JET ANALYSIS PROGRESS REPORT

H. Anastassiu
D. Ross
J. L. Volakis

June, 1995

33279-1-T = RL-2459

JET ANALYSIS PROGRESS REPORT

H. Anastassiu
D. Ross
J. L. Volakis

Wright Laboratories/AARA
Wright Patterson, AFB
OHIO 45433-7001

DEMACO
Suite 303
100 Trade Center Dr.
Champaign, IL 61820

June, 1995

University of Michigan
Progress Summary

During the past 4 months, our efforts were concentrated on the following:

- improve accuracy of mode matching code and recalculate RCS patterns for mesurement model
- parallelize mode matching and FEM codes on Intel Paragon
- integrate mode matching code with xpatch_ca
- write manual for mode matching code
- develop a future, more efficient code based on new concepts
- rewrite FEM code to exploit blade periodicity.
- exploitation of blade periodicity reduces CPU time for 50λ dia. engine from hundreds of hours to minutes.
- developed a simple approach for obtaining rotating-blade modulation data using solutions of stationary blades. This has major CPU implications since it cuts down CPU requirements by orders of magnitude.
- revert FEM code to using edge elements for more robust imposition of boundary conditions. Edge elements are better suited for modeling volume regions with large PEC sections.
- update FEM solver to using QMR which has smoother and faster convergence

The following documents are included in this report:

1. Interfacing with Xpatch_ca

   Currently, xpatch_ca can only make partial use of the modal scattering matrix. In this document, it is shown that more non-zero entries are required to obtain a satisfactory solution of the overall engine scattering. Since the non-zero entries of the scattering matrix are relatively very few and can be predicted apriori (on the basis of blade periodicity), an efficient I/O and matrix storage scheme is presented.

2. Parallelizaton of the Jet Engine Mode Matching Code

   To carry out realistic engine calculations, it is necessary to execute the code on a parallel platform. This section describes our approach for parallelizing the mode matching code on the Intel Paragon. We actually developed a generic approach for code parallelization which can be used for parallelizing future codes, including the new FEM jet engine code for discrete bodies of revolution. The transportable sections of the code are included and performance results are presented. It is demonstrated that the parallelized code has linear CPU requirements whereas the unparallelized code's CPU is at least quadratic.

3. Mode Matching Code Improvements and Manual

   A major revamping of the code was carried out as described in this document with the goal of speeding-up the code and improving its accuracy (previous comparisons with measured data were as much as 5 dB off in certain regions). The code was

restructured to take advantage of simplifications afforded due to blade periodicity and the curved blades mode normalization was corrected. Also, the Bessel functions integration routines were improved using an adaptive integration approach along with double precision. This resulted in substantial accuracy, convergence and stability improvements and is demonstrated by the excellent comparisons between measured and calculated results. The latter part of this document discusses the CPU and memory requirements of the code. A short code manual is also given in the Appendix.

4. Proposed Future Integral Equation Code

   To better exploit the periodicity of engine blades, two new integral equation approaches are proposed. One is based on the Adaptive Integral Equation(AIM) method which is a variation of k-space techniques and the other makes use of wavelets to generate a sparse system of equations. Regardless of the approach, the resulting system will have about 100,000 or so unknowns and this should be compared to the millions of unknowns associated with all-blade implementations. There are certain inherent risks with the proposed methodologies. Among them is the behavior of the singularity of the modal Green's function, the speed of AIM in accounting for the far zone element contributions and achievable sparsity of the wavelet-based matrices.

A report on the implementation and performance of the new FEM code will be provided at a later date and after completion of its validation.

MEMO: April 19, 1995

TO: Jet Engine Analysis Team

FROM: Dan Ross

RE: Interfacing Exact Methods to Xpatch

CC:

In order to make use of the exact methods (FEM, Mode Matching) in Xpatch, it is necessary to add more sophistication into the existing interfacing software.

Currently, Xpatch reads in a modal scattering matrix. The format of this modal scattering matrix was specified by McDonnell Douglas and allows for only block coupling between modes of the same order (n) and sense (even/odd).

We have used the case of straight blades, position 1, at 6 GHz to test if in fact non-block coupling is important. We compared the far fields using the full scattering matrix versus the far fields using scattering matrices with block coupling only.

In plots 1 and 2 (attached) note that when only block n coupling is assumed, the results are very different.

In plots 3 and 4 note that when all n coupling is used but only even to even and odd to odd modes are allowed to couple, the results are almost identical for phi-phi polarization but slightly different for theta- theta polarization. However, this is a special case. When the blades are rotated to position 2 (plots 5 and 6), both polarizations are very different, implying that all even to odd and odd to even coupling is required.

The only elements of the scattering matrix that can be assumed to be zero satisfy

$$n_{out} \neq n_{in} \pm kN_s \qquad k \text{ :any integer}$$

The format of the scattering matrix must be modified. We suggest a sparse storage format that is portable. The file should represent the elements of an 8 dimensional sparse system and would look like this:

1 1 0 1, 1 1 0 1 (-.247915198739783,-.12889095703847)
1 1 0 1, 1 1 0 2 (.3321378820867255,-.126351876946842)
1 1 0 1, 1 1 0 3 (-.237983092018485,-.139980415917646)
1 1 0 1, 1 1 0 4 (-.256793274432231,-.190063307218177)
1 1 0 1, 1 1 0 5 (8.760710687399284E-02,-.186127667977141)
1 1 0 1, 1 1 0 6 (.1942273229811178,-.107719258138243)
1 1 0 1, 1 1 1 1 (-3.897148288485881E-11,3.718494601411517E-12)
1 1 0 1, 1 1 1 2 (-1.233306264807076E-10,-1.626344477011836E-11)
1 1 0 1, 1 1 1 3 (-7.211708346235451E-12,-2.923477265083082E-12)
1 1 0 1, 1 1 1 4 (1.052208928527540E-10,9.984907603377483E-12)
1 1 0 1, 1 1 1 5 (8.674671097681725E-11,1.498112948143606E-11)
.
.
.

Each record in the file contains the fields

i1 i2 i3 i4, j1 j2 j3 j4 s

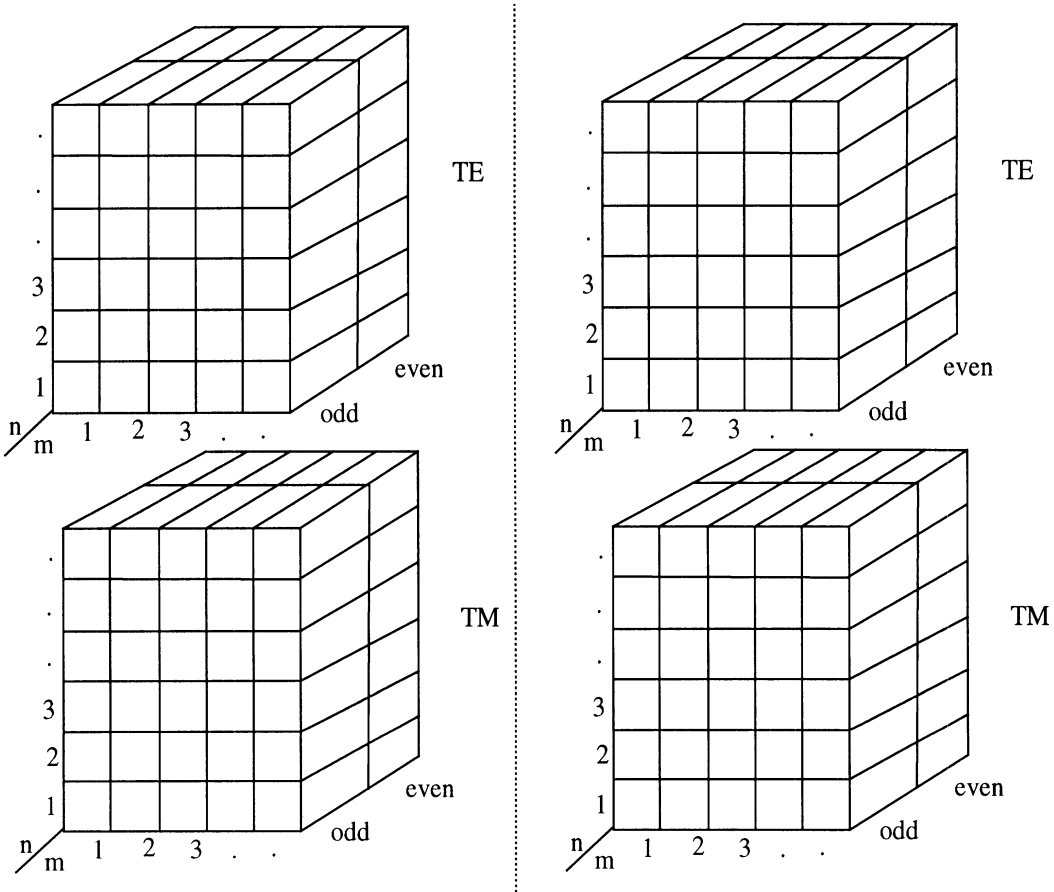
where:

i1: 0 TE,1 TM (Incoming mode)
i2: 0 odd, 1 even (Incoming mode)
i3: n (Incoming mode)
i4: m (Incoming mode)
j1: 0 TE,1 TM (Outgoing mode)
j2: 0 odd, 1 even (Outgoing mode)
j3: n (Outgoing mode)
j4: m (Outgoing mode)
s: scattering matrix entry

Any entry in the scattering matrix not present in the file is assumed to be zero.

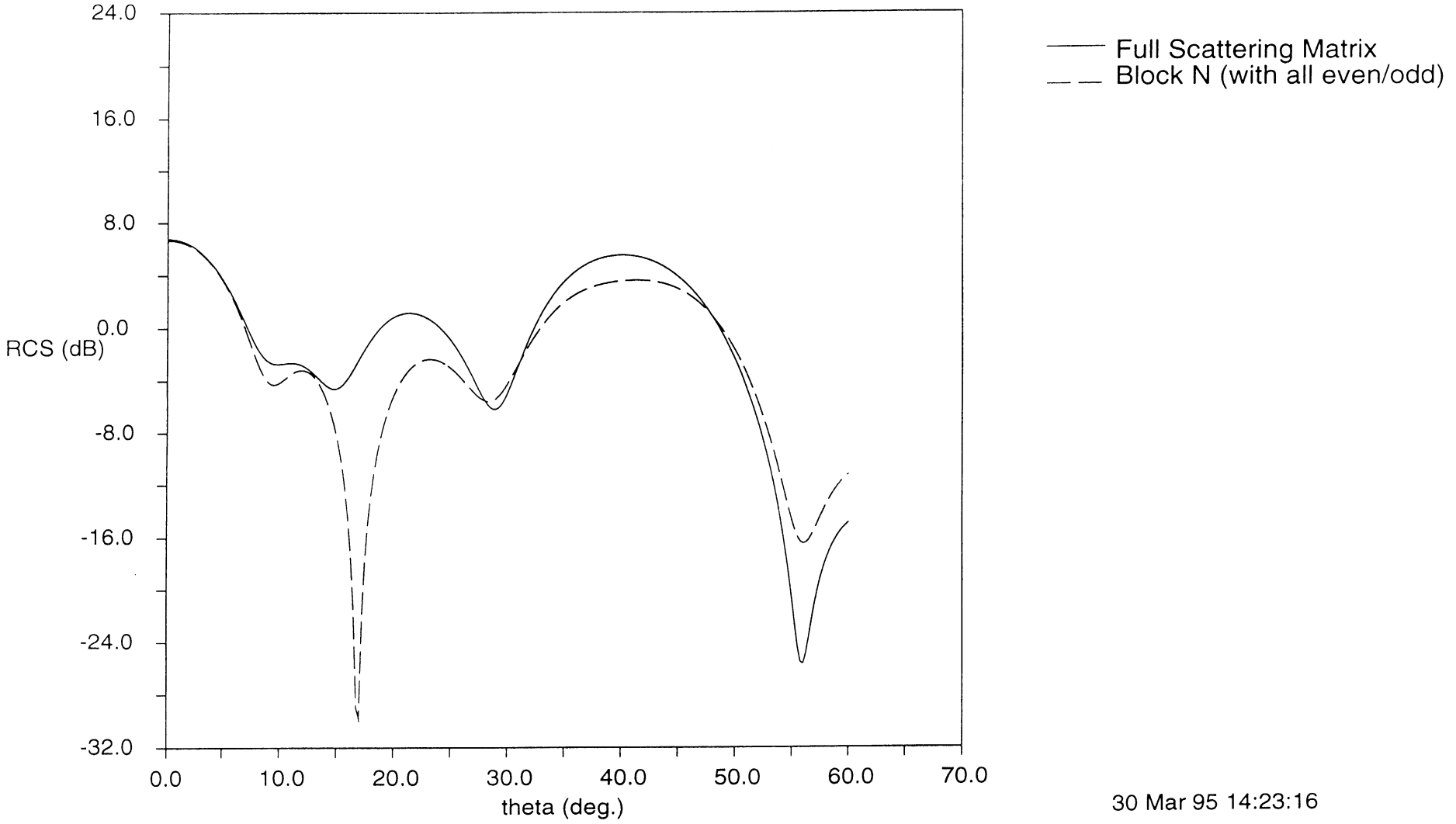
We suggest this format because it is efficient for storing a sparse scattering matrix and can be transferred to other codes without the need for any extra information.

The eight dimensional scattering matrix can be conceptualized using the following diagram:



Incoming Mode Index Outgoing Mode Index

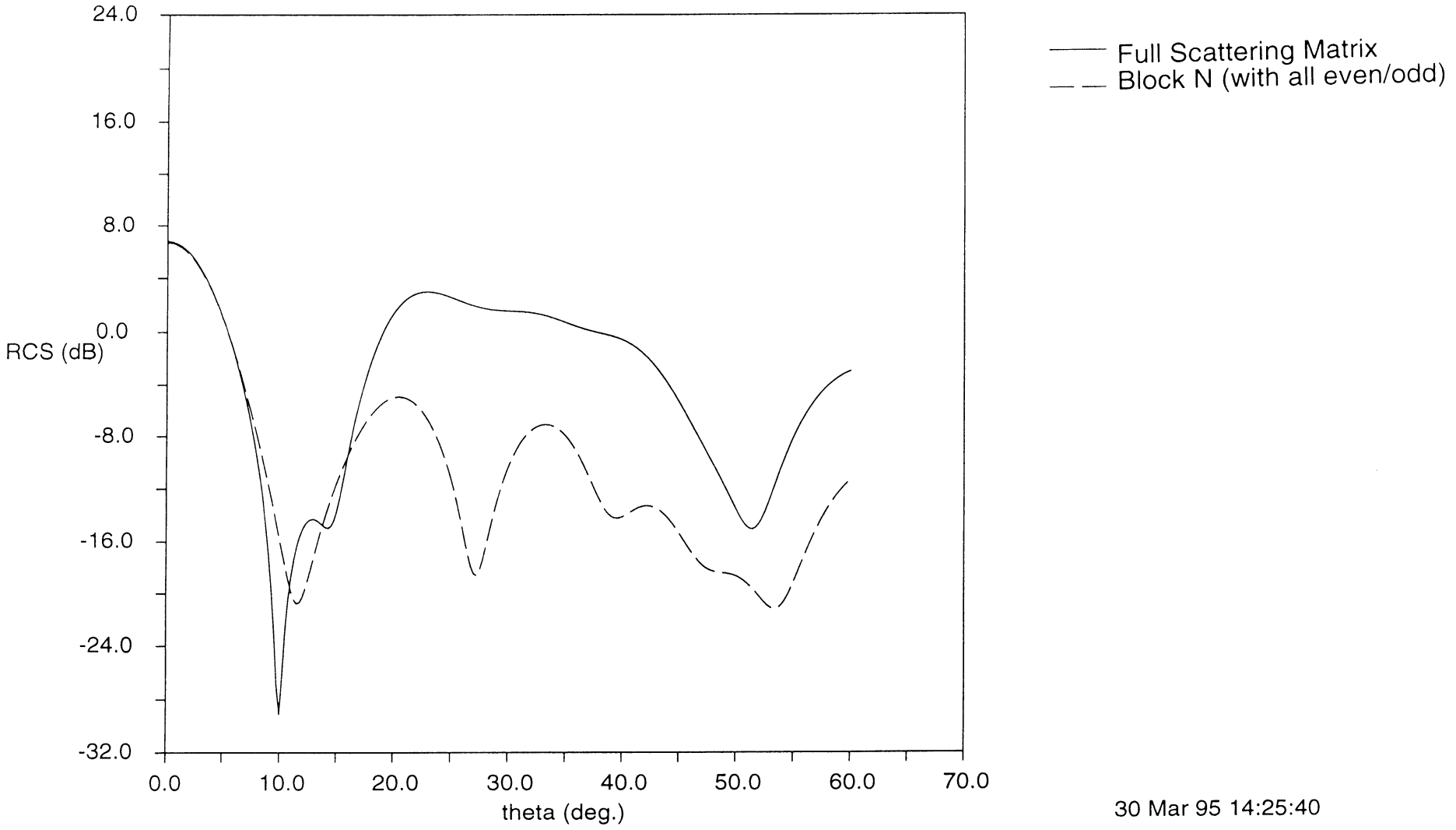
Mode Matching Solution
6 GHz, Straight blades, Position 1, $\phi\phi$



