A 3-D CAPACITANCE EXTRACTION ALGORITHM BASED ON KERNEL INDEPENDENT HIERARCHICAL METHOD AND GEOMETRIC MOMENTS

A Thesis

by

WEI ZHUANG

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2006

Major Subject: Computer Engineering

A 3-D CAPACITANCE EXTRACTION ALGORITHM BASED ON KERNEL INDEPENDENT HIERARCHICAL METHOD AND GEOMETRIC MOMENTS

A Thesis

by

WEI ZHUANG

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Weiping Shi
Committee Members,	Peng Li
	Donald Friesen
Head of Department,	Costas N. Georghiades

May 2006

Major Subject: Computer Engineering

ABSTRACT

A 3-D Capacitance Extraction Algorithm Based on Kernel

Independent Hierarchical Method and Geometric Moments. (May 2006)

Wei Zhuang, B.S., Xi'an JiaoTong University;

M.S., Xi'an JiaoTong University

Chair of Advisory Committee: Dr. Weiping Shi

A three dimensional (3-D) capacitance extraction algorithm based on a kernel independent hierarchical method and geometric moments is described. Several techniques are incorporated, which leads to a better overall performance for arbitrary interconnect systems. First, the new algorithm hierarchically partitions the bounding box of all interconnect panels to build the partition tree. Then it uses simple shapes to match the low order moments of the geometry of each box in the partition tree. Finally, with the help of a fast matrix-vector product, GMRES is used to solve the linear system. Experimental results show that our algorithm reduces the linear system's size greatly and at the same time maintains a satisfying accuracy. Compared with FastCap, the running time of the new algorithm can be reduced more than a magnitude and the memory usage can be reduced more than thirty times.

ACKNOWLEDGEMENTS

This thesis represents about two years of work at Texas A&M University. This work would not have been possible without the boundless assistance of mentors, colleagues, and friends.

It is with the deepest of gratitude that I would like to thank my advisor, Weiping Shi, who always gave me great directions. I would also like to express my heartfelt thanks to the other professors, who have always kept their doors open, ready to discuss and encourage new ideas. Particular thanks go to Peng Li and Donald Freisen who could always find time, despite any other responsibilities or deadlines they may have had.

In my time at Texas A&M University, I have had the opportunity to collaborate with a number of different people without whose help this thesis could not have been completed. I would like to thank Jiang Hu, Duncan M. (Hank) Walker, Sunil P. Khatri, Yan Shu, Xiang Lu, Ying Zhou, Ziding Yue, Zhuo Li, Yuxin Tian, Yang Yi, and Qiuyang Li for their time and their efforts. While not a collaborator as such, I would also like to thank Tammy Carda and Linda Currin for their regular guidance and tireless assistance.

Finally, I would like to express my sincere thanks to friends and family whose unwavering support has been indispensable to me over the last years. Thank you to my family: Ting Gu, Daughter, Mom, Dad, and my elder sister. And thank you to the friends that I have made along the way: Bill, Xiaoqun, Bo Wang, Xiquan Gao, Xiuquan, Qiuyang and Mankang.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

CHAPTER I

INTRODUCTION

With rapid increase of device density, the parasitic capacitance associated with interconnection has become a very important factor in determining the performance of the final circuits. Besides,due to the progresses in MCM, high density packaging and PCB technologies, it's necessary to accurately calculate the capacitance of complicated three-dimensional (3-D) structures to ensure sufficient switching speeds or other desired functionalities. Two typical examples of complicated three-dimensional structures, for which capacitance strongly affects performance, are dynamic memory cells and the chip carriers commonly used in high-density packaging. All these reasons have increased interest in computationally efficient procedures for determining capacitances of general three-dimensional structures.

The capacitance extraction is made tractable by assuming the conductors are ideal and embedded in a piecewise-constant dielectric medium. Then to compute the capacitance, Laplace's equation is solved numerically over the charge-free region with the conductors providing boundary conditions

$$\phi(r) = \int_R G(r, r')\rho(r')dr'. \qquad (1)$$

Here ϕ is the potential, R is the surfaces of the conductors, ρ is the surface charge density, dr' is the incremental conductor surface area, and the integral equation kernel G is the Green's function, which gives the potential for a unit charge.

This thesis follows the style of *IEEE Transactions on Microwave Theory and Techniques*.

The boundary element method (BEM) [1] is often used for solving the integral form Laplace equation. In this approach the surfaces of all conductors are broken into small panels and it is assumed that on each panel the charge density is a low-order polynomial. Enforcing the equation either at a set of collocation points or with a Galerkin scheme leads to a dense system. That is, the potential on each panel is computed by summing the contributions from all the panels using Green's functions. In this way, a matrix of potential coefficients, P, relating the set of n panel potentials and the set of n panel charges is constructed. And we get a dense linear system

$$Pq = v \qquad (2)$$

,where $q \in R^n$ is the vector of panel charges, and $v \in R^n$ is the vector of panel potentials. By transferring the integral form Laplace's equation into this linear system, we can compute the capacitance matrix of conductors.

The capacitance of m-conductor geometry can be summarized by an $m \times m$ capacitance matrix C [3]. Each diagonal entry C_{ii} is positive, representing the self-capacitance of conductor i; Each non-diagonal entries C_{ij} is negative, representing the coupling capacitance between conductors. To determine the j_{th} row of C, we compute the surface charges on all conductors produced by raising conductor j to one volt while grounding other conductors. That is, for equation (2), the potentials of panels on conductor j are one volt and zero elsewhere. Knowing P and v, we can solve equation (2) to get q, which is the vector of panel charges. Summing the contained panels' charges, we can get the surface charge of each conductor. To determine all the self and coupling capacitances of the m condcutors, the conductor surface charges must be computed m times, with m different sets of condcutor potentials.

To solve (2), direct methods based on triangularization, such as Gaussian elimination and Cholesky factorization, require $O(n^3)$ operations. Iterative algorithms normally require $O(n^2)$ operations per iteration. These approaches are inefficient if the number of panels is large, and limits the size of the problem that can be analyzed to one with a few conductors.

1.1 Outline

The algorithms presented in this thesis makes good use of the best of FastCap, HiCap, and Nebula, which is efficient for arbitrary interconnect configurations. It uses the same method as FastCap to partition the bounding box and get the FMM hierarchical tree; and then it uses an error approximation scheme to truncate corresponding sub-trees, which contains far away parts for a certain selected conductor. The final charges of all those tiles are calculated by GMRES [10] based on the leaf nodes of this truncated tree. Each iteration given the charges of all tiles, the potentials of these leaf nodes are computed and distributed to the contained tiles.

In Chapter II we will introduce the previous work. In Chapter III, we will show the detailed new algorithm and the complexity analysis. First, we show the main idea; second, we show how to define the geometric moments, group the far away tiles, and truncate the FMM tree; Finally, we analyze the complexity which shows that our algorithm's complexity is still $O(n)$, where n is the number of partitioned tiles. In Chapter V, we present numerical results demonstrating the performance of the proposed method.

1.2 Previous Work

People usually use Krylov iterative methods [10] to solve the linear system with various compression schemes for the required matrix-vector products. Several successful approaches to reducing the matrix-vector product computation time have been proposed.

1.2.1 FastCap

The FastCap algorithm [2] of Nabors and White has the complexity of $O(n)$ where n is the number of partitioned sections in the system. It is based on the fast multipole method (FMM) for the n-body problem designed by Greengard and Rohklin [9]. The multipole method, developed by Greengard and Rokhlin, exploits the fact that with increasing distance the interaction between well separated sets of point sources, such as charges, may be lumped together. This reduce the computational complexity of finding the potential for a given source distribution to linear time in terms of number of point sources, but without exceeding a given error bound. The far field potentials are expressed in multipole expansions first, and then transferred into local expansions, finally added to the potentials of different panels. People can trade between accuracy and speed with different expansion orders. Figure 1 shows the flow of FastCap. It has two passes. One is upward pass, which is used to compute the multipole expansion for each box. The other one is downward pass, which is used to transfer the multipole expansion into local expansion. In the end, the local expansion is evaluated and added to the nearby potentials, which are computed directly.

Initialization

Choose a number of levels so that there are, on average, s particles per box at the finest level.

Upward Pass

Begin at the finest level, and create multipole expansions from the source points and strengths. The expansions for all boxes at all higher levels are then formed by the merging procedure.

Downward Pass

From the coarsest level, we convert the multipole expansion into a local expansion about the centers of all boxes in its interaction list.

Finally, there is a local expansion in each box at each level. Beginning at the coarsest level, these local expansions are shifted to the children's level and added to the children's local expansions. After this recursive process reaches the finest level, a local expansion will have been created for each box, which describes the field due to all particles outside the box's neighbors. In the end, this expansion is evaluated and the nearby interactions are computed directly.

Fig. 1 FastCap algorithm.

Unfortunately, multipole method assumes that we are dealing with point sources. Therefore, in order to accurately capture near field effects a fine subdivision of conductors is necessary, which increases the computation time although the complexity is still $O(n)$ [†]. From the experiments of Nebula [8], we know there are strong charge density variations on the selected and nearby conductors, while smooth variations on far away conductors. Besides, the charge density reduces quickly with the distance increasing from the selected conductor. So it's unnecessary to subdivide far away conductors as fine as the selected conductor. In other words, it's unnecessary to compute the potentials of far panels in such a

fine level, because the errors of these far panels will be far below those of panels on the selected or nearby conductors. We can group these far panels to trade unnecessary accuracy for faster computation. Besides, the multipole expansion is designed for the kernel 1/r, so FastCap is kernel dependent, which means it, is hard to be used in multilayer case.

1.2.2 HiCap

Hierarchical refinement is another method to deal with the same class of problems to which parasitic extraction belongs, that is, the solution of Fredholm-type integral equation. This approach has been successfully applied to computationally expensive problems in physics and in computer graphics. Shi applied this approach to capacitance extraction in 1998 [3], where a set of conductors are given and the appropriate non-uniform sectioning is found by recursively subdividing the surfaces until the partition error drops below a given limit. The partition error is approximated by an error predictor function, which suggests that this error depends only on the potential matrix element between the two interacting sections and on their sizes. Applying the refinement procedure to a pair of panels will lead to a hierarchical partition tree for each panel, and each part of the panels is guaranteed to interact on a level as high as possible. If a set of N conductors are to be subdivided, the recursive procedure must be applied to each pair of conductors, leading to $O(N^2)$ time complexity for the data structure construction.

An example is shown in Figures 2 and 3. Two conductors A and B are partitioned according to the estimate of the potential coefficient. Figure (e) shows the final partition. We can see from it that the nearby parts of the two conductors are partitioned to a fine level, while the remote parts are partitioned to a coarse level. So HiCap in some meaning trades the unnecessary accuracy in far away parts for small problem size. The interaction matrix is

stored hierarchically which is shown in figure 3. The panels are stored as nodes in the tree and the coefficients are stored as links.

