} + \frac{1-p}{\sigma_i}} \qquad (2.8)
\end{aligned}$$

This is shown in Figure 2-2.

Equation 2.7 becomes

$$\begin{aligned}
&(g_r^{x+} + g_r^{x-} + g_r^{y+} + g_r^{y-} + g_r^{z+} + g_r^{z-})v_r \\
-g_r^{x+}v_r^{x+} - g_r^{x-}v_r^{x-} - g_r^{y+}v_r^{y+} - g_r^{y-}v_r^{y-} - g_r^{z+}v_r^{z+} - g_r^{z-}v_r^{z-} &= i_r, \qquad (2.9)
\end{aligned}$$

where $g_r^{x+}$ is the conductance of the resistor connecting the central point $r$ to $r_{x+}$ and similarly for $g_r^{x-}$, $g_r^{y-}$, $g_r^{y+}$, $g_r^{z-}$ and $g_r^{z+}$.

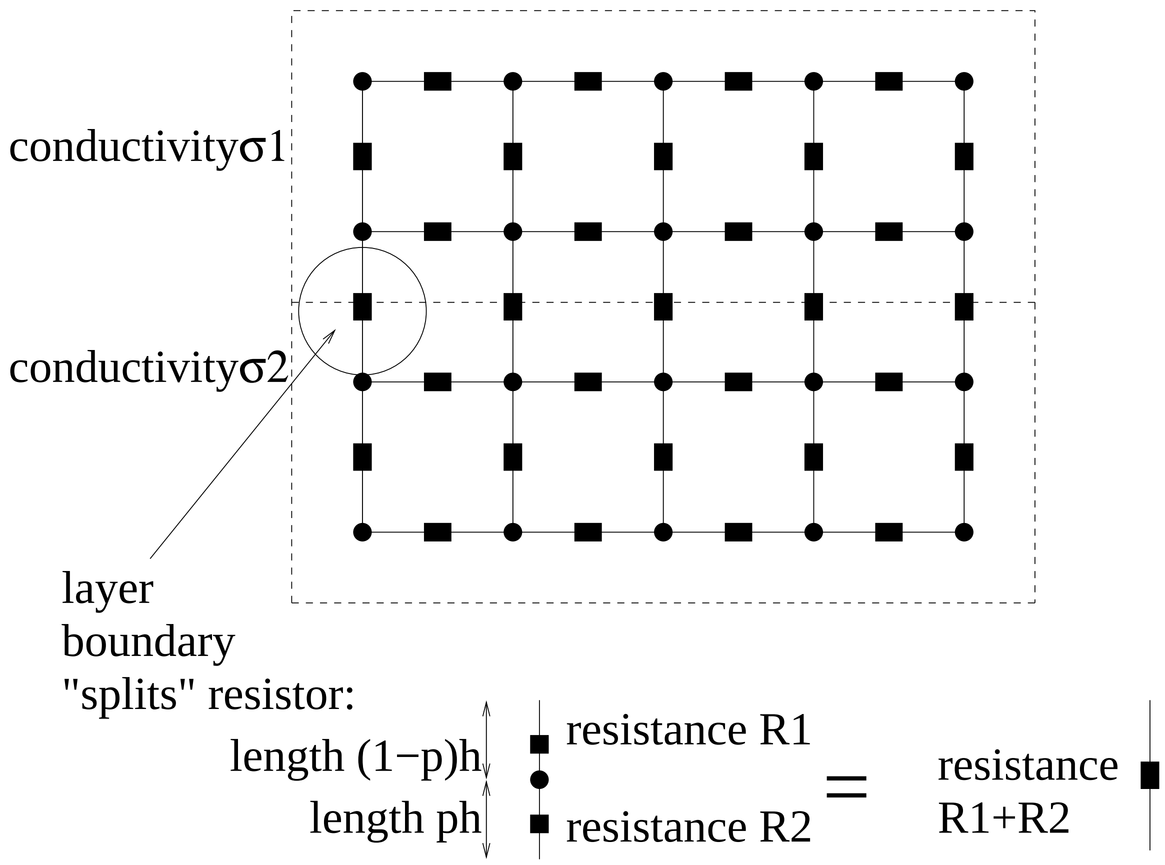

Figure 2-2: Resistance of layer-boundary resistor is found by viewing it as two resistors in series

## Exterior boundary conditions

In the grid-of-resistors discretization, the Neumann (zero-current) boundary conditions which hold at the substrate sides, the non-contact portion of the top, and, in the no-backplane-contact case, on the bottom, are naturally included by simply omitting resistors. For example, the point $r$ which has the smallest coordinate in all three dimensions will have resistors to $r_{x+}$, $r_{y+}$, and $r_{z+}$ only.

We can relate this to the continuous equation

$$\frac{\partial v}{\partial n} = 0 \tag{2.10}$$

by applying the finite-difference approximation to (2.10). For example, if $r$ is a grid point in the interior of the $x = x_{\text{min}}$ face (i.e., on a sidewall), the FD approximation of (2.10) is

$$\begin{aligned} \frac{v_r - v_r^{x-}}{h} &= 0 \\ \text{so} \quad v_r &= v_r^{x-}. \end{aligned} \tag{2.11}$$

Substituting in (2.9), we get

$$(g_r^{x+} + g_r^{y+} + g_r^{y-} + g_r^{z+} + g_r^{z-})v_r - g_r^{x+}v_r^{x+} - g_r^{y+}v_r^{y+} - g_r^{y-}v_r^{y-} - g_r^{z+}v_r^{z+} - g_r^{z-}v_r^{z-} = i_r.$$

This is exactly (2.9) with the resistor from $r$ to $r_{x-}$ removed. We know that the finite-difference approximation (2.11) is most accurate for approximating the derivative at $(r + r_{x-})/2$, so the sidewall should be halfway between $r$ and $r_{x-}$. The effect of this is shown in Figure 2.2.1. The substrate is divided into cubes and there is a grid point at the center of each cube. The grid spacing is $h$ but the spacing from the boundary grid points to the substrate boundary is $h/2$.

Finally, we need a way to set contact voltages and allow current injection into the substrate. This is a Dirichlet boundary condition at the top-surface contacts, and at the backplane contact when it exists. Where should a contact grid point be in

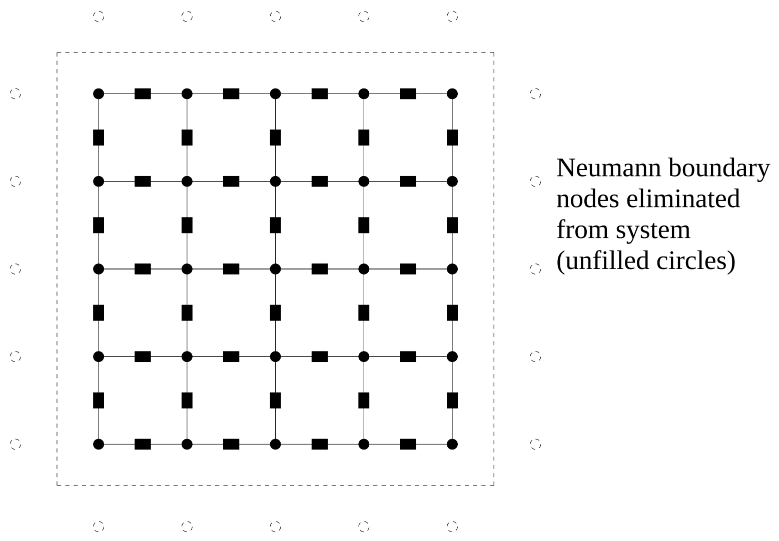

Figure 2-3: View of constant-$z$ substrate slice. Dashed line is substrate boundary, at distance $h/2$ from grid points for intergrid spacing $h$.

relation to the substrate's top surface? It seems clear that since the contact, whose voltage is known, is on the substrate surface, the grid point should be exactly on the substrate surface. But we have seen that Neumann boundary points should be at a distance $h/2$ from the substrate surface, so ideally the Dirichlet boundary points should be "offset" by $h/2$ from the rest of the grid, which has spacing $h$.

For convenience of implementation, in order to maintain a regular grid structure, we didn't do this. There are two choices for placing the Dirichlet boundary points which are "closest" to the ideal just discussed. Placing the boundary points an $h/2$ distance above the top substrate surface (the layer just outside the substrate) was the choice we made initially. See Figure 2-4.

In the system of equations, the Dirichlet nodes are *not* included as variables, because the node voltage is known. Call the node $d$. Then $v_d$ can be eliminated from equations which originally included it, by substituting its known value and moving it to the right-hand-side. This is the reason we originally chose the Dirichlet boundary points in the layer just outside the substrate: then the variables which remain form a full regular 3-D grid. With this choice, each eliminated node modifies one equation, corresponding to the node directly below it.

The other choice is to place the Dirichlet points a distance $h/2$ into the substrate

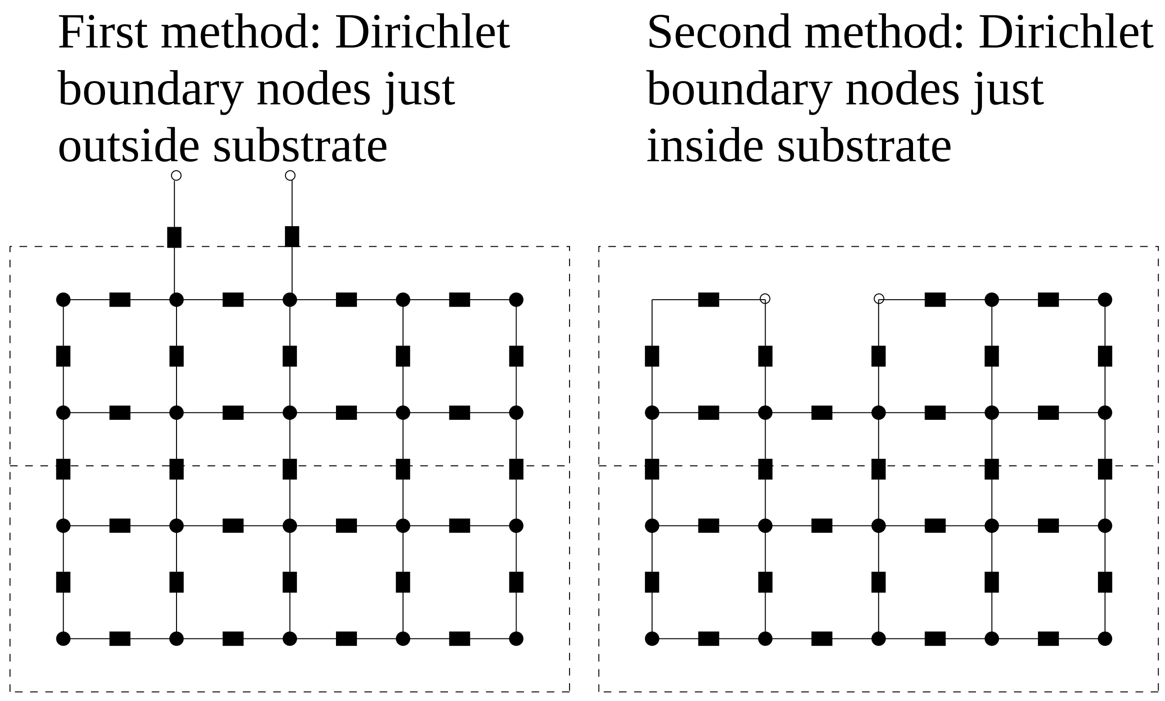
First method: Dirichlet boundary nodes just outside substrate
Second method: Dirichlet boundary nodes just inside substrate

Figure 2-4: Two choices for placement of contact (Dirichlet boundary) nodes

from the top substrate surface. This is less convenient to implement, because once these Dirichlet points are eliminated, the variables in the system no longer form a regular grid. Also, each eliminated node now modifies several equations, the nodal equations for its (generally five) neighbors, instead of just one.

Although it is clear that in the limit as $h \to 0$, the choices will give the same results, we needed to use fairly coarse grid spacings to be able to try problems with a reasonably large number of contacts. In fact, we found substantial differences in the results. The first choice resulted in substantially better sparsification performance. As a result, in order to present our results conservatively, we use the *second* choice for the FD solver used to generate the results presented in this work.

### 2.2.2 Solving the system

There are several choices for solving the system of equations generated by discretizing Poisson's equation as described. The number of variables can be quite large: in the largest examples we've tried with the finite-difference approach, there are about 4 million variables. Since the solve step needs to be repeated many times, it's critical to get an efficient solver to be able to try even reasonably large examples. We discuss several possible approaches, starting with the simplest.

### Cholesky factorization

Since the Laplacian matrix $A$ for which we're trying to solve $Ax = b$ is symmetric positive definite (see Section 2.4), the obvious method is Cholesky factorization. It requires $O(n^3)$ operations for a dense $A$, clearly unacceptable for the $n = 4 \cdot 10^6$ example mentioned above. Luckily $A$ is sparse, and in particular the 3D grid structure of the connections makes it possible to use a sparse Cholesky method requiring only $O(n^2 \log n)$ operations for the factorization and $O(n^{4/3} \log n)$ nonzero entries in $L$ and $U$. However, this is still not acceptable for large problems.

The $n^{4/3} \log n$ figure assumes a true 3D grid structure with equal numbers of grid points in each dimension. One might argue that since the substrate is typically a "thin" domain, so few layers need to be used in the $z$ dimension that it is really a quasi-2D problem. This is a questionable assumption (in [9], the real example used is 1 mm $\times$ 1 mm in the $xy$-plane and 0.4 mm in the $z$ direction.) Even with this assumption, though, the best case of a 2D grid requires $O(n^{3/2} \log n)$ operations for the solve.

### ICCG: PCG with incomplete Cholesky preconditioning

Our first attempt at a finite-difference substrate solver used preconditioned conjugate gradient (PCG). The conjugate gradient method is a well-known Krylov subspace method for solving $Ax = b$ in the case where $A$ is symmetric positive definite (s.p.d.). Krylov methods attempt to find an approximation $x_k$ to the exact solution $x^*$ in the *Krylov space*

$$\mathcal{K}_k(A, b) = \langle b, Ab, A^2b, \ldots, A^{k-1}b \rangle \tag{2.12}$$

where the $\langle\rangle$ notation means "space spanned by".

(An nice exposition of the conjugate gradient method is given in [24].) To summarize the important points about Krylov methods for us:

- They require a "black box" which, given a vector $x$, produces the matrix-vector product $Ax$.
- The matrix-vector product operation is repeated $k$ times (or iterations) leading to more accurate approximations $x_k$ to the exact solution $x^*$ of $Ax = b$ as $k$ grows larger.
- The number of iterations needed to converge to a given residual tolerance is related to conditioning of $A$: a poorly conditioned $A$ may require many iterations while a well-conditioned $A$ will require few.

The "black box" in our case is just the operation of multiplying the sparse $A$ by a vector, using the standard sparse matrix-vector product algorithm. In general, though, the matrix doesn't even need to be represented explicitly, as long as there's a way to compute $Ax$ given $x$ (for example, a circulant matrix can be applied efficiently using FFT-based convolution).

The effect of the conditioning of $A$ on iteration count is crucial. For many matrices that come up in practice, including our Laplacian, the conditioning is very poor and the iteration count is too large for conjugate gradient (or other Krylov methods) to be immediately practical. The key to making these algorithms practical is the idea of *preconditioning*. For PCG, the idea is to solve the equivalent equations

$$M^{-1/2}AM^{-1/2}\tilde{x} = M^{-1/2}b \tag{2.13}$$

$$x = M^{-1/2}\tilde{x} \tag{2.14}$$

for an appropriate choice of $M$. $M$ is called the *preconditioner*. The straightforward implementation is to apply the conjugate gradient algorithm to (2.13), and then apply $M^{-1/2}$ to the approximate solution $\tilde{x}_k$ to get $x_k$ in (2.14). $M$ is required to be s.p.d., so that $M^{-1/2}$ exists and $M^{-1/2}AM^{-1/2}$ is s.p.d. This would seem to require the ability to apply $M^{-1/2}$ to a vector. In fact, however, it's possible to implement an equivalent procedure (in the sense that for all $k$, in exact arithmetic the result $x_k$ after $k$ iterations is the same as in the original procedure), which requires only the ability to apply $M^{-1}$. See [25] for details.

The strategy is to choose $M$ so that $M^{-1/2}AM^{-1/2}$ is well-conditioned. As an extreme example, choosing $M = A$ results in

$$M^{-1/2}AM^{-1/2} = A^{-1/2} * A * A^{-1/2} = I.$$

with minimum possible condition number of 1. But we need to be able to apply $M^{-1}$, which is exactly the original problem if $M = A$! In practice $M$ should be a cheap-to-apply approximate inverse of $A$, where approximation quality versus quickness of applying $M^{-1}$ is a tradeoff.

A common choice for $M$, and the first one we tried, is the "incomplete Cholesky" operator. $A$ is approximately factored into $A \approx L_a L_a'$ by applying Cholesky factorization to $A$ but only allowing nonzero entries in $L_a$ in positions with nonzeros in $A$ (i.e., disallowing fill-in). Thus $L_a$ is twice as sparse as $A$ (it has nonzeros only in the lower-triangular part of $A$). The approximate $M^{-1} = (L_a L_a')^{-1} \approx A^{-1}$ is applied to a vector $y$ to give a result $z$ by

$$z = (L_a L_a')^{-1} y = L_a'^{-1} L_a^{-1} y.$$

$L_a^{-1}$ is applied using forward substitution and then $L_a'^{-1}$ is applied using back-substitution, for a total computational cost equal to that of applying $A$.

The incomplete Cholesky preconditioner is cheap, but unfortunately not very effective. We found that even for fairly simple examples hundreds of iterations were required for convergence to a reasonable tolerance.

### Fast-solver preconditioners

We found that using preconditioners in the class of fast Poisson solvers reduced the iteration count substantially. Fast Poisson solvers [26, 27] provide exact, direct solutions in $O(n \log n)$ time to the grid-of-resistors problem when the grid is regular and the boundary conditions are *uniform* on each face. That is, the boundary conditions are either Dirichlet (given voltage) or Neumann (given current–0 in our case) conditions on each face, but not a mixture of both on the same face.

The methods work by using the fact that in the case of uniform boundary conditions there is a change of basis, which can be applied cheaply using the 2-D Discrete Cosine Transform (DCT) in the $x$ and $y$ directions [28, 29], in which the grid-of-resistors system becomes many decoupled tridiagonal systems, each of which is easy to solve. An implementation can be found in [30].

We emphasize that the uniform boundary condition requirement is *not* satisfied for us, so the fast solvers cannot be used directly. In particular, the top face has nodes which correspond to contact surfaces (Dirichlet b.c.) and nodes corresonding to non-contact surfaces (Neumann b.c.). However, it is reasonable to view the fast solver operator (call it $M^{-1}$) as an *approximate* inverse of $A$. Then $M$ can be used as the preconditioner in PCG.

For simplicity, we describe how the preconditioner is implemented for the *first* choice of placement of the Dirichlet nodes described in Section 2.2.1, where they are placed just outside of the substrate surface so that eliminating them from the system results in a regular 3D grid of variables. The details are different for the second choice of placement but the essential idea is the same. Also to keep things simple, assume that the two layers of grid points directly below the top substrate surface are in the same conductivity layer, with conductivity $\sigma_L$.

If $r$ is a top-layer node directly below a Dirichlet boundary node, the equation (2.7) becomes

$$\sigma_L h(6v_r - v_r^{x+} - v_r^{x-} - v_r^{y+} - v_r^{y-} - v_r^{z-}) = i_r. \tag{2.15}$$

On the other hand, if $r$ is a top-layer Neumann boundary node, (2.7) becomes

$$\sigma_L h(5v_r - v_r^{x+} - v_r^{x-} - v_r^{y+} - v_r^{y-} - v_r^{z-}) = i_r. \tag{2.16}$$

The only difference is the coefficient of $v_r$. We thus have two possibilities for a fast-solver preconditioner: a pure-Dirichlet preconditioner where $M$ is created by assuming there is a Dirichlet node above every top-layer node, or a pure-Neumann precondi-

| Preconditioner | Average # iterations |
|---|---|
| Dirichlet | 22.2 |
| Neumann | 7.9 |
| area-weighted | 6.8 |

Table 2.1: Preconditioner effectiveness

tioner where $M$ is created by assuming that every top-layer node is a Neumann node.

In fact, these are not the only possibilities. One can reduce the conductance of the resistor connecting the top-layer nodes to the Dirichlet nodes above in the construction of $M$ to a fraction $p$ of its original value. Then $p = 1$ gives the pure-Dirichlet preconditioner while $p = 0$ gives the pure-Neumann preconditioner (the "Dirichlet node" above is totally disconnected from the rest of the grid and has no effect on it). We can choose any intermediate value of $p$ as well. One choice that makes sense intuitively is to choose

$$p = \frac{\text{total area of contacts}}{\text{total area of substrate top surface}}.$$

In Table 2.1, we present results giving the average number of iterations per solve for PCG with a fast-solver preconditioner based on these ideas, for a simple regular contact layout, over the several hundred solves required to implement one of the sparsification algorithms described later. The pure-Dirichlet preconditioner, while a vast improvement over no preconditioner and the incomplete Cholesky preconditioner, is not as good as the pure-Neumann preconditioner, and the area-weighted idea just described works best of all.

### Multigrid

Multigrid techniques for Poisson's equation [31, 32] have a long history [33, 34], and may be very useful here. The iteration counts could possibly be reduced somewhat, and each iteration would probably cost less than for PCG, where each iteration involves applying a fast Poisson solver. Dealing with layer boundaries properly in the coarse-grid representation would be the major issue.

