389757-1-T

THE UNIVERSITY OF MICHIGAN

COLLEGE OF ENGINEERING

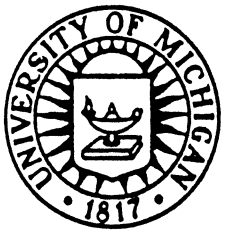
DEPARTMENT OF ELECTRICAL ENGINEERING & COMPUTER SCIENCE

Radiation Laboratory

VECTOR AND PARALLEL IMPLEMENTATION OF THE CONJUGATE GRADIENT FFT METHOD FOR THE SOLUTION OF LARGE ELECTROMAGNETIC SCATTERING PROBLEMS ON SUPERCOMPUTERS

Kasra Barkeshli

Radiation Laboratory
Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, Michigan 48109-2122



March 14, 1989

Ann Arbor, Michigan

Abstract

There is a growing interest in the application of the Conjugate Gradient FFT (CGFFT) method to electromagnetic radiation and scattering problems primarily because of the low storage requirement associated with this method.The CGFFT lends itself well to efficient execution in vectorized fashion and in this study, a vector-concurrent form of the CGFFT method suitable for implementation on parallel, multiprocessor supercomputer systems is applied to the problem of scattering from electrically large planar structures. The vectorizable nature of the CGFFT algorithm is exploited by identifying the major processes involved in a given iteration including the Fast Fourier Transforms, vector dot products and norm calculations. Since for a given iteration there is no data recurrence in the operations involved, these processes may be vectorized to increase the computation speed.

JMR Ø220

Contents

List of Figures

List of Tables

1 Introduction

Computational electromagnetics relies heavily on vector-oriented algorithms to simulate complex problems. With the computer technology approaching the limits of semiconductor speeds, the exploitation of parallel processing has emerged in order to meet the processing demands of computationally intensive applications in electromagnetics. In such applications machine instructions are vectorized and distributed across different vector processors for concurrent execution as opposed to the traditional approach where the computers are limited to sequential processing of data on a single scalar processing unit.

In this report, an example of vector and parallel processing as applied to an iterative technique, namely, the conjugate gradient method [1,2,3] is explored. In general, iterative techniques become more attractive for the solution of operator equations arising in electromagnetic problems as the size of the problem increases. This is mainly because the iterative schemes often involve only the multiplication of the matrices with vectors and thus do not require an explicit storage of the system matrix. Also, since these methods avoid the process of matrix inversion, they are numerically more stable for ill-conditioned systems.

In electromagnetic scattering, the pertaining integral operators are usually formulated as convolutions over the unknown current density. In these situations, the Fast Fourier Transform can be used to evaluate the convolution integrals efficiently. When combined with FFT, the conjugate gradient method requires roughly $4N(1+\log_2 N)$ operations per iteration. The required number of iterations before the conjugate gradient method can yield a reasonable accuracy is often a fraction

of the total number of unknowns. This depends primarily on the distribution of the dominant eigenvalues of the operator projected onto the system matrix.

The CGFFT lends itself to efficient execution in vectorized fashion and in this report, a vector-concurrent form of the CGFFT method suitable for implementation on parallel, multiprocessor systems is applied to the problem of scattering from electrically large planar structures. Optimized algorithms are implemented on IBM 3090-600E supercomputer and on an Alliant FX/8 mini-supercomputer with both vector and vector-concurrent capabilities.

Recently [4], it was shown that the convergence of the CGFFT may be improved by the incorporation of subdomain basis functions in the expansion of the unknown current distributions. The improved convergence was attributed to a more accurate representation of the current distribution in the spectral domain. Implementation of this formulation in conjunction with the optimized algorithm will result in a drastic improvement in the efficiency of the method. Numerical results are presented to demonstrate the corresponding improvements in the processing speed.

2 The Conjugate Gradient FFT Method

The integral equations arising in the study of electromagnetic scattering problems can be written in a general operator form as (Fig. 1)

$$A[\mathbf{J}] = \mathbf{E}^{\mathrm{i}} \tag{1}$$

where $\mathbf{E}^{\mathrm{i}}$ denotes the excitation field and $\mathbf{J}$ is the unknown current density vector. A typical form of such integral equations is

$$\mathbf{E}^{\mathrm{i}}(\mathbf{r}) = \bar{\bar{\eta}}(\mathbf{r}) \cdot \mathbf{J}(\mathbf{r}) + \int_{v'} \bar{\Gamma}(|\mathbf{r} - \mathbf{r}'|) \cdot \mathbf{J}(\mathbf{r}')dv', \tag{2}$$

where $\bar{\Gamma}$ is the associated dyadic Green's function and $\bar{\bar{\eta}}$ is a tensor specific to the electrical properties of the scatterer.

The integral in (2) implies a convolution over the geometrical limits of the target. Thus, introducing the forward and inverse Fourier transform pair[1]

$$\tilde{g} \Longleftrightarrow g = \mathcal{F}^{-1}\{\tilde{g}\}, \tag{3}$$

equation (2) can be alternatively written as [4]

$$\mathbf{E}^{\mathrm{i}} = \bar{\bar{\eta}} \cdot \mathbf{J} + \mathcal{F}^{-1}\{\tilde{\bar{\bar{\Gamma}}} \cdot \tilde{f}\hat{\mathbf{J}}\} = A[\mathbf{J}] \cdot \tag{4}$$

In the above, $\tilde{f}$ is the Fourier transform of the basis function used in the expansion of the current density and $\hat{\mathbf{J}}$ is the discrete Fourier transform of its sampled train so that

$$\tilde{\mathbf{J}} = \tilde{f}\hat{\mathbf{J}} \cdot \tag{5}$$

Clearly, this formulation avoids the generation of the square matrix corresponding to operator A and thus implies a storage requirement of $\mathcal{O}(N)$ as compared to

[1] See Appendix I for the definitions of the Fourier transform relations used.

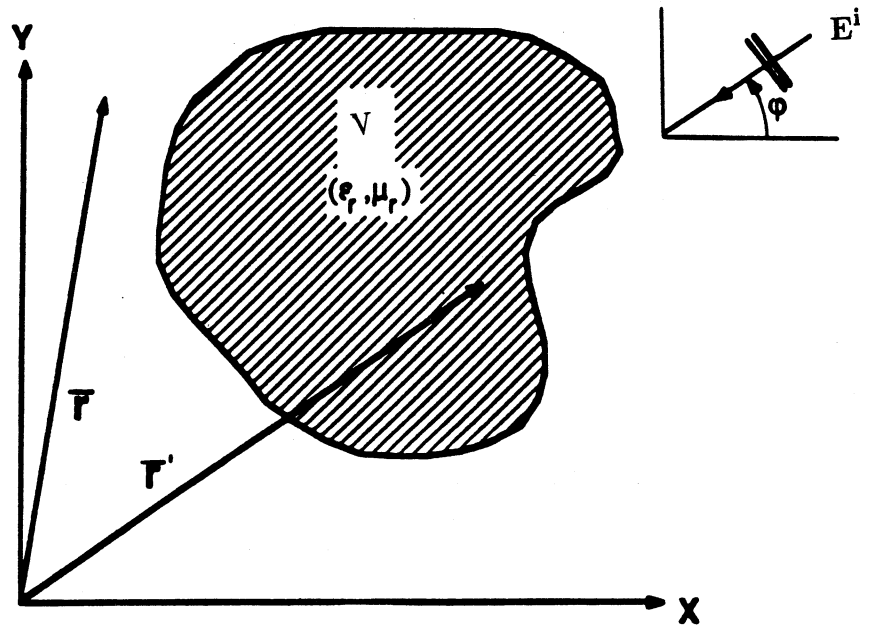


Fig. 1 Plane wave scattering from an arbitrary composite material target.

$\mathcal{O}(N^2)$ required for an implementation of the corresponding direct method. This storage economy has made the formulation suitable for large radiating systems. Several iterative methods for the solution of (4) are available. In the conjugate gradient method, the unknown current density is calculated by minimizing a functional through a finite number of successive searches in an iterative fashion. A conjugate gradient algorithm suitable for numerical computation is [2,3]

$$R_0 = A[J_0] - E^i$$

$$P_0 = -b_{-1}A^a[R_0]$$

Main Iteration Loop

For $n \geq 0$ Do

$$t_n = \frac{1}{\|A[P_n]\|^2}$$

$$J_{n+1} = J_n + t_n P_n$$

$$R_{n+1} = R_n + t_n A[P_n] \tag{6}$$

$$b_n = \frac{1}{\|A^a[R_n]\|^2}$$

$$P_{n+1} = P_n - b_n A^a[R_{n+1}]$$

$$\frac{|R_{n+1}|}{\|E^i\|} \stackrel{?}{\leq} tolerance$$

Repeat If Necessary.

In the above, R is the residual vector (the functional to be minimized) and P is the search direction. The norm and the adjoint operator are defined in terms of an inner product as

$$|g|^2 = < g, g > \tag{7}$$

and

$$< A[g], f >=< g, A^a[f] > \cdot \tag{8}$$

It may be shown that[5]

$$A^a[\mathbf{J}] = \bar{\bar{\eta}}^*(\mathbf{r}) \cdot \mathbf{J}(\mathbf{r}) + \int_{v'} \bar{\bar{\Gamma}}^*(|\mathbf{r} - \mathbf{r}'|) \cdot \mathbf{J}(\mathbf{r}')dv', \tag{9}$$

where $*$ denotes the complex conjugate.

Each iteration in (6) requires two convolution operations, two norm calculations and three scalar products. The iterations are repeated until a preset tolerance on the residual is met.

3 Optimization of the CGFFT Algorithm

Before an assessment of the vectorizability of the CGFFT algorithm, a brief review of some general concepts in vector and parallel processing will be presented. This discussion is followed by an overview of the computer systems used in this study.

3.1 Vectorization

The crux of parallel computing is the process of vectorization and distribution of code among multiple processors. Typically, a vector instruction is capable of operating on 32 to 128 elements of data at once, depending on the machine used, resulting in two to four times gain in speed over the corresponding sequential scalar instruction. Vectorization requires some degree of independence in the access of data by the code. A dependence occurs when two statements—or iterations of the same statement—refer to the same storage location. Some data dependences *inhibit* vectorization; they are called *recurrence.* By changing the structure of the code, it may be possible to eliminate a recurrence and vectorize the modified code.

As an example of a vectorizable code, consider the following simple loop

```
DO I=1,N
   C(I)=A(I)+B(I)
END DO.
```

In scalar mode, registers for scalar arithmetic hold only one element at a time. To add two vectors A and B, each element in the vector B has to be added individually to the appropriate element in the vector A, and then assigned to the appropriate

element in vector C. The sequence of machine instructions generated for such a scalar computation would be

1. Load counter N < 1 >
2. Load A(I) into scalar register through the loop < 1 >
3. Add B(I) to the scalar register < 3 >
4. Store C(I) from the scalar register < 1 >
5. Decrement counter by 1 and repeat < 1 >.

The time cycles required for each operation are shown in the brackets. Since in this particular case, the calculations in a given value of the Do loop index do not depend on a previous value of the index, there is no data recurrence in the loop and the operation may be carried out in vector mode as follows

1. Load counter N < 1 >
2. Load A(1)-A(128) into vector register $< 1 \times 128 >$
3. Add B(1)-B(128) to the vector register $< 3 + 127 >$
4. Store C(1)-C(128) from the vector register $< 1 \times 128 >$
5. Decrement counter by 128 and repeat < 1 >.

When the instruction to add vectors is issued, the first elements of the vectors, A(1) and B(1), are transmitted to the add functional unit at time t_0. Since the functional unit time for an addition is three clock periods, the first result $(A(1) + B(1))$ will

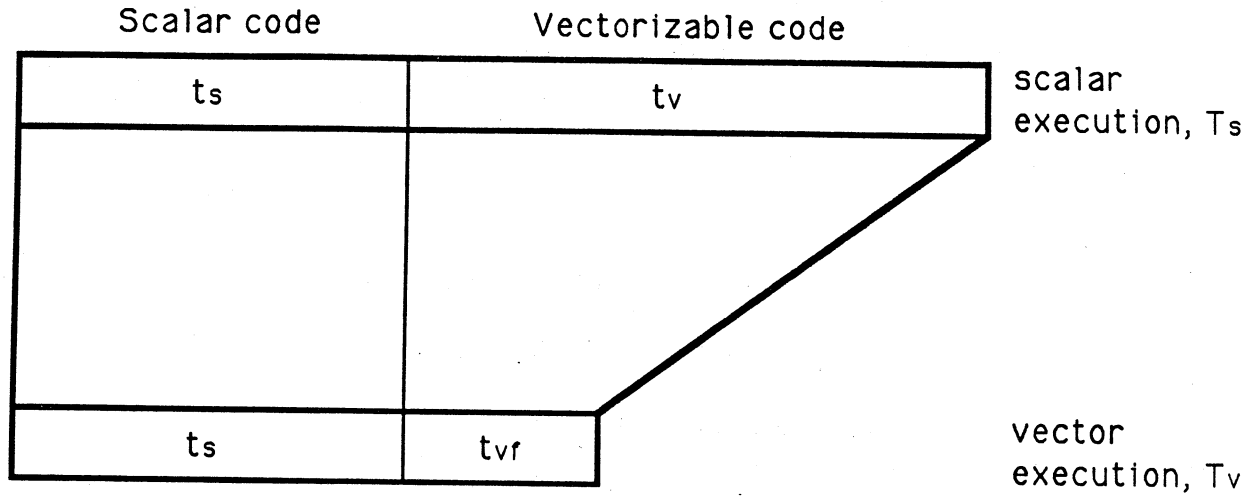
be produced at t_2. It is important to note that while elements A(1) and B(1) were entering the functional unit, elements A(2) and B(2) were being transmitted to it. Due to full segmentation, the result of $A(2) + B(2)$ is produced at t_3. From this point on, the process of one pair of vector register elements entering the functional unit and a result being produced every clock cycle continues until all elements of the vectors are added. So the addition of the first 128 pair of elements is accomplished in $127 + 3 = 130$ time cycles as opposed to the corresponding scalar processing time of $128 \times 3 = 384$ cycles. This process is called *pipelining* and is the major underlying process of optimization. If in the above example N=256, the scalar time would be 1537 clock cycles while a vector implementation would yield 775 cycles.

In order to measure the improvement in the speed of a vectorized program, several parameters are defined. The *program speedup* is defined as the increase in the speed of execution when a code is run in vector mode relative to that in the scalar mode. Therefore, referring to figure 2, if a code runs for T_s seconds in scalar mode and for T_v seconds in vector mode (i.e. after optimization), the corresponding program speedup is given by

$$\text{program speedup} = T_s/T_v.$$

Also, the *vector content* of a program is that percent of the scalar code which vectorizes. Thus, if for a given code, the scalar portion which may not be vectorized runs in t_s seconds and that which is vectorizable runs in t_v seconds in scalar mode and in t_{vf} seconds in vector mode[2], respectively, we have

[2] t_{vf} would be the time the code actually spends in the vector facilities.



Vector Content= tv/ts

Vector Speedup= tv/tvf

Program Speedup= Ts/Tv

Fig. 2 Scalar and vector execution times in a typical vectorized code.

$$\text{vector content} = t_v/T_s \times 100$$

and

$$\text{vector speedup} = t_v/t_{vf}.$$

Typically, programs with more than about 70% vector content, run 1.5 to 2 times faster in vector mode. Higher vector contents, however, are achievable as will be shown to be the case for the CGFFT algorithm.

3.2 Concurrency

Although it is increasingly expensive to make a single processor faster, fairly fast processors are inexpensive. Therefore using several relatively inexpensive processors in parallel is often more efficient than using a single fast processor. A multiple processor system can devote several processors to the execution of different parts of a single program simultaneously. The compiler inserts protective synchronization code into the optimized loops so that the multiple processors work together without interfering with each other. Synchronization is needed to prevent conflicts in the use of memory shared by parallel tasks and is considered the major cost of parallel processing. Optimization is suppressed whenever the possibility of a data dependence exists. Also, from Amdhal's law the increase in the speed of execution in concurrent mode as the number of processors is increased, approaches an upper limit set by the presence of the sequential constructs in the algorithm.

In general, the concepts of data dependence in parallel processing are the same as those in vector processing; the consequences of certain dependences, however, are different.

A parameter of interest when processing in concurrent mode is the *efficiency* of execution. Efficiency is a measure of parallelism in the algorithm. An efficient parallel tasking system makes possible a nearly linear speedup in performance as processors are added, if the algorithm is parallel. Thus, if $T^{(n)}$ denotes the time required to run on n processors, we may define

$$\text{concurrency speedup} = T^{(1)}/T^{(n)}$$

and

$$\text{efficiency}\% = T^{(n)}/(nT^{(1)}) \times 100.$$

3.3 Compiler Directives

Once a well written code is compiled, most vector and parallel constructs are recognized by the compiler and automatically optimized for efficient execution. In addition to automatic optimization, however, most vector compilers provide directives for additional control. Compiler directives are user supplied control structures to override decisions made by the compiler and to give additional information to it. Upon compilation, the directives are interpreted by the processor and converted to library calls to be executed in a more efficient manner. In particular, the associative transformation directive recognizes operations like dot products and norm computations as reduction functions and optimizes these otherwise non-vectorizable loops.

Some general guidelines for code optimization in various architectural levels are given in Appendix II and have been followed in optimizing the CGFFT algorithm in the present study.

3.4 The IBM 3090 supercomputer

The supercomputer used in this study was an IBM 3090-600E with six processors, each with a vector facility and 64 kilobytes cache memory. An overview of the system is given in figure 3. It has 256 megabytes of real memory with an expanded storage of 512 megabytes and utilizes VM/XA operating system. The cycle time is 17.2 ns delivering a theoretical peak performance of 116 MFLOPS (mega floating point operations per second)[3]. The vector facility provides 16 vector registers of 32 bit words (or 8 vector registers of 64 bit words) operating on up to 128 data elements.

The operating system of the IBM 3090 provides detailed timing information on the execution of the code. The total CPU time and the time fraction spent in the vector facility when program is run in vector mode are reported upon execution. This information can be used to estimate the vector content of the code. The overall vector performance of the IBM 3090 is depicted in figure 4 where the program and vector speedups are given for various vector contents.

3.5 The Alliant FX/8 Mini-supercomputer

The Alliant FX/8 is a high performance system with both parallel and vector computing capabilities. The FX/8 runs the Concentrix operating system which conforms to the external specifications of UNIX and fully supports Transmission Control Protocol/Internet Protocol (TCP/IP) and Network File System (NFS). The Alliant used in this study has four vector processors (expandable to 8), 56 Mbytes of main memory, 256 kbytes of cache memory and a bus which moves 188

[3]The Cray X-MP has a cycle time of 8.5 ns.

IBM 3090-600 Processor Unit Design

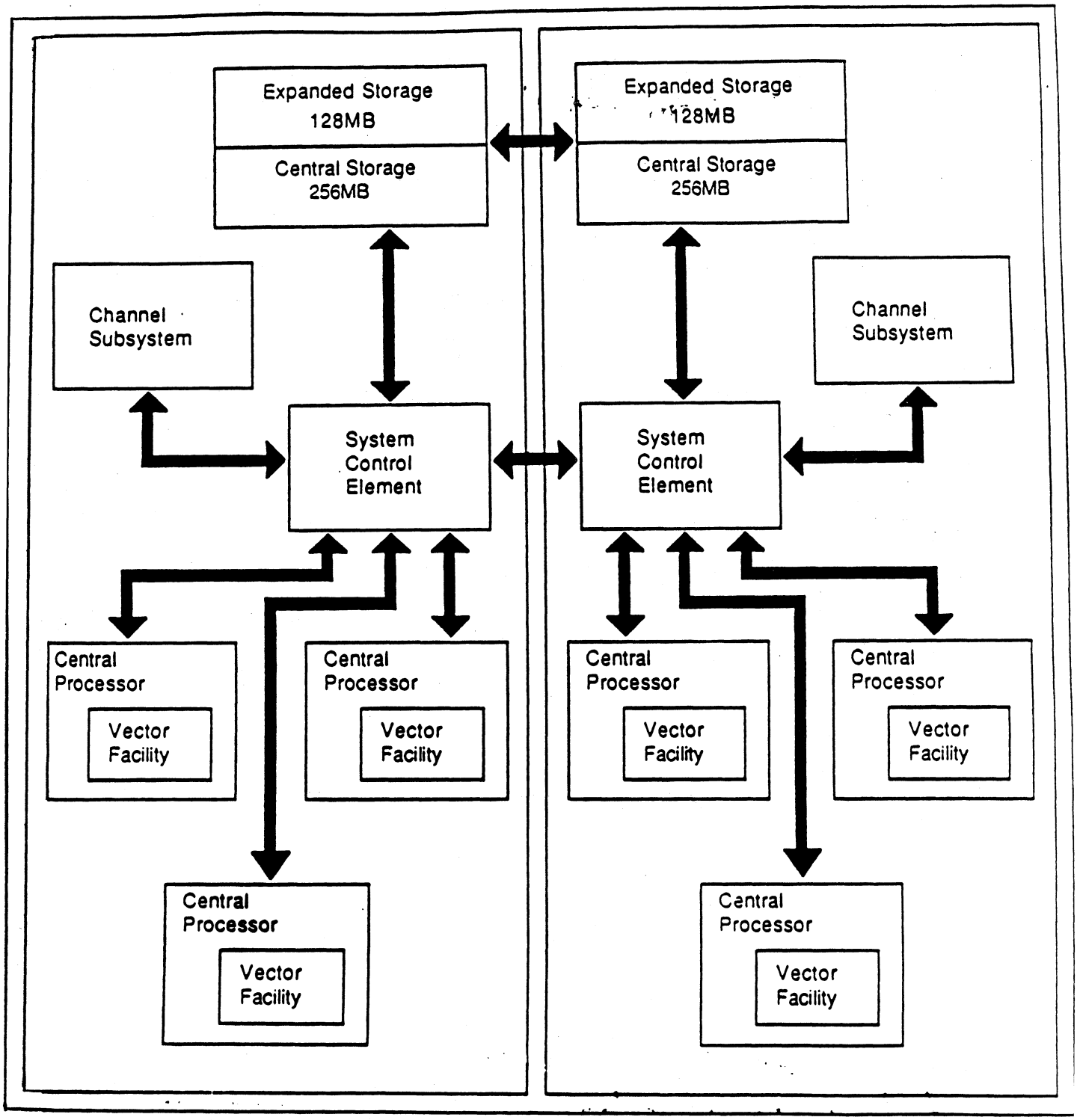


Fig. 3(a) An overview of the IBM 3090-600E supercomputer.

