Low Rank Matrix Algebra for the Method of Moments[1]

John Shaeffer
Matrix Compression Technologies, LLC
Marietta, Georgia
john@shaeffer.com

***Abstract*—This tutorial presents the use of low rank R_k matrix block formulations for advancing the problem size capability N of the Method of Moments full wave approach to solving Maxwell's integral equations. When MOM unknowns are spatially grouped, the group-group interaction matrix blocks become low rank for electrically large problems. A very significant advantage of this R_k property along with use of the Adaptive Cross Approximation leads to dramatic reduction in memory storage and in operations count. While early R_k approaches focused on iterative approaches, this work shows how R_k methods can be applied to direct solve LU factorization approaches and, for scattering problems, R_k methods can be used to compute the full polarization scattering matrix.**

Keywords—ACA, Direct Factor Method of Moments, electromagnetic scattering, R_k math.

I. BACKGROUND

Full wave solvers for Maxwell's integral equations are the much-preferred approach when they can be implemented. And direct factor rather than iterative solutions avoids the well-known failures of the latter. However, the direct factor computational cost for N unknowns is immense: N^2 for matrix storage, N^3 for matrix LU factorization and N^2 for each RHS excitation solution.

Researchers in the last 20-30 years have improved on this situation by recognizing that MOM system matrices, when unknowns are spatially grouped, lead to low rank blocks within the overall system matrix. Early researchers developed iterative solvers based on analytic Green's Function expansions (FMM and MLFMM). While these approaches significantly increased problem sizes they suffered from several problems: 1) They required special development of analytical Green's functions; 2) They were not well adapted for multiple RHS problems where the iterative scheme has to restart fresh for each new RHS vector which limits FMM and MLFMM usefulness for backscatter problems with many RHS's; 3) They often have slow and no convergence for the iterative solution for problems involving significant interactions; and 4) They often require matrix preconditioners.

More recently we have seen the development of the Adaptive Cross Approximation (ACA) for computing the low rank UV approximation to system matrix blocks, based on spatial grouping of unknowns. Whole new approaches to solving MOM system matrices has resulted.This includes the author's development in 2006 of Mercury MOM, the first code to LU factor a problem with one million unknowns [1,2] and more recently to five million unknowns [3]. Other researchers have also used R_k methods to extend MOM, each with their own methodology [4,5,6].

Significant ACA advantages over the early FMM/MLFMM approaches are: 1) There is no need for analytic Green's function formulations; 2) The ACA does not require computation of a full matrix block before obtaining its UV R_k approximation as only specific rows and columns of a block need computation; 3) The ACA low rank blocks of a system matrix can be used for iterative as well as direct LU factorizations. In this work a block LU factorization/solve solution is used where all of the Z, L and U factors, V and J are computed and stored in a compressed outer product form.

In the 50 years since the introduction of MOM [7], Moore's law should have suggested a problem size increase N, using factorization, of approximately $(2^{25})^{1/3} = 322$. Assuming a maximum N = 1000 for 1968 computer capability, we should have progressed by technology advance alone to N ~ 322,000. Clearly, we have done 20 to 40 times better. R_k formulations are responsible for this increase. Algorithms count!

This tutorial is organized as follows: II) Cobble stone spatial

[1] Early portions of this work were funded by the late Mrs. Arlene K Shaeffer. NASA funded validation and development of SIE/VIE capability as well the multiple SIE boundary condition capability.

Submitted On: September 30, 2018
Accepted On: October 3, 2018

grouping; III) What is low rank; IV) A Rank Fraction metric; V) R_k matrix math; VI) Thrill of R_k multiplication; VII) Agony of R_k addition; VIII) Adaptive cross approximation; IX) R_k Addition "Recompression"; X) LU factorization using the ACA; XI) Solve using R_k forms for scattering problems; XII) Mercury MOM scattering code and XIII) Concluding Remarks.

II. Cobble Stone Spatial Grouping

Spatial grouping of MOM unknowns is the starting point for creating low rank interaction blocks in the MOM system matrix. The key notion is to minimize the solid angle subtended by each pair of test and source groups of unknowns. As the distance between test/source groups increases, the subtended solid angle decreases thus reducing the numerical rank of the interaction, Fig. 2.

With spatial grouping for electrically large problems (as characterized by tens of thousands to several million unknowns with group sizes from 500 to 10,000 unknowns) most all blocks in the system matrix, except for diagonal self-blocks, become R_k. This includes not only Z blocks but also its L and U factors. For scattering problems with many RHS illumination angles, the RHS voltage excitation matrix is R_k as well as the current solution J and/or M:

$$\begin{bmatrix} [Z]_{11} & [Z]_{12} & [Z]_{13} & [Z]_{14} \\ [Z]_{21} & [Z]_{22} & [Z]_{23} & [Z]_{24} \\ [Z]_{31} & [Z]_{32} & [Z]_{33} & [Z]_{34} \\ [Z]_{41} & [Z]_{42} & [Z]_{43} & [Z]_{44} \end{bmatrix} \begin{bmatrix} [J]_1 \\ [M]_2 \\ [J]_3 \\ [M]_4 \end{bmatrix} \quad . (1)$$

$$= \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ [L]_{21} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ [L]_{31} & [L]_{31} & \mathbf{I} & \mathbf{0} \\ [L]_{41} & [L]_{41} & [L]_{41} & \mathbf{I} \end{bmatrix} \begin{bmatrix} [U]_{11} & [U]_{12} & [U]_{13} & [U]_{14} \\ \mathbf{0} & [U]_{22} & [U]_{23} & [U]_{24} \\ \mathbf{0} & \mathbf{0} & [U]_{33} & [U]_{34} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & [U]_{44} \end{bmatrix} \begin{bmatrix} [J]_1 \\ [M]_2 \\ [J]_3 \\ [M]_4 \end{bmatrix} = \begin{bmatrix} [V]_1 \\ [H]_2 \\ [V]_3 \\ [H]_4 \end{bmatrix}$$

