Running SuperNEC on the 22 Processor IBM-SP2 at Southampton University

D C. Nitch, A P.C. Fourie and J S. Reeve

Jeff Reeve
Department of Electronics and Computer Science
University of Southampton
Southampton
SO17 1BJ, UK

Derek Nitch / Andre Fourie
Department of Electrical Engineering
University of the Witwatersrand
South Africa
email:nitch@odie.ee.wits.ac.za
email:fourie@odie.ee.wits.ac.za

ABSTRACT

SuperNEC (SNEC) is an object-oriented version of NEC-2 which has been modified to execute on a network of distributed memory processors. The matrix filling, solving and pattern computation routines are capable of running in parallel. A number of structures have been simulated using this code on the 22 processor IBM-SP2 machine at Southampton University. The principal problem studied was the DC-3 at 90 MHz. LU decomposition and an iterative matrix solution scheme were used in the study. The simulation time for this structure (which includes 3-D radiation patterns) dropped from 2.5 hours on a single processor to about 17 minutes when simulated on 12 processors using LU decomposition. Execution times are about half of these times when using the iterative solver. The far field patterns obtained from the simulation are compared with measured data and show good agreement. The largest problem tackled on the IBM machine was the DC-3 simulated at 160 MHz. This problem requires 17035 segments and was simulated in 5.3 hours on 21 processors.

INTRODUCTION

SuperNEC is an object-oriented electromagnetics code that implements the same theory as the FORTRAN program NEC2 [1]. The program is considerably more advanced that NEC2, as it incorporates features such as model based parameter estimation (MBPE), fast iterative solvers and a UTD-MOM hybrid capability amongst other features. One of its primary advantages is that SNEC has been written to operate in parallel on a network of processors.

The advantage of being able to operate in parallel is two fold. First, there is a potential reduction in the time taken to solve a problem. Secondly, a network of processors generally has more memory at its disposal than a single processor, thus much larger problems may be solved on a network of processors.

This paper reports on the performance of SNEC on the distributed memory, parallel IBM-SP2 machine located at Southampton University. The principal problem analysed is a DC-3 at 90 MHz. This structure is modelled using 5500 segments and generates an interaction matrix of 240 Mbytes. Run times for larger problems are also presented.

A Brief History of SNEC

In 1989 the FORTRAN program, NEC2, was modified to operate in parallel on a network of transputers [2-3]. During this project, all the numerically intensive routines were re-written to operate in parallel. The efficiencies achieved for the filling and factoring of the interaction matrix approached 90 percent, and even higher efficiencies were achieved for radiation pattern and near field calculations. As such, the runtime performance of the code was very satisfactory, however, the effort required to change the FORTRAN code was considerable. This was not necessarily due to the structure of the code, but rather due to the paradigm in which it was written. The use of common blocks (global variables) and implicit variable typing were two of the main features of the program which hindered the development and debugging of the parallel code. Making further modifications to the FORTRAN

1054-4887 © 1998 ACES

code was therefore predicted to be as difficult as adapting the code to run in parallel. One method of bypassing this problem was to rewrite NEC2 using a different programming paradigm [4]. The paradigm chosen for the implementation was the object-oriented paradigm, and the language of implementation, C++.

Converting FORTRAN to C++ is not merely a matter of translating from one language to the other. Rather, it requires the identification of the physical and mathematical entities present in the problem domain (segments, two-port networks, matrices etc.) and the assignment of responsibilities and attributes to each of these entities. For example, one of the elements identified in NEC2 was the concept of a wire segment. A segment was assigned the responsibilities of being able to move and rotate itself. The attributes assigned to a segment include the starting and end co-ordinates of the wire and the radius of the segment.

