View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Woods Hole Open Access Server

WHOI-91-31

c. 1

Woods Hole
Oceanographic
Institution

Benchmarking the Two-dimensional Finite Difference Synthetic Seismogram Code

by

J.M. Allen and R.A. Stephen

September 1991

Technical Report

Funding was provided by the Office of Naval Research
under Contract No. N00014-89-J-1012.

Approved for public release; distribution unlimited.

DOCUMENT
LIBRARY
Woods Hole Oceanographic
Institution

WHOI-91-31

Benchmarking the Two-dimensional Finite Difference Synthetic Seismogram Code

by

J.M. Allen and R.A. Stephen

Woods Hole Oceanographic Institution
Woods Hole, Massachusetts 02543

September 1991

Technical Report

Funding was provided by the Office of Naval Research
under Contract No. N00014-89-J-1012.

Reproduction in whole or in part is permitted for any purpose of the United States Government. This report should be cited as Woods Hole Oceanog. Inst. Tech. Rept., WHOI-91-31.

MBL/WHOI
0 0301 0096280 9

Approved for public release; distribution unlimited.

Approved for Distribution:

[illegible]

G. Michael Purdy, Chairman
Department of Geology and Geophysics

Contents

List of Tables

List of Figures

Abstract

During the past six months, the two-dimensional finite difference synthetic seismogram code was installed and run on a number of different computer systems. The results were compared for timing, accuracy and the ease with which the code was adapted to each system.

This report documents the software modifications and the methods used to implement the finite difference code on each computer, and presents the results of the benchmark survey.

1 Introduction

The finite difference method is used to solve the elastic wave equation in order to generate synthetic seismograms in two-dimensions. The program that was used to benchmark the finite difference code is described by Hunt, et al. (1983). The only modifications made to the code were those necessary to make the code compile and run on each specific machine. Modifications are discussed in more detail in section 3.

The software package consists of three programs, a model parameter input file and a batch command file (VMS ©) or shell file (UNIX ©) to facilitate the execution of the program.

The model to be calculated is divided into three horizontal layers. The top layer is assigned the physical properties of water and the bottom layer is the sub-surface stratum. Both of these layers are assumed to be homogeneous, with a constant velocity structure. The layer between them is the sea-floor boundary layer and contains the part of the model that varies. The velocities within this layer are calculated by the SCNTBNY program.

The first program, SCNTPREP, reads the model parameter file and initializes the system for the current model. The array bounds are calculated and written to a common file, SCOMFD8.FOR, which is then included in the following two programs, SCNTBNY and SCNTDIF. These two programs are re-compiled for each new model so that the program size exactly matches the model. The benchmark model that was calculated for this report requires approximately 15.2 Mb of memory.

The next step is to compile and run the program SCNTBNY, which sets up the transition zone velocity structure.

The final program, SCNTDIF, does the actual calculation of the elastic wave equation in two dimensions by finite differences.

The output files include 'snapshot' files generated at selected time steps, log files with accuracy information and timing statistics and a time series file (if selected).

2 Overview of benchmark results

Table 1 shows the timing results of the benchmark runs; Appendix B contains a partial comparison of the accuracy of the different machines.

The main difference in the code that was run on the various machines was in the subroutine TIMIT.FOR, which calculates the timing statistics. The timing utilities on the various machines were not always comparable. Whenever possible, CPU (system) time was measured but elapsed time was also recorded. The benchmark runs were made in a "stand-alone" mode when possible but on some of the larger time-sharing machines, other processes may have been running concurrently with the benchmark process. In these instances, comparison of the elapsed and CPU times usually indicated that there was little or no impact from the competing processes.

On computers with optimizing compilers, separate runs were made for each optimization level. In the design of the programs, no effort was made to optimize the code for any particular machine or machine architecture. The code was written to simply represent the physical phenomena in a logical fashion.

The check for accuracy was made by comparing the values of the array **A1DATA** at selected grid points at the end of each computer run. **A1DATA** is the array that contains the time series at each receiver location.

The following sections describe how the benchmark program was run on each of the hardware platforms.

3 Discussion of benchmark runs

The source code was installed on each computer in four separate sub-directories:

prep
bny
diff
cnt01

The **prep** sub-directory contains all of the source code for the **SCNTPREP** program. On UNIX machines, there is a **makefile**; on computers running the VMS operating system, there is a file called **SCNTPREP.LNK** to perform the compilation and link. Similarly, the **bny** and **diff** sub-directories contain the source code and makefiles or link files for the programs **SCNTBNY** and **SCNTDIF**, respectively.

The sub-directory **cnt01** contains the input model parameter file, **CNT01.PAR** and a batch or shell file used to run the series of programs (**CNT01.BCH**). The parameter file and some examples of the batch and shell files used for this benchmark are shown in Appendix A. On some of the UNIX machines, an additional shell file (**cnt01.sh**) was used to generate the timing statistics via the UNIX **/bin/time** utility.

The FORTRAN logical unit numbers were assigned in the VMS batch command files via the **ASSIGN** statement; on the Cray there is a COS (Cray Operating System) equivalent of the **ASSIGN**. On the Convex, there is a **setenv** statement which performs the logical assignment but on the other UNIX machines it was necessary to copy or rename the required files to the filename **fort.nn** where **nn** is the FORTRAN logical unit number to be assigned.

3.1 VAX 4000, 6420, 8600, 8800, 9000

All of the VAX/VMS computers (4000, 6420, 8600, 8800 and 9000) ran identical code. The timing functions called in the TIMIT.FOR subroutine were the VAX/VMS library routines LIB$INIT_TIMER and LIB$SHOW_TIMER.

As indicated in Appendix B, all of the VAX/VMS machines produced identical results.

The VAX 4000 is the new MicroVAX (uses the same CMOS ChipSet as the VAX6410). The computer used in the benchmark had 128MB of memory. The VAX 4000 has a single processor and no vector processor.

Although the VAX 6420 can do multiprocessing and can be configured with up to six processors, the benchmark code was not parallelized. The VAX 6420 also has a vector processor; benchmark runs were made both with and without vectorization.