## 2.3 Surface variable methods

The finite-difference approach requires the discretization of the whole substrate volume. Since the quantities of interest, contact currents and voltages, are on the top substrate surface only, it is natural to try to develop methods which use surface variables only. We might hope for an approach analogous to the well-known integral equation method for computing charge density given potentials on conductor surfaces:

$$\phi(f) = \frac{1}{4\pi\epsilon_0} \int_S \rho(s) \frac{1}{||f - s||} dA$$

where $S$ is the collection of conductor surfaces, $f$ is the field point, and $s$ is the source point. The multiplier $1/||f - s||$, traditionally denoted $1/r$, is called the Green's function. In fact this is also possible for the multilayer substrate coupling problem (see [7]). Unfortunately the substrate coupling current density-to-potential Green's function is much more complicated than $1/r$, actually an infinite series.

### 2.3.1 Eigenfunction-based approach

Fortunately, an alternative approach which also uses only surface variables has been developed [35, 36, 9]. The operator $\mathcal{A}$ which is applied to surface current density $i(x, y)$ and gives surface potential $v(x, y)$ can be analyzed in terms of its eigenfunctions. An *eigenfunction* $f(x, y)$ satisfies

$$\mathcal{A}f(x, y) = \lambda f(x, y). \tag{2.17}$$

If $a$ and $b$ are the substrate length and width, as in Figure 1-1, it turns out that the eigenfunctions form an infinite family of functions $f_{mn}$, one for each ordered pair of nonnegative integers $(m, n)$:

$$f_{mn}(x, y) = \cos\left(\frac{m\pi x}{a}\right) \cos\left(\frac{n\pi y}{b}\right). \tag{2.18}$$

The eigenvalues $\lambda_{mn}$ associated with the $f_{mn}$ are easy to calculate given the layer

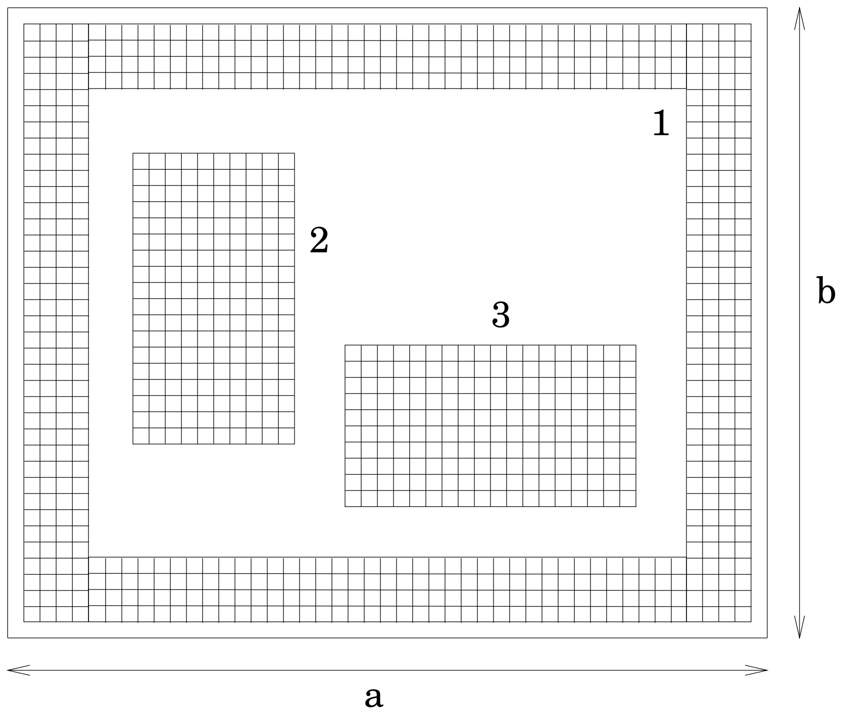

Figure 2-5: Example: three contacts discretized into panels. Figure courtesy of Mike Chou.

thicknesses and conductivities. A procedure is given in [9] for the case of a grounded backplane contact, but it isn't derived there. Later in this section we give a derivation which also covers the easy generalization to the no-backplane-contact case.

The many additional pieces that make up an eigenfunction-based substrate coupling solver are discussed in [9]. The substrate is discretized into square panels (see Figure 2-5). Call the operator which takes contact panel currents to contact panel potentials $A$. (Non-contact panel currents are 0.) We have

$$Ai = v$$

and the $i$ (current) vector has one entry per contact panel, representing total current on that panel. The $v$ (voltage) vector also has one entry per contact panel, representing average potential on that panel. The simple cosine-mode form of the eigenfunctions carries over to the discretized version. The result is a simple procedure for applying $A$, which starts by putting zeros in all non-contact panels, then

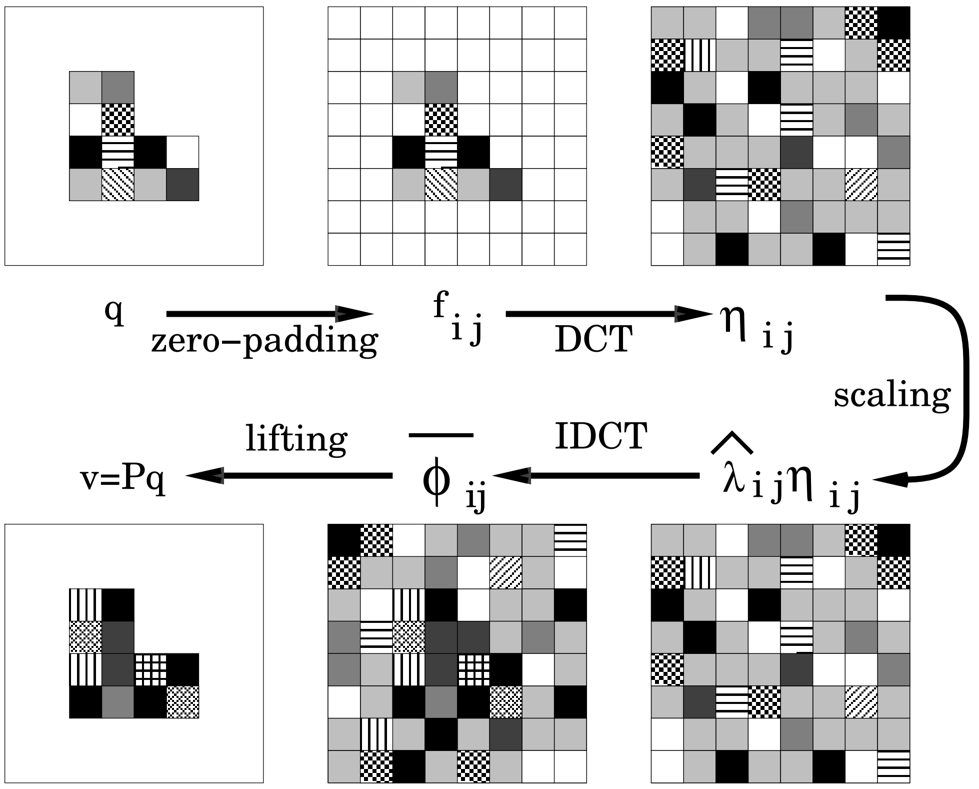

Figure 2-6: Applying the current-to-potential operator using eigendecomposition. Figure courtesy of Mike Chou.

applies a 2-D DCT to convert to the basis of surface eigenfunctions, multiplies by the eigenvalues, applies the 2-D inverse DCT to convert back to the standard basis. Now the average potential has been calculated on all panels, but is needed only on the contact panels. This is shown schematically in Figure 2-6 ($q$ is used for the vector of currents). Then this algorithm for applying $A$ can be used in an iterative method for solving $Ai = v$.

### "Fast-solver" preconditioner?

We might consider a "fast-solver" type of preconditioner for the eigendecomposition approach. That is, can we use an exact solver for a closely related problem as a preconditioner? To see how this might work, examine the steps in Figure 2-6. All the arrows are reversible except the last one ("lifting"). The IDCT is inverted to a DCT, scaling by $\lambda_{ij}$ becomes scaling by $1/\lambda_{ij}$, the DCT becomes an IDCT, and the zero-padding becomes lifting.

The "lifting" step can't easily be reversed, since we do not have the voltages on the non-contact surfaces. (Voltages are only specified on the contact surfaces when we solve for the currents.) This is in contrast to the "zero-padding" step—we know that currents are 0 on non-contact surfaces. However, for purposes of a preconditioner we may simply zero-pad the voltages as well, or make the non-contact voltages some simple function of the (known) contact voltages.

Experiments we did using this idea indicate that it is not promising (the number of iterations isn't reduced much, if at all). Why are fast-solver preconditioners effective in the finite-difference formulation and not here? Intuitively, the reason may be that the proportion of variables for which the preconditioner $M$ is not doing the same thing as the original $A$ is much larger in the surface-variable case. In both the FD and surface-variable formulations, the non-contact substrate surface is where $M$ is different from $A$, but in the surface-variable case, this is a much larger proportion (typically something like 75%) of the problem domain than in the FD case, where $M$ and $A$ match in the substrate interior.

### Performance comparison

The approach developed in [9] is quite different, based on multigrid ideas. The use of multigrid for solving integral equations is developed here and in [37]. The result is consistently fast convergence (fewer than 10 iterations for $10^{-6}$ relative residual tolerance). With permission of the author of [9], we have used the QuickSub code he developed for the substrate problem in our work.

As mentioned earlier, a substrate can either have a backplane contact or no such contact (floating backplane). We have been more interested in the floating-backplane case, since it results in more global coupling. The QuickSub code, however, requires a backplane contact.

The way we actually dealt with this in using QuickSub to produce results was to include a very resistive layer adjacent to the groundplane. This isn't an ideal solution for the floating-backplane case, because in order to accurately approximate the lack

| | Iterations/solve | Time per solve (s) |
|---|---|---|
| finite difference | 7.0 | 3.8 |
| eigenfunction | 6.0 | 0.4 |

Table 2.2: Solve speed for finite-difference versus eigenfunction methods

of a groundplane, the layer needs to become so resistive that the system becomes nearly singular (slight changes in the net current injected from the top will produce very large changes in contact potentials). This destroys accuracy and slows down convergence.

However, our purpose was not to get an extremely accurate simulation of the floating-backplane case, but simply to look at a situation where there is significant global coupling, in order to evaluate our sparsification algorithms. For this purpose, inserting a resistive layer of only moderately high resistance works fine.

Table 2.2 shows the average time per solve for 10 solves on an example problem, as well as the average number of iterations per solve. The eigenfunction-based approach is approximately 10 times as fast as the finite difference approach for this example, which is based on the substrate thickness of the phase-lock-loop example from [9]. While speedups will vary by example, in general the eigenfunction approach is much faster. There are two reasons for this: iteration counts are generally smaller, and the time per iteration is smaller because the number of variables is much smaller in the surface-based approach.

### Derivation of eigenfunctions and eigenvalues

Although [9] gives a recursive formula for the eigenvalues associated with the eigenfunctions $f_{mn}$, there appears to be no derivation of the formula in [9] or its references. While this type of Green's function derivation is typical in computational electromagnetics, we believe that presenting a complete derivation serves a useful purpose. First, we see that applying the eigendecomposition approach to applying $A$ is just as easy in the floating-backplane case as in the grounded-backplane case which [9] restricts itself to. Whether this can be extended to the multigrid methods developed in [9] is an interesting topic for further work. Second, seeing the derivation presented clearly may be helpful to workers starting out in this area without an extensive computational EM background.

Poisson's equation (2.1) becomes

$$\sigma_i \left( \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} \right) \phi_{m,n}(z) = 0$$

in the substrate interior (there are no sources and thus no current flux density $\rho$). We are led to look for functions of the form

$$\phi(x, y, z) = \tilde{\phi}(z) \cos(\alpha x) \cos(\beta y) \tag{2.19}$$

as solutions essentially because cosine functions are eigenfunctions of the second derivative operator, reducing the Laplacian to a simple single-variable differential equation.

$$\begin{aligned} & \left( \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} \right) \tilde{\phi}(z) \cos(\alpha x) \cos(\beta y) \\ = & \left( -\alpha^2 - \beta^2 + \frac{\partial^2}{\partial z^2} \right) \tilde{\phi}(z) \cos(\alpha x) \cos(\beta y) = 0 \qquad (2.20) \\ \left( -\alpha^2 - \beta^2 + \frac{\partial^2}{\partial z^2} \right) \tilde{\phi}(z) = & \ 0 \qquad (2.21) \end{aligned}$$

There are many possible choices of $\alpha$ and $\beta$, but we want to choose those which satisfy the sidewall Neumann boundary conditions $\frac{\partial \phi}{\partial x}|_{x=0} = \frac{\partial \phi}{\partial x}|_{x=a} = \frac{\partial \phi}{\partial y}|_{y=0} = \frac{\partial \phi}{\partial y}|_{y=b} = 0$. This leads to the choices $\alpha = m\pi/a$, $\beta = n\pi/b$ for any ordered pair $(m, n)$ of nonnegative integers. (Note that Dirichlet boundary conditions would require the use of sines instead of cosines.)

For a particular pair $(m, n)$, set $\gamma_{mn} = \sqrt{(m\pi/a)^2 + (n\pi/b)^2}$. Then the general solution to (2.21) is

$$\tilde{\phi}_{m,n}(z) = \zeta_{mn}^{(k)} e^{\gamma_{mn}(d+z)} + \xi_{mn}^{(k)} e^{-\gamma_{mn}(d+z)} \tag{2.22}$$

for $m, n$ not both 0. (If $m$ and $n$ are both zero the solution is of the form $\xi_{00} + \zeta_{00} z$.

The coefficients are obtained from boundary and interface conditions as in the general case—we will not give details.) The superscript $k$ indicates the solution holds in layer $k$. The idea is to "glue" solutions in different layers together, guided by the interface and boundary conditions.

We are looking for solutions of the form (2.19). We can make the guess that functions of the form $f_{mn} = c\cos(m\pi x/a)\cos(n\pi y/b)$ giving the top-layer surface current density will result in a solution of this form. If this is true (we will see that it is), this means that $f_{mn}$ is an eigenfunction of the surface current density to surface potential operator $\mathcal{A}$.

To get the potential solution in the whole substrate, we need to find the coefficients in (2.22). We have several conditions (see [38] for a general discussion of continuity conditions):

- top layer current in (Neumann boundary condition)

$$\sigma_L \frac{\partial \phi}{\partial z} = c\cos(m\pi x/a)\cos(n\pi y/b) \tag{2.23}$$

$$\frac{d\tilde{\phi}}{dz} = \frac{c}{\sigma_L} \quad \text{at } z = 0 \tag{2.24}$$

$$\zeta_{mn}^{(L)}\gamma_{mn}e^{\gamma_{mn}d} - \xi_{mn}^{(L)}\gamma_{mn}e^{-\gamma_{mn}d} = c/\sigma_L \tag{2.25}$$

- continuity of potential at layer boundaries

$$\zeta_{mn}^{(k)}e^{\gamma_{mn}(d-d_k)} + \xi_{mn}^{(k)}e^{-\gamma_{mn}(d-d_k)} = \zeta_{mn}^{(k-1)}e^{\gamma_{mn}(d-d_k)} + \xi_{mn}^{(k-1)}e^{-\gamma_{mn}(d-d_k)} \tag{2.26}$$

- current density component in layer boundary direction continuous

$$\sigma_k \frac{d\tilde{\phi}^+}{dz} = \sigma_{k-1}\frac{d\tilde{\phi}^-}{dz} \tag{2.27}$$

$$\begin{array}{c} \sigma_k\gamma_{mn}\zeta_{mn}^{(k)}e^{\gamma_{mn}(d-d_k)} \\ -\sigma_k\gamma_{mn}\xi_{mn}^{(k)}e^{-\gamma_{mn}(d-d_k)} \end{array} = \begin{array}{c} \sigma_{k-1}\gamma_{mn}\zeta_{mn}^{(k-1)}e^{\gamma_{mn}(d-d_k)} \\ -\sigma_{k-1}\gamma_{mn}\xi_{mn}^{(k-1)}e^{-\gamma_{mn}(d-d_k)} \end{array} \tag{2.28}$$

- boundary condition at bottom

– grounded backplane contact

$$\begin{aligned} \tilde{\phi}(-d) &= 0 && (2.29) \\ \zeta_{mn}^{(1)} &= -\xi_{mn}^{(1)} && (2.30) \end{aligned}$$

– no backplane contact

$$\begin{aligned} \left.\frac{d\tilde{\phi}}{dz}\right|_{z=-d} &= 0 && (2.31) \\ \zeta_{mn}^{(1)} &= \xi_{mn}^{(1)} && (2.32) \end{aligned}$$

The conditions (2.28) and (2.26) together give

$$\begin{pmatrix} e^{\gamma_{mn}(d-d_k)} & e^{-\gamma_{mn}(d-d_k)} \\ \gamma_{mn} e^{\gamma_{mn}(d-d_k)} & -\gamma_{mn} e^{-\gamma_{mn}(d-d_k)} \end{pmatrix} \begin{pmatrix} \zeta_{mn}^{(k)} \\ \xi_{mn}^{(k)} \end{pmatrix} \tag{2.33}$$
$$= \begin{pmatrix} e^{\gamma_{mn}(d-d_k)} & e^{-\gamma_{mn}(d-d_k)} \\ \frac{\sigma_{k-1}}{\sigma_k} \gamma_{mn} e^{\gamma_{mn}(d-d_k)} & -\frac{\sigma_{k-1}}{\sigma_k} \gamma_{mn} e^{-\gamma_{mn}(d-d_k)} \end{pmatrix} \begin{pmatrix} \zeta_{mn}^{(k-1)} \\ \xi_{mn}^{(k-1)} \end{pmatrix}$$

$$\begin{pmatrix} \zeta_{mn}^{(k)} \\ \xi_{mn}^{(k)} \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\left(1 + \frac{\sigma_{k-1}}{\sigma_k}\right) & \frac{1}{2}\left(1 - \frac{\sigma_{k-1}}{\sigma_k}\right) e^{-2\gamma_{mn}(d-d_k)} \\ \frac{1}{2}\left(1 - \frac{\sigma_{k-1}}{\sigma_k}\right) e^{2\gamma_{mn}(d-d_k)} & \frac{1}{2}\left(1 + \frac{\sigma_{k-1}}{\sigma_k}\right) \end{pmatrix} \begin{pmatrix} \zeta_{mn}^{(k-1)} \\ \xi_{mn}^{(k-1)} \end{pmatrix} \tag{2.34}$$

This gives a recursive formula for the level-$k$ coefficients in terms of those at level $k-1$. To start the recursion, note that for a grounded backplane, the Dirichlet condition (2.30) requires $(\zeta_{mn}^{(1)}, \xi_{mn}^{(1)}) = \theta(1, -1)$ for some scalar $\theta$. In fact we can choose $\theta = 1$ because the only condition which is affected by the choice is (2.25) and any scaling can be absorbed by the constant $c$. For the no-grounded-backplane (Neumann) case, we similarly get $(\zeta_{mn}^{(1)}, \xi_{mn}^{(1)}) = (1, 1)$. The *only* difference between the situations is in the base case of the recursion!

Now that we have the coefficients, the eigenvalue $\lambda_{mn}$ in $\mathcal{A} f_{mn}(x, y) = \lambda_{mn} f_{mn}(x, y)$ follows immediately by noting that $f_{mn}(x, y)$ is the surface current density at the top from (2.25) and $\mathcal{A}f_{mn}(x,y)$ is the potential at the top from (2.22):

$$\lambda_{mn} = \frac{\mathcal{A}f_{mn}(x,y)}{f_{mn}(x,y)} = \frac{\zeta_{mn}^{(L)} e^{\gamma_{mn} d} + \xi_{mn}^{(L)} e^{-\gamma_{mn} d}}{\sigma_L \left(\zeta_{mn}^{(L)} \gamma_{mn} e^{\gamma_{mn} d} - \xi_{mn}^{(L)} \gamma_{mn} e^{-\gamma_{mn} d}\right)} \tag{2.35}$$

For the $m = n = 0$ case (i.e., a uniform current density into the substrate from the top), going through the same process gives the recursion

$$\begin{pmatrix} \zeta_{00}^{(k)} \\ \xi_{00}^{(k)} \end{pmatrix} = \begin{pmatrix} \frac{\sigma_{k-1}}{\sigma_k} & 0 \\ \left(\frac{\sigma_{k-1}}{\sigma_k} - 1\right) d_k & 1 \end{pmatrix} \begin{pmatrix} \zeta_{00}^{(k-1)} \\ \xi_{00}^{(k-1)} \end{pmatrix} \tag{2.36}$$

and the eigenvalue is given by $\lambda_{00} = \xi_{00}^{(L)}/(\sigma_L \zeta_{00}^{(L)})$. Initial values are $\zeta_{00}^{(1)} = 1$ and $\xi_{00}^{(1)} = d$ in the grounded-backplane case. The no-backplane-contact case is interesting: $\frac{d}{dz}(\xi_{00}^{(1)} + \zeta_{00}^{(1)} z) = 0$ at $z = 0$ requires $\zeta_{00}^{(1)} = 0$. Then from the previous recursion it's clear than $\zeta_{00}^{(L)} = 0$ as well. But then $\lambda_{00} = \infty$. This makes sense physically: it's impossible to push a uniform current into the top of the substrate when there's no backplane contact for it to leave through.

## 2.4 Solution properties

It is useful to examine some of the basic properties of the conductance matrix $G$ being computed, for various reasons. First, it is a useful check in debugging, both for debugging the solvers and the sparsification algorithms which give a representation of $G$. If the computed $G$ doesn't satisfy the basic properties of symmetry and diagonal dominance, something is wrong. Second, symmetry of $G$ is useful in the wavelet algorithm and crucial to the development of the low-rank algorithm for getting a sparse representation of $G$. Finally, it is interesting to relate properties of the finite-difference matrix $A$ to those of $G$. These are well-known standard properties, but we try to give some intuition.