IBM 3090 Central Processor

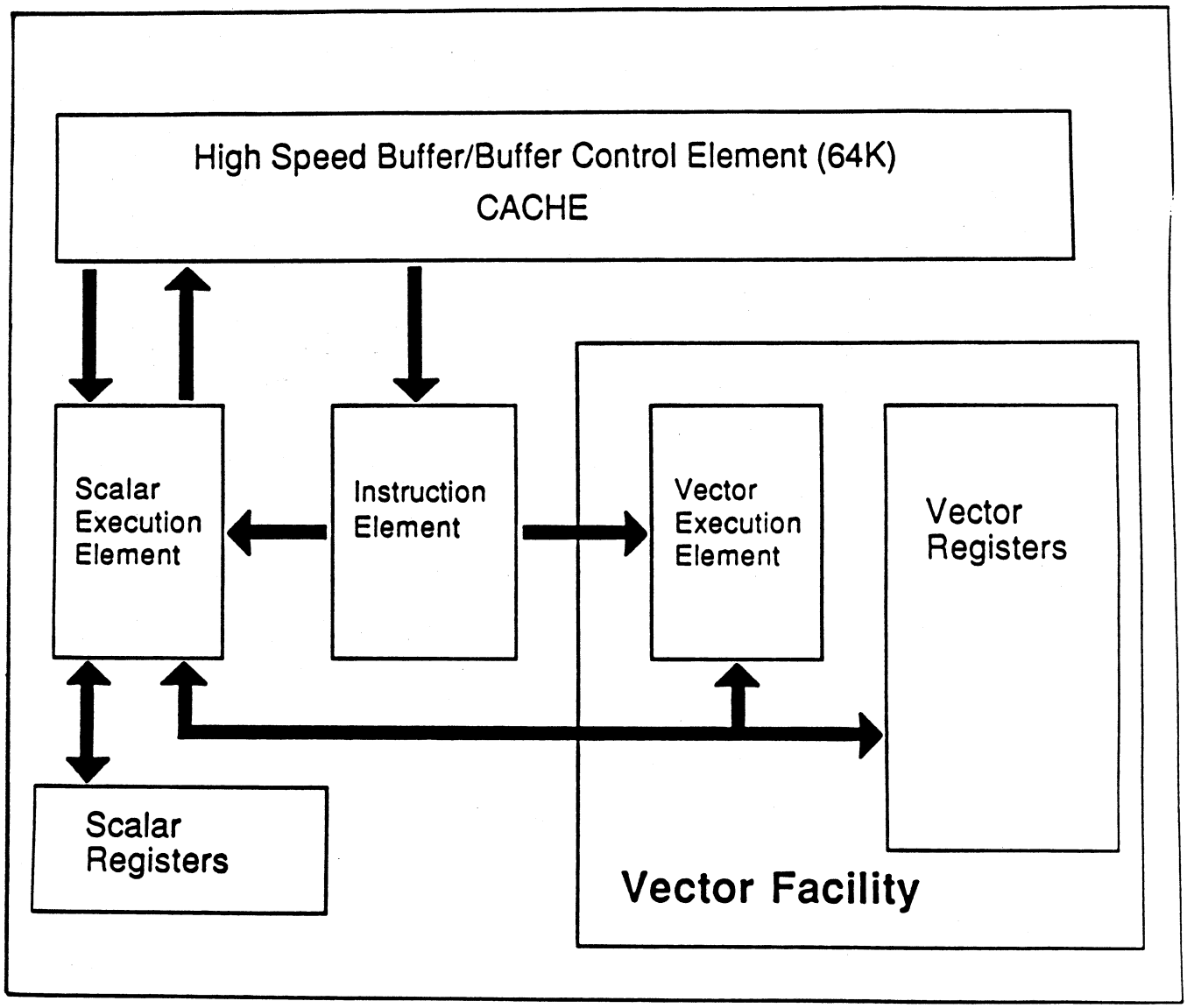


Fig. 3(b) IBM 3090- The centeral processor unit.

megabytes per second. The machine comprises of three main sections: vector unit, concurrency control unit and the FX/Fortran language processor which extends Fortran 77 to permit assignment and other operations on full arrays. The language extensions may be efficiently used to expose the inherent data independence which may have been masked by over-modularizing the program.

The Alliant's Computational Elements can run in three different modes as shown in figure 5, namely,

- Complex mode: all specified processors attack a single job,
- Detached mode: processors run independent jobs in parallel, and
- Dynamic complex mode: the complex attacks a single job an the detached processors run jobs concurrently.

The complex mode provides the optimum situation among the above modes since it provides maximum throughput for a single job; the user may specify the number of processors in the complex to achieve desired efficiency and speed. In the detached mode, the user specifies the number of CEs statically detached and dedicated to independent jobs. And in the dynamic mode, maximum flexibility is achieved for mixed mode production and development installations where user can favor detached or complex CEs to meet specific needs.

The Alliant's Fortran compiler automatically optimizes programs written in Fortran 77 for speed of execution through loop level concurrency and vectorization thus providing automatic parallelization of sequential code. The compiler's optimization strategy may be summarized as follows:

IBM 3090 Supercomputer Vector Performance

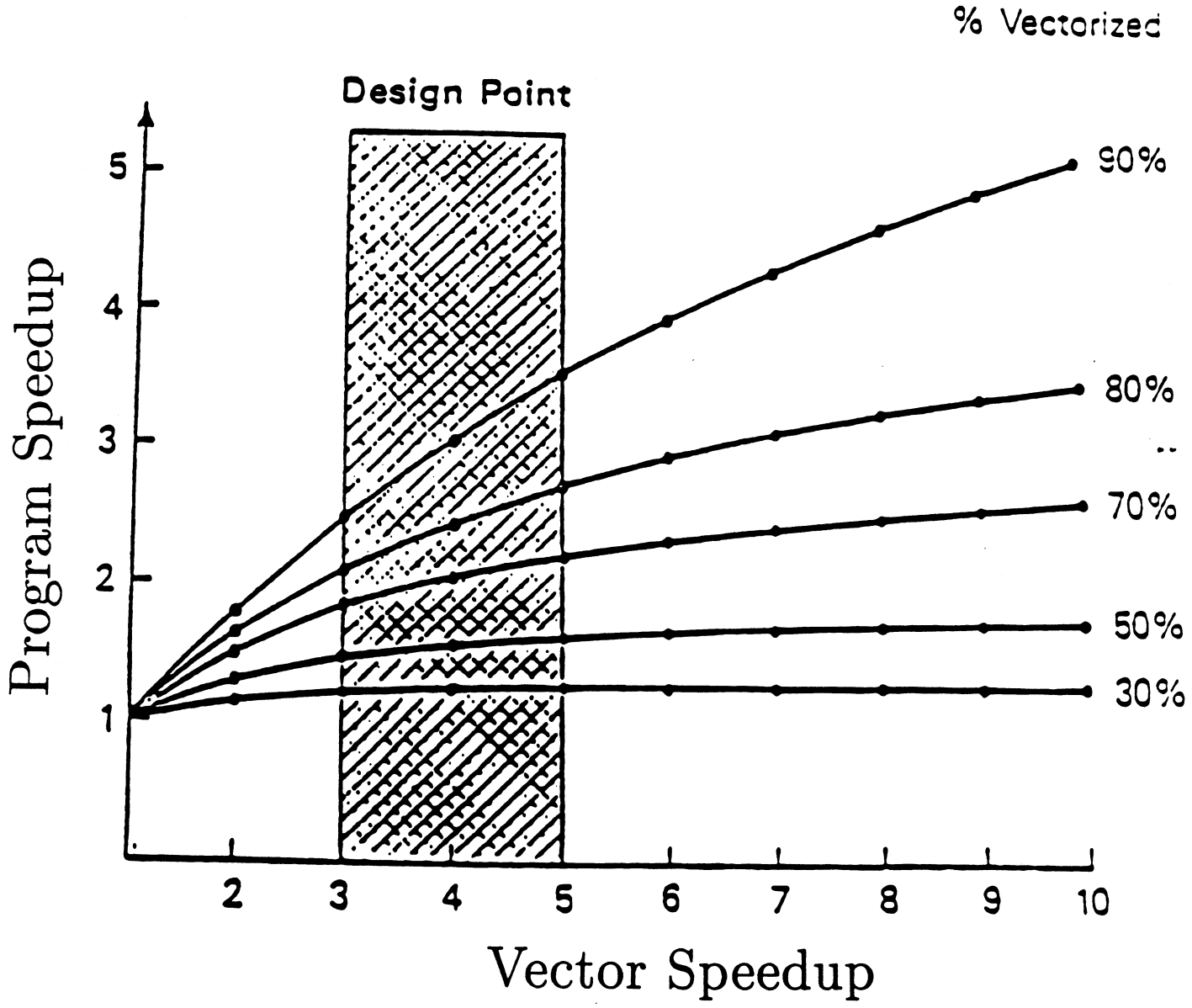
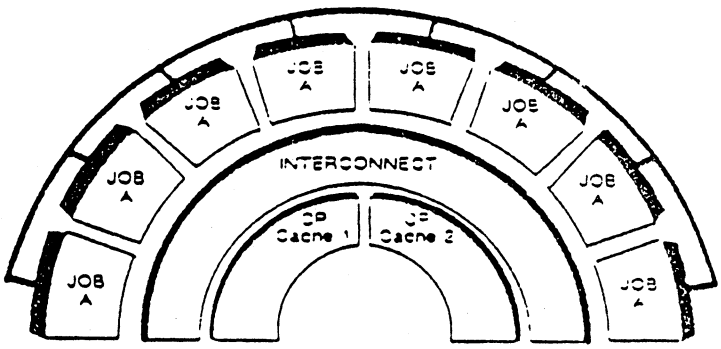


Fig. 4 IBM 3090- The vector performance.

Three Modes of Execution under Software Control

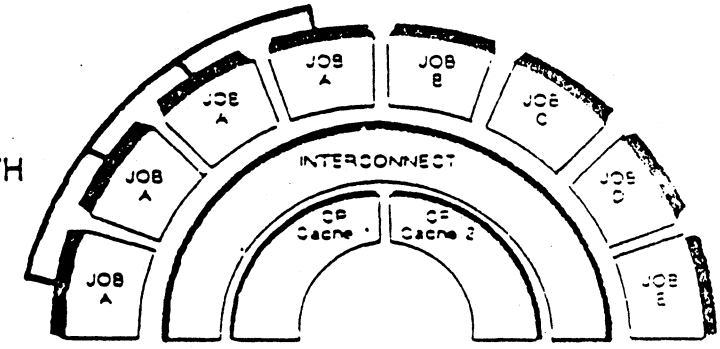
COMPLEX MODE:

OPTIMUM TIME-TO-SOLUTION



DETACHED MODE:

HIGH THROUGHPUT COMBINED WITH PRODUCTION EXECUTION
(User specifies number of CEs statically detached)



DYNAMIC COMPLEX MODE:

MAXIMUM FLEXIBILITY FOR MIXED MODE PRODUCTION AND DEVELOPMENT INSTALLATIONS
(User can favor detached or complex to meet changing needs)

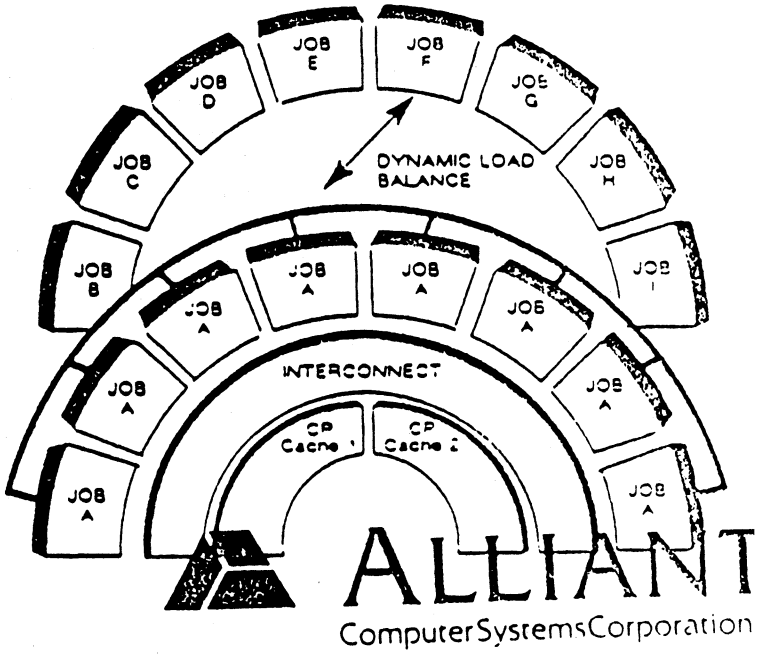


Fig. 5 Alliant FX/8 modes of operation.

For *each innermost loop*:

- If **vectorizable** then
  - if *next outer loop* is **parallelizable** then
    - Concurrent-Outer Vector-Inner
  - else
    - Vector-Concurrent
  - endif
- elseif **parallelizable** then
    - Scalar-Concurrent
- else
    - Not Optimized
- endif

The vector-concurrent execution in the complex mode provides the fastest performance since it utilizes all the available processors to attack a single task concurrently with vector operations performed on strides of data on each processor. The resulting high-performance, low-level parallelism can significantly boost the performance of compute-intensive operations such as Fourier transformations.

3.6 Optimized Algorithm

Here, the vectorizable nature of the CGFFT algorithm is exploited by identifying the major processes involved in a given iteration. From (6), it can be seen

that a considerable amount of computation time is spent in the calculation of the convolutions carried out in $A[P_n]$ and $A^a[R_n]$. Each convolution includes a pair of forward and inverse Fourier transform operations on the relevant components of the current density vector along with a Hadamard (element by element) multiplication of the current and the dyadic Green's function in the spectral domain. Since for a given current component there is no data recurrence at a particular point in an iteration, these operations may be vectorized to increase the speed of calculations. The same observation is true for the computation of the norms and the dot products used to update the current J, the residual vector R, and the search vector P. More importantly, since the FFT is a highly vectorizable algorithm, it plays a major role in the speed and efficiency of the optimized code.

In this study the scalar and optimized FFT routines available on the IBM 3090's ESSL library and the Alliant's Fortran math library were used in both scalar and vector modes. These routines are written in conjunction with assembler language and generate instructions appropriate for the architecture of the processors-they manage data to make efficient uses of the memory hierarchy.

4 Scattering from Large Planar Structures

The integral equation for a thin perfectly conducting strip of width w illuminated by a H-polarized plane wave of intensity H_0 incident at an angle ϕ_0 (Fig. 6) is given by

$$H_0 \sin\phi_0 e^{jk_0 x\cos\phi_0} = \frac{1}{4k_0}(k_0^2 + \frac{\partial^2}{\partial x^2})\int_{-w/2}^{w/2} J_x(x')H_0^{(2)}(k_0|x-x'|)dx' \qquad (10)$$

where $H_0^{(2)}$ is the zeroth order Hankel function of the second kind and $J_x(x)$ is the unknown surface current density on the strip. By expanding the current in terms of sub-domain basis functions as

$$J_x(x) = \sum_{n=1}^{N} J_n f(x - x_n), \qquad (11)$$

equation (10) takes the form

$$H_0 \sin\phi_0 e^{jk_0 x\cos(\phi_0)} = \frac{1}{4}\mathcal{F}^{-1}\{\widetilde{\mathcal{W}}(f_x)\hat{J}\} \qquad (12)$$

where

$$\widetilde{\mathcal{W}}(f_x) = (1 - f_x^2)\widetilde{H}_0^{(2)}(f_x)\tilde{f}(f_x), \qquad (13)$$

$$\widetilde{H}_0^{(2)}(f_x) = \frac{2j}{k_0\sqrt{f_x^2 - 1}}. \qquad (14)$$

and f_x is the transform variable.

Consider now a perfectly conducting rectangular plate as shown in figure 7. The required integral equations for the current density are given by

$$E_x^i(x,y) = [(k_0^2 + \frac{\partial^2}{\partial x^2})\Pi_x + \frac{\partial^2}{\partial x\partial y}\Pi_y] \qquad (15)$$

$$E_y^i(x,y) = [\frac{\partial^2}{\partial x\partial y}\Pi_x + (k_0^2 + \frac{\partial^2}{\partial y^2})\Pi_y] \qquad (16)$$

where $\mathbf{E}^{\mathbf{i}}$ denotes the incident plane wave, $\mathbf{\Pi}$ is the electric Hertz vector defined as

$$\mathbf{\Pi}(\mathbf{r}) = \frac{jZ_0}{k_0} \int_{s'} \mathbf{J}(\mathbf{r}')G(\mathbf{r};\mathbf{r}')ds', \tag{17}$$

and G is the free space Green's function. Introducing the two dimensional expansion

$$J(x,y) = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} J_{nm} f(x - x_n, y - y_m), \tag{18}$$

the system of equations (15) and (16) may be rewritten as

$$\mathbf{E}^{\mathbf{i}} = jk_0 Z_0 \mathcal{F}^{-1}\{\tilde{\bar{\mathcal{D}}} \cdot \tilde{\mathbf{J}}\} \tag{19}$$

where $\bar{\mathcal{D}}$ denotes the dyadic Green's function,

$$\tilde{\bar{\mathcal{D}}} \equiv \begin{pmatrix} (1 - f_x^2)\tilde{G} & (-f_x f_y)\tilde{G} \\ (-f_x f_y)\tilde{G} & (1 - f_y^2)\tilde{G} \end{pmatrix}, \tag{20}$$

f_x and f_y are the transform variables, and

$$G(x,y) = \frac{e^{-jk_0\sqrt{x^2+y^2}}}{4\pi\sqrt{x^2+y^2}} \Longleftrightarrow \tilde{G}(f_x,f_y) = \frac{1}{2k_0\sqrt{f_x^2+f_y^2-1}}. \tag{21}$$

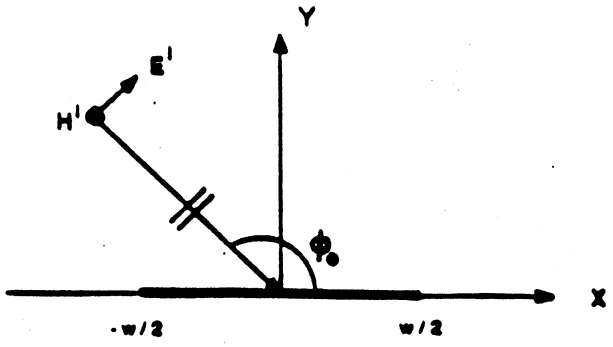


Fig. 6 Geometry of a perfectly conducting strip illuminated by a plane wave.

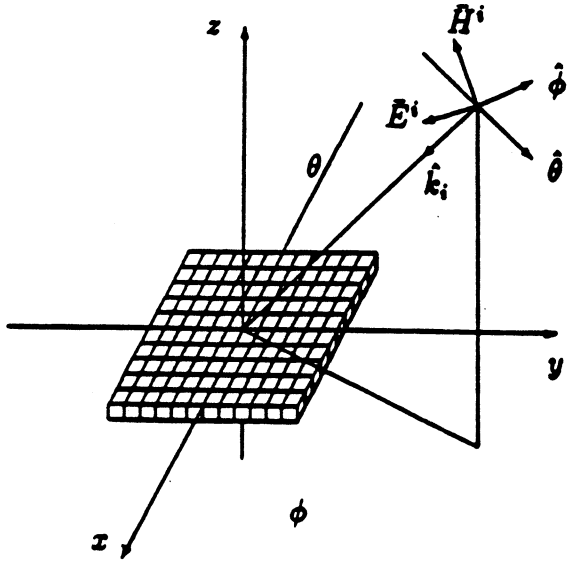


Fig. 7 Geometry of the perfectly conducting plate illuminated by a plane wave.

5 Results

Figure 8 shows the performance improvement of the CGFFT method when executed in vector-concurrent mode over the scalar mode for strips of various sizes at normal incidence. The actual FFT pad used in this experiment was of order one($\rho = 1$). Thus, the size of the pad varied from $N' = 32$ for the 1λ strip to $N' = 512$ for the 17λ one(15 unknowns/λ). Clearly, the optimized algorithm provides a superior performance especially for larger problems. Figure 8 also includes the effect of incorporating the piecewise sinusoidal expansion functions into the CGFFT formulation which results in the improvement in the speed of the method by reducing the number of iterations required to meet the tolerance. This is also shown in figure 9 for both scalar and vector-concurrent processing modes. The magnitude of the surface current density on the 17λ strip is shown in figure 10 at normal incidence.

A detailed examination of the performance of the optimized code is given in Appendix III. The appendix lists the source code for each routine and enumerates the loops and array operations that were candidates for optimization. The information presented includes the location of the loop within the code, the structure of the loop, the nature of optimization applied and possible data dependences inhibiting the optimization of the loop. Clearly, all the important (CPU-intensive) loops are vectorized and processed concurrently on multiple processors.

