

Large-Scale Capacitance Calculation

Sharad Kapur David E. Long

November 1, 1999

Abstract

We describe a new method for accurate large-scale capacitance calculations. The algorithm uses an integral equation formulation, but with a new representation for charge distributions that decouples charge variation from conductor geometry. This separation significantly reduces the problem size compared to a traditional discretization, resulting in a large speed increase. The full capacitance matrix of typical interconnect problems with thousands of nets can be computed in a few hours.

1 Introduction

We discuss a new method for capacitance calculation based on solving the integral equation:

$$\phi(r) = \int_R G(r, r') \rho(r') dr'.$$

Here ϕ is the potential, ρ is a surface charge density, G is the Green's function giving the potential for a unit point charge, and R ranges over the conductor surfaces. For capacitance computation, the potential is known (one volt on the selected conductor and zero volts elsewhere), and the problem is to find ρ . The capacitance is obtained by integrating ρ over the conductor surfaces.

The standard approach for discretizing the integral equation is to cut the conductor surfaces into elements, with the unknown ρ assumed to be a low-order polynomial over each element. Enforcing the equation either at a set of collocation points or with a Galerkin scheme leads to a dense linear system. Because these systems are large, they are usually solved with Krylov iterative methods [7]. The required matrix-vector products are accelerated

with compressed-matrix schemes. Examples of previous work following this strategy include versions of FastCap [5, 6], IES³ [4], and the method of Shi et al. [8].

A weakness of the previous work is that the discretization tends to be dictated by the problem geometry rather than by the charge variation. This can be seen by simply plotting the charge density solutions obtained by methods such as the above; an example is shown in figure 2. There is a strong variation in the charge on the selected conductor and a smooth variation on the other conductors. When the geometry is complicated, the need to capture it results in problems with millions of unknowns, which overwhelms even the fastest solvers. Our innovation is a representation for charge distributions that decouples the charge variation from geometry. This allows us to capture the smoothly varying parts of the solution with only a few numbers, regardless of how complex the geometry is. Hence, we can greatly reduce the discretization size, and the time and memory requirements, compared to the previous approaches.

Our new algorithm, which we call Nebula, uses this new representation together with an iterative linear solve and a new kernel-independent version of the Fast Multipole Method (FMM) [3]. Nebula is efficient enough to compute the full capacitance matrix of typical interconnect problems with thousands of nets in a few hours.

2 The Key Idea

Consider solving for the capacitance of conductor C_1 in figure 1. The surface charge density obtained when C_1 is at one volt and C_2 and C_3 are grounded is shown in figure 2. Note that the charge density on C_2 and C_3 is much smoother than the density on C_1 . In our new representation, we take advantage of this observation by approximating the solution on C_3 using a low-order polynomial that is *projected on the conductor geometry*. Even if the geometry of C_3 were significantly more complicated, this would still be possible. In contrast, the traditional approach chops C_3 into pieces and uses a low-order polynomial on each piece. The reason for the discretization is to describe the geometry, not to capture the charge variation. With the standard representation, the number of unknowns grows as the geometry becomes more complicated, while in our new decoupled representation, the amount of information needed is independent of geometry.