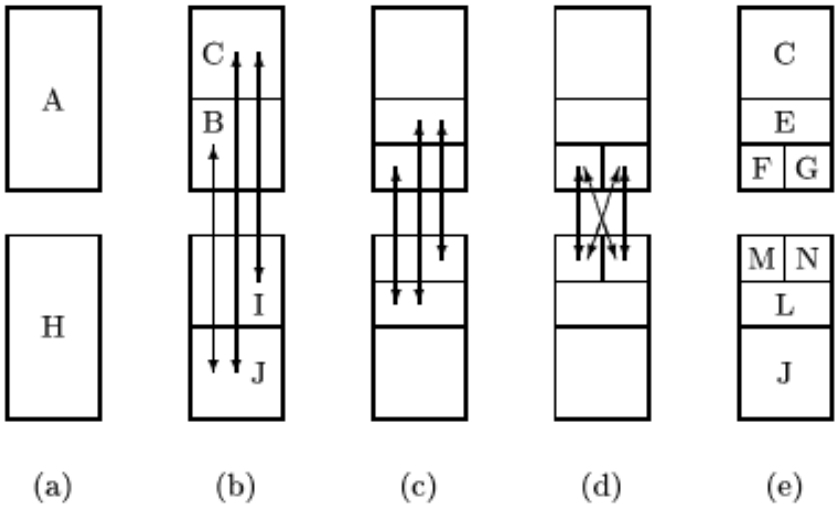


Fig. 2 Partition the conductor surfaces into panels.

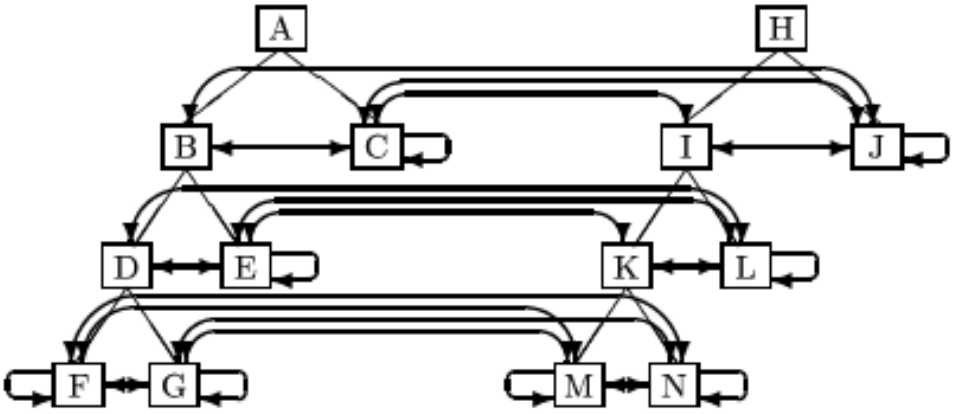


Fig. 3 Potential coefficients stored as links.

between nodes. The value of each coefficient is stored as a floating-point number associated with each link. Each tree represents one conductor surface, each non-leaf node

represents one panel further subdivided, and each leaf node represents one panel not further subdivided. The union of all the leaf nodes completely covers the surfaces of the conductors.

Hierarchical refinement works well in some cases, while detailed analysis shows that refinement does not maintain *O(N)* † complexity for systems where the average segment size is much smaller than the overall system size, such as for typical on-chip interconnect systems [7]. For these systems, if the accuracy requirements are chosen reasonably, the hierarchical refinement procedure will not lead to any refinement since most conductor pairs are far apart enough.

Analyzing the recursive refinement procedure of HiCap, which is used to subdivide large panels, we can see it decides whether to subdivide the larger one of two panels by judging whether the potential coefficient of these two panels is below a given precision limit. So HiCap subdivides the conductors according to the mutual potential coefficients between conductors, which are determined by the relative positions. This is different from FastCap, which partitions the conductors only by the geometry of each conductor. Experimental results show that in most cases HiCap is faster than FastCap because its partition of surfaces considers the potential coefficients' accuracy, which trades some unnecessary accuracy for speed. But it still has some shortcomings.

The first one is in refinement procedure HiCap considers the mutual potential coefficients very well, but forgets the self ones. When conductors are close to each other, this will not cause problems. But when some conductors are very far from the others, this will not lead to good partition on those far away conductors. See the following example

---

† N is the number of original segments in the system.

shown in figure 4, conductors 1 and 2 are very close to each other but 3 are very far from 1 and 2. HiCap will subdivide conductors 1 and 2 to a very fine level because they are very close and the potential coefficients of panels between them are large. For conductor3, HiCap even won't subdivide it because it is very far from other conductors, and the mutual potential coefficients between it and other conductors are very small. If conductor3 is not subdivided, when computing the capacitance column corresponding to it we will get a large error, especially for the self-capacitance of conductor3. If we want to subdivide conductor 3, we need to set a very small *peps*, but this will also subdivide conductors 1 and 2 to an unnecessary fine level, which causes the problem size very large.

The second shortcoming is the subdivision of HiCap usually causes unnecessary partition of conductors. Let's still see the set of conductors in figure5. When we compute the capacitance column corresponding to conductor3, there is a strong charge variation on conductor3 while a smooth and small charge variation on conductor 1 and 2. So we should use a fine partition level on conductor3 while a coarse level on 1 and 2 to get enough accuracy. But the HiCap gives an opposite partition. The reason is that the partition of HiCap only considers the mutual potential coefficient without considering the charge distribution. We know the potential caused by a point charge in distance r is

$$v = q / (4\pi\varepsilon \cdot r) \qquad (3)$$

Where q is the charge of the source and r is the distance from the source. So the potential is determined by not only distance by also charge. Only considering the distance of course will introduce errors and cause unnecessary problems size in some cases.

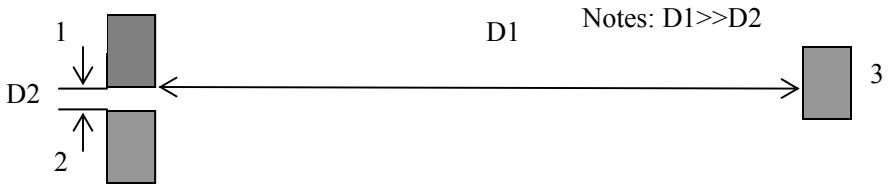


Fig. 4 A set of three conductors.

1.2.3 Nebula

Kapur and David proposed another approach [8], which uses a representation for charge distribution that decouples the charge variation from geometry. They turned it into a commercial software called Nebula and claimed it's efficient enough to compute the full capacitance matrix of typical interconnect problems with thousands of nets in a few hours.

The key idea of Nebula is to use a new representation for charge distributions that decouples charge variation from conductor geometry, which significantly reduces the problem size compared to a traditional partition and results in a large speed improvement. *Nebula* comes from the observation that there is a strong variation in the charge on the selected conductor while a smooth variation on the far conductors, and the partition of previous methods, such as, FMM, IES^3, HiCap, tends to be dictated by the problem geometry rather than by the charge variation. Nebula decouples the charge variation from geometry, which allows it to capture the smoothly varying parts with only a few numbers regardless of how complex the geometry is. This reduces the problem size and computation time greatly. In Nebula, the charge density, geometry, and potential distribution are described by expansions based on Legendre polynomials.

The following figure shows the main flow of Nebula algorithm.

Initialization

For a certain selected conductor, recursively subdivide the space containing the full geometry into boxes to isolate the selected conductor and any nets in its immediate neighborhood. In boxes containing the selected conductor and in adjacent boxes, it uses a traditional Galerkin discretization.

Upward Pass

For each Nebula box, it calculates moments of the geometry and forms the matrix P_R, which can be used to find the product of charge and geometry expansions according to P_R * *f*, where *f* is the charge expansion of the box.

Fast Matrix-Vector Product

It uses a similar scheme to FMM to compute the matrix-vector product, which is called FDM. It uses an upward pass to find the charge distribution of each box. Then it uses a downward pass to find the box-to-box interactions throughout the tree. Finally, it interpolates the potential distributions downward to the leaves. The interactions between pair of tiles for the part of the problem done with a traditional discretization are done directly.

After the final charge expansions are got for each box, integrating the charge on each conductor gives the capacitance column.

Fig. 5 Nebula algorithm.

Our experiments also show that, if replace a geometry with an approximated one which has the same 0^{th}, 1^{st}, and 2^{nd} geometric moments, the error from approximating the geometry decreases exponentially beyond a distance proportional to the size of the geometry. Normally, the error is <1% in all directions at the distance equal to the size of the geometry. The error in the near domain is not small, but we know the charge density is very small in far away parts, and this geometry approximation is only used on these far away parts, so the final total error will not exceed required accuracy limit.

Nebula is superior in the case of huge size problems, but suffers from re-partitioning and re-computing geometric moments for every selected conductor in the case of medium and small size problems. Besides, the computations of geometric moments are done by integration with the basis of Legendre polynomials, which takes much time.

1.2.4 Others

There also exist other efficient algorithms that are not based on the n-body problem, such as pre-corrected fast Fourier transform (FFT) [4] algorithm of Phillips and White, and the singular values decomposition (SVD) algorithm [5] of Kapur, both of which have the complexity of $O(n\log n)$. Le Coz and Iverson proposed a Monte Carlo algorithm [6] and successfully turned it into the popular commercial software QuickCap. Beattie and Pileggi combined the FMM with the hierarchical refinement method [7] to improve the speed further.