The benchmark times for the VAX 4000 and the VAX 6420 run in scalar mode (not vectorized)

		Time			
Computer	Date	CPU (real)	Elapsed	Memory	Comments
VAX 9000	21/12/90	0:09:08.74	0:10:46.26	512 mB	scalar
	21/12/90	0:07:47.22	0:09:25.23	512 mB	vectorized
VAX 8800	12/6/90	1:27:33.94	1:42:49.93	48 mB	optimized (WHOI)
	30/5/90	2:43:52.48	2:57:50.51	48 mB	not optimized (WHOI)
	11/5/90	2:45:34.90	3:06:30.46	48 mB	not optimized (WHOI)
	6/6/90	1:24:22.90	1:25:57.51	128 mB	optimized (DEC)
VAX 8600	11/10/90		1:55:37.0	44 mB	optimized
	8/11/90		1:19:00.0	44 mB	using FPS264, OPT=0
	8/11/90		0:49:00.0	44 mB	using FPS264, OPT=1
	9/11/90		0:34:00.0	44 mB	using FPS264, OPT=3
VAX 6420	23/5/90	1:07:05.87	1:08:28.77	128 mB	scalar
		0:57:00.00	0:59:00.00	128 mB	vectorized
VAX 4000	10/7/90	1:06:34.54	1:08:27.04	128 mB	
DEC 5000	5/2/91	1:37:56.8	1:58:11.0	16 mB	no optimization
	6/2/91	1:15:09.2	1:18:45.1	16 mB	optimization level -O1
	7/2/91	0:36:22.0	2:21:48.0	16 mB	optimization level -O2
	8/2/91	0:36:03.6	1:04:04.3	16 mB	optimization level -O3
IBM RS/6000 model 320	30/5/90	0:55:45.9	0:56:14.6	16 mB	no optimization
	13/6/90	0:55:44.9	0:56:30.4	16 mB	no optimization
	7/2/91	0:55:55.8	0:58:13.0	16 mB	no optimization
	2/1/91	0:13:33.4	0:15:05.2	16 mB	optimized
	7/2/91	0:15:22.6	0:17:20.9	16 mB	optimized and inlined
SUN SPARC model 4/60	2/1/91	2:58:45.2	3:15:22.3	40 mB	no optimization
	2/6/91	1:29:37.3	1:42:47.6	40 mB	optimization level -O3
CONVEX model 220	30/ 5/90		0:58:48.0	128 mB	no optimization
	8/ 6/90		0:55:32.0	128 mB	no optimization
	4/12/90		1:43:18.0	128 mB	no optimization (vers 6)
	13/ 6/90		0:34:44.0	128 mB	scalar level 0
	4/12/90		0:35:01.0	128 mB	scalar level 0 (vers 6)
	13/ 6/90		0:31:19.0	128 mB	scalar level 1
	4/12/90		0:29:40.0	128 mB	scalar level 1 (vers 6)
	23/ 5/90		0:17:31.0	128 mB	vectorized
	25/ 5/90		0:17:19.0	128 mB	vectorized
	4/12/90		0:17:48.0	128 mB	vectorized (vers 6)
CRAY X-MP	10/2/90	0:07:54.59		16 mW	64-bit words

Table 1: Timing results

were nearly the same (Table 1). When the code was re-compiled using the VAX optimizing compiler and run using the vector processor, the speed was not significantly improved.

The benchmark was run on two different VAX 8800's, one at WHOI and one at DEC. The WHOI VAX 8800 (48 mB memory) was not run in standalone mode but judging from the small difference between the elapsed time and the CPU time, there was not a great load on the system. The code was compiled both with and without optimization on the WHOI VAX 8800. Table 1 shows that the optimized code was significantly faster. The optimized code run on the WHOI VAX 8800 (48 mB of memory) was only slightly slower than the optimized code run in standalone mode on the DEC VAX 8800 (128 mb of memory).

Four runs were made on the VAX 8600, using the four different levels of optimization available. Without the FPS (Floating Point Accelerator), the VAX 8600 was comparable with the optimized VAX 8800. Using the FPS considerably improved the speed.

The benchmark was run on a VAX 9000 both in scalar and vector mode. There was no significant difference in the times of the two runs; the VAX 9000, in vector mode, had the fastest runtime of all the hardware platforms tested.

3.2 DECstation 5000

The DECstation 5000 used in this study was running the ULTRIX operating system. The FORTRAN code was identical to that run on the VAX machines running under VMS; the only modification was to the batch command (shell) files. It is interesting to note, however, that the results of the benchmark run on the DECstation 5000 were not identical to the results of the runs on the VAX machines that were running VMS (Appendix B), although the results of the different optimization levels on the DECstation 5000 were the same. The benchmark runs that were performed on UNIX platforms were never identical, but varied after the fourth or fifth decimal place of precision. All of the VAX/VMS runs produced identical results. There is apparently a floating-point precision difference in the various compilers that were used on the UNIX-based machines.

The DECstation 5000 FORTRAN 77 compiler (f77) allows for four levels of optimization:

-O0	used to turn off all optimizations
-O1	turn on all optimizations that complete quickly (default)
-O or -O2	invoke the global *ucode* optimizer
-O3	perform all optimizations, include global register allocation

According to the online manual, when the highest level of optimization is specified (with the **-O3** flag), each FORTRAN 77 source file is compiled into a *ucode* object file with the extent .u. All files are then *ucode* linked. Optimization is done on the resulting *ucode* linked file and then it is linked as normal, producing an executable file.

3.3 IBM RS 6000

The IBM 6000 model 320 that was used in this study was running the UNIX System 5 operating system. The IBM C timing function time() as well as the UNIX /bin/time utility were used to generate the timing statistics. The UNIX timing utility /bin/time was run from a shell, with the output redirected to a file.

Modifications were made to several of the subroutines to remove the non-standard FORTRAN 77 code. In subroutines ZDIVCURL.F and ZSNPOUT.F the qualifier DISP='SAVE' was removed from the OPEN and CLOSE statements. The non-standard qualifier CARRIAGECONTROL='LIST' was removed from the OPEN statement in SOPNCOM.F.