30 Mar 95 14:23:16

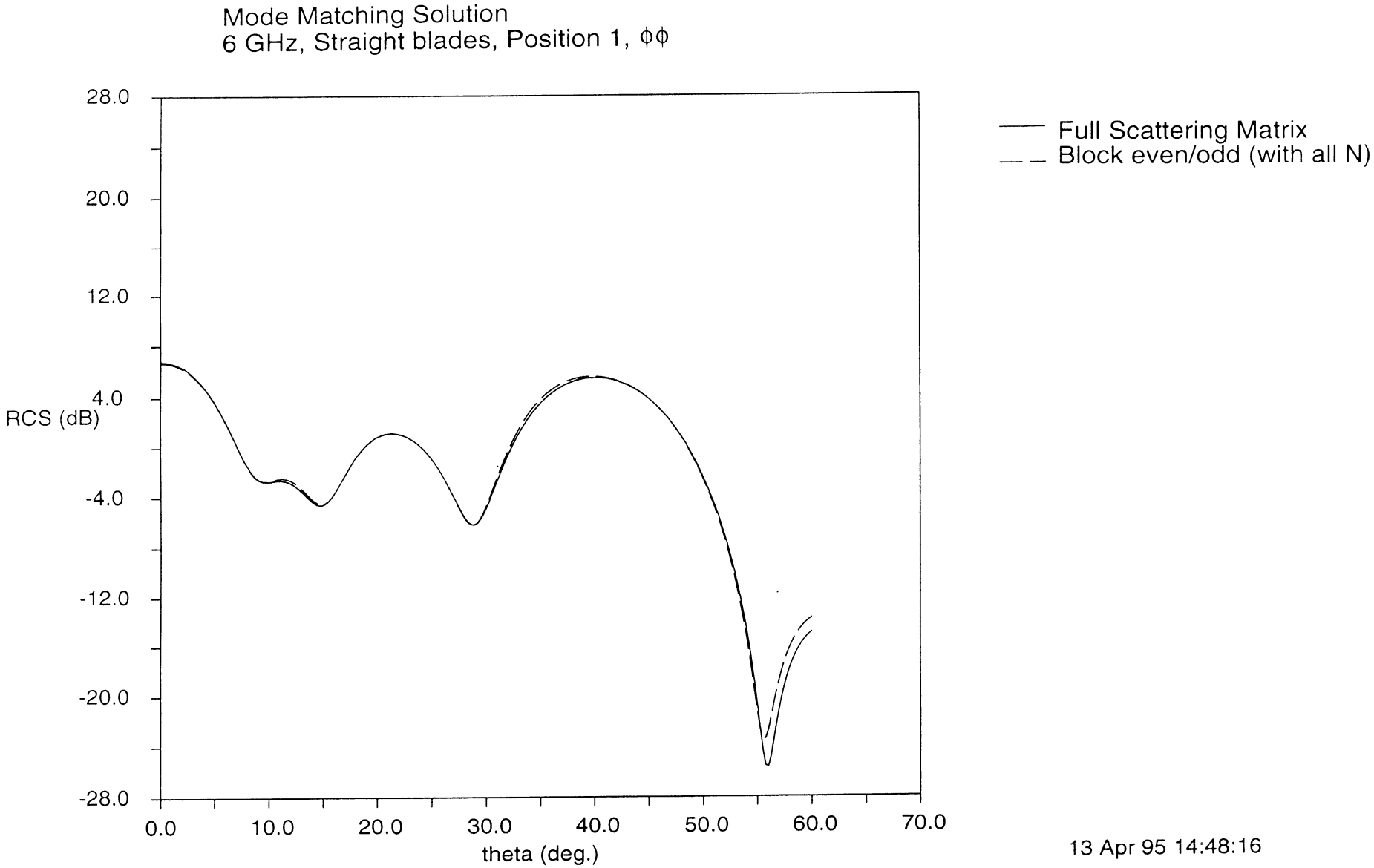
Plot 1

Mode Matching Solution
6 GHz, Straight blades, Position 1, θθ



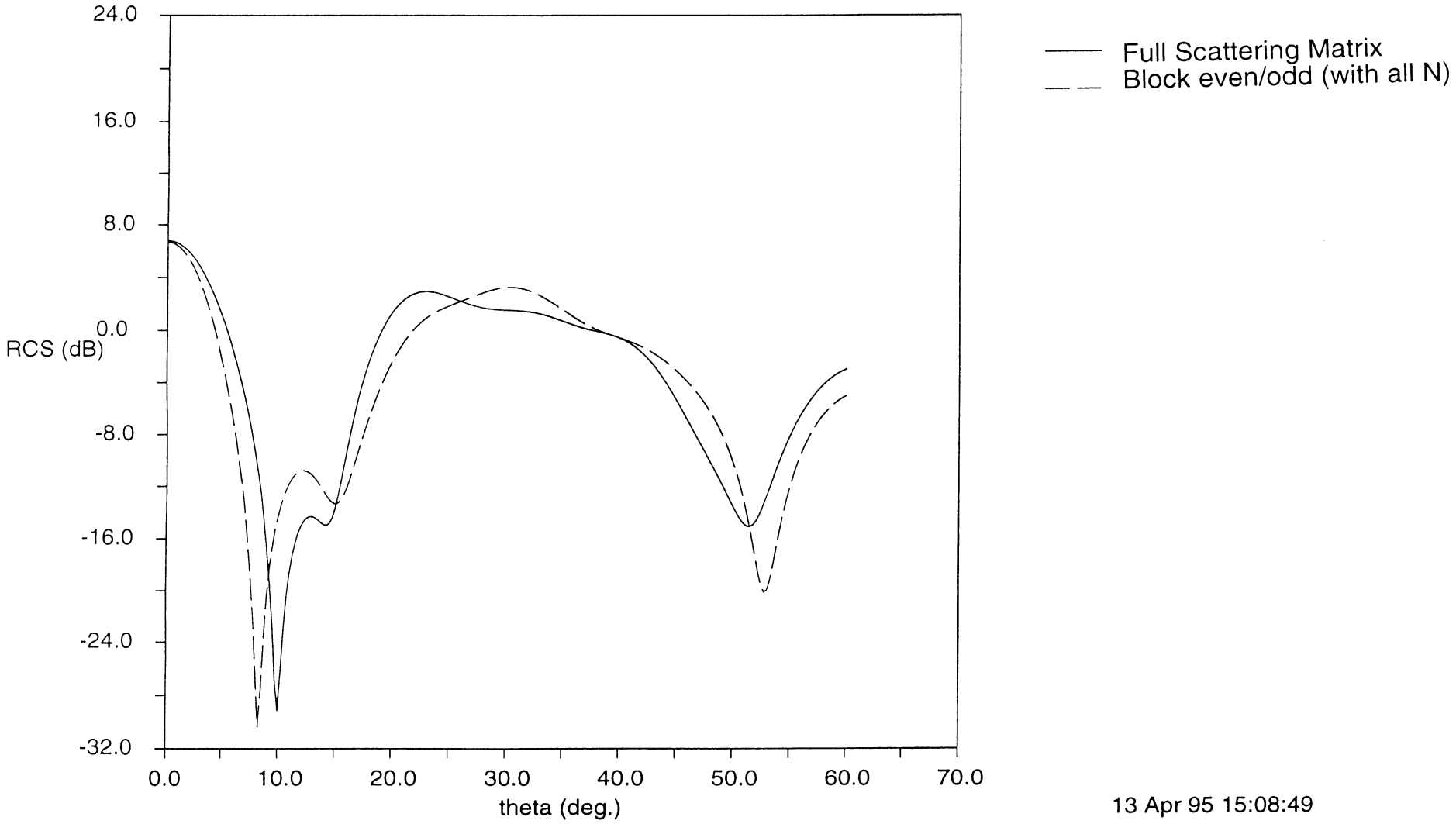
30 Mar 95 14:25:40

Plot 2


Mode Matching Solution
6 GHz, Straight blades, Position 1, φφ
28.0
20.0
12.0
4.0
-4.0
-12.0
-20.0
-28.0
RCS (dB)
0.0
10.0
20.0
30.0
40.0
50.0
60.0
70.0
theta (deg.)
Full Scattering Matrix
Block even/odd (with all N)
13 Apr 95 14:48:16

Plot 3

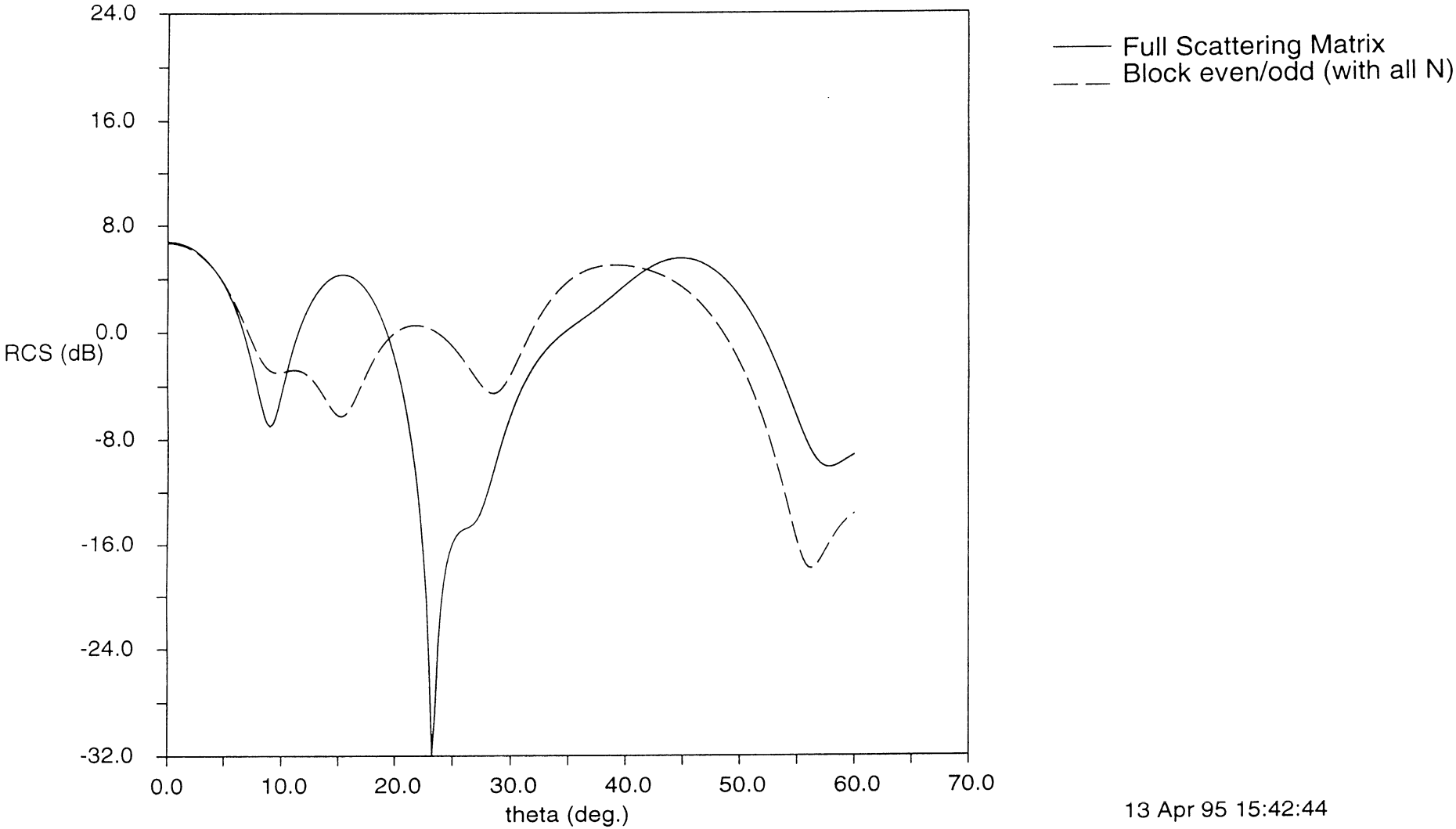
Mode Matching Solution
6 GHz, Straight blades, Position 1, θθ



13 Apr 95 15:08:49

Plot 4

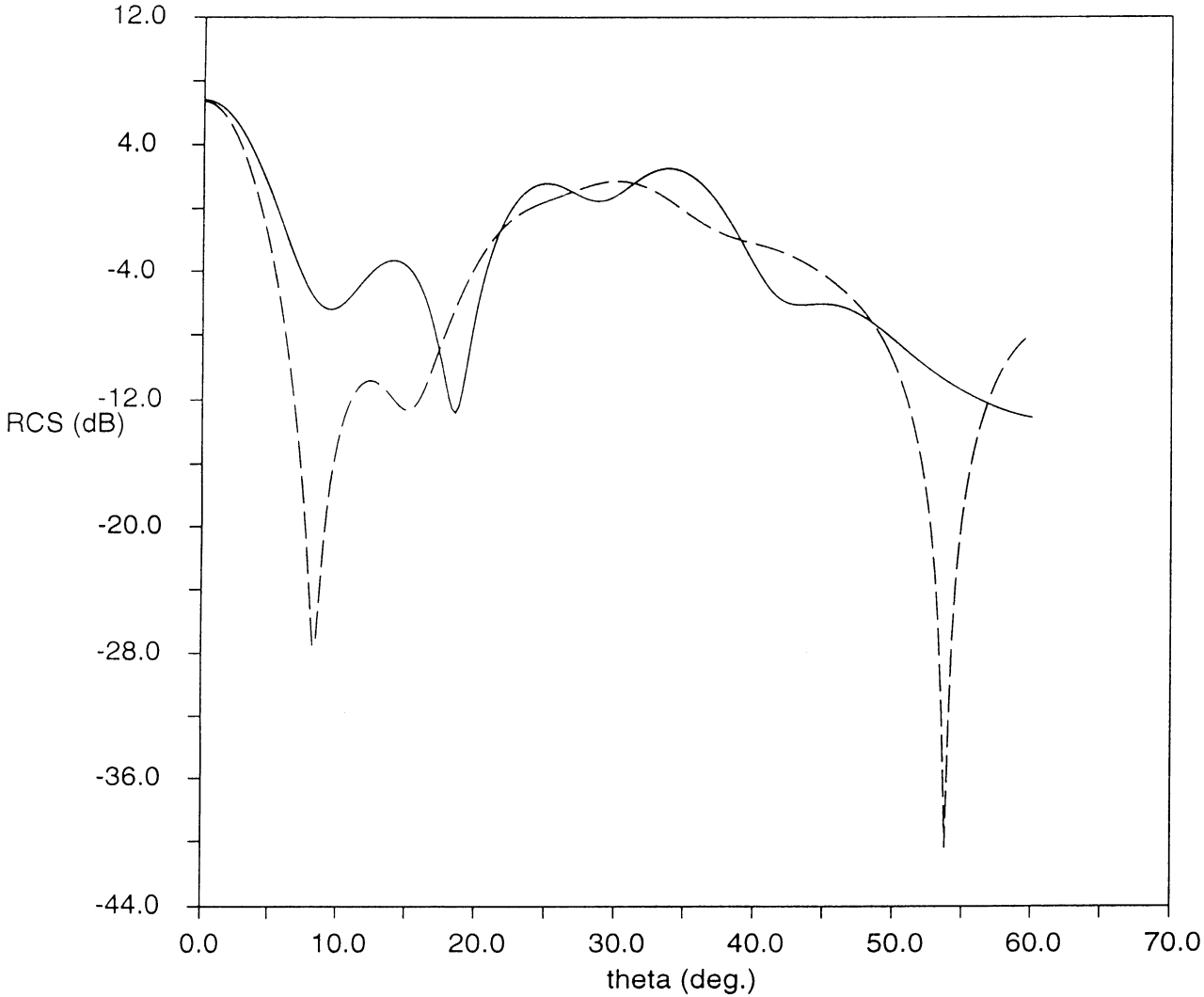
Mode Matching Solution
6 GHz, Straight blades, Position 2, $\phi\phi$



13 Apr 95 15:42:44

Plot 5

Mode Matching Solution
6 GHz, Straight blades, Position 2, θθ



Full Scattering Matrix
Block even/odd (with all N)

13 Apr 95 15:40:11

Plot 6

Parallelization of a Jet Engine Mode Matching Code

Daniel C. Ross, John L. Volakis and Hristos T. Anastassiu

Radiation Laboratory

Department of Electrical Engineering and Computer Science

University of Michigan

Ann Arbor MI 48109-2122

abstract -- Issues relating to the parallelization of a mode matching code for simulating radar scattering from jet engine-like structures are presented. The process of re-programming an existing serial code for parallel execution is explored from an engineering point of view where we wish to gain the most benefit from parallelization with the minimum amount of labor. We discuss how to profile the existing serial code to find where parallelization can be used to scale performance and how to choose a parallel model for optimum scaling and efficient use of resources. For the jet engine mode matching code, we achieve scalable performance on the Intel Paragon by computing the modal inner products and far fields in parallel using a manager/worker model. Our parallel scheme is coded in a reusable way so that other applications with similar profiles can be easily parallelized.

1.0 Introduction

An on-going effort aimed at the simulation of radar interactions with jet engines has led to the development of a semi-analytic, mode matching code [1]. This code is capable of predicting the radar scattering from a hollow, circular metal cylinder, terminated by a set of canonical, straight or curved blades. Since this code is the only known benchmark (other than measurements) for validating more flexible numerical techniques capable of modeling an actual jet engine, it will be of great value to optimize this mode matching code for scalable, parallel execution. The steps taken in transforming a working, debugged serial code to a parallel code will be discussed from an engineering point of view. Here we are concerned with gaining the largest performance increase with the minimum amount of programming labor. In this presentation, it will be assumed that the reader either already has an understanding of the mode matching technique used in [1] or simply doesn't care about the mode matching method and is concerned only with issues of parallelization.

A brief description of the serial mode matching code is given in the first section with a description of the physical input and output parameters. In the next section we describe the profiling procedure and a discussion on choosing the most appropriate level of parallelization. Since it is found that the modal inner products and the far field computations are the most appropriate points for parallelization, the next two sections focuses on the choice of a manager/worker parallelization model for the numerical integrations and for the matrix-vector multiplications needed for the far field calculations. Portable FORTRAN source code for the Intel Paragon is given, demonstrating the programming techniques used to incorporate parallelization into the original program. The performances of the two parallel schemes are evaluated independently and in then the performance of the overall parallelized application is compared with run times on high speed serial computers. In the last section we briefly describe other measures that could be taken to moderately increase the performance of the code over what has been presented here.

2.0 Code description

The jet engine mode matching code [1] predicts the radar scattering from an open ended cylinder terminated at the closed end by an inner circular hub and a set of canonical straight or curved veins (see Figure 1). The user is required to input geometric information which consists of the set of variables shown in Table 1.

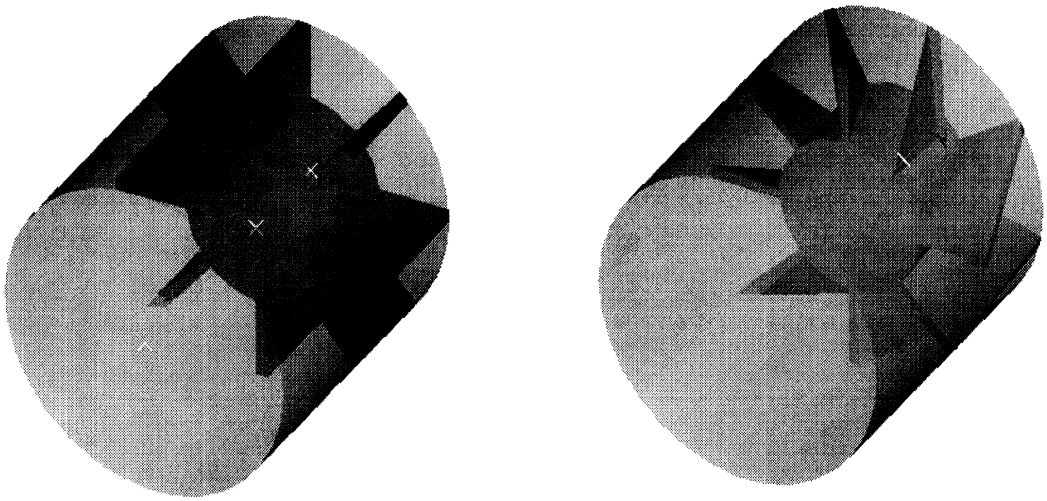


FIGURE 1. Geometry of canonical jet engine inlet. Left: straight blades, right: curved blades.

Also, the extent of the mathematical model is given by the user with the parameters shown in Table 2.

TABLE 1. Geometric model parameters

Variable	Description
a (λ)	Radius of inlet
b (λ)	Radius of inner hub
L1 (λ)	Length of inlet section
L2 (λ)	Length of engine section
N_b	Number of blades
gap ($^\circ$)	Angular span of air gap between blades
orient ($^\circ$)	Angular position of first blade
twist ($^\circ$)	Total twist angle of blades
RimFlag	Rim scattering contributions (on/off)
fGHz	Frequency (GHz)

TABLE 2. Mathematical model parameters

Parameter	Description
IN (Integer)	Max. mode index 1 for inlet region
II (Integer)	Max. mode index 2 for inlet region
JN (Integer)	Max. mode index 1 for engine region
JJ (Integer)	Max. mode index 2 for engine region
bgas (Real)	Error tolerance for adaptive integration
divmax	Max. number of integration subdivisions: 2^{divmax}

The code solves for the amplitudes and phases of the modes in both regions (inlet region and termination region) so that all boundary conditions are met. This is done directly (not

iteratively) and requires the evaluation of several matrices whose sizes are determined by the number of modes used. The semi-analytical solution is written in terms of the inverse and eigenvalue/eigenvector expansion of the matrices. Once the modal coefficients are known, the scattered field is computed using an analytical evaluation for the far fields of each mode radiating from the end of the open cylinder. The output of the code consists of backscatter patterns for both polarizations, computed along the principle arcs around the mouth of the inlet.

3.0 Code profile/Parallelization level

The engineering process of parallelizing an existing, debugged, serial code involves first determining if there are a few operations that require large fractions of the applications time and/or resources (memory). This is known as profiling the code. Profiling often gives unexpected results when performed on very modular code. It is often found that a low level routine is gobbling up a large fraction of the overall time and this routine has not been written by or even understood by the programmer. A simple change to the code which removes this modularity often causes a new low level routine to become the dominant consumer of computer time. Optimizing the serial code in this way will naturally separate the expensive and cheap operations until it is clear that one, or a few operations are the dominant user of time or memory. By focusing only on a few expensive operations, efficient code optimization can be done by only parallelizing some routines. The remainder of the operations in the code may represent a substantial amount of source but consume relatively little compute time during execution. Parallelization of the remaining portion of the code usually requires a complete redesign and may be unwarranted in practice.

Once the expensive operations are found, it is necessary to find if parallelization can actually be used to increase the performance of the expensive operations. Many algorithms are inherently serial as they require an iterative solution whereas others are inherently parallel. Even a serial algorithm however may benefit from parallelization if it is done at the correct level. For example, an iterative linear system solver can be parallelized at the matrix vector product level, thus scaling its performance even though it is inherently a serial algorithm. Ultimately though, the serial nature of any algorithm limits the scalability of the application on parallel computers. This is known as Ahmdal's law. Continuing with the iterative linear system solver example, Ahmdals law predicts that if the serial operations needed each iteration consume 5% of the overall time, while the

matrix vector products which can be parallelized consume 95% of the overall time, the maximum possible speedup would be a factor of 20 regardless of how many processors are used. In fact, adding too many processors will slow the application down as more time is spent doing communications. Therefore in practice, there will be an optimum number of processors with this number depending on the size of the problem and the ratio of parallel to serial operations.