The grid-of-resistors matrix $A$ giving the voltage-to-current operator of (2.6) has interesting properties. First, it is symmetric. This is clear, because a resistor with conductance $\sigma$ between nodes $a$ and $b$ adds $\sigma$ to $A_{aa}$ and $A_{bb}$, and adds $-\sigma$ to $A_{ab}$ and

$A_{ba}$. So starting from the (symmetric) 0 matrix, each "stamping" operation maintains the symmetry of $A$. It is also diagonally dominant, i.e. $|A_{xx}| \geq \sum_{y \neq x} |A_{xy}|$. This diagonal dominance is strict at the nodes adjacent to Dirichlet boundary nodes, since one of the $-\sigma$ entries is removed from the row while the others remain the same. The diagonal dominance is "tight" at all other nodes (internal and Neumann boundary nodes). Diagonal entries are positive and all other non-zero entries are negative.

The symmetry of the computed $G$ ultimately comes from the symmetry of $A$ in the finite difference approach. In the eigendecomposition approach, the symmetry of $G$ results from the symmetry of the panel current-to-potential matrix, which can be seen by the orthogonality of the $Q$ in the $QDQ'$ eigendecomposition. The symmetry of the actual $G$, without approximation, comes from the self-adjointness of the Laplacian operator. Diagonal dominance holds because the diagonal entry $G_{jj}$ represents the current into contact $j$ when unit voltage is placed on contact $j$ and all other contacts are grounded. Thus current will flow into contact $j$, and out all the other contacts. The total current flowing out all the other contacts must exactly match the current flowing into contact $j$.

If there is a backplane contact, some of the current will flow out through it, so since $G$ represents only the interaction of the top-surface contacts, in this case the diagonal dominance is strict. If there is no backplane contact, the diagonal dominance is tight in every row. Again the diagonal entries are positive and all others negative. Note, however, a major difference from the $A$ matrix: $G$ is dense, as voltage at one contact produces current in all other contacts.

For the no-backplane case, $\sum_{i=1}^{n} G_{ij} = 0$ for all $j$ since the total current into the top substrate surface is 0. Thus there is a rank-one deficiency in the computed $G$. This should not be surprising, since with no backplate contact, increasing all the top-surface contact voltages by a constant DC offset will not affect the currents.

# Chapter 3

# Wavelet methods

We begin by giving some intuition for why a change-of-basis approach, approximating $G$ by $QG_{ws}Q'$, can be useful for sparsification of substrate coupling, and why the change of basis is obtained by a multilevel construction. We proceed to describe this construction more formally, by presenting an algorithm for forming the change-of-basis matrix $Q$. This algorithm guarantees an orthogonal $Q$. From this we get a solution (although not the most effective one, as we show in Chapter 4) to the sparsification problem, in the following way. We want

$$G = QG_wQ', \tag{3.1}$$

and using the orthogonality of $Q$ we obtain

$$G_w = Q'GQ. \tag{3.2}$$

Since this step can be reversed so that (3.2) implies (3.1), we can set $G_w = Q'GQ$ and then get a sparse approximation $G_{ws}$ of $G_w$ simply by dropping small entries in $G_w$. We give results later which show that this leads to much more accurate results than simply dropping small entries in the original $G$ for realistic problems.

The algorithms for computing $Q$ are based on those developed in [22] for $1/r$ potential-from-charge matrices. Here is a quick summary of the main differences.

First, since $G$ is a current-from-potential matrix, for us the analogous quantity to charge in [22] is potential, and the analogous quantity to potential in [22] is current. We use polynomial moments rather than multipole moments. The hierarchy is one of squares rather than cubes since the moments are on the 2-D substrate surface. For the $1/r$ kernel, [22] gives an error analysis showing that the wavelet sparsification is guaranteed to be effective: this analysis does *not* apply to our case, which is part of the motivation for developing the low-rank method of Chapter 4.

However, this only deals with the problem of getting a good sparsification of $G$, not with efficiently extracting that representation. In fact, naively applied, the method just described would require $n$ black-box calls and $O(n^2)$ work just to list the entries of $G$. A method of combining solves is applied to this situation to reduce the number of black-box solves to $O(\log n)$ and total work to $O(n \log n)$.

## 3.1 Intuition

Conductance matrix entries may decay slowly. That is, entries decay slowly in the standard basis, and there may be significant interaction even between the two most widely-separated contacts. By changing basis, we can get faster decay. To be more precise, the entry $G_{ij}$ can be expressed in terms of $G$ as

$$G_{ij} = e_i' G e_j \tag{3.3}$$

where $e_k$ is the $k$th standard basis vector—which is why the decay of entries of $G$ is called decay "in the standard basis". On the other hand, with a change-of-basis matrix $Q$,

$$(G_w)_{ij} = (Q'GQ)_{ij} = Q(:,i)'GQ(:,j). \tag{3.4}$$

So the $(i, j)$ entry of $G_w$ is obtained by projecting the current response to the $j$th basis vector onto the $i$th basis vector in the new basis given by $Q$. As a first step, we can look at how to obtain a $GQ$ with fast-decaying entries. We concentrate on four neighboring contacts. The standard-basis functions associated with these contacts are

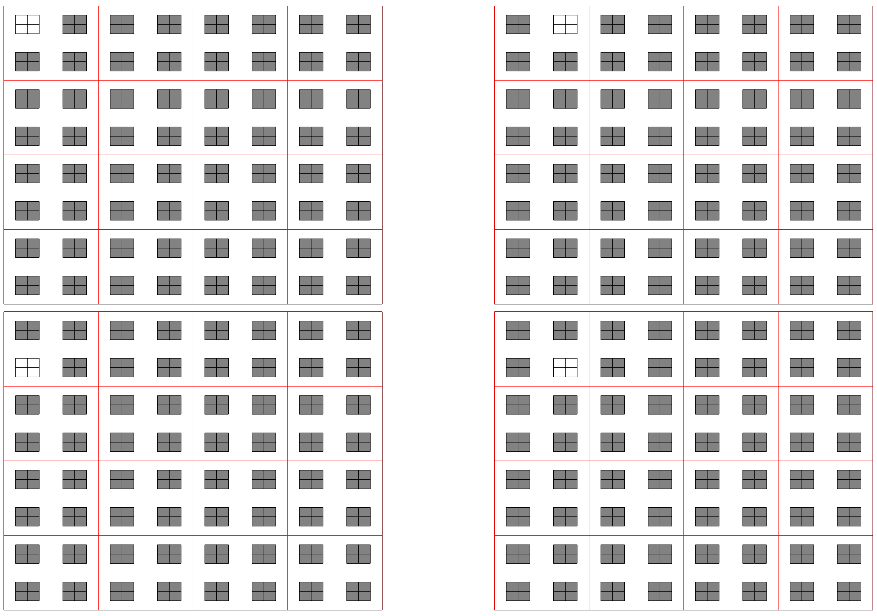
Figure 3-1: Standard basis voltage functions. White = 1 V, gray = 0 V

shown in Figure 3-1. To get fast-decaying entries in $GQ$, we try to find basis functions which are nonzero only in a small area, and have nearly zero current response well outside this area.

The intuition for finding basis functions with this property is that faraway current responses to nearby contacts look similar. Then by linearity, putting 1 volt on two of the four contacts and $-1$ volt on the other two should result in near-zero faraway current response. This can be used to find 3 basis functions with support only on the four contacts; but to cover the entire space of possible voltages on the contacts, which of course includes non-balanced voltage functions, we need one basis function which doesn't satisfy the balanced-voltage property. For this the all-1 volt function is used. The new basis functions are shown in Figure 3-2. The same idea can be used for all 16 groups of four contacts.

At this point, 3/4 of the basis functions have the balanced-voltage property, but 1/4 do not and will not have fast-decaying faraway currents. This is what leads to the idea of a multilevel construction for the new basis. Four all-1 basis functions from the finest level are shown in Figure 3-3. They can be recombined to form four basis functions with support only in one square on the next-coarser level. This is shown in

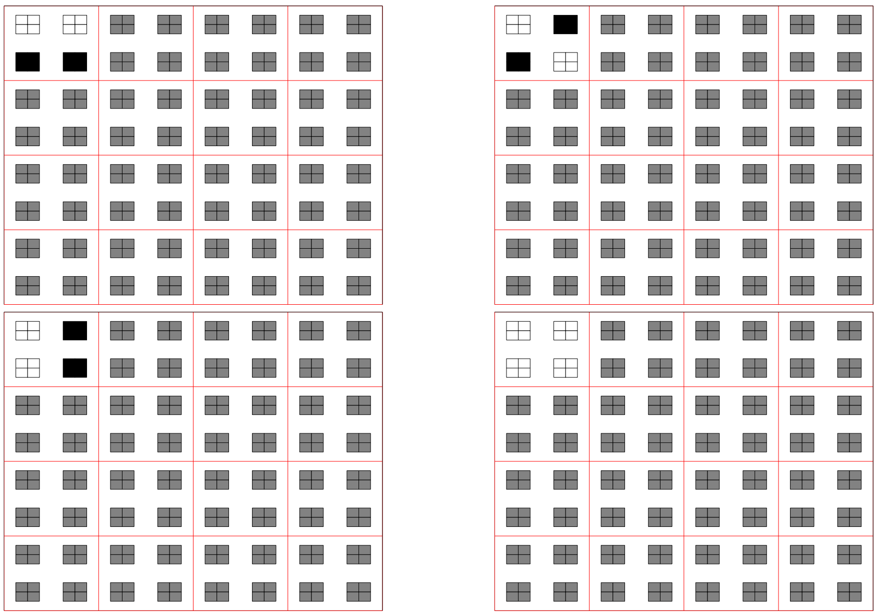

Figure 3-2: Transformed basis voltage functions. White = 1 V, gray = 0 V, black = -1 V.

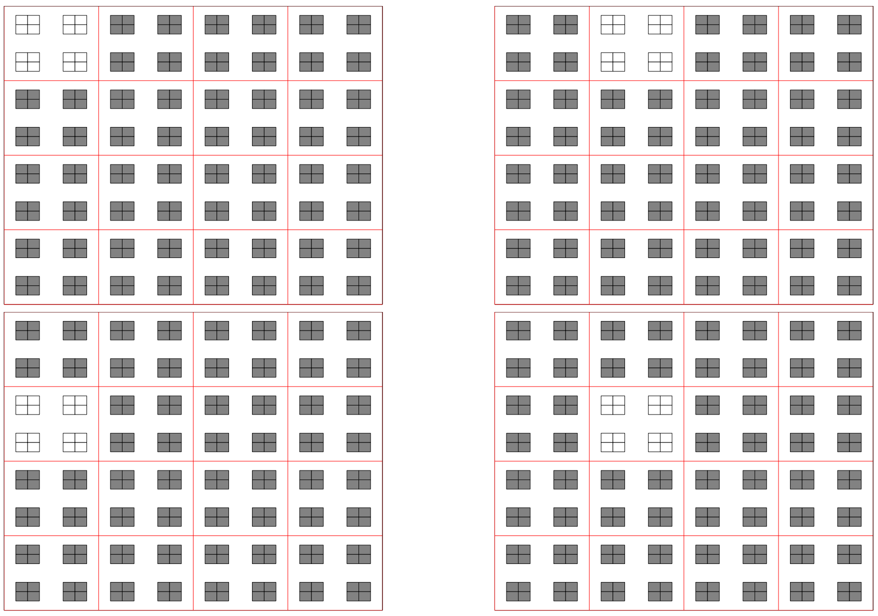

Figure 3-3: Basis functions pushed up to coarser level. White = 1 V, gray = 0 V, black = -1 V.

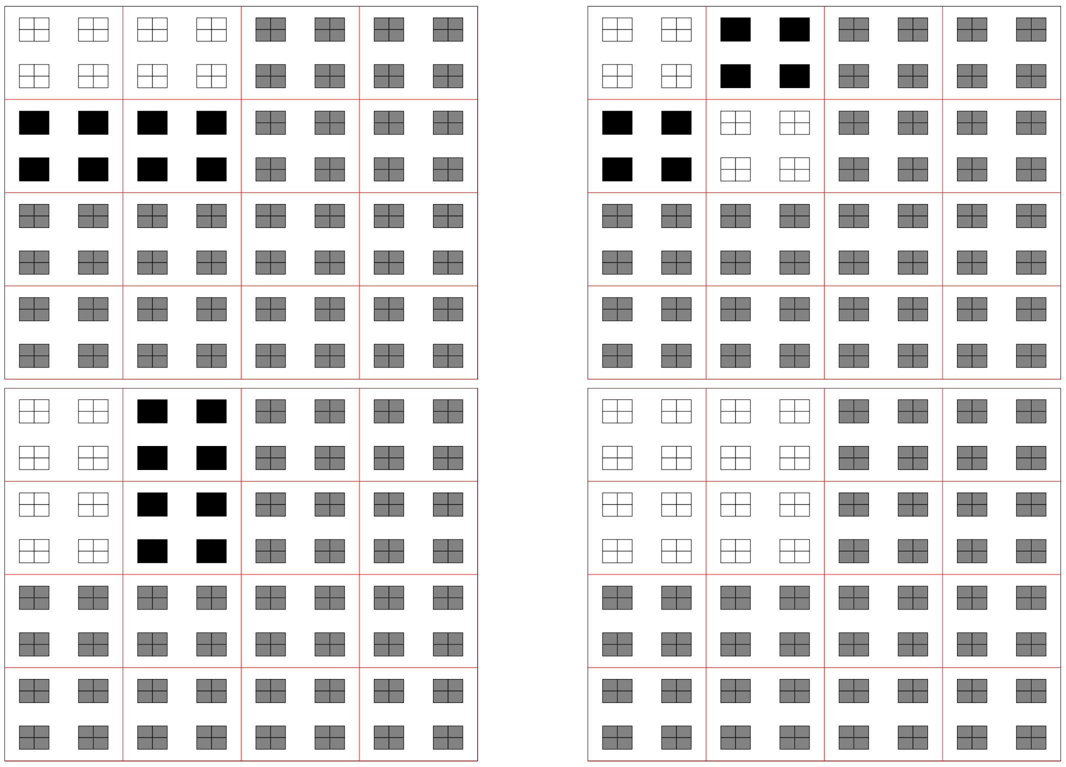
Figure 3-4: Transformed basis functions on coarser level. White = 1 V, gray = 0 V, black = -1 V.

Figure 3-4. Just as on the finest level, three have the balanced-voltage property and one is left over. If this process is continued through the levels (just one more level in our case, the coarsest, since the division into squares is just one square containing the whole substrate surface), all the new basis functions have the balanced-voltage property at some level, except one (the all-1 function).

We note that this simple example is essentially the same as the Haar wavelet construction described in [39]. The applications described in [39] are in signal and image processing, often as an alternative to Fourier analysis. The basic ideas of locality of support of the basis functions and the multilevel nature of the basis are the same.

We now have an idea of why entries in $GQ$ are mostly close to zero. This numerical sparsity is improved in $Q'GQ$. A column of $GQ$ is a current response to a new basis vector. From (3.4), one entry $(G_w)_{ij}$ of $G_w = Q'GQ$ is given by projecting column $i$ of $GQ$ onto the $j$th basis vector of $Q$. Since current responses *at* neighbor contacts to a faraway voltage will look similar, this projection results in near-cancellation and small entries.

## 3.2 Some definitions and notation

The generalization of the ideas described for a regular grid of contacts to an algorithm which produces $Q$ for general contact layouts is based on the idea of vanishing moments for localized basis functions. We first describe the division of the substrate surface into squares at several levels, which the multilevel algorithm relies on, and we introduce some notation.

Assume for simplicity that the top surface of the substrate is a square with side length 1. This square can be subdivided into four squares, each of half the sidelength. In general, doing $l$ levels of subdivision leads to a partition of the surface into $2^{2l}$ squares of sidelength $2^{-l}$. The set of $2^{2l}$ squares on level $l$ is denoted by $S_l$. The maximum level of refinement $L$ is chosen so that each square at level $L$ contains at most a small constant number of contacts. We assume that contacts do not cross square boundaries at any level. Splitting large contacts into many smaller ones using the finest level square boundaries may be necessary to achieve this. A square is denoted by $((i,j),l)$, where $l$ is the level and $(i,j)$ gives the $(x,y)$ position of the square, with $1 \leq x,y \leq 2^l$. The contacts are numbered $c_1,\ldots,c_n$. The union of the contacts in square $s$ is denoted by $C_s$. In particular $C_{((1,1),0)}$ is the union of all the contacts and will be denoted by $C$.

### 3.2.1 Moments

The idea of "balancing" voltages for neigboring contacts of equal size can be generalized to require the contact area-weighted average voltage to be zero for (most) new basis functions. The *zeroth-order moment* $\mu_{0,0,s}$ of a basis function $\sigma$ in square $s$ is defined by

$$\mu_{0,0,s}(\sigma) = \int_{C_s} \sigma(x,y)dx\,dy. \tag{3.5}$$

Requiring $\mu_{0,0,s}(\sigma) = 0$ is exactly the requirement that the area-weighted average voltage *on the contacts* is 0. It is important to note that the integration of voltage is done over the contact area in square $s$ only, not over the whole area of square $s$. The reason for this is simply that we don't immediately know the voltages in the non-contact areas from voltages on the contacts. (In particular, it is not true that the voltage outside the contacts is 0. On the other hand, the surface current density outside the contacts *is* 0.)

Even faster decay of current response can be achieved by imposing more constraints. In particular, we choose a parameter $p$ and require all moments of order $\leq p$ to vanish, not just the zeroth-order moment. The $(\alpha, \beta)$ moment of a function $\sigma$ in square $s$ is defined by

$$\mu_{\alpha,\beta,s}(\sigma) = \int_{C_s} x'^{\alpha} y'^{\beta} \sigma(x,y) dx\, dy \tag{3.6}$$

where $(x', y') = (x, y) - \text{centroid}(s)$. The *order* of $\mu_{\alpha,\beta,s}$ is $\alpha + \beta$. Hence,

$$\text{number of moments of order } \leq p = \sum_{\alpha=0}^{p} (p - \alpha + 1) = \frac{(p+1)(p+2)}{2}. \tag{3.7}$$

That faster decay is achieved by increasing $p$ is provably true for sparsifying $1/r$ matrices—see [22] for details. We found $p = 2$ to be effective in our experiments.

A vector $v$ of voltages is just a vector in $\mathcal{R}^n$. This vector is associated with a voltage function on the contacts in the obvious way by the function $f$ defined by

$$f(v) = \sum_{i=1}^{n} v_i \chi_i, \tag{3.8}$$

where $\chi_i : C \to \mathcal{R}$ is the characteristic function of the $i$th contact; that is, $\chi_i(r) = 1$ for $r \in c_i$ and $\chi_i(r) = 0$ for $r \notin c_i$.

When we refer to "moments of a vector", this is just a shorthand for moments of the associated voltage function. Also, we introduce some notation for indexing characteristic functions. $\chi_{s,i}$ will refer to the characteristic function of the $i^{\text{th}}$ contact in square $s$. We can use the same notation for standard basis vectors, i.e. $e_{s,i}$ refers to the standard basis vector corresponding to the $i^{\text{th}}$ contact in square $s$.

## 3.3 Multilevel structure

We start with $\mathcal{R}^n$ and perform an orthogonal decomposition at each level as indicated below. The space $\mathcal{W}^{(i)}$ is spanned by the "fast-decaying" basis vectors on level $i$ (i.e., their associated voltage functions have fast-decaying current response, while the space $\mathcal{V}^{(i)}$ is spanned by the "leftovers", like the voltage vectors whose associated functions are shown in our example in Figure 3-3, which must be pushed to the next level.

$$\begin{aligned} \mathcal{R}^n &= \mathcal{W}^{(L)} \oplus \mathcal{V}^{(L)} \\ \mathcal{V}^{(L)} &= \mathcal{W}^{(L-1)} \oplus \mathcal{V}^{(L-1)} \\ &\vdots \\ \mathcal{V}^{(1)} &= \mathcal{W}^{(0)} \oplus \mathcal{V}^{(0)} \end{aligned} \tag{3.9}$$

Only vectors in the $\mathcal{W}^{(i)}$ are included in the new basis, with the exception of the coarsest level space $\mathcal{V}^{(0)}$. This corresponds to

$$\mathcal{R}^n = \mathcal{W}^{(L)} \oplus \mathcal{W}^{(L-1)} \oplus \ldots \oplus \mathcal{W}^{(0)} \oplus \mathcal{V}^{(0)}, \tag{3.10}$$

which can be seen by applying the equations (3.9) in succession.

This gives a high-level view. Specifically, the space $\mathcal{W}^{(i)}$ is defined to be the subspace of $\mathcal{V}^{(i+1)}$ of vectors for which *all* moments up to order $p$ in squares at level $i$ vanish:

$$\mathcal{W}^{(i)} = \{v \in \mathcal{V}^{(i+1)} : \mu_{\alpha,\beta,s}(v) = 0 \text{ for all } \alpha + \beta \leq p, s \in S_i.\}$$

The space $\mathcal{V}^{(i)}$ is defined to be the orthogonal complement of $\mathcal{W}^{(i)}$ in $\mathcal{V}^{(i+1)}$, leading to the structure of (3.9). Because any vector $w$ in $\mathcal{W}^{(i)}$ whose $\mu_{\alpha,\beta,s}(w)$ vanish for all $s$ can be written as a sum of vectors, each with support in exactly one square, we have

$$\mathcal{V}^{(i)} = \oplus_{s \in S_i} \mathcal{V}_s \tag{3.11}$$

$$\mathcal{W}^{(i)} = \oplus_{s \in S_i} \mathcal{W}_s \tag{3.12}$$

where $\mathcal{V}_s$ is defined to be the subspace of $\mathcal{V}^{(i)}$ consisting of vectors which are zero outside square $s$, and $\mathcal{W}_s$ is defined similarly.

The key to the multilevel construction is the relationship between these individual-square subspaces on successive levels. Specifically, fix a parent square $p$ on level $i$, and its four child squares $s_1 \dots s_4$ on the finer level $i+1$. Then, since $\mathcal{W}^{(i)}$ is a subspace of $\mathcal{V}^{(i+1)}$ and $\mathcal{V}^{(i)}$ is the orthogonal complement of $\mathcal{W}^{(i)}$ in $\mathcal{V}^{(i+1)}$, it is clear that

$$\mathcal{W}_p \oplus \mathcal{V}_p = \mathcal{V}_{s_1} \oplus \mathcal{V}_{s_2} \oplus \mathcal{V}_{s_3} \oplus \mathcal{V}_{s_4}. \tag{3.13}$$

We will now show how to construct bases for the spaces $\mathcal{W}_s^{(L)}, \mathcal{V}_s^{(L)}$, that is, the finest-level spaces. Then we will use (3.13) to construct bases for the coarser-level spaces.