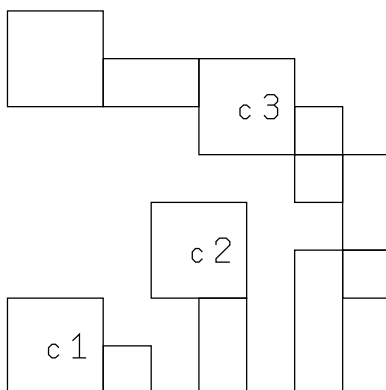


Figure 1: Three conductors

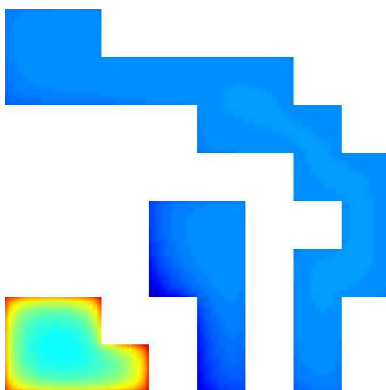


Figure 2: The solution charge density for distant conductors is smooth

Of course, we must still be able to compute the influence of the charge that is present on C_3 . If this computation requires accessing the full geometry description, we can only save memory, not time. Fortunately, *all the details of C_3 's geometry are not critical*. As an example, if we replace C_3 by the conductor C'_3 as in figure 3 and then apply one volt to C_1 , the charge on C_1 and C_2 changes by less than one percent. We can make this replacement rigorous by using the idea of *matching geometric moments*. The important characteristic of the replacement C'_3 is that it maintains the general shape of C_3 ; it has the same area and roughly the same distribution of area.

Our new Nebula representation for charge distributions uses these two ideas.

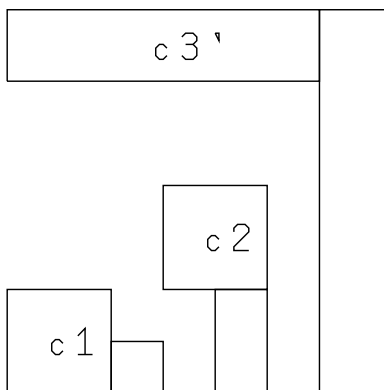


Figure 3: C_3 approximated

- Charge density is given by a smooth function that is projected onto geometry.
- Distant conductors are approximated with by matching geometric moments when computing potential from charge.

3 Mathematical Preliminaries

Before describing the new representation, we establish some notation. Let f and g be polynomials with real coefficients. We write $\langle \cdot, \cdot \rangle$ for an inner product on polynomials. When $\langle f, g \rangle = 0$, f and g are orthogonal. A set of polynomials $\{f_1, f_2, \dots\}$ is orthonormal if $\langle f_i, f_j \rangle = 0$ for $i \neq j$ and $\langle f_i, f_i \rangle = 1$. The inner product for polynomials that we will be using is

$$\langle f, g \rangle = \int_{-1}^1 f(x)g(x) dx.$$

The Legendre polynomials, appropriately scaled, are orthonormal under this inner product and will be denoted $\{l_i(x)\}$.

A smooth function h on $[-1, 1]$ can be expressed by an infinite series

$$h = \sum_{i=0}^{\infty} \langle h, l_i \rangle l_i.$$

The numbers $\langle h, l_i \rangle$ are called the moments of h . To approximate h , the series can be truncated at a finite number of terms. We shall also be interested in

the case where h is discontinuous at a finite set of points, or even when h is a delta function. In these cases, the series converges in a distributional sense: if f is a polynomial on $[-1, 1]$, then

$$\int_{-1}^1 fh = \lim_{n \rightarrow \infty} \int_{-1}^1 f \sum_{i=0}^n \langle h, l_i \rangle l_i.$$

Orthonormal polynomials over three dimensional regions can be formed from products of one dimensional orthonormal polynomials. We write

$$f(r) = \sum_{i=0}^{\infty} \langle f(r), l_i(r) \rangle l_i(r),$$

where $r = (x, y, z)$ and the l_i are products of one-dimensional polynomials.

Let R be a set of conductor surfaces contained in a three dimensional volume V , and let f be any function with domain V . The characteristic function χ_R of R is the (generalized) function for which:

$$\int_R f = \int_V f \chi_R.$$

As an example, if R is the plate $[0, 1] \times [0, 1]$ in the xy -plane, then

$$\chi_R(x, y, z) = s(x)s(y)\delta(z),$$

where $\delta(z)$ is the Dirac delta function, and $s(x)$ is 1 for $x \in [0, 1]$ and 0 elsewhere. The product $f\chi_R$ is called the projection of f onto R .

4 The Nebula Representation

The Nebula representation for the charge density in the volume V consists of:

- a polynomial f defining the charge variation; and
- moments of the function χ_R .

