Technical Memorandum No. 4-86
A User's Manual for Finite Difference Synthetic Seismogram Codes on the CYBER 205 and CRAY XMP-12
Mary M. Hunt
Ralph A. Stephen
November 1986
A User's Manual for Finite Difference Synthetic Seismogram Codes on the CYBER 205 and CRAY XMP-12

by

Mary M. Hunt
Ralph A. Stephen

Woods Hole Oceanographic Institution
Woods Hole, Massachusetts 02543

November 1986

TECHNICAL MEMORANDUM

Funding was provided by the National Science Foundation under grant Nos. OCE-8117571 and OCE-8409155; and by the Office of Naval Research under contract No. N00014-85-C-001NR.
A USER'S MANUAL FOR FINITE DIFFERENCE
SYNTHETIC SEISMOGRAM CODES ON
THE CYBER 205 and CRAY XMP-12

Mary M. Hunt
Ralph A. Stephen
CONTENTS

Abstract

A. FINDIF on the CYBER 205
   Introduction
   To Log on to the Purdue System
   File Handling on the Front-End
   Control Characters on the Front-End
   Job Submission and Retrieval
   Implementing the INCLUDE
   Tapes
   Steps for Running FINDIF on Cyber 205

B. FINDIF on the CRAY-XMP 12

   I. Introduction
   II. General System Description
       How to Log-on
       Intercomputer Communications
       Some CRAY Control Commands
       Tapes
       Comparison to Cyber

   III. Implementation of FINDIF Software
       Programs
       Implementing the INCLUDE
       Vectorization
       Permanent Files on VAX
       Permanent Files on CRAY

   IV. Steps to Run a Model

C. Acknowledgements

D. References

   Appendix I: Summary of CYBER use at the Purdue University Computer Center

   Appendix II: NRL-Central Computer Facility Log-on Procedures
Abstract

Over the past eight years, a software package has been developed to solve the elastic wave equation by the method of finite differences (Hunt et al., 1983; Stephen, 1983; Stephen, 1984a; Stephen, 1984b; Nicoletis, 1981). The elastic wave equation can be solved in two dimensions for point sources in cylindrical coordinates or line sources in rectangular coordinates. Compressional and shear velocity and density are allowed to vary both vertically and radially.

Since the code is very computationally intensive for realistic size models, it has been implemented on two Class VI super computers: the Cyber 205 at Purdue University and the Cray XMP-12 at the Naval Research Laboratory. This technical report is a user's manual for running the code on these machines. (It is assumed that the reader is already familiar with running the code on the VAX 11-780) (Hunt et al., 1983).
A. FINDIF on the CYBER 205

(Based on Notes by Mary M. Hunt - May, 1985)

Introduction

The CYBER system at Purdue includes two different computers and two different operating systems. All jobs must be submitted through a Front-End machine, which is a Cyber 6600 with the MACE Operating System (developed at Purdue). Interactive use, except for simple jobs like editing, is not encouraged even on the Front End. The main differences, for our purposes, between the Purdue system and the usage on the VAX are:

1. Method of including alternate code during compilation (INCLUDE).
2. No interactive operation on Cyber.
3. Greater difficulty of use of the Cyber operating systems.

The Finite Difference software, as implemented at Purdue, includes four programs. Operation is similar to that on the VAX, but different enough to make it necessary to document the Cyber usage. This document has two parts. The first explains the Cyber features which will be used the most, and the second gives step by step instructions for running the FINDIF software.
To Log on to the Purdue System

1. Set your terminal to 1200 Baud.

2. Neither Telenet nor Purdue allows type ahead.

3. You can call Telenet through the PACX or through a modem. 
   Dial Telenet 540-7500 (or 7501, 7502, 7503, 7504) 
   (or dial direct 1-800-424-9494)

4. When you are on-line, strike the RETURN key twice. Telenet responds

   TERMINAL =
   Enter D1 for VT100 type terminal, or A8 for the IPC Decwriter.