To run the program SCNTPREP, the file CNT01.PAR must be assigned to FORTRAN logical unit number 55. Program SCNTBNY creates a file named CNT01.BNY which needs to be associated with FORTRAN logical unit number 54. The following lines were added to the file CNT01.BCH to accomplish this:

Before running SCNTPREP:	**% cp CNT01.PAR fort.55**
after running SCNTBNY:	**% mv CNT01.BNY fort.54**

The IBM RS 6000 FORTRAN 77 compiler (xlf) has only one level of optimization available. The benchmark code was compiled and run with no optimization and with optimization. Another compiler option (**-Q**), which inlines all appropriate subprograms, was also tested. The optimized code ran significantly faster than the non-optimized code. Inlining the code caused the performance to deteriorate slightly (see Table 1). The results (from the *.LG4 files) were identical for all runs.

3.4 SUN SPARCstation

The SUN SPARCstation used to run the FINDIF code has 40 mB of memory and is running the FORTRAN 77 compiler version 1.2.

On the SUN, the subroutine SOPNCOM.F was edited to remove the non-standard FORTRAN 77 qualifier CARRIAGECONTROL='LIST' in the OPEN statements. Subroutines ZSNPOUT.F, ZDIVCRL.F and ZTSOUT.F were edited to remove the non-standard FORTRAN 77 qualifier DISP='SAVE' in the OPEN statements. Also, the call to DATE was commented out in subroutine DATIM.F. The UNIX FORTRAN timing function **dtime()** was used in subroutine **timit.f**.

The FORTRAN logical unit numbers were assigned in the same manner as on the IBM RS 6000.

On the SUN SPARCstation, runs were made with no optimization and with optimization level -O3 (global optimization). Optimization of the code reduced the runtime by approximately 50%. The results of the two runs were identical.

3.5 Convex C220

The Convex C220 FORTRAN compiler operates at several different optimization levels. The FINDIF code was compiled and run on the Convex at all of the optimization levels except for level 3 which does parallel optimization. The first run was made with the **-no** compiler option selected; only machine-dependent optimizations are performed at this level and they cannot be disabled.

The next level of optimization is selected by including the **-O0** compiler option. At this level, local scalar optimization is performed. This optimization eliminates unnecessary computations on sequences of code with one entrance and exit.

Global scalar optimization is selected with the **-O1** compiler option. At this level, optimization is performed over entire program units, including conditional statements and loops.

Vectorization is selected by compiling the program with the **-O2** option. This option causes the processor to use vector registers to calculate up to 128 elements of an array with a single instruction. Local and global scalar optimization are also included.

The only modification required for the code was to use the Convex UNIX FORTRAN function **dtime()** in the subroutine **timit.f**.

A bug has been reported in the Convex optimizing FORTRAN compiler (Version 6.0) which has the possibility of causing wrong answers with the **-O2** and **-O3** optimization levels. The results of the FINDIF runs on the Convex are presented in Appendix B. Although the values that were calculated at the different optimization levels are not identical, they are still within the range of differences found between the various platforms.

3.6 Cray X-MP

The FINDIF code was run on the Cray X-MP/216 located at the Naval Research Laboratory (NRL) Central Computing Facility (CCF). The Cray is a dual processor supercomputer with 16 million (64-bit) words of main memory. User interaction with the Cray is via a front-end computer. The Cray at NRL runs under COS (Cray Operating SYSTEM) and is accessed via VAX front-ends, running under VMS.

Editing and other file preparation was done on the VAX and submitted to the Cray for execution via the CBATCH utility.

Implementation of the benchmark programs on the Cray required some modification of the FORTRAN code and the development of COS Job Control Files to:

- transfer the source code and data files from the VAX onto the Cray
- compile the subroutines and programs on the Cray
- run the programs on the Cray
- transfer the log and output files from the Cray to the VAX

The following sections describe the process of logging onto the Cray, creating the Cray JOB files, converting the FORTRAN programs to run under COS, running the FINDIF code on the Cray and examining the results.

3.6.1 Logging onto the Cray X-MP

The NRL Cray can be accessed from WHOI via Internet. The procedure for logging onto the Cray from the Red VAX is as follows:

```
$ telnet ccf.nrl.navy.mil
USERNAME: STEPHEN
PASSWORD:
```

3.6.2 Using the Cray Operating System (COS)

The Cray Operating System (COS) allows the user to submit, monitor and terminate batch jobs on the Cray.

Some of the more useful COS commands include:

$ CRAY STATUS/OWNER	lists any jobs currently running or waiting in the batch queues and gives job sequence (jsq) number
$ CRAY SHOW QUEUE	show the status of jobs in the input queue and returns their queue identification (qid) numbers
$ CBATCH jobfilename	submits a JOB file to the Cray batch queues
$ CRAY SET TERMINAL INFORM	permits display of a message whenever a file transfer between Cray and VAX occurs
$ CRAY KILL jsq	stops a job and deletes its output files
$ CRAY RELEASE jsq	releases a job in holding status on the Cray
$ CRAY REMOVE qid	removes the entry from the input queue

Programs are submitted to the Cray in batch. There are certain statements which must be included in each job, including the Cray account number and the user's password. CBATCH is a utility developed at NRL to submit jobs to the Cray. CBATCH uses a single encrypted file (**$CRAY$.ACCOUNT**) to store the required information (username, account number and password) and is the most secure method of submitting jobs. To submit a job using CBATCH, type:

```
$ CBATCH job_file_name
```

Cray passwords are set to expire every 90 days. When the password expires, you are required to change it before Cray jobs will run. To change the Cray password, the /NUPW qualifier is used on the CBATCH command:

```
$ CBATCH/NUPW job_file_name
New user password (NUPW): new_password
```

3.6.3 Create COS Job Control Files

All source code and data files (datasets) must be transferred (staged) to the Cray from the VAX front-end before compilation and execution can occur. Files on the Cray are considered either local or permanent. Local files are known only to a particular job and are lost when the job completes, unless they are specifically saved. Permanent files are contained in the Cray Master File Directory and must be made local in order to be accessed by a job.