We require two main operations on charge distributions. The first is a (semi-definite) inner product over conductor surfaces:

$$\langle f, g \rangle_R = \int_R fg = \int_V f \chi_R g. \tag{1}$$

The inner product is used for orthogonalization during the iterative solution of the discretized system. The second operation is the computation of a potential distribution from a charge distribution:

$$\phi(r) = \int_R G(r, r') f(r') dr' = \int_V G(r, r') f(r') \chi_R(r') dr'.$$

where G is the Green's function.

Suppose that both f and g are given by p th-order moment series. The two operations above involve the product $f\chi_R$. We can calculate p th-order moments for $f\chi_R$ using the following theorem.

Theorem 1 *Suppose that R is an arbitrary set of conductor surfaces, and f is the p th-order expansion*

$$f(r) = \sum_{i=0}^N \langle f(r), l_i(r) \rangle l_i(r),$$

where l_0, \dots, l_N are the orthonormal polynomials of order at most p . Then the p th-order expansion of the product $f\chi_R$ can be computed exactly using only a $2p$ th-order expansion for χ_R .

The proof is simply to observe that the product of f with a basis function l_i of order at most p is a polynomial of degree at most $2p$, and hence any moments of χ_R with order greater than $2p$ are orthogonal to this polynomial.

Once we calculate the p th-order moments for $f\chi_R$, then the inner product $\langle f, g \rangle_R$ is determined by a standard dot product with the moments of g . Because g has degree at most p and the p th-order moments of $f\chi_R$ are computed exactly, the inner product is also exact. The potential calculation involves an approximation. Within V and in regions close to V , the effect is that the geometry has been replaced by an approximate one that has the same low-order moments, as in figure 3. Different levels of approximation to χ_R for the set of three conductors of figure 1 are shown in figure 4. Beyond a distance proportional to the size of V , the error from approximating $f\chi_R$ with a p th-order moment series decreases exponentially with p .

If the moments in the expansion f are written as a vector, then the vector of moments for $f\chi_R$ can be obtained by applying a positive semidefinite linear transformation. The matrix for this linear transformation, which we denote by P_R , is the matrix whose (i, j) entry is

$$\sum_{k=0}^{N'} \langle \chi_R, l_k \rangle \int_V l_i l_j l_k,$$

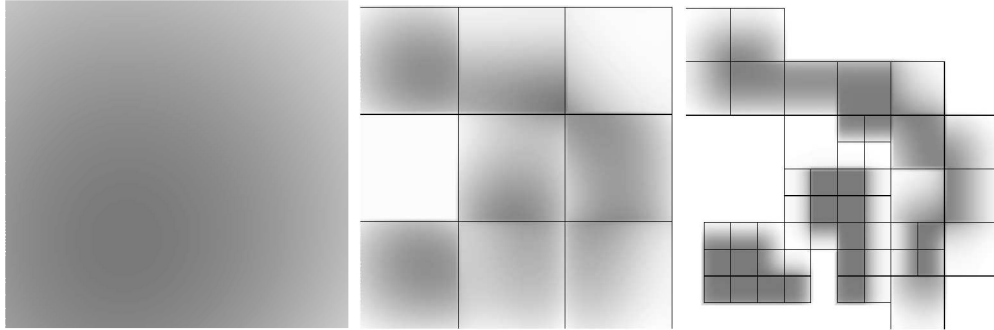


Figure 4: Increasingly accurate approximations for the conductors in figure 1

where $l_0, \dots, l_{N'}$ are of order up to $2p$.

5 Capacitance Calculation with Nebula

The Nebula algorithm to solve for the capacitance of a give conductor is as follows. We recursively subdivide the space containing the full geometry into boxes to isolate the selected conductor and any nets in its immediate neighborhood (see figure 5). In boxes containing the selected conductor and in adjacent boxes, we use a traditional Galerkin discretization (to avoid obscuring the selected conductor, the figure does not show these boxes). The new representation is used for boxes separated from the selected conductor (the boxes shown in the figure). For each Nebula box, we calculate moments of the geometry and form the matrix P_R . Note that Nebula is used for larger parts of the geometry as we get farther from the selected conductor. We run an iterative linear solver to find the charge distribution that results in a potential of one volt on the selected conductor and zero volts elsewhere. Finally, integrating the charge density on each conductor gives the column of the capacitance matrix corresponding to the selected net. We now describe the operations required in more detail.