## 3.4 Basis construction

Just as in our definitions above, the construction of the multilevel basis proceeds level-by-level, starting at the finest level.

### 3.4.1 Finest level

We need to form a basis for $\mathcal{W}^{(L)}$ and a basis for its orthogonal complement $\mathcal{V}^{(L)}$, and we do this by forming bases $W_s$ and $V_s$ for each square $s$ at level $L$. For convenience of notation, we work with vectors expressed in the standard basis of square $s$: that is, we write $v = \begin{pmatrix} v_1 & \dots & v_{n_s} \end{pmatrix}'$ for the voltage function $v_1\chi_{s,1} + \dots + v_{n_s}\chi_{s,n_s}$. (For any given finest-level square $s$, $n_s$ is defined simply as the number of contacts in square $s$.) These can be zero-padded to make whole-substrate standard basis vectors. In an efficient implementation, the zeroes will not be explicitly stored—$Q$ will be stored as a sparse matrix.

We first form the $(p+1)(p+2)/2 \times n_s$ matrix of moments $M_s$ whose entries are given by

$$(M_s)_{(\alpha,\beta),j} = \mu_{\alpha,\beta,s}(e_{s,j}).$$

The columns of $M_s$ are just the moments of the square-$s$ standard basis vectors $e_1 \dots e_{n_s}$ each representing 1 volt on one contact in square $s$ and 0 volts on all other contacts in square $s$. The goal is to get $w_s$ basis vectors for $\mathcal{W}_s$ (square-$s$ vanishing moments vectors) and $v_s$ basis vectors for $\mathcal{V}_s$. By linearity, $M_s v = 0$ exactly when the first $(p+1)(p+2)/2$ moments of $v$ vanish. We will put the basis for $\mathcal{W}_s$ in columns of $W_s$ and the basis for $\mathcal{V}_s$ in columns of $V_s$. Then we want

$$\begin{pmatrix} \quad M_s \quad \end{pmatrix} \begin{pmatrix} \\ V_s \quad W_s \\ \\ \end{pmatrix} = \begin{pmatrix} \text{non-0} & 0 \\ \text{columns} & \end{pmatrix} \tag{3.14}$$

In (3.14), $V_s$ is $n_s \times v_s$ and $W_s$ is $n_s \times w_s$. One way to achieve this is to take a singular value decomposition

$$M_s = U_s \Sigma_s Q_s',$$

or

$$M_s = (U_s) \begin{pmatrix} \Sigma_s^0 & 0 \end{pmatrix} \begin{pmatrix} V_s' \\ W_s' \end{pmatrix} \tag{3.15}$$

where $\Sigma_s^0$ is the matrix whose columns contain the *nonzero* singular values of $M_s$. $v_s$ is defined to be the number of nonzero singular values, and $w_s := n_s - v_s$. $V_s$ is taken to be the first $v_s$ columns of $Q_s$ and $W_s$ the remaining $w_s$ columns. (It is possible for $W_s$ to be an empty matrix, i.e. $w_s = 0$, as happens when $(p+1)(p+2)/2 > n_s$. This is not a case that needs any special treatment.) Thus the SVD of $M_s$ *defines* the bases $V_s$ and $W_s$. Substituting (3.15) into the left-hand side of (3.14) shows that the equation (3.14) is satisfied, which was what we needed. (To see this, note that the columns of $W_s$ are orthogonal to those of $V_s$ in (3.15) because $Q_s$ is orthogonal by definition of the SVD.)

An interesting property of this construction, which is important in analyzing the sparsification performance of wavelets, is that $v_s \leq (p+1)(p+2)/2$. This can be seen from the fact that the $v_s$ columns of $\Sigma_s^0$ each contain a nonzero singular value in the diagonal position, so the number of columns ($v_s$) of $\Sigma_s^0$ is less than or equal to the number of rows $(p+1)(p+2)/2$.

### 3.4.2 Coarser levels

Assume that the multilevel construction has been carried out from level $L$ through coarser levels to level $i+1$ inclusive. We now express vectors in the standard basis of the square $p$ on level $i+1$. The square $p$ is the parent of child squares $s_1, \ldots, s_4$, and in order to satisfy the relation (3.13), we need to take the vectors in $V_{s_1} \ldots V_{s_4}$, now expressed in the standard basis of $p$, and recombine them to form the bases $V_p$ and $W_p$ in the parent square. Collecting the $V_{s_j}$ vectors from the four child squares results in a matrix, denoted $V_{p(\text{children})}$, defined by

$$V_{p(\text{children})} = \begin{pmatrix} V_{s_1} & 0 & 0 & 0 \\ 0 & V_{s_2} & 0 & 0 \\ 0 & 0 & V_{s_3} & 0 \\ 0 & 0 & 0 & V_{s_4}. \end{pmatrix}$$

Just as we took the SVD of $M_s$ on the finest level to split the basis into $V_s$ and $W_s$, on coarser levels we will take the SVD of the moments of vectors in $V_{p(\text{children})}$, i.e. of $M_p V_{p(\text{children})}$. This could be accomplished simply by computing $M_p V_{p(\text{children})}$ and taking the SVD, but it can be done more efficiently using moments available from previous levels. The idea is that from the SVD calculation on the finer level in square $s_i$, we use the relation $M_{s_i} = U_{s_i} \Sigma_{s_i} Q'_{s_i}$, which gives the moments of basis vectors in $Q_{s_i}$ by multiplying both sides by $Q_{s_i}$ to get $M_{s_i} Q_{s_i} = U_{s_i} \Sigma_{s_i}$ and

$$M_{s_i} V_{s_i} = U_{s_i} \Sigma_{s_i}^0.$$

The only issue is that $M_{s_i}$ calculates moments using a different center from $M_p$. But the moments in the new center are related to those in the old center by a $(p+1)(p+2)/2 \times (p+1)(p+2)/2$ matrix which can be calculated by expanding out $(x-x_0)^\alpha(y-y_0)^\beta$ for a shift of $(x_0, y_0)$.

We form the bases $V_p$ for $\mathcal{V}_p$ and $W_p$ for $\mathcal{W}_p$ by an orthogonal transformation of the vectors in $V_{p(\text{children})}$. That is, we define $V_p = V_{p(\text{children})}T_p$ and $W_p = V_{p(\text{children})}R_p$, where $Q_p = \begin{pmatrix} T_p & R_p \end{pmatrix}$ is orthogonal and defined by an SVD:

$$M_p V_{p(\text{children})} = (U)\begin{pmatrix} \Sigma_p^0 & 0 \end{pmatrix}\begin{pmatrix} T_p' \\ R_p' \end{pmatrix} \tag{3.16}$$

The goal is for the moments to vanish for vectors in $V_{p\text{children}}R_p$ but not those in $V_{p\text{children}}T_p$. This is achieved, as seen by substituting (3.16) into $M_p V_{p(\text{children})}Q_p$:

$$(M_p)(V_{p(\text{children})})\begin{pmatrix} T_p & R_p \end{pmatrix} = \begin{pmatrix} U\Sigma_p^0 & 0 \end{pmatrix}. \tag{3.17}$$

Notice that the SVDs taken are always of small-constant size matrices, since the number of columns of each of the four $V_{s_i}$ is $\leq (p+1)(p+2)/2$, so $M_p V_{p(\text{children})}$ is a $(p+1)(p+2)/2 \times 2(p+1)(p+2)$ matrix at most. Also, the new vectors $V_p$ and $W_p$ don't need to be stored explicitly if we are only interested in *applying* $Q$ rather than having the actual $Q$ matrix—it is enough to keep the transformation matrices $T_p$ and $R_p$ for each square $p$ on each level, along with the $V_s$ and $W_s$ for each finest-level square $s$.

### 3.4.3 Complexity analysis: sparsity of $Q$ for regular contact layouts

Here we show, for a simple case, what level of sparsity can be expected in the constructed $Q$. Specifically, we assume that the $2^{2L}$ finest-level squares are each populated with a *constant* number $c$ of contacts, which is greater than the number $d = (p+1)(p+2)/2$ of moment constraints. So the total number of contacts $n$ is $2^{2L}c$

and $Q$ is $n \times n$. The columns of $Q$ are exactly zero-padded columns of $W_s$ for each square at each level along with zero-padded columns of $V_{((1,1),0)}$ (the nonvanishing-moments basis vectors on level 0 in a square which is the whole substrate). (The zero-padding just takes an $n_s$-length vector, defined on the contacts in square $s$ only, to an $n$-length vector defined on all the contacts.) So we can determine the sparsity of $Q$ by looking at the number of columns, and nonzeros per column, in each $W_s$ and $V_{((1,1),0)}$.

We mentioned in the discussion of the construction of $V_s$ and $W_s$ that there are at most $d = (p+1)(p+2)/2$ vectors in $V_s$ for any square at any level. On the finest level, in each square there are generally $d$ columns in $V_s$, and thus $c - d$ columns in $W_s$ for each square $s$, because the total number of columns in $V_s$ and $W_s$ is the number of contacts per square $c$. The reason the total number of columns is always $c$ is that the vectors in $V_s$ and $W_s$ together form a basis for the dimension-$c$ space of square-$s$ voltage vectors. There may be fewer than $d$ columns in $V_s$ (though this is very unlikely).

We count the finest-level nonzeros first. There are $2^{2L}$ finest-level squares. Any given square $s$ has at most $c$ columns in $W_s$, and because each column of $W_s$ has support only in square $s$, the total number of nonzeros for each column of $W_s$ is at most $c$. This gives at most $2^{2L}c^2$ nonzeros in $Q$ from finest-level vanishing-moments basis vectors.

Now consider the coarser level squares for a given level $i < L$. For a coarser level (parent) square $p$, the number of basis vectors in $V_p$ and $W_p$ together equals the number of basis vectors in the child-square spaces $V_{s1}, \dots, V_{s4}$ (this follows from (3.13)). Because $V_t$ can have at most $d$ vectors for any square $t$ on any level, there are at most $4d$ basis vectors total in $V_p$ and $W_p$, so certainly at most $4d$ basis vectors in $W_p$. Since each vector in $W_p$ has support only in the level-$i$ square $p$, there are at most $c2^{2(L-i)}$ nonzeros per column of $W_p$. Since there are $2^{2i}$ squares at level $i$, the total number of nonzero level-$i$ entries is at most

$$4d \cdot 2^{2i} \cdot c2^{2(L-i)} = 4cd2^{2L}.$$

On the coarsest level, there are at most $d$ vectors in $V_{((1,1),0)}$, each of which may have $n = 2^{2L}c$ nonzeros, so this contributes $cd2^{2L}$ nonzeros. Thus, each level has at most a constant times $2^{2L}$ nonzeros contributed at that level, and there are $L+1$ levels. But $L+1 = O(\log n)$, and $2^{2L} = O(n)$, so there are at most $O(n \log n)$ nonzeros in $Q$. This of course means that $Q$ can be applied to a vector in $O(n \log n)$ operations. Because $Q$ is orthogonal, its inverse is $Q'$ and can also be applied in $O(n \log n)$ operations.

We mentioned earlier that a more efficient representation can be obtained if we don't need the explicit matrix $Q$, just a way to apply $Q$. This can be done by representing $Q = Q^{(L)}Q^{(L-1)} \dots Q^{(0)}$, where the columns of $Q^{(L)}$ are the (zero-padded) columns of $V_s$ and $W_s$ for all the finest-level squares $s$. There are at most $c$ nonzeros per column in $Q^{(L)}$, for a total of at most $cn = c^2 \cdot 2^{2L}$ nonzeros. For any given coarser level $i$, $Q^{(i)}$ has, for each square $p$ at level $i$, the transformation matrices $T_p$ and $R_p$ which re-combine vectors from the child square bases $V_{s_1} \dots V_{s_4}$ of $p$. That is, $\begin{pmatrix} T_p & R_p \end{pmatrix}$ is, for any given $p$, a square matrix of size at most $4d \times 4d$. So the total number of entries in any $\begin{pmatrix} T_p & R_p \end{pmatrix}$ block is bounded by the constant $16d^2$, and there are $2^{2i}$ such transformation blocks in $Q^{(i)}$, one for each level-$i$ square $p$. This gives a total of $16d^2 \cdot 2^{2i}$ nonzero entries for $Q^{(i)}$. We then have

$$\begin{aligned} \text{total \# nonzeros} &\leq c^2 \cdot 2^{2L} + 16d^2(2^{2(L-1)} + 2^{2(L-2)} + \cdots + 1) \\ &\leq 16c^2(2^{2L} + 2^{2(L-1)} + \cdots + 1) \\ &\leq 16c^2 \cdot \frac{4}{3}(2^{2L}) \end{aligned} \tag{3.18}$$

From (3.18) and $n = c \cdot 2^{2L}$ we see that the number of nonzeros in $Q^{(L)}$ plus the number of nonzeros in the transformation-matrix blocks of the $Q^{(i)}$ for $i < L$ is $O(n)$. (Note that we didn't count diagonal identity-matrix blocks in the $Q^{(i)}$, since these do not need to be explicitly stored and applying these blocks to vectors is an empty operation.)

We conclude that storing $Q^{(L)}$ and the non-identity parts of the $Q^{(i)}$ for $i < L$ gives an $O(n)$ method for applying $Q$. This immediately gives a method for applying

$Q^{-1} = Q'$ in $O(n)$ operations as well since

$$Q' = Q^{(0)'} Q^{(1)'} \cdots Q^{(L)'}.$$

## 3.5 Reducing the number of solves

In the last section, we constructed the change-of-basis matrix $Q$. We have seen that it can be used to get a sparse representation of $G$ by setting $G_w = Q'GQ$, truncating small entries of $G_w$ to get a sparse approximation $G_{ws}$ of $G_w$, and approximating $G \approx QG_{ws}Q'$, which can be applied using three sparse matrix-vector products. However, this assumes we have $G$ available, which would require $n$ black-box solver calls, one per column, and is an $O(n^2)$ algorithm (just to read the entries of $G$) in any case. Here we give a method for obtaining a sparse representation of $G$ in the same form $G \approx QG_{ws}Q'$, but with a very small number of black-box solver calls ($O(\log n)$).

The basic idea for obtaining $A$, given a black box which outputs $Av$ given $v$, is to exploit a priori knowledge (or assumptions) about the sparsity structure of $A$. Instead of calculating $Ae_1, \ldots, Ae_n$ to obtain $A$, we may choose a set $S$ of several basis vectors and calculate the response $A(\sum_{v\in S} v) = \sum_{v \in S} Av$. If, for any given distinct vectors $v$ and $w$ in $S$, the nonzeros of $Av$ and $Aw$ don't overlap, then $Av$ can be extracted from $\sum_{v\in S} Av$ for each $v \in S$. To see how this might work, suppose we know *a priori* that $A$ is tridiagonal. Take the simple example of a $7 \times 7$ $A$. We can apply $A$ to $e_1 + e_4 + e_7$ to obtain

$$\begin{bmatrix} a_{11} & a_{12} & & & & & \\ a_{21} & a_{22} & a_{23} & & & & \\ & a_{32} & a_{33} & a_{34} & & & \\ & & a_{43} & a_{44} & a_{45} & & \\ & & & a_{54} & a_{55} & a_{56} & \\ & & & & a_{65} & a_{66} & a_{67} \\ & & & & & a_{76} & a_{77} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} \\ a_{12} \\ a_{34} \\ a_{44} \\ a_{54} \\ a_{67} \\ a_{77} \end{bmatrix} \qquad (3.19)$$

Thus every third column of $A$ starting from the first can be obtained with just one application of $A$. Similarly, with $A(e_2 + e_5)$ we can find every third column of $A$ starting from the second, and with $A(e_3+e_6)$ we find every third column of $A$ starting from the third. For any tridiagonal $A$, three matrix-vector products suffice to obtain $A$.

Of course, the sparsity structure in our case is much more complicated. In fact, $G_w = QGQ'$ is not sparse at all. It does, however, have many entries which we expect to be so small that dropping them will not compromise accuracy very much. So we need to specify our assumption on which entries of $G_w$ are small. The entries of $G_w$ are mostly of the form

$$W_{((k',r'),l')}(:,m')GW_{((k,r),l)}(:,m) \tag{3.20}$$

and there are also the special cases involving the finest-level nonvanishers

$$\begin{aligned} V_{((1,1),0)}(:,m')GW_{((k,r),l)}(:,m), \quad & (3.21)\\ W_{((k',r'),l')}(:,m')GV_{((1,1),0)}(:,m), \quad & (3.22)\\ V_{((1,1),0)}(:,m')GV_{((1,1),0)}(:,m). \quad & (3.23) \end{aligned}$$

First we deal with these special cases. In our assumptions, *none* of the entries of these forms are assumed to be small. All the entries of the forms (3.22) and (3.23) can be found by applying $G$ to the small constant number of columns of $V_{((1,1),0)}$. The entries of the form (3.21) can be found by symmetry of $G$ from the entries of the form (3.22): we have

$$\begin{aligned} V_{((1,1),0)}(:,m')GW_{((k,r),l)}(:,m) &= (V_{((1,1),0)}(:,m')GW_{((k,r),l)}(:,m))' \\ &= W_{((k,r),l)}(:,m)'G'V_{((1,1),0)}(:,m') \\ &= W_{((k,r),l)}(:,m)'GV_{((1,1),0)}(:,m') \end{aligned}$$

We still need to specify our assumption for when entries of the form (3.20) are small

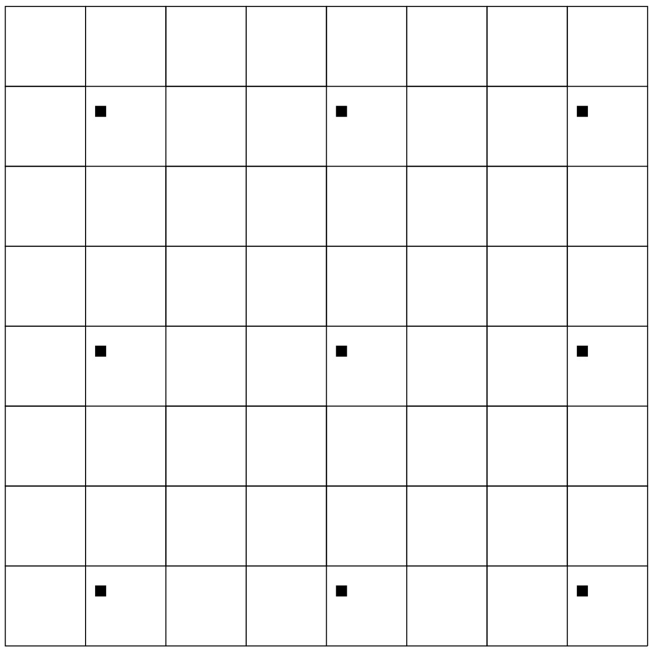

Figure 3-5: Schematic representation of basis vector constituents of voltage vector for solve-reduction technique. Each basis vector is represented by a black square. Note that neighbor squares of distinct basis vector squares do not overlap.

enough to ignore. The definition is that basis vectors in two squares are considered to have a small interaction if the squares are well-separated. We define well-separated in a conservative way. Assume without loss of generality that $l \leq l'$ (the case $l' \leq l$ is defined symmetrically). Consider a square $s$ on level $l$ and a square $s'$ on level $l'$. Let $p'$ be the ancestor square of square $s'$ on level $l$. Then $s$ and $s'$ are defined to be well-separated if $s$ and $p'$ are not the same square or neighboring squares. So entries are not assumed small exactly when $s$ and $p'$ are the same or neighbors. We will use the word "local" for this "same or neighbors" situation for two squares.

Now we are ready to define the sums of vectors to which we apply $G$. To provide the needed separation, we only combine basis vectors which are on the same level and which are at least 3 squares apart. See Figure 3-5. Then we have vectors for each level $l$, $i = 0 \ldots 2$, $j = 0 \ldots 2$, $m = 0 .. \max_{s \in S_l} w_s$ given by

$$\theta_{((i,j),m),l} := \sum_{(k,r)=(i \bmod 3, j \bmod 3)}^{1 \leq k,r \leq 2^l} W_{((k,r),l)}(:,m). \tag{3.24}$$

Since the number of vanishing-moments basis vectors may differ from square to square on a given level, there may be some values of $m$ for which some squares do not have vanishing-moments basis vectors—that is, the number of columns in $W_{((k,r),l)}$ is $< m$. In this case, $W_{((k,r),l)}(:, m)$ is simply defined to be the 0 vector.

Now, we claim that from these vectors $\theta_{((i,j),m),l}$, we can extract all entries of the form (3.20), if we also assume $l \leq l'$. This assumption creates no problems, since the entries with $l > l'$ can be filled in by symmetry of $G_w$. We call the approximate version of $G_w$ created by this technique $G_{ws}$.

Notice first that for any $k$, $r$, $l$, and $m$, $W_{((k,r),l)}(:, m)$ is included in exactly one $\theta_{((i,j),m),l}$. By our assumption on small entries, we only need to consider vectors $W_{((k',r'),l')}(:, m')$ such that the ancestor square $p'$ on level $l$ of $((k', r'), l')$ is local to $((k, r), l)$ (other entries are small by assumption). Then, by our assumption, for any $l \leq l'$, for this $\theta_{((i,j),m),l}$, we have

$$W_{((k',r'),l')} G \theta_{((i,j),m),l} \approx W_{((k',r'),l')} G W_{((k,r),l)}(:, m). \tag{3.25}$$

The reason is that $W_{((k,r),l)}$ is the same as or a neighbor of the ancestor square $p'$ of square $((k', r'), l')$. Thus no other vector in the sum defining $\theta_{((i,j),m),l}$ can be a neighbor of $p'$, because any other vector in the sum is at least 3 squares away from $((k, r), l)$ in either the $x$ or $y$ direction, and $p'$ is at most 1 square away from $((k, r), l)$, so that $p'$ must be at least 2 squares away from any other vector in the sum, that is, not local to it. So the other terms are small by our assumption.