The COS Job Control Language (JCL) is used to create JOB files to run programs on the Cray. The basic commands are as follows:

CFT77	compiles FORTRAN code
SEGLDR	loads relocatable binaries into Cray memory
FETCH	transfers a front-end file to the Cray (local)
DISPOSE	transfers a local file from the Cray to the front-end
SAVE	makes a local file permanent
ACCESS	makes a permanent file local to a job
AUDIT	displays a listing of permanent files
DELETE	deletes a permanent dataset
ASSIGN	used to assign files to Fortran logical unit numbers
EXIT	allows for exit processing

The Cray JOB files created for the FINDIF code are included in Appendix C.

The FINDIF code consists of three programs. The first program, SCNTPREP, runs on the VAX front-end, and creates the include files needed to run the other programs on the Cray. Input to SCNTPREP is the file CNT01.PAR which describes the model to be calculated.

The next program, SCNTBNY, runs on the Cray, and creates a file CNT01.BNY which is required by the final program SCNTDIF. All of the subroutines which do not require re-compilation at run-time were transferred to the Cray, compiled and included in a binary module library.

3.6.4 Converting FORTRAN programs to run under COS

Several modifications were made to the standard FORTRAN 77 FINDIF programs. When a program is compiled on the Cray, all INCLUDE files must be local (local dataset names consist of up to 7 characters, the first character must be an uppercase A through Z). Therefore, the INCLUDE statements in the source code must contain the local dataset name (the name of the file that resides on the Cray), not the name of the file that resides on the VAX front-end.

The OPEN statements were removed from SLOGOUT.FOR and replaced by COS ASSIGN statements in the .JOB files. Unit 54 (CNT01.BNY) is for unformatted output. Unit 55 (CNT01.PAR) is for input. Unit 66 is the output log file (CNT01.LG1).

3.6.5 Running FINDIF on the Cray

The CBATCH utility can also be invoked with options to change the default parameters that the system uses to schedule and process the job. The CBATCH commands used to run the 3 programs in the FINDIF code were:

```
$ CBATCH/P=2 PREP.JOB
$ CBATCH/P=2/MFL=12288000 BNY.JOB
$ CBATCH/P=2/MFL=12288000/T=800 DIFF.JOB
```

The **/P=2** option submits the job at a priority level of 2 (deferred priority). The **/MFL=12288000** qualifier specifies that the maximum amount of memory available on the Cray is to be used for the job. The **/T=800** qualifier indicates the job's Cray CPU time limit in seconds (default is 60 seconds).

The time series data are created on the Cray and then transferred back to the VAX front-end at the end of the job in a file called **diff.log**. The timing statistics are found in the file **diff.cpr**.

The results of the Cray run of the FINDIF code are summarized in Appendix B. The values for **A1DATA** and **BMAX** are considerably different from those calculated on other platforms. The reason for this is unknown, although it should be mentioned that all arithmetic on the Cray is calculated with 64-bit words.

The benchmark code was run again on the Convex using the **-pd8** (and no optimization) compiler option, which causes all arithmetic to be performed in double precision. The values of **BMAX** and **A1DATA** were only slightly different from the Convex run with single precision arithmetic and no optimization. The Cray results therefore remain suspicious, assuming that the 64-bit arithmetic on the Cray is equivalent to the double precision arithmetic on the Convex. It was not possible to resolve the problem with the FINDIF code on the Cray for this report.

4 Summary

The FINDIF code was run successfully on a number of hardware platforms and with various degrees of compiler optimization. The values of **BMAX** (maximum value of the vertical displacement on the whole grid at a given timestep) and **A1DATA** (array that contains the time series at each receiver location) were output to a file after timestep 201. The values of **BMAX** and **A1DATA** from each benchmark run are compared in Appendix B. On the VAX/VMS machines, the results were always exactly the same; on the UNIX machines there were slight differences, probably due to floating point round-off. In general, comparison of the results between different platforms was good within 4 to 5 digits of precision. The only exception was the Cray, whose values were very different from the rest.

The VAX 9000 (using the vectorizing compiler) and the Cray X-MP ran the benchmark code the fastest (approximately 8 minutes), but the results of the Cray run are suspect. The IBM 6000 (optimized) and the Convex (vectorized) runs were next in speed at about 13.5 - 17.5 minutes. Figure 1 shows the relative speeds of the hardware platforms discussed in this report.