5. Telenet prompts with @
   Enter ID ;31264/PURDUE118
   Telenet asks for PASSWORD
   You should now be connected with PURDUE.

6. Purdue asks Port type?
   Enter P, no carriage return until the system completes the word PROCSY.

7. System asks for ACCOUNT? which is 60118
   USER ID? which is BJH
   PASSWORD?

8. Purdue asks for SYSTEM? Respond with RETURN.

   However, if you are using the Decwriter, and want to use all 132 columns,
   respond to SYSTEM? WITH TTYSET,WIDTH=132

9. You are almost ready to do your thing, but before proceeding, you should
   send a few commands to Telenet to allow control characters to get to
   Purdue. First, strike CR @ CR. You should get the Telenet prompt (@).
   Enter the following commands:

   ENAB TFLO
   SET 3:254
   PAGE 132 66 if you want 132 column output
   CONT to get back to Purdue.

10. You are now all set. A sample of this procedure is given in Figure 1.

11. To log off the system: LOG. Then you can turn off your terminal.
Figure 1
Sample Log On

TELENET
:17 17C

TERMINAL=A8

ID ;31264/PURDUE118
PASSWORD =

312 64 CONNECTED

Port type? PROCSY
TCB R472 16,00,08. 04/24/85. HALF DUPLEX
ACCOUNT? 60118
USER ID? BJH
PASSWORD?
THIS USERID LAST LOGGED OFF AT 15,44,10. 04/24/85.
SYSTEM? TTYSET,WIDTH=132
SYSTEM?
PIRATE
++
TELENET
@ENAB TFLO
@SET 3:254
@PAGE 132 66
@CONT
+++GET DOPREP
File Handling on the Front-End

In order to access a file, a copy of the file must be moved into the user's local work area. This is done with the GET command. For example, if you want to edit a file named JOBFL, you must first issue the command

```
GET, JOBFL
```

to get a copy of the file. Then, after you have edited it, if you want to store the new copy back into permanent file storage, use the command

```
PUT, JOBFL
```

If you forget the PUT, your edited version of the file will disappear at the end of your job.

Control Characters on the Front-End

Use **CTR B** to stop a job, equivalent to **CTR Y** on the VAX

Use **CTR H** to delete previous character in line

Job Submission and Retrieval

To submit a job, use the command

```
XMIT,xxxx,nnnnnn
```

where

- **xxxx** is a job identification, can be any 4 characters
- **nnnnnn** is the name of the job file, must be a local file

This same procedure is used for both Front-End jobs and 205 jobs. You should not have two jobs with the same identification in the system at the same time. If you want to wait for the job to run, enter the command

```
PREVIEW xxxx
```

When a job runs, it creates a DAYFILE, the same as a .LOG file on the VAX. This file will not remain in the system for much more than 1/2 hour, so to save it to look at later, you must ROUTE it to WAIT, but you must give it a numeric identification:

```
ROUTE xxxx TO WAIT AS mmmm
```

where mmmm is numeric

You can check to see if a job has run with the command

```
SEARCH
```

If the letters PR follow the job id., then the job has run. To look at it, you must

```
ROUTE mmmm FROM WAIT TO PROCSY
```
Job Submission and Retrieval (Continued)

Then you can PREVIEW mmmm

If you want it printed out at PURDUE,

ROUTE mmmm TO MATH AS 2000.

They will mail it to you.

Implementing the INCLUDE

The procedure required to include COMMON files into a source deck, using UPDATEP, is a rather cumbersome two-step process. I have worked it out so it is more or less automatic.

First, your basic source file is subdivided into COMDECK's and DECK's, as in Figure 2. COMMON blocks which do not change are completely specified, but variable blocks remain undefined. The first step is to incorporate the COMDECK's into the DECK's, to create what they call the Program Library. There is a job-file to do this step for each program, as shown in Table I. These create the files BNYLIB, SORLIB, and FINLIB. Probably the programs should be stored this way. However, if they will need any editing, you might as well use the QED editor, and start from the DECK files.