For the mode matching code, the fourth round of profiling (once serial refinements were made) indicated that about 90 percent of the overall time (3.8 hours on an HP 9000/715) for a typical run (input parameters: a=4, b=2, L1=5.33, L2=2.66, N_b=8, gap=40.2, orient=2.4, twist=0, RimFlag=off, fGHz=8, IN=25, II=10, JN=7, JJ=8, bgas=.0001, divmax=10) was spent filling the matrices used in the semi-analytical expression for the mode coefficients. It was observed that this percentage will increase as the problem size (radar frequency) is increased. During the remaining 10 percent of the time, a matrix inversion is done, and matrix-matrix products are required along with the evaluation of the far fields, which requires a matrix-vector product. The inversion and the matrix-matrix product are needed during the evaluation of equation (31) in [1]. After the third round of profiling, it was determined that the large number of matrix-matrix products required in the evaluation of equation (31) in [1] would make the code impractical for terminations with many blades (more than twenty). By using the properties of overlapping modal and geometric symmetry as discussed in [3], we were able to reduce the summation in equation (31) in [1] down to one term for any number of blades. This is accomplished by introducing modes with exponential angular dependance rather than trigonometric angular dependance and noting that equation (32) is the same for each slice except for a constant phase factor which can be summed in closed form. Finally a matrix-vector product is required for each look angle. The matrix-matrix product and the matrix inversion require a great deal of nodal communications, and when considering the application as a whole, account for a smaller portion of the overall time while requiring more communications as the problem size is increased. Parallelization of the matrix fill operation and the far field calculations are therefore the only operations that will be modified. Both of these operations become more dominant as the problem size is increased. In fact, for a slightly electrically larger problem (input parameters: a=5, b=2.5, L1=6.67, L2=3.33, N_b=8, gap=40.0, orient=2.5, twist=0, RimFlag=off, fGHz=10, IN=30, II=12, JN=7, JJ=12, bgas=.0001, divmax=10) the total run time on the HP 9000/715 jumped up to 3 days with one full day required to calculate the far fields and all but a few hours spent calculating the modal inner products for the matrix fills.

For the matrix fill operations, the elements of the matrices are the inner products of different mode combinations over the interface from region one (inlet region) to region two (termination region) as given by equations (33) and (34) of [1]. The integrals require the numerical evaluation of Bessel function along the radial directions, while the integration in the angular direction is done analytically. It is the one dimensional numerical integration of different combinations of Bessel function products that consumes nearly all of the matrix fill time. The evaluations of higher order (order greater than one) Bessel functions are inherently serial since it is only possible to calculate higher order Bessel functions by first computing all lower order Bessel functions using a recurrence relations. A summation (series) evaluation of the higher order Bessel functions, which is inherently parallel is unfortunately numerically unstable [2]. Clearly, the level of parallelization must be above the Bessel function computations, allowing each node to do its own serial computations of its needed Bessel functions. And, by filling the matrix so that the highest order Bessel functions are needed first, the intermediate order Bessel functions can be saved and recalled as needed. (This is an example of serial refinement.)

Parallelization of the matrix fill can be done at either of two levels: (a) the matrix level or (b) the integration level. Using parallelization level (a) we have each processor working on a separate piece of the overall matrix while using (b) all processors contribute to the computations for each matrix element. In either case, the net matrix must be collected on one of the node's local memory (node 0) since other operations in the application are not parallelized. Thus, the total amount of communication will be the same for level (a) or (b) but parallelization at level (a) will require each node to store a portion of the matrix, which would then be copied back to the node 0. We therefore choose the lower level (b) since the overall memory demand will be a factor of two lower.

Parallelization of the far field computation will be done at the matrix-vector product level since only one matrix (the scattering matrix) is needed for all excitations. This operation can be made very efficient as will be seen in Section 5.0.

4.0 Parallel numerical integration

Numerical integration of the different combinations of Bessel function products is accomplished in an optimum way using three point Gaussian quadrature with the integration domain subdivided into 2^n subdomains where n is increased until the integral converges to some acceptable error (bgas in Table 2) or n becomes larger than some preset value (divmax in Table 2). Although the use of an adaptive integration scheme within a

production code is costly, a study of the number of subdivisions needed for convergence would represent an enormous task given the vast number of integrals that need to be evaluated. And, as the problem's electrical size is increased, higher order Bessel functions with more oscillations will be encountered, thus driving up the number of subdivisions needed for convergence. We will assume that in practice the number of processors will be smaller than the number of integration subdivisions needed and we therefore choose to break up the integrals separately for parallel computation. In fact, when calling the parallel integration routine within an adaptive integration loop, we will always choose the first n such that the number of subdivisions (2^n) for the first iteration is greater than or equal to the number of processors. As more processors are added then, we can increase the accuracy of the numerical integrations without increasing the overall run time. It has been found by experimentation that the accuracy of these integration has a substantial impact on the final solution and convergence of the method requires increasing accuracy as the electrical size is increased.

Since the parallelization will be done at the integration level, we choose a manager/worker model with domain decomposition on the integration interval. Processing node 0 will become the manager and nodes 0 through p will all have worker processes (one per node) in a constant state of either performing integrations or waiting. The amount of waiting time is minimized by not requiring communications between worker nodes directly. That is, the worker nodes do not need to be synchronized. The manager does not do anything except post requests that an integral be done and collect results. By making use of the Intel Paragon's global sum routine *gdsum* (see Intel Paragon User's Guide) the worker process on node 0 is given the task of collecting the results from all other workers by moving each worker's value through the Intel Paragon's mesh connected processors in an efficient way. The sum is then passed to the manager process which is also on node 0. The special message passing hardware within the Intel Paragon allows this global sum to be performed nearly as efficiently as a systolic algorithm which would only let each node communicate with its neighbors. Natural load balancing occurs since the integration domain is divided up evenly across all worker processes. Even though the global sum causes the worker processes to block until the sum is complete, the load balanced worker processes are completed at virtually the same time thus minimizing any waiting periods. This fact is born out by observing that this scheme does indeed scale almost as 1/p where p is the number of processing nodes (as will be shown later in this section).

With the following FORTRAN source listings, we show how the integration scheme is implemented in a portable fashion. Any number of subdivisions can be requested, not only

powers of two. There are two independent programs (one manager and many workers) that communicate using the Intel Paragon's message passing system calls. The worker processes execute the following program (intwork.f):

```
      include 'fnx.h'                                  Intel supplied header file

      Integer msg_data,msg_answer                      Message passing data
      Parameter (msg_data=2,msg_answer=3)              structures
      Real*8 x1c,x2c,integral,tmp,xgauss(3),
     &     wgauss(3),width,c
      Real*8 delx,x1,x2,sum,integral
      Common /dmsg/ x1c,x2c,c,xdivs,fnum               Common blocks used to
      Common /amsg/ integral                           pass messages with
                                                       different data types.

      Real*8 Fn1,Fn2                                   Integrand functions
      Integer xdivs,fnum                               Local variables
      Integer divs,nodes,dpn,i,p,drem
      Data wgauss /0.2777777778,0.4444444444,          Constants for 3 pt.
     &    0.2777777778/                                Gaussian Quadrature
      Data xgauss /0.1127016654,0.5000000000,
     &    0.8872983346/

1     Call crecv(msg_data,x1c,32)                      Wait for message to
                                                       integrate

      If (x1c.EQ.0 .AND. x2c.EQ.0 .AND.                Check if terminated
     &         xdivs.EQ.0) Goto 2

      divs=xdivs                                       Set local variables
      nodes=numnodes()
      delx=( x2c - x1c )/divs

      If (divs.GT.nodes) Then                          Find my portion of integral
          dpn=divs/nodes
          drem= IMod(divs,nodes)
          If ( mynode() .LT. drem) Then
              dpn=dpn+1
              x1=x1c+( mynode()*dpn*delx)
              x2=x1+(dpn*delx)
          Else
              x1=x1c+( drem*(dpn+1)*delx) +
     &               (mynode()-drem)*dpn*delx
              x2=x1+(dpn*delx)
          End If
      Else
          If ( mynode().LT.divs ) Then
              dpn=1
          Else
              dpn=0
          End If
          x1=x1c+( mynode()*dpn*delx)
          x2=x1+(dpn*delx)
```

```
        End If
        sum=0                                          Do my portion of integral
        Do p=1,3
          Do i=0,dpn-1
            Goto (11,12) fnum                          Use integrand given by fnum
11          sum = sum + wgauss(P)
     &    * Fn1(  ( xgauss(P) + I) * delx + x1)        Integrate function 1
            Goto 3
12          sum = sum + wgauss(P)
     &     * Fn2( ( xgauss(P) + I ) * delx + x1,c)     Integrate function 2
            Goto 3

3         Enddo
        Enddo
        integral = sum * delx
        Call gdsum(integral,1,tmp)                     Perform global sum
        If (mynode().EQ.0) Then
            Call csend(msg_answer,integral,8,0,0)      Send result to manager
        End if
        Goto 1
2       End

        Real*8 Function Fn1(x)                         Integrand function 1
        real*8 x
        Fn1=4./(1+x**2.)
        Return
        End

        Real*8 Function Fn2(x,c)                       Integrand function 2
        real*8 x,c
        Fn2=Cos(c*x)
        Return
        End
```

Every node executes the worker program and enters an infinite loop, waiting for a message to integrate a function or to terminate. The original serial integration routine is replaced by the integration manager subroutine and the integrand functions are removed from the serial code and put into the worker programs as Fn1, Fn2, Fn3 etc. When an integration of one of these functions is needed, a high level call to the manager subroutine is made indicating the function number (fnum) to integrate. Each worker has its own copy of the functions. Any integration constants are passed along with the message block as is shown above for integrand function 2. Note that the integrand functions can be made to be very "smart" since they have access to all of the nodes local memory for storing previously computed values.

The following source code shows how a high level call is made from the original serial code (serial.f) to the parallel integration manager routine:

```
      include 'fnx.h'                          Intel supplied header file
      .
      .                                        Existing serial code
      .
       Real*8 x1,x2,answer,dparint,c           Additional data structures needed
       Real*8 prev
       Integer xdivs,n,divmax
       Integer fnum
       Common /dmsg/ x1,x2,c,xdivs,fnum
       Real*8 re2,re1,crit,bgas,bound
       parameter (bgas=.0001)                  Adaptive integration parameters
       parameter (divmax=11
       .
       .
       .
5      n=1                                     Start with the number of subdivisions
       Do 6 While (2**n.LT.numnodes())         greater than or equal to number of
           n=n+1                               nodes
6      Continue

       bound=bgas                              Adaptive integration of function 1
       x1=0.
       x2=1.
       xdivs=2**n
       fnum=1
       re1=dparint()                           Call to parallel integration routine

       n=n+1
10     xdivs=2**n                              Iterate until error tolerance is
       re2=dparint()                           achieved
       if (re1.eq.0.0d0) then                  Calculate error
           crit=dabs(re1-re2)
       else
           crit=dabs((re1-re2)/re1)
       end if
       if (crit.gt.bound) then                 Check if converged
           n=n+1
           if (co.gt.divmax) then              Integral did not converge
               write(6,*) 'convergence '//     Warn the user of problem
     & 'problems for function ',fnum
               bound=10.0d0*bound              Lower requirements
               goto 5
           end if
           re1=re2
           goto 10                             Try again
       else
           answer=re2                          Got answer
       end if
       .
       .
       .
       return
       end
```

```
      x1=0                                  Do another integral (non-adaptively)
      x2=1
      xdivs=1000
      fnum=2                                Use integrand function 2
      c=100                                 Pass along an integration constant
      answer=dparint()
      .
      .
      .                                     Kill worker processes before exiting
      x1=0
      x2=0
      xdivs=0
      answer=dparint()

      End


      Real*8 Function dparint()             Parallel integration manager

      Include 'fnx.h'                       Intel supplied header file

      Integer msg_data,msg_answer           Message passing data structures
      Parameter (msg_data=2,msg_answer=3)
      Real*8 x1,x2,answer,integral,tmp,c
      Integer xdivs,fnum
      Common /dmsg/ x1,x2,c,xdivs,fnum
      Common /amsg/answer


      If (xdivs.LT.1) Then
          If ( x1.EQ.0 .AND. x2.EQ.0 .AND.
     &         xdivs.EQ.0) Then
              Call csend(msg_data,x1,32,-1,1)Terminate workers
              Call csend(msg_data,x1,32,0,1)
              Goto 1
          End If
          Write (6,*) '***dparint ERROR: '//
     &         'Must have at least one subdivision.'
          Stop
      End If

      Call csend(msg_data,x1,32,-1,1)       Send message to all other nodes
      Call csend(msg_data,x1,32,0,1)        Send message to worker on node 0
                                            also
      Call crecv(msg_answer,answer,8)       Wait for answer

      dparint=answer                        Send answer back to serial code

1     End
```

Note that the high level call to dparint() can be treated as a normal function call except that the run time of this routine will scale with the number of processors used. The two programs (original serial code with the modifications shown above, and the worker code) are compiled separately. The worker program (intwork) is executed on all available nodes while the original serial code (serial) is executed on node 0 only. This is accomplished (using 120 nodes for example) with the Paragon OSF command:

```
serial -sz 120 -on 0 -pt 0 \; intwork -pt 1
```

The -sz modifier indicates the total number of processors to use. The -on 0 flag indicates that the program serial is to run only on node 0 and the -pt flag gives the manager process an identification number of 0 and the worker processes an identification number of 1. This is necessary for the message passing scheme we have used.

4.1 Performance

The performance of the portable, parallel integration scheme presented above was measured by computing the value of π using

$$\pi = \int_0^1 \left(\frac{4}{\sqrt{1+x^2}} \right) dx$$

We compute the integral 1000 times, with an increasing number of subdivisions, evenly spaced, starting with 10 and ending with 10,000. As shown in Figure 2, the run times of the portable integration routine scale nearly as 1/p where p is the number of processors.

It is crucial that the numerical integrations be done using double precision. A single precision version of the integration code is subject to substantial inaccuracies as the number of subdivisions is increased. Also, due to roundoff errors, a single precision code will give different results depending on the number of processors used. It is therefore crucial that all of the calculations be done using double precision.

5.0 Parallel far field computation

Once the modal inner products have been computed, the computation of the modal scattering matrix is performed by evaluating equation (54) of [1]. The portion of the mode matching code that evaluates equation (54) of [1] is not parallelized in this presentation. However, as mentioned earlier, we have reduced the summation in equation (54) down to one term by introducing modes with exponential angular dependance thus making the cost

of this operation scale by the number of blades. Once the scattering matrix is obtained, the far fields are found by evaluating a vector of modal coefficients of the incoming modes for a given incidence angle and polarization and then premultiplying by the scattering matrix. Each of these matrix vector multiplies (one for each incidence angle and polarization) can make use of all of the available processing nodes in an efficient manner since the scattering matrix does not change.

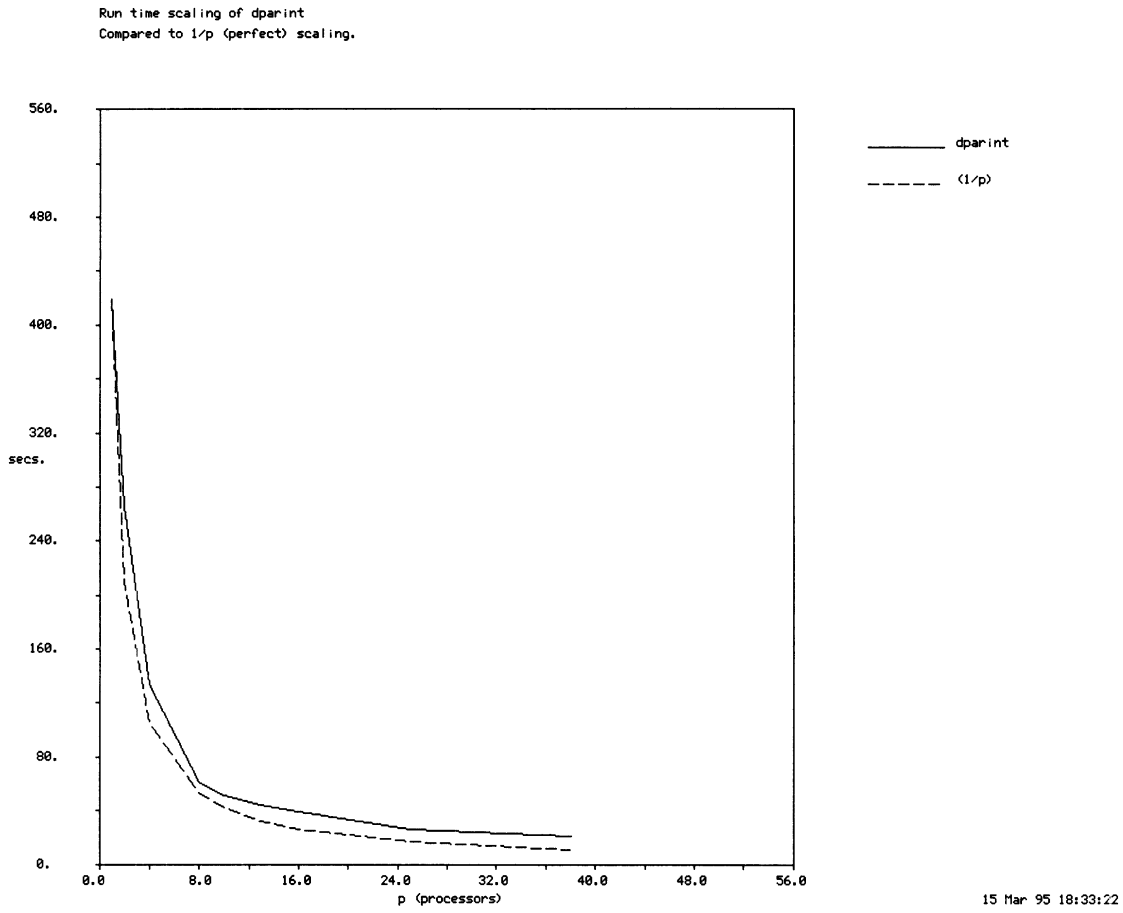


FIGURE 2. Run time scaling of portable integration routine dparint.

The matrix-vector product needed for this operation can be coded using the same techniques that we have used for the parallel integration routine. The manager routine breaks the scattering matrix into nearly equal sized pieces by dividing up the rows and passing them to each processor. That is, each worker gets some of the rows of the matrix. Then, the manager passes all of the vector to each worker and the workers perform their portion of the product. A global concatenation of each node's result is accomplished using the Intel supplied routine *gcolx* (see Intel Paragon User's Guide).

5.1 Performance

The matrix-vector product coded using the scheme described above scales very effeciently as more processors are added. However, there is an optimum number of processors for which the run time will be a minimum. Adding more processors than the optimum will tend to decrease the performance. Using a system of size 1440 by 1440 and calculating 482 matrix-vector products (which correspond to the far field calculations needed for the 10 GHz case mentioned previously) we show the run times as the number of processors is increased in Figure 3. These run times include the time required to distribute the scattering matrix out to each of the nodes. Note that about 64 processors is optimum for this case. It is the increased communications needed when more processors are used that eventually washes out the gains from performing the floating point operations in parallel.

It is important to know the optimum number of nodes ($P_{optimum}$) before running the program. We have therefore arrived at the following empirical relation indicating the optimum number of processing nodes to use for a given system size (N).

$$P_{optimum} = Int(\frac{N}{22.5})$$

Note that the system size for the scattering matrix will be $N = 4\,(\mathrm{IN} \cdot \mathrm{II})$.

Since the parallel integration routine requires relatively little communication time, it does not display the same kind of performance peak for a given number of processors, at least it will not until far more processors are used. Therefore, we let the parallel integration worker processes run concurrently on the nodes with the matrix-vector product processes and also on all other available nodes. The process identifiers must be different for the integration and matrix-vector product workers in order to keep their respective messages separate.

6.0 Performance of the parallelized mode matching code

With the three modifications to the serial mode matching code:

- Parallelization of modal inner products.
- Use of overlapping modal and geometric symmetry [3] to reduce equation (54) of [1] down to a single term rather than a sum of matrix products.
- Parallelization of the far field computations

the complete jet engine mode matching code was run on the Intel Paragon using the optimum number of processors for the matrix-vector products and all available nodes for the modal inner products. The middle, serial portion of the code was actually run on a high performance serial computer (HP 9000/750). Overall run times for the application are given in Figure 4 for increasing frequency. The model parameters for each frequency are given in Table 3. The overall runtimes include the time required to dump out intermediate data (modal inner products) to the HP 9000/750, compute and dump out the scattering matrix, sending it to the Paragon and finally computing the far fields in parallel.