The primary aim of the object-oriented design was to make changing the electromagnetic parts of the code as easy as possible. For example, one obvious requirement is the ability to change the basis functions used in the MOM solution. This was partly achieved by introducing the concept of a Propagator class. The responsibilities of a Propagator are primarily to describe how an element in the structure propagates. That is, given some current distribution (known only to the Propagator and the Current class), a Propagator is able to compute the electric and magnetic fields at any point in space. Encapsulating the basis function in the Propagator class allows the entire code to be written independent of the function used to represent the current in a segment of the structure. Changing the basis function in SNEC requires only the writing of a new Propagator class and supplying the associated boundary conditions. Thus it is considerably easier to change the basis function in SNEC than in its FORTRAN counterpart.

The object-oriented version of NEC was completed in 1993. To illustrate the ease with which the new program could be modified, the sequential, object-oriented program was adapted to run on a network of transputers. This task took a total of two weeks to complete.

Transputer technology has severe limitations, the most overbearing being that a transputer machine is not an all-purpose machine and hence is not widely available (when compared to workstations). Thus restricting SNEC to a network of transputers was not a good long term prospect for the parallel code.

Currently there are a number of communications libraries that allow processors to communicate with one another over a local area network. One of these libraries is PVM (Parallel Virtual Machine). This library allows one to build up a parallel machine using a heterogeneous collection of existing processors. For example, one could connect a number of SUN 10's and RISC 6000 machines to form a fairly powerful parallel machine.

SNEC was freed from the limitations of the transputer by using PVM to implement the required communications. The parallel routines used in the PVM implementation are very similar to those used in the transputer version [2]. Subsequently, SNEC has been adapted to incorporate a hybrid MOM-UTD method, MBPE, fast iterative solvers and many other features.

The IBM SP2 machine at Southampton University.

The POWER parallel System (SP2) at Southampton University is a 22 processor distributed memory parallel machine. The first 16 nodes are called 'thin1' nodes, and have the following configuration :

Processor	66 MHz Power2
Memory	128 Mbytes
Memory Bus	64 bit
Instruction Cache	32 kbit
Data Cache	64 kbit
Disks	Two 2Gbyte SCSI disks

Table 1 : Specifications of nodes 1-16

Nodes 17 to 22 are 'thin2' nodes with the following configuration :

Processor	66 MHz Power2
Memory	256 Mbytes
Memory Bus	128 bit
Instruction Cache	32 kbit
Data Cache	64 kbit
Disks	Two 2Gbyte SCSI disks

Table 2 : Specifications of nodes 17-22

There is a single log-on node which is used primarily for testing, debugging and submitting jobs to the machine.

All processors are connected by "IBM high performance S2 adapters". This communications switch generates a maximum theoretical point-to-point bandwidth of 40 Mbits/s.

Communication libraries available on the SP2 include MPI (Message passing interface) and PVM.

Generating the SNEC model of the DC-3

SNEC requires that the structure to be modelled be specified in terms of straight wire segments. Generating such a model for the DC-3 is an exceptionally tedious task when tackled by hand. One of the tools developed at the University of the Witwatersrand for the conversion of complicated structures into wire segments is a package called SIG (Structure Interpolation and Gridding package) [7]. The input to SIG are curves that define the cross section of the structure at various points along its length. The cross-sections are defined in an ASCII file in a specified format. Given this information and the frequency at which the structure is to be modelled, SIG produces a valid wire element model suitable for input to either NEC2 or SNEC. The user defined cross-sections and the model generated by SIG for the DC-3 at 90 MHz are given in Figure 1 and Figure 2 respectively.