CHAPTER II

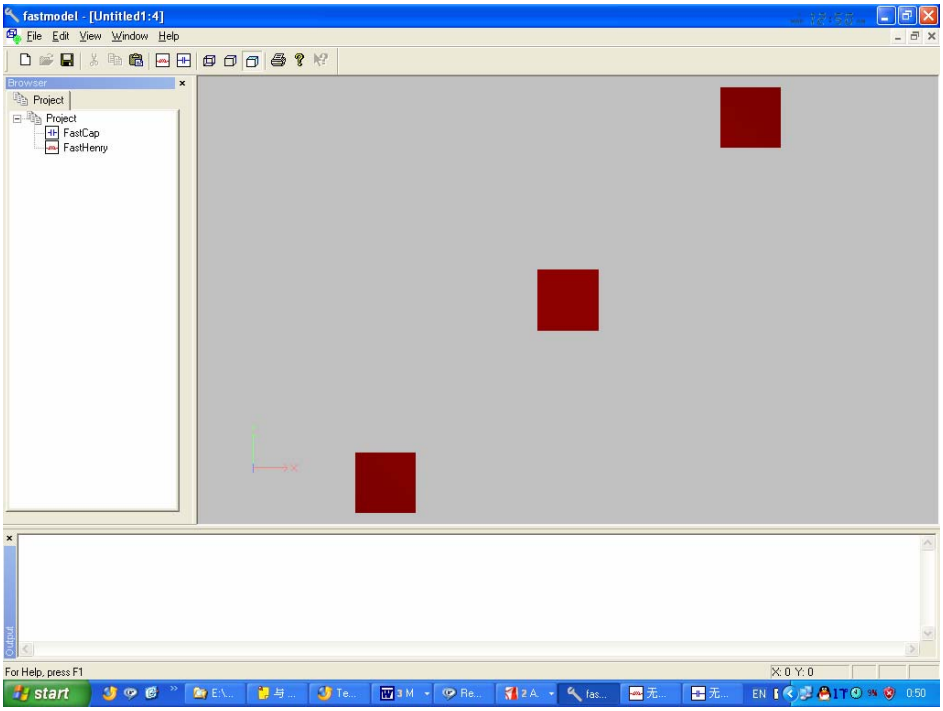
NEW ALGORITHM

Our algorithm gets the idea from FastCap, Nebula and HiCap, that is, we use a kernel independent multipole method with geometric moments and hierarchical refinement to speed up the matrix-vector computation time.

Analyzing the FastCap, we know in order to accurately capture near field effects between adjacent conductors, a very fine subdivision of conductors is used for all conductors, which increases the computation time. Of course, the complexity is still *O(n),* but n is much large that the total increase greatly. From the experiments of Nebula, we know the charge density only has a strong variation on selected and nearby conductors, while a smooth variation on far away conductors. Besides, the charge density value and variation reduce quickly with the increase of distance from the selected conductor. An example is shown in Figure 6. There are three 2-D conductors (plates) in the space, which are shown in (a); we need to compute the capacitance matrix, which is a 3x3 matrix. Figure (b) and (c) show the charge density distributions on conductor1 and conductor2 when we compute the column corresponding to conductor1. We can see there is a strong charge density variation on conductor1 especially on edges, while the charge variation on conductor2 is smooth and small. The average charge density on conductor1 is more than 10 times of that one on conductor2.

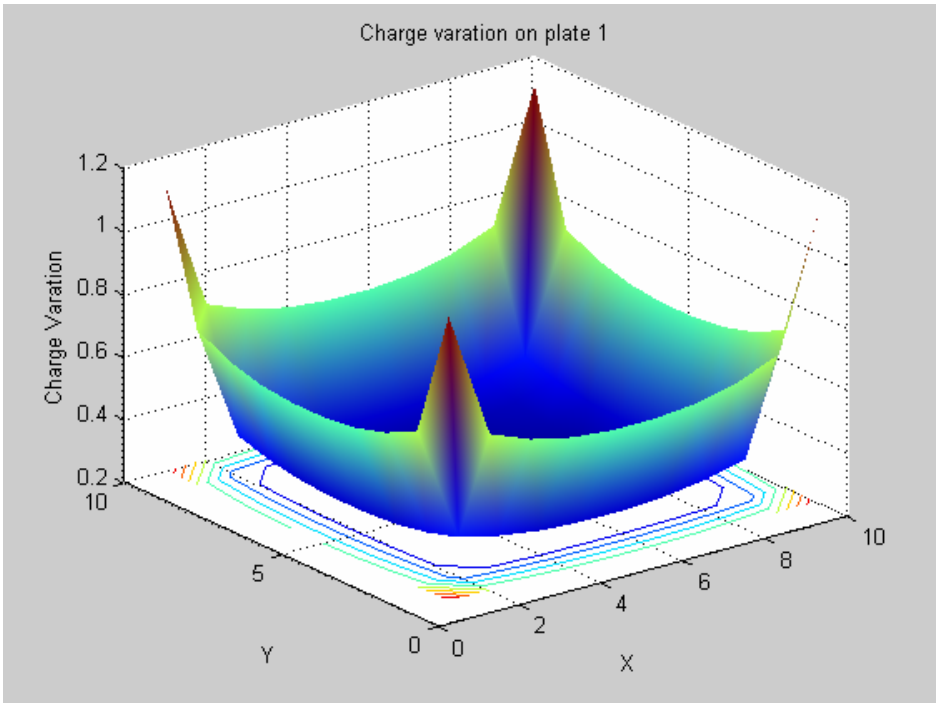
Because the charge density on far away conductors is small and smooth, if we use the same fine subdivision for these conductors, the error will be so far below than those on selected conductors. Here comes our idea that we can use a coarser subdivision on these far conductors to reduce the size of system in order to trade unnecessary accuracy

for faster computation.

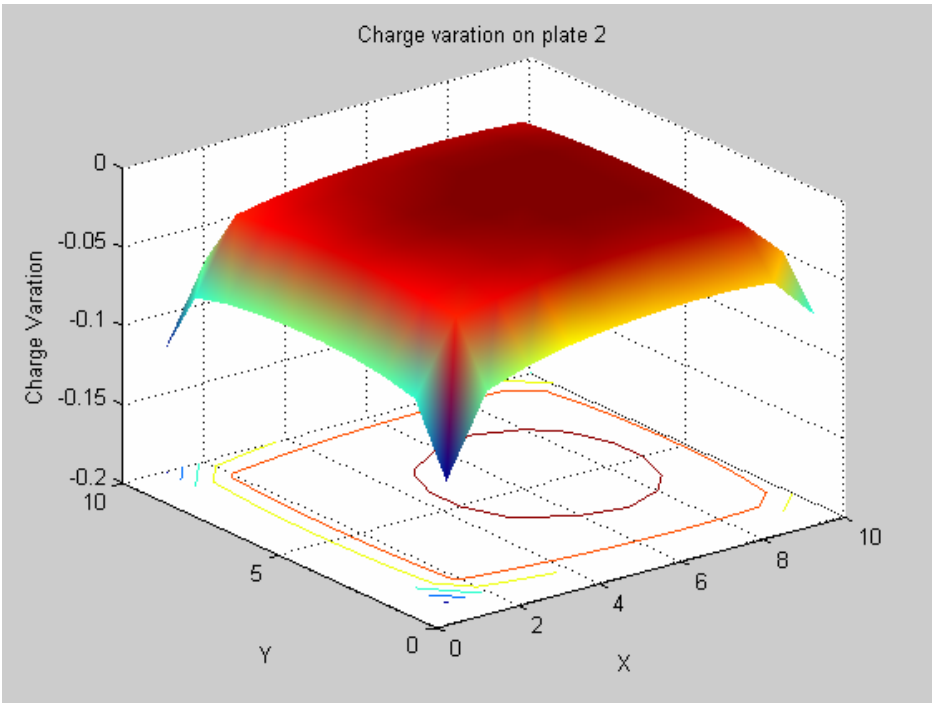


(a)

Fig. 6 The charge density variation is strong on selected conductor and weak on far conductors. (a) 3 square conductors are in the space, we compute the capacitance column corresponding to conductor1, which is located in the left bottom. (b) The charge variation on conductor 1. (c) The charge variation on conductor 2(the middle one).



(b)



(c)

Fig. 6 continued.

In Nebula, for different selected conductors, it repartitions the space containing the full geometry. In the boxes containing the selected conductor and in adjacent boxes, it uses a traditional Galerkin partition. With the distance increasing from the selected conductor, the geometry is partitioned into larger and larger boxes. These boxes are called Nebula boxes, then for each box it calculates the moments of the geometry and form the matrix P_R, which can be used to compute the product of geometry and charge variation through $P_R * f$, where P_R is the linear transformation matrix and the f is the charge variation polynomial. A modified GMRES [10] is used to find the 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. Nebula has several shortcomings: (1) For each selected conductor, it needs to repartition the space and re-compute the geometric moments for each box and this computation is a integration based on the Legendre polynomials which needs much time; (2) For those tiles partitioned from the selected and nearby conductors, potentials are computed directly. This needs about one-half to one-third of the total time in Nebula. (3) After Nebula gets the final moments of the charge density for each box, it still needs to integrate the charge density to get the charge. For higher order moments, this still takes time.

Analyzing FastCap and Nebula, we can develop a new algorithm based on FMM which can overcome the shortcomings of Nebula. That is, with the new algorithm, we don't need to rediscretize the space and re-compute the geometric moments for each box for different selected conductors and can speed up the computation for tiles discretized by the traditional method. For the purpose of speeding up the computation for nearby tiles, the FMM can solve this and the accuracy can be controlled with some scheme, such as

changing the distance to interact for high level boxes. For the purpose of avoiding recomputing the geometric moments for each Nebula box, we need to find a new scheme. Analyzing the FMM structure, we find we can use the FMM tree to find the final structure for different selected conductors. We can go from the bottom level to the top, and compute the new geometry information for each box from low level boxes. After that, we can find the geometry information for all boxes in the tree. And then, for a certain selected conductor, we use some scheme to decide which level the far away parts should be used to compute the potential to get a truncated tree which has a similar structure to the one of Nebula. The benefit of this scheme is that for the whole geometry, we only need to compute geometric information for the FMM tree one time which can save much time. Besides, in FMM for all those tiles, we get the charge of each tile directly in the end. So to find the charges of all conductors, we do not need to integrate by moments but adding the contained tiles' charges. In a word, use the FMM structure; we can solve the typical three problems of Nebula. The left problems are how to define geometric information and how to truncate the FMM tree to approximate the far away parts.

2.1 Shape Approximation

Nebula uses moments based on Legendre polynomials to express the geometric information; we use shape to approximate the geometry. One benefit of shape approximation is that it is much easy to implement in FMM tree, and the other benefit is that we have closed forms to commpute the moments based on the shapes unlike Nebula needs to do integration . The idea of shape approximation is that for an original geometry, we replace it with a new shape which has the same specified p-th order geometric moments. In FastCap, the interaction of two well seperated boxed are done by multipole expansion.

While in our algorithm, the interaction of two well-seperated boxes are computed according to the approximate shapes.