The plate problem was examined in more detail. A $2\lambda \times 2\lambda$ plate was considered at normal incidence for E-polarization using 63×63 unknowns and 128×128 FFT pad. The vectorized algorithm was executed on the IBM 3090 under three

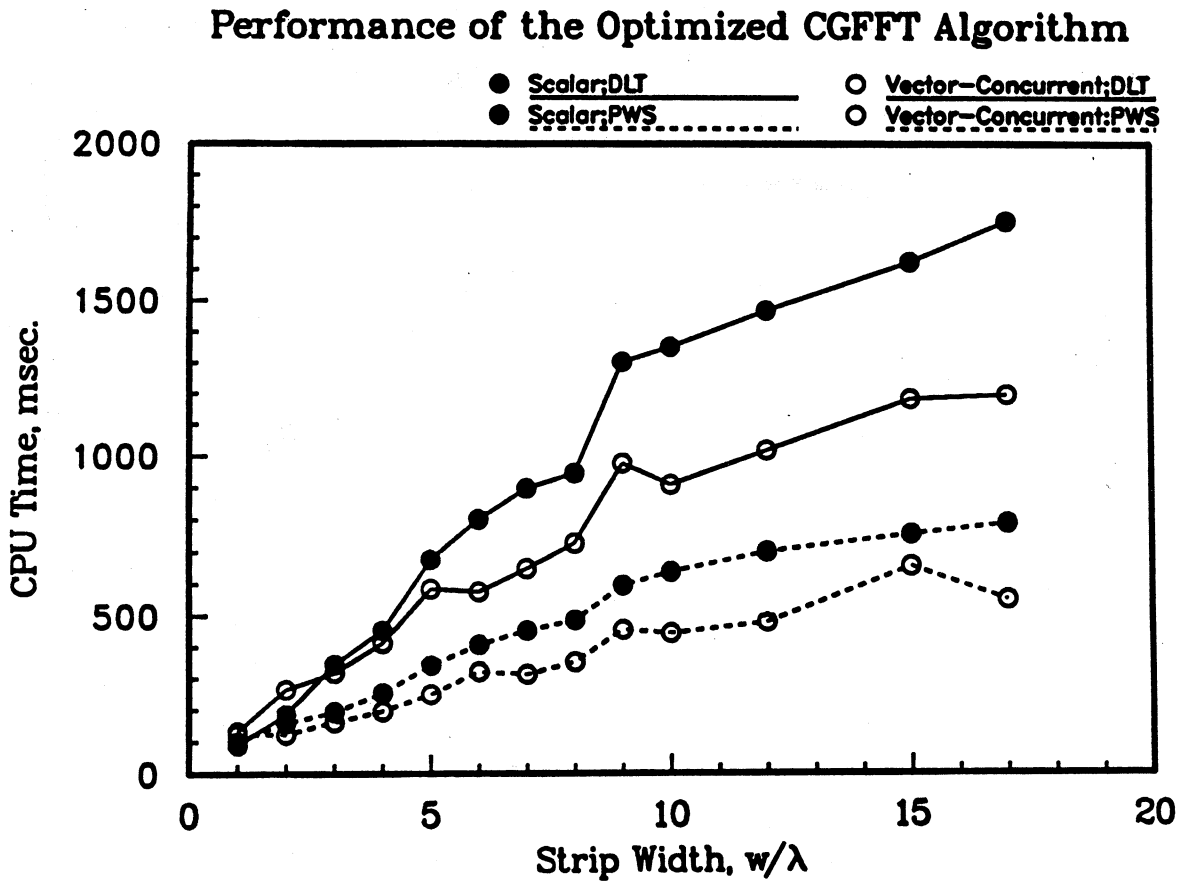


Fig. 8 **Comparison of the CPU times associated with the scalar and vector-concurrent execution of the CGFFT method for various strips using different basis functions(15 unknowns/λ and FFT pad of order 1).**

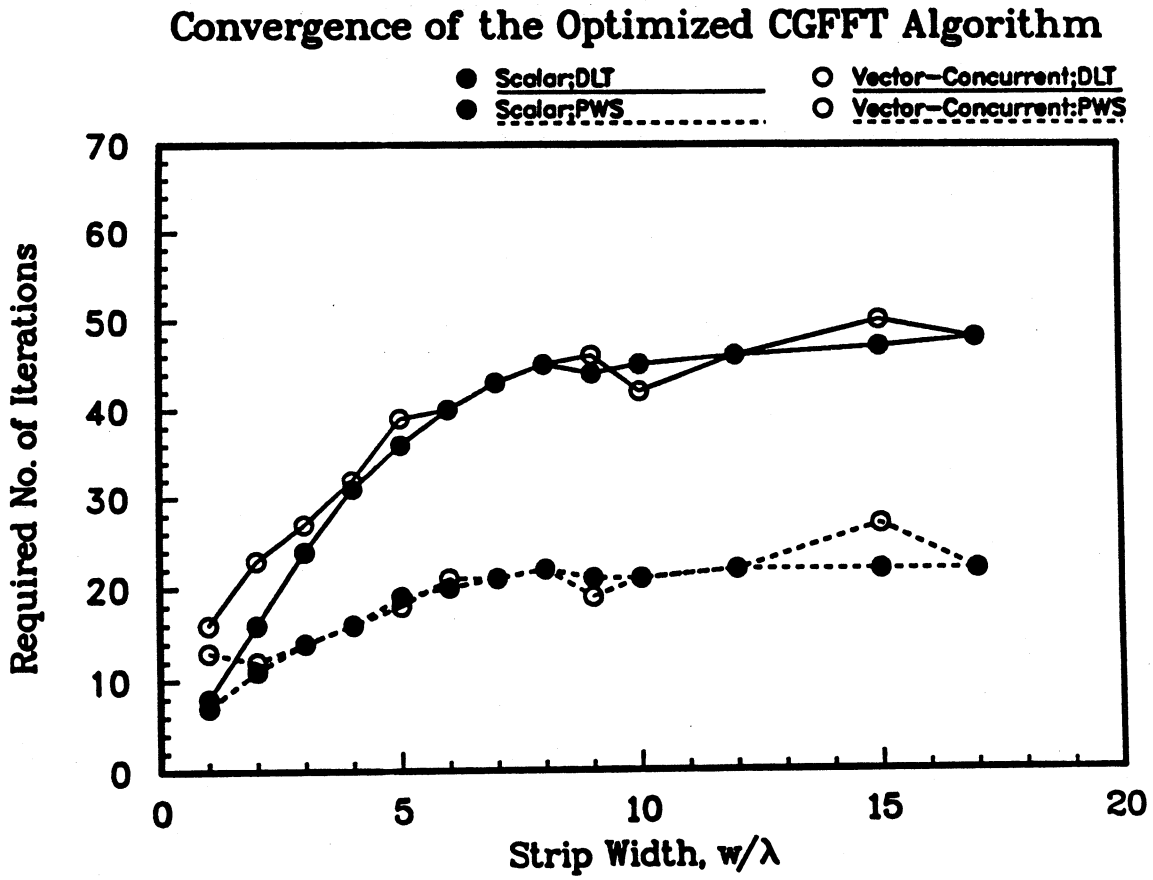


Fig. 9 A comparison of the required number of iterations associated with the convergence of the optimized CGFFT algorithm for various strips using different basis functions(15 unknowns/λ; FFT pad of order 1).

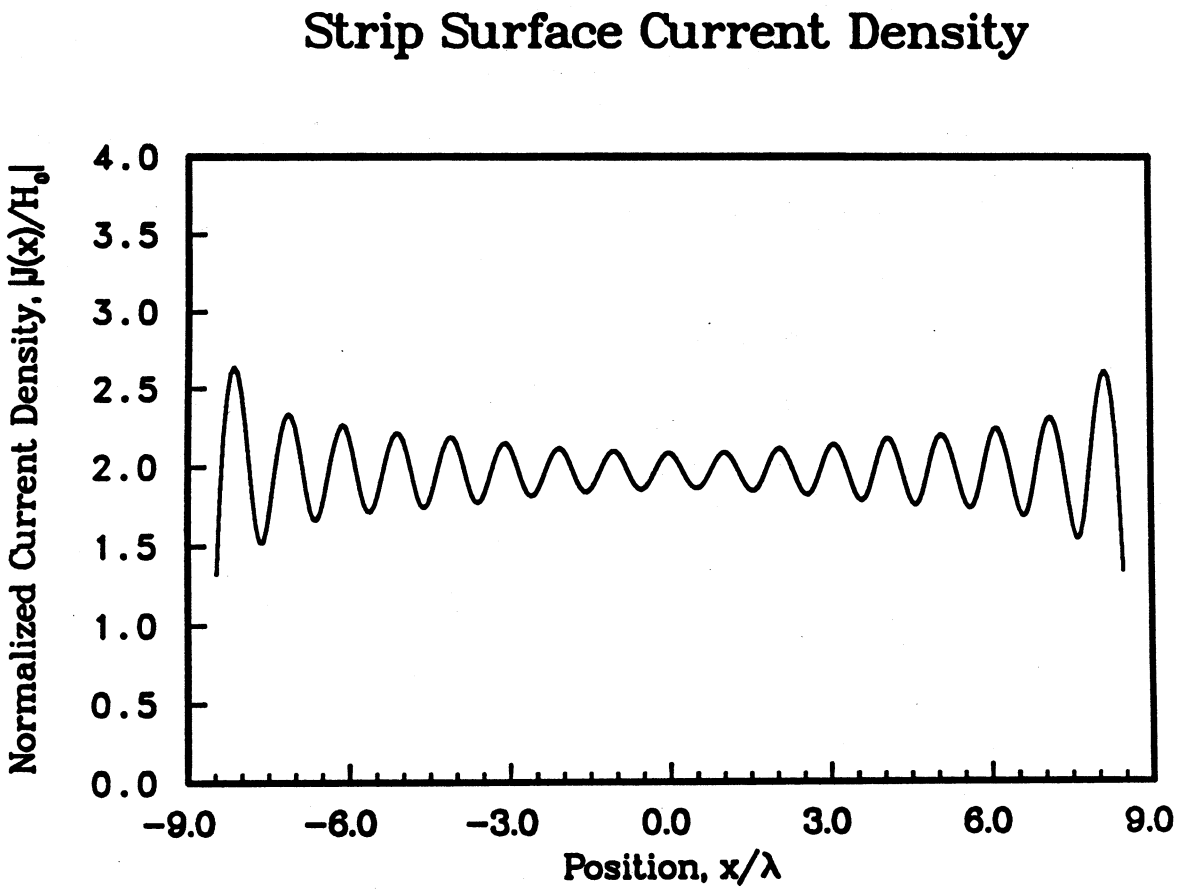


Fig. 10 **Surface current density magnitude for the 17λ conducting strip illuminated by an H-pol. plane wave at normal incidence computed by the optimized CGFFT algoritm(15 unknowns/λ) using PWS basis functions.**

situations. In scalar mode using a scalar radix 2 FFT routine and in vector mode with scalar and vectorized FFT routines both provided by the IBM's ESSL library. The results are tabulated in Table 1 showing the total CPU time and the time spent in the vector facilities. The percent vectorizable code of 97% indicates a highly optimized algorithm resulting in a program speedup of more than four times. This is mostly attributed to the employed vectorized FFT routine. The performance point of the supercomputer for this particular case is given in figure 11.

Table 2 shows the corresponding vectorization data for the Alliant. The entries in the column denoted by MFLOPS are calculated based on the estimated number of operations from [5]. The distribution of the CPU time among the compute-intensive routines in the scalar and vector modes is shown in figure 12(a). Again, it is clear that calls to the FFT routine are the most significant factor in the solution process and about 90% of the CPU time is spent in these computations(Fig. 12(b)).

To examine the efficiency of the algorithm for vector-concurrent execution, the optimized code was run on one to four vector processors on the Alliant and the results are shown in Table 3. A speedup of 3.5 (over the vector execution on one processor) was achieved when employing four concurrent processors at an efficiency of 88%. This represents a speedup of more than 20 times (per iteration) over the sequential scalar execution.

To achieve near peak performance, vectors of length nR_v should be used, where R_v is the size of vector register (32 or 128) and n is the number of processors. In order to further evaluate the efficiency of the method when the size of the problem grows, the cases of $5\lambda \times 5\lambda$ and $10\lambda \times 10\lambda$ plates were also considered with the

corresponding results reported in Tables 4 and 5. The sampling density in these cases were 25 unknowns/λ with a FFT pad of order one. A summary of the vector-concurrent executions for the plate problem is given in Table 6. The slight decrease in the efficiency, as the problem size becomes larger, is attributed to an increase in the complexity of the memory cache references. However, considering the fact that the cache memory size has been exceeded in all the cases reported here, the performance of the optimized algorithm is quite impressive. Figures 13 to 15 show the corresponding surface current distributions excited on the conducting plates.

CGFFT Code	Execution Mode		
Performance	Scalar (scalar FFT)	Vector (scalar FFT)	Vector (vector FFT)
ELAPSED CPU, sec	148	129	34
VECTOR CPU, sec	-	9	30
VECTORIZABLE CODE, sec	-	28	144
VECTOR CONTENT	-	18.9%	97.3%
VECTOR SPEED UP	-	3.1	4.8
PROGRAM SPEED UP	-	1.15	4.35

Table 1: Performance of the scalar and vectorized 3-d code on the IBM 3090.

IBM 3090 Supercomputer Vector Performance

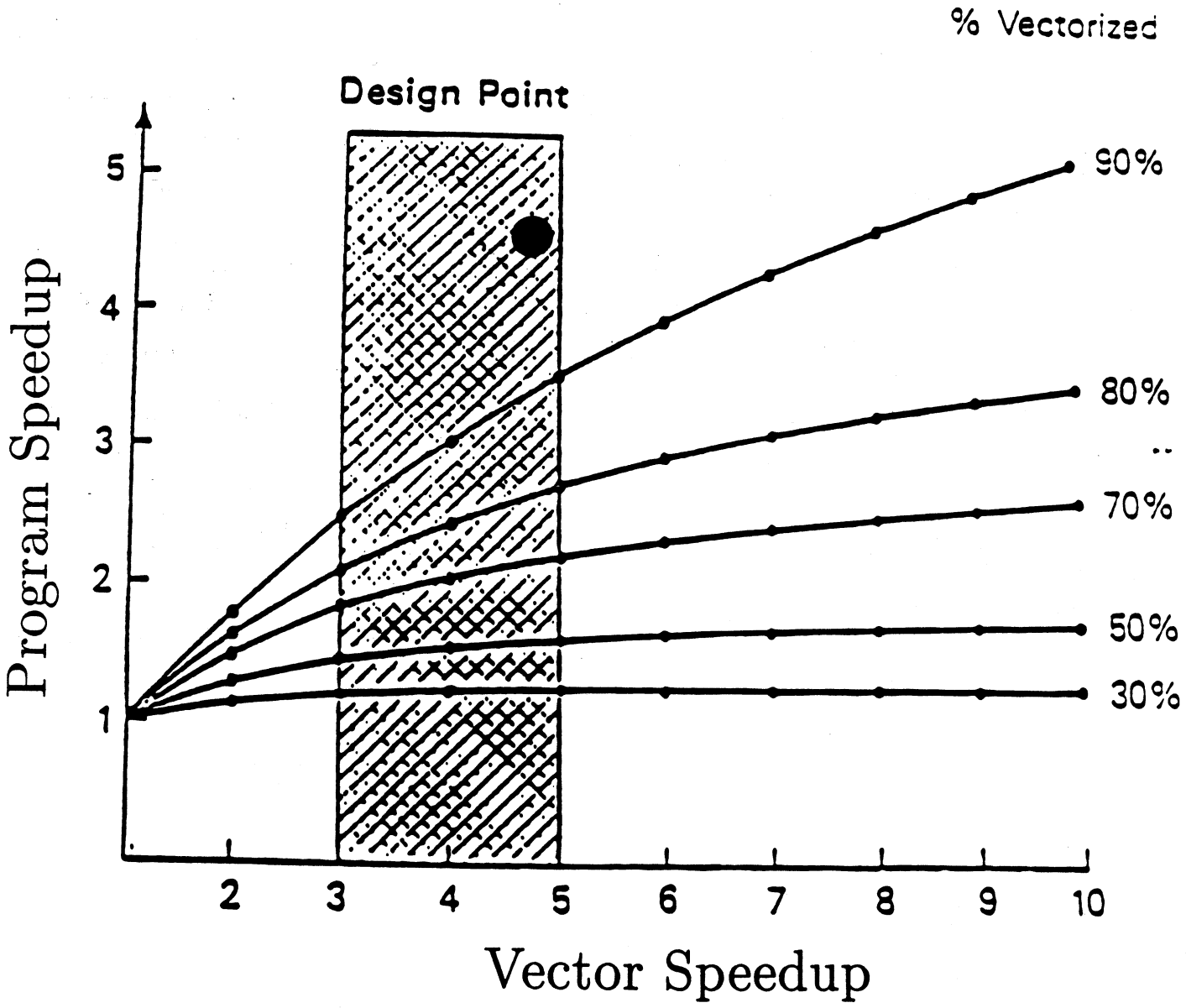


Fig. 11 Performance of the optimized CGFFT algorithm on the IBM 3090.

CGFFT Code	Execution Mode		
Performance	Scalar (scalar FFT)	Vector (scalar FFT)	Vector (vector FFT)
INITIALIZATION, sec	2.39	0.72	0.80
CGFFT LOOP, sec	4307.0	1255.2	342.8
TOTAL CPU TIME, sec	4309.3	1255.9	343.6
ITERATIONS	111	111	59
PER ITERATION, sec	38.80	11.31	5.8
MEGAFLOPS	0.0527	0.1807	0.3512
PROGRAM SPEED UP	-	3.43	12.54
SPEED UP/ITERATION	-	3.43	6.69

Table 2: Performance of the scalar and vectorized 3-d code on the Alliant FX/8.

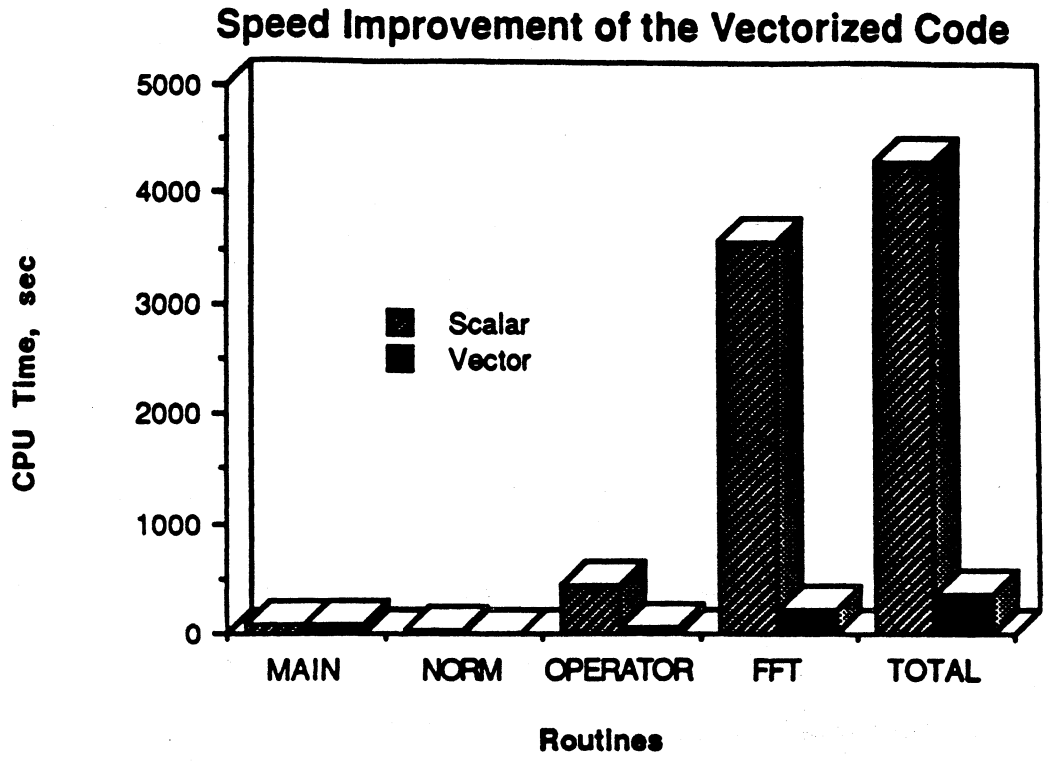


Fig. 12(a) Distribution of CPU time among the compute-intensive routines.

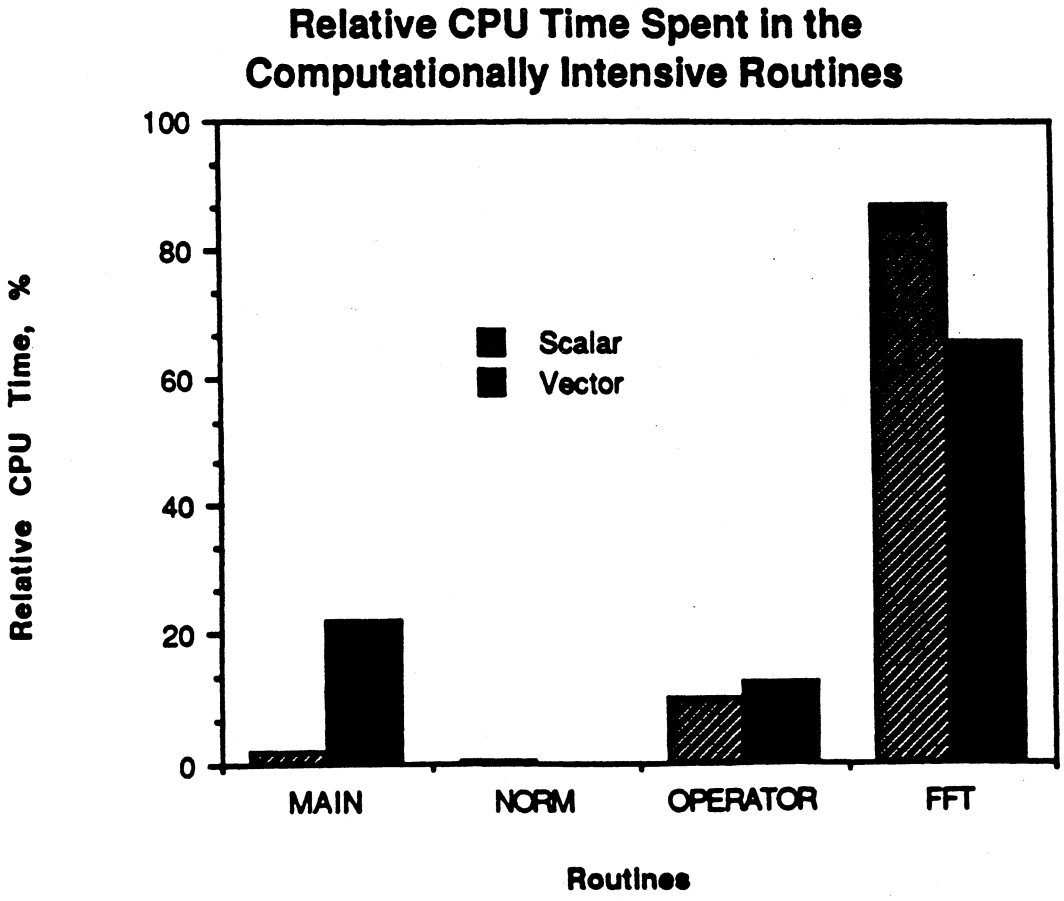


Fig. 12(b) Relative CPU time spent in the compute-intensive routines.

Optimized CGFFT	NO. of processors			
Performance	1	2	3	4
INITIALIZATION, sec	1.60	0.86	0.61	0.50
CGFFT LOOP, sec	407.4	211.0	146.7	115.6
TOTAL CPU TIME, sec	409.0	211.8	147.3	116.1
ITERATIONS	59	59	59	59
PER ITERATION, sec	6.91	3.58	2.49	1.96
MEGAFLOPS	0.2951	0.5697	0.8192	1.0340
SPEED UP	-	1.93	2.78	3.52
EFFICIENCY	-	96.5%	92.7%	88.0%

Table 3: Vector-Concurrent performance for a $2\lambda \times 2\lambda$ plate.

Optimized CGFFT	NO. of processors			
Performance	1	2	3	4
INITIALIZATION, sec	6.35	3.26	2.32	1.90
CGFFT LOOP, sec	1096.6	568.9	398.4	319.0
TOTAL CPU TIME, sec	1103.0	572.2	400.7	320.9
ITERATIONS	46	46	46	46
PER ITERATION, sec	23.98	12.44	8.71	6.98
MEGAFLOPS	0.3848	0.7417	1.0590	1.3226
SPEED UP	-	1.93	2.75	3.43
EFFICIENCY	-	96.5%	91.7%	85.8%

Table 4: Vector-Concurrent performance for a $5\lambda \times 5\lambda$ plate.

Optimized	NO. of processors			
Performance	1	2	3	4
INITIALIZATION, sec	24.55	13.60	9.10	7.23
CGFFT LOOP, sec	3673.3	1973.0	1349.8	1080.5
TOTAL CPU TIME, sec	3697.8	1986.6	1358.9	1087.7
ITERATIONS	38	38	38	38
PER ITERATION, sec	97.31	52.28	35.76	28.62
MEGAFLOPS	0.4223	0.7861	1.1492	1.4357
SPEED UP	-	1.86	2.72	3.40
EFFICIENCY	-	93.0%	90.7%	85.0%

Table 5: Vector-Concurrent performance for a $10\lambda \times 10\lambda$ plate.

Configuration				No. of processors*									
Plate	FFT SIZE		PROCESS MEMORY	1	2			3			4		
Size	Pad	KBYTES	MBYTES	MFLOPS	MFLOPS	Speedup	Efficiency	MFLOPS	Speedup	Efficiency	MFLOPS	Speedup	Efficiency
$2\lambda \times 2\lambda$	128×128	128	1.6	0.30	0.57	1.93	96.5%	0.82	2.78	92.7%	1.03	3.52	88.8%
$5\lambda \times 5\lambda$	256×256	512	6.0	0.38	0.74	1.93	96.5%	1.06	2.75	91.7%	1.32	3.43	85.8%
$10\lambda \times 10\lambda$	512×512	2048	24.0	0.42	0.79	1.86	93.0%	1.15	2.72	90.7%	1.44	3.40	85.0%

*The speedup and efficiency are relative to execution on a single processor.