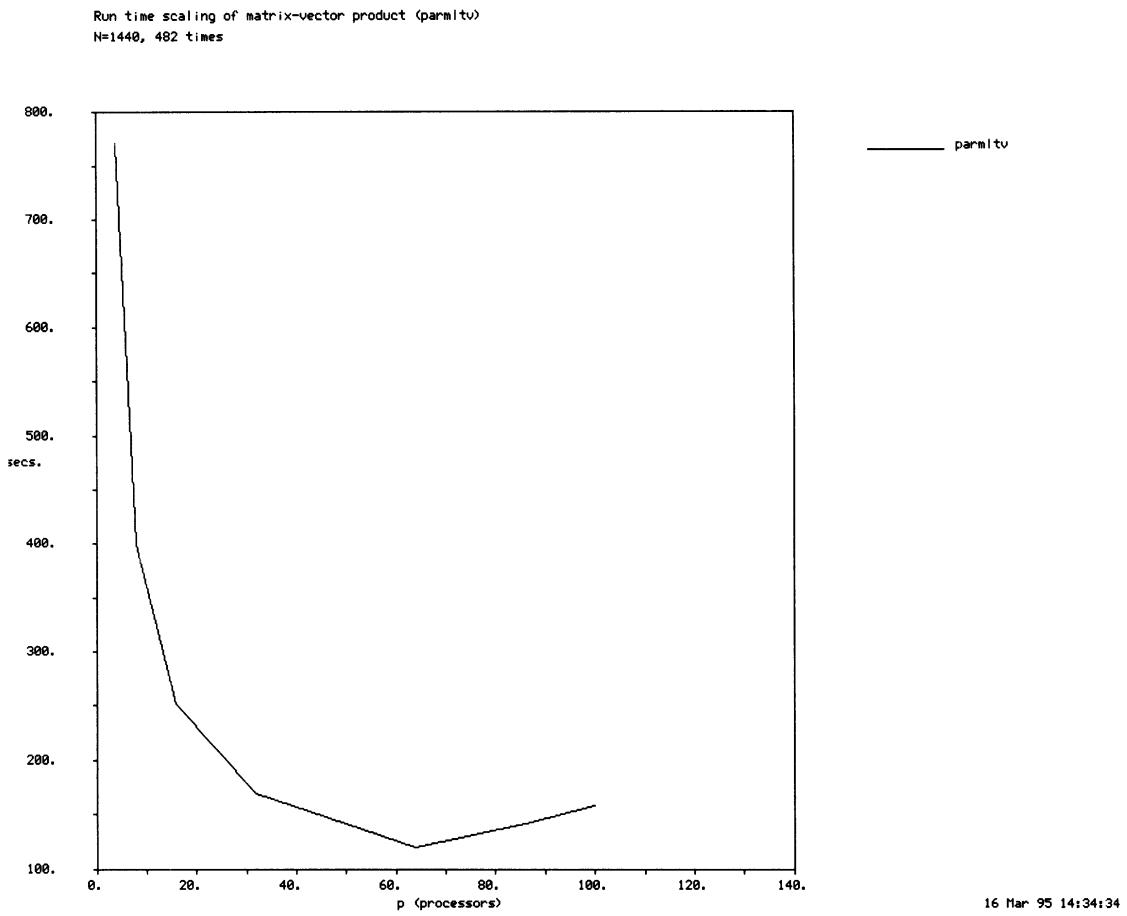


FIGURE 3. Run time scaling of matrix-vector product (N=1440, 482 times)

7.0 Conclusions/Further optimizations

The only portion of the overall application that does not run in parallel is the evaluation of equation (31) in [1]. We have however scaled this operation analytically by the number of blades using the inherent overlapping modal and geometric symmetries [3]. Still, the evaluation of the matrix inverse and matrix-matrix multiplication require substantial

compute time for large problems. Unfortunately, when using a mesh-connected machine such as the Intel Paragon, the communication time required for these operations washes out the gains from parallelization. Newer, shared memory machines such as the KSR or the Convex alieviate the need for message passing and have faster communications. These shared memory machines are more appropriate for the operations that we have not parallelized on the Intel Paragon and could be used to parallelize the entire mode matching code. However, the shared memory machines do not have as many processors as are available on the Intel Paragon due to the cost. Therefore, it must first be determined if parallelization of the entire code on a shared memory machine with far fewer processors than is available on the Intel Paragon can improve the overall performance over what has been presented here.

TABLE 3. Model parameters for run times shown in Figure 4.

fGHZ	6	8	10	12	14
a	3	3.5	4	4.5	5
b	1.5	1.75	2	2.25	2.5
L1	4	4.667	5.334	6	6.667
L2	2	2.333	2.667	3	3.333
N_b	8	8	8	8	8
gap	40	40	40.2	40	40
orient	2.5	2.5	2.4	2.5	2.5
twist	0	0	0	0	0
RimFlag	on	on	on	on	on
IN	21	23	25	28	30
II	10	10	10	11	12
JN	6	7	7	7	7
JJ	8	8	8	10	12
bgas	.0001	.0001	.0001	.0001	.0001
divmax	10	10	10	10	10

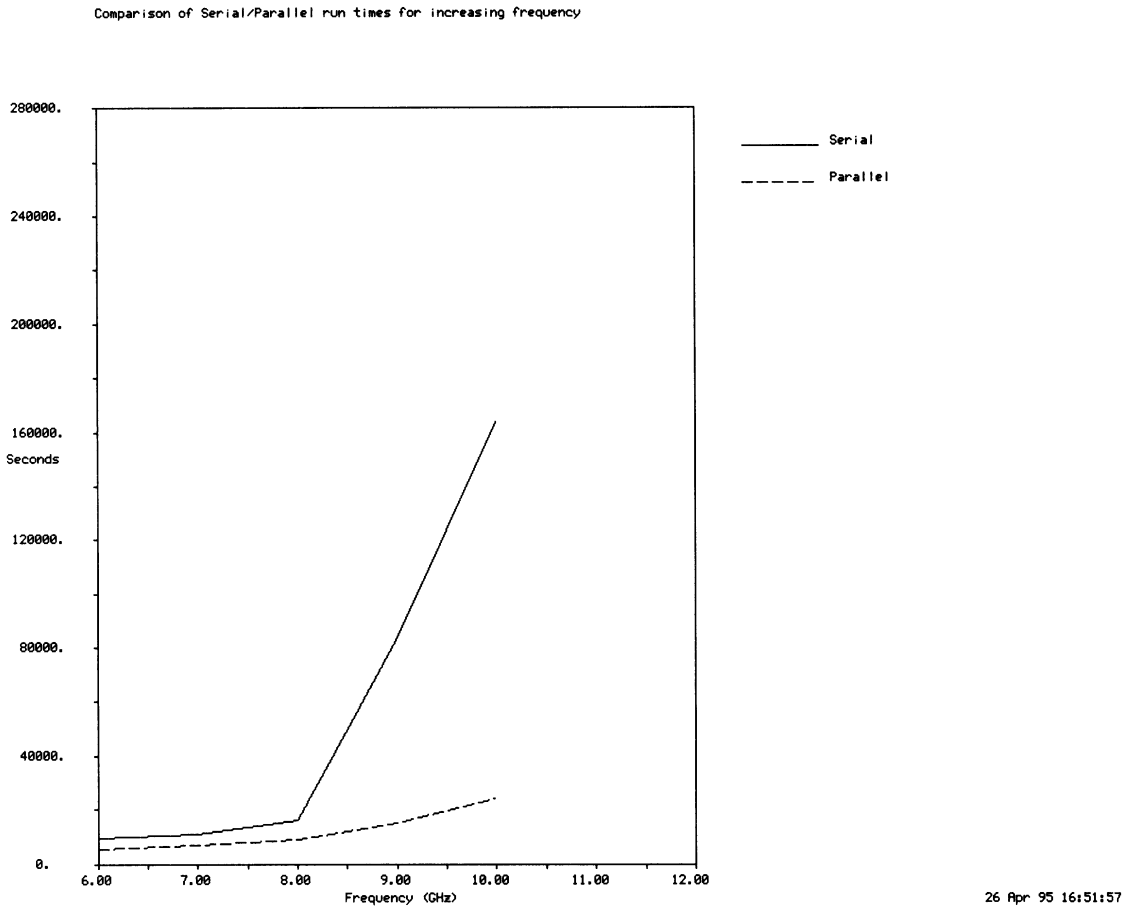


FIGURE 4. Run times for the jet engine mode matching code for increasing frequency.

8.0 References

[1] J.L Volakis, H. Anastassiu, D. Ross, "Analysis of Jet Engine Inlets: Annual Report", Technical Report 030395-6-T, The University of Michigan, EECS Radiation Laboratory, Ann Arbor MI, December 1994.

[2] Cornelis F. Du Toit, "The Numerical Computation of Bessel Functions of the First and Second Kind for Integer Orders and Complex Arguments", *IEEE Transactions on Antennas and Propagation*, vol. 38, no. 9, Sept. 1990.

[3] Daniel C. Ross, John L. Volakis and Hristos T. Anastassiu, "Overlapping Modal and Geometric Symmetries for Computing Jet Engine Inlet Scattering", *IEEE Transactions on Antennas and Propagation* - to appear.

The Mode Matching Technique for Electromagnetic Scattering by Jet Engine Inlets

Hristos T. Anastassiu and John L. Volakis
Radiation Laboratory
Department of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor MI 48109-2122

May 1995

Abstract

The Mode Matching Technique is employed for the evaluation of the Radar Cross-Section (RCS) of cylindrical inlets terminated by a cylindrical array of grooves that may be straight or curved. The numerical results obtained are in very good agreement with actual measurements. The computational requirements and the limitations of the method are described in full detail.

1 Introduction

In a series of earlier reports [1, 2, 3, 4] we presented the application of the semi-analytic Mode Matching (MM) technique to the evaluation of the Radar Cross Section (RCS) of engine-like structures. Three different geometries were analyzed and mathematical expressions for their RCS were derived. The most serious difficulty that we encountered was the complete lack of reference data and our subsequent inability to validate our relevant computer code. Under the request and the guidance of the Radiation Laboratory, an actual model of the geometries was built and measured at GE Aircraft Engines, Cincinnati OH [5]. The initial comparisons of our data with the measurements [4] were not considered sufficiently good. In this report new, improved results for various engine configurations are given. Futhermore, the modifications in the input data and the main body of the code that made these results possible are explained. The good agreement with the measurements implies that the computer code has been completely debugged, and is therefore ready for release. A brief manual explaining the use and the capabilites of the code is included in an Appendix. Finally, the computational requirements and the limitations of the technique are discussed.

2 Corrections to the old version of the code and improved results

In [4] we presented some preliminary RCS predictions for the geometries depicted in Figs. 1 and 2. The comparisons between calculated and measured data in [4] were not satisfactory. The reasons for the disrepancies are listed below:

- The numerical integration subroutine in the old version of the code was not accurate enough. In the new version, double precision arithmetic is used along with an adaptive numerical integration scheme and rigorous convergence checks.
- The mode normalization used in the curved blades geometry was incorrect. Careful study of the original paper by Reiter [6] revealed that the generally accepted form of the Generalized Telegraphist's Equations is valid only if the following normalization is used:

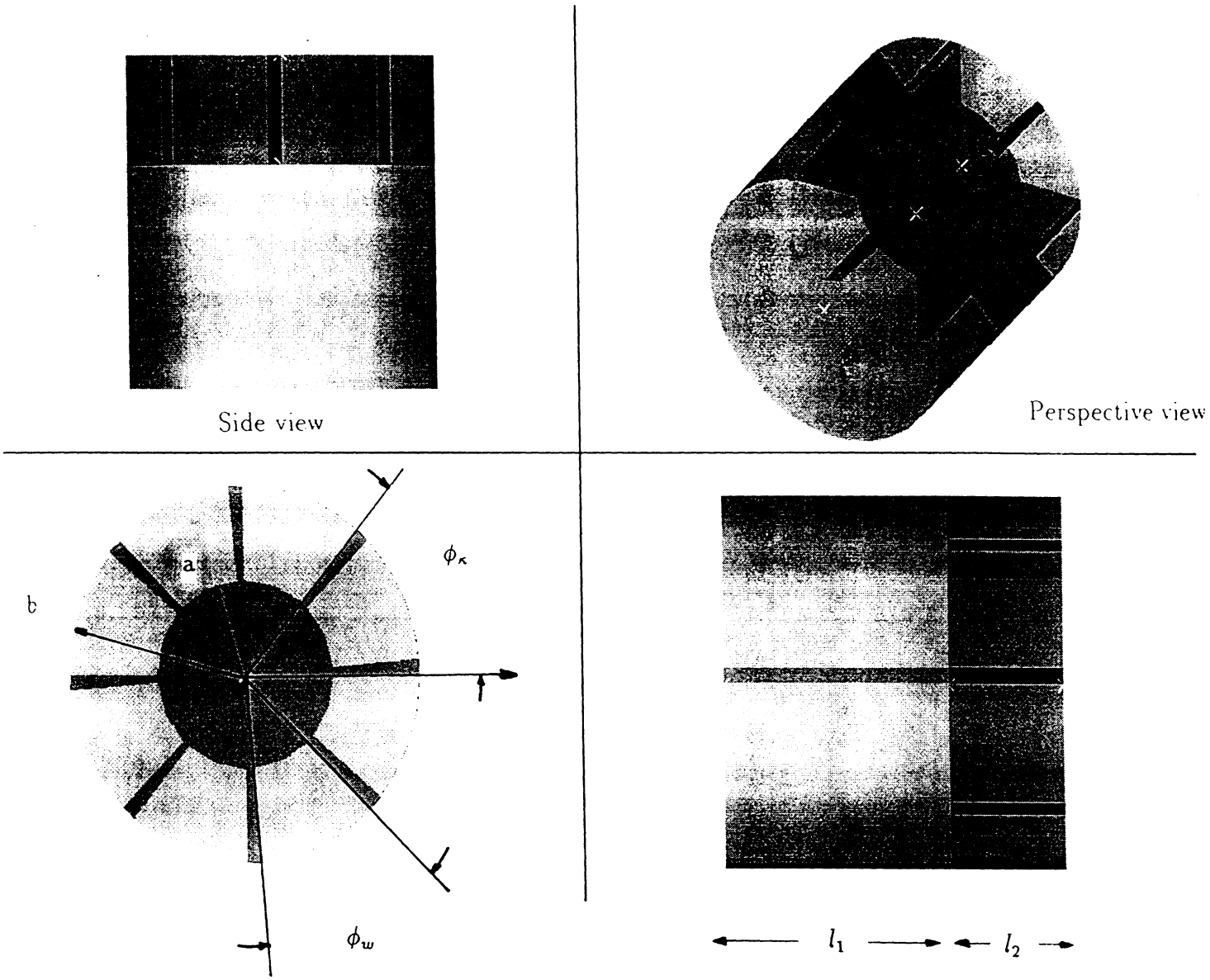


Figure 1: Inlet terminated by an array of straight blades.

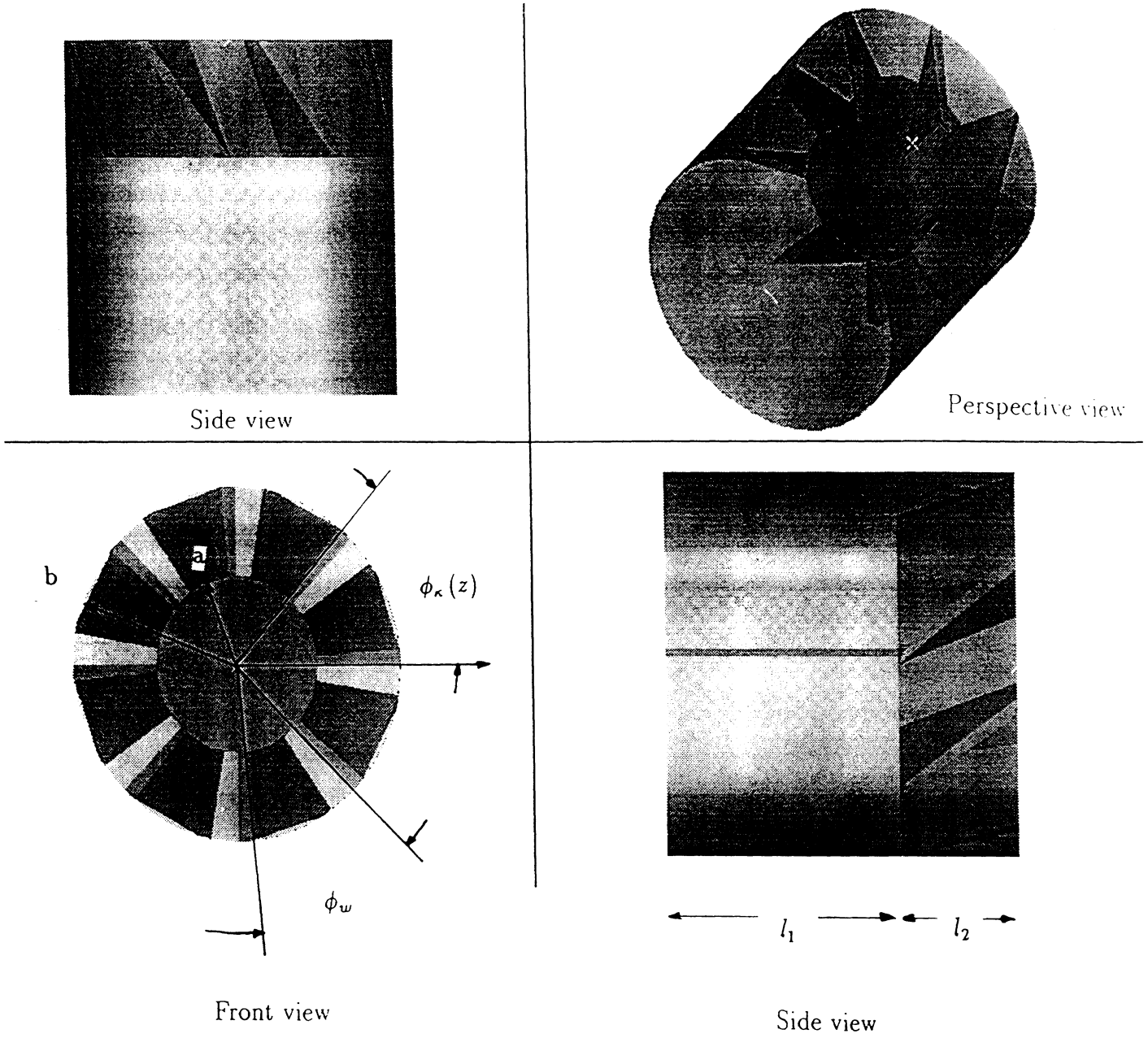


Figure 2: Inlet terminated by an array of curved blades.

$$\int_S \int \mathbf{e}_i \cdot \mathbf{e}_j d^2S = \delta_{ij} \quad (1)$$

$$\int_S \int \mathbf{h}_i \cdot \mathbf{h}_j d^2S = \delta_{ij} \quad (2)$$

where $\mathbf{e}_i$ and $\mathbf{h}_i$ are the transverse electric and magnetic fields corresponding to the i^{th} mode and S is the cross-section of the waveguide. It is worth mentioning that the importance of this normalization is not stressed out in either [6] or any other paper that makes use of the same equations. The aforementioned normalization is obviously inconsistent with the more widely accepted

$$\frac{1}{2} \int_S \int \mathbf{e}_i \times \mathbf{h}_j \cdot d^2\mathbf{S} = \delta_{ij} \quad (3)$$

which was used in the old version of the code.

• The blade width of the actual model turned out to be 5^o, and not 4^o as requested [5]. The 5^o width was used in the new set of results.

• The orientation of the blades in the actual measurements was not identical with the test-matrix specification [5]. In the new set of results, the orientation was such that the x-axis bisected the first blade of the array, so that the measurement configuration is matched by the computational model.

After the aforementioned modifications the generated numerical results agreed very well with the measured data. The new set of results is presented in Figs. 3- 7. The solid line corresponds to calculated and the dotted line to measured data. Also, a plot of the tangential fields at the interface between the hollow region and the termination showing field continuity is given in Figs. 8 and 9.

To give more details about the extraction of the calculated data, let us define the following parameters: N_1: number of orders of cylindrical functions used in region 1, M_1: number of modes per order used in region 1, N_2: number of orders of cylindrical functions used in region 2, M_2: number of modes per order used in region 2, ϕ_{tw}: twist angle per wavelength of the curved blades. Futhermore, let $a, b, l_1, l_2, \phi_w, \phi_1$ be the same parameters defined in Figs. 1, 2. For each one of the analyzed geometries the input parameters were as follows:

6 GHz, straight blades: $b = 3\lambda$, $a = 1.5\lambda$, $l_1 = 4\lambda$, $l_2 = 2\lambda$, $\phi_w = 40^o$, $\phi_1 = 2.5^o$, $N_1 = 21$, $M_1 = 10$, $N_2 = 6$, $M_2 = 8$.