Now, we need to find the way to approximate the geometric moments. The two-dimensional geometric moment of order $p+q$ of a function f (x, y) is defined as

$$M_{pq} = \int_{a1}^{a2} \int_{b1}^{b2} x^p y^q f(x,y) dxdy \qquad (4)$$

, where p ,q = 0,1,2,…,∞. Note that the monomial product $x^p y^q$ is the basis function for this moment definition. A set of n moments consists of all M_{pq} for $p + q \leqslant n$, i.e., the set contains (n+1)*(n+2)/2 elements. For example, the 2nd order moments are M_{00}, M_{01}, M_{10}, M_{11}, M_{20}, M_{02}. For a geometry, if we define f(x,y) = 0 on the surface and 0 elsewhere, according to equation (4), it's easy for us to find the physical meaning of low order geometric moments. $M_{00} = \int_{a1}^{a2} \int_{b1}^{b2} dxdy = (a2-a1)*(b2-b1)$ is the area of the surface, $M_{10} = \int_{a1}^{a2} \int_{b1}^{b2} xdxdy, M_{01} = \int_{a1}^{a2} \int_{b1}^{b2} ydxdy$ are the gravity center, and $M_{11} = \int_{a1}^{a2} \int_{b1}^{b2} xydxdy, M_{02} = \int_{a1}^{a2} \int_{b1}^{b2} y^2 dxdy, M_{20} = \int_{a1}^{a2} \int_{b1}^{b2} x^2 dxdy$ are the area distribution directions. Based on this observation, we design the shceme to approxiamate the geometry with a shape matching the low order geometric moments. Our experiments show that approximating the low order moments up to 2nd is accurate enough, but in general higher order moments are supported by the same scheme.

For each box in the FMM tree, it contains many partitioned tiles. For the purpose of approximating the low 2nd order moments we use a rectangle. A centered rectangle can not be used to approximate the 2nd moment M_{xy} because the M_{xy} of a symmetric rectangle is always 0, but the experimental result show that approximating the geometry with the other

five moments are accurate enough. To approximate the zero order geometric moments, which is in fact area, we let the area of the rectangle equal to the total area of all contained tiles; to approximate the 1st order, we locate the rectangle's center at the gravity center; finally, to approximate the M_{xx} and M_{yy}, we need to determine the horizontal and vertical edge lengths. We know for a rectangle show in figure 7, according to equation (4),

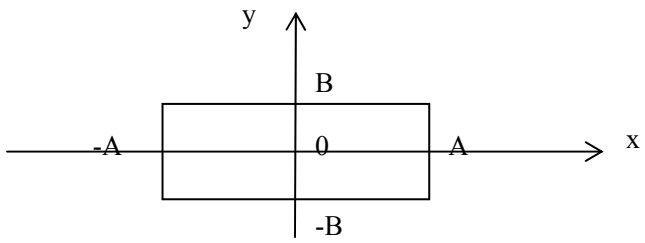


Fig. 7 A rectangle located at the origin.

$M_{00} = 4AB$; $M_{x0} = M_{0y} = 0$; $M_{xx} = 4*A^3*B/3$; $M_{yy} = 4*B^3*A/3$, and we can get

$$A = \sqrt[8]{\frac{9}{16} \cdot \frac{M_{xx}^3}{M_{yy}}} \qquad (5)$$

$$B = \sqrt[8]{\frac{9}{16} \cdot \frac{M_{yy}^3}{M_{xx}}} \qquad (6)$$

Based on (5) and (6), we can determine the horizontal and vertical edge lengths of the approximated rectangle to match M_{xx} and M_{yy}. But it's not hard to see, there are only two parameters of the rectangle: A and B, so a rectangle cannot be used to match M_{00}, M_{xx}, M_{yy}. What we do is to match M_{xx}, M_{yy}. The reason why we use a rectangle to approximate the geometry is that we have simple closed forms to compute the potential coefficients for these rectangles which are easy to implement and time saving. Figure 8 briefly shows the idea of computing the interaction between two well-separated boxes by multipole

expansion of FMM and shape approximation of our algorithm. The benefit of this scheme is that it is much easy to implement by a hierarchical scheme and it does not need to compute any integration, which saves much time.

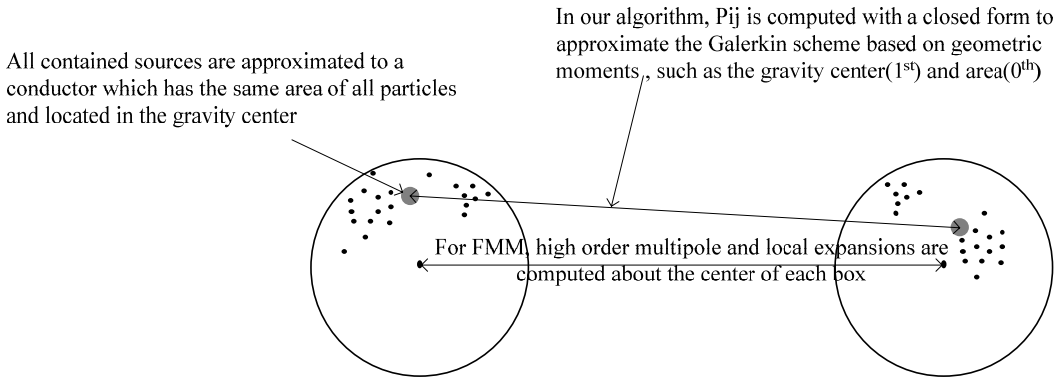


Fig. 8 Use low order geometric moments to compute potential.

Now, we explain in detail how we find these 2nd order geometric moments of each box in the FMM tree. For zero order geometric moments, we need to find the total area. Because we partition the geometry into uniform quad lateral squares, to find the total area we only need to know the number of contained tiles.

For the bottom box, we just find the number of contained tiles. For boxes at higher levels this number can be found by adding up those of its children. For the gravity center of a bottom box, we find the averages of x and y coordinates of all contained tiles, which are in fact the gravity center's coordinates. For the boxes at higher levels the gravity center's position can be found based on the gravity centers and numbers of tiles of its children. For example, for a high-level box m, it has four children boxes 1, 2 3, and 4. And the gravity centers' coordinates and numbers of tiles of these children boxes are n1, (x1, y1), n2, (x2, y2), n3, (x3, y3) and n4, (x4, y4) respectively. Then the gravity center of box m can be

found by

$$x_m = \frac{n1 \times x1 + n2 \times x2 + n3 \times x4 + n4 \times x4}{n1 + n2 + n3 + n4} \quad (7)$$

$$y_m = \frac{n1 \times y1 + n2 \times y2 + n3 \times y4 + n4 \times y4}{n1 + n2 + n3 + n4} \quad (8)$$

To approximate the second geometric moments we need to find the Mxx and Myy of each box. For a bottom box, it contains tiles, which are uniform lateral squares. According to equation (5) and (6), we know for a square (A =B) located at the origin, Mxx = Myy = $4*A^4/3$. For a square which is not located at the origin shown in figure 9,

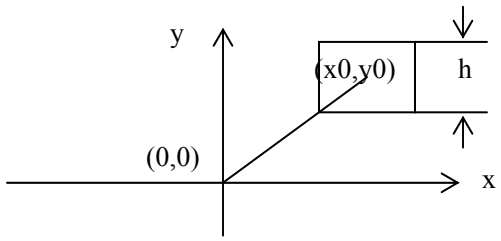


Fig. 9 An equal lateral square located at (x0, y0).

We can compute the Mxx and Myy by

$$M_{xx} = \int_{x_0-h/2}^{x_0+h/2} \int_{y_0-h/2}^{y_0+h/2} x^2 dxdy = \frac{h}{3} \cdot \left[(x_0 + h/2)^3 - (x_0 - h/2)^3 \right] \quad (9)$$

$$M_{yy} = \int_{x_0-h/2}^{x_0+h/2} \int_{y_0-h/2}^{y_0+h/2} y^2 dxdy = \frac{h}{3} \cdot \left[(y_0 + h/2)^3 - (y_0 - h/2)^3 \right] . \quad (10)$$

For each tile contained in the bottom box, we compute the Mxx and Myy. Add up these values for all these tiles we get the Mxx and Myy of the bottom box.

For boxes at all coarser levels, we need to shift the Mxx and Myy of its children boxes to its center. See the rectangle in figure 10, which is not located at the origin, we can

compute the Mxx and Myy by

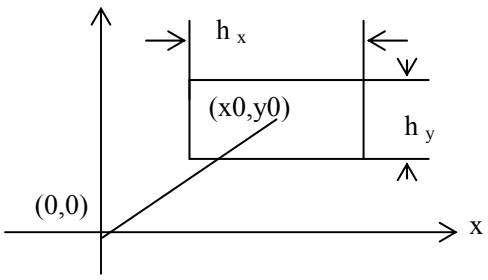


Fig. 10 A rectangle located at (x0, y0).

$$M_{xx} = \int_{x_0-h_x/2}^{x_0+h_x/2} \int_{y_0-h_y/2}^{y_0+h_y/2} x^2 dxdy = \frac{h_y}{3} \cdot \left[(x_0 + h_x/2)^3 - (x_0 - h_x/2)^3 \right] \quad (11)$$

$$M_{yy} = \int_{x_0-h_x/2}^{x_0+h_x/2} \int_{y_0-h_y/2}^{y_0+h_y/2} y^2 dxdy = \frac{h_x}{3} \cdot \left[(y_0 + hy/2)^3 - (y_0 - h_y/2)^3 \right] \quad (12)$$

In 2-D case, a coarser level box has four such children boxes. For each of these children boxes, computing the Mxx and Myy and adding them up we get the Mxx and Myy of the box, with which we can determine the horizontal and vertical edge lengths of it according to equations (5) and (6).

First, we use a rectangle located at the gravity center to match the 2nd moments set including $M_{x0,}$ $M_{y0,}$ M_{xx}, $M_{yy.}$ Experimental results show that matching above four moments already gives accurate enough results as long as the designer sets a reasonable partition depth of the hierarchical tree and precision limit.

To match M_{00}, we use a different scheme. We put four squares with the area of $M_{00}/4$ along x,y axis symmetrically. See figure 11, all these four squares are put on the axis with the distance from origin A and B respectively. Mxx, Myy determine the values of A and B.