Table 6: A summary of the vector-concurrent performances of the optimized CGFFT algorithm for different configurations.

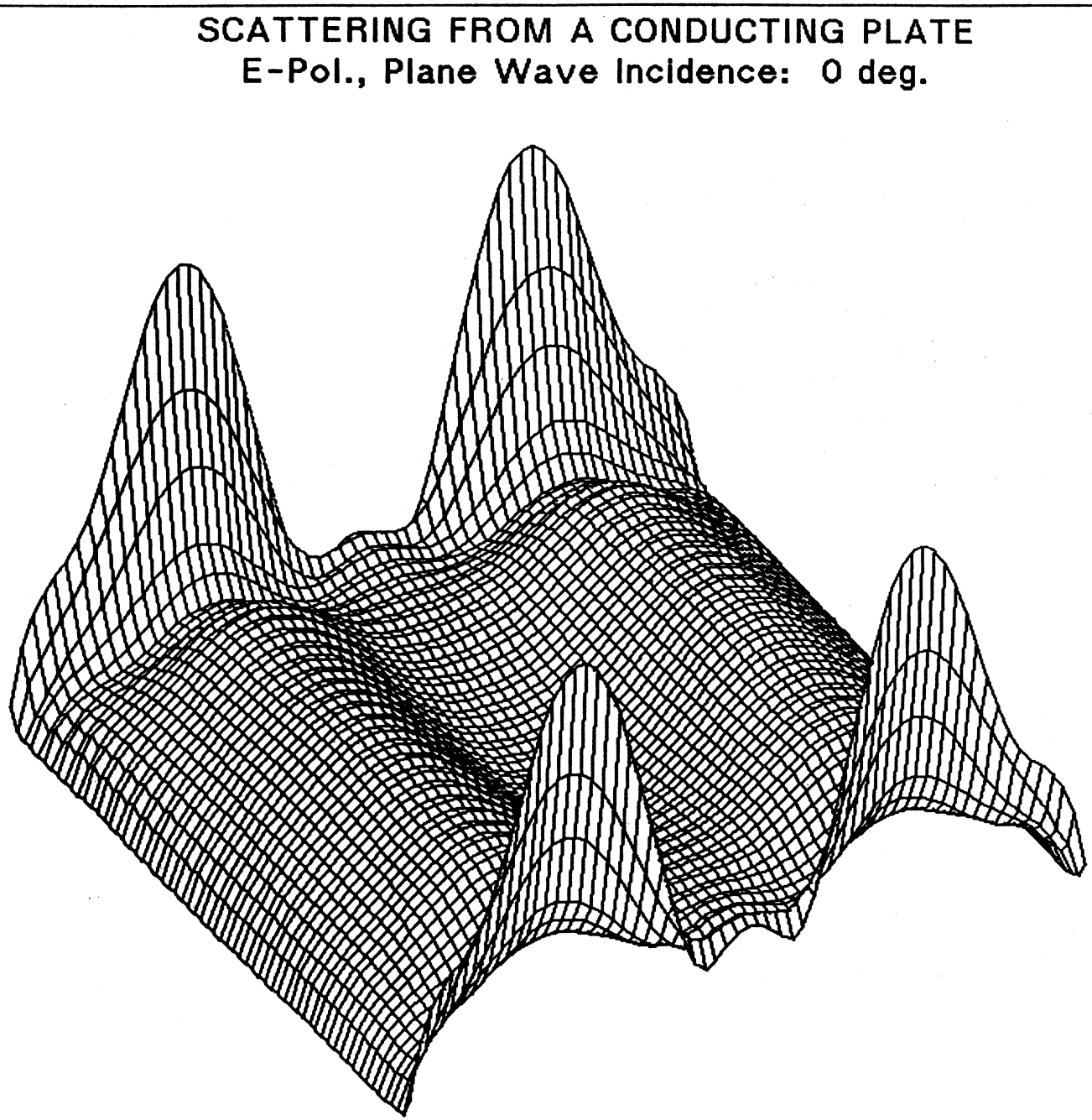
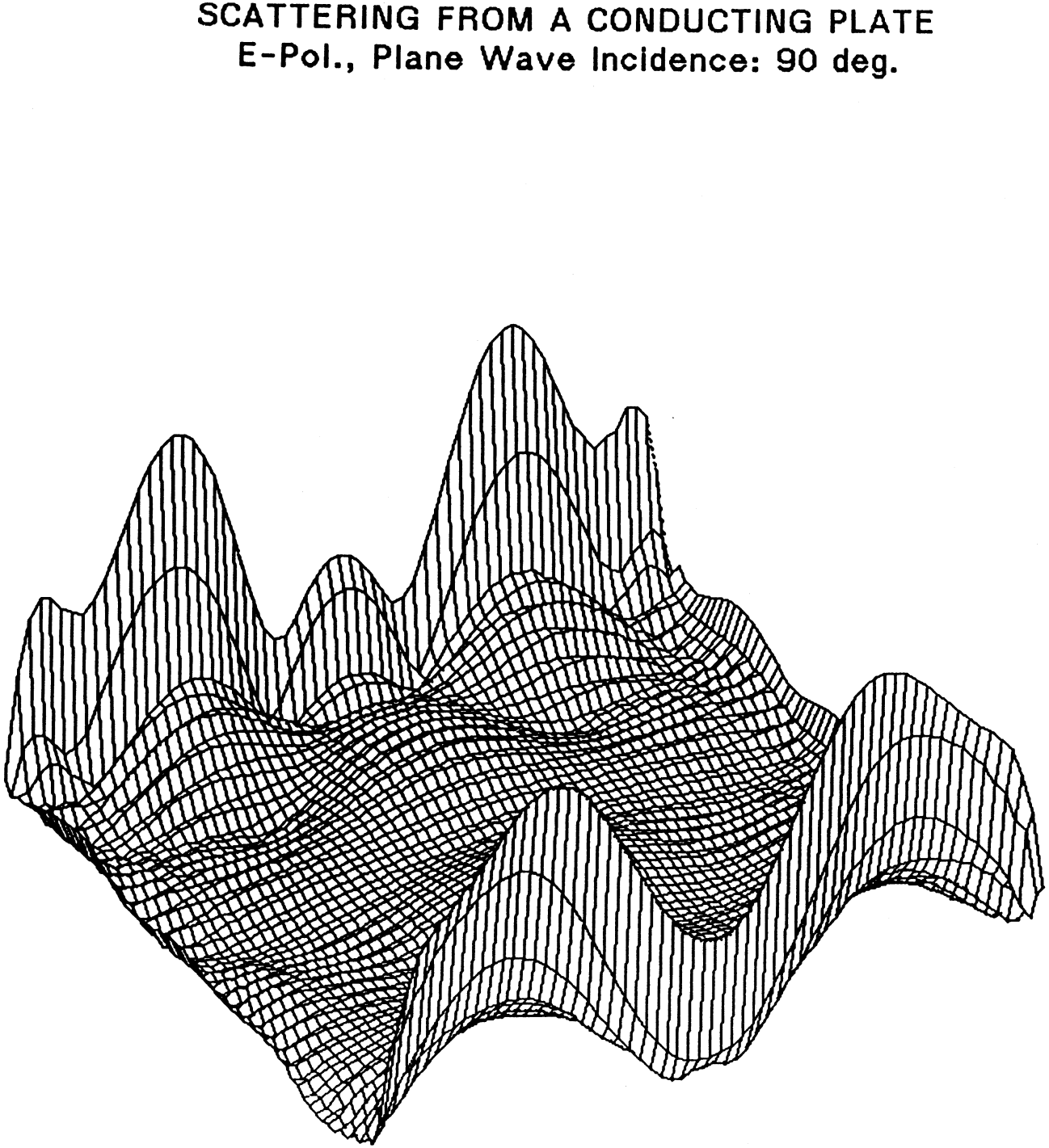


Fig. 13 The surface current density on a $2\lambda \times 2\lambda$ conducting plate illuminated by an E-polarized normally incident plane wave (63×63 unknowns and FFT pad of order 1).
(a)Like-polarized component, $|J_x(x,y)|$



(b)Cross-polarized component, $|J_y(x, y)|$

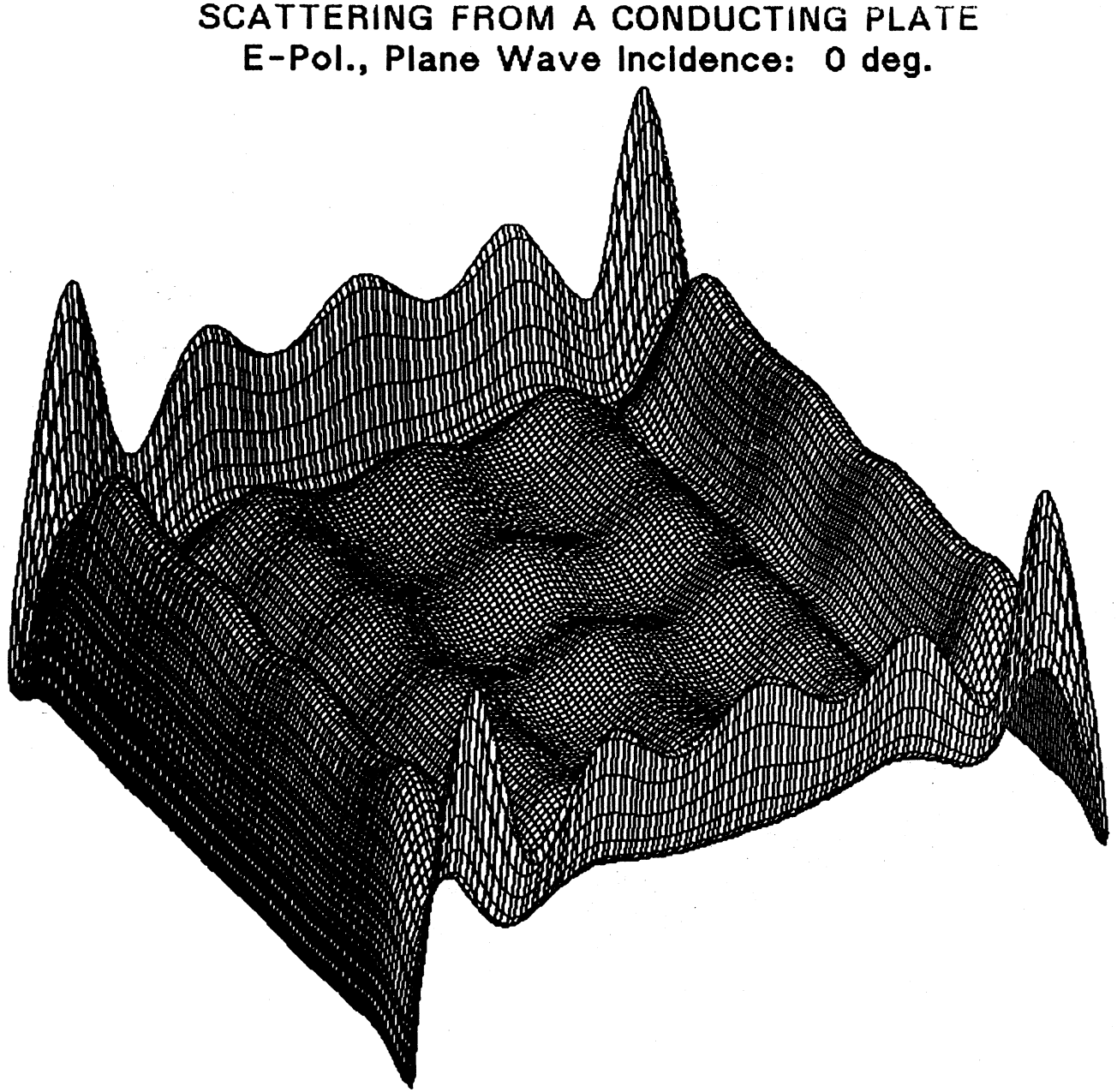
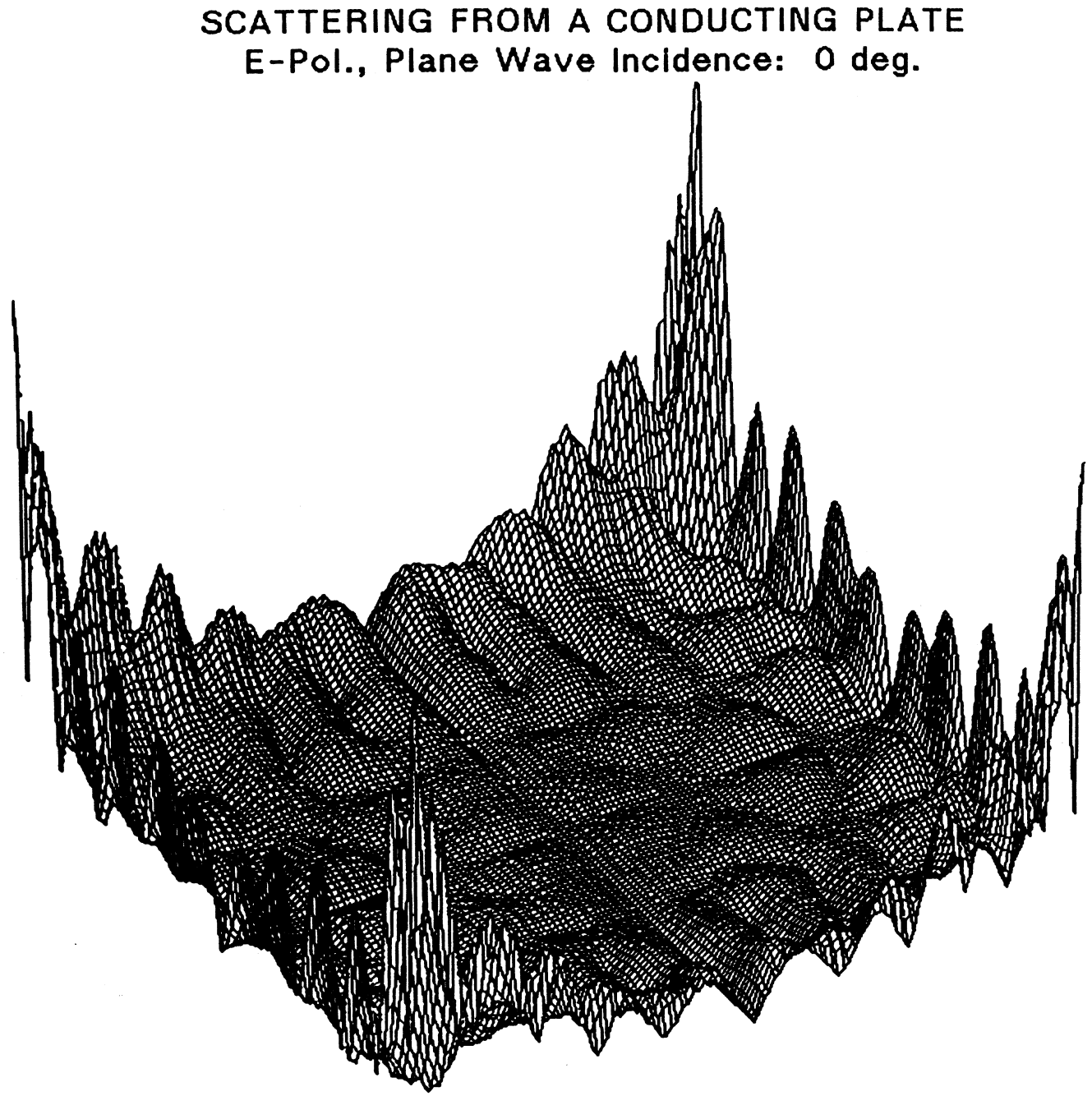
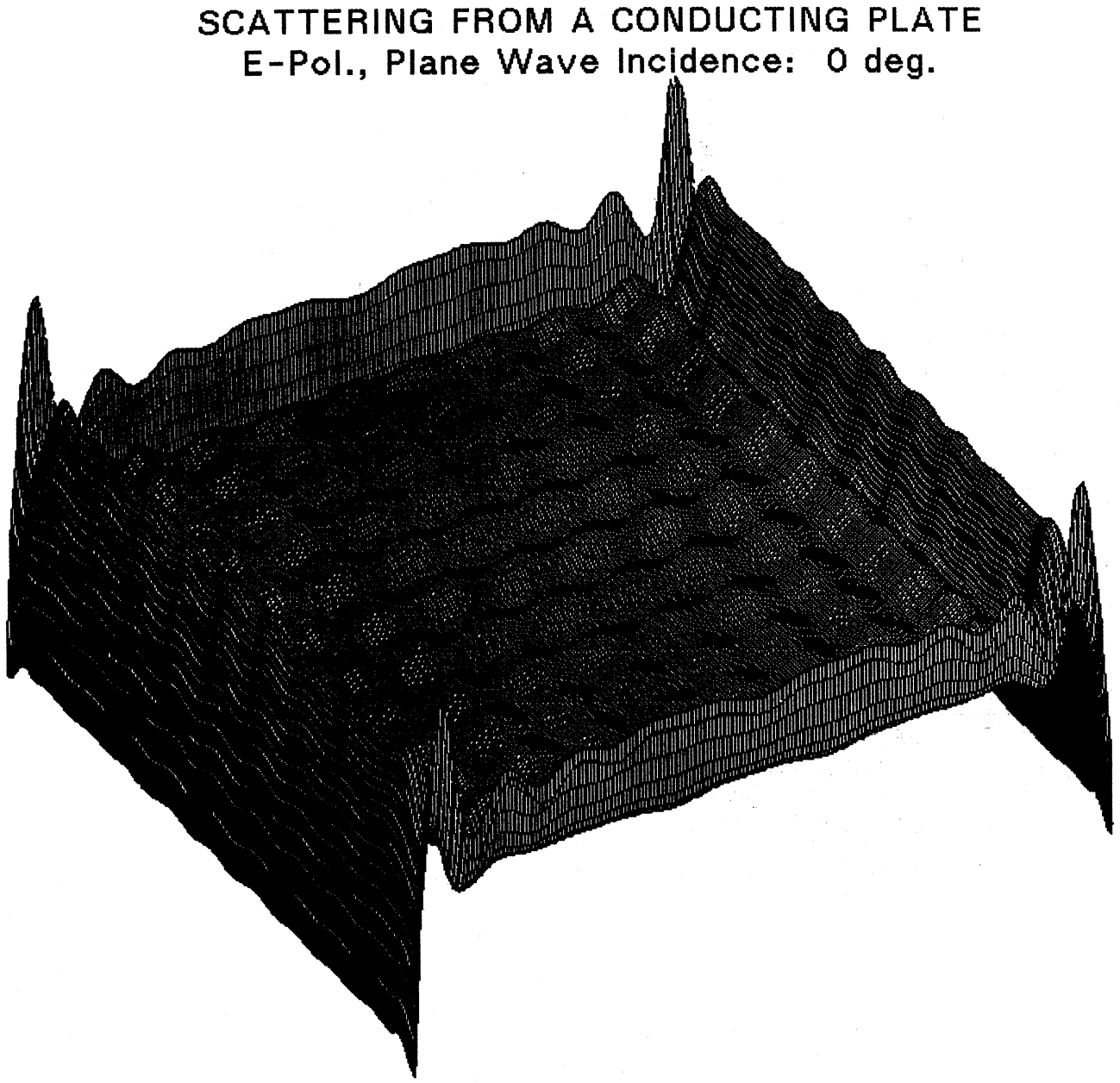


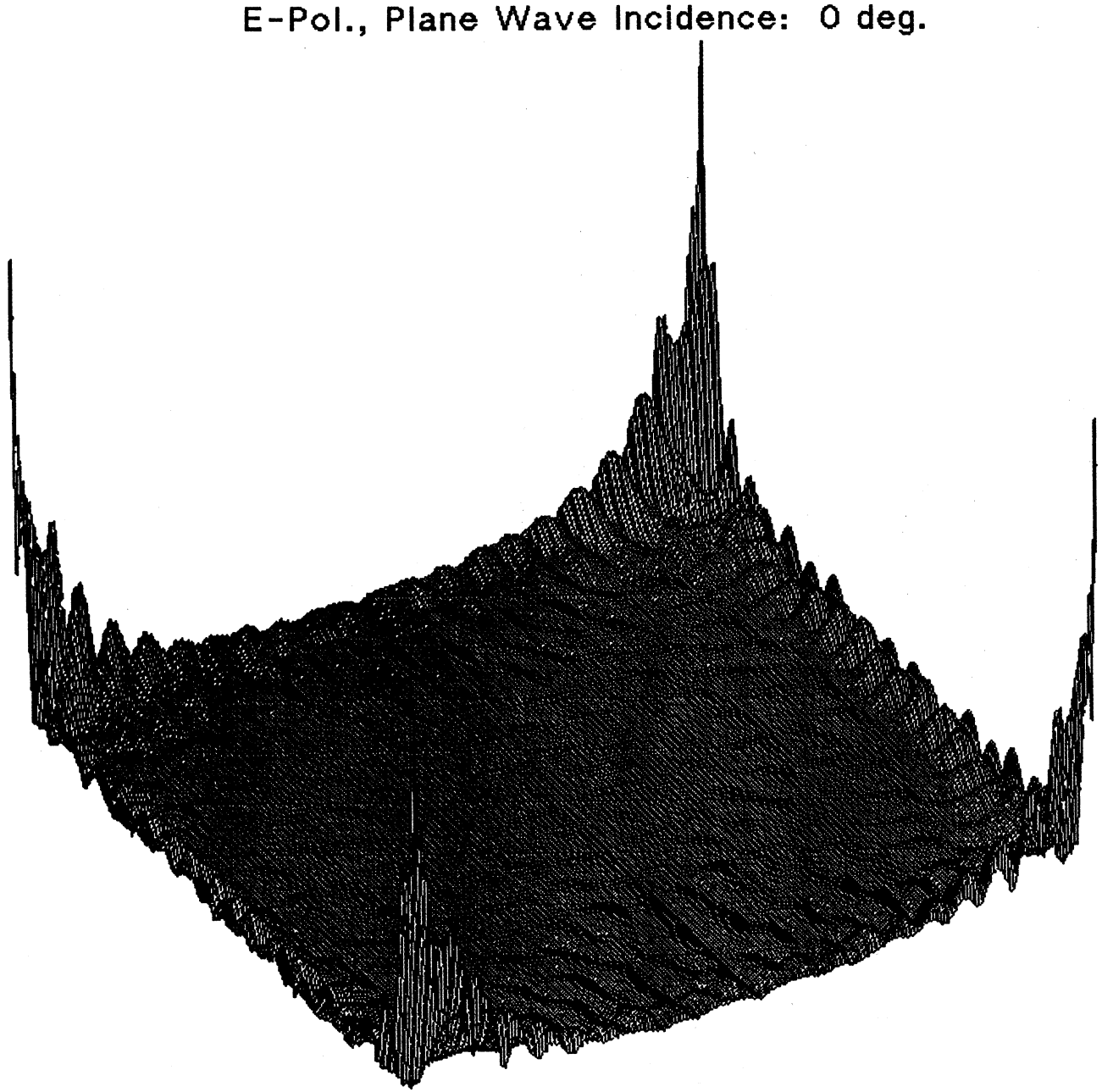
Fig. 14 The surface current density on a $5\lambda \times 5\lambda$ conducting plate illuminated by an E-polarized normally incident plane wave (125×125 unknowns and FFT pad of order 1).
(a)Like-polarized component, $|J_x(x, y)|$



(b)Cross-polarized component, $|J_y(x, y)|$



(a)Like-polarized component, $|J_x(x, y)|$



(b)Cross-polarized component, $|J_y(x,y)|$

6 Conclusion

It was shown that the conjugate gradient FFT algorithm is suitable for vector-concurrent optimization and may be efficiently implemented on multi-processor computers. The FFT plays a crucial role in the speedup and the efficiency of such an application. As the size of the problem becomes larger, there is a corresponding degradation in the performance of the optimized code due to the complexity of the memory cache references. This complexity, however, has not proven restrictive in the examples considered because the CGFFT method does not suffer from the same memory requirements as the direct methods do. Thus, relatively large problems can be handled without considerable loss of efficiency and speed.