This is not a rigorous argument unless "small" means exactly 0. The argument can also be made rigorous if there is a sufficiently fast guaranteed decay of the "small" entries with distance of the squares. Though we don't have any such guarantee, our results later indicate that the method is effective in practice.

### 3.5.1 Complexity analysis: number of solves required for regular contact layouts

It is clear that the combine-solves technique reduces the number of solves required from the $n$ required by the naive method. To get a handle on how large this reduction is, we count the number of solves required for a simple regular-layout case. We make the same assumptions as for our discussion of the sparsity of $Q$: $c > d$ contacts per finest-level square, where $d$ is the number of moment constraints.

Looking at (3.24), this is easy to do. There are 3 choices for $i$ and $j$ (0,1,2), at most $\max(c, 4d)$ values of $m$ ($m \leq$ the maximum number of basis vectors in $W_{((k,r),l)}$ for all $k$,$r$,$l$) at any level, and $O(\log n)$ levels. The number of choices for $(i, j, m)$ is thus a constant, so the total number of choices is $O(\log n)$. The number of solves required has been reduced from $n$ to $O(\log n)$! While there is no guarantee that this will hold for very irregular contact layouts, we have found very substantial reductions in the number of solves in practice.

Finally, we note that alternative approaches to reducing the number of solves may be useful. In particular, the methods described in [40] could be applied to the substrate coupling situation to greatly reduce the number of variables in the problem when the voltages have support in only one square at a sufficiently fine level. Then, instead of reducing the number of solves, we gain efficiency because each solve is less costly. This method, however, has the disadvantage that it requires the use of an underlying substrate solver based on the techniques of [40], whereas we can use arbitrary solvers.

## 3.6 Complexity analysis: nonzeros in $G_{ws}$ for regular contact layouts

The assumption on which entries are zeroed out makes it possible to analyze the number of nonzeros in $G_{ws}$ formed using the wavelet basis and the combine-solves technique. We again use the regular-layout assumption. There are at most $d$ columns in $V_{((1,1),0)}$. The expressions (3.22) and (3.23) are the interactions of these $d$ vectors with all $n$ basis vectors in $Q$, so this contributes at most $dn$ nonzeros. Expression (3.21) is the interaction of $d$ vectors with fewer than $n$ basis vectors in $Q$, so this also contributes at most $dn$ nonzeros.

For the $W$-$W$ interactions, we use the fact that there are at most $\max(c, 4d)$ vectors in $W_{((k,r),l)}$ for any square on any level. Call this constant $C$. How many entries of the form (3.20) are there if we require $l' \leq l$? For a given level $l$, there are $2^{2l}$ squares on that level, each with $\leq C$ vanishing-moments basis vectors. Fix one such square $s$. Responses to $W_s$ on level $l'$ are nonzero in our approximation only in the $\leq 9$ squares local to the level-$l'$ ancestor of $s$, i.e. at $\leq 9C$ basis vectors. Since there are $\leq \log n$ levels, there are a total of $2^{2l}C \cdot 9C \log n$ nonzero entries for responses to $W$-vectors on a given level $l$. Summing this through the levels produces

$$\begin{aligned}
\text{for } l' \leq l, \text{total nonzeros} &\leq \sum_{l=0}^{L} 2^{2l} 9C^2 \log n \\
&\leq \frac{4}{3} 2^{2L} 9C^2 \log n \\
&= O(n \log n)
\end{aligned}$$

Including the $l < l'$ entries at most doubles the number of nonzeros (since $G_{ws}$ is symmetric by construction). Under our assumptions, there are $O(n \log n)$ nonzeros in the matrix extracted using the combine-solves technique with the $Q$ we defined.

## 3.7 Computational results

For all the examples in this section, we used a two-layer substrate with the bottom-layer conductivity 100 times the top-layer conductivity. (Actually, as we discuss briefly later, we also used a thin third layer to try to simulate the effect of having no groundplane while using an integral equation solver which requires a groundplane.) The dimensions of the substrate are 128 by 128 $(x, y)$ by 40 $(z)$. Note that we do not give units since everything scales and our error measures will not depend on the scale. The interface between the two layers is just below the top surface of the substrate, at $z = -0.5$.

The intention is to study situations where the couplings drop off slowly. For the finite-difference solver, using no backplane contact helped achieve this. The integral-equation solver we used currently requires the use of a groundplane. We were able to reduce its effects and get reasonably slow drop-off in the couplings by placing a second interface near the bottom of the substrate, at $z = -39$, and a layer below it with one-tenth the top-layer conductivity.

Our sparsification methods do not provide an exactly correct representation of $G$—there is always some loss of accuracy. Since any desired sparsity can be achieved if enough accuracy is sacrificed (the 0 matrix is sparse indeed—but not so accurate), in order to present meaningful results we need to consider sparsity and accuracy together.

For the relatively small examples we present here, it is feasible to actually calculate the exact $G$ using the naive method. Then we can look at the sparsity of the $G_{ws}$ obtained by the wavelet method with the combine-solves technique, versus the error in $G_s := QG_{ws}Q'$ compared to the exact $G$. How do we make this comparison? There are various ways—the one we have chosen is to look at relative errors in the entries of $G_s$. The relative error of an individual entry $(i,j)$ of an approximation $G\text{approx}$ to $G$ is given by

$$\text{error}(i,j) = \frac{|G\text{approx}(i,j) - G(i,j)|}{|G(i,j)|}.$$

Because it is a *relative* measure, this is an especially difficult standard to meet for small entries of $G$. However, it is reasonable to measure our results this way since small contacts (which will have small interaction entries) may be connected to sensitive circuitry for which small errors could be important.

For the results, we look at two possible ways of using the wavelet method. In the first, we drop only the entries of $G_w$ which are small according to the conservative assumption given in Section 3.5, obtaining $G_{ws}$. In this high-accuracy approach, we look at the maximum relative error of any entry. On the other hand, perhaps we want better sparsity and are willing to sacrifice some accuracy. For this, we drop

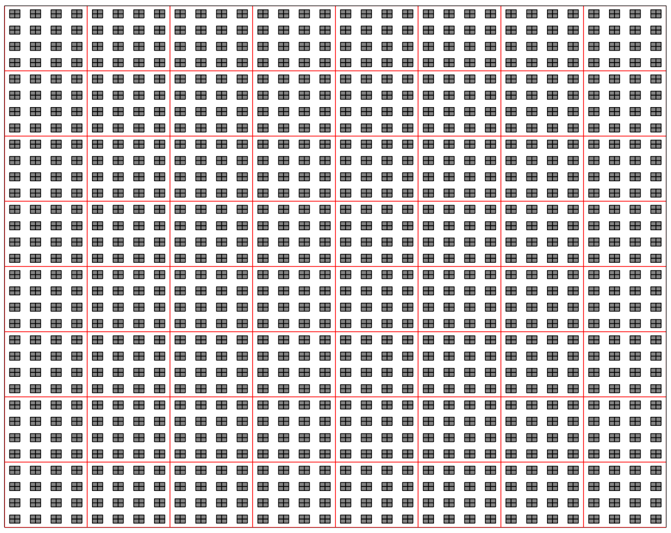

Figure 3-6: Regular contact layout

| Example | Sparsity of unthresholded $G_{ws}$ | Maximum relative error | Sparsity of thresholded $G_{wt}$ | Proportion of entries with rel. err $> 10\%$ |
|---|---|---|---|---|
| 1a | 2.5 | 0.2% | 15.3 | 0.1% |
| 1b | 2.5 | 0.2% | 15.4 | 5.2% |
| 2 | 3.5 | 0.2% | 20.6 | 1.1% |
| 3 | 2.5 | 47% | 15.3 | 80% |

Table 3.1: Sparsity and accuracy for wavelet sparsification

additional entries by choosing a threshold $t$ so that the sparsity will be approximately 6 times greater than that obtained with only the conservative assumption. We call the approximation to $G_w$ thus obtained $G_{wt}$ ($t$ for thresholded). In this case, we look at the fraction of entries whose relative error is greater than 10%.

We give results for four examples here. All examples except for Example 1b are solved using the integral-equation solver. Example 1a is a regular grid of contacts, shown in Figure 3-6. Example 1b is the same layout, but is solved using the finite-difference solver. Example 2, shown in Figure 3-7, is an irregular layout of contacts, with many large gaps between contacts, but the contacts are still all the same shape and size. Example 3, shown in Figure 3-8, is a regular layout of contacts which alternate in size. Table 3.1 gives results. The "sparsity" of a matrix is the ratio of $n^2$ (total number of entries) to the number of nonzeros. We didn't include sparsity of $Q$

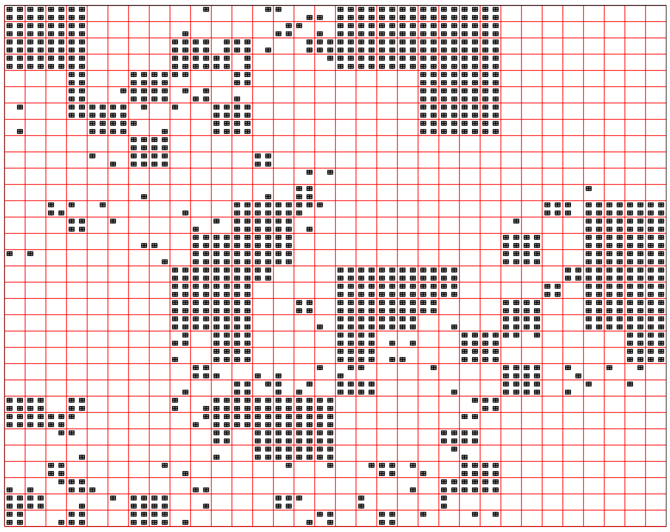
Figure 3-7: Same-size contacts, with irregular placement

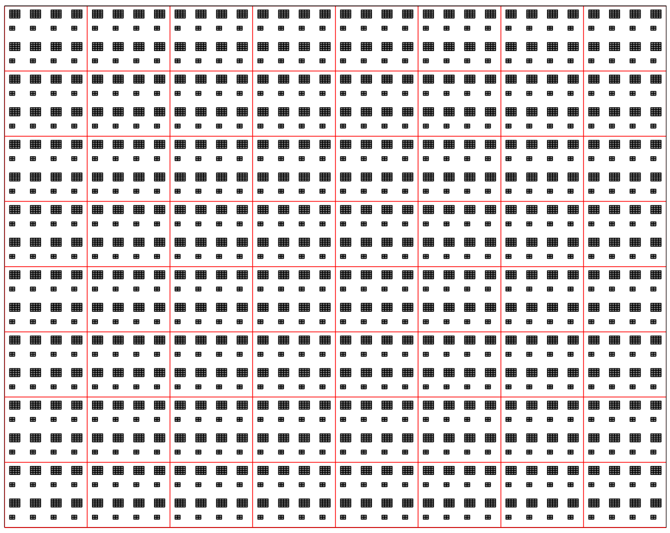
Figure 3-8: Alternating-size contact layout

in the table only because it is normally much better than the unthresholded sparsity of $G_{ws}$. For all of these examples, the sparsity of $Q$ was at least 15. Of course, a $Q$ which is not sparse would provide another way to "cheat" on the sparsity of $G_{ws}$: for example, one can take $QDQ' = G$ with a diagonal (and thus very sparse) $D$ by diagonalizing $G$, but the $Q$ will be both expensive to obtain and dense.

Examples 1a, 1b, and 2 show very good performance by the wavelet method. These all consist of many contacts of the same size and shape, although in Example 2 the spacing is irregular and there are many large gaps, which may be why it has a high fraction of high-relative-error entries compared to Example 1. Example 1b also shows a high proportion of high-relative error $G_{wt}$ entries, perhaps due to "noise" in the solution given by the comparatively inaccurate finite-difference solver. Example 3, though, shows most clearly the shortcomings of our wavelet-based approach. The main difference between it and the others is the presence of contacts of different sizes. Exactly how this becomes a problem is discussed at the beginning of Chapter 4, where a new method for sparsifying the conductance matrix is developed which we've found to be much more effective than the wavelet method on a variety of examples.

### 3.7.1 Spy plots

It is instructive to look at the Matlab sparsity structure (spy) plots for the $G_{ws}$ and $G_{wt}$ matrices. We show the $G_{ws}$ plot for Example 2 in Figure 3-9 and the $G_{wt}$ plot in Figure 3-10. The plots are 2-D pictures of the matrix entries, filled in only in positions with nonzero entries. The sparsity structure has clear origins in the multilevel structure of the wavelet construction. To understand this, we need to specify the order chosen for the transformed basis vectors. (Different orderings will lead to the same $G_{ws}, G_{wt}$ matrices with rows and columns permuted, so computational cost is not affected, but the spy picture depends on the ordering.)

The coarsest level $V_{((1,1),0)}$ basis vectors are first in the ordering. Then we continue level-by-level to finer levels. On a given level $l$, the vectors in $W_{((i,j),l)}$ are included for all squares $((i,j),l)$ on level $l$. The ordering of the squares within each level is *quadrant-hierarchical.* This means that squares in the top-left quadrant of the come

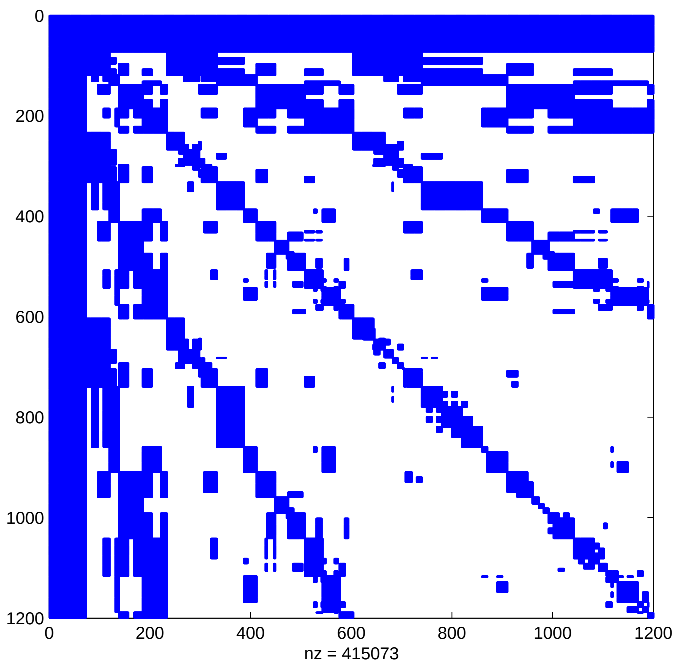

Figure 3-9: Spy plot for Example 2

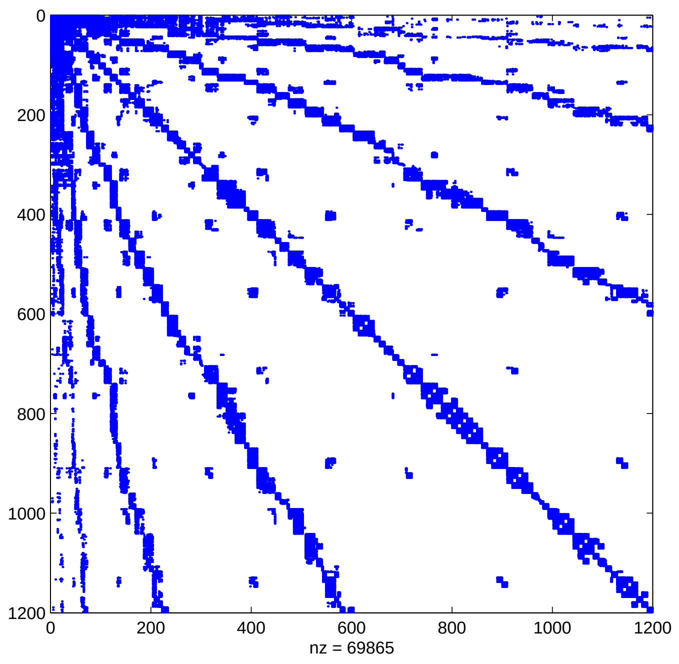

Figure 3-10: Spy plot after thresholding for Example 2

first in the ordering, followed by those in the top-right quadrant, the bottom-left quadrant, and the bottom-right quadrant in turn, and also that each quadrant is itself quadrant-hierarchically ordered. The "base case" of this recursive definition is that any single finest-level square is quadrant-hierarchically ordered.

There are several "rays" of nonzeros visible in the plot, most noticeably along the diagonal, horizontally along the top, and vertically at the left side. The diagonal ray is composed of nearby interactions of squares on the same level. The vertical and horizontal rays are interactions with everything else of the coarsest level $V$ and $W$ vectors and the second-coarsest level $W$ vectors. If we look at any one of the other rays, it represents the interaction of overlapping squares which are a fixed number of levels apart. This is somewhat oversimplified since squares which are neighbors may not be close in the quadrant-hierarchical ordering, which is responsible for the blocks which aren't on a ray.

# Chapter 4

# Low-rank methods

The method of forming the basis by multilevel geometric moment-matching, discussed in Chapter 3, uses only information about the geometry of the contact layout. No information from actually applying the operator $G$ is used in forming $Q$. The fundamental idea of the low-rank method is that we may be able to get a better $Q$ (in terms of accuracy-sparsity tradeoff of $G_{ws}$, while maintaining a sparse $Q$) by using information obtained from applying $G$. In our examples, we have found that the new low-rank approximation method is always competitive with and often far superior to the wavelet method.

First we give some intuition on why the wavelet method is inadequate in many cases, and how this can be remedied. This is followed by a presentation of the low-rank algorithm in the following sections, and finally a discussion of results. In our discussion of the new algorithm, we try to start from the simplest possible approach, and show why it isn't enough and how to fix it.

## 4.1 Some intuition

Consider the contact layout shown in Figure 4-1. If we applied the wavelet method to this layout in its simplest form $p = 0$ (matching only the zeroth-order moment), it would form a vanishing-moments basis function with support in the square $s$ containing the two gray-shaded contacts. As discussed in Chapter 3, this is done by

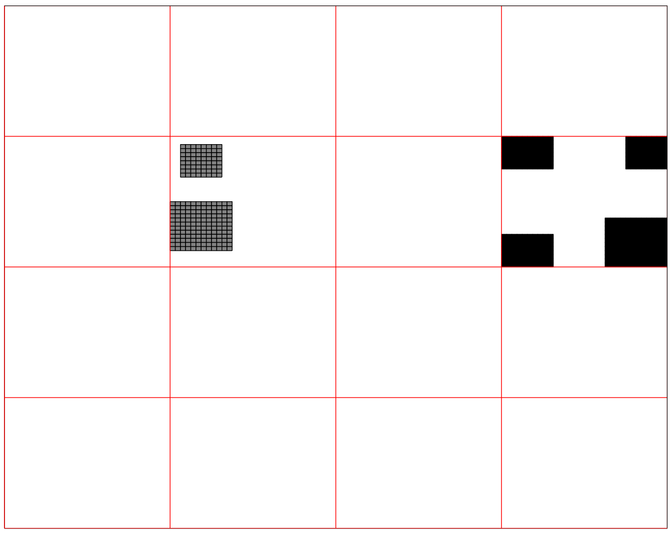

Figure 4-1: Simple example contact layout

enforcing a constraint requiring

$$\int_{C_s} \sigma(x,y)dx\,dy = 0, \tag{4.1}$$

where $\sigma(x,y)$ gives the voltage at position $(x,y)$ on the substrate surfaces and $C_s$ is the contact area in square $s$. This means the area-weighted average voltage in square $s$ must be 0. Since the larger contact (contact 2) is 2.25 times the area of the smaller contact (contact 1) in square $s$, this suggests that using the vector $\begin{pmatrix} 2.25 & -1 \end{pmatrix}'$, normalized to unit length to give $v = \begin{pmatrix} .9138 & -.4061 \end{pmatrix}$ would result in small current response at contacts 3 through 6 in square $d$ (shaded in black).

Unfortunately, this basis function's response at the four black contacts is not very small. We computed $G$ using the integral equation solver for the contact layout of Figure 4-1 with a two-layer substrate, plus a thin layer adjacent to the groundplane (substrate dimensions $64 \times 64$ (x,y) and 40 (z), conductivities 1, 100, 0.1, interfaces $z = -2$, $z = -39$). Forming the interaction matrix $G_{ds}$ giving current responses at contacts 3 through 6 from voltages at 1 and 2, for this example we get

$$G_{ds}v = \begin{pmatrix} -0.5020 & -0.9461 \\ -0.3651 & -0.6888 \\ -0.7054 & -1.3341 \\ -0.4984 & -0.9435 \end{pmatrix} \begin{pmatrix} .9138 \\ -.4061 \end{pmatrix} = \begin{pmatrix} -0.0744 \\ -0.0539 \\ -0.1028 \\ -0.0723 \end{pmatrix} \tag{4.2}$$

These are somewhat smaller than the entries of $G_{ds}$, but we can do much better. To see how, observe by inspecting $G_{ds}$ that its second column is very close to a multiple of its first:

$$G_{ds}(:,2)./G_{ds}(:,1) = \begin{pmatrix} 1.8848 \\ 1.8864 \\ 1.8914 \\ 1.8928 \end{pmatrix} \tag{4.3}$$

That is, the current responses at contacts 3 through 6 to unit voltage on contact 2 are very close to 1.89 times the current response at 3 through 6 to unit voltage on contact 1. This suggests taking $\begin{pmatrix} 1.89 & -1 \end{pmatrix}'$, normalizing it to unit length to give $\begin{pmatrix} .8839 & -.4677 \end{pmatrix}'$, and using this as our basis function. In fact, doing this gives $G_{ds}v = \begin{pmatrix} -.0012 & -.0006 & .0005 & .0007 \end{pmatrix}'$, a much smaller faraway current response.