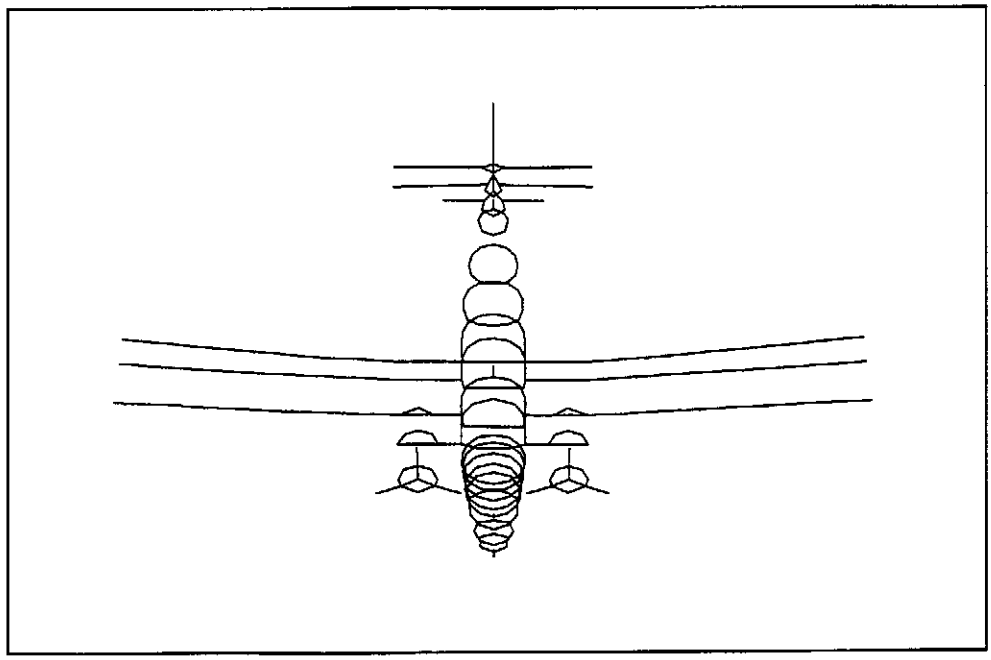

Figure 1 : The user defined cross-sections for the DC-3.

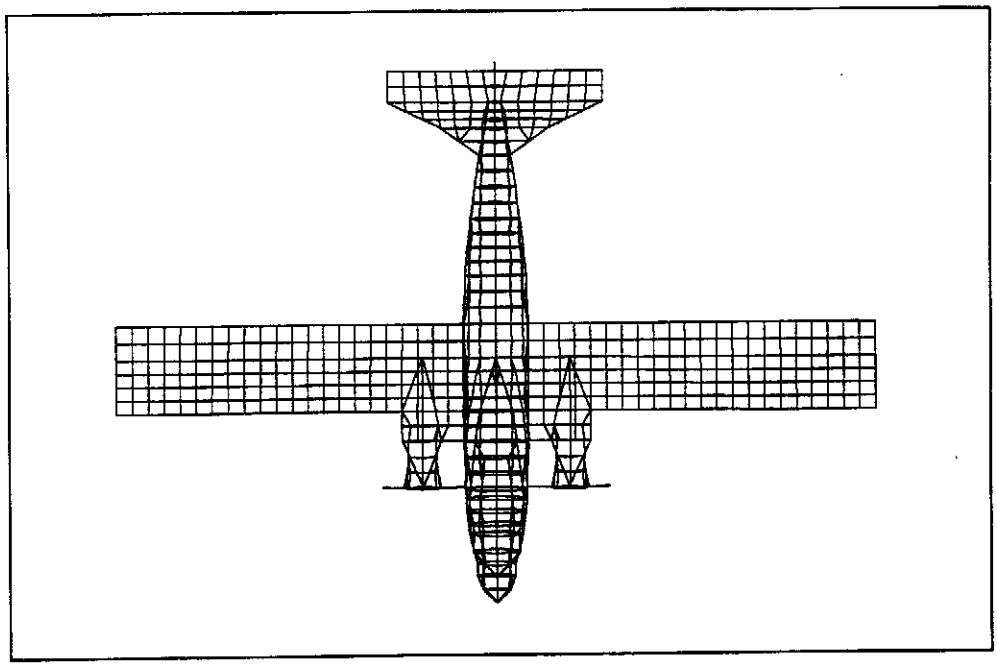

Figure 2 : The top view of the SIG generated model of the DC-3.