Each R_k matrix block is written as a low rank approximation outer product of a column matrix times a row matrix:

$$\begin{bmatrix} \\ \mathbf{A} \\ \\ \end{bmatrix} \cong \begin{bmatrix} \mathbf{Au} \\ \\ \\ \end{bmatrix} \begin{bmatrix} \mathbf{Av} \end{bmatrix}. \quad (2)$$

This R_k form is typically computed using the Adaptive Cross Approximation (ACA) which means that only a few rows/columns of **A** are required. The numerical rank k will depend on the desired error tolerance. Each matrix block **A** never needs to be computed beforehand, i.e., **A** is virtual in the sense that only a subroutine capable of computing ACA required rows/columns of **A** is required. The cobble stone grouping algorithm is straight forward: .

- Create a box of all ungrouped unknowns;
- Compute the longest diagonal box vector;
- Pick an arbitrary starting point, typically the end of diagonal box vector;
- Compute distance from this point to all other unknowns;
- Sort these distances from close to far (e.g., Q sort);
- Pick nGroup of closet points to form 1st group; and
- Start over, but now with only ungrouped edges.

The grouping pattern is shown in Fig. 1 results.

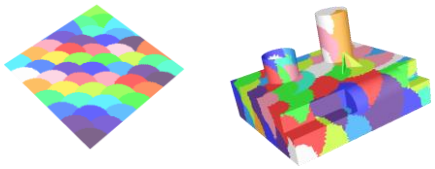

Fig. 1. Cobble stone grouping examples.

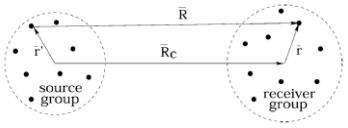

Fig. 2. Spatial source group test group.

III. What is Low Rank

Full rank of a (block) matrix is the minimum of the number of rows or columns. In this case each row or column is independent. In the less than full rank R_k case, each row or column is not independent, at least to within some numerical tolerance. In spatially grouped MOM system block matrices, we usually have $R_k << R$ and can approximate an (m,n) block as in (2) where the column **U** matrix is (m,k) and the row **V** matrix is (k,n).

Rank deficient block matrices means that the singular values of a Singular Value Decomposition (SVD) of that matrix drop precipitously. SVD theory is the basis for low rank outer product approximations where a matrix **A** is expressed as a product of three matrices **A = USV** where **U** and **V** are unitary where each column of **U** and row of **V** are independent and **S** is a diagonal only matrix of ordered real singular values. In the example below, let the 3rd singular value be less than some tolerance ε, then **A** can be approximated as the outer product of a column matrix times a row matrix:

$$\mathbf{A} = \mathbf{USV} = \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \\ u_{31} & u_{32} & u_{33} \end{pmatrix} \begin{pmatrix} s_{11} & 0 & 0 \\ 0 & s_{22} & 0 \\ 0 & 0 & s_{33} \end{pmatrix} \begin{pmatrix} v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \\ v_{31} & v_{32} & v_{33} \end{pmatrix}. \quad (3)$$

With $S_{11} > S_{22} > S_{33}$. If $S_{33} < \varepsilon \sim 0$, then

$$\mathbf{A} = \mathbf{USV} \approx \begin{pmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \\ u_{31} & u_{32} \end{pmatrix} \begin{pmatrix} s_{11} & \\ & s_{22} \end{pmatrix} \begin{pmatrix} v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \end{pmatrix} = \begin{pmatrix} & \\ & \\ & \end{pmatrix} \begin{pmatrix} & \\ & \end{pmatrix}. \quad (4)$$

The numerical rank of **A** is the number of singular values greater than some judiciously chosen value or tolerance ε. If the singular values decrease exponentially, as they do in many MOM problems with unknown grouping, then **A** is very compressible. Fig. 3 shows the single precision (6 digits)

singular values for a 220 by 214 MOM Z interaction block (rank R = 214) on a $\log_{10}$ scale. There we see that the singular values have decreased to machine precision (6 orders of magnitude) by k = 30.

The next question is how to set the tolerance ε. This involves the Frobenius norm of matrix **A** [8] which is nothing more than the square root of the sum of the squares of all elements of **A**:

$$\|\mathbf{A}\| = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} \left\| a_{i,j} \right\|^2} . \tag{5}$$

The SVD approximation error in the outer product approximation is given in terms of singular values of **A**. Let matrix **A** have rank r and the approximation rank k (using singular values 1 to k). The error involves the remaining singular values from k+1 to r. If $\sigma_k / \sigma_1 \sim 10^{-5}$, the fractional error in the Frobenius norm is small [8]:

$$\frac{\| \mathbf{A} - \mathbf{A}_k \|_F}{\| \mathbf{A} \|_F} = \frac{\sqrt{\sigma_{k+1}^2 + \cdots + \sigma_r^2}}{\sqrt{\sigma_1^2 + \cdots + \sigma_r^2}} . \tag{6}$$

We see that low rank and tolerance are related and that if the singular values of **A** decrease exponentially, then the compressed form for **A** (2) has significantly less memory requirements and perhaps more importantly, significantly less operations count for matrix multiply operations. The relative norm error then becomes the compressed matrix of approximate rank k minus the full matrix divided by norm of the full matrix:

$$\frac{\left\| \mathbf{A}_{\mathbf{compressed}}(k) - \mathbf{A} \right\|}{\|\mathbf{A}\|} = \frac{\|\mathbf{Au}\ \mathbf{Av} - \mathbf{A}\|}{\|\mathbf{A}\|} \leq \varepsilon . \tag{7}$$