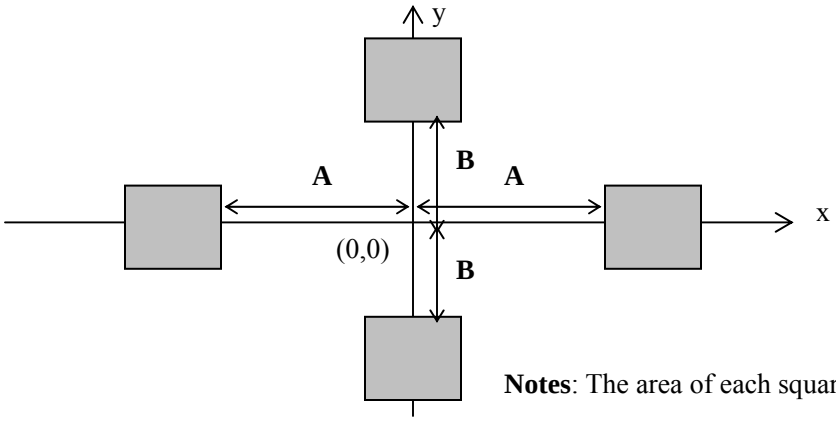


Fig. 11 Use four squares to match five moments.

$$M_{00} = area_1 + area_2 + area_3 + area_4 = M_{00} / 4 \quad (13)$$

$$M_{xx} = \frac{M_{00}}{2} \cdot A^2 + \frac{M_{00}^{\frac{3}{2}}}{4} \cdot A + \frac{5}{96} M_{00}^2 \quad (14)$$

$$M_{xx} = \frac{M_{00}}{2} \cdot B^2 + \frac{M_{00}^{\frac{3}{2}}}{4} \cdot B + \frac{5}{96} M_{00}^2 \quad (15)$$

If we put the gravity center of these four squares at the gravity center of a box in the hierarchical tree, we can use these four squares to match $M_{0x,}$ M_{0y}. The M_{00} is already matched because the total areas of these four squares are $M_{00.}$ And according to equations (14) and (15) we can set reasonable A and B to match M_{xx}, M_{yy} by equations (16) and (17). Experimental results show that the accuracy improvement of this scheme compared with the rectangle scheme is very small, while the time increases much.

$$A = \frac{-\frac{M_{00}^{\frac{3}{2}}}{4} + \sqrt{(\frac{M_{00}^{\frac{3}{2}}}{4})^2 - 2 \cdot M_{00}(\frac{5}{96} M_{00}^2 - M_{xx})}}{M_{00}} \quad (16)$$

$$B = \frac{-\frac{M_{00}^{\frac{3}{2}}}{4} + \sqrt{(\frac{M_{00}^{\frac{3}{2}}}{4})^2 - 2 \cdot M_{00}(\frac{5}{96} M_{00}^2 - M_{yy})}}{M_{00}} \quad (17)$$

To match geometric moments higher that 2nd, a more complicated shape is needed. For example, we can use a cross, which is located at the gravity center. Figure 12 shows how to approximate the geometry with 0th and 1st geometric moments and the idea to approximate higher order ones.

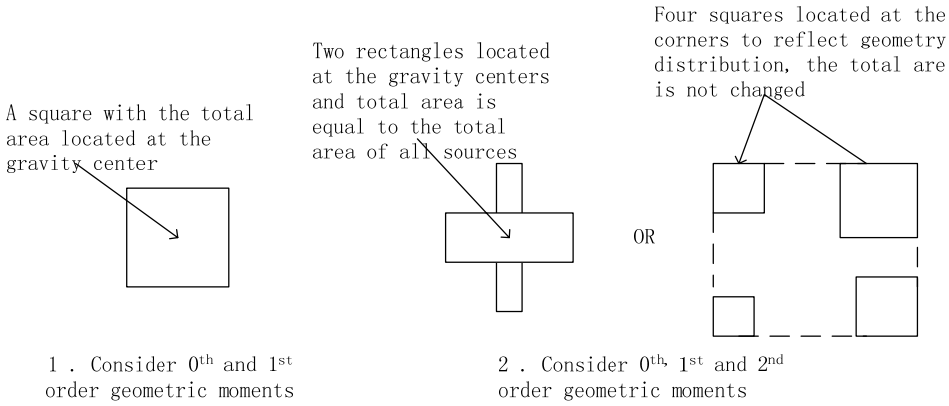


Fig. 12 Shape approximation of 1st and higher order.

2.2 Grouping Far Away Parts

To group the far away parts, what we do is to traverse the FMM tree from top to bottom, and check whether the boxes separated from the ones containing selected conductors are far enough, if so, we record it as a leaf node of the truncated FMM tree. And then, when we compute the potentials of downward tiles, we do not go down but just take them as a whole conductor (approximated rectangle described in the previous section). After we get the potential of this conductor, we distribute this value to all contained tiles, that is, all contained tiles have the same potential of the approximated conductor. By doing

this, the problem size will be reduced greatly and experimental results show that this approximation will not affect the converge speed of the iterative method. For example, there are totally three squares and they are partitioned into 300 square tiles in testcase3.lst, which is shown in Figure 6 (a). For FMM, no matter which conductor is selected, the problem size is 300×300; the compression comes from mutipole approximation. While in our algorithm, the problem size is reduced greatly, and this contributes much saving on time and memory. For example, when conductor 1 is selected to compute the capacitance column, conductor 2 will be taken as four rectangles each of which represents 25 bottom tiles, and conductor 3 is taken as one rectangle, which represents contained 100 bottom tiles. Figure 13 shows the final approximation for conductor 1 selection, after the approximation, the problem size is reduced from 300×300 to 105×105.

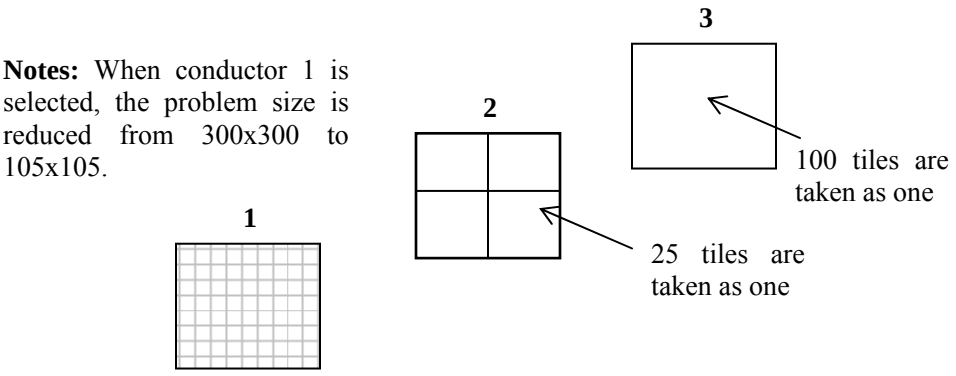


Fig. 13 Final partition panels of above problem.

Figure 14 briefly shows the idea of grouping far away parts. We can see from this graph that the further the far parts away from the selected conductor, the higher level the boxes will be. In other words, the further the larger approximation will be used. Notice for the real FMM algorithm, the tree is not a binary tree but a quad tree in 2-D case or an octal tree in 3-D case. Finally, we use these leaf nodes and bottom tiles on selected and nearby

conductors to compute the potentials. Of course, there exist potential variations on these tiles. But we know the charge density variation and value are very small on these tiles such as those shown in figure 7, so the potential variations are also small. Thus, assuming the potential is uniform on these tiles is reasonable, and does not introduce much error. As mentioned before experimental results show that this approximation has very little effect on the final result, and does not affect the converge speed of the iterative solving.

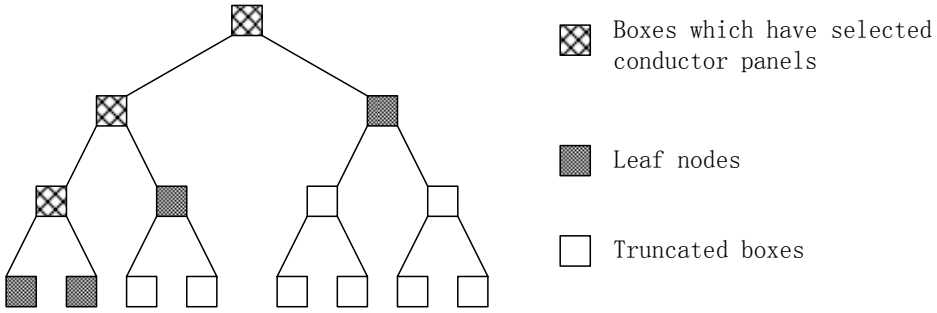


Fig. 14 Use an error estimate function to truncated FMM tree.

How can we say two boxes are far apart enough? We use a simple error estimate similar to the one in HiCap to judge. We will explain this scheme in detail in next section.

2.3 Separation

HiCap uses [3]

$$P_{ij} = \frac{1}{area(A_i)} \int_{x_i \in A_i} \frac{1}{area(A_j)} \int_{x_j \in A_j} \frac{1}{\|x_i - x_j\|} da_j da_i \qquad (18)$$

to estimate the potential coefficient to judge whether it's necessary to subdivide a panel further. The users give a precision limit, if the estimated potential coefficient is below the limit, the accuracy is good enough, do not need to divide. We use a similar but simpler scheme to judge whether two boxes are far enough and can interact at current level. Given

the precision limit *peps*, for two boxes, assume one's area is A1, the other's area is A2, and the distance between two boxes is d, if A1*A2 / d < peps, we think the accuracy is enough. Else, they cannot interact but need to subdivide further to satisfy the precision requirement.

When compute the far field potentials of nearby tiles, we use this scheme to determine the interaction list of each box. For example, see the case shown in figure 6, for the box X. we check the children of those boxes who are the nearby boxes of X's parent. For a box Y belongs to these box, if Ax*Ay / d < peps, it can interact with X at current level. One thing needs to be mentioned is that this is similar to the one of FastCap, but different. In FastCap, only neighboring boxes are taken as nearby boxes; while in our case, whether it is a nearby box is determined by the given precision limit. A box taken as well separated in FastCap could be taken as nearby in our algorithm.

As mentioned in the previous section, when truncate the FMM tree; we use this scheme to determine which level the far away tiles should be approximated. For example, in Figure 15, the tiles contained in Y will be approximated by one conductor. When computing the potentials of these contained tiles, they are taken as one, and the final potential will be distributed to all these potentials.

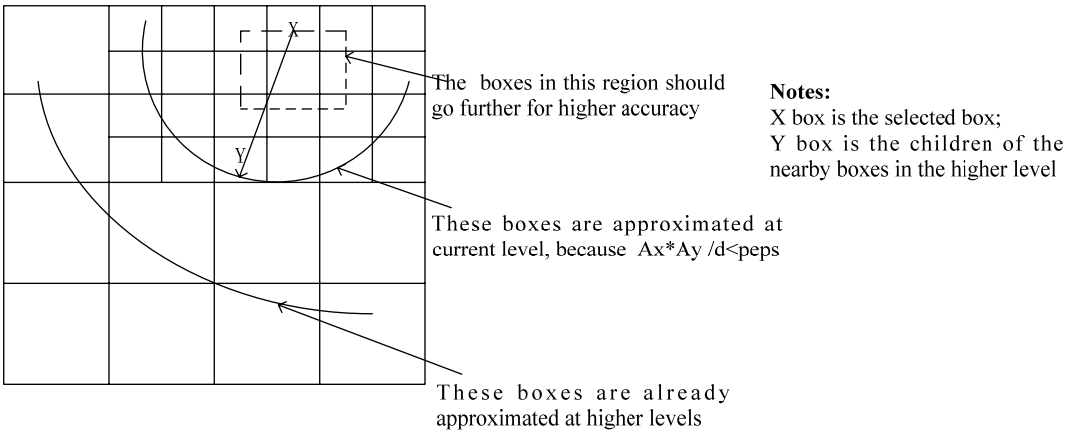


Fig. 15 Use the estimate of potential coefficient to determine interaction list.