Although this study was concerned with automatic parallelization, which is limited to optimization of individual loops, parallelism at a larger granularity can be specified by the programmer to achieve a superior performance for more complex problems. An example of such an application is the problem of scattering by a dielectric plate of finite thickness [2] where the normal component of the current density is totally independent of the planar components and can be solved for by a dedicated processor in parallel with them.

Acknowledgements

The vector computations on the IBM 3090 were conducted on the Cornell National Supercomputer Facility (CNSF), a resource of the Center for Theory and Simulation in Science and Engineering. The vector-concurrent computations on the Alliant FX/8 were carried out on the University of Michigan's Computer Aided Engineering Network (CAEN).

The author is grateful to Professors John L. Volakis and R. J. Lomax for their support and encouragement in the course of this work.

7 Appendix I: The Fourier Transform Pair

The forward and inverse Fourier transform pair are defined for one and two dimensional cases, respectively, as follows:

$$\tilde{g}(f_x) = \int_{-\infty}^{\infty} g(x) e^{-j2\pi f_x x} dx, \tag{22}$$

$$g(x) = \int_{-\infty}^{\infty} \tilde{g}(f_x) e^{j2\pi f_x x} df_x, \tag{23}$$

and

$$\tilde{g}(f_x, f_y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(x,y) e^{-j2\pi(f_x x + f_y y)} dxdy, \tag{24}$$

$$g(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \tilde{g}(f_x, f_y) e^{j2\pi(f_x x + f_y y)} df_x df_y. \tag{25}$$

Since in practice the spatial variables are normalized with respect to λ, the free space wave number $k_0 = 2\pi$ and the following relations hold

$$\frac{\partial}{\partial x} \Longleftrightarrow j2\pi f_x \quad , \frac{1}{k_0^2}\frac{\partial^2}{\partial x^2} \Longleftrightarrow -f_x^2 \quad , \frac{1}{k_0^2}\frac{\partial^2}{\partial x \partial y} \Longleftrightarrow -f_x f_y.$$

For numerical computations the Discrete Fourier Transform (DFT) is introduced as

$$\hat{J}_k = \sum_{m=1}^{N} J_m e^{-2\pi jmk/N} \quad k = 1, \cdots, N, \tag{26}$$

where each component of $\hat{\mathbf{J}}$ is in effect the Fourier transform of a periodic waveform, one period of which is assumed to be the N samples of the original function. The two dimensional DFT is defined similarly with summations over both dimensions.

The discrete Fourier transforms are carried out by a radix 2 FFT routines. The sampling intervals and the size of the FFT pads are chosen so that the Nyquist criterion in both the spatial and frequency domains as well as the requirements of

linearity in the convolutions are satisfied. Thus, the period N' of the array to be transformed is chosen so that[6]

$$N' = 2^{\gamma} : N' > N_{Nyquist}, N' > 2 \times N - 1 \tag{27}$$

where N is the number of unknowns and γ is an integer. In practice γ is chosen according to the rule

$$\gamma \geq \log_2(2N - 1) + \rho \tag{28}$$

where ρ is an integer setting the *order* of the FFT pad dimension to ensure adequate frequency sampling in the spectral domain when performing the inverse transform operation. The array elements beyond the scatterer's physical range are set to zero.

8 Appendix II: Optimization Techniques

ARCHITECTURAL TECHNIQUES	PROGRAMMER ACTIONS†
SCALAR PIPELINING	Use compiler switches (global optimization).
VECTOR PROCESSING	1. Restructure loops and use compiler directives. 2. Maximize vector lengths by renesting, merging, unrolling loops; largest iterates to be inside. 3. Eliminate conditionals in loops & distribute them. 4. "Supervector" loops to fill vector registers. 5. Move I/O statements out of the loops. 6. Turn off vectorization for short loops,or some vectorized loops with conditionals, dependences.
CONCURRENCY	1. Eliminate/relocate dependences, scalar carry-arounds. 2. Restructure for Concurrent-Outer Vector-Inner. 3. Create concurrently-callable subroutines. 4. Renest/merge loops for more concurrent iterations.
CACHE/MEMORY ACCESS	1. Possible problems with strides. 2. Leftmost array dimension should be the largest. 4. Outer loop corresponds to the leftmost array index. 3. Process compact vectors, columns instead of rows. 4. Localize memory references.

†*Extracted from IBM 3090 and Alliant FX/8 programming manuals*

9 Appendix III: Listings

The source codes and compiler listings for the strip and plate programs are included in this appendix. The listings include informational messages pertaining to the optimization of the respective algorithms. For the plate program, the listings from both the Alliant FX/8 and the IBM 3090 are included.

```
Alliant FX/Fortran Compiler V4.0.28 (2.24D20)  15 Oct 1988 21:04:05  Page   1
Source Listing of /u/barkeshli/strip.d/strip.f

    1           PROGRAM STRIP
    2   C   ***********************************************************************
    3   C                                 MAIN
    4   C   ***********************************************************************
    5   C    TIMING PARAMETERS
    6           REAL          ETIME1,ETIME2,ETIMEOH
    7           REAL          USTIME1(2),USTIME2(2),UTIMEOH,STIMEOH
    8           REAL*8        KX(512),C(512)
    9           REAL          X(512),ETA(512),ECHO(360)
   10           COMPLEX       J(512),E(512),A(512),R(512),P(512)
   11           COMPLEX       DFGL(512),W(512),WORK(384),XJ,SUMIT,MYCSQRT
   12           CHARACTER*1   POL,BASIS
   13           CHARACTER*30  FILE(4)
   14           COMMON        /DIM/MX,NX
   15           COMMON        /BLK/DFGL,ETA
   16           DATA XJ/(0.,1.)/,PI/3.14159262/,TPI/6.28318524/,Z0/376.99111488/
   17   C    TIMING OVERHEAD CALCULATION
   18   C    TAKE LAST OF THREE TO ELIMINATE CACHE/PAGING EFFECTS
   19   CVD$ NOVECTOR
   20   CVD$ NOCONCUR
   21           DO I=1,3
   22              ETIME1=ETIME(USTIME1)
   23              ETIME2=ETIME(USTIME2)
   24              ETIMEOH=ETIME2-ETIME1
   25              UTIMEOH=USTIME2(1)-USTIME1(1)
   26              STIMEOH=USTIME2(2)-USTIME1(2)
   27           ENDDO
   28   C    --------------
   29   C    INITIALIZATION
   30   C    --------------
   31           OPEN(UNIT=5,FILE='strip.input')
   32           READ(5,'(A30)',END=3,ERR=4)(FILE(I),I=1,4)
   33           READ(5,'(A1)',END=3,ERR=4)POL
   34           READ(5,*,END=3,ERR=4)FMHZ,H0
   35           READ(5,*,END=3,ERR=4)ISCAT,I1PHI,I2PHI,INC
   36           READ(5,*,END=3,ERR=4)PHI0
   37           READ(5,*,END=3,ERR=4)WIDTH,IDIEL,ET,ETC,ETE,TPR
   38           READ(5,*,END=3,ERR=4)MX,IPAD
   39           READ(5,*,END=3,ERR=4)ITMAX,TOL
   40           READ(5,'(A1)',END=3,ERR=4)BASIS
   41           CLOSE(5)
   42           NX=2**(INT(ALOG(2.*MX-1)/ALOG(2.))+IPAD)
   43           WIDTH2=WIDTH/2.
   44           DLTX=WIDTH/MX
   45           DLTFX=1./(DLTX*NX)
   46           NXH=NX/2+1
   47           XL1=300./FMHZ
   48           XK0=2.*PI/XL1
```

```
Alliant FX/Fortran Compiler V4.0.28 (2.24D20)  15 Oct 1988 21:04:05  Page   2
Source Listing of /u/barkeshli/strip.d/strip.f

  49           XK2=XK0*XK0
  50           FACTOR=XK0/2.
  51   C     ----------------------
  52   C    STATEMENT OF THE PROBLEM
  53   C     ----------------------
  54           WRITE(*,*)
  55           WRITE(*,*)'SCATTERING FROM A STRIP'
  56           WRITE(*,*)'FREQUENCY= ',FMHZ,' MHz,   LAMBDA= ',XL1,' m'
  57           WRITE(*,*)'STRIP RELATIVE LENGTH; W/LAMDA=',WIDTH/XL1
  58           IF(IDIEL.EQ.1)THEN
  59             WRITE(*,*)'UNIFORM RESISTIVITY: ETA=',ET
  60           ELSE
  61             WRITE(*,*)'NON-UNIFORM RESISTIVITY:'
  62             WRITE(*,*)'ETA(C)=',ETC,' ETA(E)=',ETE,' TPR=',TPR
  63           ENDIF
  64           WRITE(*,*)POL,'-POLARIZATION',' PHI0=',PHI0
  65           WRITE(*,*)'NO. OF UNKNOWNS MX= ',MX,'  PER LAMBDA= ',MX/WIDTH,
  66        +            ' DLTX/LAMBDA= ',DLTX
  67           WRITE(*,*)'FFT PAD ',NX,' OF ORDER ',IPAD
  68           WRITE(*,*)'SCATTERING PATTERN COMPUTED:'
  69           IF(ISCAT.EQ.1)THEN
  70             WRITE(*,*)' BISTATIC'
  71           ELSE
  72             WRITE(*,*)' BACKSCATTERING'
  73           ENDIF
  74           WRITE(*,*)'TOLERANCE= ',TOL
  75           WRITE(*,*)
  76   C     ----------------
  77   C     SPATIAL VARIABLE
  78   C     ----------------
  79           DO I=1,MX
  80              X(I)=-WIDTH2+(I-.5)*DLTX
  81           ENDDO
  82   C     -----------------
  83   C     STRIP RESISTIVITY
  84   C     -----------------
  85   C     HOMOGENEOUS DIELECTRIC (UNIFORM)
  86           IF(IDIEL.EQ.1)THEN
  87             ETA(1:MX)=ET
  88   C     INHOMOGENUOUS DIELECTRIC (LINEAR OR PARABOLIC)
  89           ELSE
  90             ETA(1:MX)=ETC+ETE*ABS(X(1:MX)/WIDTH2)**TPR
  91           ENDIF
  92   C     ------------------------
  93   C     PHYSICAL OPTICS CURRENTS
  94   C     ------------------------
  95           SPH=SIN(PHI0*PI/180.)
  96           CPH=COS(PHI0*PI/180.)
```

```
Alliant FX/Fortran Compiler V4.0.28 (2.24D20)  15 Oct 1988 21:04:05  Page   3
Source Listing of /u/barkeshli/strip.d/strip.f

 97           IF(POL.EQ.'E')THEN
 98             J(1:MX)=SPH/(.5+ETA(1:MX)*SPH)*H0*CEXP(XJ*X(1:MX)*CPH)
 99           ELSE
100             J(1:MX)=SPH/(ETA(1:MX)+.5*SPH)*H0*CEXP(XJ*X(1:MX)*CPH)
101           ENDIF
102   C     ---------------------------------------------
103   C     SPECTRAL VARIABLES AND TRANSFORM OF THE KERNEL
104   C     ---------------------------------------------
105           DO I = 1,NXH
106              KX(I)=TPI*(I-1)*DLTFX
107           ENDDO
108           DO I = NXH+1,NX
109              KX(I)=TPI*(I-NX-1)*DLTFX
110           ENDDO
111           C(1:NX)=DBLE(XK2)-KX(1:NX)*KX(1:NX)
112   CVD$ CNCALL
113           DO I=1,NX
114              DFGL(I)=FACTOR/MYCSQRT(C(I))
115           ENDDO
116   C       DFGL(1:NX)=FACTOR/MYCSQRT(C(1:NX))
117           IF(POL.EQ.'H')THEN
118             DFGL(1:NX)=SNGL(C(1:NX))/XK2*DFGL(1:NX)
119           ENDIF
120           IF(BASIS.EQ.'C')THEN
121   CVD$ CNCALL
122             DFGL(1:NX)=DFGL(1:NX)*SINC(KX(1:NX)*DLTX/2.)
123           ELSE IF(BASIS.EQ.'S')THEN
124   CVD$ CNCALL
125             DFGL(1:NX)=
126        +    DFGL(1:NX)*2.*XK0*(COS(KX(1:NX)*DLTX)-COS(XK0*DLTX))/
127        +    (DLTX*SNGL(C(1:NX))*SIN(XK0*DLTX))
128           ENDIF
129   C     INITIALIZE THE WORKING AREAS
130           A(1:512)=(0.,0.)
131           W(1:512)=(0.,0.)
132           WORK(1:384)=(0.,0.)
133           OPEN(UNIT=4,FILE=FILE(4))
134           KK=PHI0
135   C     ----------------------------------
136   C     START ON THE NEW ANGLE OF INCIDENCE
137   C     ----------------------------------
138   1       PH0=PHI0*PI/180.
139           ITER=0
140           RSS=0.
141           CPH0=COS(PH0)
142   C     --------------
143   C     INCIDENT FIELD
144   C     --------------
```

```
Alliant FX/Fortran Compiler V4.0.28 (2.24D20)  15 Oct 1988 21:04:05  Page   4
Source Listing of /u/barkeshli/strip.d/strip.f

 145           IF(POL.EQ.'E')THEN
 146             E(1:MX)=H0*CEXP(XJ*XK0*X(1:MX)*CPH0)
 147           ELSE
 148             SPH0=SIN(PH0)
 149             E(1:MX)=H0*SPH0*CEXP(XJ*XK0*X(1:MX)*CPH0)
 150           ENDIF
 151           ENORM=SUM(CABS(E(1:MX))*CABS(E(1:MX)))
 152   C     *-------------------------*
 153   C     BEGIN TIMING THE CGFFT LOOP
 154   C     *-------------------------*
 155           ETIME1=ETIME(USTIME1)
 156   C     --------------------
 157   C     CGFFT INITIALIZATION
 158   C     --------------------
 159           CALL OPERATOR(A,J,W,WORK,-1)
 160           R(1:MX)=A(1:MX)-E(1:MX)
 161           P(1:MX)=(0.,0.)
 162   C     -------------------------
 163   C     CGFFT MAIN ITERATION LOOP
 164   C     -------------------------
 165   2       CONTINUE
 166           CALL OPERATOR(A,R,W,WORK,1)
 167           B=1./SUM(CABS(A(1:MX))*CABS(A(1:MX)))
 168           P(1:MX)=P(1:MX)-B*A(1:MX)
 169           CALL OPERATOR(A,P,W,WORK,-1)
 170           T=1./SUM(CABS(A(1:MX))*CABS(A(1:MX)))
 171           R(1:MX)=R(1:MX)+T*A(1:MX)
 172           J(1:MX)=J(1:MX)+T*P(1:MX)
 173           ITER=ITER+1
 174           RSS=SQRT(SUM(CABS(R(1:MX))*CABS(R(1:MX)))/ENORM)
 175   C     -------------------------------
 176   C     END OF CGFFT MAIN ITERATION LOOP
 177   C     -------------------------------
 178   C       WRITE(*,*)'ITER= ',ITER,'  RSS= ',RSS
 179           WRITE(4,*)ITER,RSS
 180           IF(RSS.LE.TOL)THEN
 181             WRITE(*,*)'CONVERGENCE ACHIEVED'
 182           ELSE IF(ITER.EQ.ITMAX) THEN
 183             WRITE(*,*)'ITMAX EXCEEDED; CONVERGENCE CRITERION NOT MET.'
 184           ELSE
 185             GO TO 2
 186           END IF
 187   C     *----------------------*
 188   C     END TIMING THE CGFFT LOOP
 189   C     *----------------------*
 190           ETIME2=ETIME(USTIME2)
 191   C     --------------------
 192   C     BISTATIC COMPUTATIONS
```

```
Alliant FX/Fortran Compiler V4.0.28 (2.24D20)  15 Oct 1988 21:04:05  Page   5
Source Listing of /u/barkeshli/strip.d/strip.f

 193   C     --------------------
 194   CVD$ CNCALL
 195            IF(ISCAT.EQ.1)THEN
 196              DO K=I1PHI,I2PHI,INC
 197                 PH=K*PI/180.
 198                 CPH=COS(PH)
 199                 SUMIT=(0.,0.)
 200                 DO I=1,MX
 201                    SUMIT=SUMIT+J(I)*CEXP(XJ*XK0*X(I)*CPH)
 202                 ENDDO
 203                 IF(POL.EQ.'E')THEN
 204                   ECHO(K)=(XK0*DLTX*SINC(XK0*DLTX/2.*CPH))**2
 205       +                    *CABS(SUMIT)**2/(8.*PI)
 206                 ELSE
 207                   SPH=SIN(PH)
 208                   ECHO(K)=(XK0*DLTX*SPH*SINC(XK0*DLTX/2.*CPH))**2
 209       +                    *CABS(SUMIT)**2/(8.*PI)
 210                 ENDIF
 211              ENDDO
 212   C     SAVE CURRENT VALUES
 213              OPEN(UNIT=1,FILE=FILE(1))
 214              WRITE(1,'(A,/,I3)')'STRIP SURFACE CURRENT;INTEGRAL EQN',MX
 215              WRITE(1,'(2(1X,F10.4))')(X(I),CABS(J(I)),I=1,MX)
 216              CLOSE(1)
 217   C     COMPUTE AND SAVE CURRENT SPECTRUM
 218              CALL CFFT1D(NX,W,0,WORK)
 219              W(1:MX)=J(1:MX)
 220              W(MX+1:NX)=(0.,0.)
 221              CALL CFFT1D(NX,W,-1,WORK)
 222              OPEN(UNIT=2,FILE=FILE(2))
 223              WRITE(2,'(A,/,I3)')'CURRENT SPECTRUM',NX
 224              WRITE(2,'(2(1X,F10.4))')(KX(I),CABS(W(I)),I=NXH+1,NX)
 225              WRITE(2,'(2(1X,F10.4))')(KX(I),CABS(W(I)),I=1,NXH)
 226              CLOSE(2)
 227   C     --------------------------
 228   C     BACKSCATTERING COMPUTATIONS
 229   C     --------------------------
 230            ELSE IF(ISCAT.EQ.2)THEN
 231              CPH=COS(PH0)
 232              SUMIT=(0.,0.)
 233              DO I=1,MX
 234                 SUMIT=SUMIT+J(I)*CEXP(XJ*XK0*X(I)*CPH)
 235              ENDDO
 236              IF(POL.EQ.'E')THEN
 237                ECHO(KK)=(XK0*DLTX*SINC(XK0*DLTX/2.*CPH))**2
 238       +              *CABS(SUMIT)**2/(8.*PI)
 239              ELSE IF(POL.EQ.'H')THEN
 240                SPH=SIN(PH0)
```

```
Alliant FX/Fortran Compiler V4.0.28 (2.24D20)  15 Oct 1988 21:04:05  Page   6
Source Listing of /u/barkeshli/strip.d/strip.f

 241                ECHO(KK)=(XK0*DLTX*SPH*SINC(XK0*DLTX/2.*CPH))**2
 242         +            *CABS(SUMIT)**2/(8.*PI)
 243              ENDIF
 244              KK=KK+INC
 245              PHI0=PHI0+INC
 246              IF(PHI0.LE.FLOAT(I2PHI))THEN
 247                WRITE(*,*)'ANGLE OF INCIDENCE: ',PHI0
 248                GOTO 1
 249              ENDIF
 250            ENDIF
 251            WHERE(ECHO(I1PHI:I2PHI:INC).NE.0.)
 252              ECHO(I1PHI:I2PHI:INC)=10.*ALOG10(ECHO(I1PHI:I2PHI:INC))
 253            OTHERWISE
 254              ECHO(I1PHI:I2PHI:INC)=1.E-15
 255            END WHERE
 256            OPEN(UNIT=3,FILE=FILE(3))
 257            WRITE(3,*)'STRIP ECHOWIDTH; PHI0= ',PHI0
 258            WRITE(3,*)(I2PHI-I1PHI)/INC+1
 259            WRITE(3,'(2(1X,F10.4))')(FLOAT(K),ECHO(K),K=I1PHI,I2PHI,INC)
 260            CLOSE(3)
 261            CLOSE(4)
 262            WRITE(*,*)'NORMAL TERMINATION'
 263            WRITE(*,*)
 264            PRINT *,'ITERATIONS:    ',ITER
 265            WRITE(*,*)'CPU TIME INFORMATION(msec.):'
 266            PRINT *,'OVERHEADS (TOTAL,USER,SYSTEM)'
 267         +          ,1000.*ETIMEOH,1000.*UTIMEOH,1000.*STIMEOH
 268            PRINT *,'START TIME:    ',1000.*ETIME1,1000.*USTIME1
 269            PRINT *,'END TIME:      ',1000.*ETIME2,1000.*USTIME2
 270            PRINT *,'TOTAL TIME:    ',1000.*(ETIME2-ETIME1-ETIMEOH)
 271            PRINT *,'USER TIME:     ',1000.*(USTIME2(1)-USTIME1(1)-UTIMEOH)
 272            PRINT *,'SYSTEM TIME:   ',1000.*(USTIME2(2)-USTIME1(2)-STIMEOH)
 273            NOPER=(4.*NX*ALOG(FLOAT(NX))/ALOG(2.)+10.*MX+2)*ITER
 274            PRINT *,'MEGAFLOPS:     ',NOPER/((ETIME2-ETIME1-ETIMEOH)*1.0E6)
 275            PRINT *,'USER MFLOPS:   ',NOPER/((USTIME2(1)-USTIME1(1)-UTIMEOH)
 276         +           *1.0E6)
 277            GO TO 5
 278    3       PRINT *,'END OF INPUT FILE; MORE INPUT EXPECTED'
 279    4       PRINT *,'ERROR IN INPUT DATA; CHECK INPUT FILE'
 280    5       STOP
 281            END
```

Alliant FX/Fortran Compiler V4.0.28 (2.24D20) 15 Oct 1988 21:04:05 Page 7
Source Listing of /u/barkeshli/strip.d/strip.f