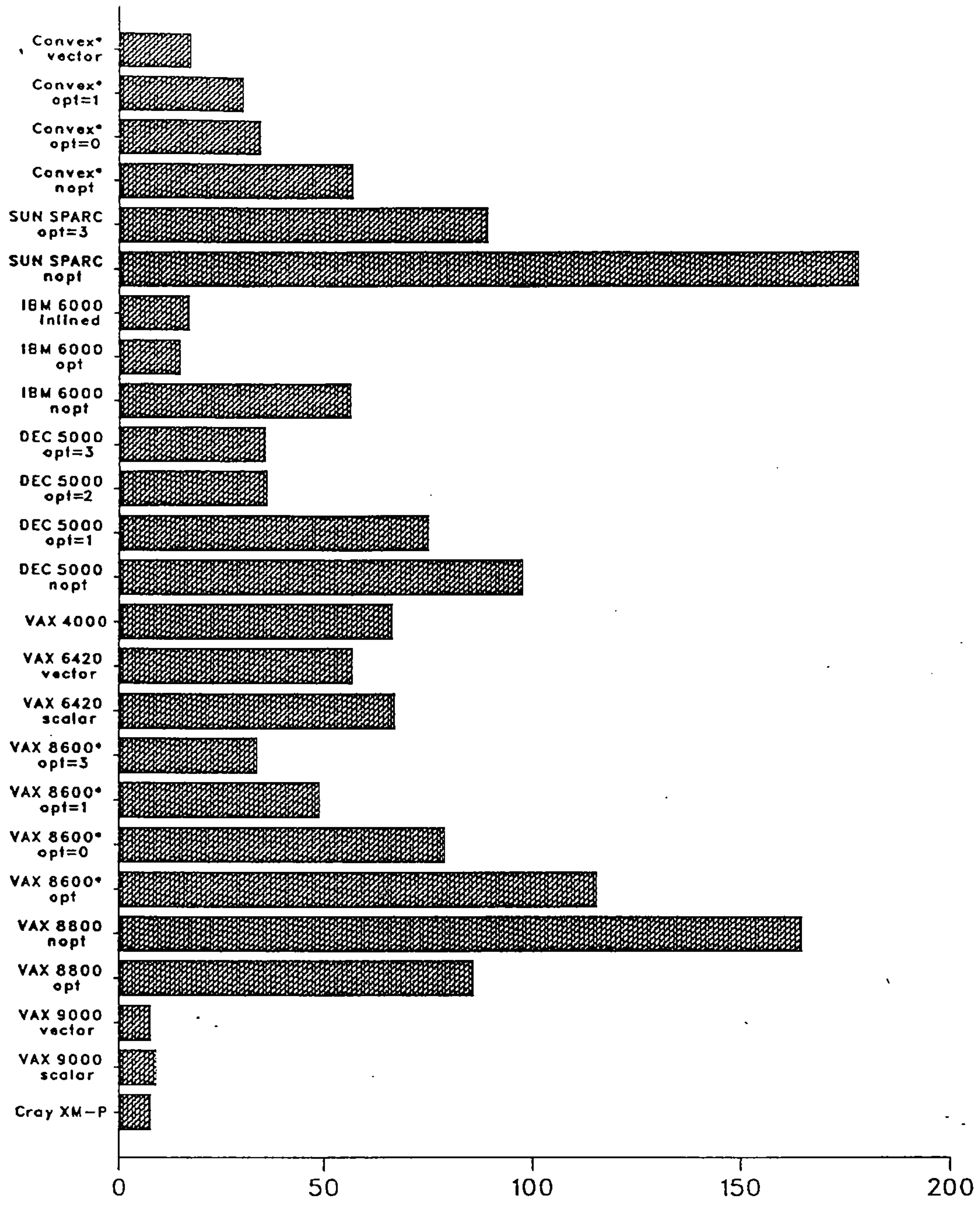

Figure 1: Comparison of benchmark times (in minutes) for the FINDIF code (* denotes elapsed time; no star denotes CPU time)

Appendices

A Parameter and Batch files

The following parameter file contains all of the input information required for the benchmark FINDIF program (Hunt, et al, 1983).

```
  CNT01  -  SOFT BOTTOM  -  HOMO  -  THIN
CNT01
1,  1,  0,   -2,  1,  0,   0,   1
1201,   401,   201,   1,   0.000625,   0.005,   0.005
1,  201,  1,  1201,  12,  142
8,  50,  10,  50,  50
1.50,   0.0,   1.0,   1.7,   0.45,  1.42,  21,  223, 0.0, 0.0, 0.0
12,  1, 11,  2
11,  657.0,  0.146
```

The batch command file used to run the benchmark FINDIF code on a computer running VAX/VMS is:

```
$!
$! Command file to run 2D finite differences programs
$! 10 May 1990
$!
$! Define directories
$!
$  DEFINE/TRANS=CONCEAL BENCH RNR2:[FIND.WSL1.JULIE.]
$!
$  PUR *.LOG
$  PUR *.BNY
$  PUR *.LG*
$  PUR *.TSV
$  PUR *.VRT
$  PUR *.FOR
$  PUR *.TSP
$  PUR *.TSH
$  PUR *.HRZ
$!
$  SET DEFAULT BENCH:[CNT01]
$  ASSIGN BENCH:[CNT01]CNT01.PAR FOR055 !input parameter file
$  ASSIGN BENCH:[CNT01]CNT01.BNY FOR054 !output data file
$  RUN BENCH:[PREP]SCNTPREP
$!
$  set verify
$  COPY BENCH:[CNT01]*.FOR BENCH:[BNY]*.*
$  SET DEFAULT BENCH:[BNY]
$  FOR SCNTBNY01
$  @SCNTBNY01.LNK
$  SET DEFAULT BENCH:[CNT01]
$  RUN BENCH:[BNY]SCNTBNY01
$!
$  COPY BENCH:[CNT01]*.FOR BENCH:[DIFF]*.*
$  SET DEFAULT BENCH:[DIFF]
$  FOR SCNTDIF.FOR
$  FOR SCNTTS9.FOR
$  FOR SCNTSUB7.FOR
$  @SCNTDIF.LNK
$  SET DEFAULT BENCH:[CNT01]
$  RUN BENCH:[DIFF]SCNTDIF
$  set noverify
$!
$  EXIT
$!
$! End of command file
$!
```

The shell script used to run the benchmark FINDIF code on the Convex:

```
#
cd /mnt7/julie/benchmark/cnt01
rm *.log
rm *.BNY
rm *.LG*
rm *.TSV
rm *.VRT
rm *.FOR
rm *.TSP
rm *.TSH
rm *.HRZ
setenv FOR055 CNT01.PAR
setenv FOR054 CNT01.BNY
/mnt7/julie/benchmark/prep/scntprep
cp *.FOR /mnt7/julie/benchmark/bny
cd /mnt7/julie/benchmark/bny
make scntbny01
cd /mnt7/julie/benchmark/cnt01
/mnt7/julie/benchmark/bny/scntbny01
cp *.FOR /mnt7/julie/benchmark/diff
cd /mnt7/julie/benchmark/diff
touch scntdif.f
touch scntts9.f
touch scntsub7.f
make scntdif
cd /mnt7/julie/benchmark/cnt01
/mnt7/julie/benchmark/diff/scntdif
```

B Accuracy of Results

Table 2 shows a comparison of the BMAX values at four different timesteps. BMAX corresponds to vertical displacement; it is the maximum value on the whole grid at a particular timestep.