We of course need a method more automated than "eyeballing" $G_{ds}$ to find a basis function with small faraway response. One such method is to factor $G_{ds}$ using the reduced SVD:

$$G_{ds} = U\Sigma V' = \begin{pmatrix} -0.4710 & -0.7231 \\ -0.3428 & -0.3529 \\ -0.6637 & 0.3634 \\ -0.4692 & 0.4697 \end{pmatrix} \begin{pmatrix} 2.2740 & 0 \\ 0 & 0.0016 \end{pmatrix} \begin{pmatrix} 0.4677 & 0.8839 \\ 0.8839 & -0.4677 \end{pmatrix} \tag{4.4}$$

Notice that $\sigma_{22}$ is very small. This implies that

$$G_{ds}V(:,2) = U\Sigma V'V(:,2) = U\Sigma \begin{pmatrix} 0 \\ 1 \end{pmatrix} = U \begin{pmatrix} 0 \\ \sigma_{22} \end{pmatrix} \tag{4.5}$$

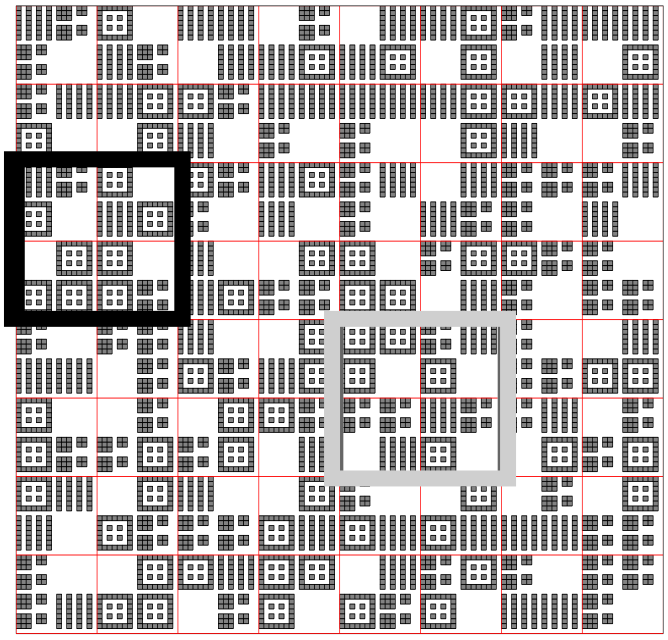

Figure 4-2: Two squares of interacting contacts

is small as well. Thus the SVD gave us a vector $V(:,2)$ with small faraway current response. We emphasize that the ability to find such a vector was critically dependent on having a very small singular value in the reduced SVD. The same idea can be applied to arbitrary-sized matrix sections.

### 4.1.1 SVDs of matrix sections

Can we generalize the preceding observations to interactions between larger sets of contacts? Consider the layout shown in Figure 4-2 with two highlighted squares, a source square $s$ (on the left, with black boundary) and a destination square $d$ (to the right of and below center, with gray boundary). The matrix section relating voltages on contacts in $s$ to currents on contacts in $d$ is denoted $G_{ds}$.

We can take the SVD of $G_{ds}$ (a 53 by 56 matrix). The singular values decay very rapidly. In contrast, if we look at $G_{ss}$, the self-interaction of square $s$ (56 by 56), there is only very slow decay in the singular values. This is shown in the semi-log plot of Figure 4-3, and is an example of a general principle: we can expect rapid decay of the singular values for interactions between two squares only when they are

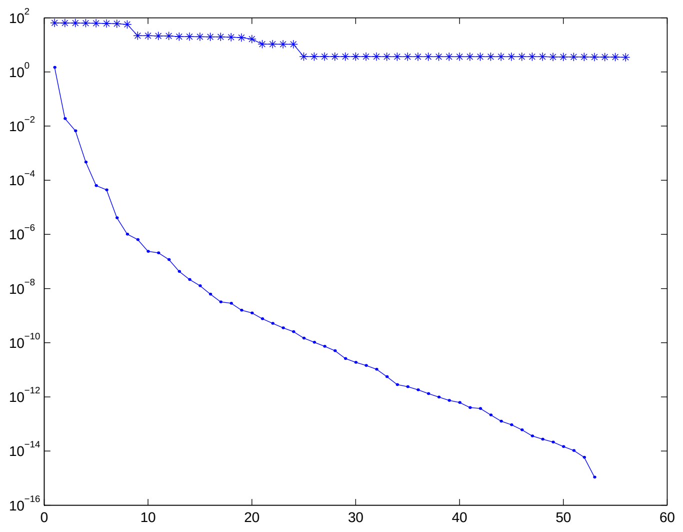

Figure 4-3: Singular values (line of stars shows square s self-interaction, line of dots shows s to d interaction)

well-separated (not the same or neighbors).

Approximating $G_{ds}$ by dropping all but the largest few singular values gives a fast method for approximately applying $G_{ds}$. Experimentally, the accuracy is very good in the fast-decay case but not in the slow-decay case. That is, the accuracy is good when $s$ and $d$ are well-separated.

If only interactions between well-separated squares on the same level as the gray and black squares of Figure 4-2 can be approximated this way, a large part of the operator is not represented. This naturally leads to the idea of a multilevel algorithm, since the "gap" of interactions which aren't represented can be reduced by including interactions between well-separated squares on finer levels.

## 4.2 The algorithm

In this section, we put these ideas and others together to design an algorithm for computing $Q$ and $G_{ws}$ such that $G \approx QG_{ws}Q'$. Like others which have been proposed for this and similar problems ([18, 41]—the work of [41] was discussed in more detail in Chapter 3), it is multilevel. We use the same definitions of levels and squares as in the wavelet chapter. 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 proceeds 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. 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-basis representation obtained in the first phase to create a transformed basis $Q$ and obtain a sparse $G_{ws}$ such that $G \approx Q G_{ws} Q'$. The second phase proceeds through the levels from finest to coarsest. This has two advantages over the multilevel row-basis representation: first, we can drop additional entries in $G_{ws}$ by thresholding to trade off accuracy for sparsity if desired, and second, this is in the same form as the representation in Chapter 3, which facilitates performance comparisons.

We now describe more precisely the goals of each phase and how they are achieved.

## 4.3 Coarse-to-fine sweep: multilevel row-basis representation

The goal of the first phase is to form a *multilevel row-basis* representation of $G$. More precisely, consider an interaction matrix $G_{I_s s}$ which is applied to voltages in square $s$ and gives currents in the *interactive squares* $I_s$ of square $s$. (The terminology of interactive and local squares is due to Greengard [12].) In general, we use the notation $G_{ba}$ to mean the operator which takes a length $n_a$ vector $v$ of the region-$a$ contact voltages and returns a length $n_b$ 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, $G_{ba} v = i$ and $G_{ba}$ has $n_a$ columns and $n_b$ rows.

The *interactive squares* $I_s$ 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

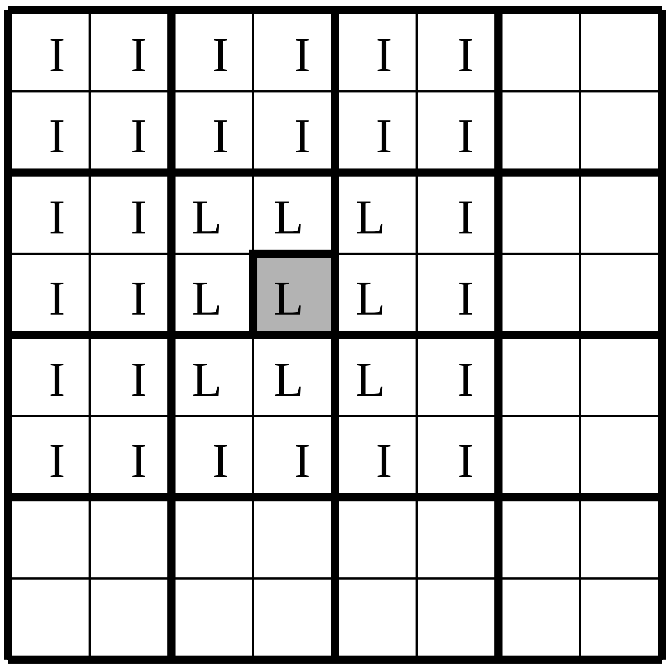

Figure 4-4: Interactive (labeled I) and local (labeled L) squares of shaded square: next-coarser level squares shown with bold lines

(adjacent or have a common corner). The *local* squares $L_s$ of a level-$l$ square are $s$ itself and its neighbors on level $l$. See Figure 4-4. Notice that interactive and local are symmetric definitions—if $d \subseteq I_s$(respectively, $L_s$), then $s \subseteq I_d$(respectively, $L_d$). In this case we say $s$ and $d$ are interactive (respectively, local) to each other. We generalize the notation $n_s$ introduced in Chapter 3 for number of contacts in a square in the obvious way to number of contacts in the interactive region $n_{I_s}$, and local region $n_{L_s}$.

The important point is that the interaction matrix is numerically low-rank. Specifically, there is a small number (which we assume is upper-bounded by a constant $c$) of rows, each of which is a linear combination of rows of $G_{I_s s}$, such that these rows form a basis (to a close approximation) for the row space of $G_{I_s s}$. This is the reason for our use of the term "row basis". (In our examples, we've found that choosing $c = 6$ gives a close enough approximation for very good accuracy.) If we have some way of getting these rows (we will see later how to do this with an SVD), we can write them down in a matrix $V_s'$ ($c$ rows, $n_s$ columns). Any vector $v$ of voltages in square $s$ which is orthogonal to the rows of $V_s'$ (i.e. $V_s'v = 0$) will have

$$G_{I_s s} v \approx 0, \tag{4.6}$$

because all the rows of $G_{I_s s}$ are approximate linear combinations of rows of $V_s'$. Thus, we can get an accurate representation of $G_{I_s s}$ by projecting $v$ onto the rows of $V_s'$ and knowing the responses to the $\leq c$ voltage vectors given by columns of $V_s$—in matrix notation the responses are $G_{I_s s} V_s$. If we construct $V_s'$ so that its rows are orthonormal, we get a compact representation of our approximation:

$$G_{I_s s} \approx (G_{I_s s} V_s) V_s'. \tag{4.7}$$

Using the new representation to do matrix-vector products results in a dramatic efficiency improvement over simply having the dense $G_{I_s s}$ and applying it in the naive way when the number of contacts in $s$ and $I_s$ is large. This is true both for storage and running time. $G_{I_s s}$ requires $n_s n_{I_s}$ entries of storage, while $G_{I_s s} V_s$ requires $\leq c n_{I_s}$ storage and $V_s'$ requires $\leq c n_s$ storage, for a total of at most $c(n_{I_s} + n_s)$ storage. The running time, in muliply-add operations, is the same as the storage for matrix-vector product. (More accuracy can be obtained using a variant of this idea which exploits the symmetry of $G$—this is discussed in Section 4.3.1.)

The goal of the first phase is to form the *row basis* representation of $G_{I_s s}$ for every square $s$ on every level, which consists of the row basis $V_s'$ (so-called because it is an approximate basis for the row space of $G_{I_s s}$) and approximations to the responses to the row-basis vectors $G_{I_s s} V_s$, as well as the finest-level local interaction matrices $G_{L_s s}$ for each finest-level square $s$. How the approximations are made is discussed in Section 4.3.3. We also call $V_s$ the "important vectors", since knowing responses to them is enough to approximate responses to anything. In Section 4.3.3, we address the question of how to form this representation. We will show how to apply (an approximate version of) $G$ once we have a row-basis representation for $G$, after first showing a refinement which exploits the symmetry of $G$ to improve the accuracy of the row-basis representation.

### 4.3.1 Exploiting symmetry of $G$ to improve accuracy

For each square $s$, we can consider the space of important vectors $\langle V_s \rangle$ as well as the orthogonal space $\langle W_s \rangle$ of $\langle V_s \rangle$ in $\mathcal{R}^{n_s}$. The bases $V_s$ and $W_s$ are chosen so that $\begin{pmatrix} V_s & W_s \end{pmatrix}$ is an orthogonal matrix. This is similar to the wavelet definition, but now we have

$$\langle W_s \rangle \oplus \langle V_s \rangle = \mathcal{R}^{n_s}, \tag{4.8}$$

the whole space of voltage vectors on the contacts in $s$, as opposed to (3.13), where the direct sum is not the whole space.

In our algorithm, $W_s$ will not be formed explicitly (except on the finest level), because it wouldn't be efficient to do so, but it is useful for thinking about our assumptions. Just as we only assumed that interactions between two vanishing-moments basis vectors in well-separated squares are small, here we only want to assume that

$$W_d' G_{ds} W_s \approx 0. \tag{4.9}$$

for non-local squares $d$ and $s$. Notice that this is actually assuming *less* than (4.7). One way to look at the situation is to write $G_{ds}$ in the following way:

$$\begin{aligned}
G_{ds} &= \begin{pmatrix} V_d & W_d \end{pmatrix} \begin{pmatrix} V_d' \\ W_d' \end{pmatrix} G_{ds} \begin{pmatrix} V_s & W_s \end{pmatrix} \begin{pmatrix} V_s' \\ W_s' \end{pmatrix} && (4.10)\\
&= \begin{pmatrix} V_d & W_d \end{pmatrix} \begin{pmatrix} V_d' G_{ds} V_s & V_d' G_{ds} W_s \\ W_d' G_{ds} V_s & W_d' G_{ds} W_s \end{pmatrix} \begin{pmatrix} V_s' \\ W_s' \end{pmatrix} && (4.11)\\
G_{ds} &= V_d V_d' G_{ds} V_s V_s' + V_d V_d' G_{ds} W_s W_s' \\
&\quad + W_d W_d' G_{ds} V_s V_s' + W_d W_d' G_{ds} W_s W_s'. && (4.12)
\end{aligned}$$

Under the strong assumption (4.7), the terms in the right column of the $2 \times 2$ matrix in (4.11) can be ignored, and (4.12) becomes

$$\begin{aligned}
G_{ds} &\approx V_d V_d' G_{ds} V_s V_s' + W_d W_d' G_{ds} V_s V_s' \\
G_{ds} &= G_{ds} V_s V_s',
\end{aligned}$$

as expected. We used the identity $VV' + WW' = I$, which holds when $\begin{pmatrix} V & W \end{pmatrix}$ is orthogonal.

If we use the weaker assumption of (4.9), we can expect to achieve higher accuracy because only the lower-right entry in the $2 \times 2$ matrix in (4.11) is ignored. In this case the approximation becomes

$$\begin{aligned} G_{ds} &\approx V_d V_d' G_{ds} V_s V_s' + V_d V_d' G_{ds} W_s W_s' + W_d W_d' G_{ds} V_s V_s' && (4.13) \\ &= G_{ds} V_s V_s' + V_d V_d' G_{ds} (I - V_s V_s') && (4.14) \\ &= (G_{ds} V_s) V_s' + V_d (G_{sd} V_d)' (I - V_s V_s') && (4.15) \\ &\approx (G_{ds} V_s)^{(r)} V_s' + V_d (G_{sd} V_d)^{(r)'} (I - V_s V_s') && (4.16) \end{aligned}$$

The second exact equality comes from the symmetry of $G$. The last line has the superscript $(r)$ to indicate the use of the approximate row-basis responses returned by the algorithm described in 4.3.3. Notice that no new information is needed to apply this refined approximation—we assumed we have the row basis vectors and responses to them in every square, and that allows us to apply (4.16). From now on, we will make only the weaker assumption and thus use (4.16).

This assumption is also related to how the combine-solves technique is used. This is the same as the method developed in Section 3.5, and we use it in our algorithm for constructing the row-basis representation. With our assumption one can show, in exactly the same way as used in Section 3.5, that it is possible to add many vectors, apply $G$ to their sum, and extract responses to the individual components. The key point is the following: if $v_s$ with support in square $s$ is one vector in a sum used in combine-solves, and $v_s \in \langle W_s \rangle$, then we can extract $v_d' G_{ds} v_s$ accurately for any $v_d \in \langle W_d \rangle$. *The projection onto a vector in $W_d$ is required* for accuracy, i.e. we can't expect to get just $G_{ds} v_s$ accurately using the combine-solves method.

Our initial implementation of the row-basis method used the stronger assumption, and the accuracy was not what we had hoped. Reworking the algorithm using the weaker assumption, as just described, resulted in a dramatic improvement in accuracy at a constant factor ($< 2$) increase in computational cost of applying the operator.

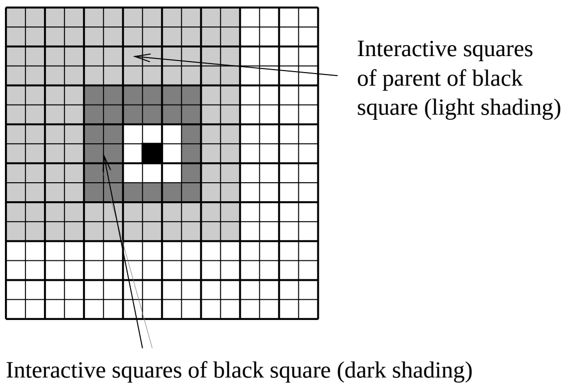

Figure 4-5: Nesting of interactive spaces: outer unshaded squares are interactive squares of grandparent square of small black square $s$

Also, this increase in cost of applying the operator applies only to the row-basis representation; the wavelet-like representation developed in Section 4.4 is just as efficient with the weaker assumption.

### 4.3.2 Applying the operator

Given $v$, how do we obtain an approximate $Gv$ using our row-basis representation? To see, let $s$ be the small black square in Figure 4-5. To get responses to voltages in $s$ everywhere, we first need an approximate $G_{I_g g}$ where $g$ is the grandparent square of $s$, because the interactions of $s$ with the unshaded outer squares are obtained from interactions of $g$ with the interactive squares of $g$. Then we need an approximate $G_{I_p p}$ where $p$ is the parent square of $s$ to get the interactions with the lightly shaded squares. Then we use an approximate $G_{I_s s}$ to get the response in the interactive squares of $s$, and finally we use an approximate $G_{L_s s}$ to get the responses in the squares local to $s$. The nesting of interactive regions is shown in Figure 4-5. In general, the response to a vector $v$ can be found by going through the levels, and for each square $s$ on each level, finding $G_{I_s s}v_s$, where $v_s$ is the restriction of $v$ to square $s$, and adding the current response to the total current. Finally the local interactions $G_{L_s s}v_s$ must be added in for all squares $s$ on the finest level.

We summarize this procedure in the following pseudocode, which includes the refinement discussed in Section 4.3.1 to improve accuracy. The levels start at level

2 since the interactive region of any square on a coarser level (1 or 0) is empty. The superscripts $(r)$ and $(f)$ indicate the use of an approximation to the quantity in parentheses, from the multilevel row-basis algorithm which will be described in the next section.

```
for lev := 2 to maxlev
    for each square $s$ on level lev
        for each square $d$ on level lev in $I_s$
            compute $i_d = (G_{ds}V_s)^{(r)}V_s'v_s + V_d(G_{sd}V_d)^{(r)\prime}(v_s - V_sV_s'v_s)$
            $i := i + \text{zeropad}(i_d)$
        end
    end
end
Now deal with finest-level local interactions
for each square $s$ on level maxlev
    for each square $d$ in $L_s$
        compute $i_d = G_{ds}^{(f)}v_s$
        $i := i + \text{zeropad}(i_d)$
    end
end
```

## Complexity analysis for regular contact layouts

We now show that for a regular contact layout the apply-operator algorithm is $O(n \log n)$ in memory and time. Assume there are $d$ contacts in each finest level square, and at most $c$ vectors in $V_s$ for any square $s$ at any level.

Then on the finest level $L$ there are $2^{2L}$ squares, giving a total of $n = 2^{2L}d$ contacts. Each square has a maximum of 27 interactive squares. Each interaction of two squares on level $i$ (the "compute" step inside the 3 nested for loops at the start of the algorithm) costs at most $5cd2^{2(L-i)}$ multiply-add operations. The reason is that the matrices $V_s$ and $V_d$ each have at most $cd2^{2(L-i)}$ entries ($c$ columns, number of rows equals number of contacts in the square, which is $d2^{2(L-i)}$). $V_s'$ can be applied to $v_s$ just once, and in addition $V_s$, $V_d'$, $(G_{ds}V_s)^{(r)}$, and $(G_{sd}V_d)^{(r)'}$ must each be applied once. Each of these 5 matrices costs the same $cd2^{2(L-i)}$ multiply-adds to apply.

For level $i$, there are $2^{2i}$, $\leq 27$ interactions per square with other squares, and $5cd2^{2(L-i)}$ multiply-adds per interaction, giving at most

$$27 \cdot 2^{2i} \cdot 5cd2^{2(L-i)} \cdot 2^{2i} = 135cd2^{2L}$$

multiply-adds for this level. Notice that this is independent of $i$, and the total operation count for all levels is thus at most $135Lcd2^{2L}$, except for the finest-level local interactions. Since $n = 2^{2L}d$, $L = O(\log n)$, so $L \cdot 135 \cdot 2^{2L}d$ is $O(n \log n)$. On the finest level we use the full $G_{ds}^{(f)}$ for each local interaction. Each such matrix has $d^2$ entries, and there are $\leq 9$ local squares for each of the $2^{2L}$ finest level squares, giving a bound of $9d^2 2^{2L}$ operations for this part of the algorithm, which is $O(n)$, so the total complexity remains $O(n \log n)$.

### 4.3.3 Forming the representation

We now show how to form the multilevel row-basis representation. This is easiest to do on the coarsest level (level 2); additional techniques are required to make the algorithm work efficiently on finer levels.