Results

The results presented in this section discuss the performance of SNEC on the SP2 and compare the far field patterns obtained from the simulation with measured values. These results will be discussed separately in the following sections.

Performance on the SP2

The simulation was performed on a network of 1 to 12 processors. The performance of the parallel code is assessed in terms of its speed up and execution times. The speed up of a parallel program is defined as :

$$\text{speedup} = \frac{\text{simulation time on one processor}}{\text{simulation time on p processors}}$$

The first stage in the MoM procedure is that of filling the interaction matrix. When executed in parallel, each processor is allocated an equal portion of the matrix to fill. Since filling the matrix does not require any communication between processors, it is expected that the speed up increases linearly with an increase in the number of processors. In actual fact, the speed up increases slightly faster than linear. The reason for this super-linear speedup could be attributed to the fact that when many processors are used, each processor has less memory to manipulate and this results in a more efficient use of cache memory.

The second stage of the MoM procedure is solving the matrix equation, for which SNEC has a number of different methods. In this study, the parallel performance of two of these techniques are investigated. The first technique involves factoring the matrix and then performing backward and forward substitution. This method is a parallel implementation of the matrix solving routines used in NEC2. The second technique is an iterative solver. The iterative solver is a two stage process that uses the Sparse Iterative Method (SIM) [5] for the initial iterations, and then, when the rate of convergence of this solver falls below a pre-set threshold, the bi-conjugate gradient stabilised algorithm is used to complete the iterative process. The SIM is similar to the banded matrix iteration (BMI) used in the GEMACS code. A comparison of the two methods is given in [8], whist the rationale behind the combination of the stationary and non-stationary techniques is detailed in [5].

Performance graphs for each major stage of the MoM solution have been generated.

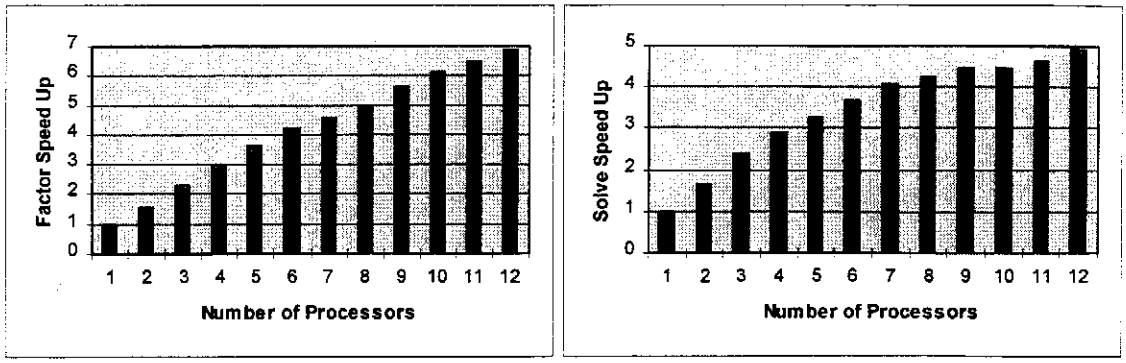

Figure 3 : The speed up of the LU Solver.

Figure 3 shows the speed up of the factoring and solving stages of the LU solver. The slow increase in speed up of this algorithm is attributed to the fact that there is not enough computation for each processor to amortise the communication overhead. Increasing the size of the problem on a 12 processor network will result in a more efficient use of the available processing power. The improvement in speed up could not be demonstrated in this study as the 5500 segment (240Mbyte) problem is the largest problem that could be simulated on a single processor with the memory available.