2.4 The New Algorithm

2.4.1 Preprocessing

This step is used to prepare the input data. All the conductors of the problem are discretized into small tiles. In out current research, we discretize the conductors uniformly into equal lateral squares. Generally, triangle and adaptive discretization can be used for accuracy and efficiency. After that, we get a list file in the standard form of FastCAP list file, which depicts the coordinates of these small panels.

2.4.2 The Main Flow

The following Figure 16 shows the 2-D algorithm. With some work, it can be easily expanded to 3-D.

Initialization

Choose a number of levels so that there are, on average, s particles per box at the finest level. The bounding box of the system is repeatedly spatially subdivided using a quad-tree structure in 2-D or octal-tree in 3-D until reach the finest level. Each particle is then assigned to a box on the lowest level of the tree. Finally, the depth of the quad-tree is $h = \left\lceil \log_4 \frac{N}{s} \right\rceil$, and there are 4^h boxes in the finest level, where N is the total number of tiles.

Upward Pass to Find the 2nd Order Geometric Moments

Comment [We proceed from the finest level, computing the geometric moments for every box based on the number of particles it has.]

Step1

Comment [At the finest level, the moments of each box are found according to the particles within it.]

Fig. 16 New algorithm

$do\ \ ibox = 1, 2, ..., 4^h$

$area = \sum_{i=1}^{n} area_i$;

$gravity\_x = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i$; $gravity\_y = \frac{1}{n} \cdot \sum_{i=1}^{n} y_i$;

$M_{xx} = \sum_{i=1}^{n} M_{xxi}$; $M_{yy} = \sum_{i=1}^{n} M_{yyi}$

enddo

Step2

Comment [For boxes at all coarser mesh levels, the gravity center and area are computed by merging those values of their children boxes.]

$do\ l = h-1, ..., 0$

$do\ ibox = 1, 2, ..., 4^l$

$area = area_{1st} + area_{2nd} + area_{3rd} + area_{4th}$

$$gravity\_x = \frac{gravity\_x_{1st} \times n_{1st} + gravity\_x_{2nd} \times n_{2nd} + gravity\_x_{3rd} \times n_{3rd} + gravity\_x_{4th} \times n_{4th}}{n_{1st} + n_{2nd} + n_{3rd} + n_{4th}}$$

$$gravity\_x = \frac{gravity\_x_{1st} \times n_{1st} + gravity\_x_{2nd} \times n_{2nd} + gravity\_x_{3rd} \times n_{3rd} + gravity\_x_{4th} \times n_{4th}}{n_{1st} + n_{2nd} + n_{3rd} + n_{4th}}$$

$M_{xx} = \sum_{i=1}^{4} Shift\_of\_M_{xxi}$; $M_{yy} = \sum_{i=1}^{4} Shift\_of\_M_{yyi}$;

enddo

enddo

Compute Potentials of Approximate Boxes Directly

Step3

Comment [For the potentials of approximated boxes, we compute directly. For an approximated box, we add up the potentials which come from other boxes and nearby tiles.]

$do\ ibox = 1, ..., M$

$do\ \ jbox = 1, ..., M$

$$P_{ij} = \frac{1}{area(ibox)} \times \int_{x_i \in ibox} \frac{1}{area(jbox)} \int_{x_j \in jbox} \frac{1}{4\pi\varepsilon_0 \| x_i - x_j \|} da_j da_i$$

$ibox.potential = ibox.potential + P_{ij} \times jbox.charge$

$jbox.potential = jbox.potential + P_{ij} \times ibox.charge$

enddo

Fig. 16 continued.

do ktile = 1,…,K

$$P_{ik} = \frac{1}{area(ibox)} \times \int_{x_i \in ibox} \frac{1}{area(ktile)} \int_{x_k \in ktile} \frac{1}{4\pi\varepsilon_0 \| x_i - x_k \|} da_k da_i$$

$$ibox.potential = ibox.potential + P_{ik} \times ktile.charge$$

$$ktile.potential = ktile.potential + P_{ik} \times ibox.charge$$

enddo

enddo

Notes:

P*ij* is computed based on the approximated rectangle of each box, which is determined with the 2nd order moments.

Downward Pass to Find Far Field Potential for Nearby Tiles

Comment [In this phase, we use the FMM scheme to find the far filed potentials of nearby tiles. We proceed from the coarsest level, computing the far filed potential for each box with given peps, which guarantees that interactions are consistently computed at the coarsest possible level. When computing the potential from a well-separated box, we use the approximated shapes and an analytic form to find the potential coefficient.]

Step4

do $l = 1,...,h-1$

do $ibox = 1,2,...,4^l$

$far\_potential = 0$

for $jbox \in interactionlist$

$$P_{ij} = \frac{1}{area(ibox)} \times \int_{x_i \in ibox} \frac{1}{area(jbox)} \int_{x_j \in jbox} \frac{1}{4\pi\varepsilon_0 \| x_i - x_j \|} da_j da_i$$

$$far\_potential = far\_potential + P_{ij} \times jbox.charge$$

endfor

enddo

enddo

Notes:

$$interactionlist = \{box_k : distance(ibox.parent - box_k.parent) \le \left\lceil \frac{1}{peps} \right\rceil \& distance(ibox - box_k) > \left\lceil \frac{1}{peps} \right\rceil \}$$

for small size problems, the potentials can be solved directly by direct methods.

Fig. 16 continued.

Final Potential Evaluation

Comment [In this phase, we combine the far and local field potentials to get the final potential for each nearby tile. For far tiles approximated by boxes, we distribute the potential of the box to its contained tiles.]

Step5

Comment [A downward pass is executed to distribute the far filed potential to the finest level. Because potential is not expansion, no shift operation is needed.]

$do\ l = 1,...,h-1$
  $do\ ibox = 1,2,...,4^l$
    $ibox.1st->potential = ibox.1st->potential + ibox.potentil$
    $ibox.2nd->potential = ibox.2nd->potential + ibox.potentil$
    $ibox.3rd->potential = ibox.3rd->potential + ibox.potentil$
    $ibox.4th->potential = ibox.4th->potential + ibox.potentil$
  enddo
enddo

Step6

Comment [For each box, distribute the potential of the box to the contained tiles.]

$do\ ibox = 1,...,M$
  DistributePotentialToTheBottomTiles(ibox);
enddo

Step7

Comment [Compute potential due to nearest neighbors directly.]

$do\ ibox = 1,2,...,4^h$
  For every nearby tile p_j in box $ibox$, directly compute interactions with all other particles within the box and its nearest neighbors.
enddo

Step8

Comment [For each nearby tile adds direct and far-filed terms together.]

Fig. 16 continued.

do $ibox = 1, 2, ..., 4^h$

For every particle p_j in box $ibox$, add direct and far-field terms together.

enddo

Fig. 16 continued.

If we use iterative methods to solve the linear system $Pq=v$ where P is the potential coefficient matrix, q is the unknown charge vector, and v is the desired potential vector, we need to evaluate the matrix-vector product Pq in each iteration. It's clear that the fast matrix-vector multiplication only needs the later 6 steps of above algorithm. We can call it "Fast Matrix-Vector Multiplication".

2.4.3 Complexity Analysis

A brief analysis of the algorithmic complexity is shown in Figure 17. We assume there are N partitioned tiles totally; for a selected conductor, we assume there are N1 nearby tiles, and M approximated boxes in the FMM tree for far away tiles.

Step#	Operation count	Explanation
Step1	order *N*	each tile is accessed three times to compute the gravity center and area
Step2	order *N*	at the l^{th} level, we need 25 operations for each box
Step3	order *(N1+M)*M*	each box interacts with other boxes and nearby tiles
Step4	order $\le 3\times(1+2\times\left\lceil \frac{1}{peps} \right\rceil)^2 \times N1$	There are at most $3\times(1+2\times\left\lceil \frac{1}{peps} \right\rceil)^2$ entries in the interaction list for each box at each level. For example, if $peps = 1$, there are at most 27 entries. If we use fixed # of sub boxes to compute the P_{ij}, e.g.

Fig. 17 Complexity analysis

		4 for each box, the # of operations for each box is also fixed (constant).
Step5	order *N1*	each FMM internal box needs 4 operations
Step6	order *N-N1*	the number of far away tiles is *N-N1*, and the operations of downward pass to distribute the potential of a sub tree is linear of the bottom tiles
Step7	order $\le \frac{(1+2\times\left\lceil \frac{1}{peps} \right\rceil)^2}{2} N1\cdot k_n$	Let k_n be a bound on the number of nearby tiles per box at the finest mesh level. Interactions must be computed within the box and its nearest neighbors and we only need to compute half of the pair wise.
Step8	order *N1*	Adding two terms for each particle.

Fig. 17 continued.

If we omit the hidden number of operations in each step, the estimate for the running time is therefore

$$5\cdot a\cdot N + b\cdot M\cdot(N1+M) + c\cdot N1((3+k_n)\cdot(1+2\times\left\lceil \frac{1}{peps} \right\rceil)^2+1) \qquad (19)$$

with the constants a, b and c determined by the computer system, language, implementation, etc. Note that implicit in the complexity estimate is a condition that the number of tiles per box at the finest mesh level is bounded.

It's easy to see from equation (19) that the complexity is a function of *pesp* , and when $peps\downarrow$, the $complexity\uparrow$ and $accuracy\uparrow$. In other words, we can use *pesp* to trade off the accuracy and complexity. In some meaning, *pesp* is equal to the percentage of far-filed potential computed by approximation. Figure 18 shows the peps ~ approximation percentage curve of a test case which has five conductors and partitioned into five hundred uniform quad-lateral squares without considering grouping far away tiles. Generally

speaking, if all partitioned tiles are distributed uniformly,

$$\text{Approximation percentage} \approx 1-(1+2\times\left\lceil \frac{1}{peps} \right\rceil)^2 / \frac{N}{s} \qquad (20)$$

where N is the total number of particles and s is the number of particles in each bottom box. This function is drawn in figure 19 with $N/s = 256$, that is, the quad-tree height is 5.

If we let p = *approximation percentage,* according to (5) and (6), we can rewrite the complexity as the function of p

$$(1-p)\cdot c\cdot(3+k_n)\cdot N1\cdot N/s+5\cdot a\cdot N+b\cdot M\cdot(N1+M)+c\cdot N1 \qquad (21)$$

It's not hard to see from equation (21) that the complexity is sensitive to the multipole expansion percentage p, especially in large systems, in which case N and N1 are much large. In such systems, when the percentage gets low the complexity grows drastically. This is why we suffer in large systems to get higher accuracy using low multipole expansion percentage. In our algorithm, we use new a scheme to group far away tiles as much as possible, which increase the approximation percentage as much as possible. So it can reduce the total time greatly.