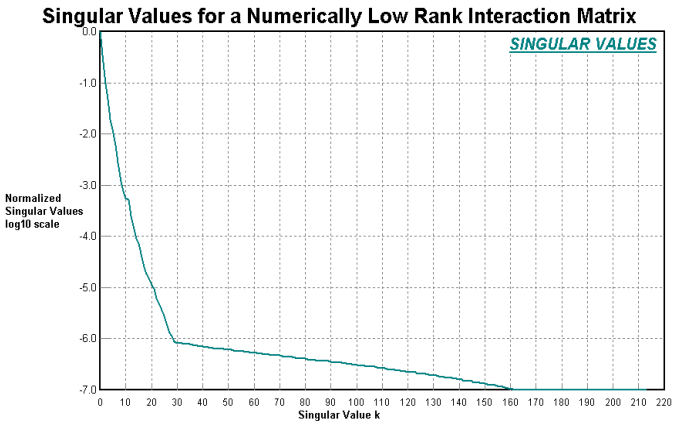

Fig. 3. Singular values for a Rk MOM interaction matrix.

IV. A Rank Fraction Metric

An important metric for the compressibility of a matrix is its Rank Fraction (RF) which is defined as the ratio of compressed or approximation memory storage to that of the full matrix. For an m x n matrix with approximation rank k, RF is:

$$RF = \frac{k(m+n)}{mn} . \tag{8}$$

If m = n, RF = 2k/m. If **A** is 99% compressed with only 1% stored, then RF = 0.01. Often a dB scale is used.

V. R_K Matrix Math

The two very significant advantages of R_k formulations are the reduced storage for a (block) matrix and the operations count involving matrix multiplication. For full matrix reference, the storage for an m x n matrix is O(mn) while the operations count for the multiplication of two matrices, one m x k, the other k x n is O(mkn).

R_k matrix math starts by reviewing inner and outer matrix products. The dot or inner product of two vectors, each of k elements, is a row times a column and is characterized by a small operations count of order O(k):

$$r = \mathbf{x} \bullet \mathbf{y} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}, \tag{9}$$

while a 2 x k row inner product with a k x 2 column matrix has an operations count of O(4k):

$$\begin{bmatrix} \sqrt{\sigma_{\theta\theta}} & \sqrt{\sigma_{\theta\varphi}} \\ \sqrt{\sigma_{\varphi\theta}} & \sqrt{\sigma_{\varphi\varphi}} \end{bmatrix} = \mathbf{R}\ \mathbf{J} = \begin{bmatrix} R_{1\theta} & R_{2\theta} & R_{3\theta} & R_{4\theta} \\ R_{1\varphi} & R_{2\varphi} & R_{3\varphi} & R_{4\varphi} \end{bmatrix} \begin{bmatrix} J_{1\theta} & J_{1\varphi} \\ J_{2\theta} & J_{2\varphi} \\ J_{3\theta} & J_{3\varphi} \\ J_{4\theta} & J_{4\varphi} \end{bmatrix} . \tag{10}$$

The outer product of a single column times a single row is a full matrix, albeit with rank 1. Let **U** be (m x 1) and **V** (1 x n), then **A** is (m x n), **A** = **U V** with $a_{ij} = u_i v_j$:

$$\begin{bmatrix} u_1 \\ \vdots \\ \vdots \\ u_m \end{bmatrix} \begin{bmatrix} v_1 & \vdots & \vdots & v_n \end{bmatrix} = \begin{bmatrix} a_{11} & \vdots & \vdots & a_{1n} \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & \vdots & \vdots & a_{mn} \end{bmatrix} . \tag{11}$$

This example dramatically shows the utility of storing a matrix in its **UV** outer product form. The ***equivalent*** information can be stored in O(m+n) versus O(mn). If m = n, the outer product storage is only 2/m of that required for the full matrix. No additional information can be obtained by storing the full matrix. Matrix algebra involving **A** in its compressed UV form has significantly less operations count.

The sum of two rank 1 outer products is a rank $R_k = 2$ matrix

where $a_{i,j} = u_{i,1} v_{1,j} + u_{i,2} v_{2,j}$:

$$\begin{bmatrix} a_{11} & \vdots & \vdots & a_{1n} \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & \vdots & \vdots & a_{mn} \end{bmatrix} = \sum_{k=1}^{k=2} \begin{bmatrix} u_1 \\ \vdots \\ \vdots \\ u_m \end{bmatrix}_k \begin{bmatrix} v_1 & \vdots & \vdots & v_n \end{bmatrix}_k \quad . \quad (12)$$

$$= \begin{bmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \\ u_{31} & u_{32} \\ u_{41} & u_{42} \end{bmatrix} \begin{bmatrix} v_{11} & v_{12} & v_{13} & v_{14} \\ v_{21} & v_{22} & v_{23} & v_{24} \end{bmatrix}$$

R_k storage is O(2(m+n)) or if m = n, O(4m) compared to $O(m^2)$, a storage ratio of 4/m.

The sum of k rank 1 outer products is:

$$\mathbf{A} = \mathbf{UV} = \left(\begin{bmatrix} \mathbf{u} \\ \\ \\ \end{bmatrix} [\mathbf{v} \quad \quad \quad] \right)_{k=1} + \left(\begin{bmatrix} \mathbf{u} \\ \\ \\ \end{bmatrix} [\mathbf{v} \quad \quad \quad] \right)_{k=2} + \cdots \left(\begin{bmatrix} \mathbf{u} \\ \\ \\ \end{bmatrix} [\mathbf{v} \quad \quad \quad] \right)_{k=k}$$

$$= \sum_{i=1}^{i=k} \left(\begin{bmatrix} \mathbf{u}_i \\ \\ \\ \end{bmatrix} [\mathbf{v}_i \quad \quad \quad] \right)_i \quad .(13)$$

$$= \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & . & \mathbf{u}_k \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 & . & . & . \\ \mathbf{v}_2 & . & . & . \\ . & . & . & . \\ \mathbf{v}_k & . & . & . \end{bmatrix}$$

This is the form for the ACA approximation of k rank 1 outer products used to approximate (compress) **A**.