#### Coarsest level

There are 16 squares on the coarsest level lev = 2. We fix one of these squares $s$, and show how to form the row basis $V_s'$ for $G_{I_s s}$. One approach is to take a truncated singular value decomposition. That is, we choose a threshold $\epsilon$ and take the SVD

$$G_{I_s s} = U\Sigma V' = \begin{pmatrix} U\Sigma_{\text{large}} & U\Sigma_{\text{small}} \end{pmatrix} \begin{pmatrix} V_s' \\ W_s' \end{pmatrix}, \tag{4.17}$$

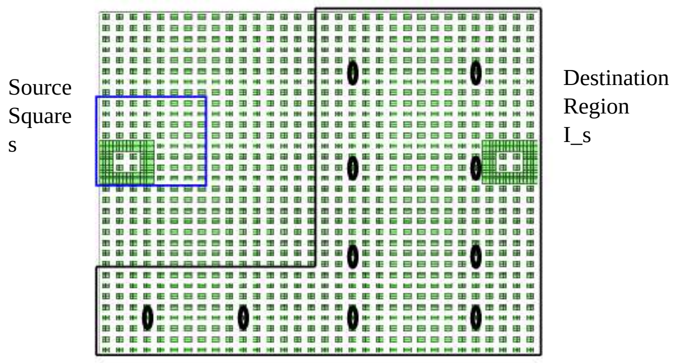

Figure 4-6: Sampling illustrated: coarsest-level sample contacts for square A are circled

where $\Sigma_{\text{large}}$ consists of the columns of $\Sigma$ with singular values bigger than $\epsilon$ and $\Sigma_{\text{small}}$ is the rest of the columns of $\Sigma$. Other criteria can be used instead for $\Sigma_{\text{large}}$: keeping the first $c$ singular values, for example, or a relative error criterion such as keeping singular values $> \epsilon\Sigma(1,1)$. If $n_{s(\text{large})}$ is the number of large singular values (number of columns of $\Sigma_{\text{large}}$), then $V_s'$ is defined to be the first $n_{s(\text{large})}$ rows of $V'$ and $W_s'$ is the remaining rows of $V'$. The rows of $V_s'$ form our approximate row basis for $s$, and $\langle W_s \rangle \oplus \langle V_s \rangle = \mathcal{R}^{n_s}$. There is a problem with this approach, though: it requires obtaining the whole dense $G_{I_s s}$, which would require a number of black-box calls equal to $n_s$ for every square $s$.

In order to have only $O(\log n)$ black-box calls, we use a version of *sampling*, similar to [18]. The idea is to get a few sample rows of $G_{I_s s}$ and get the row basis by taking the SVD of these sample vectors. In its simplest form, a sample row gives the current due to voltages in $s$ on a sample contact in $I_s$. We can write this in matrix form by making a matrix of sample vectors $S_s$ each of whose columns is a 0-1 vector which is 1 only at the index of the sample contact. Then the matrix of sample rows is $S_s'G_{I_s s}$. This type of interaction is illustrated in Figure 4-6. Actually, there is no reason to restrict to these 0-1 vectors; we can use any vectors with support only in $I_s$ as the rows of $S_s'$ and call them the sample vectors. We can obtain $S_s'G_{I_s s}$ by using the symmetry of $G$:

$$S_s'G_{I_s s} = (G_{s I_s} S_s)'. \tag{4.18}$$

We need only obtain $G_{sI_s}S_s$, which can be done with $n_{\text{samp}(s)}$ solver calls, where $n_{\text{samp}(s)}$ is the number of sample vectors (columns) in $S_s$. 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. So, on a given level, we can choose 1 sample vector per square, and then for a given square $s$, $S_s$ consists of those sample vectors with support in $I_s$. We actually choose the sample vector with support in square $s$ randomly (MATLAB randn). There are never more than 27 interactive squares for a given square $s$ on any level, and thus the number of sample vectors in $S_s$ is bounded by a small constant. (For very irregular contact layouts, it's possible there might not be enough sample vectors in the interactive squares if many of these squares have no contacts. In this case, one could use more than 1 sample vector per square, an approach we have implemented. However, this still doesn't cover all cases, for example if all of the interactive squares of $s$ are empty. One way to solve this problem would be to choose sample vectors in faraway squares if there aren't enough contacts in the interactive squares; we have not yet implemented this.)

We take the reduced SVD of $G_{sI_s}S_s$, an $n_s \times n_{\text{samp}(s)}$ matrix, obtaining

$$G_{sI_s}S_s = U\Sigma V' = \begin{pmatrix} V_s & W_s \end{pmatrix} \begin{pmatrix} \Sigma_{\text{large}} \\ \Sigma_{\text{small}} \end{pmatrix} V', \tag{4.19}$$

where $\Sigma_{\text{large}}$ is defined to be the first $n_{\text{large}}$ *rows* of $\Sigma$ (those with large enough singular values according to whatever threshold criterion is used), and $\Sigma_{\text{small}}$ is the rest of the rows of $\Sigma$. Notice that because we took the reduced rather than the full SVD, $U$ is $n_s \times \min(n_{samp}, n_s)$, so that no matter how large $n_s$ is, the fact that $n_{samp}$ is $\leq 27$ means that $U$ has at most a small constant number of columns. $V_s$ is defined to be the first $n_{\text{large}}$ columns of $U$ and $W_s$ is the rest of $U$. This $V_s$ is an approximate column basis for $G_{sI_s}S_s$, as can be seen by approximating $\Sigma_{\text{small}} = 0$ in (4.19), obtaining

$$G_{sI_s}S_s \approx V_s\Sigma_{\text{large}}V', \tag{4.20}$$

the right-hand side of which consists of a matrix whose columns are in the column space of $V_s$. Thus $V_s'$ is an approximate row basis for $(G_{sI_s}S_s)' = S_s' G_{I_s s}$, which is what we wanted.

With $\leq c$ additional black-box solver calls per coarsest-level square $s$, the responses $G_{I_s s}V_s$ can be obtained. Since there are only 16 such squares, the number of black-box solver calls required is bounded by the constant $16c$. (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 $G_{P_s s}V_s$, 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 $s$. The same approach of choosing one sample vector per square and sharing these among the source squares to form an approximate $G_{I_s s}$ that was described for the coarsest level is used here.

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 solver 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 $\Omega(n)$, so $\Omega(n)$ solves would be needed.

To reduce the number of black-box solver calls required, we combine solves as described in Section 3.5. For a square $s$ on level $l$, a vector $v_s$ (length $n_s$) of the voltages in that square can be expressed as a sum of two vectors in the parent square as follows. First extend $v_s$ to a vector $v$ of voltages in $p$ (length $n_p$) 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

$$v = V_p V_p' v + (I - V_p V_p')v \tag{4.21}$$

So we have

$$\begin{aligned} G_{P_s s} v_s = G_{P_s p} v &= (G_{P_s p} V_p) V_p' v + G_{P_s p}(I - V_p V_p') v \\ &\approx (G_{P_s p} V_p)^{(r)} V_p' + (G_{P_s p}(I - V_p V_p') v)^{(c_a)} \end{aligned} \qquad (4.22)$$

(The superscripts $(r)$ and $(c_a)$ indicate the use of approximations which will be explained shortly.) 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 $G_{P_s p}$ from the parent level. We have $G_{P_s p}$ 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 $P_s$ in (4.22).

The first term on the right-hand-side of (4.22) is computed from the next-coarser level $(l - 1)$ row-basis representation; the superscript $(r)$ indicates that the exact $G_{P_s p} V_p$ is not used (because we have no way of getting it efficiently on finer levels); instead the approximation to it from this algorithm applied on the parent level is used. The second term is $G_{P_s p}$ applied to $(I - V_p V_p') v$, and $(I - V_p V_p') v$ is orthogonal to the level $l-1$ row basis in square $p$, i.e., in $\langle W_p \rangle$. We can use the combine-solves technique to group many such vectors (from squares spaced 3 apart, as described in Section 3.5) into one black-box call. The superscript $(c_a)$ indicates the approximation implied by use of the combine-solves technique. The relations among regions of contacts used in the splitting method are shown in Figure 4-7.

We are not quite done. As mentioned in Section 4.3.1, we can only expect to accurately approximate entries of the form $w_q G_{qp} w_p$, where $w_q \in W_q$, $w_p \in W_p$, and $q$ is a square local to $p$. We cannot expect that the "raw" output $(G_{qp} w_p)^{c_a}$ of combine-solves will be accurate. By the "raw" output, we mean the entries of $G v_{\text{sum}}$ which give currents on contacts in local squares of $p$, where $v_{\text{sum}}$ is the sum (including $w_p$ as a summand) sent to combine-solves.

What we do is to separate $G_{P_s p}$ into its parts $G_{qp}$ for every square $q$ local to $p$. (The local squares of the parent $p$ of $s$ together form the set $P_s$ of interactive and local squares of $s$.) Then for each local square $q$, we just need to approximate

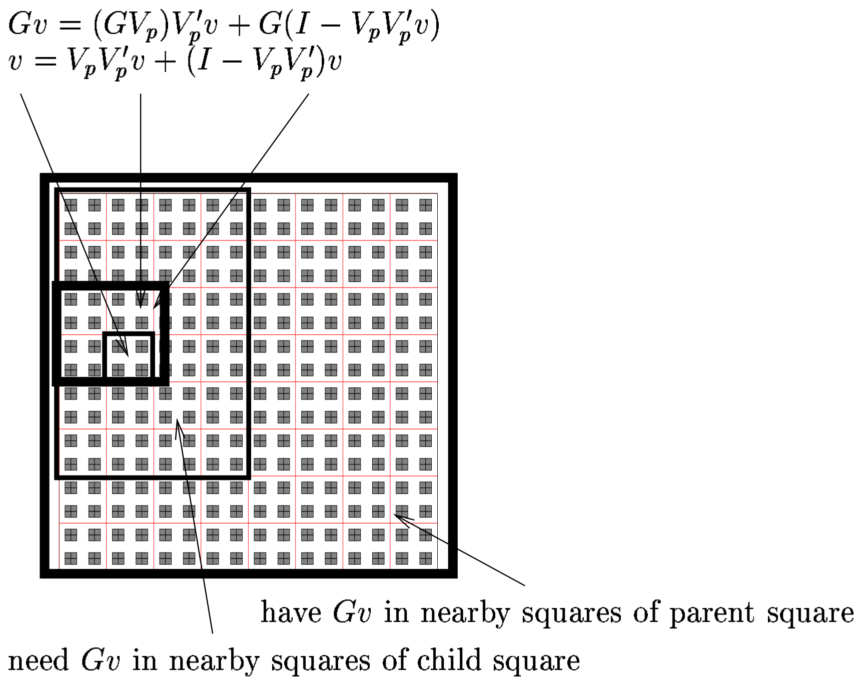
$Gv = (GV_p)V_p'v + G(I - V_pV_p'v)$
$v = V_pV_p'v + (I - V_pV_p')v$
have $Gv$ in nearby squares of parent square
need $Gv$ in nearby squares of child square

Figure 4-7: Splitting method illustrated

$G_{qp}(I - V_pV_p')v$. We do this by another splitting. Write $w = (I - V_pV_p')v$. Then

$$\begin{aligned}
G_{qp}(I - V_pV_p')v &= (V_qV_q' + (I - V_qV_q'))G_{qp}w \\
&= V_qV_q'(G_{qp}w) + (I - V_qV_q')(G_{qp}w) \\
&= V_q(G_{pq}V_q)'w + (I - V_qV_q')(G_{qp}w) && (4.23) \\
&\approx V_q((G_{pq}V_q)^{(r)})'w + (G_{qp}w)^{(c_a)} - V_qV_q'(G_{qp}w)^{(c_a)} && (4.24)
\end{aligned}$$

The first term of (4.23) is approximated using the row basis $V_q$ in the destination square $q$ and the approximate response $(G_{pq}V_q)^{(r)}$ to it. Since each entry in the second term is a row of $(I - V_qV_q')$ multiplied by $G_{qp}w$, we just need to show that each row of $(I - V_qV_q')$ is in the space $\langle W_q \rangle$. But this is clear, since $I - V_qV_q' = W_qW_q'$. Thus the second term can be extracted accurately using the combine-solves technique. We denote the more accurate approximation to $G_{P_sp}W_p$ obtained by (4.24) by $(G_{P_sp}W_p)^c$.

Exactly the same splitting technique can be used to obtain the approximate responses $(G_{P_ss}V_s)^{(r)}$ to the row basis in square $s$.

### Finest level

At this point, we have represented all of $G$ corresponding to squares which are interactive at some level. The only parts of $G$ that remain to be computed are the interactions of local squares on the finest level. For each finest-level square $s$ we already have the responses to the row-basis vectors $V_s$. Because finest-level squares each contain at most a small constant number of contacts, the explicit formation of the orthogonal space $W_s$ is not computationally prohibitive, unlike on higher levels ($W_s$ is $n_s \times (\leq n_s)$). The combine-solves technique is used, with the technique to improve accuracy described by (4.24), to obtain local responses to the columns of $W_s$.

Also, we can explicitly form

$$\begin{aligned} G_{L_{ss}s} &= G_{L_{ss}s}(V_sV_s' + W_sW_s') &(4.25)\\ (G_{L_{ss}s})^{(f)} &:= (G_{L_{ss}s}V_s)^{(r)}V_s' + (G_{L_{ss}s}W_s)^{(c)}W_s', &(4.26)\end{aligned}$$

where $(G_{L_{ss}s})^{(f)}$ means finest-level approximation.

## 4.3.4 Making the multilevel row-basis representation: algorithm summary

The following is a pseudocode summary of the algorithm just described:

```
*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 square $p$ row-basis,
            $o_s$ orthogonal to parent level row-basis)
        end
        Use combine-solves technique to approximate $(G_{P_{sp}} o_s)^{(c)} \approx G_{P_{sp}} o_s$.
        for each square $q$ local to $p$ on parent level
            Use (4.24) to refine approximation of $G_{qp} o_s$
        end
        Combine refined approximations to get $(G_{P_{sp}} o_s)^{(c)}$
        for each square $s$ on level lev
            Use parent-level row-basis responses to get local and interactive
            approximate responses $(G_{P_{sp}} V_s)^{(r)} V_s' r_s$ to $r_s$
            Approximate $G_{P_s s} m_s$ by $(G_{P_{sp}} o_s)^{(c)} + (G_{P_{sp}} V_s)^{(r)} V_s' r_s$
        end
    end *the lev$\neq$2 case*
    for each square $s$ on level lev
        Get row basis $V_s$ using sample vectors and SVD
        Get approximate responses $(G_{P_s s} V_s)^{(r)}$ to row basis:
        same method as getting responses to sample vectors
    end
end
for each square $s$ on finest level (lev==maxlev)
    Get $G^{(f)}_{L_s s}$ using (4.26) and
    combine-solves technique
end
```

## 4.4 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 [22, 42, 41], obtaining $G \approx QG_{ws}Q'$. In this section, whenever we use the notation $G_{ab}$, it means the approximate $G_{ab}$ obtained from (4.16).

This is a simpler representation to work with and has the advantage that further sparsity can be obtained by thresholding out small entries in $G_{ws}$ to form an even sparser $G_{wt}$, trading off better sparsity for decreased accuracy. It also makes comparisons to previous work [41] 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 $U_s$ 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, $Z_s = \begin{pmatrix} T_s & U_s \end{pmatrix}$, should have orthogonal columns.

The finest level is very easy: for a square $s$ on the finest level, $U_s$ consists of the row basis for $s$; i.e. $U_s = V_s$. The columns of $T_s$ form a basis for the orthogonal space of $\langle U_s \rangle$ in $\mathcal{R}_{n_s}$; i.e. $T_s = W_s$.

### 4.4.1 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 SVD. Let $X_p$ be the matrix whose columns are the columns of $U_{s_1}$, $U_{s_2}$, $U_{s_3}$, and $U_{s_4}$, zero-padded so they are written in the standard basis of the parent square $p$, for each of the four children $s_1$, $s_2$, $s_3$, $s_4$ of $p$. Our method is to take the SVD of $G_{I_p p}X_p$. This represents the interaction of the child-square slow-decaying basis functions with the contacts in the interactive squares of $p$. Take the reduced SVD of $G_{I_p p}X_p$, and set $U_p$ and $T_p$ to sections of $V$ as shown:

$$G_{I_p p} X_p = U \Sigma V' = U \begin{pmatrix} \Sigma_{\text{large}} & 0 \\ 0 & \Sigma_{\text{small}} \end{pmatrix} \begin{pmatrix} U'_p \\ T'_p \end{pmatrix}. \tag{4.27}$$

For very irregular contact layouts, it is possible that there will be zero or very few contacts in the interactive squares, making it impossible to use this interaction to effectively distinguish the fast-decaying vectors in $p$. In this case, the algorithm needs to be modified to take in contacts in the faraway squares of $p$ as well; we do not consider this here, and assume that $n_{I_p} > \#\ \text{columns}(X_p)$. (This also assures that $\Sigma$ in (4.27) will be square as shown.)

Choose the number of columns in $U_p$ equal to the number of singular values in $\Sigma_{\text{large}}$ (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 get

$$G_{I_p p}(X_p T_p) = U(:, |\Sigma_{\text{large}}| + 1 : |\Sigma_{\text{large}}| + |\Sigma_{\text{small}}|) \Sigma_{\text{small}}. \tag{4.28}$$

The right-hand side is close to zero (assuming the $\Sigma_{\text{small}}$ are in fact very small), suggesting that the columns of $X_p T_p$ 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. The coarsest-level slow-decaying basis functions and the fast-decaying basis functions on every level form the columns (once zero-padded) of our orthogonal change-of-basis matrix $Q$.

We briefly sketch how the entries of $G_{ws}$ 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. (Just as in the wavelet method, 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 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 $U_s$ basis vectors in each square at that level. We have this already for the finest level, from the multilevel row-basis 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, and the interaction with interactive squares of that child. We get the local interactions from the data structure maintained on the child level, and the interactive square interactions can be obtained using the row basis of $s_i$ and the response to it in the interactive squares $I_{s_i}$ of $s_i$.

Although we don't give a formal complexity analysis of the two phases (coarse-to-fine and fine-to-coarse), it is possible to do so for reasonably regular contact layouts, obtaining a cost in storage and time of $O(n \log n)$. Also, the $G_{ws}$ and $Q$ obtained by the fine-to-coarse method each have $O(n \log n)$ nonzeros. The key facts used are that approximately applying $(G_{sd}V_s)^{(r)}$, where $s$ and $d$ are interactive, is an $O(n_s + n_d)$ operation, and that applying $V_s'$ is an $O(n_s)$ operation. Then the need to apply many such operations on the finer levels is balanced by the smaller number of contacts in each finer-level square.

### 4.4.2 Algorithm summary: fine-to-coarse sweep

```
Phase 2: get wavelet-structure basis $Q$
for lev:=maxlev downto 2
    Form $T_s$ (fast-decaying response),
    $U_s$ (slow-decaying response)
    for each square at level lev:
    if lev==maxlev
```

```
        for each square $s$ at level maxlev
            set $U_s = V_s$, set $T_s = W_s$
        end
    else
        for each square $s$ at level lev
            use SVD to get $T_s$, $U_s$ from
            children $W_{c_1} \dots W_{c_4}$
        end
    end
    For each square $s$ on level lev,
    put zero-padded vectors from $T_s$ into $Q$
    if lev==2
        For each square $s$ on level lev,
        put zero-padded vectors from $U_s$ into $Q$
    end
end
Fill in $G_{ws}$: form interactions of fast-decaying basis functions
with each other and with coarsest-level slow-decaying basis functions,
and coarsest-level slow-decaying basis vector interactions
with each other
```

## 4.5 Previous work using SVD-based methods

Our work is certainly not the first sparsification method to rely on low-rank approximation and the singular value decomposition. Here we attempt to give a brief overview of previous work which uses the SVD for similar problems, and to highlight what is unique in our contribution. The main features of our algorithm which distinguish it from others which have been proposed for similar problems are the following: the reliance on a fast black-box solver which we want to call only a near-constant number of times (*without* assuming constant-time access to individual matrix entries), the use of a wavelet-like change-of-basis, a "multipole-like" algorithmic structure, and the use of sampling to avoid taking SVDs of large dense matrices.

The first feature is, to the best of the author's knowledge, unique to our algorithm. We rely only on the ability to apply $G$ $O(\log n)$ times, aided by use of the combine-solves technique. Other techniques which use the SVD require either explicit knowledge of the kernel and analytic properties of it, or access to individual entries of $G$. We have mentioned earlier that there is no known way to get constant-time access to individual entries of $G$.

Even if we leave this issue aside, all of the other approaches for matrix sparsification of integral operators which we are aware of lack at least one of the other features mentioned. We briefly discuss some of these methods.

The approach of [18] and the associated IES$^3$ code was the first SVD-based method which became well-known in the electronic design automation community. It is applicable to matrices, such as Galerkin $1/r$ matrices, for which entries are accessible in constant time. It uses the idea of sampling and is thus able to achieve an $O(n \log n)$ extraction cost. However, the structure is not multipole-like. By multipole-like, we mean that interactions between source and destination squares are computed using a representation for the source square and a representation for the destination square *separately* (such as the multipole and local Taylor expansions of the multipole algorithm). In the approach of [18], a low-rank representation is computed by taking the SVD of the (sampled) interaction between every *pair* of interactive squares. In contrast, our low-rank approach uses a representation $V_s$ (and responses to it) for the source square and $V_d$ (and responses to it) for the destination square. Also, because the "important vectors" for the IES$^3$ approach differ for different destination squares given the same source square, their approach is not a global change of basis.

The machinery of H-matrices and H$^2$-matrices [20, 21, 43] is another SVD-based sparsification approach. The H-matrix construction is very similar to that of IES$^3$. Truncated SVDs (in which the few largest singular values kept) of large matrix blocks are required. When Taylor expansions of the kernel are available, the truncated SVD can be obtained efficiently, but the approach of sampling matrix entries is not used. The $H^2$-matrix construction, unlike the H-matrix construction, has a multipole-like structure. Again sampling is not used.

The wavelet approach of [22] of course gives a wavelet-basis representation of the operator, and has a multipole-like structure. The SVD is used efficiently, but only to choose additional "fast-decaying" basis functions based on geometric moment-matching (i.e., SVDs of interactions between well-separated squares are not used here).