The iterative solve is a Krylov algorithm based on GMRES [7]. For Nebula boxes, the unknowns are the p th-order moments of the charge density. Orthogonalization for these unknowns uses the inner product $\langle \cdot, \cdot \rangle_R$ (equation 1) rather than the standard dot product of vectors. In those boxes with a standard representation, we take P_R to be an identity matrix and $\langle \cdot, \cdot \rangle_R$ to be the usual dot product. We write the norm corresponding to the inner product



Figure 5: A selected conductor (highlighted) and Nebula boxes of the discretization

as

$$\|f\|_R = \sqrt{\langle f, f \rangle_R}.$$

The process of computing a potential distribution from a charge distribution (after projection onto the geometry) will be denoted by the linear operator M . Note that the charge in a box affects the potential in all boxes. Hence M is a global operator, unlike P_R which is applied on a box-by-box basis. Fast application of M is discussed below. Let ψ be the desired potential distribution (one on the selected conductor, zero elsewhere). We want to find f satisfying

$$\|MP_R f - \psi\|_R = 0.$$

That is, if we project f onto the geometry and compute the potential from the resulting charge distribution, the result should agree with ψ on the conductor surfaces. Equivalently,

$$P_R M P_R f = P_R \psi.$$

The solution procedure is shown in figure 6; the difference from standard GMRES is the inner product.

For computing a potential distribution

$$\phi(r) = \int_R G(r, r') f(r') dr',$$

from the charge distribution f , we use a variant of the Fast Multipole Method (FMM) [3]. The FMM computes pointwise potentials from a set of point charges in space with a standard $1/|r - r'|$ Green's function. Since the input and desired output with Nebula are charge and potential distributions, we will call the modified algorithm the Fast Distribution Method (FDM). The

```

 $f_1 := \psi$ 
 $\beta := \|\psi\|_R$ 
 $f_1 := f_1/\beta$ 
for  $j := 1, 2, \dots$ 
     $f_{j+1} := MP_R f_j$ 
    for  $i := 1, 2, \dots, j$ 
         $h_{i,j} := \langle f_i, f_{j+1} \rangle_R$ 
         $f_{j+1} := f_{j+1} - h_{i,j} f_i$ 
    end
     $h_{j+1,j} := \|f_{j+1}\|_R$ 
     $f_{j+1} := f_{j+1}/h_{j+1,j}$ 
    solve the  $(j+1) \times j$  least-squares system  $Hx = \beta e_1$ ,
        where  $H$  is the Hessenberg matrix with  $(i, j)$  entry  $h_{i,j}$ 
    let  $r$  be 2-norm of the least-squares residual
     $f := \sum_{i=1}^j x_i f_i$ 
    if  $r/\beta$  is sufficiently small, stop ( $f$  is the solution)
end

```

Figure 6: Pseudo-code to solve for the charge density using GMRES with inner product $\langle \cdot, \cdot \rangle_R$

main difference between the FMM and the FDM is that the FDM kernel can be an arbitrary layered-media Green's function [1]. There are other minor differences. For example, the FDM uses moment series based on Legendre polynomials to represent distributions instead of the multipole and local expansions that are appropriate for a Laplace kernel.

Like the FMM, the FDM recursively subdivides space into boxes. We use the same subdivision discussed at the start of this section. The algorithm works by summarizing charge distributions upward through the tree of boxes, computing box-to-box interactions throughout the tree, and then interpolating potential distributions downward to the leaves. Interactions between higher-level boxes are used to propagate far-field interactions between widely separated leaves.