The product of a matrix **A** with a column vector **y** is a column vector **x, x = A y** which is a matrix vector product:

$$\begin{bmatrix} \mathbf{x} \\ \\ \\ \end{bmatrix} = \begin{bmatrix} \\ \mathbf{A} \\ \\ \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ \\ \\ \end{bmatrix} . \quad (14)$$

The product of a row vector with a matrix **A** is another row vector, $\mathbf{x^T = [Ay]^T = y^T A^T}$ where $\mathbf{x^T}$ and $\mathbf{y^T}$ are row vectors. This is a vector matrix product:

$$\begin{bmatrix} \quad \mathbf{x}^t \quad \end{bmatrix} = \begin{bmatrix} \quad \mathbf{y}^t \quad \end{bmatrix} \begin{bmatrix} \\ \mathbf{A} \\ \\ \end{bmatrix}^T . \quad (15)$$

VI. Thrill of R_K Multiplication

The product of two or more R_k matrices has a significantly low operations count compared to that for their full-size counterparts. R_k matrix-matrix multiply reduces to one of: a) matrix-vector multiplication; b) vector-matrix multiplication; or c) matrix-vector plus a small matrix-matrix multiply. Let us use three cases.

Let case 1 be **C = A B** where **A** is R_k and **B** is full. Then **C** is R_k and note that **Cu** = **Au** and **Cv = Av B**:

$$\begin{bmatrix} \mathbf{Cu} \\ \\ \\ \end{bmatrix} [\mathbf{Cv} \quad \quad] = \begin{bmatrix} \mathbf{Au} \\ \\ \\ \end{bmatrix} [\mathbf{Av} \quad \quad] \begin{bmatrix} \quad \\ \mathbf{B} \\ \\ \end{bmatrix} \quad .(16)$$

$$\begin{bmatrix} \mathbf{Cu} \\ \\ \\ \end{bmatrix} = \begin{bmatrix} \mathbf{Au} \\ \\ \\ \end{bmatrix} ; [\mathbf{Cv} \quad \quad] = \left([\mathbf{Av} \quad \quad] \begin{bmatrix} \quad \\ \mathbf{B} \\ \\ \end{bmatrix} \right)$$

The opts count is that of a row matrix with a full matrix.

For case 2, let **C = A B** where **A** is full and **B** is R_k then **C** is R_k and note that **Cu = A Bu** and **Cv = Bv**:

$$\begin{bmatrix} \mathbf{Cu} \\ \\ \\ \end{bmatrix} [\mathbf{Cv} \quad \quad] = \left(\begin{bmatrix} \mathbf{A} \\ \\ \\ \end{bmatrix} \begin{bmatrix} \mathbf{Bu} \\ \\ \\ \end{bmatrix} \right) [\mathbf{Bv} \quad \quad] \quad .(17)$$

$$\begin{bmatrix} \mathbf{Cu} \\ \\ \\ \end{bmatrix} = \left(\begin{bmatrix} \mathbf{A} \\ \\ \\ \end{bmatrix} \begin{bmatrix} \mathbf{Bu} \\ \\ \\ \end{bmatrix} \right) ; [\mathbf{Cv} \quad \quad] = [\mathbf{Bv} \quad \quad]$$

Again, the opts count is that of a full matrix with a column matrix.

For case 3 let **C = A B** where each matrix is R_k and note than **Cu = Au** or **Cv = Bv** depending on which gives the lowest rank for **C**. Assuming **Au** is the lower rank option then:

$$\begin{bmatrix} \mathbf{Cu} \\ \\ \\ \end{bmatrix} [\mathbf{Cv} \quad \quad] = \begin{bmatrix} \mathbf{Au} \\ \\ \\ \end{bmatrix} [\mathbf{Av} \quad \quad] \begin{bmatrix} \mathbf{Bu} \\ \\ \\ \end{bmatrix} [\mathbf{Bv} \quad \quad] \quad .(18)$$

$$\begin{bmatrix} \mathbf{Cu} \\ \\ \\ \end{bmatrix} = \begin{bmatrix} \mathbf{Au} \\ \\ \\ \end{bmatrix} ; [\mathbf{Cv} \quad \quad] = \left([\mathbf{Av} \quad \quad] \begin{bmatrix} \mathbf{Bu} \\ \\ \\ \end{bmatrix} \right) [\mathbf{Bv} \quad \quad]$$

For cases 1 and 2 matrix-matrix multiply reduces to matrix-vector multiply. For case 3 we have a low operations count inner product multiply followed by a "small" matrix - row matrix multiply. In each of these cases the operations count involving R_k matrices is significantly lower than their full-size matrix counter parts. When multiplying R_k matrices, one strives to do the low operations count inner products first as suggested in case 3 and then a matrix-vector or vector-matrix multiply. Often the left- or right-hand side of a R_k product is simply either the left or right-hand side **U** or **V**.