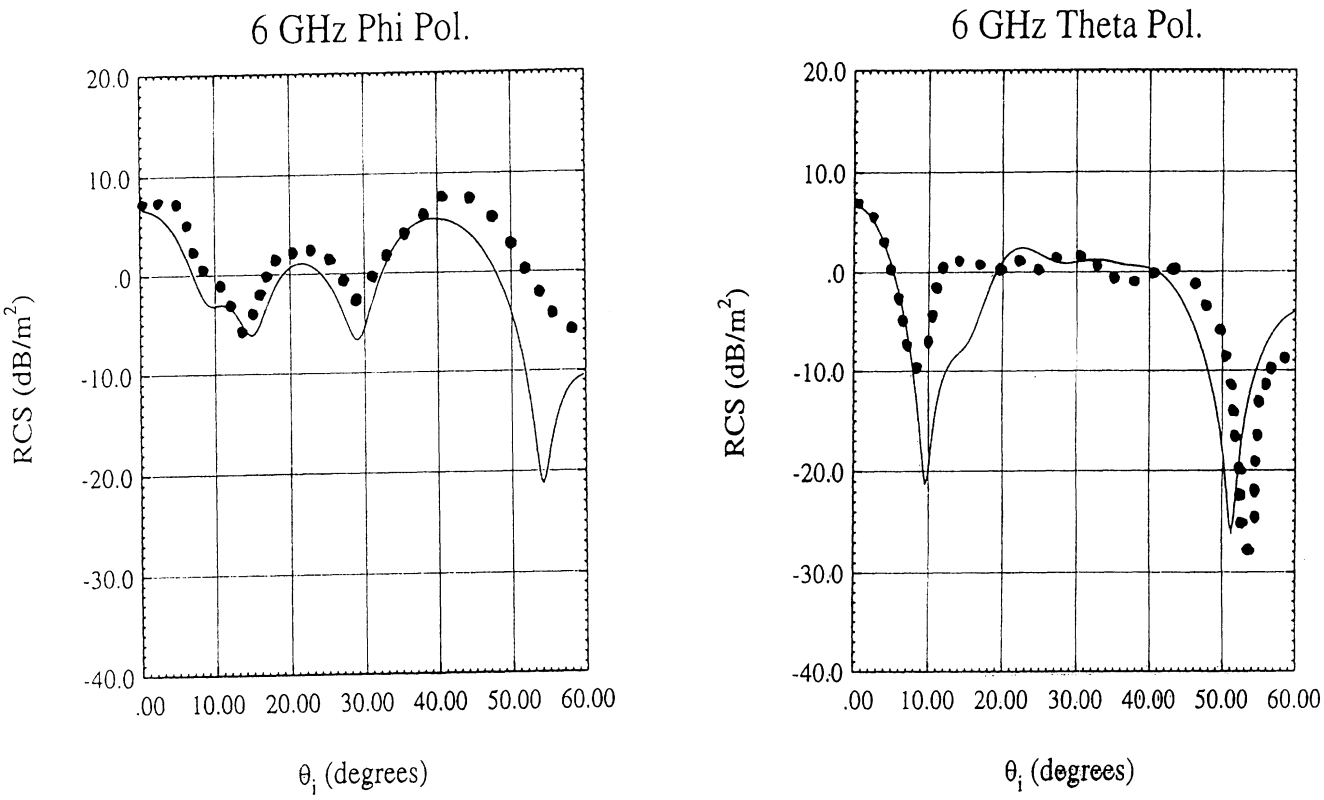


Figure 3: Data for the straight blades geometry at 6 GHz.

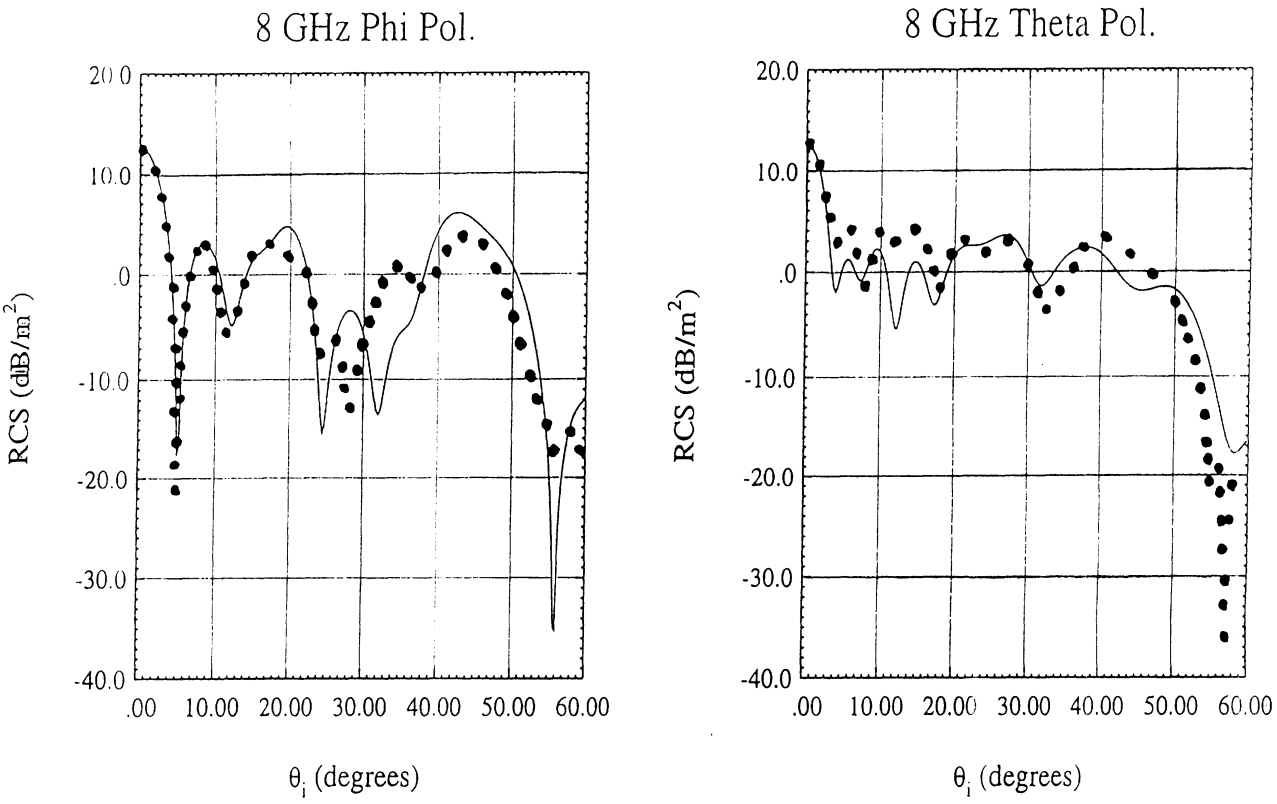


Figure 4: Data for the straight blades geometry at 8 GHz.

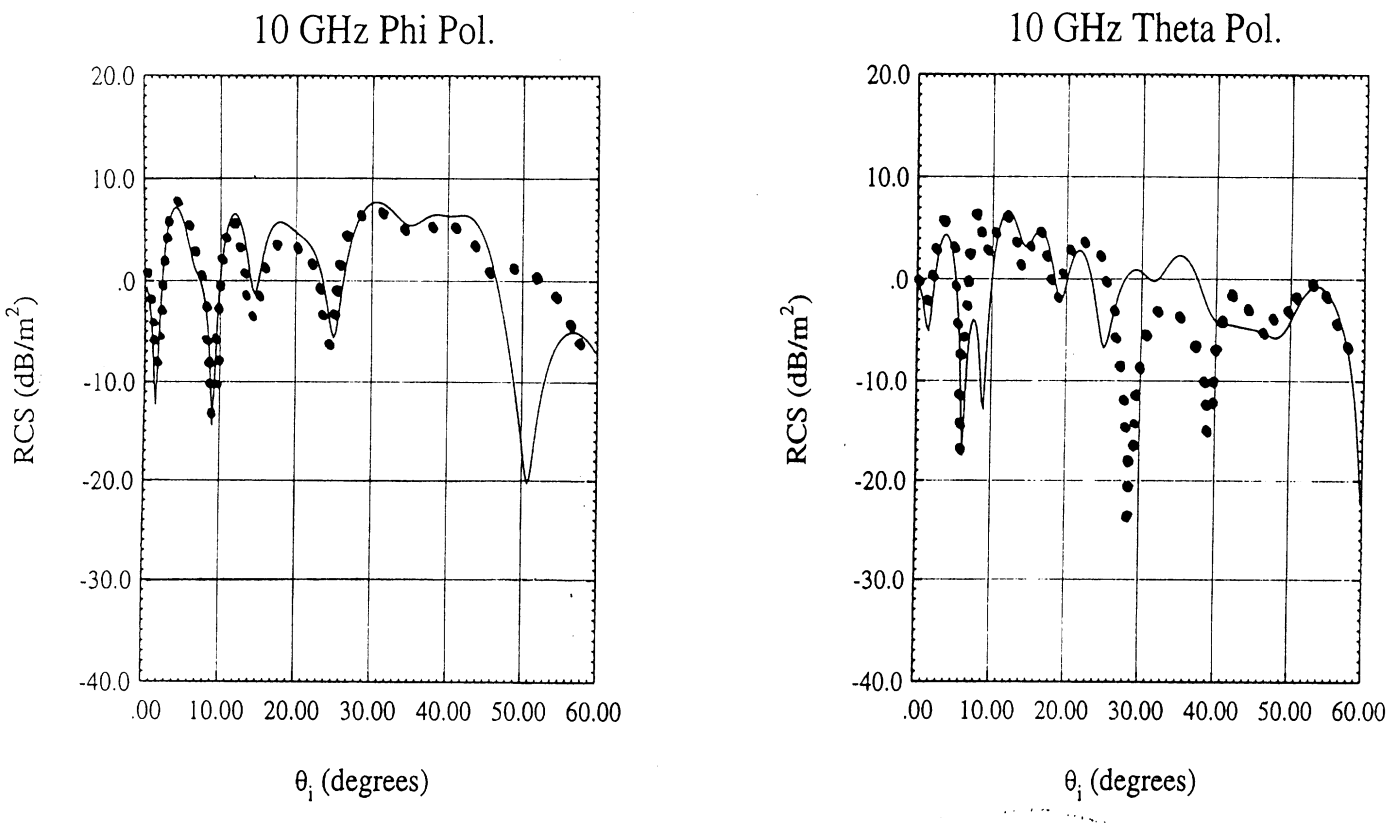


Figure 5: Data for the straight blades geometry at 10 GHz.

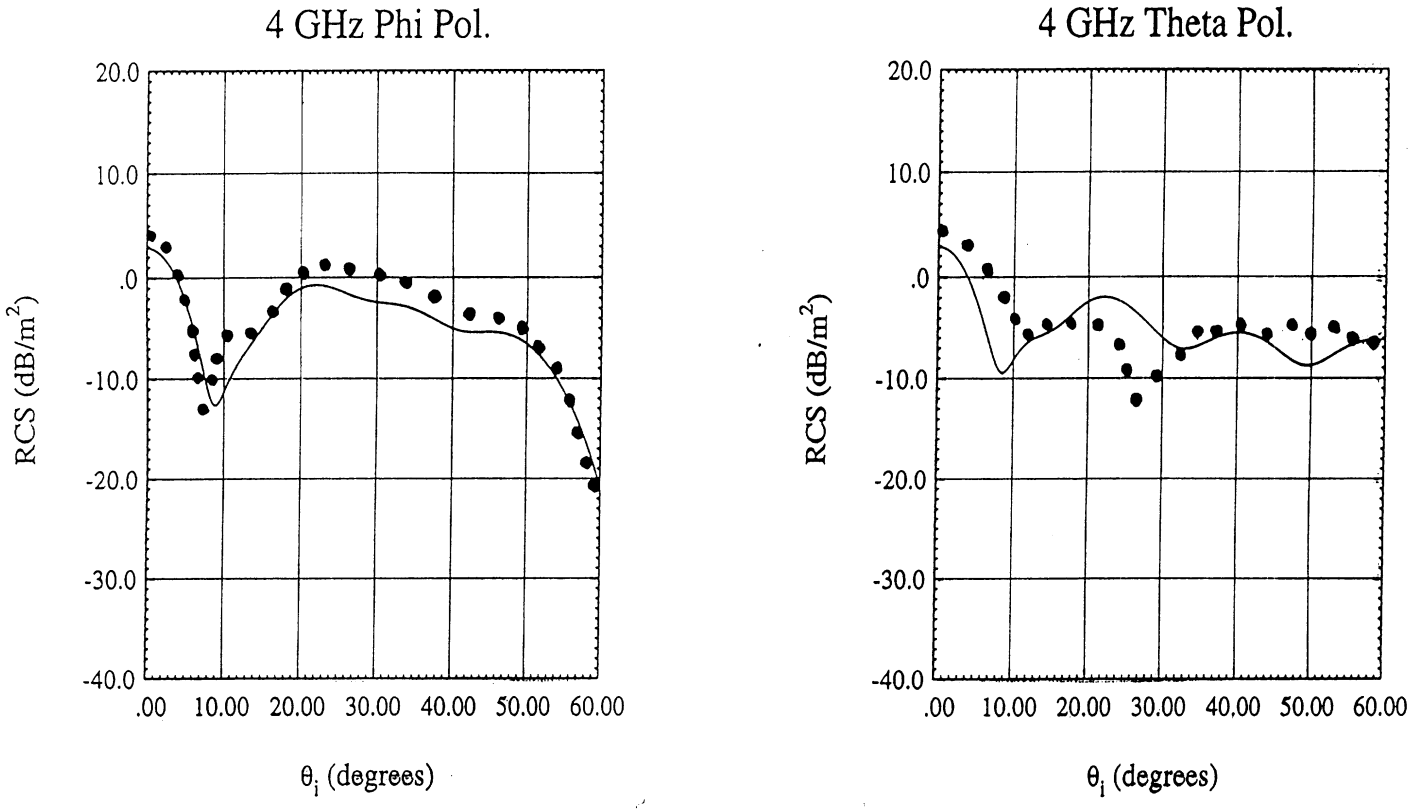


Figure 6: Data for the curved blades geometry at 4 GHz.

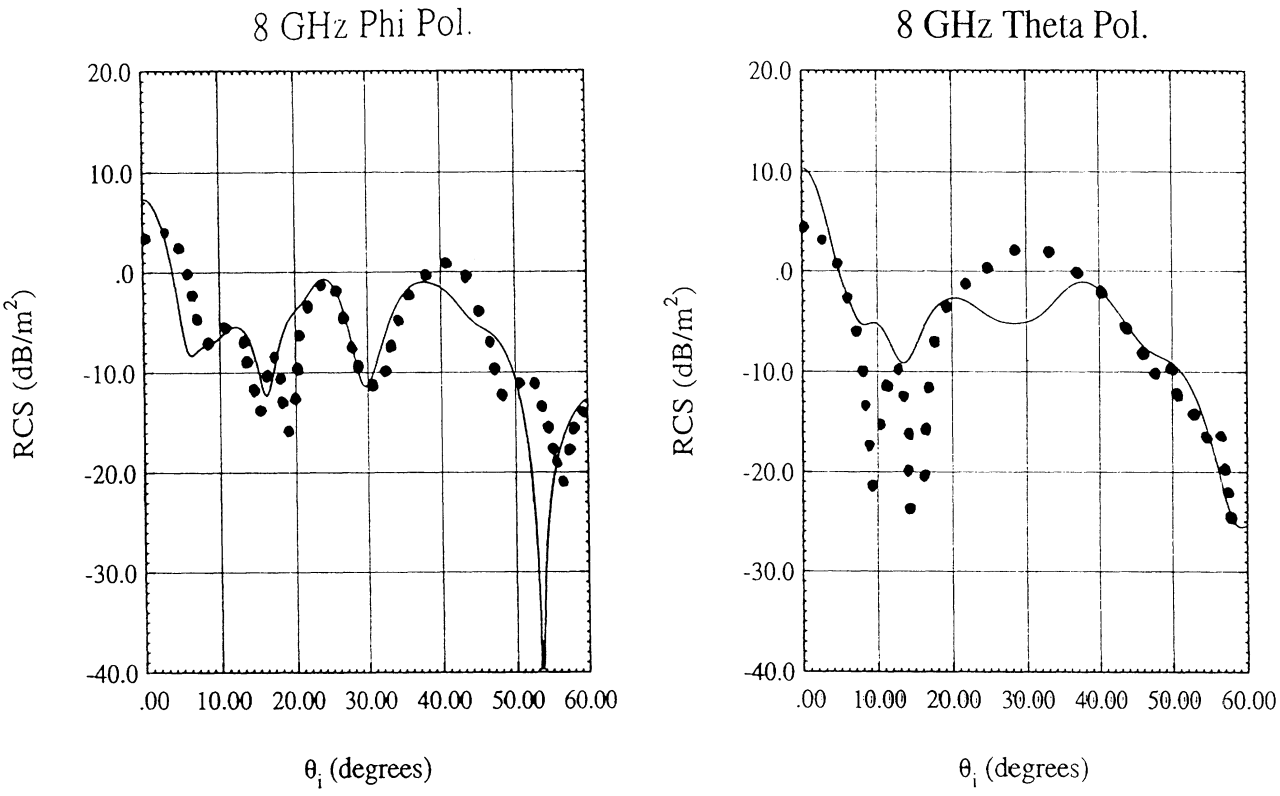


Figure 7: Data for the curved blades geometry at 8 GHz.

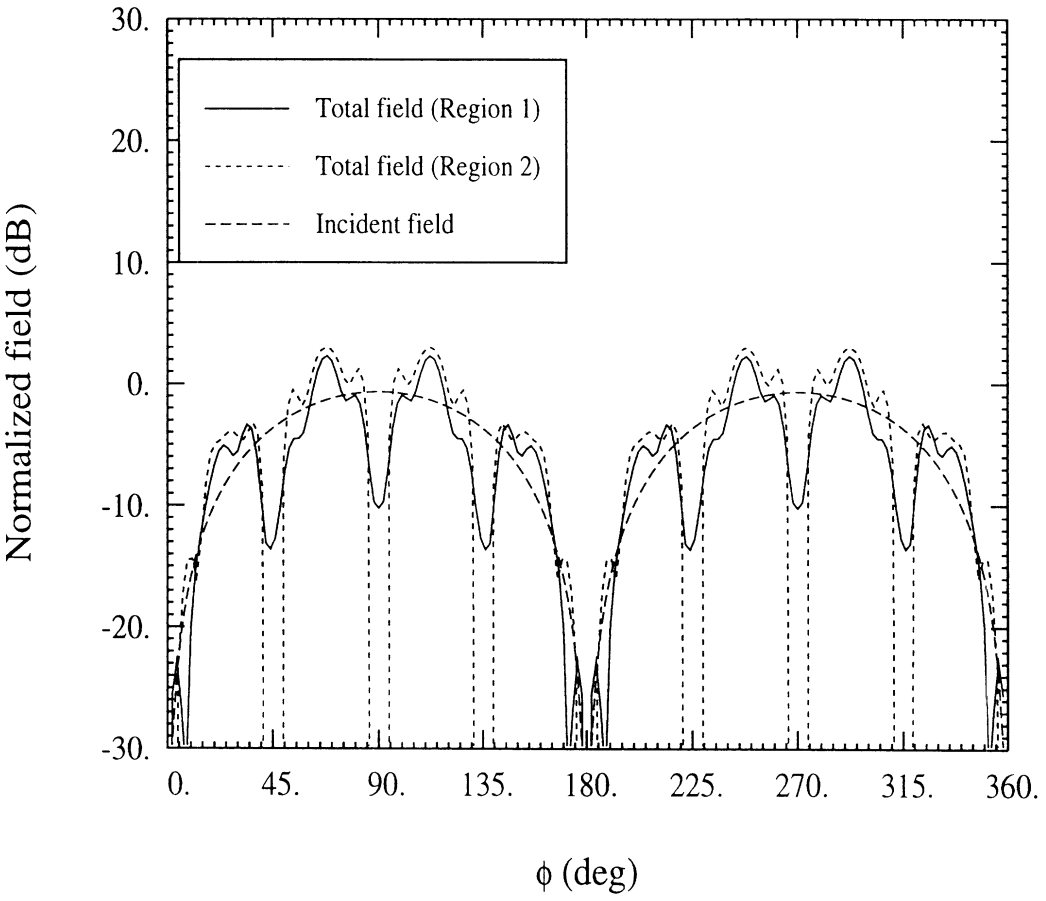


Figure 8: Transverse electric fields (ρ component) at both sides of the interface between the hollow region and the termination for the straight blades geometry at 6 GHz. The fields are plotted at $\rho = 2\lambda$ for an incidence angle of $\theta_i = 0^\circ$ and ϕ polarization. The azimuth angle ϕ is measured counterclockwise from the measurement plane.