Kernel independence is achieved by having different charge-to-potential interaction matrices (translation operators in FMM terminology) depending on the size and position of the interacting boxes. The number of possible

interaction matrices is small because layered media Green's functions are translation invariant in the horizontal direction and because we use a fixed set of discrete box sizes. Given a description of the layered medium, all the possible interaction matrices are computed once in a preprocessing step.

6 Implementation Notes

There are two main costs associated with the capacitance calculation. The first is in computing the Galerkin interaction coefficients between pairs of polygons for the part of the problem done with a traditional discretization. We use an ad hoc mixture of interpolation, moment expansions, analytic forms, and Gaussian quadratures to reduce this cost, but it still requires about one-half to one-third of the total time. The other main cost is the FDM. Like the FMM, the computational cost of the FDM is linear in the number of leaf boxes. Still, the time requirements can be large, mainly because of the many interactions between each box and those in its neighborhood.

In the context of the FMM, a succession of increasingly clever schemes have been proposed to reduce this cost, culminating in a method to diagonalize the interaction matrices [3]. We use the same essential idea to accelerate the FDM: change the basis describing the charge and potential distributions so that each interaction becomes cheap. Because we do not specialize to a particular Green's function, we cannot diagonalize the interaction matrices, but we can significantly reduce their size by using an SVD to compress the interactions. By concatenating the possible interaction matrices for a box and doing a single SVD, we can get a good basis for the charge distribution for doing *all* the interactions of that box. Similarly, we can get a good basis for the potential within a box by computing an SVD of the concatenation of all interaction matrices that could contribute to it. With these new bases, the individual interaction matrices can be significantly sparsified. Let U and V^T be the matrices used to change basis for the potential and the charge respectively. In essence, we are using a low-rank approximation of the form $U_k P_k V_k^T$ to the k th interaction matrix, where the columns of U_k and V_k are subsets of the columns of U and V . Interaction computation then consists of:

1. applying V^T to the moment coefficients for the charge (changing the basis);

2. applying a series of small P_k matrices and accumulating the results; then
3. applying U to change back to the basis of standard moment coefficients for the potential.

We also exploit symmetries to further reduce the time requirements.

When solving for the capacitance of many nets, we wish to avoid recomputation where feasible. Many of the boxes produced for calculating the capacitance of one net may also be necessary when solving for the capacitance of a nearby net. Hence there may be duplicate computation of some direct interaction matrices. We use caching to avoid the duplication and try to order the conductors according to their spatial location so that the cache is of maximum effectiveness.

We use a block diagonal preconditioner to speed convergence of the iterative linear solve. For Nebula boxes, we must take the possible rank deficiency of P_R into account. Let M be the self-interaction matrix for the box. We compute the SVD of P_R and truncate it when the singular values have dropped by a factor of about 100. Let VSV^T be the truncated SVD. The operator that we want to “invert” is MP_R , which is approximated by $MVSV^T$. The preconditioner will just invert the part of the space spanned by the columns of V and leave the rest unchanged. Projecting onto V gives $V(V^T MVS)V^T$; the matrix $V^T MVS$ is well-conditioned and can be inverted. The remainder of the space can be obtained by $I - VV^T$, so the complete preconditioner is:

$$I + V((V^T MVS)^{-1} - I)V^T.$$

7 Examples

All examples were run on a 400MHz Sun UltraSPARC workstation. Tolerances were set to give an accuracy of 3% in the calculated net capacitances. The FDM and the Nebula representation used 3rd-order moment series (20 basis functions per series). Times do not include layout processing: the input to Nebula is a set of polygons, each labeled with a net number. All other activities, from reading the input to printing the output, are included. Table 1 summarizes the results for all the examples in this section.