VII. Agony of R_K Addition

The sum of two or more R_k matrices has undesirable consequences. Let **D = A + B + C** where **A**, **B** and **C** are R_k:

$$\begin{bmatrix} \mathbf{A}u \\ \\ \end{bmatrix}\begin{bmatrix} \mathbf{A}v & \end{bmatrix}+\begin{bmatrix} \mathbf{B}u \\ \\ \end{bmatrix}\begin{bmatrix} \mathbf{B}v & \end{bmatrix}+\begin{bmatrix} \mathbf{C}u \\ \\ \end{bmatrix}\begin{bmatrix} \mathbf{C}v & \end{bmatrix} \quad . \quad (19)$$

$$=\left[\begin{array}{c|c|c} \mathbf{A}u & \mathbf{B}u & \mathbf{C}u \\ & & \end{array}\right]\left[\begin{array}{c} \mathbf{A}v \\ \hdashline \mathbf{B}v \\ \hdashline \mathbf{C}v \end{array}\right]=\begin{bmatrix} \mathbf{D}u \\ \\ \end{bmatrix}\begin{bmatrix} \mathbf{D}v & \end{bmatrix}$$

A terrible thing just happened, the R_k rank of **D** is sum of the R_k ranks of **A**, **B**, and **C**. This does not mean that **D** is not low rank, but it does mean that **D** requires "recompression." More on this in section IX.

VIII. Adaptive Cross Approximation

A number of techniques can be used to compute the low rank R_k approximation to a (block) matrix **A**. SVD and QR factorizations are possible but they suffer from two very significant drawbacks: a) all of the matrix **A** must be computed beforehand and b) the operations count is prohibitive, typically $O(m^3)$. The Adaptive Cross Approximation (ACA) popularized in [9,11] does not suffer these problems. The ACA operations count is $k^2(m+n)$ and, most significantly, the ACA does not require a prior computation of the full matrix **A**. The ACA only requires various rows and columns of **A**. **A** is virtual in the sense that it only needs to exist as a subroutine which can compute needed rows and columns required by the ACA. It is not compressed in the sense of an SVD or QR.

There are several requirements for successful ACA computation. First, the singular values must drop exponentially which is usually the case for MOM PEC problems with spatially grouped unknowns. Second, the integral operator should have a smooth decay with distance. Our free space Green's function is not quite smooth but the ACA still works well. For the PEC L operator, ACA works very well. For the curl K operator, where the integrand also has geometric terms, ACA may be problematic since sub-blocks can be exactly low rank. In this case special efforts are required to obtain the UV approximation. The ACA is adaptive in the sense that the algorithm always looks for the largest element in a row/column to compute the next k^{th} row/column in the approximation. The ACA terminates when the desired approximation tolerance is obtained.

A six-step algorithm as outlined in [9] starts by writing (block) matrix **A** as the sum of its approximation plus its error and then solving for the error:

$$\mathbf{A} = \mathbf{A}_{approximation} + \mathbf{E}$$
$$\mathbf{E} = \mathbf{A} - \mathbf{A}_{approximation} = \mathbf{A} - \mathbf{UV} \quad . \quad (20)$$

The goal at each step is to make the error as small as possible with each successive rank one **UV** outer product term where successive terms add columns to **U** and rows to **V**. At the k^{th} step, the error is:

$$\mathbf{E}_k = \mathbf{A} - \mathbf{U}_k\mathbf{V}_k = \mathbf{A} - \sum_{p=1}^{k}\left(\begin{bmatrix} \mathbf{u}_p \\ \vdots \\ \end{bmatrix}\begin{bmatrix} \mathbf{v}_p & \dots & \dots & \dots \end{bmatrix}\right)_p \quad . \quad (21)$$

The next row/column in the approximation is obtained by setting the pivot row/column error to zero in order to compute the next **UV** outer product term. The choice of row/column is made by finding the maximum element in the last **UV** approximation which is called pivoting. Note that only rows/ columns of **E**, **A** and **UV** are required at each step.

The ACA approximation stops when the norm of the next **UV** term is less than the desired tolerance, where $\mathbf{A}_k$ is a recursive norm computed based only on terms to date [9]:

$$\frac{\|\mathbf{u}_k\|\|\mathbf{v}_k\|}{\|\mathbf{A}_k\|} \le \varepsilon \; . \quad (22)$$

Fig. 4 shows the ACA approximation error for a 220 x 214 MOM interaction matrix as a function of the iteration. Convergence to 10^{-4} is achieved by k=20. Also plotted are the corresponding normalized singular values and the true error norm as computed from the full matrix.

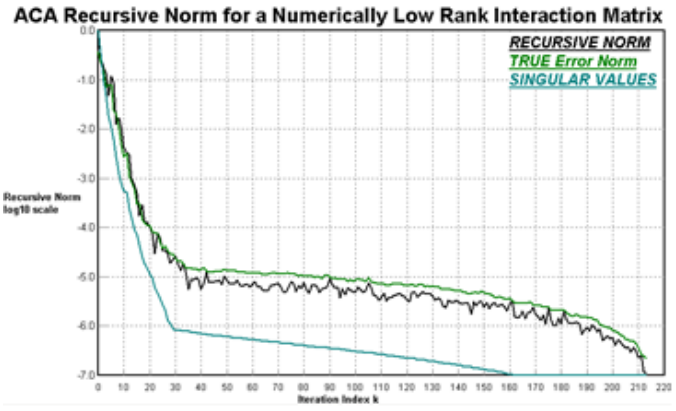

Fig. 4. ACA recursive norm, true error norm and singular values for an R_k MOM interaction matrix.

For a given outer product error tolerance the SVD approximation has the least rank [8]; however, the ACA approach is still quite acceptable.

An example of the Z block rank fraction compression (on a 20 dB scale) achieved by the ACA for an open pipe target with 92,220 unknowns, block size 800, for $\varepsilon = 10^{-5}$ is shown in Fig. 5. The corresponding LU block rank fraction compression is shown in Fig. 6. While there is some fill-in, the LU blocks are never the less very sparse.

