**ECE 7650 (Advance Matrix Algorithm)**

**PROJECT REPORT**

*On*

**“Adaptive Cross Approximation Algorithm”**

*By*

**Jamiu Babatunde Mojolagbe**

(Student ID: #7804719)

The Department of Electrical and Computer Engineering

University of Manitoba

*Submitted to*

**Ian Jeffrey, PhD**

(Course Instructor)

**CHAPTER ONE**

1. **Introduction**

The sole purpose of this project is to explore the generation of low-rank approximation, in form of an outer product approximants, of rank deficient matrices by an algorithm known as “Adaptive Cross Approximation (ACA)”. Perusal into merits, demerits as well as complexity of the algorithm in contrast with existing traditional method (Singular Value Decomposition) is also presented.

While as contained in the original paper proposed by Bebendorf et al in [1-3], the purpose of the algorithm is to obtain low-rank approximants for matrices obtained from solving integral equations – which are in fact, full rank, but contains rank deficient sub-blocks. However, for the sake of this presentation, the performance of this algorithm was based on randomly generated rank deficient matrices.

**1.1 Merit of ACA**

With ACA the memory requirement and CPU time requirement are less compare to the tradition truncated SVD. It has been shown that ACA resulted in O(nlogn) while it has been determined that the computational (time) complexity of SVD is said to be O((nm2 + mn2) according to [5]; where **n** is the number of rows and **m** is the number of columns. ACA requires O(r(m+n)) memory storage for a given matrix

One more beautiful thing about this algorithm is its rank revealing nature and as such can be dubbed rank-revealing LU decomposition [1] - without undergoing computationally demanding Modified Gram Schmidt process.

**1.2 Demerit of ACA**

While the advantages of ACA are something to write home about, it as well as its setbacks, though its merits may outweigh its disadvantages.

It has been observed that when a matrix is singular, ACA breaks down but for the traditional SVD still works fine.

**CHAPTER TWO**

1. **The Algorithm**

The mathematical formulations of this algorithm are left out from this presentation, however those formulations can be found in the original papers [1-3]. The focus, here, would be on how the algorithm works cum its implementation.

* 1. **Outline of ACA**

Consider a matrix of rank r which is r << min (m, n) or r < min(m, n). This matrix A can be represented as a low rank matrix as follow:

As the name of the algorithm suggests, ACA algorithm aims at approximating a given matrix to an approximant matrix ; however it does so through a product form that looks like the one presented above as follows:

where “r” is the effective rank of original matrices (for which approximation is shown above), and .

The goal of ACA is to achieve:

where “” is a given error tolerance and is the error matrix.

Therefore, instead of storing the entire entries of original matrix A, the algorithm only requires to store entries of the approximants.

* 1. **Implementation Steps of ACA**

1st iteration: input matrix A, error tolerance

1. Randomly choose starting row index such that
2. \*\*Set approximant matrix to zero
3. Set the first row of the error matrix, R such that
4. Obtain - the first column index – such that
5. Calculate first approximant vector
6. Set the first column of the error matrix, R such that
7. Calculate first approximant vector
8. Obtain approximate matrix
9. For the next iteration, obtain row index

such that and

For iteration, k = 2 to rank (r) of matrix A do

1. Update error matrix R (row),
2. Obtain - the kth column index – such that and
3. Calculate kth approximant vector
4. Update error matrix R ( column),
5. Calculate kth approximant vector
6. Obtain approximate matrix
7. If then the iteration has converged, so exit the iteration return
8. Else continue the iteration by obtaining row index for the next iteration, such that

that and

**Note:**  This algorithm as shown decomposes a matrix into sum of low-rank approximants and error matrix – which are not computed completely or explicitly as a matrix. Again the choice of error tolerance , is of utmost important in ACA in order to avoid undesirable numerical error in the approximation; as used in this project, the tolerance was set to be very small .

**2.3 Implementation Note**

Functions called “**aca.m**” and “**aca2.m**” were implemented in MATLAB as shown in section 2.2 above. The main differences between the two variants are the choice of row and column indices. While as shown in the algorithm presented in the previous page, the first starting row index is chosen randomly and column index is obtained by taking the index of absolute maximum of the chosen row. While for the consecutive row index I and column index J, similar procedure is followed. This is the way function “**aca.m**” was implemented. Whereas for “**aca2.m**”, it follows a different approach, in that the choice of first row and column indices are set to be fixed and are chosen to be 1 in each case. Such that the consecutive row or column index is obtained by incrementing the previous index.

The implementation files are well commented and the details of the input parameters as well as return values are well documented.

The driver program for this functions is named “**drive.m**”.

The input matrices supplied to these function were obtained from a function called “**createRankDefMatrix.m**”. The function returns rank deficient matrix depending on the parameters supplied to it – this function also is well documented and details of the inputs and the return values are contained in the implementation file.

**CHAPTER THREE**

**3.0 Results and Discussions**

**References**

[1] M. Bebendorf, “Approximation of boundary element matrices,” *Numer. Math.*, vol. 86, no. 4, pp. 565-589, Jun. 2000.

[2] S. Kurz, O. Rain, and S. Rjasanow, “The adaptive cross-approximation technique for the 3-D boundary element method,” *IEEE Trans. Magn.*, vol. 38, no. 2, pp. 421–424, Mar. 2002.

[3] M. Bebendorf and S. Rjasanow, “Adaptive low-rank approximation of collocation matrices,” *Computing*, vol. 70, no. 1, pp. 1–24, Mar. 2003.

[4] K. Zhao, M. N. Vouvakis and J. Lee, “The adaptive cross approximation algorithm for accelerated method of moments computations of EMC problems,” *IEEE Trans. Magn.*, vol. 47, no. 4, pp. 763–773, Nov. 2005.

[5] Z. Liu et al, “Using adaptive cross approximation for efficient calculation of monostatic scattering with multiple incident angles,” *ACES Journal*, vol. 26, no. 4, pp. 325–333, Apr. 2011.