LOOP SUMMARY FOR ROUTINE STRIP

LABEL	INDEX	START	END	NEST	COMMENT	CD?	DP?	ITERATIONS
	I	21	27	1	NO ACTION REQUESTED			3
	I	79	81	1	VECTOR-CONCURRENT			MX
		87	87	1	VECTOR-CONCURRENT			MX
		90	90	1	VECTOR-CONCURRENT			MX
		98	98	1	VECTOR-CONCURRENT			MX
		100	100	1	VECTOR-CONCURRENT			MX
	I	105	107	1	VECTOR-CONCURRENT			NXH
	I	108	110	1	VECTOR-CONCURRENT			NX-NXH
		111	111	1	VECTOR-CONCURRENT			NX
	I	113	115	1	CONCURRENT			NX
		118	118	1	VECTOR-CONCURRENT			NX
		125	125	1	VECTOR-CONCURRENT			NX
		130	131	1	VECTOR-CONCURRENT			512
		132	132	1	VECTOR-CONCURRENT			384
		146	146	1	VECTOR-CONCURRENT			MX
		149	149	1	VECTOR-CONCURRENT			MX
		151	151	1	VECTOR-CONCURRENT			MX
		160	161	1	VECTOR-CONCURRENT			MX
		167	167	1	VECTOR-CONCURRENT			MX
		168	168	1	VECTOR-CONCURRENT			MX
		170	170	1	VECTOR-CONCURRENT			MX
		171	172	1	VECTOR-CONCURRENT			MX
		174	174	1	VECTOR-CONCURRENT			MX
	K	196	211	1	CONCURRENT OUTER			
	I	200	202	2	VECTORIZED			MX
		219	219	1	VECTOR-CONCURRENT			MX
		220	220	1	VECTOR-CONCURRENT			NX-MX
	I	233	235	1	VECTOR-CONCURRENT			MX
		251	255	1	VECTOR-CONCURRENT	Y		

1 VECTOR	1 CONCURRENT	25 VECT-CONCR	1 CON. OUTER
1 NOT OPTMZD	0 NOT INNER	0 TOO SHORT	0 DEPENDENT
29 TOTAL	29 EXAMINED	28 OPTIMIZED	

Alliant FX/Fortran Compiler V4.0.28 (2.24D20) 15 Oct 1988 21:04:05 Page 8
Source Listing of /u/barkeshli/strip.d/strip.f

```
C   ***********************************************************************
C                        INTEGRO-DIFFERENTIAL OPERATOR
C   ***********************************************************************
        SUBROUTINE  OPERATOR(A,Z,W,WORK,ISTAR)
        COMPLEX     A(*),Z(*),DFGL(512),W(*),WORK(*)
        REAL        ETA(512)
        COMMON      /DIM/MX,NX
        COMMON      /BLK/DFGL,ETA
C    ------------------
C    FFT INITIALIZATION
C    ------------------
        CALL CFFT1D(NX,W,0,WORK)
C    -----------
C    FORWARD FFT
C    -----------
        W(1:MX)=Z(1:MX)
        W(MX+1:NX)=(0.,0.)
        CALL CFFT1D(NX,W,-1,WORK)
C    --------------------
C    DISCRETE CONVOLUTION
C    --------------------
        IF(ISTAR.EQ.-1)THEN
          W(1:NX)=DFGL(1:NX)*W(1:NX)
        ELSE
          W(1:NX)=CONJG(DFGL(1:NX))*W(1:NX)
        ENDIF
C    -----------
C    INVERSE FFT
C    -----------
        CALL CFFT1D(NX,W,1,WORK)
        A(1:MX)=ETA(1:MX)*Z(1:MX)+W(1:MX)
        RETURN
        END
```

```
Alliant FX/Fortran Compiler V4.0.28 (2.24D20)  15 Oct 1988 21:04:05  Page   9
Source Listing of /u/barkeshli/strip.d/strip.f


------------------    LOOP SUMMARY FOR ROUTINE OPERATOR------------------
 LABEL   INDEX    START    END NEST    COMMENT          CD? DP?  ITERATIONS

                    297    297  1  VECTOR-CONCURRENT             MX
                    298    298  1  VECTOR-CONCURRENT             NX-MX
                    304    304  1  VECTOR-CONCURRENT             NX
                    306    306  1  VECTOR-CONCURRENT             NX
                    312    312  1  VECTOR-CONCURRENT             MX

   0 VECTOR          0 CONCURRENT     5 VECT-CONCR     0 CON. OUTER
   0 NOT OPTMZD      0 NOT INNER      0 TOO SHORT      0 DEPENDENT
   5 TOTAL           5 EXAMINED       5 OPTIMIZED
```

```
Alliant FX/Fortran Compiler V4.0.28 (2.24D20)  15 Oct 1988 21:04:05  Page  10
Source Listing of /u/barkeshli/strip.d/strip.f

  315   C   ***************************************************************
  316   C                     SQUARE ROOT OF A SIGNED REAL NUMBER
  317   C   ***************************************************************
  318           COMPLEX FUNCTION MYCSQRT(A)
  319           REAL*8  A
  320           COMPLEX XJ
  321           DATA    XJ/(0.,1.)/
  322           IF(A.GE.0D0)THEN
  323             MYCSQRT=SNGL(DSQRT(A))
  324           ELSE
  325             MYCSQRT=XJ*SNGL(DSQRT(-A))
  326           ENDIF
  327           RETURN
  328           END
```

```
Alliant FX/Fortran Compiler V4.0.28 (2.24D20)  15 Oct 1988 21:04:05  Page  11
Source Listing of /u/barkeshli/strip.d/strip.f

  329   C   ***********************************************************************
  330   C                                      SINC
  331   C   ***********************************************************************
  332           REAL FUNCTION SINC(A)
  333           IF(A.NE.0)THEN
  334             SINC=SIN(A)/A
  335           ELSE
  336             SINC=1.
  337           ENDIF
  338           RETURN
  339           END
```

```
Alliant FX/Fortran Compiler V4.0.28 (2.24D20)   5 Dec 1988 02:10:39  Page   1
Source Listing of /u/barkeshli/slab.d/platevc.f

      1          PROGRAM SLAB
      2     C   ********************************************************************
      3     C   *SOLUTION OF A THIN DIELECTRIC SLAB BY COMPLEX CONJUGATE-FFT METHOD*
      4     C   *  VECTOR-CONCURRENT IMPLEMENTATION ON ALLIANT SUPER-MINI COMPUTER *
      5     C   ********************************************************************
      6     C                                MAIN PROGRAM
      7     C   ********************************************************************
      8     C                           PARAMETER DESCRIPTION
      9     C   ====================================================================
     10     C       DLX,DLY...................RELATIVE LENGTH IN EACH DIMENSION
     11     C                                 IN TERMS OF LAMBDA
     12     C       UPL.......................NO. OF UNKNOWNS PER WAVELENGTH
     13     C       MX,MY.....................NO. OF UNKNOWNS PER DIMENSION
     14     C       NX,NY.....................COMPOSITE NO. OF SAMPLES IN  EACH
     15     C                                 DIMENSION SUCH THAT :
     16     C                                 N > 2*M | N = 2**L , L-INTEGER
     17     C       E.........................INCIDENT FIELD VECTOR
     18     C       JS........................SURFACE CURRENT VECTOR
     19     C       A.........................INTEGRO-DIFFERENTIAL OPERATOR MX
     20     C       ER........................SLAB PERMITTIVITY
     21     C       TK........................SLAB RELATIVE THICKNESS
     22     C       R,P,T,B...................CG VARIABLES
     23     C       ITMAX.....................MAX. NO. OF ITERATIONS
     24     C       TOL.......................CONVERGENCE TOLERANCE
     25     C       RSS.......................RELATIVE RESIDUAL ERROR
     26     C       ZT,W......................WORKING AREA
     27     C       ISORCE....................SOURCE TYPE
     28     C                                    1 = E-POL
     29     C                                    2 = H-POL
     30     C       TETA,PHI..................ANGLES OF INCIDENCE FOR PLANE WAVE
     31     C                                       0 < TETA < PI
     32     C                                       0 <  PHI < 2*PI
     33     C       PSI.......................POLARIZATION ANGLE
     34     C           PLANE WAVE INCIDENCE     PSI = 90 E-POL, IDIM=2
     35     C                                        =  0 H-POL, IDIM=3
     36     C   ====================================================================
     37           REAL    ETIME1,ETIME2,ETIME3,ETIMEOH
     38           REAL    USTIME1(2),USTIME2(2),USTIME3(2),UTIMEOH,STIMEOH
     39           INTEGER IRING(:)
     40           REAL    DFG(:)
     41           COMPLEX JS(:),E(:),A(:),R(:),P(:)
     42           COMPLEX ZT(:),AT(:),W(:)
     43           COMPLEX XJ,RS,CRS,FCT,ER
     44           COMMON  /DIM/MX,MY,MT,MTI,NX,NY,NXH,NYH,NT,NTI,NG,NA,DLT,XK1,Z0
     45           COMMON  /PAR/RS,CRS,FCT
     46           COMMON  IBASIS
     47           DATA    PI/3.141592653589793/,TPI/6.28318530717959/,XJ/(0.,1.0)/
     48     C     ---------------------------
```

```
Alliant FX/Fortran Compiler V4.0.28 (2.24D20)   5 Dec 1988 02:10:39  Page   2
Source Listing of /u/barkeshli/slab.d/platevc.f

   49   C     ESTIMATE THE TIMING OVERHEAD
   50   C     ---------------------------
   51   CVD$ NOVECTOR
   52   CVD$ NOCONCUR
   53            DO I=1,3
   54               ETIME1=ETIME(USTIME1)
   55               ETIME2=ETIME(USTIME2)
   56               ETIMEOH=ETIME2-ETIME1
   57               UTIMEOH=USTIME2(1)-USTIME1(1)
   58               STIMEOH=USTIME2(2)-USTIME1(2)
   59            ENDDO
   60   C     ------------
   61   C     START TIMING
   62   C     ------------
   63            ETIME1=ETIME(USTIME1)
   64   C     ==============
   65   C     INITIALIZATION
   66   C     ==============
   67            OPEN(UNIT=10)
   68            READ(10,*)E0
   69            READ(10,*)TETA,PHI,PSI,TMAX,INCT
   70            READ(10,*)DLX,DLY,UPL,IPAD
   71            READ(10,*)ER,TK
   72            READ(10,*)IBASIS
   73            READ(10,*)ITMAX,ITMIN,TOL
   74            CLOSE(10)
   75            Z0=120.*PI
   76            XL1=1.
   77            DLT=XL1/UPL
   78            XK1=TPI/XL1
   79            MX=DLX*UPL
   80            MY=DLY*UPL
   81            MT=MX*MY
   82            MTI=2*MT
   83            NX=2**(INT(ALOG(2.*MX-1)/ALOG(2.))+IPAD)
   84            NY=2**(INT(ALOG(2.*MY-1)/ALOG(2.))+IPAD)
   85            NXH=NX/2+1
   86            NYH=NY/2+1
   87            NT=NX*NY
   88            NTI=2*NT
   89            NG=4*NT
   90            NA=NX+3*NY/4+1
   91            NP=(TMAX-TETA)/INCT+1
   92            IFILOFF=12
   93            RS=-XJ*Z0/((ER-1)*XK1*TK)
   94            CRS=CONJG(RS)
   95            FCT=XJ*Z0/XK1
   96   C     ---------------
```

```
Alliant FX/Fortran Compiler V4.0.28 (2.24D20)   5 Dec 1988 02:10:39  Page   3
Source Listing of /u/barkeshli/slab.d/platevc.f

  97    C     ALLOCATE ARRAYS
  98    C     ---------------
  99             ALLOCATE(IRING(MT),DFG(NG))
 100             ALLOCATE(JS(MTI),E(MTI),A(MTI),R(MTI),P(MTI))
 101             ALLOCATE(ZT(NTI),AT(NTI),W(NT))
 102    C     -------------
 103    C     FORCE CACHING
 104    C     -------------
 105             IRING(1:MT)=0
 106             DFG(1:NG)=0.
 107             JS(1:MTI) = (0.,0.)
 108             E(1:MTI) = (0.,0.)
 109             A(1:MTI) = (0.,0.)
 110             R(1:MTI) = (0.,0.)
 111             P(1:MTI) = (0.,0.)
 112             ZT(1:NTI) = (0.,0.)
 113             AT(1:NTI) = (0.,0.)
 114             W(1:NT) = (0.,0.)
 115    C     ================================================
 116    C     COMPUTE THE TRANSFORM OF THE GREEN'S FUNCTION
 117    C     ================================================
 118             CALL DIFREQ(DFG,IRING,IRMAX)
 119             OPEN(UNIT=11)
 120    10       CONTINUE
 121             ITER = 0
 122             RSS = 0.
 123             WRITE(11,200)TETA,PHI,PSI
 124    C     ============================
 125    C     CALCULATE THE INCIDENT FIELD
 126    C     ============================
 127             CALL INCIDENCE(E,E0,TETA,PHI,PSI)
 128             ENORM=SUM(CABS(E(1:MTI))*CABS(E(1:MTI)))
 129    C     --------------------------
 130    C     BEGIN TIMING THE CGFFT LOOP
 131    C     --------------------------
 132             ETIME2=ETIME(USTIME2)
 133    C     ================================
 134    C     ZEROTH ORDER RESIDUAL CALCULATION
 135    C     ================================
 136             CALL OPERAT(A,-1,JS,ZT,DFG,IRING,IRMAX,AT,W)
 137             R(1:MTI) = A(1:MTI)-E(1:MTI)
 138             P(1:MTI) = (0.,0.)
 139    C     ===============
 140    C     CGFFT MAIN LOOP
 141    C     ===============
 142    30       CONTINUE
 143             CALL OPERAT(A,1,R,ZT,DFG,IRING,IRMAX,AT,W)
 144             B=1./SUM(CABS(A(1:MTI))*CABS(A(1:MTI)))
```

```
Alliant FX/Fortran Compiler V4.0.28 (2.24D20)   5 Dec 1988 02:10:39  Page   4
Source Listing of /u/barkeshli/slab.d/platevc.f

 145           P(1:MTI) = P(1:MTI)-B*A(1:MTI)
 146           CALL OPERAT(A,-1,P,ZT,DFG,IRING,IRMAX,AT,W)
 147           T=1./SUM(CABS(A(1:MTI))*CABS(A(1:MTI)))
 148           R(1:MTI) = R(1:MTI)+T*A(1:MTI)
 149           JS(1:MTI) = JS(1:MTI)+T*P(1:MTI)
 150           ITER = ITER+1
 151           RSS = SQRT(SUM(R(1:MTI)*R(1:MTI))/ENORM)
 152           WRITE(11,300)ITER,RSS
 153           IF((RSS.LE.TOL).AND.(ITER.GT.ITMIN)) THEN
 154             WRITE(11,400)
 155           ELSE IF(ITER.GE.ITMAX) THEN
 156             WRITE(11,500)
 157           ELSE
 158             GO TO 30
 159           END IF
 160    90     CONTINUE
 161   C     ------------------------
 162   C     END TIMING THE CGFFT LOOP
 163   C     ------------------------
 164           ETIME3=ETIME(USTIME3)
 165           CALL OUTPUT(IFILOFF,1,JS,ZT,TETA,SIG,ITER,RSS)
 166           IFILOFF=IFILOFF+2
 167           TETA = TETA+INCT
 168           IF(TETA.LE.TMAX) THEN
 169             GO TO 10
 170           ENDIF
 171   C     -----------------
 172   C     DEALLOCATE ARRAYS
 173   C     -----------------
 174           DEALLOCATE(W,AT,ZT,P,R,A,E,JS,DFG,IRING)
 175           PRINT *,'USER CPU TIME INFORMATION (sec)'
 176           PRINT *,'------------------------------'
 177           PRINT *,'INITIALIZATION..',USTIME2(1)-USTIME1(1)-UTIMEOH
 178           PRINT *,'CGFFT LOOP......',USTIME3(1)-USTIME2(1)-UTIMEOH
 179           PRINT *,'TOTAL...........',USTIME3(1)-USTIME1(1)-UTIMEOH
 180           PRINT *,'ITERATIONS......',ITER
 181           NOPER=(4.*NTI*ALOG(FLOAT(NTI))/ALOG(2.)+10.*MTI+2)*ITER
 182           PRINT *,'MEGAFLOPS.......',NOPER/((USTIME3(1)-USTIME1(1)
 183          +  -UTIMEOH)*1.0E6)
 184           WRITE(11,*)'PROGRAM TERMINATED.'
 185           CLOSE(11)
 186           WRITE(*,*)'PROGRAM TERMINATED.'
 187    200    FORMAT(29('='),/,'TETA=',F4.1,
 188          +  2X,'PHI=',F4.1,2X,'PSI=',F4.1,/, 'ITER',7X,'RSS(R)',/,29('-'))
 189    300    FORMAT(I4,1X,G13.4)
 190    400    FORMAT('CONVERGENCE  ACHIEVED.')
 191    500    FORMAT('ITMAX EXCEEDED; CONVERGENCE CRITERION NOT MET.')
 192    600    FORMAT(1X,'TETA',3X,'SIG0 dB',
```

```
Alliant FX/Fortran Compiler V4.0.28 (2.24D20)   5 Dec 1988 02:10:39  Page   5
Source Listing of /u/barkeshli/slab.d/platevc.f

 193        +   2X,'SIG0(PO) dB',2X,'PHI=',F5.1,2X,'PSI=',F5.1,/,I2)
 194            STOP
 195            END
```

Alliant FX/Fortran Compiler V4.0.28 (2.24D20) 5 Dec 1988 02:10:39 Page 6
Source Listing of /u/barkeshli/slab.d/platevc.f

LOOP SUMMARY FOR ROUTINE SLAB

LABEL	INDEX	START	END	NEST	COMMENT	CD?	DP?	ITERATIONS
	I	53	59	1	NO ACTION REQUESTED			3
		105	105	1	VECTOR-CONCURRENT			MT
		106	106	1	VECTOR-CONCURRENT			NG
		107	111	1	VECTOR-CONCURRENT			MTI
		112	113	1	VECTOR-CONCURRENT			NTI
		114	114	1	VECTOR-CONCURRENT			NT
		128	128	1	VECTOR-CONCURRENT			MTI
		137	138	1	VECTOR-CONCURRENT			MTI
		144	144	1	VECTOR-CONCURRENT			MTI
		145	145	1	VECTOR-CONCURRENT			MTI
		147	147	1	VECTOR-CONCURRENT			MTI
		148	149	1	VECTOR-CONCURRENT			MTI
		151	151	1	VECTOR-CONCURRENT			MTI

0 VECTOR	0 CONCURRENT	12 VECT-CONCR	0 CON. OUTER
1 NOT OPTMZD	0 NOT INNER	0 TOO SHORT	0 DEPENDENT
13 TOTAL	13 EXAMINED	12 OPTIMIZED	

```
Alliant FX/Fortran Compiler V4.0.28 (2.24D20)   5 Dec 1988 02:10:39  Page   7
Source Listing of /u/barkeshli/slab.d/platevc.f

 196   C   ******************************************************************
 197   C                                    OUTPUT
 198   C   ******************************************************************
 199           SUBROUTINE OUTPUT(IUNT,IFLG,Z,ZT,TETA,SIG,ITER,RSS)
 200           COMPLEX Z(MTI),ZT(NTI)
 201           COMMON  /DIM/MX,MY,MT,MTI,NX,NY,NXH,NYH,NT,NTI,NG,NA,DLT,XK1,Z0
 202   CVD$ NOVECTOR
 203   CVD$ NOCONCUR
 204           DO 10 I = 1,2
 205              IUNTI = I+IUNT-1
 206              OPEN(UNIT=IUNTI)
 207              WRITE(IUNTI,*)TETA,S,ITER,RSS
 208              WRITE(IUNTI,*)MX,MY
 209              WRITE(IUNTI,*)(J,J=1,MX)
 210              WRITE(IUNTI,*)(K,K=1,MY)
 211              L1=(I-1)*MT+1
 212              L2=I*MT
 213              WRITE(IUNTI,20)(CABS(Z(L)),L=L1,L2)
 214              CLOSE(IUNTI)
 215   10      CONTINUE
 216   20      FORMAT(10(1X,G10.4))
 217           RETURN
 218           END
```

Alliant FX/Fortran Compiler V4.0.28 (2.24D20) 5 Dec 1988 02:10:39 Page 8
Source Listing of /u/barkeshli/slab.d/platevc.f

```
------------------    LOOP SUMMARY FOR ROUTINE OUTPUT  -------------------
 LABEL  INDEX   START   END NEST    COMMENT          CD? DP?  ITERATIONS

    10   I       204   215  1  NO ACTION REQUESTED           2

   0 VECTOR        0 CONCURRENT    0 VECT-CONCR    0 CON. OUTER
   1 NOT OPTMZD    0 NOT INNER     0 TOO SHORT     0 DEPENDENT
   1 TOTAL         1 EXAMINED      0 OPTIMIZED
```

Alliant FX/Fortran Compiler V4.0.28 (2.24D20) 5 Dec 1988 02:10:39 Page 9
Source Listing of /u/barkeshli/slab.d/platevc.f

```
219   C   ***********************************************************************
220   C                                INCIDENT FIELD
221   C   ***********************************************************************
222   C                            PARAMETER DESCRIPTION
223   C   =======================================================================
224   C   ISORCE.............................SOURCE TYPE:
225   C                                       1 : E-POL
226   C                                       2 : H-POL
227   C   E0..................................AMPLITUDE OF THE PLANE WAVE
228   C   TETA,PHI............................ORIENTATION OF THE PLANE WAVE OR
229   C                                       ORIENTATION OF THE ANTENN    XIS
230   C   PSI.................................POLARIZATION ANGLE FOR
231   C                                       PLANE WAVE
232   C   =======================================================================
233           SUBROUTINE INCIDENCE(E,E0,TETA,PHI,PSI)
234           COMPLEX E(MTI),XJ
235           REAL    XP(MX),YP(MY)
236           COMMON  /DIM/MX,MY,MT,MTI,NX,NY,NXH,NYH,NT,NTI,NG,NA,DLT,XK1,Z0
237           DATA    PI/3.141592653589793/,XJ/(0.,1.)/
238           MXO = MX/2+1
239           MYO = MY/2+1
240           TET = TETA*PI/180.
241           PH  = PHI*PI/180.
242           PS  = PSI*PI/180.
243           ECS=E0*(COS(PS)*COS(TET)*COS(PH)-SIN(PS)*SIN(PH))
244           ESC=E0*(COS(PS)*COS(TET)*SIN(PH)+SIN(PS)*COS(PH))
245           XI = XK1*SIN(TET)*COS(PH)
246           YI = XK1*SIN(TET)*SIN(PH)
247   CVD$R NODEPCHK
248           DO 10 K=1,MY
249              YP(K)=(K-MYO)*DLT
250              DO 20 J=1,MX
251                 XP(J)=(J-MXO)*DLT
252                 L1 = (K-1)*MX+J
253                 E(L1)=ECS*CEXP(XJ*(XI*XP(J)+YI*YP(K)))
254                 E(L2)=ESC*CEXP(XJ*(XI*XP(J)+YI*YP(K)))
255   20         CONTINUE
256   10      CONTINUE
257           RETURN
258           END
```

Alliant FX/Fortran Compiler V4.0.28 (2.24D20) 5 Dec 1988 02:10:39 Page 10
Source Listing of /u/barkeshli/slab.d/platevc.f

```
------------------     LOOP SUMMARY FOR ROUTINE INCIDENC-------------------
 LABEL  INDEX    START    END NEST    COMMENT          CD? DP?  ITERATIONS

    10   K        248    256  1  CONCURRENT OUTER              MY
    20   J        250    255  2  VECTORIZED                    MX

   1 VECTOR          0 CONCURRENT     0 VECT-CONCR     1 CON. OUTER
   0 NOT OPTMZD      0 NOT INNER      0 TOO SHORT      0 DEPENDENT
   2 TOTAL           2 EXAMINED       2 OPTIMIZED
```