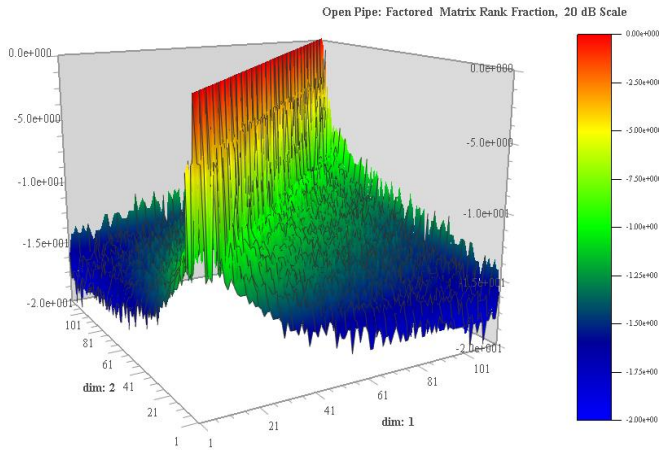

Fig. 5. Rank fraction, 20 dB scale, for block matrices for an LU factored matrix.

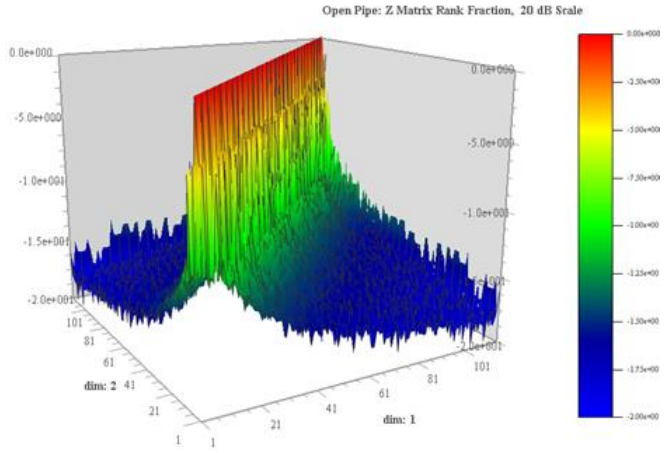

Fig. 6. Rank fraction, 20 dB scale for Z block matrices in system matrix.

IX. R_K Addition "Recompression"

As we saw in section VII, the sum of a number of R_k terms is itself R_k. However, if we add all terms directly, we end up with the sum R_k equal to sum of the ranks of each term in the sum. This is untenable. A methodology is required to accomplish such R_k sums. Two possible addition "recompression" approaches can be utilized.

Before we outline each approach, let us put into perspective the sizes involved when using either of these techniques when doing block wise LU R_k factorization and solve operations. As an example, let us consider a one million unknown problem where the R_k group size is 5000. This translates into a system matrix of size 200 x 200 blocks with each block 5000 x 5000. The LU factorization and solve formulas (discussed below) have sums involving up to 200 terms of (5000 x k) and (k x 5000) UV terms where k is the rank of each term.

In the 1st approach [2,10] we utilize the ACA to directly compute the R_k form for **D = UV** using the rows/columns from each term S_k in the sum:

$$\mathbf{D} = \mathbf{UV} = \sum_p \left(\begin{bmatrix} \mathbf{S}_u \\ \\ \end{bmatrix} \begin{bmatrix} \mathbf{S}_v & & \end{bmatrix} \right)_p . \tag{23}$$

After choosing an ACA tolerance ε, the required ACA rows are easily computed from the RHS of (23):

$$\begin{bmatrix} \rightarrow & \rightarrow & \rightarrow \end{bmatrix} = \sum \left(\begin{bmatrix} \rightarrow \\ \\ \end{bmatrix} \begin{bmatrix} \downarrow & \downarrow & \downarrow \end{bmatrix} \right)_p , \tag{24}$$

while the required ACA columns are computed from the RHS of (23) as:

$$\begin{bmatrix} \downarrow \\ \downarrow \\ \downarrow \end{bmatrix} = \sum \left(\begin{bmatrix} \rightarrow \\ \rightarrow \\ \rightarrow \end{bmatrix} \begin{bmatrix} \downarrow & \end{bmatrix} \right)_p . \tag{25}$$

This is a speedy process which encompasses a vector-matrix or matrix-vector multiply. In spite of the tedious looking process, it can be accomplished surprising quickly. This procedure is in essence a methodology for computing the R_k sum using the ACA without the necessary requirements for many QR recompressions required when using the 2nd approach. It should be noted that many of these operations can be computed in parallel, OpenMP for outer loops and BLAS libraries for lower level matrix-vector operations.

The 2nd "recompression" approach involves doing a QR factorization of the U and of V components of **D = UV** coupled with an SVD compression using the desired tolerance. Details can be found in [11], but briefly the process involves:

$\mathbf{U} = \mathbf{Q_u R_u}; \mathbf{V} = \mathbf{Q_v R_v}$ Perform a QR factorizaton on U and V

$$\mathbf{D} = \mathbf{Q_u R_u R_v^T Q_v^T} \tag{26}$$

$\mathbf{R_u R_v^T} = \hat{\mathbf{U}}\hat{\mathbf{\Sigma}}\hat{\mathbf{V}}$ Perform SVD on the small $\left[\mathbf{R_u R_v^T}\right]$ matrix

X. LU Factorization Using the ACA

The rationale for solving the MOM system matrix via LU factorization is well known [4,12]. In this section we will examine the procedure for a symmetric system matrix and, most importantly, show a technique for performing R_k addition using the ACA which builds on R_k multiplication.