Table 2: Comparison of BMAX values for selected time steps

Computer	BMAX Timestep=56, Point=1	BMAX Timestep=104 Point=1	BMAX Timestep=152 Point=1	BMAX Timestep=200 Point=1
VAX 9000	0.1041646E+05	0.4976261E+06	0.6936726E+07	0.2869944E+08
VAX 8800	0.1041646E+05	0.4976261E+06	0.6936726E+07	0.2869944E+08
VAX 8600				0.2869944E+08
DEC 5000	0.1041644E+05	0.4976252E+06	0.6936703E+07	0.2869932E+08
IBM 6000	0.1041643E+05	0.4976249E+06	0.6936704E+07	0.2869926E+08
SUN SPARC	0.1041644E+05	0.4976251E+06	0.6936702E+07	0.2869933E+08
Convex (no)	0.1041643E+05	0.4976253E+06	0.6936711E+07	0.2869925E+08
Convex (s0)	0.1041643E+05	0.4976253E+06	0.6936711E+07	0.2869925E+08
Convex (s1)	0.1041644E+05	0.4976253E+06	0.6936706E+07	0.2869930E+08
Convex v6 (no)	0.1041644E+05	0.4976253E+06	0.6936706E+07	0.2869932E+08
Convex v6 (s0)	0.1041643E+05	0.4976253E+06	0.6936711E+07	0.2869925E+08
Convex v6 (s1)	0.1041644E+05	0.4976253E+06	0.6936706E+07	0.2869930E+08
Convex v6 (v)	0.1041644E+05	0.4976249E+06	0.6936712E+07	0.2869922E+08
Convex (dp)	0.1041644E+05	0.4976249E+06	0.6936703E+07	0.2869928E+08
Cray X-MP	0.1041644E+05	0.4976280E+06	0.6936852E+07	0.2870085E+08

no	no optimization
O0	VAX 8600 optimization level 0
O1	VAX 8600 optimization level 1
O3	VAX 8600 optimization level 3
s0	Convex 220 optimization level scalar 0
s1	Convex 220 optimization level scalar 1
v	Convex 220 optimization level vectorized
v6	Convex operating system version 6.0
dp	double precision

Table 3 is a comparison of the values of the array A1DATA at selected grid points at the end of the run (timestep 201). A1DATA is the array that contains the time series at each receiver location; it corresponds to vertical displacement.

Table 3: Performance statistics at time step 201

Computer	M=1,N=12	M=51,N=12	M=1,N=22	M=51,N=22
VAX 9000	1.0506451E+07	1.8830447E-08	60360.65	3.8512233E-08
VAX 8800	1.0506451E+07	1.8830447E-08	60360.65	3.8512233E-08
VAX 8600 (no)	1.0506451E+07	1.8830447E-08	60360.65	3.8512233E-08
VAX 8600 (O0)				
VAX 8600 (O1)				
VAX 8600 (O3)				
DEC 5000	1.0506432E+07	1.8830420E-08	60361.45	3.8511907E-08
IBM 6000	1.0506366E+07	1.8830317E-08	60360.61	3.8511843E-08
SUN SPARC	1.05064 E+07	1.88304 E-08	60359.8	3.85119 E-08
Convex (no)	1.0506382E+07	1.8830448E-08	60360.71	3.8511896E-08
Convex (s0)	1.0506382E+07	1.8830448E-08	60360.71	3.8511896E-08
Convex (s1)	1.0506408E+07	1.8830358E-08	60360.59	3.8511885E-08
Convex v6 (no)	1.0506388E+07	1.8830358E-08	60360.50	3.8511867E-08
Convex v6 (s0)	1.0506382E+07	1.8830448E-08	60360.71	3.8511896E-08
Convex v6 (s1)	1.0506408E+07	1.8830358E-08	60360.59	3.8511885E-08
Convex v6 (v)	1.0506348E+07	1.8830351E-08	60360.80	3.8511939E-08
Convex (dp)	1.0506393E+07	1.8830345E-08	60360.42	3.8511928E-08
Cray X-MP	1.0507879E+07	1.8821599E-08	56352.02	1.1813252E-08

Computer	M=1,N=32	M=51,N=32	M=1,N=42	M=51,N=42
VAX 9000	2819.799	2.7487550E-09	207.6557	3.8319286E-14
VAX 8800	2819.799	2.7487550E-09	207.6557	3.8319286E-14
VAX 8600	2819.799	2.7487550E-09	207.6557	3.8319286E-14
DEC 5000	2819.736	2.7487332E-09	207.6548	3.8318961E-14
IBM 6000	2819.782	2.7487286E-09	207.6547	3.8318883E-14
SUN SPARC	2819.80	2.74873 E-09	207.655	3.83190 E-14
Convex (no)	2819.763	2.7487332E-09	207.6545	3.8318961E-14
Convex (s0)	2819.763	2.7487332E-09	207.6545	3.8318961E-14
Convex (s1)	2819.743	2.7487337E-09	207.6547	3.8318957E-14
Convex v6 (no)	2819.741	2.7487326E-09	207.6548	3.8318957E-14
Convex v6 (s0)	2819.763	2.7487332E-09	207.6545	3.8318961E-14
Convex v6 (s1)	2819.743	2.7487337E-09	207.6547	3.8318957E-14
Convex v6 (v)	2819.790	2.7487324E-09	207.6548	3.8318961E-14
Convex (dp)	2819.775	2.7487343E-09	207.6547	3.8319003E-14
Cray X-MP	372.265	2.7143833E-13	2.009895	9.0819405E-24

Table 3, continued.

Computer	M=1,N=52	M=51,N=52	M=1,N=62	M=51,N=62
VAX 9000	3.038062	1.3996069E-21	1.4176460E-04	3.9228071E-31
VAX 8800	3.038062	1.3996069E-21	1.4176460E-04	3.9228071E-31
VAX 8600	3.038062	1.3996069E-21	1.4176460E-04	3.9228071E-31
DEC 5000	3.038037	1.3995938E-21	1.4176313E-04	3.9227662E-31
IBM 6000	3.038038	1.3995912E-21	1.4176310E-04	3.9227601E-31
SUN SPARC	3.03804	1.39959 E-21	1.41763 E-04	3.92277 E-31
Convex (no)	3.038034	1.3995939E-21	1.4176313E-04	3.9227619E-31
Convex (s0)	3.038034	1.3995939E-21	1.4176313E-04	3.9227619E-31
Convex (s1)	3.038038	1.3995938E-21	1.4176322E-04	3.9227610E-31
Convex v6 (no)	3.038039	1.3995936E-21	1.4176317E-04	3.9227615E-31
Convex v6 (s0)	3.038034	1.3995939E-21	1.4176313E-04	3.9227619E-31
Convex v6 (s1)	3.038038	1.3995938E-21	1.4176322E-04	3.9227610E-31
Convex v6 (v)	3.038034	1.3995938E-21	1.4176316E-04	3.9227624E-31
Convex (dp)	3.038037	1.3995968E-21	1.4176334E-04	3.9227794E-31
Cray X-MP	2.2160686E-03	3.4794050E-37	1.0939469E-07	5.0383479E-52