In our implementation, we can also specify the depth of the FMM tree. Analyzing the algorithm, we know changing the depth along will not change the approximation of far away parts, it only affects the complexity to compute nearby tiles' potentials. The reason is we go from the top of the FMM tree to decide which level we should approximate the far away parts. So the leaf nodes of the truncated FMM tree will not change with the depth. But the depth in deed will affect the accuracy of geometric moments and the complexity to compute nearby tiles' potentials. So people need to combine the depth and peps parameters at the same time to trade the accuracy and speed most economically.

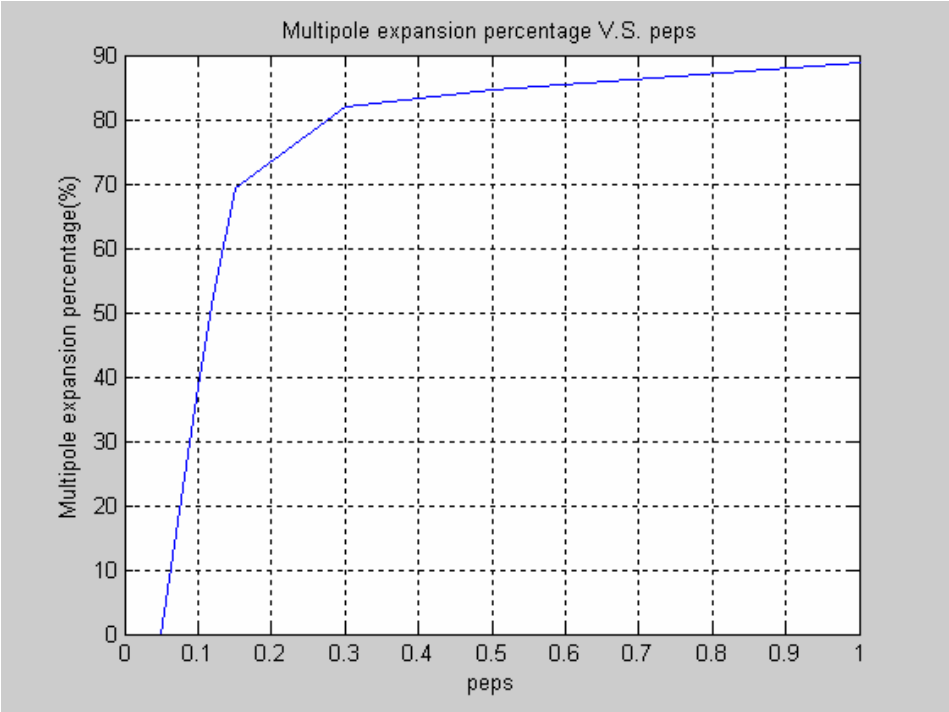


Fig. 18 Peps versus approximation percentage for one test case with five conductors.

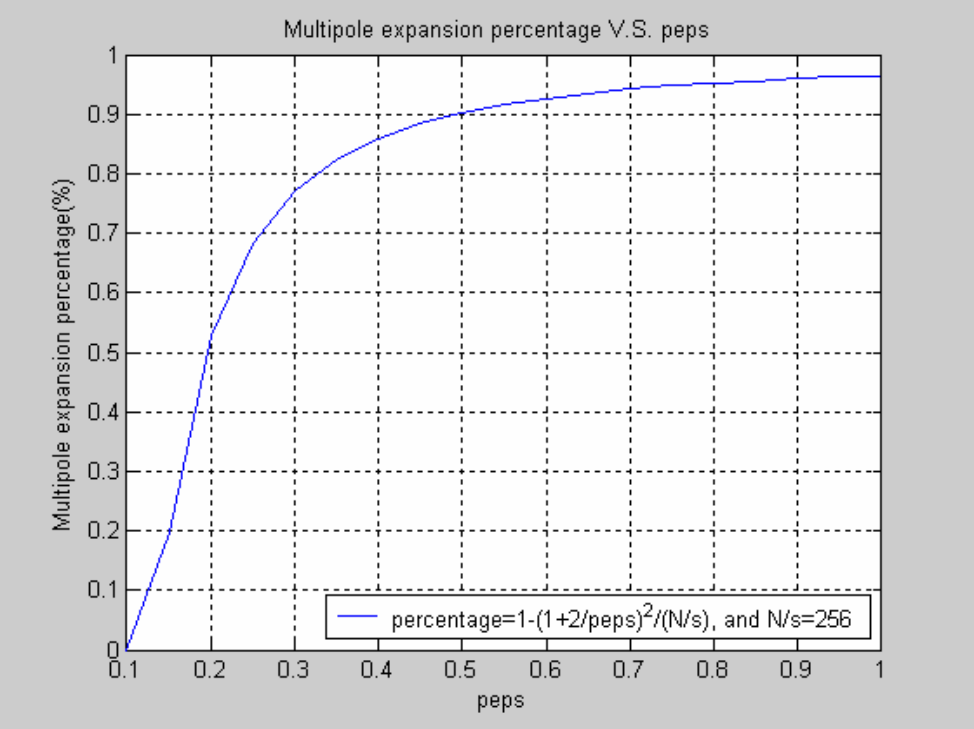


Fig. 19 Peps versus approximation percentage for uniform distribution.

CHAPTER III

IMPLEMENTATION AND RESULTS

The new algorithm is implemented in C/C++ and simulation results are reported. The new algorithm, the FastCap, and HiCap are complied and executed on a PC with CPU1.3G, 256M DRAM and Windows XP OS. FastCap(0) is the fastest in the FastCap package, and is about twice as fast as FastCap(2). However FastCap(0) has 5% to 10% relative error with respect to FastCap(2).

The difference or error of capacitance matrices is defined as follows. Let the capacitance matrix computed by FastCap(2) be C and the capacitance matrix computed by another program be C'. Then the difference is estimated according to the norm [11]: $||C - C'|| / ||C||$. This is the standard way to measure the difference between two matrices.

First, we compare our algorithm with FastCap. To avoid the effect of different implementation, we also implement the zero order FastCap with the same data structure and other implementation methods. Testcase1 has three squares and have been partitioned into 300 tiles, which are shown in figure 6(a). Testcase2 has 5 complicated conductors, which is shown in figure 20. Table I shows the experimental results. We can see the new algorithm will reduce the problem size and number of interactions greatly while at the same time remains an acceptable accuracy. In fact, the data in the table is with parameters: depth = 4 and peps = 0.5, we still can use peps and the tree depth parameters to trade the accuracy and problem size. Other test cases have similar geometries.

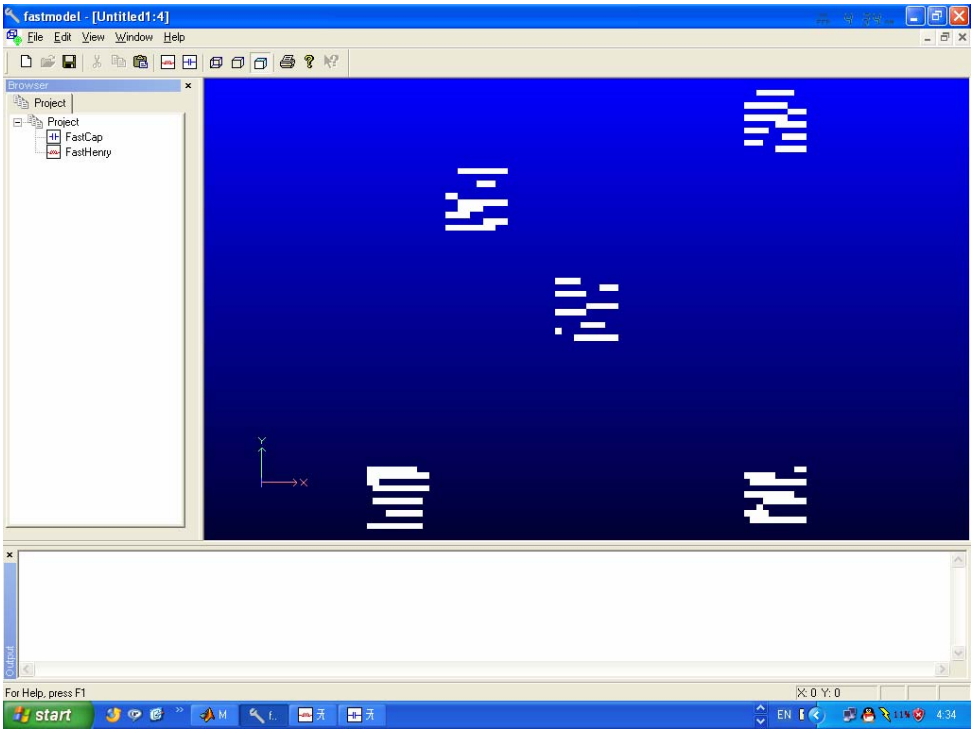


Fig. 20 Five complicated conductors separated into 216 tiles.

TABLE I
COMPARISON WITH FASTCAP WITH A RECTANGLE

Test Case	Error		Problem Size Compression		# of interactions	
	New Alg.(2)	FastCAP(0)	New Alg.	FastCAP	New Alg.	FastCAP
1	0.74%	2.77%	(0) 108x108 (1) 108x108 (2) 108x108	300x300	(0) 108x108 (1) 108x108 (2) 108x108	29604
2	2%	3%	(0) 62x62 (1) 66x66 (2) 54x54 (3) 55x55 (4) 64x64	216x216	(0) 62x62 (1) 66x66 (2) 54x54 (3) 55x55 (4) 64x64	11288
3	1.31%	2%	(0) 53x53 (1) 48x48 (2) 33x33 (3) 39x39 (4) 45x45	145x145	(0) 53x53 (1) 48x48 (2) 33x33 (3) 39x39 (4) 45x45	4429
4	0.8%	4.63%	(0) 57x57 (1) 76x76 (2) 57x57 (3) 67x67 (4) 79x79 (5) 76x76	282x282	(0) 57x57 (1) 76x76 (2) 57x57 (3) 67x67 (4) 79x79 (5) 76x76	13374

Notes: (1) One thing we need to mention is that in our algorithm, the # of interactions is in fact the number of operations no matter how high the order of geometric moments is because we use the single term to compute the potential. While for FastCap the operations are a linear function of expansion order and the number of interactions. So for FastCap (2) the number of operations is two times of the number of interactions. (2) Besides, there are several iterations for the GMRES to converge, so total difference of operations of the new algorithm and FastCap is much large.

Second, we compare it with HiCap to show it works well in the case HiCap does not. See the test case shown in figure 21, it has three conductors. Two of them are very close to each other while the other one are very far from these two. We use a very small p for HiCap to compute the final value. If too small p is used, the accuracy will not be improved, but the running time increases greatly because the two close tiles are partitioned to too many unnecessary panels and the far panels is still not be partitioned to a necessary fine level. Table II shows the experimental results.