The symmetric matrix factor form [13] is:

$$\mathbf{Z} = \mathbf{U^T D U}$$

$$= \begin{bmatrix} \mathbf{I} & & & \\ \mathbf{U_{12}^T} & \mathbf{I} & & \\ \mathbf{U_{13}^T} & \mathbf{U_{23}^T} & \mathbf{I} & \\ \mathbf{U_{14}^T} & \mathbf{U_{24}^T} & \mathbf{U_{34}^T} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{D_{11}} & & & \\ & \mathbf{D_{22}} & & \\ & & \mathbf{D_{33}} & \\ & & & \mathbf{D_{44}} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{U_{12}} & \mathbf{U_{13}} & \mathbf{U_{14}} \\ & \mathbf{I} & \mathbf{U_{23}} & \mathbf{U_{24}} \\ & & \mathbf{I} & \mathbf{U_{34}} \\ & & & \mathbf{I} \end{bmatrix}, \tag{27}$$

which can be recast into standard form as:

$$\mathbf{Z}=\begin{bmatrix}\mathbf{I}&&&\\ \mathbf{L}_{21}&\mathbf{I}&&\\ \mathbf{L}_{31}&\mathbf{L}_{32}&\mathbf{I}&\\ \mathbf{L}_{41}&\mathbf{L}_{42}&\mathbf{L}_{43}&\mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{U}'_{11}&\mathbf{U}'_{12}&\mathbf{U}'_{13}&\mathbf{U}'_{14}\\ &\mathbf{U}'_{22}&\mathbf{U}'_{23}&\mathbf{U}'_{24}\\ &&\mathbf{U}'_{33}&\mathbf{U}'_{34}\\ &&&\mathbf{U}'_{44}\end{bmatrix}. \quad (28)$$

The solution for each **U** block [2] is:

$$\mathbf{U}_{iBlk,jBlk}=\mathbf{Z}_{iBlk,jBlk}-\sum_{pBlk=1}^{iBlk-1}\mathbf{L}_{iBlk,pBlk}\mathbf{D}^{-1}_{pBlk,pBlk}\mathbf{U}_{pBlk,jBlk}, \quad (29)$$

where the $\mathbf{D}^{-1}$'s are the dense (not R_k) inverse diagonal blocks (computed using standard LU factorization). Note the sum terms involving R_k forms for $\mathbf{U}^T$ and **U**. Depending on total number of unknowns, these blocks can range in size from 2500 to 10,000. The R_k form for the **U**'s and **Z**'s (29) becomes:

$$\left(\left[\mathbf{U}_u\right]\left[\mathbf{U}_v\quad\right]\right)_{i,j}=\left(\left(\left[\mathbf{Z}_u\right]\left[\mathbf{Z}_v\quad\right]\right)_{i,j}-\sum_{p=1}^{p=i-1}\left(\left[\mathbf{L}_u\right]\left[\mathbf{L}_v\quad\right]\right)_{i,p}\left[\mathbf{D}\quad\right]^{-1}_{p,p}\left(\left[\mathbf{U}_u\right]\left[\mathbf{U}_v\quad\right]\right)_{p,j}\right)_{i,j=iBlk,jBlk}. \quad (30)$$

In this expression, we need to compute the left-hand **U** in R_k form, using the ACA, where all sum terms on the right-hand side are known. In order to accomplish this, the ACA needs rows/columns of the right-hand side. The approach is to recast the RHS as a sum of S matrices in R_k form as shown in section VII:

$$\sum_{p=1}^{k-1}\left(\left[\mathbf{S}_u\right]\left[\mathbf{S}_v\quad\right]\right)_p. \quad (31)$$

Each $\mathbf{S}_p$ matrix in the sum is:

$$\left[\mathbf{S}_u\right]\left[\mathbf{S}_v\quad\right]=\left[\mathbf{Z}_u\right]\left[\mathbf{Z}_v\quad\right] \text{ or}$$

$$\left[\mathbf{S}_u\right]\left[\mathbf{S}_v\quad\right]=\left[\mathbf{L}_u\right]\left(\left[\mathbf{L}_v\quad\right]\left[\mathbf{D}\quad\right]^{-1}\left[\mathbf{U}_u\right]\left[\mathbf{U}_v\quad\right]\right). \quad (32)$$

Each S_k sum term is computed using R_k multiplication before using the 1st ACA "recompression" approach outlined in section VII.

XI. Solve Using R_K Forms for Scattering Problems

For scattering problems, once the LU factored matrix is computed in R_k form, the solution for the currents can be computed in a similar approach where the RHS voltage forcing function and resulting currents may also be expressed block wise in R_k form:

$$\mathbf{V}^{iPol=\theta\text{ or }\varphi}=\begin{bmatrix}v^{\theta\text{ or }\varphi}_{1,1}&\cdots&v^{\theta\text{ or }\varphi}_{1,nAng}\\ &\vdots&\\ v^{\theta\text{ or }\varphi}_{k,1}&&v^{\theta\text{ or }\varphi}_{k,nAng}\end{bmatrix}=\begin{bmatrix}\mathbf{V}u\\ \\ \end{bmatrix}\left[\mathbf{V}v\quad\right]. \quad (33)$$

The resulting block wise current solution may also be expressed in R_k from:

$$\mathbf{J}^{iPol=\theta\text{ or }\varphi}=\begin{bmatrix}j^{\theta\text{ or }\varphi}_{1,1}&\cdots&j^{\theta\text{ or }\varphi}_{1,nAng}\\ &\vdots&\\ j^{\theta\text{ or }\varphi}_{k,1}&&j^{\theta\text{ or }\varphi}_{k,nAng}\end{bmatrix}=\begin{bmatrix}\mathbf{J}u\\ \\ \end{bmatrix}\left[\mathbf{J}v\quad\right]. \quad (34)$$

The LU block wise forward/back solve formulas [2] are:

$$\mathbf{Y}_i=\left[\mathbf{V}_i-\sum_{p=1}^{i-1}\mathbf{L}_{ip}\mathbf{D}^{-1}_{pp}\mathbf{Y}_p\right], \quad (35)$$

$$\mathbf{J}_i=\mathbf{D}_i^{-1}\left[\mathbf{Y}_i-\sum_{p=n-1}^{i-1}\mathbf{U}_{ip}\mathbf{J}_p\right]. \quad (36)$$