To validate Nebula, we compared the results to those from a commercial capacitance calculator, denoted X, on two moderate-sized examples. X is based on stochastic methods [2]. The examples were:

Example	Polygons	Nets	Program	Time	Memory
Validation 1	93,000	118	Nebula	9.5 min	45 MB
			X, tol 5%	3 min	—
			X, tol 2%	7.5 min	—
			X, tol 1%	20 min	—
			X, tol 0.5%	70 min	—
Validation 2	260,000	838	Nebula	44 min	160 MB
			X, tol 10%	3.3 hr	—
			X, tol 5%	3.8 hr	—
			X, tol 2%	4 hr	—
RF chip	530,000	123	Nebula	37 min	350 MB
Digital example	680,000	3426	Nebula	4.7 hr	530 MB

Table 1: Summary of time and memory requirements

1. a portion of an analog circuit, using a four-level metal 0.25 micron BiCMOS process, with 118 nets; and
2. a piece of digital logic with 838 nets, fabricated in a four-level metal 0.25 micron CMOS process.

X was the only program we had access to that could solve these problems. Previous integral equation tools [4, 5, 6, 8] cannot handle problems of even this moderate size.

X calculates capacitances of the form $c \pm e$, where c is the value and e is an uncertainty representing one standard deviation. With 68% confidence, the true capacitance value lies between $c - e$ and $c + e$. The program stops when the relative magnitude e/c for each diagonal value of the capacitance matrix falls below a user-specified threshold. We ran the program at various tolerances and compared the results with Nebula.

Time and memory for the tools is summarized in table 1. X does not report memory sizes. For example one, histograms showing the norm of the differences in individual columns in the capacitance matrix compared to Nebula are shown in figure 7. When X is run at a tolerance of 0.5%, the two tools agree on all nets to within 5%. For Nebula, power and ground nets account for about two minutes of the time and for the peak memory use; all other nets require just over 6.5 minutes and 18 MB. (These times add up to

less than 9.5 minutes since they do not include reading, printing, and other miscellaneous processing that is not associated with individual nets.)

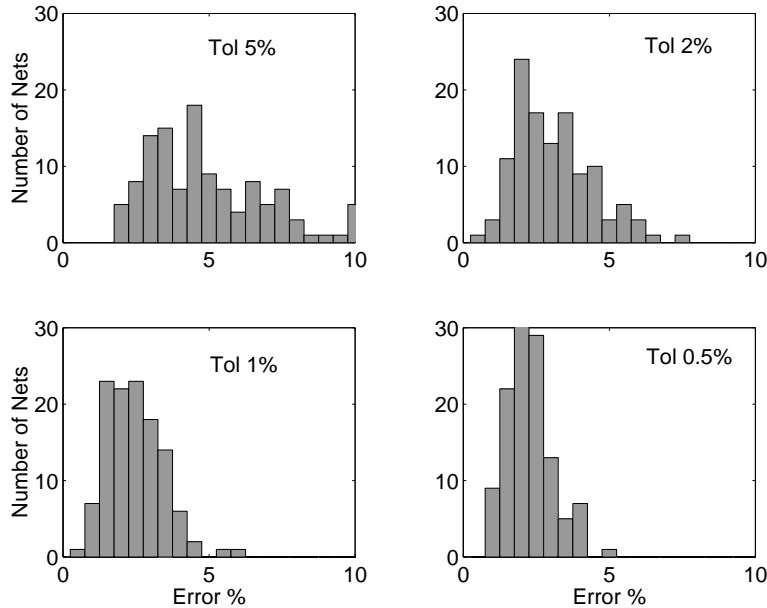


Figure 7: Error histograms for validation example one

Error histograms comparing the tools for example two are shown in figure 8. Unfortunately, time constraints prevented us from running X at higher tolerances, but the answers agree with Nebula on almost all nets to within 5% at a tolerance of 2%. Nebula required 4.5 minutes/125 MB and 6 minutes/160 MB for the power and ground nets, respectively. All other nets took 33 minutes and 18 MB.

We also ran two larger examples. The first is an RF chip: a precision quadrature generator and I/Q mixer chip designed as part of an up-conversion modulator. It was fabricated in a four-level metal 0.25 micron BiCMOS process and is 1.3 mm on a side. Nebula took 6 minutes and 100 MB for the power net, 21 minutes and 350 MB for the ground net, and 7.5 minutes and 14 MB for all other nets. The second example is a part of a digital chip, 0.4 mm by 0.55 mm, using a 0.25 micron four-level metal CMOS process; layout is shown in figure 9. Peak memory use of 530 MB was for the ground net, and total time for the power, ground, and clock nets was one hour. The remaining nets took 3.7 hours and a peak memory of 43 MB.