Computer	M=1,N=72	M=51,N=72	M=1,N=82	M=51,N=82
VAX 9000	1.0583878E-11	0.	7.1139738E-21	0.
VAX 8800	1.0583878E-11	0.	7.1139738E-21	0.
VAX 8600	1.0583878E-11	0.	7.1139738E-21	0.
DEC 5000	1.0583762E-11	1.7025776E-42	7.1138930E-21	0.
IBM 6000	1.0583759E-11	1.7011763E-42	7.1138890E-21	0.
SUN SPARC	1.05838 E-11	1.70258 E-42	7.11389 E-21	0.
Convex (no)	1.0583762E-11	0.	7.1138938E-21	0.
Convex (s0)	1.0583762E-11	0.	7.1138938E-21	0.
Convex (s1)	1.0583767E-11	0.	7.1138938E-21	0.
Convex v6 (no)	1.0583764E-11	0.	7.1138946E-21	0.
Convex v6 (s0)	1.0583762E-11	0.	7.1138938E-21	0.
Convex v6 (s1)	1.0583767E-11	0.	7.1138938E-21	0.
Convex v6 (v)	1.0583765E-11	0.	7.1138946E-21	0.
Convex (dp)	1.0583786E-11	1.7108603E-42	7.1139143E-21	1.8141990E-55
Cray X-MP	2.5145367E-14	7.0540859E-67	4.2665326E-23	7.6653262E-83

Computer	M=1,N=92	M=51,N=92	M=1,N=102	M=51,N=102
VAX 9000	1.0261487E-31	0.	0.	0.
VAX 8800	1.0261487E-31	0.	0.	0.
VAX 8600	1.0261487E-31	0.	0.	0.
DEC 5000	1.0261366E-31	0.	5.6051939E-44	0.
IBM 6000	1.0261362E-31	0.	4.6242849E-44	0.
SUN SPARC	1.02614 E-31	0.	5.60519 E-44	0.
Convex (no)	1.0261380E-31	0.	0.	0.
Convex (s0)	1.0261380E-31	0.	0.	0.
Convex (s1)	1.0261380E-31	0.	0.	0.
Convex v6 (no)	1.0261382E-31	0.	0.	0.
Convex v6 (s0)	1.0261380E-31	0.	0.	0.
Convex v6 (v)	1.0261387E-31	0.	0.	0.
Convex (dp)	1.0261406E-31	6.0940049E-70	5.2429849E-44	7.3540342E-86
Cray X-MP	2.4992885E-33	5.6515166E-100	1.1158869E-45	2.7138691E-118

C Cray JOB files

This appendix contains listings of the Cray JOB files used to run the benchmark programs on the Cray XM-P at the Naval Research Laboratory.

```
*
* BNY.JOB - compile, link and run scntbny
*
* Written by J. Allen
* 27 September 1990
*
FETCH, DN=SRC, TEXT = 'SCNTBNY.FOR'.
FETCH, DN=SMODPAR, TEXT = 'SMODPAR.INC'.
FETCH, DN=SCOMFD8, TEXT = 'COMFD8.INC'.
FETCH, DN=SIOUNIT, TEXT = 'SIOUNIT.INC'.
CFT77, I=SRC, ON=MX, L=LISTDN.
DISPOSE, DN=LISTDN, TEXT = 'SCNTBNY.LIS'.
ACCESS, DN=FDOLB.
SEGLDR, LIB=FDOLB, ABS=BNYEXE, MAP=STAT.
SAVE, DN=BNYEXE.
FETCH, DN=CNT01, TEXT = 'CNT01.PAR'.
ASSIGN, DN=CNT01, A=FT55.
ASSIGN, DN=BNYOUT, A=FT54.
ASSIGN, DN=BNYLOG, A=FT66.
BNYEXE.
SAVE, DN=BNYOUT.
DISPOSE, DN=BNYLOG, TEXT='BNY.LOG'.
EXIT.
```

```
*
* DIFF.JOB - compile, link and run scntdif
*
* Written by J. Allen
* 27 September 1990
*
FETCH, DN=SRC, TEXT = 'SCNTDIF.FOR'.
FETCH, DN=SMODPAR, TEXT = 'SMODPAR.INC'.
FETCH, DN=SCOMFD8, TEXT = 'COMFD8.INC'.
FETCH, DN=SIOUNIT, TEXT = 'SIOUNIT.INC'.
FETCH, DN=SCBLCK, TEXT = 'SCBLCK.INC'.
CFT77, I=SRC, ON=MX, L=LISTDN.
DISPOSE, DN=LISTDN, TEXT = 'SCNTDIF.LIS'.
FETCH, DN=SUB1, TEXT = 'SCNTTS9.FOR'.
CFT77, I=SUB1, ON=MX, L=LISTDN.
```

```
DISPOSE, DN=LISTDN, TEXT = 'SCNTTS9.LIS'.
FETCH, DN=SUB2, TEXT = 'SCNTSUB7.FOR'.
CFT77, I=SUB2, ON=MX, L=LISTDN.
DISPOSE, DN=LISTDN, TEXT = 'SCNTSUB7.LIS'.
FETCH, DN=SUB3, TEXT = 'ZDIVCRL.FOR'.
CFT77, I=SUB3, ON=MX, L=LISTDN.
DISPOSE, DN=LISTDN, TEXT = 'ZDIVCRL.LIS'.
ACCESS, DN=FDOLB.
SEGLDR, LIB=FDOLB, ABS=DIFEXE, MAP=STAT.
FETCH, DN=CNT01, TEXT = 'CNT01.PAR'.
ASSIGN, DN=CNT01, A=FT55.
ACCESS, DN=BNYOUT.
ASSIGN, DN=BNYOUT, A=FT54.
ASSIGN, DN=DIFLOG, A=FT66.
DIFEXE.
DISPOSE, DN=DIFLOG, TEXT='DIF.LOG'.
SAVE, DN=HRZ0005.
SAVE, DN=VRT0005.
SAVE, DN=HRZ0010.
SAVE, DN=VRT0010.
SAVE, DN=HRZ0015.
SAVE, DN=VRT0015.
SAVE, DN=HRZ0020.
SAVE, DN=VRT0020.
SAVE, DN=CNT01TSH.
SAVE, DN=CNT01TSP.
SAVE, DN=CNT01TSV.
AUDIT.
EXIT.


*
* PREP.JOB - compile, link and run scntprep
*
* Written by J. Allen
* 26 September 1990
*
FETCH, DN=SRC, TEXT = 'SCNTPREP.FOR'.
FETCH, DN=SMODPAR, TEXT = 'SMODPAR.INC'.
CFT77, I=SRC, ON=MX, L=LISTDN.
DISPOSE, DN=LISTDN, TEXT = 'SCNTPREP.LIS'.
ACCESS, DN=FDLIB, PDN=FDOLB.
SEGLDR, LIB=FDLIB, ABS=PREPEXE, MAP=STAT.
SAVE, DN=PREPEXE.
FETCH, DN=CNT01, TEXT = 'CNT01.PAR'.
ASSIGN, DN=CNT01, A=FT55.
ASSIGN, DN=COMFD8, A=FT02.
ASSIGN, DN=COMSOR, A=FT03.
```

```
PREPEXE.
DISPOSE, DN=COMFD8, TEXT='COMFD8.INC'.
DISPOSE, DN=COMSOR, TEXT='COMSOR.INC'.
DELETE, PDN=PREPEXE, ED=-1.
EXIT.
```