Alliant FX/Fortran Compiler V4.0.28 (2.24D20) 5 Dec 1988 02:10:39 Page 11
Source Listing of /u/barkeshli/slab.d/platevc.f

```
259   C   ***********************************************************************
260   C                        TRANSFORM OF THE EFFECTIVE KERNEL
261   C   ***********************************************************************
262           SUBROUTINE  DIFREQ(DFG,IRING,IRMAX)
263           INTEGER IRING(MT)
264           REAL*8  X1(NT),X2,G,KX(NX),KY(NY),FRX,FRY,C11,C121,C22,TPI,XK2
265           REAL DFG(NG)
266           COMMON /DIM/MX,MY,MT,MTI,NX,NY,NXH,NYH,NT,NTI,NG,NA,DLT,XK1,Z0
267           COMMON  IBASIS
268           DATA    PI/3.141592653589793/,TPI/6.28318530717959D0/
269           XK2=DBLE(XK1*XK1)
270           FRX=1.D0/DBLE(DLT*NX)
271           FRY=1.D0/DBLE(DLT*NY)
272   C     ==============================
273   C     COMPUTE THE SPECTRAL VARIABLES
274   C     ==============================
275           DO 10 J=1,NXH
276              KX(J)=TPI*(J-1)*FRX
277   10      CONTINUE
278           DO 20 J=NXH+1,NX
279              KX(J)=TPI*(J-NX-1)*FRX
280   20      CONTINUE
281           DO 30 J=1,NYH
282              KY(J)=TPI*(J-1)*FRY
283   30      CONTINUE
284           DO 40 J=NYH+1,NY
285              KY(J)=TPI*(J-NY-1)*FRY
286   40      CONTINUE
287   C     ================================
288   C     TRANSFORM OF THE GREEN'S FUNCTION
289   C     ================================
290   CVD$ NODEPCHK
291           DO 50 K=1,NY
292              C22=XK2-KY(K)**2
293              L=(K-1)*NX
294   CVD$ NODEPCHK
295              DO 60 J=1,NX
296                 L1=L+J
297                 L2=NT+L1
298                 L3=NT+L2
299                 L4=NT+L3
300                 C11=XK2-KX(J)**2
301                 C121=-KX(J)*KY(K)
302                 X1(L1) = KX(J)**2+KY(K)**2-XK2
303                 X2=DABS(X1(L1))
304                 G = 1.D0/(2.*DSQRT(X2))
305                 DFG(L1)=SNGL(C11*G)
306                 DFG(L2)=SNGL(C121*G)
```

Alliant FX/Fortran Compiler V4.0.28 (2.24D20) 5 Dec 1988 02:10:39 Page 12
Source Listing of /u/barkeshli/slab.d/platevc.f

```
307                  DFG(L3)=SNGL(C121*G)
308                  DFG(L4)=SNGL(C22*G)
309   60          CONTINUE
310   50       CONTINUE
311   C     ==================================
312   C     IDENTIFY THE GREEN'S FUNCTION RING
313   C     ==================================
314           INDX=0
315   CVD$ NOVECTOR
316   CVD$ NOCONCUR
317           DO 70 L1=1,NT
318              IF(X1(L1).GT.0.D0)THEN
319              ELSE
320                INDX=INDX+1
321                IRING(INDX)=L1
322              END IF
323   70      CONTINUE
324           IRMAX=2*INDX
325   CVD$ NODEPCHK
326           IRING(INDX+1:IRMAX)=IRING(1:INDX)+NT
327           RETURN
328           END
```

Alliant FX/Fortran Compiler V4.0.28 (2.24D20) 5 Dec 1988 02:10:39 Page 13
Source Listing of /u/barkeshli/slab.d/platevc.f

LOOP SUMMARY FOR ROUTINE DIFREQ

LABEL	INDEX	START	END	NEST	COMMENT	CD?	DP?	ITERATIONS
10	J	275	277	1	VECTOR-CONCURRENT			NXH
20	J	278	280	1	VECTOR-CONCURRENT			NX-NXH
30	J	281	283	1	VECTOR-CONCURRENT			NYH
40	J	284	286	1	VECTOR-CONCURRENT			NY-NYH
50	K	291	310	1	CONCURRENT OUTER			NY
60	J	295	309	2	VECTORIZED			NX
70	L1	317	323	1	NO ACTION REQUESTED			NT
		326	326	1	VECTOR-CONCURRENT			INDX

1 VECTOR	0 CONCURRENT	5 VECT-CONCR	1 CON. OUTER
1 NOT OPTMZD	0 NOT INNER	0 TOO SHORT	0 DEPENDENT
8 TOTAL	8 EXAMINED	7 OPTIMIZED	

Alliant FX/Fortran Compiler V4.0.28 (2.24D20) 5 Dec 1988 02:10:39 Page 14
Source Listing of /u/barkeshli/slab.d/platevc.f

```
C   ***************************************************************
C                          CONVOLUTION & DIFFERENTIATION
C   ***************************************************************
        SUBROUTINE  OPERAT(A,ISIGN,Z,ZT,DFG,IRING,IRMAX,AT,W)
        REAL    DFG(NG)
        INTEGER IRING(MT)
        COMPLEX XJ,A(MTI),Z(MTI),ZT(NTI),AT(NTI),W(NT)
        COMPLEX RS,CRS,FCT
        COMMON  /DIM/MX,MY,MT,MTI,NX,NY,NXH,NYH,NT,NTI,NG,NA,DLT,XK1,Z0
        COMMON  /PAR/RS,CRS,FCT
        COMMON  IBASIS
        DATA    XJ/(0.,1.)/
C    ========================
C    FORWARD FOURIER TRANSFORM
C    ========================
        CALL SPECTRA (Z,ZT,-1,W)
C    ================================================
C    CARRY OUT THE CONVOLUTIONS IN THE TRANSFORM DOMAIN
C    ================================================
        DO 10 I=1,NT
           AT(I)=DFG(I)*ZT(I)+DFG(I+NT)*ZT(I+NT)
10      CONTINUE
        DO 20 I=NT+1,NTI
           AT(I)=DFG(I+NT)*ZT(I-NT)+DFG(I+NTI)*ZT(I)
20      CONTINUE
C    ============================================
C    CHOOSE THE RIGHT BRANCH OF THE GREEN'S FUNCTION
C    ============================================
CVD$ PERMUTATION(IRING)
        AT(IRING(1:IRMAX))=ISIGN*XJ*AT(IRING(1:IRMAX))
C    ========================
C    INVERSE FOURIER TRANSFORM
C    ========================
        CALL SPECTRA (A,AT,1,W)
        IF(ISIGN.EQ.1)THEN
          A(1:MTI)=CRS*Z(1:MTI)-FCT*A(1:MTI)
        ELSE
          A(1:MTI)=RS*Z(1:MTI)+FCT*A(1:MTI)
        ENDIF
        RETURN
        END
```

Alliant FX/Fortran Compiler V4.0.28 (2.24D20) 5 Dec 1988 02:10:39 Page 15
Source Listing of /u/barkeshli/slab.d/platevc.f

LOOP SUMMARY FOR ROUTINE OPERAT

LABEL	INDEX	START	END	NEST	COMMENT	CD?	DP?	ITERATIONS
10	I	348	350	1	VECTOR-CONCURRENT			NT
20	I	351	353	1	VECTOR-CONCURRENT			NTI-NT
		358	358	1	VECTOR-CONCURRENT			IRMAX
		364	364	1	VECTOR-CONCURRENT			MTI
		366	366	1	VECTOR-CONCURRENT			MTI

0 VECTOR	0 CONCURRENT	5 VECT-CONCR	0 CON. OUTER
0 NOT OPTMZD	0 NOT INNER	0 TOO SHORT	0 DEPENDENT
5 TOTAL	5 EXAMINED	5 OPTIMIZED	

```
Alliant FX/Fortran Compiler V4.0.28 (2.24D20)   5 Dec 1988 02:10:39  Page  16
Source Listing of /u/barkeshli/slab.d/platevc.f

 370    C   ***********************************************************************
 371    C                          FOURIER TRANSFORM VIA FFT
 372    C   ***********************************************************************
 373            SUBROUTINE SPECTRA(Z,ZT,ISIGN,W)
 374            COMPLEX Z(MTI),ZT(NTI),W(NT)
 375            COMPLEX AUX(NA)
 376            COMMON  /DIM/MX,MY,MT,MTI,NX,NY,NXH,NYH,NT,NTI,NG,NA,DLT,XK1,Z0
 377    C     ==========================
 378    C     INITIALIZE FFT PARAMETERS
 379    C     ==========================
 380            CALL CFFT2D(NX,NY,W,NX,0,AUX)
 381            IF(ISIGN.EQ.-1)THEN
 382    CVD$ NOCONCUR
 383              DO 10 I = 1,2
 384    C     ================================
 385    C     PRECONDITIONING IN SPATIAL DOMAIN
 386    C     ================================
 387                  W(1:NT)=(0.,0.)
 388                  L1 = (I-1)*MT
 389    CVD$ NODEPCHK
 390                  DO 20 K = 1,MY
 391                     L2 = L1+(K-1)*MX
 392                     L3 = (K-1)*NX
 393    CVD$ NODEPCHK
 394                     DO 30 J = 1,MX
 395                        W(L3+J) = Z(L2+J)
 396    30               CONTINUE
 397    20            CONTINUE
 398    C     ===========
 399    C     FOREARD FFT
 400    C     ===========
 401                  CALL CFFT2D(NX,NY,W,NX,-1,AUX)
 Line 401  Informational message # 1342
              Concurrent and vector loop optimization inhibited by
              subroutine call without Cncall directive -- CFFT2D
 402                  L1 = (I-1)*NT
 403                  DO 40 J = 1,NT
 404                     ZT(L1+J) = W(J)
 405    40            CONTINUE
 406    10          CONTINUE
 407             ELSE IF(ISIGN.EQ.1)THEN
 408    CVD$ NOCONCUR
 409               DO 50 I = 1,2
 410                  L1 = (I-1)*NT
 411                  DO 60 J = 1,NT
 412                     W(J) = ZT(L1+J)
 413    60            CONTINUE
 414    C     ===========
```

```
Alliant FX/Fortran Compiler V4.0.28 (2.24D20)   5 Dec 1988 02:10:39  Page  17
Source Listing of /u/barkeshli/slab.d/platevc.f

 415    C     INVERSE FFT
 416    C     ===========
 417                  CALL CFFT2D(NX,NY,W,NX,1,AUX)
 Line 417  Informational message # 1342
              Concurrent and vector loop optimization inhibited by
              subroutine call without Cncall directive -- CFFT2D
 418                  L1 = (I-1)*MT
 419    CVD$ NODEPCHK
 420                  DO 70 K = 1,MY
 421                     L2 = (K-1)*NX
 422                     L3 = L1+(K-1)*MX
 423    CVD$ NODEPCHK
 424                     DO 80 J = 1,MX
 425                        Z(L3+J) = W(L2+J)
 426    80               CONTINUE
 427    70            CONTINUE
 428    50          CONTINUE
 429              END IF
 430              RETURN
 431              END
```

Alliant FX/Fortran Compiler V4.0.28 (2.24D20) 5 Dec 1988 02:10:39 Page 18
Source Listing of /u/barkeshli/slab.d/platevc.f

---------- LOOP SUMMARY FOR ROUTINE SPECTRA ----------

LABEL	INDEX	START	END	NEST	COMMENT	CD?	DP?	ITERATIONS
10	I	383	406	1	NOT OPTIMIZED			2
		387	387	2	VECTOR-CONCURRENT			NT
20	K	390	397	2	CONCURRENT OUTER			MY
30	J	394	396	3	VECTORIZED			MX
40	J	403	405	2	VECTOR-CONCURRENT			NT
50	I	409	428	1	NOT OPTIMIZED			2
60	J	411	413	2	VECTOR-CONCURRENT			NT
70	K	420	427	2	CONCURRENT OUTER			MY
80	J	424	426	3	VECTORIZED			MX

2 VECTOR	0 CONCURRENT	3 VECT-CONCR	2 CON. OUTER
0 NOT OPTMZD	2 NOT INNER	0 TOO SHORT	0 DEPENDENT
9 TOTAL	7 EXAMINED	7 OPTIMIZED	

CORNELLF

USERID: M9OY ORIGIN: CORNELLF CREATED: 07/21/88 16:17:17

FILENAME: SLABVTM LISTING CLASS: A FORMAT:J

SPOOLID: 2110 RECS: 711 COPY: 1 DUPLICATE: 1

PRINTED AT: CORNELLD ID: OLNY AT: 07/21/88 16:17:39

```
************************************************************
*
* THIS FILE WAS SENT BY THE COMMAND:
* PRT3812  SLABVTM LISTING A (OLIN
*
************************************************************
```

ROGRAM HAS TERMINATED; RC (0)
ORTIAD
ROGRAM SAMPLING INTERVAL WAS 4 MS; TOTAL NUMBER OF SAMPLES WAS 5279.
IRECT SAMPLES:

ROGRAM UNIT		SAMPLES	%TOTAL	
LAB		104	1.97	
UTPUT		2	0.04	
NCIDENCE		2	0.04	
NORM		53	1.00	
RING		2	0.04	
IFREQ		9	0.17	
PERAT		683	12.94	***
PECTRA		452	8.56	**
#SQRT		5	0.09	
FSCOCS#		1	0.02	
C#EXP		0	0.00	
C#ABS		296	5.61	*
41$		1657	31.39	******
CFTA$	87.22009.31.41	40	0.76	
CFT$		12	0.23	
CFT2	87.21917.36.06	3	0.06	
4F$		934	17.69	****
2L$		903	17.11	***
INDS$	87.22009.33.35	8	0.15	
INDL$	87.21918.20.27	27	0.51	
WF$	87.21918.14.00	23	0.44	
W2L$		8	0.15	
WT$		6	0.11	
#ABSARY		43	0.06	
UNKNOWN		1	0.02	

ORTIAD
ROGRAM SAMPLING INTERVAL WAS 4 MS; TOTAL NUMBER OF SAMPLES WAS 5279.
IRECT SAMPLES:

TATEMENT	SAMPLES	%UNIT	%TOTAL
UTPUT.ENTRY/EXIT	0	0.00	0.00
UTPUT.6	0	0.00	0.00
UTPUT.7	0	0.00	0.00
UTPUT.8	0	0.00	0.00
UTPUT.9	0	0.00	0.00
UTPUT.10	0	0.00	0.00
UTPUT.11	0	0.00	0.00
UTPUT.12	0	0.00	0.00
UTPUT.13	0	0.00	0.00
UTPUT.14	0	0.00	0.00
UTPUT.15	2	100.00	0.04
UTPUT.16	0	0.00	0.00
UTPUT.17/10	0	0.00	0.00
UTPUT.18	0	0.00	0.00
UTPUT.19	0	0.00	0.00

ORTIAD
ROGRAM SAMPLING INTERVAL WAS 4 MS; TOTAL NUMBER OF SAMPLES WAS 5279.
IRECT SAMPLES:

TATEMENT	SAMPLES	%UNIT	%TOTAL
NCIDENCE.ENTRY/EXIT	0	0.00	0.00
NCIDENCE.6	0	0.00	0.00
NCIDENCE.7	0	0.00	0.00
NCIDENCE.8	0	0.00	0.00
NCIDENCE.9	0	0.00	0.00
NCIDENCE.10	0	0.00	0.00
NCIDENCE.11	0	0.00	0.00
	0	0.00	0.00

```
NCIDENCE.32                    0   0.00   0.00
NCIDENCE.33                    0   0.00   0.00
ORTIAD
NVALID OPERAND SYNTAX SPECIFIED IN "LISTSAMP DIFREQ"
ORTIAD
ROGRAM SAMPLING INTERVAL WAS 4 MS; TOTAL NUMBER OF SAMPLES WAS 5279.
IRECT SAMPLES:
TATEMENT                SAMPLES  %UNIT %TOTAL
IFREQ.ENTRY/EXIT              0   0.00   0.00
IFREQ.7                       0   0.00   0.00
IFREQ.8                       0   0.00   0.00
IFREQ.9                       0   0.00   0.00
IFREQ.10                      0   0.00   0.00
IFREQ.11                      0   0.00   0.00
IFREQ.12                      0   0.00   0.00
IFREQ.13                      0   0.00   0.00
IFREQ.14                      0   0.00   0.00
IFREQ.15                      0   0.00   0.00
IFREQ.16                      0   0.00   0.00
IFREQ.17                      0   0.00   0.00
IFREQ.18                      0   0.00   0.00
IFREQ.19                      0   0.00   0.00
IFREQ.20                      1  11.11   0.02
IFREQ.21                      0   0.00   0.00
IFREQ.22                      0   0.00   0.00
IFREQ.23                      0   0.00   0.00
IFREQ.24                      0   0.00   0.00
IFREQ.25                      0   0.00   0.00
IFREQ.26                      0   0.00   0.00
IFREQ.27                      0   0.00   0.00
IFREQ.28                      0   0.00   0.00
IFREQ.29                      1  11.11   0.02
IFREQ.30                      0   0.00   0.00
IFREQ.31                      0   0.00   0.00
IFREQ.32                      1  11.11   0.02
IFREQ.33                      0   0.00   0.00
IFREQ.34                      1  11.11   0.02
IFREQ.35                      0   0.00   0.00
IFREQ.36                      0   0.00   0.00
IFREQ.37                      0   0.00   0.00
IFREQ.38                      0   0.00   0.00
IFREQ.39                      4  44.44   0.08
IFREQ.40                      1  11.11   0.02
IFREQ.41                      0   0.00   0.00
IFREQ.42                      0   0.00   0.00
IFREQ.43                      0   0.00   0.00
IFREQ.44/20                   0   0.00   0.00
IFREQ.45/10                   0   0.00   0.00
IFREQ.46                      0   0.00   0.00
IFREQ.47                      0   0.00   0.00
ORTIAD
ROGRAM SAMPLING INTERVAL WAS 4 MS; TOTAL NUMBER OF SAMPLES WAS 5279.
IRECT SAMPLES:
TATEMENT                SAMPLES  %UNIT %TOTAL
PERAT.ENTRY/EXIT              0   0.00   0.00
PERAT.9                       0   0.00   0.00
PERAT.10                      0   0.00   0.00
PERAT.11                      0   0.00   0.00
```

PERAT.35	0	0.00	0.00
PERAT.36	30	4.39	0.57
PERAT.37	0	0.00	0.00
PERAT.38	0	0.00	0.00
PERAT.39	0	0.00	0.00
PERAT.40	29	4.25	0.55
PERAT.41	3	0.44	0.06
PERAT.42	0	0.00	0.00
PERAT.43	25	3.66	0.47
PERAT.44	2	0.29	0.04
PERAT.45	0	0.00	0.00
PERAT.46	0	0.00	0.00
PERAT.47	0	0.00	0.00

ORTIAD
ROGRAM SAMPLING INTERVAL WAS 4 MS; TOTAL NUMBER OF SAMPLES WAS 5279.
IRECT SAMPLES:

TATEMENT	SAMPLES	%UNIT	%TOTAL
PECTRA.ENTRY/EXIT	0	0.00	0.00
PECTRA.5	0	0.00	0.00
PECTRA.6	0	0.00	0.00
PECTRA.7	0	0.00	0.00
PECTRA.8	53	11.73	1.00
PECTRA.9	30	6.64	0.57
PECTRA.10	0	0.00	0.00
PECTRA.11	0	0.00	0.00
PECTRA.12	0	0.00	0.00
PECTRA.13	0	0.00	0.00
PECTRA.14	2	0.44	0.04
PECTRA.15	32	7.08	0.61
PECTRA.16	8	1.77	0.15
PECTRA.17/20	2	0.44	0.04
PECTRA.18	0	0.00	0.00
PECTRA.19	0	0.00	0.00
PECTRA.20	0	0.00	0.00
PECTRA.21	0	0.00	0.00
PECTRA.22	117	25.88	2.22
PECTRA.23	27	5.97	0.51
PECTRA.24/10	0	0.00	0.00
PECTRA.25	0	0.00	0.00
PECTRA.26	0	0.00	0.00
PECTRA.27	0	0.00	0.00
PECTRA.28	0	0.00	0.00
PECTRA.29	108	23.89	2.05
PECTRA.30	18	3.98	0.34
PECTRA.31	0	0.00	0.00
PECTRA.32	0	0.00	0.00
PECTRA.33	0	0.00	0.00
PECTRA.34	0	0.00	0.00
PECTRA.35	0	0.00	0.00
PECTRA.36	0	0.00	0.00
PECTRA.37	2	0.44	0.04
PECTRA.38	37	8.19	0.70
PECTRA.39	13	2.88	0.25
PECTRA.40/70	3	0.66	0.06
PECTRA.41/50	0	0.00	0.00
PECTRA.42	0	0.00	0.00
PECTRA.43	0	0.00	0.00
PECTRA.44	0	0.00	0.00

ORTIAD
ROGRAM SAMPLING INTERVAL WAS 4 MS; TOTAL NUMBER OF SAMPLES WAS 5279.
IRECT SAMPLES:

```
C     ================================================================
      PARAMETER    (IDIM=2,DLX=2.,DLY=2.,FMHZ=300.,UPL=31.5,IPAD=1,
     +               ITMAX=200,ITMIN=5,ICMAX=100,TOL=.04,
     +              MX=63,MY=63,NX=128,NY=128,
     +              MT=63*63,NT=128*128,
     +              MTI=63*63*2,NTI=128*128*2,NTFG=128*128*(2+2))
      INTEGER      NN(2),IRING(MT)
      REAL         DFG(NTFG)
      COMPLEX      JS(MTI),E(MTI),A(MTI),R(MTI),P(MTI),
     +             ZT(NTI),AT(NTI),W(NT),RS(3),XJ,ER,JSG
      COMMON       /DIM/ID,MXA,MYA,MTA,MTIA,NXA,NYA,NTA,NTIA,
     +                   NTFA,DLTA,XK1A,Z0A
      COMMON       IBASIS
```

```
ISN  FLAG NESTING *....*...1.........2.........3.........4.........5.........6.........7.*.......8 MESSAGES

0008                      DATA         PI/3.141592653589793/,XJ/(0.,1.)/
0009                      Z0=120.*PI
0010                      XL1=300./FMHZ
0011                      DLT=XL1/UPL
0012                      XK1=2.*PI/XL1
0013                      ID=IDIM
0014                      MXA=MX
0015                      MYA=MY
0016                      MTA=MT
0017                      MTIA=MTI
0018                      NXA=NX
0019                      NYA=NY
0020                      NTA=NT
0021                      NTIA=NTI
0022                      NTFA=NTFG
0023                      DLTA=DLT
0024                      XK1A=XK1
0025                      Z0A=Z0
                    C   READ IN THE INPUT INFORMATION
0026                      OPEN(UNIT=10)
0027                      READ(10,*)ISORCE
0028                      READ(10,*)TETA,PHI,PSI,TMAX,INCT
0029                      READ(10,*)ANTL,ANTPX,ANTPY,ANTPZ
0030                      READ(10,*)E0,CI0
0031                      READ(10,*)ER,TK
0032                      READ(10,*)IBASIS
0033                      CLOSE(10)
                    C      _________________
                    C     |                 |
                    C     | INITIALIZATION  |
                    C     |_________________|
0034                      RS(1) = -XJ*Z0/((ER-1)*XK1*TK)
0035                      RS(2) = RS(1)
0036                      RS(3) = -XJ*Z0*ER/((ER-1)*XK1*TK)
0037                      CALL DIFREQ(DFG)
0038                      CALL GRING(IRING,IRMAX)
0039 VECT +-------        DO 9 I = 1,MTI
0040      |_______           JS(I) = (0.,0.)
0042                      NP = (TMAX-TETA)/INCT+1
0043                      IFILOFF=12
0044                      OPEN(UNIT=11)
0045                10    CONTINUE
                    C   CONVERGENCE PARAMETERS
0046                      ITER = 0
0047                      ICONT = 0
0048                      RSS = 0.
0049                      WRITE(11,200)TETA,PHI,PSI
0050                      CALL INCIDENCE(E,E0,TETA,PHI,PSI)
0051                      ENORM = VNORM(E)
                    C     ZEROTH ORDER RESIDUAL CALCULATION
0052                      CALL OPERAT(A,-1,JS,RS,ZT,DFG,IRING,IRMAX,AT,W)
0053 VECT +-------        DO 20 I = 1,MTI
0054      |                  R(I) = A(I)-E(I)
0055      |_______           P(I) = (0.,0.)
```