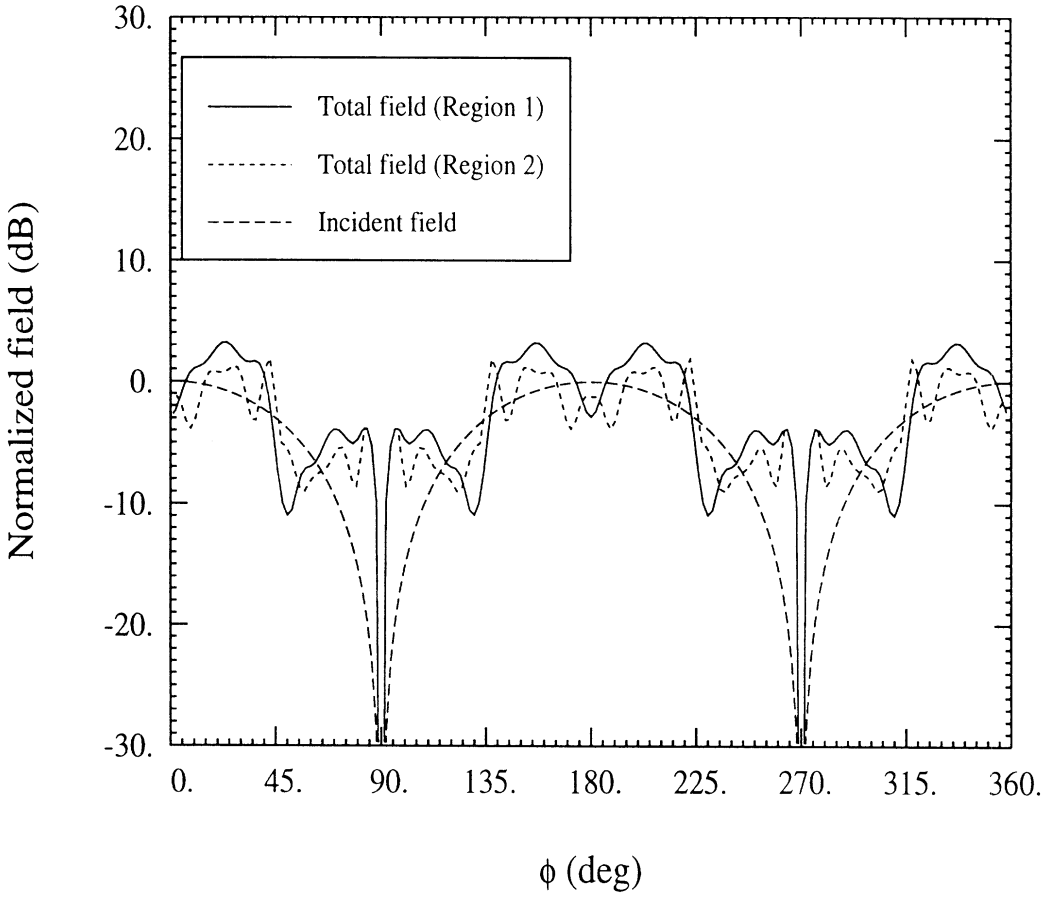


Figure 9: Transverse electric fields (ϕ component) at both sides of the interface between the hollow region and the termination for the straight blades geometry at 6 GHz. The fields are plotted at $\rho = 2\lambda$ for an incidence angle of $\theta_i = 0^o$ and ϕ polarization. The azimuth angle ϕ is measured counterclockwise from the measurement plane.

8 GHz, straight blades: $b = 4\lambda$, $a = 2\lambda$, $l_1 = 5.334\lambda$, $l_2 = 2.667\lambda$, $\phi_w = 40^o$, $\phi_1 = 2.5^o$, $N_1 = 25$, $M_1 = 10$, $N_2 = 7$, $M_2 = 8$.

10 GHz, straight blades: $b = 5\lambda$, $a = 2.5\lambda$, $l_1 = 6.667\lambda$, $l_2 = 3.333\lambda$, $\phi_w = 40^o$, $\phi_1 = 2.5^o$, $N_1 = 30$, $M_1 = 12$, $N_2 = 7$, $M_2 = 12$.

4 GHz, curved blades: $b = 2\lambda$, $a = 1\lambda$, $l_1 = 2.667\lambda$, $l_2 = 1.333\lambda$, $\phi_w = 40^o$, $\phi_1 = 2.5^o$, $\phi_{tw} = 0.3928\ rad \cdot \lambda^{-1}$, $N_1 = 25$, $M_1 = 10$, $N_2 = 9$, $M_2 = 9$.

8 GHz, curved blades: $b = 4\lambda$, $a = 2\lambda$, $l_1 = 5.334\lambda$, $l_2 = 2.667\lambda$, $\phi_w = 40^o$, $\phi_1 = 2.5^o$, $\phi_{tw} = 0.1963\ rad \cdot \lambda^{-1}$, $N_1 = 30$, $M_1 = 10$, $N_2 = 9$, $M_2 = 9$.

3 Computational Requirements and Limitations

The results prove that the MM method is very accurate, nevertheless it can become computationally intensive for large size engines. The number of modes that are necessary for matching the tangential fields at the interface between the hollow region and the termination grows drastically with size, resulting to large modal matrices. Using the amount of memory that was necessary for relatively small diameters (up to 10λ), third-order curve fitting generated the plot in Fig. 10. Similarly, Fig. 11 estimates by extrapolation the operation time on a serial machine for large geometries. For the analysis of realistic problems, where the engine is about 50λ in diameter, the computing facilities at the Radiation Laboratory are not sufficient. A parallelized version of the code has, though, been prepared [7] so that the operation time is reduced as much as possible. The storage requirements, though, are still very high.

Another limitation of the method is that strict convergence tests are very hard to implement. If one excessively increases the number of modes in a specific geometry, the condition number of the involved matrices and the individual eigenvalues of the G.T.E. system in the curved blades case deteriorates, yielding unreliable results. Unfortunately, it appears that these numerical problems usually begin to occur before convergence has been absolutely guaranteed. Nevertheless, at least for the geometries that have been analyzed in this report, the computed data are adequately close to the mea-

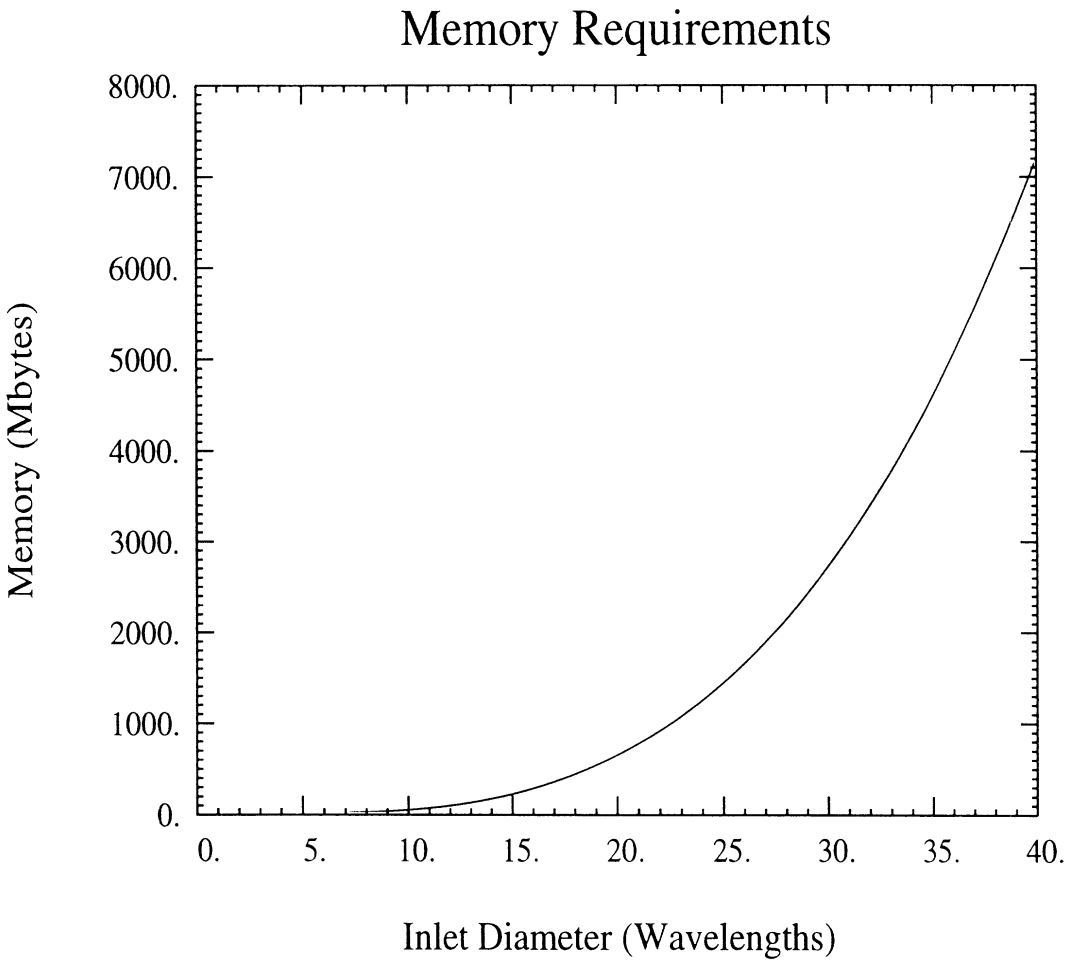


Figure 10: Memory requirements of the computer code.

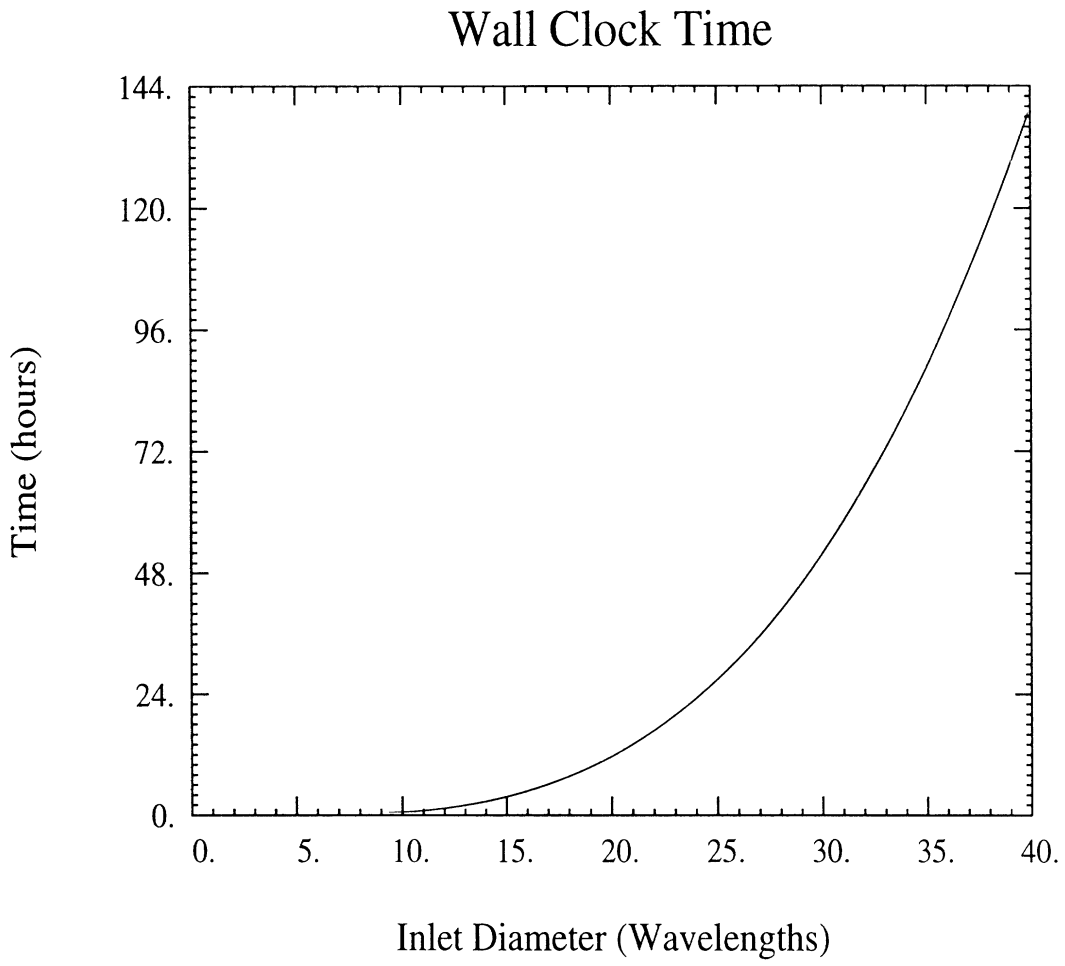


Figure 11: Operation time on a serial machine at 275 MFlops.

sured ones, even if some fluctuation around the correct values can still be observed.

4 Summary and Conclusions

In this report we presented numerical results of the MM technique applied to the evaluation of the RCS of engine-like configurations. The calculated data were in very good agreement with measurements. We explained the modifications in the old version of the computer code that resulted in the improved data given here. Finally, we addressed the efficiency, demands and the limitations of the method.

5 Appendix: The Computer Code Manual

5.1 Introduction

This is a brief user's manual for the Mode Matching (MM) code developed at the University of Michigan between 1993 and 1995, written in FORTRAN 77 for the actual computation of the Radar Cross Section (RCS) of specific versions of the inlet geometry. The purpose of this manual is to describe the code, give useful guidelines to potential users and point out what the limitations of the code are.

5.2 General Description of the Code

The code follows strictly the mathematical analysis described in [4] for the geometries described in Figs. 1 and 2. Given the specific parameters of the geometry, the generalized scattering matrix of the termination is computed. The coefficients of the ingoing modes are computed by coupling of the incident plane wave to the cylindrical waveguide fields. Multiplication of the scattering matrix with the ingoing modes coefficient vector yields the outcoming modes coefficient vector. The outcoming fields give rise to equivalent currents on the open end (aperture) of the inlet, and Kirchhoff integration of these equivalent currents yields the scattered field and the RCS of the structure.

The numbering of the modes in region 1 (the hollow part of the inlet) has been coded as follows: Assume we use N_1 orders of the Bessel function (i.e. use functions $J_0(k_{m0}\rho)$ to $J_{N_1-1}(k_{m,N_1-1}\rho)$) and M_1 modes per order, i.e. let m range from 1 to M_1. Then, the total number of modes is $4N_1M_1$ (N_1M_1 TM modes, plus N_1M_1 TE modes, multiplied by 2 for even and odd ϕ dependence of the corresponding vector potential). The TM_{0m} even modes ($m = 1, ..., M_1$) correspond to the first M_1 modes of the numbering scheme. The TM_{1m} even modes ($m = 1, ..., M_1$) correspond to modes $M_1 + 1...2M_1$, e.t.c. Finally, the $TM_{N_1-1,m}$ even modes ($m = 1, ..., M_1$) correspond to modes $(N_1-1)M_1+1...N_1M_1$. Using the same numbering scheme, the TE_{nm} odd modes correspond to mode numbers $N_1M_1 + 1...2N_1M_1$, the TM_{nm} odd modes correspond to mode numbers $2N_1M_1+1...3N_1M_1$ and finally the TE_{nm} even modes correspond to mode numbers $3N_1M_1 + 1...4N_1M_1$. To visualize the numbering scheme, it is helpful to think of any coefficient vector as a

collection of four subvectors, each corresponding to some particular set of modes, in the following format:

$$\left\{ \begin{array}{c} TMeven \\ TEodd \\ TModd \\ TEeven \end{array} \right\} \tag{4}$$

The scattering matrix has the format of the cartesian product of two vectors of this kind, i.e. the numbering of both rows and columns follows exactly the same rules.

5.3 Running the Code

Some of the parameters of the geometry are entered by the user in the file "param4.inc", while the rest are given by the user in an interactive manner.

The input parameters in the file "param4.inc" are:

in: equal to N_1, as defined in the previous section.

ii: equal to M_1, as defined in the previous section.

jn: equal to N_2, as defined in the previous section.

ji: equal to M_2, as defined in the previous section.

ngas: equal to the number of abcissas per subinterval used in the Gaussian integration. A typical value is 3.

bgas: equal to the tolerance accepted in the Gaussian integration. A typical value is 10^{-5}.

The user has to enter the rest of the parameters interactively by answering the following questions:

Enter the large inlet radius in wavelengths: Enter the hollow cylinder radius b.

Enter the small inlet radius in wavelengths: Enter the hub radius a.

Enter l1 (length of region 1) in wavelengths: Enter length l_1 of the hollow section of the inlet.

Enter l2 (length of region 2) in wavelengths: Enter length l_2 of the termination. The blades are assumed to be backed by a PEC plate.

Enter number of blades: Self explanatory.

Enter fw (angular width of each groove) in degrees: Self explanatory.

Enter fst (phi angle of first groove) in degrees: Enter the angle formed by the measurement plane and the rightmost edge of the first groove. This angle is identical to ϕ_1 in Fig. 1.

Straight or curved blades? (s=1,c=2) Self explanatory.

Enter the gradient of each blade (rad/lambda): Enter the twist angle per wavelength for the curved blades.

Enter the lowest order for the asymptotic evaluation of Bessel functions in region 2: If N_2 has to be extremeley large in region 2, it may be more efficient to use asymptotic formulas of the Bessel functions in the evaluation of the elements of the GTE matrix. This option should be used with caution, and only if exact computation causes serious numerical problems. Enter a very large number to avoid any asymptotic calculation. Recall that the maximum order of Bessel functions in region 2 is equal to $(N_2 - 1)\,\pi/\phi_w$.

Enter 1 for inclusion of rim contribution: Enter 1 if you want to include the contribution of the fringe wave at the lip of the inlet.

Enter measurement frequency in GHz: Enter frequency so that the RCS is expressed in dB/m^2.

Do you need the scattering matrix (y=1,n=2)? Enter 1 if you want the scattering matrix stored in the file "scatmat". Make sure that enough memory is available for this matrix storage.

As soon as all the parameters are given, the code calculates the eigenvalues of the modes. A list of the modes and their characteristic wavenumber k_ρ expressed in λ^{-1} appears on the screen. Index 1 in the third column corresponds to traveling modes and index -1 corresponds to evanescent modes.

The GTE matrix is also determined and the eigenvalues and the eigenvectors are computed. There is an option for checking the calculated eigenvalues and how accurate they are. A subroutine calculates the products $(\mathbf{M} - \lambda\mathbf{I})\mathbf{x}$, where $\mathbf{x}$ is the eigenvector corresponding to the eigenvalue λ. Theoretically the resulting vectors should be exactly zero. Their maximum numerical entry appears on the screen, showing the accuracy of the eigenvalues. Further comments on this calculation are given in the relevant subroutine.

The procedure described in [4] involves the inversion of three matrices. The inverse condition number of each matrix appears on the screen, yielding a measure of the accuracy of each numerical inversion. Condition numbers greater that 10^{12} should be considered poor for double precision arithmetic (or greater than 10^6 for single precision) and the corresponding inversion is not likely to be of acceptable accuracy.

The scattering matrix is readily obtained upon completing the aforementioned matrix inversions. The RCS can then be plotted as a function of the incidence angle θ_i. There is also an option for checking the continuity of the tangential fields at the interface between regions 1 and 2, which constitutes the basis of the MM technique. The fields at the interface are plotted as a function of the azimuth angle ϕ, at a constant ρ and for incidence angles specified by the user. The first column is the angle ϕ, the second column gives the ρ component of the total field at the interface and the third column gives the ϕ component of the field. The numerical values should be the same in regions 1 and 2. Also, the incident field in region 1 is plotted (ρ and ϕ component in columns 4 and 5 respectively). All fields are normalized to the maximum value of the incident field.

Finally, the RCS pattern is plotted in dB/m^2 as a function of the incidence angle (0 to 60 degrees) for both ϕ and θ polarizations.

5.4 Comments on the Use of the Code

The MM technique yields very accurate results, provided the modes in regions 1 and 2 are carefully chosen by the user. All propagating modes should be used, along with a number of evanescent modes, so that continuity of the tangential fields is achieved at the interface between the two regions. Unfortunately, the number of necessary evanescent modes cannot be determined a priori. Only successive runs of the code can show if sufficient convergence of the results has been achieved. Poor choice of the modes may lead to ill-conditioned, even non-invertible matrices, and to avoid such situations, a general rule of thumb is that the highest order modes in regions 1 and 2 should have approximately the same cut-off frequency. It is therefore inferred that the number of modes used in region 1 should always be larger than the number of modes in region 2. Also, the solution of the GTE equation may cause numerical problems if any two of the eigenvalues happen to be very close to each other. In this case a different, smaller set of modes should be used in region 2.

So far the code has been proven to yield fairly accurate results for a series of test geometries as described in previous sections of this report. Further investigation on the convergence properties will take place using the parallelized version of the code.

References

[1] H. T. Anastassiu and J. L. Volakis, "The mode-matching technique for electromagnetic scattering by circular inlets with canonical terminations", Report 030395-2-T, University of Michigan, Radiation Laboratory, May 1993.

[2] H. T. Anastassiu and J. L. Volakis, "The mode-matching technique for electromagnetic scattering by inlets with complex terminations ", Report 030395-3-T, University of Michigan, Radiation Laboratory, October 1993.

[3] H. T. Anastassiu, D.C. Ross and J. L. Volakis, "Progress report on the analysis of jet engine inlets", Report 030395-5-T, University of Michigan, Radiation Laboratory, July 1994.

[4] J. L. Volakis, H. T. Anastassiu and D. C. Ross, "Analysis of Jet Engine Inlets", Annual Report 030395-6-T, University of Michigan, Radiation Laboratory, December 1994.