Finally, the full polarization scattering matrix is computed using the R_k forms for the Row measurement matrix [7]. In the backscatter case **R** is the transpose of the monostatic incident block wise plane wave excitation forcing function **V** [7], each is in R_k form:

$$\begin{aligned}\mathbf{R}&=\mathbf{V}^T\\ &=\mathbf{R}_u\mathbf{R}_v=\mathbf{V}_v^T\mathbf{V}_u^T.\end{aligned} \quad (37)$$

The block wise column current solution **J** in R_k form [7] is then used with the row measurement matrix to obtain the full polarization scattering matrix:

$$\begin{bmatrix}\sqrt{\sigma_{\theta\theta}}&\sqrt{\sigma_{\theta\varphi}}\\ \sqrt{\sigma_{\varphi\theta}}&\sqrt{\sigma_{\varphi\varphi}}\end{bmatrix}=\begin{bmatrix}\mathbf{R}_\theta\\ \mathbf{R}_\varphi\end{bmatrix}\begin{bmatrix}\mathbf{J}_\theta&\mathbf{J}_\varphi\\ &\\ &\end{bmatrix}. \quad (38)$$

XII. Mercury MOM Scattering Code

The computer code MERCURY MOM, [1-3] was built based on the R_k math discussed above. This code is a frequency domain MOM SIE/VIE scattering code which uses RWG triangles and SWG tetrahedrons. SIE boundary conditions include PEC, dielectric, R card, thin dielectric and IBC. Junctions are included between SIE RWG triangles and VIE tetrahedrons. The three EM operators $\mathcal{L}$, $\mathcal{K}$ and $\mathcal{K}$_tilde are utilized to compute three of the four general unknown SIE surface currents: J^+, J^-, M^+. A Galerkin symmetric matrix is utilized, and the full polarization scattering matrix is computed.

Mercury MOM was designed to be run on inexpensive work station class computers located in engineering design environments on an engineers' desk. The computer used by the

author for the five million unknown result [3] was a circa 2013 two socket Intel Xeon processor with 10 cores each (20 total) with a cost of $8900.

XIII. Concluding Remarks

Spatial grouping / R_k math / ACA / LU factorization-solve techniques have allowed our community to significantly extend the capability of full wave solutions of Maxwell's equations for electrically large bodies, all within the standard MOM framework. This means that increasely large problem sizes can be done in MOM full wave form before we must resort back to problematic and approximate high frequency methods [14] of PO/GO/PTD/GTD/SBR. R_k methods can be implemented on inexpensive workstation class computers located on design engineers' desks and allow a much greater exploration of design space parameters.

The quote in [12] succinctly sums up our quest: "A good computation is one that does the least computation for the right answer" which for practical engineering purposes we might change 'right answer' to 'right/good enough answer.'

References

[1] J. Shaeffer, "Direct solve of electrically large integral equations for problem sizes to 1 M unknowns," IEEE Trans. Antennas Propag., vol. 56, no. 8, pp. 2306-2313, Aug. 2008.

[2] J. Shaeffer, "Direct solve of electrically large integral equations for problem Sizes to 1M unknowns," NASA/CR-2008-215353, Sept. 2008

[3] J. Shaeffer, "Five million unknown MOM LU factorization on a PC workstation," Antenna Measurement Techniques Association Meeting, Long Beach, CA, Oct. 11-16, 2015.

[4] A. Heldring, J. Rius, J. M. Tamayo, J. Parron, and E. Ubeda, "Multiscale compressed block decomposition for fast direct solution of method of moments linear system," IEEE Trans. Antennas Propag., vol. 59, no. 2, pp. 526-536, Feb. 2011.

[5] A. Manic, A. Smull, F. H. Rouet, X. S. Li, and B. Notaros, "Efficient scalable parallel higher order direct MOM-SIE method with hierarchically semiseparable structures for 3-d scattering," IEEE Trans. Antennas Propag., vol. 65, no. 5, pp. 2467-2478, May 2017.

[6] H. Guo, Y. Liu, J. Hu, and E. Michielssen, "A butterfly-based direct integral-equation solver using hierarchical LU factorization for analyzing scattering from electrically large conducting objects," IEEE Trans. Antennas Propag., vol. 65, no. 9, pp. 4742-4650, Sept. 2017.

[7] R. F. Harrington, Field Computation by Moment Methods. New York, NY, USA: Macmillan, 1968.

[8] L. Trefethen and D. Bau, Numerical Linear Algebra. Philadelphia, PA: Soc. Indust. Appl Math. (SIAM), 1997.

[9] S. Kurz, O.Rain, and S. Rjasanow, "Application of the adaptive cross approximation technique for the coupled BE-FE solution of electromagnetic problems," Presented at the Int. Association Boundary Element Methods Conf., IABEM 2002, Austin, TX, May 28-30, 2002.

[10] J. F. Shaeffer, "Systems and methods for analysis and design of radiating and scattering objects," United States Patent No. 7,844,407 B1, Nov. 30, 2010.

[11] M. Bebendorf, Hierarchical Matrices. Berlin, Springer-Verlag, 2008.

[12] S. Ambikasaran, "Fast algorithms for dense numerical linear algebra and applications," Ph.D. Dissertation, Stanford University, Aug. 2013.

[13] G. Golub and C. Van Loan, Matrix Computations. 3rd. Baltimore, MD: The John Hopkins Univ. Press, 1996.

[14] E. F. Knott, J. F. Shaeffer, and M. T. Tuley, Radar Cross Section. 2nd Edition, Raleigh, NC: Scitech Publishing, 2004.