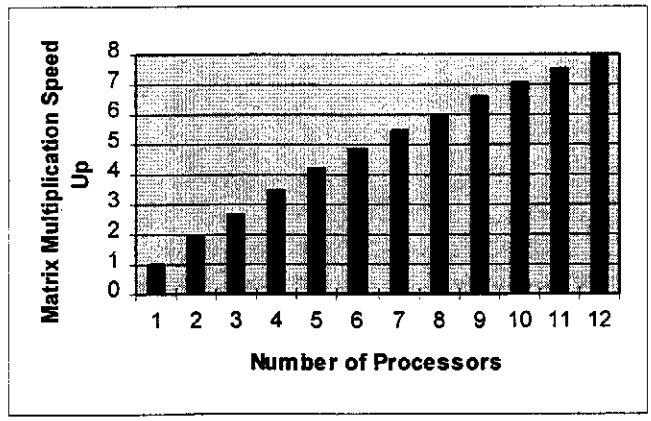

Figure 4 : Speed up attained by the matrix multiplication computation.

Most of the time used in the iterative solver is spent during a matrix-vector product operation. The speed up of the parallel matrix-vector product is given in Figure 4.

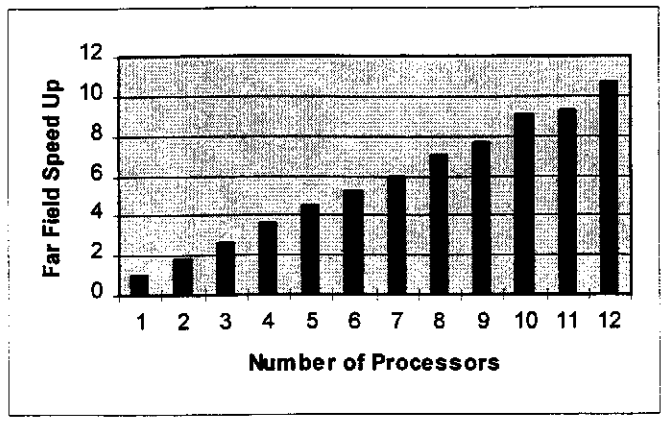

Figure 5 : The speed up of the far field computation.

In the simulation of the DC-3, four radiation patterns were computed. Three patterns were one-dimensional cuts taken in 2° increments. The fourth pattern was a three-dimension radiation pattern

sampled every 4°. The speed up of this computation is fairly consistent with the increase in the number of processor in the network and is displayed graphically in Figure 5.

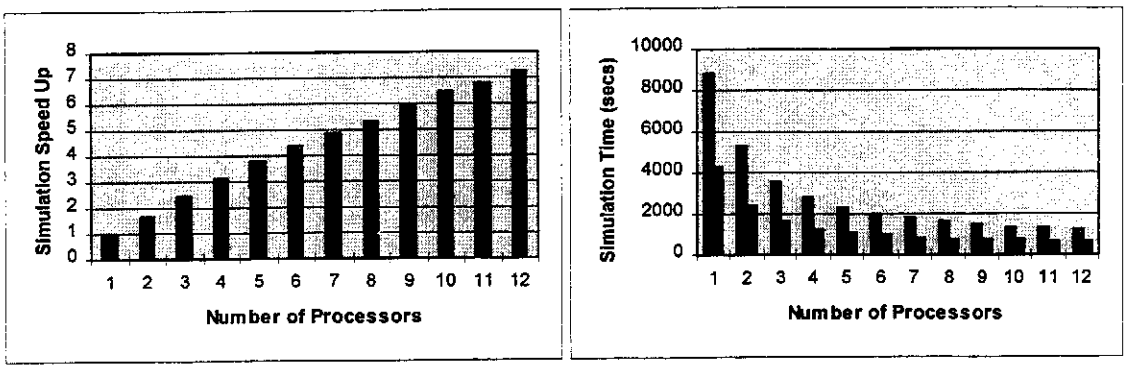

Figure 6 : The speed up of the simulation and run times on the SP2.