The method of [44] is an SVD-based algorithm with multipole structure. However, SVDs of finest-level interactions of each square with its faraway squares are formed, clearly $\Omega(n^2)$ work without the use of sampling. The authors of [44] have also used the SVD in what is essentially a multipole implementation, to represent the many operators required in a multipole algorithm (i.e., the far-field-to-local "flip" operator) [45].

Finally, it is interesting to note that one of the above methods, as well as an approach developed in the computational electromagnetics community [46], can be used to efficiently invert "sparsifiable" matrices directly. Given an H-matrix, it is possible to get an H-matrix representation of the inverse. Thus, for example, one could represent a $1/r$ potential-from-charge matrix as an H-matrix, invert the representation, and have an H-matrix representation of the charge-from-potential operator. Can we apply the same idea to our problem, representing the contact-current to contact-potential matrix as an H-matrix, and inverting the representation to get the potential-to-current (conductance) matrix? The difficulty is that the Green's function current-to-potential matrix is available for *panel* currents to panel potentials, not contact currents to contact potentials. (Notice that panel current densities are *not* constant across a contact, but are determined by the constraint that potential on each contact is constant.) The approach of [46] is also one of direct inversion, although in this case it is applied only to a particular, one-dimensional, problem. The idea of direct inversion may be useful in many contexts, perhaps for example in forming a fairly inaccurate (i.e., rank-1) approximate inverse to use as a preconditioner.

## 4.6 Computational results

Just as for the wavelet method, sparsity and accuracy must be considered together to assess the quality of the results. The fact that the two-phase low-rank algorithm produces the same type of representation $G \approx QG_{ws}Q'$ as the wavelet method facilitates comparisons.

For our threshold criterion, we chose to consider as "large" the singular values which were larger than 1/100 of the largest singular value, to a maximum of 6 large singular values (corresponding to the 6 constraints imposed by matching moments up to order 2).

There are two basic ways we use to compare the accuracy/sparsity tradeoff of the low-rank algorithm with the wavelet algorithm. The first is to look at the sparsity obtained by each without any thresholding to remove small entries, and to consider the maximum relative error in any entry for $QG_{ws}Q'$ computed by the wavelet method and $QG_{ws}Q'$ computed by the low-rank method, versus $G$ computed in the obvious way by calling the black box once per contact.

The second way (possibly more realistic, given the fairly low accuracy requirements for typical applications) is to get an even sparser representation $G \approx QG_{wt}Q'$ by truncating small entries in $G_{ws}$. We chose the truncation threshold so that $G_{wt}$ would be approximtately 6 times sparser than the sparsity of the low-rank $G_{ws}$ (binary search was used to achieve this). Instead of looking at the maximum relative error, we look at the proportion of entries in $QG_{wt}Q'$ with relative error higher than 10%.

In the second case, we compare the low-rank method to the wavelet method in two ways. First, we can threshold entries in the wavelet $G_{ws}$ to obtain a $G_{wt}$ with equivalent sparsity to the thresholded low-rank $G_{wt}$, and then compare the accuracy. Second, we can attempt to set the threshold for the wavelet representation so that the thresholded wavelet and low-rank representations have equivalent accuracy.

We show results for three examples, including two which are familiar from the wavelet chapter. Example 1 is the regular grid of contacts, and Example 2 is the oscillatory-size grid of contacts (alternating rows of large and small contacts), both

| Example | Sparsity factor (low rank) | Sparsity factor (wavelets) | Max. rel. error (low rank) | Max. rel. error (wavelets) | Solve reduction factor (low rank) | Solve reduction factor (wavelets) |
|---|---|---|---|---|---|---|
| 1 | 3.9 | 2.5 | 5.1% | 0.2% | 3.2 | 2.9 |
| 2 | 4.1 | 2.5 | 5.7% | 47% | 3.3 | 2.9 |
| 3 | 3.5 | 2.3 | 12% | 31% | 2.8 | 2.5 |

Table 4.1: Sparsity/accuracy tradeoff achieved by low-rank versus wavelet methods without thresholding

| Example | Sparsity of $G_{wt}$ (low rank rep. | Entries off by more than 10% (low rank) | Wavelet sparsity (equiv. accuracy) | Wavelet $QG_{wt}Q'$ entries off by more than 10% (equiv. sparsity) |
|---|---|---|---|---|
| 1 | 23 | 0.4% | 20 | 0.8% |
| 2 | 24 | 1.0% | 2.5 (*) | 89% |
| 3 | 21 | 1.4% | 6.6 | 94% |

Table 4.2: Sparsity/accuracy tradeoff for low-rank versus wavelet method: the (*) indicates that even with no thresholding the wavelet method didn't achieve the same accuracy as the low-rank method

described and shown in Chapter 3. Example 3 shows that our algorithm can deal with very irregularly shaped contacts. It includes some small square contacts, long thin contacts, and rings, which are all features of real substrate contact layouts. It is shown in Figure 4-8.

The results for the high-accuracy (no thresholding) approach are shown in Table 4.1. The wavelet algorithm outperforms the low-rank algorithm, in the high-accuracy case, for the regular-grid example. However, for the other two examples the low-rank method is far better, achieving a much smaller maximum relative error with slightly better sparsity and solve-reduction performance. (The solve-reduction factor is simply the ratio of the number of solves required to extract $G$ naively (i.e., the number of contacts) to the number required by the sparsification method used. It is a good measure for the efficiency of *extracting* the representation, since the solver calls are likely to be the dominant cost in an real-world implementation of our algorithms.)

The results with thresholding are presented in Table 4.2. In all cases, the low-rank

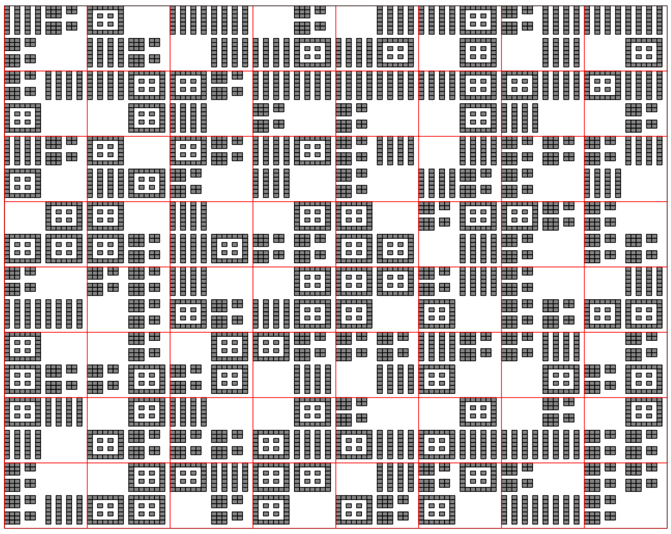
Figure 4-8: Contact layout for Example 3

| Example | Sparsity | Max. rel. error | Thresholded sparsity | Off by >10% | Solve reduction |
|---|---|---|---|---|---|
| 4 | 10 | 6.3% | 62 | 1.7% | 8.7 |
| 5 | 21 | 5.3% | 129 | 3.2% | 18 |

Table 4.3: Some results on larger examples

method is superior to the wavelet method. However, the performance difference is fairly slight for the regular-grid example. For Example 3, the contact layout is shown in Figure 4-8, and a spy plot of the low-rank $G_{wt}$ is shown in Figure 4-9.

In closing, we present some results for the low-rank algorithm on larger examples which show just how effective the algorithm can be. Table 4.3 shows results for two examples. The errors are based on taking a 10% sample of the columns of the actual $G$ because it is computationally prohibitive to form the whole $G$ when the number of contacts is so large. Example 4 is a 64 by 64 grid of contacts of alternating sizes, essentially the same as example 2 but 4 times as large. Example 5 has 10240 large and small contacts and is shown in Figure 4-10. This shows very good scaling of the low-rank algorithm as the number of contacts grows large, as one would expect for an algorithm which produces $O(n \log n)$ nonzero entries in the transformed-basis matrix. A spy plot of $G_{wt}$ for Example 5 is shown in Figure 4-11.

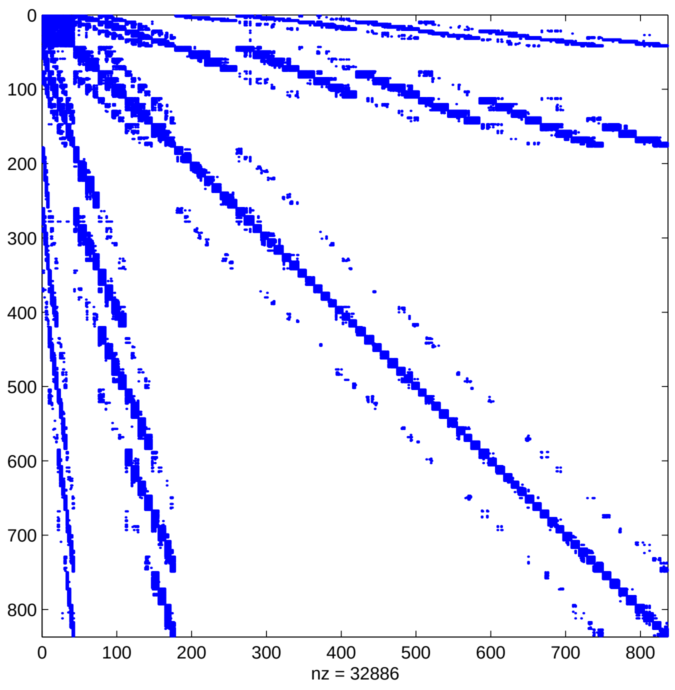

Figure 4-9: Spy plot for $G_{wt}$ for Example 3

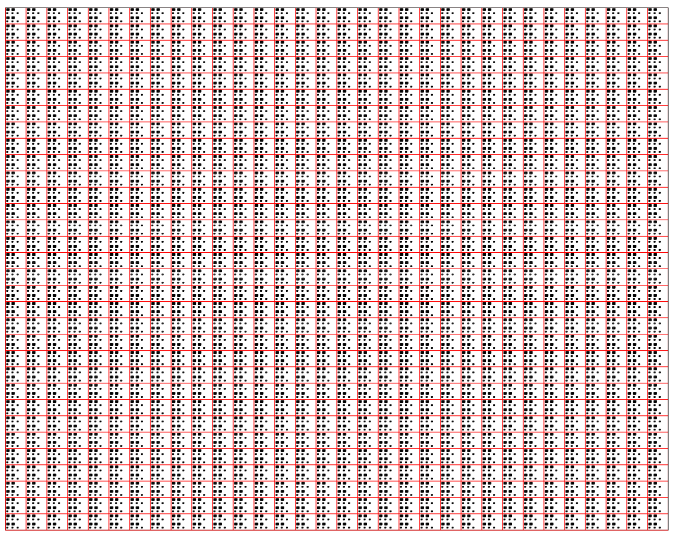

Figure 4-10: Contact layout of Example 5

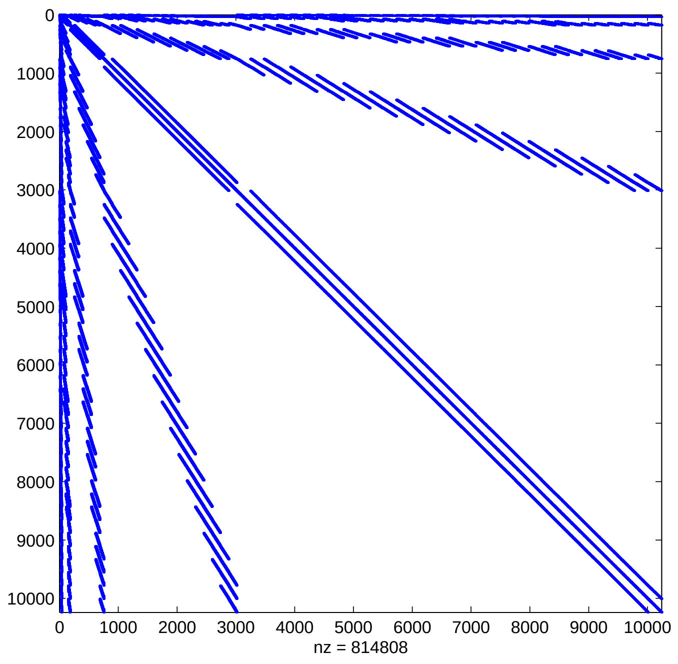

Figure 4-11: Spy plot for Example 5

# Chapter 5

# Conclusions and Future Work

## 5.1 Conclusions

We have shown two methods for extracting and sparsifying the substrate coupling conductance matrix. Both work better than the naive method of simply thresholding away small entries in the original $G$, but the low-rank method is more effective on examples which include contacts of different sizes and shapes. For our largest example (10240 contacts), we obtained a factor of about 20 in solve reduction with a factor of over 100 in sparsity with only 3 percent of the entries in the approximate $G$ off by more than 10 percent, even though the smallest entries are less than 1/500 of the largest off-diagonal entries. Because our algorithm produces a representation $G_{ws}$ of $G$ with $O(n \log n)$ nonzero entries for reasonably regular contact layouts, we expect these improvement factors to grow as the number of contacts increases.

## 5.2 Future work

There are two basic directions for future work on the low-rank method. One is further development of the extraction/sparsification tool itself, including error analysis. In terms of developing the tool, is it possible to handle extremely large or long contacts efficiently? Right now they need to be broken up into many small contacts so that each fits in a finest-level square, thus increasing the number of contacts. It would be useful to avoid this increase.

In terms of error analysis, is it possible to bound the error in some way? We have good experimental results, but it would be nice to have a theoretical guarantee. For example, one might hope to show a relation between the number of singular values kept and the maximum relative error for entries in $QG_{ws}Q'$ which are above some small value which is also a function of the number of singular values kept. If it is not possible to prove such a general statement, perhaps specific parts of the algorithm can be analyzed effectively. For example, is it possible to obtain a high-probability bound for the error in the row-basis calculation resulting from the use of random sample vectors in the interactive region, rather than the SVD of the entire interaction?

The other major direction is to try to use the tool to efficiently simulate the substrate in the context of a large circuit simulation. It is not immediately obvious how to use the ability to apply $G$ quickly in a SPICE-type circuit simulator. A method which does this has been developed in [11] and it would be interesting to see how it works on large examples and what kind of error in approximating $G$ is acceptable in the circuit simulation context.

# Appendix A

# MATLAB notation

(MATLAB is a registered trademark of The MathWorks, Inc.) We use both MATLAB-style and conventional matrix notation throughout the thesis. The prime ($'$) symbol is used to denote matrix transpose (actually, conjugate transpose, but all our matrices are real so it doesn't matter). $G(a, b)$ is the entry in row $a$, column $b$ of $G$. $G(:, b)$ is column $b$ of $G$, and $G(a, :)$ is row $a$ of $G$. Just as in MATLAB, in the diagonal $\Sigma$ matrix of singular values, the singular values are in decreasing order along the diagonal.

For submatrices, we tend to use the more conventional notation. If $s$ and $d$ are index sets, then $G_{ds}$ is the submatrix of $G$ obtained by taking the columns in $s$ of the rows in $d$ of $G$. That is, it is $G(d, s)$ in MATlAB notation.

# Bibliography

[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] 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.

[3] T. A. Johnson, R.W. Knepper, V. Marcellu, and W. Wang, "Chip substrate resistance modeling technique for integrated circuit design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits*, vol. CAD-3, no. 2, pp. 126–134, 1984.

[4] Nishath K. Verghese, David J. Allstot, and Mark A. Wolfe, "Verification techniques for substrate coupling and their application to mixed-signal ic design," *IEEE Journal Solid-State Circuits*, vol. 31, no. 3, pp. 354–365, March 1996.

[5] Nishath Verghese, *Extraction and Simulation Techniques for Substrate-Coupled Noise in Mixed-Signal Integrated Circuits*, Ph.D. thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, August 1995.

[6] 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.

[7] 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.

[8] R. Gharpurey and S. Hosur, "Transform domain techniques for efficient extraction of substrate parasitics," in *Proc. IEEE International Conference on Computer Aided Design*, November 1997, pp. 461–467.

[9] M. Chou, *Fast Algorithms for Ill-Conditioned Dense-Matrix Problems in VLSI Interconnect and Substrate Modeling*, Ph.D. thesis, Massachusetts Institute of Technology, 1998.

[10] E. Charbon, R. Gharpurey, R.G. Mayer, and A. Sangiovanni Vincentelli, "Semi-analytical techniques for substrate characterization in the design of mixed-signal ics," in *International Conference on Computer Aided-Design*, San Jose, CA, November 1996, pp. 455–462.

[11] J.R. Phillips and L.M. Silveira, "Simulation approaches for strongly coupled interconnect systems," in *Proc. IEEE International Conference on Computer-Aided Design*, San Jose, CA, 2001, pp. 430–437.

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

[13] L. Greengard and V. Rokhlin, "A fast algorithm for particle simulations," *Journal of Computational Physics*, vol. 73, no. 2, pp. 325–348, December 1987.

[14] V. Rokhlin, "Rapid solution of integral equation of classical potential theory," *J. Comput. Phys.*, vol. 60, pp. 187–207, 1985.

[15] J. E. Barnes and P. Hut, "A hierarchical $O(N \log N)$ force-calculation algorithm," *Nature*, vol. 324, no. 6270, pp. 446–449, 1986.

[16] 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.

[17] 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.

[18] 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.

[19] E. Michielssen and A. Boag, "A multilevel matrix decomposition algorithm for analyzing scattering from large structures," *IEEE Transactions on Antennas and Propagation*, vol. 44, no. 8, pp. 1086–1093, August 1996.

[20] W. Hackbusch, "A sparse matrix arithmetic based on H-matrices, part I: Introduction," *Computing*, vol. 62, pp. 89–108, 1999.

[21] W. Hackbusch and B.N. Khoromskij, "A sparse H-matrix arithmetic. part ii: Application to multi-dimensional problems," *Computing*, vol. 64, pp. 21–47, 2000.

[22] 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.

[23] J. Kanapka and J. White, "Highly accurate fast methods for extraction and sparsification of substrate coupling based on low-rank approximation," in *Proc. IEEE International Conference on Computer Aided Design*, November 2001, pp. 417–423.

[24] L.N. Trefethen and D. Bau, *Numerical Linear Algebra*, SIAM, Philadelphia, 2000.

[25] G.H. Golub and C.F. Van Loan, *Matrix computations*, Johns Hopkins University Press, Baltimore, 1996.

[26] R. Hockney, "A fast direction solution of poisson's equation using fourier analysis," *J. Assoc. Comput. Mach.*, vol. 12, pp. 95–113, 1965.

[27] P. Swarztrauber, "A direct method for the discrete solution of separable elliptic equations," *SIAM Journal on Numerical Analysis*, vol. 11, pp. 1136–1150, 1974.

[28] K. R. Rao, *Discrete Cosine Transform: Algorithms, Advantages, and Applications*, Academic Press, Inc., San Diego, CA., 1990.

[29] F. A. Kamangar and K. R. Rao, "Fast algorithms for the 2-d discrete cosine transform," *IEEE Transactions on Computers*, vol. C-31, no. 9, pp. 899–906, September 1982.

[30] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, Cambridge, 1993.

[31] W.L. Briggs, *A Multigrid Tutorial*, SIAM, Philadelphia, 1987.

[32] W. Hackbusch, *Multi-Grid Methods and Applications*, Springer-Verlag, Berlin Heidelberg New York Tokyo, 1985.

[33] A. Brandt, "Multi-level adaptive solutions to boundary-value problems," *Mathematics of Computation*, vol. 31, no. 138, pp. 333–390, April 1977.

[34] A. Brandt, "Guide to multigrid development," in *Multigrid Methods. Proceedings of the Conference Held at Koln-Porz, November 23-27, 1981*, W. Hackbusch and U. Trottenberg, Eds., pp. 220–312. Springer-Verlag, Berlin Heidelberg New York, 1982.

[35] Ranjit Gharpurey, *Modeling and Analysis of Substrate Coupling in Integrated Circuits*, Ph.D. thesis, Department of Electrical Engineering and Computer Science, University of California at Berkeley, Berkeley, CA, June 1995.

[36] J. P. Costa, M. Chou, and L. M. Silveira, "Precorrected-DCT techniques for modeling and simulation of substrate coupling in mixed-signal IC's," in *Proc. IEEE International Symposium on Circuits and Systems*, May 1998, vol. 6, pp. 358–362.

[37] A. Brandt and A. A. Lubrecht, “Multilevel matrix multiplication and fast solution of integral equations,” *Journal of Computational Physics*, vol. 90, pp. 348–370, 1990.

[38] H. Haus and J. Melcher, *Electromagnetic Fields and Energy*, Prentice Hall, Englewood Cliffs, N.J., 1989.

[39] G. Strang and T. Nguyen, *Wavelets and Filter Banks*, Wellesley-Cambridge Press, Wellesley, 1996.

[40] S. Kapur and D. Long, “Large-scale capacitance calculation,” in *Proc. 37th Design Automation Conference*, June 2000, pp. 744–749.

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

[42] 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.

[43] S. Borm and W. Hackbusch, “Data-sparse approximation by adaptive $\mathrm{H}^2$-matrices,” Tech. Rep. 86, Max-Planck-Institut fur Mathematik, Leipzig, Germany, 2001.

[44] N. Yarvin and V. Rokhlin, “A generalized one-dimensional fast multipole method with application to filtering of spherical harmonics,” *Journal of Computational Physics*, vol. 147, no. 2, pp. 594–609, December 1998.

[45] N. Yarvin and V. Rokhlin, “An improved fast multipole algorithm for potential fields on the line,” *SIAM Journal on Numerical Analysis*, vol. 36, no. 2, pp. 629–66, 1999.

[46] E. Michielssen, A. Boag, and W.C. Chew, “Scattering from elongated objects: direct solution in $O(n \log^2 n)$ operations,” *IEE Proc.-Microw. Antennas Propag.*, vol. 143, no. 4, pp. 277–283, August 1996.