REPORT(XLIST) VECTORIZATION ANALYSIS

```
ISN  FLAG NESTING *....*...1.........2.........3.........4.........5.........6.........7.*.......8 MESSAGES

                  C    _____________________
                  C   |                     |
                  C   | MAIN ITERATION LOOP |
                  C   |_____________________|
0057              30      CONTINUE
0058                      CALL OPERAT(A,1,R,RS,ZT,DFG,IRING,IRMAX,AT,W)
0059                      B = 1./VNORM(A)
0060 VECT +-------        DO 35 I = 1,MTI
0061      |_______           P(I) = P(I)-B*A(I)
0063                      CALL OPERAT(A,-1,P,RS,ZT,DFG,IRING,IRMAX,AT,W)
0064                      T = 1./VNORM(A)
0065 VECT +-------        DO 40 I = 1,MTI
0066      |                  R(I) = R(I)+T*A(I)
0067      |_______           JS(I) = JS(I)+T*P(I)
0069                      ITER = ITER+1
0070                      ICONT = ICONT+1
                  C    _____________________
                  C   |                     |
                  C   | CONVERGENCE CHECK   |
                  C   |_____________________|
                  C   COMPUTE RELATIVE RESIDUAL ERROR
0071                      RSS = SQRT(VNORM(R)/ENORM)
0072                      WRITE(*,300)ITER,RSS
0073                      WRITE(11,300)ITER,RSS
0074                      IF((RSS.LE.TOL).AND.(ITER.GT.ITMIN)) THEN
0075                        WRITE(11,400)
0076                      ELSE IF(ITER.GE.ITMAX) THEN
0077                        WRITE(11,500)
0078                      ELSE IF(ICONT.EQ.ICMAX) THEN
0079                        GO TO 90
0080                      ELSE
0081                        GO TO 30
0082                      END IF
0083               90     CONTINUE
                  CCCCC   CALL OUTPUT(12,1,JS,ZT,TETA,SIG,ITER,RSS)
                  C   IF INTERMEDIATE RESULTS, RESET THE COUNTER.
0084                      IF(ICONT.EQ.ICMAX) THEN
0085                        ICONT = 0
0086                        GO TO 30
0087                      END IF
                  C       IF(ISORCE.EQ.1)THEN
                  C   IF DONE WITH THIS ANGLE OF INCIDENCE, STORE THE CONVERGED CURRENT AND
                  C   COMPUTE  EXCITED  CURRENT AND SIGMA0  FOR NEXT ANGLE OF
                  C   INCIDENCE WITH THE LAST JS  AS THE NEW  SOLUTION GUESS.
0088                       CALL OUTPUT(IFILOFF,1,JS,ZT,TETA,SIG,ITER,RSS)
0089                       IFILOFF=IFILOFF+2
0090                       TETA = TETA+INCT
0091                       IF(TETA.LE.TMAX) THEN
0092                         GO TO 10
0093                       ENDIF
                  C       ENDIF
0094                      WRITE(11,*)'PROGRAM TERMINATED.'
```

REPORT(XLIST) VECTORIZATION ANALYSIS

```
ISN  FLAG NESTING *....*...1.........2.........3.........4.........5.........6.........7.*.......8 MESSAGES

                  C        DO I = 1,11
                  C         CLOSE(I)
                  C       END DO
0097               200    FORMAT(29('='),/,'TETA=',F4.1,
0097                   +  2X,'PHI=',F4.1,2X,'PSI=',F4.1,/, 'ITER',7X,'RSS(R)',/,29('-'))
0098               300    FORMAT(I4,1X,G13.4)
0099               400    FORMAT('CONVERGENCE  ACHIEVED.')
0100               500    FORMAT('ITMAX EXCEEDED; CONVERGENCE CRITERION NOT MET.')
0101               600    FORMAT(1X,'TETA',3X,'SIGO dB',
0101                   +  2X,'SIGO(PO) dB',2X,'PHI=',F5.1,2X,'PSI=',F5.1,/,I2)
0102                      STOP
0103 ____ ________        END                                                          ________

0001                      SUBROUTINE OUTPUT(IUNT,IFLG,Z,ZT,TETA,SIG,ITER,RSS)
0002                      COMPLEX Z(*),ZT(*)
0003                      CHARACTER*11 FMTI
0004                      CHARACTER*16 FMTR
0005                      COMMON       /DIM/ID,MX,MY,MT,MTI,NX,NY,NT,NTI,NTF,DLT,XK1,ZO
0006 UNAN                 DO 10 I = 1,ID                                           I/O OPERATION
                                                                                   UNANALYZABLE INTRINSI
0007                         IUNTI = I+IUNT-1
0008                         OPEN(UNIT=IUNTI)
0009                         WRITE(IUNTI,*)TETA,S,ITER,RSS
0010                         WRITE(IUNTI,*)MX,MY
0011                         WRITE(IUNTI,*)(J,J=1,MX)
0012                         WRITE(IUNTI,*)(K,K=1,MY)
0013                         L1=(I-1)*MT+1
0014                         L2=I*MT
0015                         WRITE(IUNTI,*)(CABS(Z(L)),L=L1,L2)
0016                         CLOSE(IUNTI)
0017               10        CONTINUE
0018                         RETURN          ____________ _____________ ________

ISN  FLAG NESTING *....*...1.........2.........3.........4.........5.........6.........7.*.......8 MESSAGES

0001                      SUBROUTINE INCIDENCE(E,EO,TETA,PHI,PSI)
0002                      COMMON       /DIM/ID,MX,MY,MT,MTI,NX,NY,NT,NTI,NTF,DLT,XK1,ZO
0003                      COMPLEX      E(*),XJ,G,RJK(7938)
0004                      REAL         XP(63),YP(63)
0005                      DATA         PI/3.141592653589793/,XJ/(0.,1.)/
0006                      MXO = MX/2+1
0007                      MYO = MY/2+1
                  C     ____________________
                  C    |                    |
                  C    | INCIDENT PLANE WAVE |
                  C    |____________________|
0008                         TET = TETA*PI/180.
0009                         PH  = PHI*PI/180.
0010                         PS  = PSI*PI/180.
0011                         ECS=EO*(COS(PS)*COS(TET)*COS(PH)-SIN(PS)*SIN(PH))
0012                         ESC=EO*(COS(PS)*COS(TET)*SIN(PH)+SIN(PS)*COS(PH))
0013                         XI = XK1*SIN(TET)*COS(PH)
0014                         YI = XK1*SIN(TET)*SIN(PH)
0015 VECT +-------           DO 10 K=1,MY
                                YP(K)=(K-MYO)*DLT
```

```
0003                    COMMON      /DIM/ID,MX,MY,MT,MTI,NX,NY,NT,NTI,NTF,DLT,XK1,ZO
0004                    COMPLEX     W(*)
0005                    SUM = 0.DO
0006 VECT +-------      DO 10 I = 1,MTI
0007       |                AMAG=DBLE(CABS(W(I)))                    VECTOR INTRINSIC FUNC
0008       |_______         SUM = SUM+AMAG*AMAG                      VECTOR SUM REDUCTION
0010                    VNORM = SNGL(SUM)
0011 ____ _______       RETURN                                       ________

0001                    SUBROUTINE  GRING(IRING,IRMAX)
0002                    INTEGER     IRING(*)
0003                    REAL*8      X,KX,KY,FRX,FRY,PI,TPI,XK2
0004                    COMMON      /DIM/ID,MX,MY,MT,MTI,NX,NY,NT,NTI,NTF,DLT,XK1,ZO
0005                    DATA        PI/3.141592653589793/
0006                    TPI=2.DO*PI
0007                    XK2=DBLE(XK1*XK1)
0008                    NXH=NX/2+1
0009                    NYH=NY/2+1
0010                    FRX=1.DO/DBLE(DLT*NX)
0011                    FRY=1.DO/DBLE(DLT*NY)
0012                    INDX=0
0013 UNSP +-------      DO 10 K = 1,NY
0014       |                IF(K.LE.NYH)KY = TPI*(K-1)*FRY           DEPENDENT ON UNSUPPOR
           |                                                         CONDITIONAL SCALAR CO
0016       |                IF(K.GT.NYH)KY = TPI*(K-NY-1)*FRY        DEPENDENT ON UNSUPPOR
           |                                                         CONDITIONAL SCALAR CO
0019 RECR  |+------           DO 20 J = 1,NX
0020       ||                    IF(J.LE.NXH)KX = TPI*(J-1)*FRX      SCALAR DEFINED BEFORE
           ||                                                        CONDITIONAL SCALAR CO
0022       ||                    IF(J.GT.NXH)KX = TPI*(J-NX-1)*FRX   SCALAR DEFINED BEFORE
           ||                                                        CONDITIONAL SCALAR CO
0025       ||                    X = KX**2+KY**2-XK2                 DEPENDENT ON UNSUPPOR
0026       ||                    IF(X.LT.0.DO)THEN                   DEPENDENT ON UNSUPPOR
0027       ||                      INDX=INDX+1                       SCALAR DEFINED BEFORE
           ||                                                        CONDITIONAL SCALAR CO
0028       ||_____               IRING(INDX)=L1                      SUBSCRIPT TOO COMPLEX
           |_______                                                  OPTIMIZER INDUCED DEP
                                                                     CONDITIONAL NON-INDUC

0032                    IRMAX=INDX
0033                    RETURN
0034                    END
```

```
NUMBER     ISN         FLAG  VS FORTRAN VECTOR REPORT MESSAGES

ILX01351   0014-0016   UNSP  THIS CODE IS CONSIDERED UNSUPPORTABLE BECAUSE IT IS LINKED TO SOME UNSUPPORTABLE STATEMENT(S)
                             MUTUAL DEPENDENCES.

ILX01581   0014-0016   SCAL  THIS CODE IS CONDITIONALLY EXECUTED. THE VECTOR REPORT LISTING MAY FAIL TO INDICATE THE BRANC
                             STATEMENT(S) THAT AFFECT THE EXECUTION OF THIS REGION.

                                   REPORT(XLIST) VECTORIZATION ANALYSIS

ISN  FLAG NESTING *....*...1.........2.........3.........4.........5.........6.........7.*.......8 MESSAGES

0001                      SUBROUTINE  DIFREQ(DFG)
0002                      REAL*8      X,KX,KY,FRX,FRY,C(3,3),P,Q,PXY,G,PI,TPI,XK2
0003                      REAL        DFG(*)
0004                      COMMON      /DIM/ID,MX,MY,MT,MTI,NX,NY,NT,NTI,NTF,DLT,XK1,Z0
0005                      COMMON      IBASIS
0006                      DATA        PI/3.141592653589793/
0007                      TPI=2.D0*PI
0008                      XK2=DBLE(XK1*XK1)
0009                      FRX=1.D0/DBLE(DLT*NX)
0010                      FRY=1.D0/DBLE(DLT*NY)
0011                      NXH = NX/2+1
0012                      NYH = NY/2+1
                  C
                  C    ____________________________________
                  C   |                                    |
                  C   | COMPUTE SPECTRAL VARIABLES AND     |
                  C   | DIFERENTIAL OPERATOR  ELEMENTS     |
                  C   |____________________________________|
0013 RECR +-------        DO 10 K=1,NY
0014      |                  IF(K.LE.NYH)KY=TPI*(K-1)*FRY                    SCALAR DEFINED BEFORE
          |                                                                  CONDITIONAL SCALAR CO
0016      |                  IF(K.GT.NYH)KY=TPI*(K-NY-1)*FRY                 SCALAR DEFINED BEFORE
          |                                                                  CONDITIONAL SCALAR CO
0018      |                  C(2,2)=XK2-KY*KY
0019 RECR |+------           DO 20 J=1,NX
0020      ||                    IF(J.LE.NXH)KX=TPI*(J-1)*FRX                  SCALAR DEFINED BEFORE
          ||                                                                  CONDITIONAL SCALAR CO
0022      ||                    IF(J.GT.NXH)KX=TPI*(J-NX-1)*FRX               SCALAR DEFINED BEFORE
          ||                                                                  CONDITIONAL SCALAR CO
0029      ||                    C(1,1)=XK2-KX*KX
0030      ||                    C(1,2)=-KX*KY
0031      ||                    C(2,1)=C(1,2)
0032      ||                    IF(ID.EQ.3)C(3,3)=KX*KX+KY*KY                 CONDITIONAL SCALAR CO
0034      ||                    X = DABS(KX*KX+KY*KY-XK2)
0035      ||                    IF(X.LT.1.D-20)X=1.D-10                       CONDITIONAL SCALAR CO
0037      ||                    G = 1.D0/(2.*DSQRT(X))
0038      ||                    DFG(L1)=SNGL(C(1,1)*G)                        STRIDE UNKNOWN
          ||                                                                  INTERCHANGE PREVENTIN
0039      ||                    DFG(L2)=SNGL(C(1,2)*G)                        STRIDE UNKNOWN
          ||                                                                  INTERCHANGE PREVENTIN
0040      ||                    DFG(L3)=SNGL(C(2,1)*G)                        STRIDE UNKNOWN
          ||                                                                  INTERCHANGE PREVENTIN
0041      ||                    DFG(L4)=SNGL(C(2,2)*G)                        STRIDE UNKNOWN
          ||                                                                  INTERCHANGE PREVENTIN
0042      ||_____               IF(ID.EQ.3)DFG(L5)=SNGL(C(3,3)*G)             STRIDE UNKNOWN
          |_____                                                            INTERCHANGE PREVENTIN
                                                                            CONDITIONAL SCALAR CO
```

```
                C   |____________________|
                C
0016                    CALL SPECTRA (Z,ZT,-1,W)
                C                     I  O
                C   ____________________________________
                C  |                                    |
                C  | CONVOLUTION AND DIFFERENTIATION    |
                C  | IN TRANSFORM DOMAIN                |
                C  |____________________________________|
0017 VECT +-------      DO 10 I=1,NT
0018      |_______          AT(I)=DFG(I)*ZT(I)+DFG(I+NT)*ZT(I+NT)

0020 VECT +-------      DO 20 I=NT+1,NTI
0021      |_______          AT(I)=DFG(I+NT)*ZT(I-NT)+DFG(I+NTI)*ZT(I)
                C   SELECT THE APPROPRIATE BRANCH ON THE GREENS FUNCTION
0023 RECR +-------      DO 40 L = 1,IRMAX                          NESTED NON-CONSTANT I
0024      |                 L1 = IRING(L)
0026      |                   L2=(I-1)*NT+L1
0025 RECR |+------          DO 30 I = 1,ID
0027      ||______            AT(L2) = ISIGN*XJ*AT(L2)             STRIDE UNKNOWN
          |_______                                               SUBSCRIPT TOO COMPLEX
                                                                 INTERCHANGE PREVENTIN
                C   ____________________________
                C  |                            |
                C  | INVERSE FOURIER TRANSFORM  |
                C  |____________________________|
                C
0030                    CALL SPECTRA (A,AT,1,W)
                C                     O  I
                C   ____________________________________
                C  |                                    |
                C  | INTEGRO-DIFFERENTIAL OPERATOR MATRIX |
                C  |____________________________________|

0031                    IF(ISIGN.EQ.1)THEN
0032 VECT +-------         DO 50 I=1,MT
```

REPORT(XLIST) VECTORIZATION ANALYSIS

```
ISN  FLAG NESTING *....*...1.........2.........3.........4.........5.........6.........7.*.......8 MESSAGES

0033       |_______              A(I)=CRS(1)*Z(I)-FCT*A(I)

0035 VECT  +-------              DO 60 I=MT+1,MTI
0036       |_______                 A(I)=CRS(2)*Z(I)-FCT*A(I)
0038                           ELSE
0039 VECT  +-------              DO 70 I=1,MT
0040       |_______                 A(I)=RS(1)*Z(I)+FCT*A(I)

0042 VECT  +-------              DO 80 I=MT+1,MTI
0043       |_______                 A(I)=RS(2)*Z(I)+FCT*A(I)
0045                           ENDIF
0046                           RETURN
0047                           END
```

```
NUMBER     ISN        FLAG  VS FORTRAN VECTOR REPORT MESSAGES

ILX01291   0023       RECR  THE LOOP VARIABLE OF THIS LOOP OR OF SOME NESTED LOOP AFFECTS THE LOOP VARIABLE OR AN AUXILIA
                             INDUCTION VARIABLE USED BY SOME OTHER NESTED LOOP.

0001                       SUBROUTINE SPECTRA(Z,ZT,ISIGN,W)
0002                       COMMON      /DIM/ID,MX,MY,MT,MTI,NX,NY,NT,NTI,NTF,DLT,XK1,Z0
0003                       REAL*8      AUX1(20000),AUX2(20000)
0004                       COMPLEX     Z(*),ZT(*),W(*)
0005                       IF(ISIGN.EQ.-1)THEN
0006 UNAN                    DO 10 I = 1,ID                                          USER FUNCTION OR SUBR
                       C   PRECONDITIONING IN SPATIAL DOMAIN
0007 VECT +--------            DO 15 II=1,NT
0008      |________               W(II)=(0.,0.)
0010                           L1 = (I-1)*MT
0011 RECR +--------            DO 20 K = 1,MY
0014 VECT |+------               DO 30 J = 1,MX
0015      ||______                  W(L3+J) = Z(L2+J)                              STRIDE UNKNOWN
          |_______
                       CCCCCCC     CALL ZPADN(W)
0018                         CALL SCFT2(1,W,1,NX,W,1,NX,NX,NY,1,1.0,AUX1,20000,AUX2,20000)
0019                         CALL SCFT2(0,W,1,NX,W,1,NX,NX,NY,1,1.0,AUX1,20000,AUX2,20000)
0020                           L1 = (I-1)*NT
0021 VECT +--------            DO 40 J = 1,NT
0022      |________               ZT(L1+J) = W(J)
0024                 10      CONTINUE
0025                       ELSE IF(ISIGN.EQ.1)THEN
0026 UNAN                    DO 50 I = 1,ID                                          USER FUNCTION OR SUBR
0027                           L1 = (I-1)*NT
0028 VECT +--------            DO 60 J = 1,NT
0029      |________               W(J) = ZT(L1+J)
0031                         CALL SCFT2(1,W,1,NX,W,1,NX,NX,NY,-1,1.0,AUX1,20000,AUX2,20000)
0032                         CALL SCFT2(0,W,1,NX,W,1,NX,NX,NY,-1,1.0,AUX1,20000,AUX2,20000)
0033                           L1 = (I-1)*MT
0034 RECR +--------            DO 70 K = 1,MY
0037 VECT |+------               DO 80 J = 1,MX
0038      ||______                  Z(L3+J) = W(L2+J)                              STRIDE UNKNOWN
          |_______
0041                 50      CONTINUE
0042                       END IF
0043                       RETURN
0044                       END
```

```
NUMBER    ISN          FLAG  VS FORTRAN VECTOR REPORT MESSAGES
AFF5501 PROGRAM SAMPLING INTERVAL WAS 4 MS; TOTAL NUMBER OF SAMPLES WAS 5279.
AFF5521 SUM OF DIRECT AND CALLED SAMPLES:
AFF5641 PROGRAM UNIT                                  PAGE  %TOTAL  DISTRIBUTION
AFF5651   SLAB                                           1  100.00  ****************+
AFF5651   OUTPUT                                        10    0.53
AFF5651   INCIDENCE                                     14    0.08
AFF5651   VNORM                                         19    6.61  ***
AFF5651   GRING                                         23    0.04
AFF5651   DIFREQ                                        27    0.27
AFF5651   OPERAT                                        32   90.09  ****************+
AFF5651   SPECTRA                                       38   77.15  ****************+
AFF5651   D#SQRT                                             0.09
AFF5651   VFSCOCS#                                           0.04
AFF5651   VC#EXP                                             0.04
AFF5651   VC#ABS                                            5.61  ***
AFF5651   F4I$                                            31.39  ***************
AFF5651   SCFTA$    87.22009.31.41                       68.31  ****************+
AFF5651   SCFT$                                          68.54  ****************+
AFF5651   SCFT2     87.21917.36.06                       68.59  ****************+
AFF5651   F4F$                                           17.69  *********
AFF5651   F2L$                                           17.11  *********
AFF5651   SINDS$   87.22009.33.35                        0.15
AFF5651   SINDL$   87.21918.20.27                        0.51
AFF5651   FWF$     87.21918.14.00                       0.59
AFF5651   FW2L$                                          0.15
AFF5651   FWT$                                           0.11
AFF5651   C#ABS                                          0.06
AFF5651   *LIBRARY                                       0.85
AFF5651   *UNKNOWN                                       0.02
```

References

[1] M. R. Hestenes, *Conjugate Direction Methods In Optimization*, Springer-Verlay, New York, 1980.

[2] T. K. Sarkar, A. Ercument, and M. Rao, Application of FFT and the conjugate gradient method for the solution of electromagnetic radiation from electrically large and small conducting bodies, *IEEE Trans. Antenna Propagat.*, AP-34(5):635–640, May 1985.

[3] T. J. Peters and J. L. Volakis, The application of a conjugate gradient FFT method to scattering from thin planar material plates, *IEEE Trans. Antenna Propagat.*, AP-36(4):518–526, April 1988.

[4] K. Barkeshli and J. L. Volakis, Improving the convergence rate of the conjugate gradient FFT method using subdomain basis functions, *IEEE Trans. Antenna Propagat.*, AP-37(4), April 1989.

[5] T. K. Sarkar, The application of the conjugate gradient method for the solution of operator equations arising in electromagnetic scattering from wire antennas, *Radio Science*, 19(5):1156–1172, September-April 1984.

[6] K. Barkeshli and J. L. Volakis, Application of the conjugate gradient FFT method to a class of large radiating and scattering problems, Technical Report 389604-3-T, Radiation Laboratory, Department of Electrical Engineering and Computer Science, The University of Michigan, April 1988.

Index

Index

THE UNIVERSITY OF MICHIGAN

DATE DUE

4/27 15:39