The first graph in Figure 6 shows the speed up of the entire simulation when using the LU solver. The execution times for the simulation are given in the second graph of this figure. The light-coloured bars in the second graph correspond to the run-times when using the LU solver, whilst the dark bars depict the run-times for the iterative solver. The iterative solver was deemed to have converged when the average tangential electric field error was less than 10^{-8} .

The DC-3 has been simulated on the IBM-SP2 machine at higher frequencies, the performance of the SP2 is depicted in Table 3.

Frequency (MHz)	Number of Segments	Number of Processors	Memory (Gbytes)	Memory per Processor (Mbytes)	Simulation Time (hours)
148	14608	16	1.7	106	3.7
160	17035	21	2.3	110	5.3

Table 3 : Run times for larger problems on the SP2 using the LU solver.

Comparison with measured data

The following figures show the theoretical and measured far fields for the DC-3 in the azimuth, pitch and roll planes. The measured patterns were performed using a 1 in 72 scale model and were obtained by Fourie, Givati, Clark and Pascoa [6].

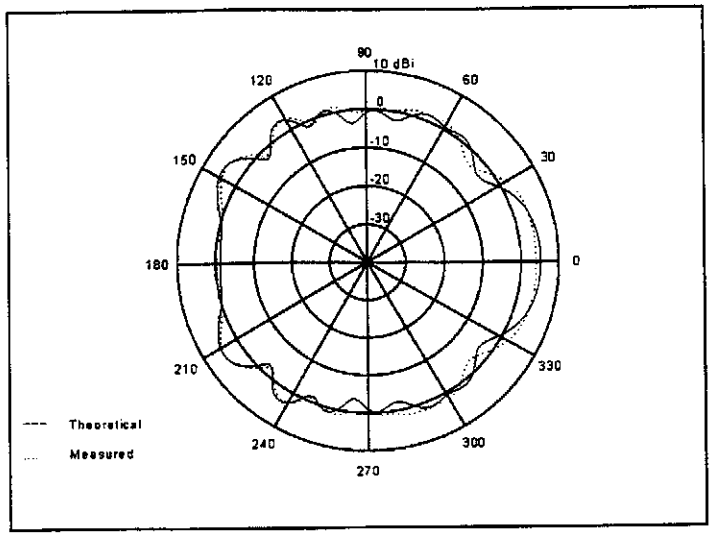

Figure 7 : The azimuth-plane radiation pattern of the top fin antenna at 90 MHz.

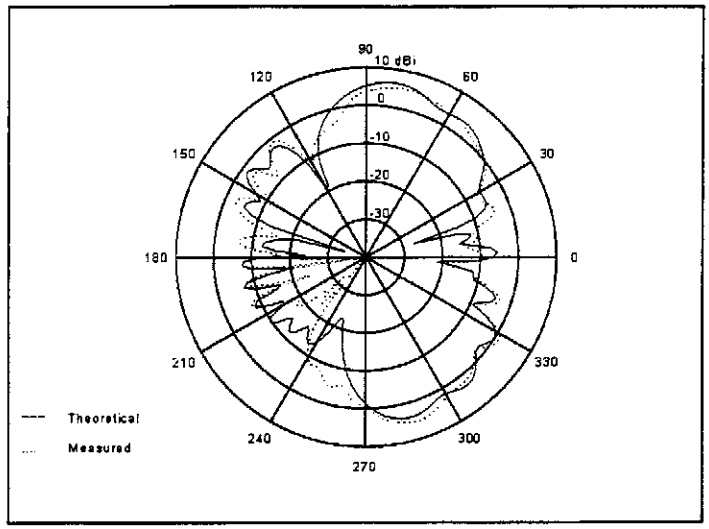

Figure 8 : The pitch roll radiation pattern of the top fin antenna at 90 MHz.