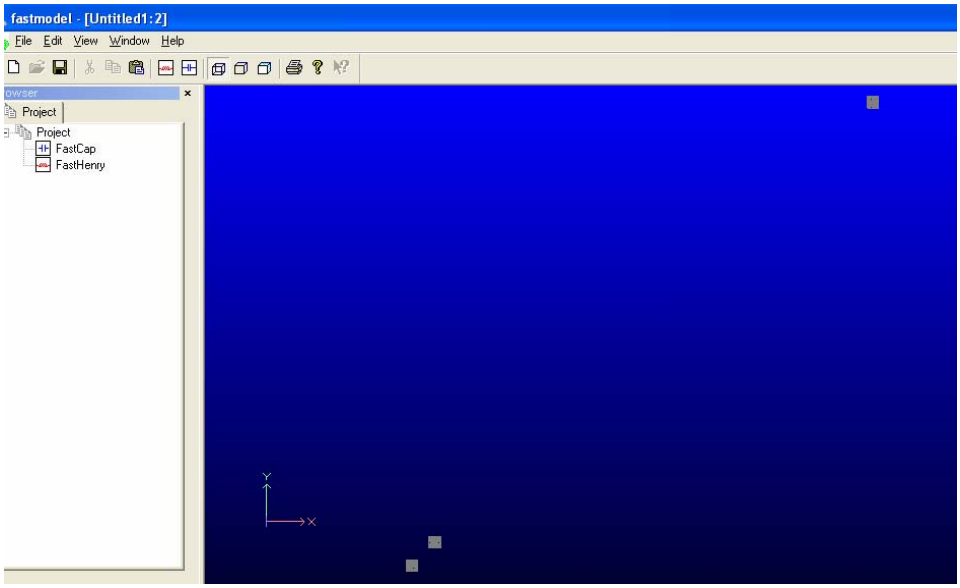


Fig. 21 Three conductors and one is very far.

TABLE II
COMPARISON WITH HICAP WITH A RECTANGLE

FastCap(2)		New Alg.(2)			HiCap		
Cap Matrix	**Panels #**	**Cap Matrix**	**Error**	**Panels #**	**Cap Matrix**	**Error**	**Panels #**
40.19 -5.123 -0.2182 -5.123 40.19 -0.2336 -0.2182 -0.2336 39.49	300	40.2323 -5.0687 -0.2993 -5.0687 40.1997 -0.2993 -0.2993 -0.2993 39.6064	0.45%	(0) 117 (1) 110 (2) 101	40.73 -5.25 -0.00 -5.25 40.73 -0.00 -0.20 0.22 36.07	7.6%	16383
N/A	400	N/A	0.98%	(0) 109 (1) 106 (2) 102 (3) 102	N/A	3.4%	8192

Third we compare the improvement of different geometric moments. We do this on several test cases, and table III shows the improvement. Experimental results show that when the geometry is complicated the improvement is much large. For the scheme of rectangle and 4 squares, we can see the accuracy improvement is small, while the experimental results show that the time increases much.

TABLE III
COMPARISON WITH DIFFERENT GEOMETRIC MOMENTS

Test Cases	1st order error	2nd order error (rectangle)	2nd order error (4 squares)
1 (300 tiles, 3 conductors)	1.1%	3.85%	3.5%
2 (145 tiles, 5 conductors)	7.5%	1.64%	1.44%
3 (282 tiles, 6 conductors)	3.5%	0.8%	0.7%
4 (154 tiles, 6 conductors)	4.2%	1.8%	2.0%
5 (282 tiles, 7 conductors)	2.8%	1.7%	1.6%

Fourth, we compare the new algorithm with FastCap in large cases. We know, in VLSI design, we often need to compute the capacitance over a large ground plane. In such cases, to capture the effects of the ground plane accurately, the ground plane should be discredited to small enough tiles, which usually causes the test case to be very large. Our new algorithm will save much time and memory in these cases. Table IV shows the experimental results. Figure 22 shows 3 conductors and a large ground plane.

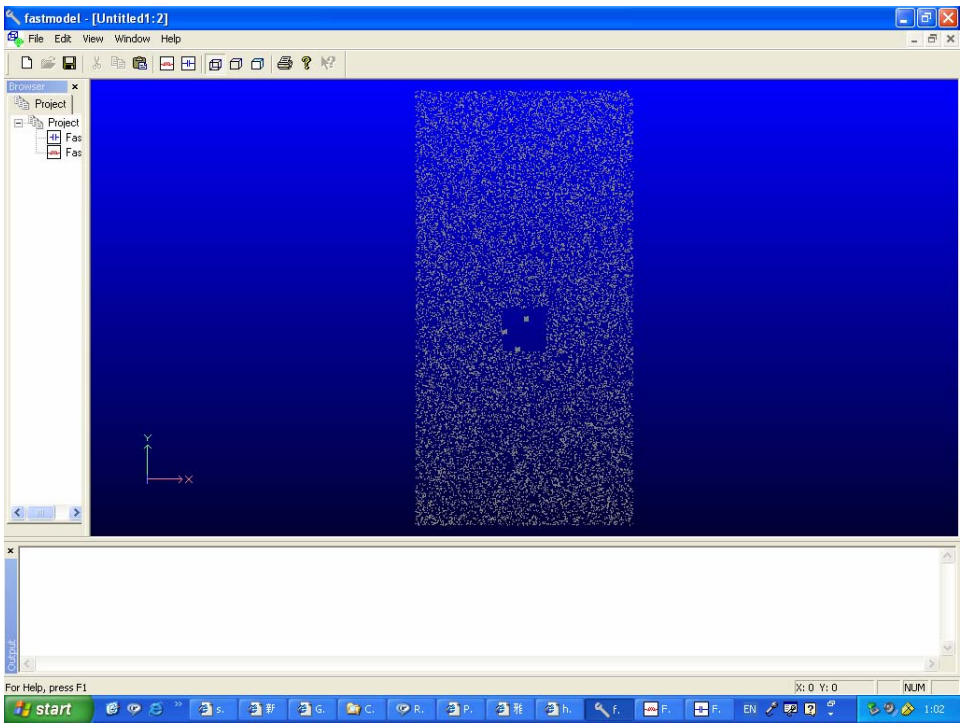


Fig. 22 Three conductors and a large ground plane.

TABLE IV
COMPARISON WITH FASTCAP IN LARGE CASES

Test Case	FastCap(2)		New Alg. (2)		ERR
	Time(s)	Mem(MB)	Time	Mem(MB)	
25g3-1	17* + 11.5**	130* + 142**	20 + 1.57	3.8 + 14	1.7%
25g3-2	16 + 16.2	125 + 141	20 + 0.932	3.8 + 13	2.2%
25g5	20 + 25	131 + 142	22 + 2.85	3.9 + 13	1.87%
3g3	17 + 30	150 +170	23 + 3.8	4.1 + 35	4.7%
3g5	--	--	27 + 5.8	4.1 + 41	--

* The time or memory of preprocessing phase.
** The time or memory of computation phase.
-- The time is too long to give the results.

CHAPTER IV

CONCLUSIONS

A three dimensional (3-D) capacitance extraction algorithm based on a kernel independent hierarchical method, geometric moments and charge distribution is described. It incorporates several techniques, which leads to a better overall performance for arbitrary interconnect systems. First, it hierarchically partitions the bounding box of all sources to build the partition tree and then uses a simple shape to match the low order moments of the geometry of each box in the partition tree to balance the accuracy and speed. The charge and potential are then approximated by a single term based on these simple shapes, which guarantees the algorithm is kernel independent. Using a simple shape to match the geometric moments rather than by Legendre polynomials has one benefit that we have closed forms to compute the geometric moments without any integration which will save much time and it is easy to be implemented within the hierarchical partition tree. Second, it uses an error estimate scheme to group far away parts with respect to certain selected conductor to trade unnecessary accuracy for speed, which is based on the fact that the charge variations on far parts are smooth and small. Experimental results show that our algorithm reduces the problem size greatly and at that same time maintains a satisfying accuracy.

REFERENCES

[1] L.J. Wardle, “An introduction to the boundary element methods,” *Numerical Solutions of Partial Differential Equations*, New York: North-Holland Publishing Company, 1982, pp. 289-312.

[2] K. Nabors and J. White, “FastCap: A multipole accelerated 3D capacitance extraction program,” *IEEE Trans CAD*, vol. 10, pp. 1447-1459, Nov. 1991.

[3] W. Shi, J. Liu, N. Kakani, and Y. Tiejun. “A fast hierarchical algorithm for 3D capacitance extraction,” *DAC*, pp.212-217, June 1998.

[4] J.R. Phillips and J.K. White, “A precorrected-FFT method for electrostatic analysis of complicated 3-D structures,” *in Proc. ICCAD*, 2000, pp. 448-455.

[5] S. Kapur and D.E. Long, “IES^3: A fast integral equation solver for efficient 3-dimensional extraction,” *IEEE Comput. Sci. and Eng*., vol.5, no.4, pp.60-67, Oct.-Dec. 1998.

[6] Y.L. Le Coz and R.B. Iverson, “A stochastic algorithm for high speed capacitance extraction in integrated circuits,” *Solid State Elect*, vol.35, no.7, pp. 1005-1012, July 1992.

[7] M. Beattie and L. Pileggi, “Electromagnetic parasitic extraction via a multipole method with hierarchical refinement,” *ICCAD*, 1999, pp. 437-444.

[8] S. Kapur and D. E. Long, “Large- scale capacitance calculation,” *DAC*,2000, pp. 744-749.

[9] L. Greengard and V. Rokhlin, "A fast algorithm for particle simulations," *Journal of Computational Physics*, vol.73, pp.325-348, 1987.

[10] Y. Saad and M.H. Schultz, “GMRES: A generalized minimal residula algorithm for solving nonsymmetric linear system,” *SIAM J.Sci. Stat. Comput.* , vol. 7 no. 3, pp.856-869, July 1986.

[11] G. H. Golub and C.F. Van Loan, *Matrix Computation*, 2nd edition, Baltimore, MD: Johns Hopkins University Press, 1989.

VITA

Name:	Wei Zhuang
Address:	Department of Electrical and Computer Engineering
	C/O Dr. Weiping Shi
	Texas A&M University
	College Station, TX 77843-3259
Email Address:	weizhuang@tamu.edu, johnsonzhuangsh@gmail.com

Education: B.S., Electronic Engineering, Xi'an JiaoTong University, 1996

M.S., Electronic Engineering, Xi'an JiaoTong University, 1999

M.S., Computer Engineering, Texas A&M University, 2006