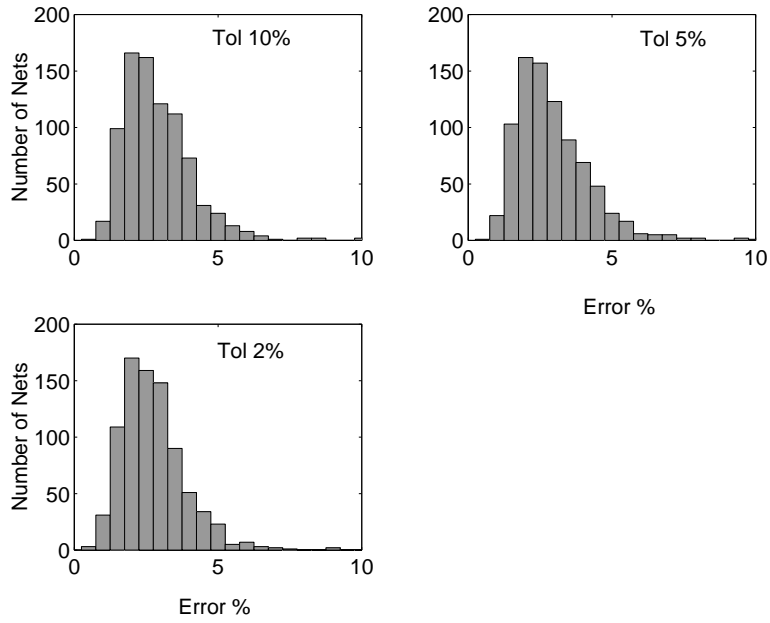


Figure 8: Error histograms for validation example two

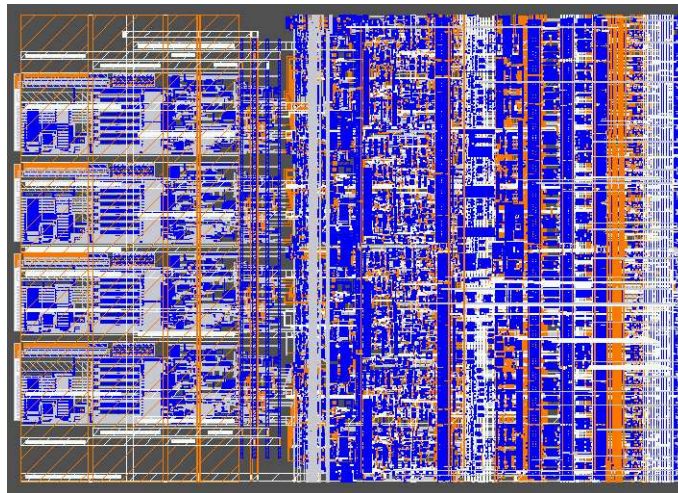


Figure 9: Layout for digital example

Figure 11 is a scatter plot showing total memory requirements versus number of unknowns. The data points are taken from all the nets in all the examples described in this section. As expected, memory increases linearly with the number of unknowns. Problems with one million unknowns require about 400 MB of memory. The time for doing a potential computation with the FDM are given in figure 10. Time also increases linearly with the number of unknowns. FDM computations with one million unknowns take about 20 seconds.