[5] G. Crabtree, W. Huegle, D. Salisbury, "RCS compact range test results for a set of simplified engine face models", Report TM94226, GE Aircraft Engines, One Neumann Way, Cincinnati OH, November 1994.

[6] G. Reiter, "Generalized Telegraphist's Equation for waveguides of varying cross-section", *Proc. I.E.E.* vol. 106B, suppl. 13, pp. 54-57, 1959.

[7] D. C. Ross, J. L. Volakis and H. T. Anastassiu, "Parallelization of a jet engine Mode Matching Code", to be submitted to the *Antennas and Propagation Magazine*

MISSING PAGE

Proposed Future Work on the Jet Engine Inlet Analysis

Hristos T. Anastassiu and John L. Volakis
Radiation Laboratory
Department of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor MI 48109-2122

May 1995

1 Introduction

The characterization of the scattering properties of jet engine inlets has been our basic research project for the past two years. Our main effort was focused on proposing simple geometrical engine-like models that are amenable to exact solutions. A number of canonical geometries that simulate a jet engine have been rigorously analyzed, and very good results have been obtained. Although this task has been successfully completed, the need for more flexible, numerical methods is obvious, since the geometrical and physical features of realistic engines are too complicated to handle via any analytical technique. The purpose of this report is to propose methods suitable for the analysis of jet engines with arbitrary, irregular, not necessarily perfectly conducting terminations.

2 On the Application of Wavelets to the Moment Method

2.1 General Theory

Although the Mode Matching technique yields very accurate results for canonical geometries, it is not applicable to structures where conventional (or generalized) mode field representations are not available. Realistic jet engine inlets belong to this category since their geometries are extremely irregular and complicated. Furthermore, sections of the geometry are coated with dielectric meterials, and this precludes use of analytical solutions. Thus, a numerical solution is more appropriate for a realistic configuration.

A Finite Element Method (FEM) formulation has been developed in the Radiation Laboratory [1], but it appears that the computational requirements for realistic problems are prohibitive, unless certain symmetries of the geometry are taken into consideration. On the other hand, Moment Method (MM) codes have been developed in the past for bodies of revolution (BOR), where the axial symmetry simplifies the computational rigors. Unfortunately, MM is difficult to apply in general three-dimensional shapes. If conventional basis functions are used (pulse, roof-top, pieciewise sinusoidal e.t.c.), the impedance matrix is fully populated, making the method computationally expensive. For realistic engine inlets (about 50 wavelengths wide in diameter) application of the MM would have exorbitant storage and CPU time requirements.

Recently, the concept of wavelets was considered in applied electromagnetics. It has been shown [2] that if the wavelets and scaling functions of a suitable multiresolution analysis (MRA) are used as basis functions in MM algorithms, the resulting impedance matrices may be fairly sparse. This interesting property has been succesfully exploited in a number of papers for solving one-dimensional problems [2, 3]. Lately, two-dimensional Daubechies wavelets were used in the analysis of a microstrip problem [4] and the impedance matrix was again quite sparse. It is therefore of interest to examine their application to general 3D problems. In this section we will show that all the properties that make wavelets attractive in one-dimensional problems are still valid in three dimensions, and therefore one should expect that their application to a full scattering problem will again lead to sparse

matrices and efficient implementation.

Let us consider the following integral equation

$$\int_\Sigma \int \bar{\bar{\mathbf{G}}}(\mathbf{r};\mathbf{r}') \cdot \mathbf{f}(\mathbf{r}') d^2 S' = \mathbf{g}(\mathbf{r}) \qquad \mathbf{r} \in \Sigma \tag{1}$$

$$\mathbf{f}(\mathbf{r}) = \mathbf{0}, \qquad \mathbf{r} \notin \Sigma \tag{2}$$

where Σ is some arbitrary, open surface, $\mathbf{g}(\mathbf{r})$ is some known excitation, $\bar{\bar{\mathbf{G}}}(\mathbf{r};\mathbf{r}')$ is a dyadic kernel and $\mathbf{f}(\mathbf{r})$ is the unknown function. Let $\mathbf{r}' = \mathbf{r}'(v_1, v_2)$ be some parametrization of the surface, where v_1, v_2 are two orthogonal coordinates with $\mathbf{u}_1, \mathbf{u}_2$ being the associated unit vectors. Eqs. (1), (2) can then be written

$$\int_a^b \int_c^d \bar{\bar{\mathbf{G}}}(\mathbf{r}; v_1, v_2) \cdot \mathbf{f}(v_1, v_2) h_1 h_2 dv_1 dv_2 = \mathbf{g}(\mathbf{r}) \qquad \mathbf{r} \in \Sigma \tag{3}$$

$$\mathbf{f}(\mathbf{r}) = \mathbf{0}, \qquad \mathbf{r} \notin \Sigma \tag{4}$$

where h_1, h_2 are the associated metric coefficients along $\mathbf{u}_1, \mathbf{u}_2$ respectively. On our way to solve for $\mathbf{f}(v_1, v_2)$ we introduce the representation

$$\mathbf{f}(v_1, v_2) = P_1(v_1, v_2)\mathbf{u}_1 + P_2(v_1, v_2)\mathbf{u}_2 + P_3(v_1, v_3)\mathbf{u}_3 \tag{5}$$

where $\mathbf{u}_3 \equiv \mathbf{u}_1 \times \mathbf{u}_2$ and P_1, P_2, P_3 are sums of weighted basis functions defined by

$$\begin{aligned} P_i(v_1, v_2) = \sum_{m=m_l}^{\infty} \sum_{n,k=1}^{\infty} [\;& \alpha_{n,k}^{(i,m)} \; \psi_{m,k}(v_1)\psi_{m,n}(v_2) + \\ & + \beta_{n,k}^{(i,m)} \; \psi_{m,k}(v_1)\phi_{m,n}(v_2) + \\ & + \gamma_{n,k}^{(i,m)} \; \phi_{m,k}(v_1)\psi_{m,n}(v_2)] + \\ & + \sum_{n,k=1}^{\infty} s_{n,k}^{(i,m_l)} \; \phi_{m_l,k}(v_1)\phi_{m_l,n}(v_2) \qquad i = 1,2,3 \end{aligned} \tag{6}$$

where the α's, β's, γ's and s's are unknown complex coefficients, m_l is an arbitrary integer and $\psi_{m,k}$, $\phi_{m,k}$ are dilated and translated versions of an appropriate mother wavelet ψ and scaling function ϕ respectively, i.e.

$$\psi_{m,k}(t) = 2^{\frac{m}{2}}\psi(2^m t - k) \tag{7}$$
$$\phi_{m,k}(t) = 2^{\frac{m}{2}}\phi(2^m t - k) \tag{8}$$

Equation (6) is exact, since the set $\{\psi_{m,k}\psi_{m,n}, \psi_{m,k}\phi_{m,n}, \phi_{m,k}\psi_{m,n}\}_{m,n,k\in\mathbf{Z}}$ is a complete basis for $\mathbf{L}^2(\mathbf{R}^2)$ [4]. Therefore, if its elements are weighted by complex coefficients they span $\mathbf{L}^2(\mathbf{R}^2) \times \mathbf{L}^2(\mathbf{R}^2)$, i.e. they span the space of complex functions of two real variables, with square integrable real and imaginary parts.

To generate a discrete system of equations from (3)- (6) we define an inner product between two functions in $\mathbf{L}^2(\mathbf{R}^2) \times \mathbf{L}^2(\mathbf{R}^2)$

$$< f(t_1,t_2), g(t_1,t_2) > \equiv \int_\Sigma \int f^*(t_1,t_2) g(t_1,t_2) d^2S \tag{9}$$

For simplicity let us use the notation $\tilde{\psi}_{m,k}$ for either $\psi_{m,k}$ or $\phi_{m,k}$. To obtain the unknown coefficients of (6) we take the inner product of the three components of (3) with the functions $\tilde{\psi}_{\mu,\kappa}(t_1)\tilde{\psi}_{\mu,\nu}(t_2)$. The resulting equations can be written explicitly as follows:

$$\begin{aligned}
\int_\Sigma \int \int_\Sigma \int dv_1 dv_2 dt_1 dt_2 \quad & \mathbf{u}_i(t_1,t_2) \cdot \left[\bar{\bar{\mathbf{G}}}(t_1,t_2;v_1,v_2)\, h_1(v_1,v_2)\, h_2(v_1,v_2) \cdot \right. \\
\cdot \sum_{j=1}^{3} \mathbf{u}_j(v_1,v_2) \{ \sum_{m=m_l}^{\infty} \sum_{n,k=1}^{\infty} [\quad & \alpha_{n,k}^{(j,m)} \psi_{m,k}(v_1)\psi_{m,n}(v_2)\tilde{\psi}_{\mu,\kappa}(t_1)\tilde{\psi}_{\mu,\nu}(t_2) + \\
+ \quad & \beta_{n,k}^{(j,m)} \psi_{m,k}(v_1)\phi_{m,n}(v_2)\tilde{\psi}_{\mu,\kappa}(t_1)\tilde{\psi}_{\mu,\nu}(t_2) + \\
+ \quad & \gamma_{n,k}^{(j,m)} \phi_{m,k}(v_1)\psi_{m,n}(v_2)\tilde{\psi}_{\mu,\kappa}(t_1)\tilde{\psi}_{\mu,\nu}(t_2)] + \\
+ \sum_{n,k=1}^{\infty} \quad & s_{n,k}^{(j,m_l)} \phi_{m_l,k}(v_1)\phi_{m_l,n}(v_2)\tilde{\psi}_{\mu,\kappa}(t_1)\tilde{\psi}_{\mu,\nu}(t_2)\}\Big] = \\
= \quad & \int_\Sigma \int \mathbf{u}_i(t_1,t_2) \cdot \mathbf{g}(t_1,t_2)\, \tilde{\psi}_{\mu,\kappa}(t_1)\tilde{\psi}_{\mu,\nu}(t_2)\} dt_1 dt_2 \\
& \forall i \in \{1,2,3\}
\end{aligned} \tag{10}$$

Similarly, when (9) is used in conjunction with (4) we obtain $\forall i \in \{1,2,3\}$

$$\int_{\Sigma^c}\int dv_1 dv_2 \{ \sum_{m=m_l}^{\infty} \sum_{n,k=1}^{\infty} [\quad \alpha_{n,k}^{(i,m)} \psi_{m,k}(v_1)\psi_{m,n}(v_2)\tilde{\psi}_{\mu,\kappa}(v_1)\tilde{\psi}_{\mu,\nu}(v_2) +$$
$$+ \quad \beta_{n,k}^{(i,m)} \psi_{m,k}(v_1)\phi_{m,n}(v_2)\tilde{\psi}_{\mu,\kappa}(v_1)\tilde{\psi}_{\mu,\nu}(v_2) +$$
$$+ \quad \gamma_{n,k}^{(i,m)} \phi_{m,k}(v_1)\psi_{m,n}(v_2)\tilde{\psi}_{\mu,\kappa}(v_1)\tilde{\psi}_{\mu,\nu}(v_2)] +$$
$$+ \sum_{n,k=1}^{\infty} \quad s_{n,k}^{(i,m_l)} \phi_{m_l,k}(v_1)\phi_{m_l,n}(v_2)\tilde{\psi}_{\mu,\kappa}(v_1)\tilde{\psi}_{\mu,\nu}(v_2)\} =$$
$$= 0 \qquad (11)$$

where Σ^c is the complement of Σ. Equations (10) and (11) cast a linear system with unknowns $\alpha_{n,k}^{(i,m)}, \beta_{n,k}^{(i,m)}, \gamma_{n,k}^{(i,m)}, s_{n,k}^{(i,m)}$. Solution of this system yields the coefficients of the expansion (6), and hence the unknown function $\mathbf{f}$.

A few comments on the final system of equations are necessary. It is obvious that the problem is separable, in the sense that all the basis functions are essentially real functions of *one* real variable. It follows that all the wavelet properties that make them appealing in one-dimension also hold in our case. Thus, the impedance matrix of the 3D MM system is expected to be sparse [2]. Specifically, due to orthogonality and space localization, most elements in (11) are exactly equal to zero, or vanishingly small. Significant terms exist only in two cases: a) When $(m,k) = (\mu,\kappa)$ and $(m,n) = (\mu,\nu)$, i.e. on the diagonal of the impedance matrix; b) When a particular wavelet or scaling function is truncated by the boundary lines of Σ^c, i.e. when the orthogonality properties are destroyed. Furthermore, the elements of (10) may also be vanishingly small when very small length scales are involved. Suppose that $\tilde{\psi}_{\mu,\kappa}(v_1)$ oscillates in v_1 much more rapidly than the rest of the integrand. Let the rest of the integrand be $K(v_1)$. Then, if $\tilde{\psi}_{\mu,\kappa}(v_1)$ is not truncated by the end points, the v_1 integration yields

$$\int_a^b K(v_1)\tilde{\psi}_{\mu,\kappa}(v_1)dv_1 \simeq K(v_0)\int_a^b \tilde{\psi}_{\mu,\kappa}(v_1)dv_1 \simeq$$
$$\simeq \begin{cases} 0, \text{ if } \tilde{\psi}_{\mu,\kappa}(v_1) \text{ is a wavelet} \\ K(v_0), \text{ if } \tilde{\psi}_{\mu,\kappa}(v_1) \text{ is a scaling function} \end{cases} \qquad (12)$$

where v_0 is some point in the support of $\tilde{\psi}_{\mu,\kappa}(v_1)$ and $K(v_0)$ is a slowly

varying function. Hence, in the first case the element is vanishingly small, while in the second case it is trivial to compute (no integration is necassary).

When the surface is closed or semi-closed the formulation should be slightly modified. For those geometries the unknown is basically a periodic extension of the function $\mathbf{f}$ and instead of the regular wavelets and scaling functions, the following set of periodic functions [5]

$$\tilde{\psi}^{p}_{\mu,\kappa}(v) \equiv \sum_{l=-\infty}^{\infty} \tilde{\psi}_{\mu,\kappa}(v-l) \tag{13}$$

which is a complete, orthonormal basis of $\mathbf{L}^2[0,1]$ appears to be more appropriate. Although the dilation-invariance property of the original wavelet is not possessed by the periodic function defined in (13), the wavelets properties which render a sparse matrix are still valid. Therefore the MM solution for open surfaces can be directly extended to closed surfaces as well.

We remark in closing this section that the above results can be generalized to the integral equation of the kind

$$\int_{\Sigma}\int \bar{\bar{\mathbf{G}}}(\mathbf{r};\mathbf{r}')\cdot\mathbf{f}(\mathbf{r}')d^2S' + \alpha\mathbf{f}(\mathbf{r}) = \mathbf{g}(\mathbf{r}) \qquad \mathbf{r}\in\Sigma \tag{14}$$

$$\mathbf{f}(\mathbf{r}) = \mathbf{0}, \qquad \mathbf{r}\notin\Sigma \tag{15}$$

where α is an arbitrary complex constant.

2.2 Application

The approach presented in the previous section can be directly applied to the Electric Field (EFIE) and the Magnetic Field Integral Equations (MFIE). These take the form [6]:

$$\begin{aligned}\mathbf{E}(\mathbf{r}) &= \mathbf{E}^{i}(\mathbf{r}) + \int_{\Sigma}\int \{\left[\nabla\times\bar{\bar{\mathbf{\Gamma}}}(\mathbf{r},\mathbf{r}')\right]\cdot[\mathbf{E}(\mathbf{r}')\times\hat{\mathbf{n}}'] + \\ &+ jkZ\bar{\bar{\mathbf{\Gamma}}}(\mathbf{r},\mathbf{r}')\cdot[\hat{\mathbf{n}}'\times\mathbf{H}(\mathbf{r}')]\}d^2S \qquad (16)\\ \mathbf{H}(\mathbf{r}) &= \mathbf{H}^{i}(\mathbf{r}) + \int_{\Sigma}\int -\{\left[\nabla\times\bar{\bar{\mathbf{\Gamma}}}(\mathbf{r},\mathbf{r}')\right]\cdot[\hat{\mathbf{n}}'\times\mathbf{H}(\mathbf{r}')] + \\ &+ jkY\bar{\bar{\mathbf{\Gamma}}}(\mathbf{r},\mathbf{r}')\cdot[\mathbf{E}(\mathbf{r}')\times\hat{\mathbf{n}}']\}d^2S \qquad (17)\end{aligned}$$

where $\bar{\bar{\Gamma}}(\mathbf{r},\mathbf{r}')$ is the pertinent dyadic Green's function and Σ is assumed to be a closed surface. If Σ is perfectly conducting, the tangential electric field vanishes on Σ, therefore (16) and (17) reduce to

$$\mathbf{0} = \hat{\mathbf{n}} \times \mathbf{E}^i(\mathbf{r}) + jkZ\hat{\mathbf{n}} \times \int_\Sigma \int \bar{\bar{\Gamma}}(\mathbf{r},\mathbf{r}') \cdot [\hat{\mathbf{n}}' \times \mathbf{H}(\mathbf{r}')]\, d^2S \quad (18)$$

$$\hat{\mathbf{n}} \times \mathbf{H}(\mathbf{r}) = \hat{\mathbf{n}} \times \mathbf{H}^i(\mathbf{r}) - \hat{\mathbf{n}} \times \int_\Sigma \int \left[\nabla \times \bar{\bar{\Gamma}}(\mathbf{r},\mathbf{r}')\right] \cdot [\hat{\mathbf{n}}' \times \mathbf{H}(\mathbf{r}')]\, d^2S \quad (19)$$

$$\forall \mathbf{r} \in \Sigma$$

It is seen that (18), (19) have the form of (1) and (14) respectively, with the tangential magnetic field as the unknown.

Similarly, when the surface Σ is characterized by an impedance boundary condition of the form

$$\hat{\mathbf{n}} \times (\hat{\mathbf{n}} \times \mathbf{E}) = -\eta Z \hat{\mathbf{n}} \times \mathbf{H} \quad (20)$$

then (16), (17) reduce to

$$\begin{aligned} \hat{\mathbf{n}} \times \mathbf{E}(\mathbf{r}) &= \hat{\mathbf{n}} \times \mathbf{E}^i(\mathbf{r}) - \hat{\mathbf{n}} \times \int_\Sigma \int \{\left[\nabla \times \bar{\bar{\Gamma}}(\mathbf{r},\mathbf{r}')\right] \cdot [\hat{\mathbf{n}}' \times \mathbf{E}(\mathbf{r}')] + \\ &+ j\frac{k}{\eta}\bar{\bar{\Gamma}}(\mathbf{r},\mathbf{r}') \cdot [\hat{\mathbf{n}}' \times (\hat{\mathbf{n}}' \times \mathbf{E}(\mathbf{r}'))]\} d^2S \quad (21) \\ \hat{\mathbf{n}} \times \mathbf{H}(\mathbf{r}) &= \hat{\mathbf{n}} \times \mathbf{H}^i(\mathbf{r}) - \hat{\mathbf{n}}' \times \int_\Sigma \int \{\left[\nabla \times \bar{\bar{\Gamma}}(\mathbf{r},\mathbf{r}')\right] \cdot [\hat{\mathbf{n}}' \times \mathbf{H}(\mathbf{r}')] + \\ &+ jk\eta\bar{\bar{\Gamma}}(\mathbf{r},\mathbf{r}') \cdot [\hat{\mathbf{n}} \times (\hat{\mathbf{n}} \times \mathbf{H}(\mathbf{r}'))]\} d^2S \quad (22) \end{aligned}$$

$$\forall \mathbf{r} \in \Sigma$$

which are of the same form as (14).

3 The Adaptive Integral Method (AIM)

It has already been mentioned in the previous sections that the conventional MM is not practical for very large geometries, due to excessive storage and

CPU requirements. Many techniques have been proposed for improving the MM efficiency. Among the most promising ones is the recently developed Adaptive Integral Method (AIM) [7]. The main idea of this method is to replace the full MM impedance matrix with a sum of matrices having special properties that reduce the computational complexity and memory requirements. This improvement can be achieved by approximating the interaction among far away current sources with delta current sources located at the nodes of a rectangular grid. It can be shown that the original impedance matrix reduces to a sum of two terms: the first is a sparse matrix and the second one is a product of sparse and Toeplitz matrices.