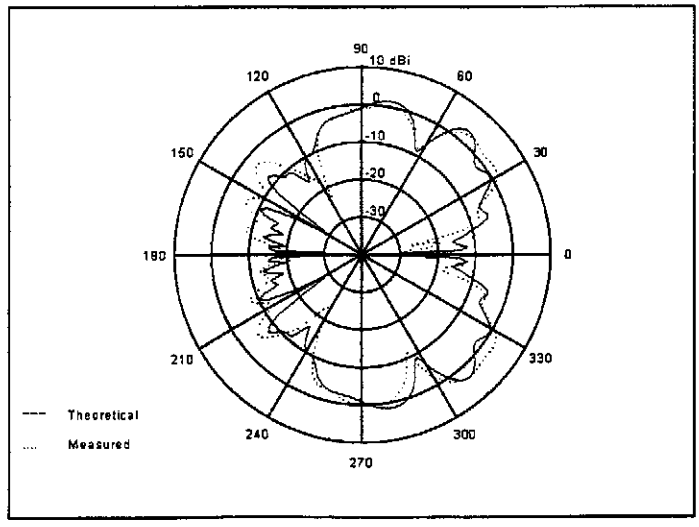

Figure 9 : The side roll radiation pattern of the top fin antenna at 90 MHz.

Conclusion

The parallel performance of SNEC on the IBM-SP2 distributed memory machine has been presented. The speed up attained by SNEC varied between 1.8 for a two processor network to 7.3 for a 12 processor network. The speed up of the larger networks will approach the number of processors for larger problems.

The largest problem reported is the simulation of the DC-3 at 160 MHz. Solving this problem on a sequential machine would require 2.3Gbytes of memory (RAM or hard-disk) and take about 4 days of computer time to simulate. Distributing the problem onto 21 processors, reduces the memory per processor requirement to 110 Mbytes and a solution was found in 5.3 hours. This demonstrates the effective use of distributed memory and processing power.

This study was done on a specialised parallel processing machine, however the software is not limited to execution on such machines. Workstations, linked by a local area network maybe used to form a reasonably powerful parallel processing machine.

References

[1]Burke G.J., Poggio A.J., "Numerical Electromagnetics Code (NEC2) - Method of

Moments," Naval Oceans Systems Center, San Diego, CA Tech Doc 116, 1981.
[2] D. C. Nitch and A. P. C. Fourie. "Adapting the numerical electromagnetics code to run in parallel on a network of transputers." Applied Computational Electromagnetics Society Journal, 5(2):76--86, 1990.
[3] D. C. Nitch and A. P. C. Fourie. "Parallel Implementation of the Numerical Electromagnetics Code", Applied Computational Electromagnetics Society Journal, Vol. 9, No. 1, March 1994, pp 51-57.
[4] D. C. Nitch and A. P. C. Fourie. "A Redesign of NEC2 Using the Object Oriented Paradigm", IEEE Antennas and Propagation Society International Symposium, Vol. 2, Seattle, June 1994, pp 1150-1153.
[5] D. C. Nitch and A. P. C. Fourie. "A Sparse Iterative Method (SIM) for Method of Moments Calculations", IEEE Antennas and Propagation Society International Symposium, Vol. 2, Seattle, June 1994, pp 1146-1149.
[6] A. P. C. Fourie, O. Givati, A.R. Clark and N. Pascoa. "Measured and Computer Results of Air-to-Ground communication performance of 2 aircraft at VHF/UHF frequencies", ANTEM-96 Symposium, Quebec, Canada, August 1996.
[7] A. P. C. Fourie, D. C. Nitch and O. Givati, "A Complex-Body Structure Interpolation and Gridding Program (SIG) for NEC", IEEE Antennas and Propagation Magazine, Vol. 36, No. 3 June 1994, pp 85-59.
[8] A. P. C. Fourie, D. C, Nitch "Comparing the Sparse Iterative Method (SIM) with the Banded Jacobi and Conjugate Gradient Techniques", IEEE Antennas and Propagation Society International Symposium, Vol. 2, Seattle, June 1994, pp 1181-1184.