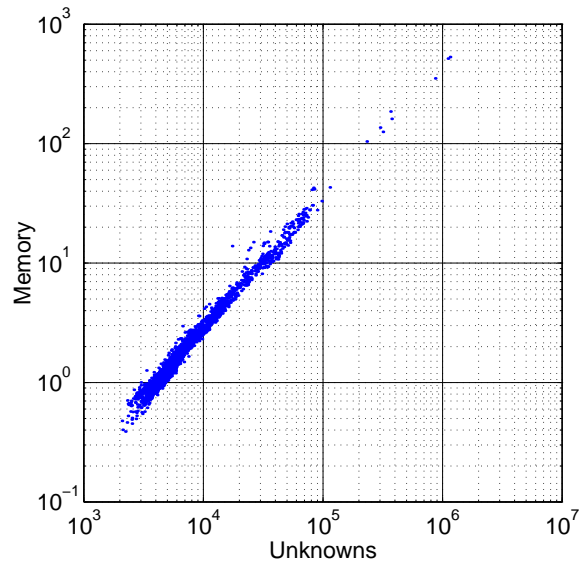


Figure 10: Total memory in MB versus number of unknowns

Memory requirements with Nebula are clearly dominated by the largest nets. Nebula is most useful when nets are reasonably localized; the direct interaction cache works well in these cases and the solve progresses at an average rate of one net every two to three seconds. Nebula is most useful in an RC modeling context, where very large nets such as power and ground have been cut up into segments.

8 Conclusion

Nebula uses a new representation to capture the far-field in a capacitance calculation efficiently. The efficiency of the discretization is dictated by solu-

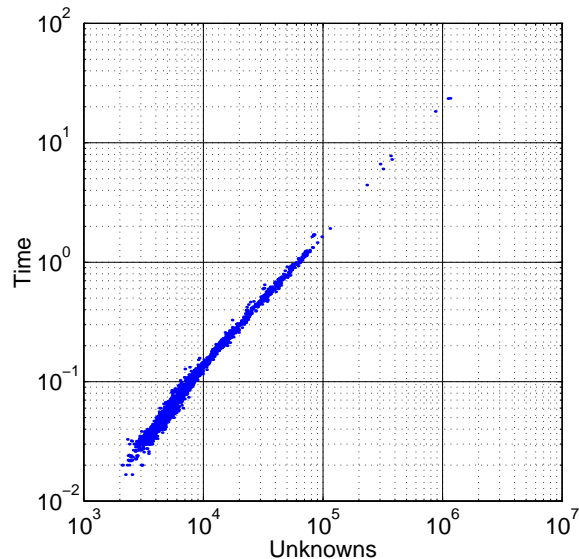


Figure 11: FDM time in seconds versus number of unknowns

tion smoothness rather than geometric complexity. Most of the information required to use the Nebula representation, including all singular integrals, can be precomputed. Nebula uses a new version of the FMM that is kernel-independent, allowing it to handle layered Green's functions efficiently. We demonstrated the efficiency of the new representation by showing that capacitances for significant pieces of layout with thousands of nets can be computed in reasonable time and memory.

References

- [1] W. C. Chew. *Waves and Fields in Inhomogeneous Media*. IEEE Press, 1995.
- [2] Y. L. Le Coz and R. B. Iverson. A stochastic algorithm for high speed capacitance extraction in integrated circuits. *Solid State Electronics*, 35(7):1005–1012, July 1992.
- [3] L. Greengard and V. Rokhlin. A new version of the fast multipole method for the Laplace equation in three dimensions. *Acta Numerica*, 6:229–269, 1997.

- [4] S. Kapur and D. E. Long. IES³: Efficient electrostatic and electromagnetic simulation. *IEEE Computational Science and Engineering*, 5(4):60–67, October–December 1998.
- [5] K. Nabors and J. K. White. FastCap: A multipole accelerated 3-D capacitance extraction program. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 10(11):1447–1459, November 1991.
- [6] J. R. Phillips and J. K. White. A precorrected-FFT method for electrostatic analysis of complicated 3-D structures. *IEEE Transactions on Computer -Aided Design of Integrated Circuits and Systems*, 16(10):1059–1072, October 1997.
- [7] Y. Saad and M. H. Shultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, July 1986.
- [8] W. Shi, J. Liu, N. Kakani, and Y. Tiejun. A fast hierarchical algorithm for 3-D capacitance extraction. In *Proceedings of the 1998 Design Automation Conference*, pages 212–217, June 1998.