To illustrate the mathematical details, let us consider the integral equation

$$\int_D d\mathbf{r}' K(\mathbf{r}-\mathbf{r}')X(\mathbf{r}') = Y(\mathbf{r}) \tag{23}$$

where X is the unknown vector function, K is a kernel and D is a domain of integration with arbitrary dimensionality. To solve this integral equation via the MM we expand X and Y as

$$X(\mathbf{r}) = \sum_{\alpha=1}^{\infty} X_\alpha \psi_\alpha(\mathbf{r}) \tag{24}$$

$$Y(\mathbf{r}) = \sum_{\alpha=1}^{\infty} Y_\alpha \psi_\alpha(\mathbf{r}) \tag{25}$$

The basis functions $\psi_\alpha(\mathbf{r})$ are usually of the edge-based type and are associated with edge α. On substituting (24), (25) into (23), the original integral equation is transformed to the linear algebraic system

$$\sum_{\beta=1}^{\infty} K_{\alpha\beta} X_\beta = Y_\alpha \tag{26}$$

where

$$K_{\alpha\beta} = \int_D \int_D d\mathbf{r} d\mathbf{r}' \psi_\alpha(\mathbf{r}) K(\mathbf{r}-\mathbf{r}') \psi_\beta(\mathbf{r}') \tag{27}$$

The basic idea of AIM is to use approximate expressions for $K_{\alpha\beta}$ when ψ_α and ψ_β are basis functions corresponding to edge elements separated by large distances. To facilitate this approximation, we write (27) in the form

$$K_{\alpha\beta} = \begin{cases} \int\int d\mathbf{r}d\mathbf{r}' \psi_\alpha K \psi_\beta & \text{if } |\mathbf{r}_\alpha - \mathbf{r}_\beta| < R \\ \int\int d\mathbf{r}d\mathbf{r}' \tilde{\psi}_\alpha K \tilde{\psi}_\beta & \text{if } |\mathbf{r}_\alpha - \mathbf{r}_\beta| \geq R \end{cases} \tag{28}$$

where ψ_α are conventional basis functions and $\tilde{\psi}_\alpha$ are auxiliary basis functions defined by

$$\tilde{\psi}_\alpha(\mathbf{r}) = \sum_{l=1}^{Q} \sum_{m=1}^{Q} \sum_{n=1}^{Q} \Lambda_{\alpha,lmn} \delta^3(\mathbf{r} - \mathbf{r}_{lmn}) \tag{29}$$

In the latter equation, $\delta^3(\mathbf{r} - \mathbf{r}_{lmn})$ is a 3-dimensional delta function, $\mathbf{r}_{lmn}$ are points on a rectangular grid surrounding the original sample points and Q is an arbitrary positive integer, equal to the desired order of approximation. The coefficients $\Lambda_{\alpha,lmn}$ are found by imposing equality of the moments up to order Q of $\tilde{\psi}_\alpha$ and ψ_α with respect to the center of edge α. These moments are defined by

$$M^{\psi}_{q_1,q_2,q_3} \equiv \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} d\mathbf{r}(x - x_\alpha)^{q_1}(y - y_\alpha)^{q_2}(z - z_\alpha)^{q_3}\psi_\alpha(\mathbf{r}) \tag{30}$$

$$M^{\tilde{\psi}}_{q_1,q_2,q_3} \equiv \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} d\mathbf{r}(x - x_\alpha)^{q_1}(y - y_\alpha)^{q_2}(z - z_\alpha)^{q_3}\tilde{\psi}_\alpha(\mathbf{r}) \tag{31}$$

and on using the definition of the delta functions, equality of the moments yields the following linear algebraic system for the coefficients $\Lambda_{\alpha,lmn}$

$$\sum_{l=1}^{Q} \sum_{m=1}^{Q} \sum_{n=1}^{Q} \Lambda_{\alpha,lmn}(x_l - x_\alpha)^{q_1}(y_m - y_\alpha)^{q_2}(z_n - z_\alpha)^{q_3} = M^{\psi}_{q_1,q_2,q_3} \tag{32}$$

The explicit expressions for the elements $K_{\alpha\beta}$ are finally

$$K_{\alpha\beta} = K^{far}_{\alpha\beta} \text{ if } |\mathbf{r}_\alpha - \mathbf{r}_\beta| \geq R \tag{33}$$

$$K_{\alpha\beta} = K^{far}_{\alpha\beta} + K^{near}_{\alpha\beta} \text{ if } |\mathbf{r}_\alpha - \mathbf{r}_\beta| < R \tag{34}$$

where

$$K^{far}_{\alpha\beta} = \sum_{l',m',n'} \sum_{l,m,n} \Lambda_{\alpha,lmn} K(\mathbf{r}_{lmn} - \mathbf{r}_{l'm'n'}) \Lambda_{\beta,l'm'n'}$$

$$K^{near}_{\alpha\beta} = < \psi_\alpha, K\psi_\beta > - \sum_{l',m',n'} \sum_{l,m,n} \Lambda_{\alpha,lmn} K(\mathbf{r}_{lmn} - \mathbf{r}_{l'm'n'}) \Lambda_{\beta,l'm'n'}$$

The triangular brackets stand for the integral

$$< f(\mathbf{r}), Kg(\mathbf{r}) > \equiv \int_D \int_D d\mathbf{r}d\mathbf{r}' f(\mathbf{r}) K(\mathbf{r}-\mathbf{r}') g(\mathbf{r}') \tag{35}$$

In the final formulation $[K_{\alpha\beta}^{near}]$ is obviously sparse, since only nearby current elements interact with each other. Moreover, $[K_{\alpha\beta}^{far}]$ is a product of the sparse matrices $[\Lambda]$, $[\Lambda]^T$ by a Toeplitz matrix with elements $K(\mathbf{r}_{lmn} - \mathbf{r}_{l'm'n'})$. Sparsity implies that the system can be efficiently solved by an iterative method, such as the Conjugate Gradient (CG) technique. Furthermore, the convolutional character of the products involving the Toeplitz matrix implies that a Fast Fourier Transform can be employed for the calculation of the matrix-vector products involved in the CG method. Therefore, the computational complexity drops from $O(N^2)$ (required for a full matrix inversion) to $O(N\log N)$ for volume problems and to $O(N^{3/2}\log N^{3/2})$ for surface problems, required by the combined CG and FFT technique (CG-FFT). Moreover, storage drops to $O(N)$ and $O(N^{3/2})$ respectively.

We have already mentioned that the very large electrical size of actual jet engine inlets makes the application of standard MM impossible, due to excessive storage and CPU requirements. Nevertheless, AIM drastically enhances the MM efficiency by exploiting the properties of sparse or Toeplitz matrices. Therefore it is expected that AIM is capable of solving very large geometries. At this moment, though, it is difficult to obtain a priori quantitative estimates of its capabilities and limitations in solving large scale problems.

4 Memory and CPU time requirements

Although is is very difficult to estimate the number of unknowns N or the number of iterations that is necessary for the solution of particular problems via the aforementioned methods, it is possible to estimate the memory storage and the number of operations for given N. For surface problems the number of operations per iteration for AIM is [8]

$$N_{op} \simeq 32(m+1)^3 N + 88 N_c (6 + \frac{5}{3}\log_2 N_c) \tag{36}$$

and the number of real variables that must be stored is

$$N_{st} = 7(m+1)^3 N + 192 N_c \tag{37}$$

where m is the order of the multipole expansion and $N_c \simeq (N/12)^{3/2}$. For the standard MM the relevant numbers are $N_{op} = 2N^2$ and $N_{st} = N^2$. Plots for the operation count and memory requirements for standard MM and AIM with $m = 2$ are given in Figs. 1 and 2. It is evident that AIM is far more efficient than standard MM for large number of unknowns.

If we define a "sparsity parameter" $a \in [0, 1]$ such that $a = 1$ for an empty matrix and $a = 0$ for a full matrix, then the operation count for a sparse matrix becomes $N_{op} \simeq 2(1-a)N^2$ and the storage becomes $N_{st} = (1-a)N^2$. It is obvious that a wavelet MM may become more efficient than AIM only if a is an increasing function of N. In Fig. 3 the minimal value of a is plotted so that a wavelet MM method is superior to AIM with respect to memory and CPU time.

Figs. 1, 2 and 3 demonstrate the relative efficiency of standard MM, AIM and wavelet MM. Nevertheless, it should be pointed out that the plots have been drawn for a given number of iterations and assuming that the same number of unknowns N is used for all methods, which is not true, in general. A wavelet expansion method may require a much larger number of unknowns for a particular problem than AIM, therefore any apparent advantage may be only superficial. Furthermore, it is impossible to estimate a priori the sparsity parameter that a wavelet expansion can achieve, or even the number of iterations that are necessary for satisfactory convergence of each method. Therefore, the efficiency of each technique can be quantified only in terms of unknown factors. Only actual implementation of the methods will show which one is more suitable for what problem.

To close this section, the advantages and disadvantages of each method are highlighted:

AIM:

+ Easy to implement
+ Grid adaptable to arbitrary geometries
+ Sparsity guaranteed
- Performance has not been tested within inlets

Wavelets/MM:

+ Rigorous, guarantees convergence
+ For high sparsity, CPU time and storage are very low

- Very complicated
- Difficult to apply to arbitrary, non-canonical geometries
- Sparsity level not predictable a priori

5 Future Plans

Our intention is to undertake the following tasks, towards the completion of the Ph. D. thesis:

• Develop an experimental AIM code for a simple 3D geometry with a known solution, to familiarize with the details of the method.

• Develop an AIM code for the geometries that have been analyzed via Mode Matching.

• Extend the analysis for coated blades, multiple blade stages and arbitrary blade shapes.

• Repeat the above three using wavelets as basis functions in the context of a MM analysis.

The proposed work will have an impact on reducing the computational requirements of large scale EM simulations. The prospects of AIM and wavelets are promising but their extension to 3D has yet to be considered. Consequently, a critical look at the suitability and computational requirements of these new approaches is appropriate. The engine simulation has computational requirements that are equal to or greater than that of the airframe and is therefore a good candidate for examining the performance of the techniques.

6 Conclusion

In this report we proposed some efficient methods for the RCS computation of jet engine inlets. The complexity of realistic engines has led us to consider flexible numerical techniques that can handle irregular shapes and dielectric material coatings. The MM is generally considered very reliable, but computationally expensive. In order to decrease the memory requirements, we considered the use of wavelet basis functions, which have proven to yield sparse matrices in one and two-dimensional problems. We claimed (and justified our claims) that even in three-dimensional problems sparse impedance

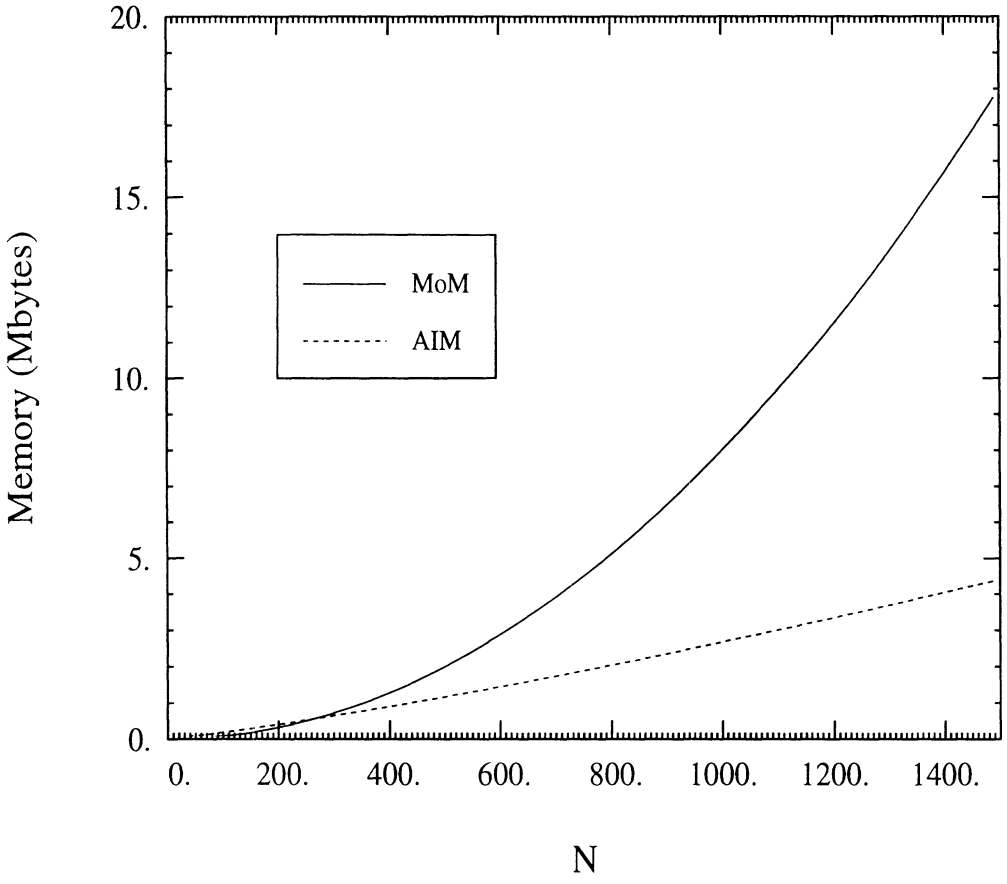


Figure 1: Memory required for standard MM and AIM (surface problems).

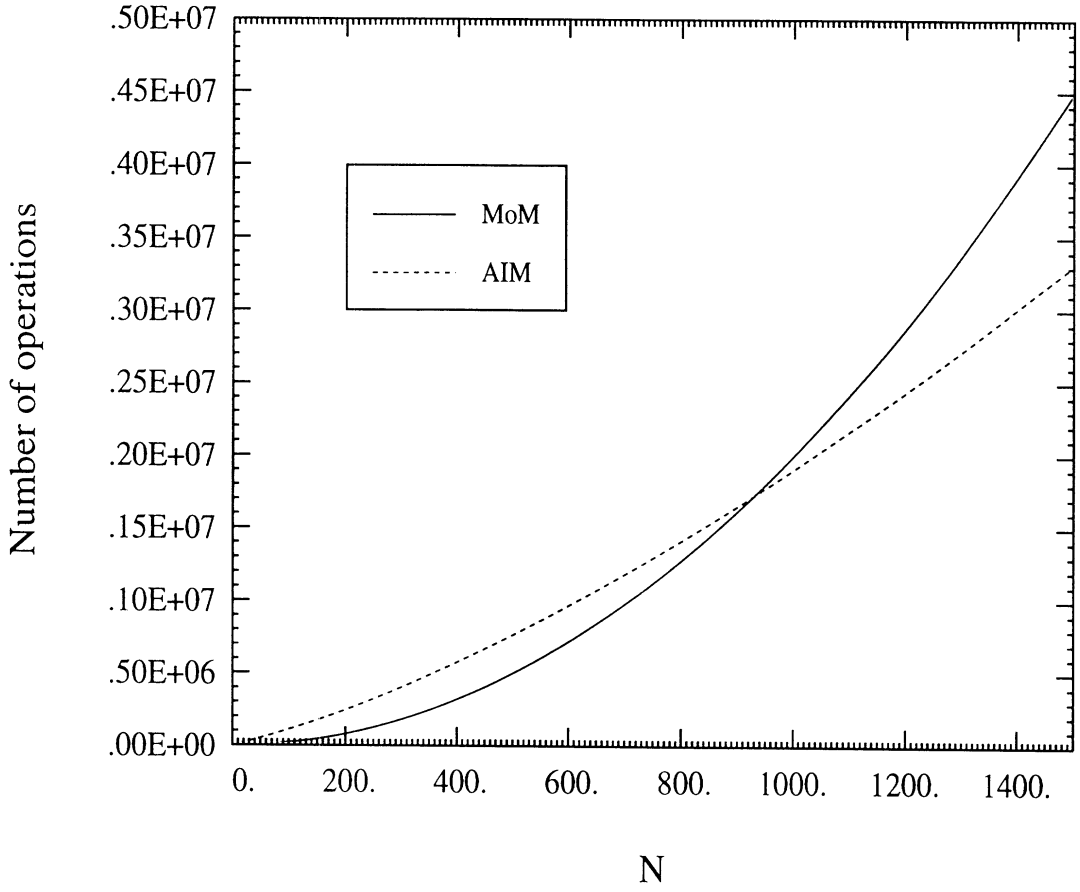


Figure 2: Number of operations per iteration (surface problems).

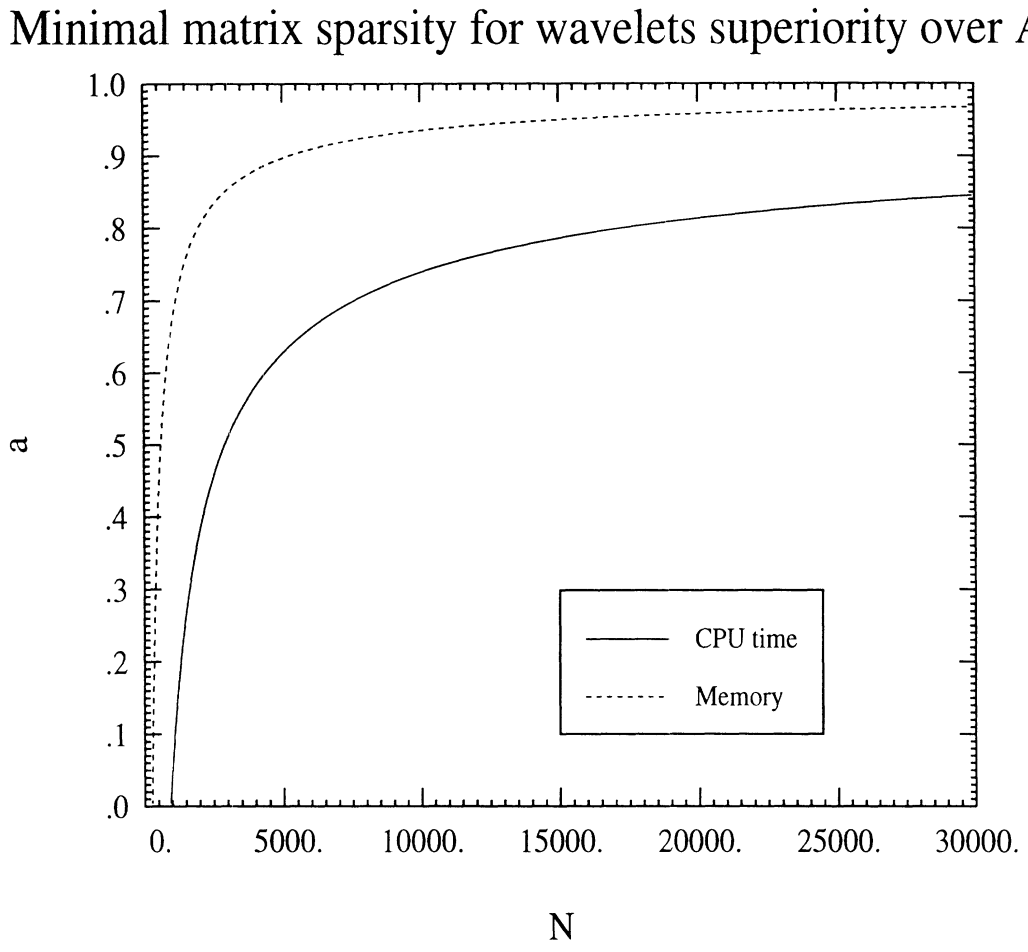


Figure 3: Minimal sparsity for wavelet MM superiotity over AIM.

matrices are expected to appear in the calculations. We also presented the basics of a recently developed numerical technique (AIM), which drastically reduces the CPU time and storage requirements of the conventional MM. Therefore, we are in a position to propose the analysis of scattering from jet-engine inlets via two different versions of the moment method: a) using multidimensional wavelets as basis functions and b) taking advantage of the simple structure of the impedance matrix achieved by AIM.

References

[1] D. C. Ross, J. L. Volakis and H. T. Anastassiu, "Hybrid Finite Element-Modal analysis of jet engine inlet scattering", *IEEE Trans. Antennas and Propagation*, vol. 43, no 3, pp. 277-285, March 1995.

[2] B. Z. Steinberg and Y. Leviatan, "On the use of wavelet expansions in the method of moments," *IEEE Trans. Antennas and Propagation*, vol. 41, no 5, pp. 610-619, May 1993.

[3] K. Sabetfakhri and L. P. B. Katehi, "Analysis of integrated millimeter-wave and submillimeter-wave waveguides using orthonormal wavelet expansions," *IEEE Trans. on Microwave Theory and Techniques*, vol. 42, no 12, pp. 2412-2422, December 1994.

[4] G. Wang and G. W. Pan, "Full wave analysis of microstrip floating line structures by wavelet expansion method," *IEEE Trans. on Microwave Theory and Techniques*, vol. 43, no 1, pp. 131-142, January 1995.

[5] B. Z. Steinberg and Y. Leviatan, "Periodic wavelet expansions for analysis of scattering from metallic cylinders," *IEEE-APS International Symposium Digest*, vol. 1, pp. 20-23, June 1994.

[6] J. L. Volakis, EECS 633 Coursepack, Fall 1993.

[7] E. Bleszynski, M. Bleszynski and T. Jaroszewicz, "A fast integral equation solver for electromagnetic scattering problems," *IEEE Int. Conf. AP-S Digest*, pp. 416-419, Seattle WA, 1994.

[8] E. Bleszynski, personal communication.