Acknowledgement

We wish to thank the following for providing computing resources for the tests of the FINDIF code:

SACLANT Undersea Research Center for the tests on the VAX 8600 and FPS264,

Naval Research Laboratory Central Computing Facility for use of the Cray X-MP, and

Digital Equipment Corporation for running the benchmark on the VAX 4000, VAX 8800 and VAX 9000.

We also thank W.S. Little for the valuable assistance he provided during the preparation of this report.

This work was funded by the Office of Naval Research under Grant Number N00014-89-J-1012.

References

Hunt, Mary M., Lee Gove and Ralph A. Stephen, 1983, FINDIF: A Software Package to Create Synthetic Seismograms by Finite Differences, Woods Hole Oceanographic Institution Technical Report, WHOI-83-42, 45pp.

DOCUMENT LIBRARY

March 11, 1991

Distribution List for Technical Report Exchange

Attn: Stella Sanchez-Wade
Documents Section
Scripps Institution of Oceanography
Library, Mail Code C-075C
La Jolla, CA 92093

Hancock Library of Biology &
Oceanography
Alan Hancock Laboratory
University of Southern California
University Park
Los Angeles, CA 90089-0371

Gifts & Exchanges
Library
Bedford Institute of Oceanography
P.O. Box 1006
Dartmouth, NS, B2Y 4A2, CANADA

Office of the International
Ice Patrol
c/o Coast Guard R & D Center
Avery Point
Groton, CT 06340

NOAA/EDIS Miami Library Center
4301 Rickenbacker Causeway
Miami, FL 33149

Library
Skidaway Institute of Oceanography
P.O. Box 13687
Savannah, GA 31416

Institute of Geophysics
University of Hawaii
Library Room 252
2525 Correa Road
Honolulu, HI 96822

Marine Resources Information Center
Building E38-320
MIT
Cambridge, MA 02139

Library
Lamont-Doherty Geological
Observatory
Columbia University
Palisades, NY 10964

Library
Serials Department
Oregon State University
Corvallis, OR 97331

Pell Marine Science Library
University of Rhode Island
Narragansett Bay Campus
Narragansett, RI 02882

Working Collection
Texas A&M University
Dept. of Oceanography
College Station, TX 77843

Library
Virginia Institute of Marine Science
Gloucester Point, VA 23062

Fisheries-Oceanography Library
151 Oceanography Teaching Bldg.
University of Washington
Seattle, WA 98195

Library
R.S.M.A.S.
University of Miami
4600 Rickenbacker Causeway
Miami, FL 33149

Maury Oceanographic Library
Naval Oceanographic Office
Stennis Space Center
NSTL, MS 39522-5001

Marine Sciences Collection
Mayaguez Campus Library
University of Puerto Rico
Mayaguez, Puerto Rico 00708

Library
Institute of Oceanographic Sciences
Deacon Laboratory
Wormley, Godalming
Surrey GU8 5UB
UNITED KINGDOM

The Librarian
CSIRO Marine Laboratories
G.P.O. Box 1538
Hobart, Tasmania
AUSTRALIA 7001

Library
Proudman Oceanographic Laboratory
Bidston Observatory
Birkenhead
Merseyside L43 7 RA
UNITED KINGDOM

Mac 90-32

50272-101

REPORT DOCUMENTATION PAGE	1. REPORT NO. WHOI-91-31	2.	3. Recipient's Accession No.
4. Title and Subtitle Benchmarking the Two-dimensional Finite Difference Synthetic Seismogram Code			5. Report Date September 1991
			6.
7. Author(s) J.M. Allen and R.A. Stephen			8. Performing Organization Rept. No. WHOI-91-31
9. Performing Organization Name and Address Woods Hole Oceanographic Institution Woods Hole, Massachusetts 02543			10. Project/Task/Work Unit No.
			11. Contract(C) or Grant(G) No. (C) N00014-89-J-1012 (G)
12. Sponsoring Organization Name and Address Office of Naval Research			13. Type of Report & Period Covered Technical Report
			14.

15. Supplementary Notes

This report should be cited as: Woods Hole Oceanog. Inst. Tech. Rept., WHOI-91-31.

16. Abstract (Limit: 200 words)

During the past six months, the two-dimensional finite difference synthetic seismogram code was installed and run on a number of different computer systems. The results were compared for timing, accuracy and the ease with which the code was adapted to each system.

This report documents the software modifications and the methods used to implement the finite difference code on each computer, and presents the results of the benchmark survey.

17. Document Analysis a. Descriptors

synthetic seismograms
finite differences
benchmarks

b. Identifiers/Open-Ended Terms

c. COSATI Field/Group

18. Availability Statement Approved for public release; distribution unlimited.	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages 23
	20. Security Class (This Page)	22. Price