Then, after you have run FDPREP to create the COMMON files, you can incorporate the COMMON file into the library file to create a COMPILE file. All the files for this procedure are given in Table I. In the case of both FDBNY and FINDIF, this second step also incorporates the code for the variable program loop. The job files for this step read the update code from the files created by FDPREP, and read the variable loop code from the specified files. This second job creates the files which are sent to the 205 for compilation.
Figure 2

Form of Source File

<table>
<thead>
<tr>
<th>COMMON /MODPAR/</th>
<th>TRECT, IDENS, IFLAT, IKELY</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMMON /MODPAR/</td>
<td>IEXPL, IVERT, ITRAV, ISORB, ISNST</td>
</tr>
<tr>
<td></td>
<td>etc.</td>
</tr>
<tr>
<td>COMMON /IOUNIT/</td>
<td>LULOG, LUINP, LUBNY, LUSRC</td>
</tr>
<tr>
<td>COMMON /IOUNIT/</td>
<td>LUSNP, LUTST</td>
</tr>
<tr>
<td>COMMON /COMFOR/</td>
<td></td>
</tr>
<tr>
<td>COMMON /COMFOR/</td>
<td></td>
</tr>
<tr>
<td>DECK FINDFE</td>
<td></td>
</tr>
<tr>
<td>PROGRAM FINDIE</td>
<td></td>
</tr>
</tbody>
</table>

*CALL MODPAR
*CALL IOUNIT
*CALL COMFOR

END

*DECK ABSSOR
SUBROUTINE ABSSOR

*CALL MODPAR
*CALL COMFOR

END

*DECK TSTEPI
SUBROUTINE TSTEPI

*CALL MODPAR
*CALL COMFOR

END
Table I

Files used during UPDATEP procedures

<table>
<thead>
<tr>
<th>Program</th>
<th>Basic Source File</th>
<th>Job-file to create Library file</th>
<th>Name of Library file</th>
<th>Job-file to Create Compile file</th>
<th>File Containing Update code</th>
<th>Compile File</th>
</tr>
</thead>
<tbody>
<tr>
<td>FDBNY</td>
<td>FDBNY</td>
<td>MAKLIB</td>
<td>BNYLIB</td>
<td>UPDBLIB</td>
<td>BNYUPD</td>
<td>BNYFOR</td>
</tr>
<tr>
<td>FDSORS</td>
<td>FDSORS</td>
<td>MAKSLIB</td>
<td>SORLIB</td>
<td>UPDSLIB</td>
<td>SORUPD</td>
<td>SORFOR</td>
</tr>
<tr>
<td>FINDIF</td>
<td>FINDIF</td>
<td>MAKFLIB</td>
<td>FINLIB</td>
<td>UPDFLIB</td>
<td>FINUPD</td>
<td>FINFOR</td>
</tr>
</tbody>
</table>
TAPES

1. Tape Storage

Tapes are stored by slot number, which is assigned by the Tape Librarian at Purdue. When you send them a tape, or buy one from them, they will let you know the slot number. To find out what tapes you have there, enter the command

SLOT

Figure 3 is a sample of the output from SLOT.

+ + SLO T
1-0K U1D=RJH:
SLOT VSN FILE ID DATE STATUS F CV DENS VPK R-KEY W-KEY UA FS
0765 unknown
2035 unknown
2036 A1055
0777

Figure 3

2. Visual Protect Key (VPK)

Every tape has a Visual Protect Key, which you must specify for each tape. This key consists of three letters (digits not allowed). The key is written on an external sticker on the tape, and you must specify it every time you want to write on the tape.

3. Labeled Tapes

The CYBER 205 can write ANS labeled tapes, which can be read on the VAX. However, when giving the VAX the file name, you must enclose the file name in double quotes "CRS01T". They do not use file types.
4. Tape Backup on the Front-End

This is one of the nicer features of their system. We have one tape reserved for Backup (in slot 777, VPK is RAS). Also, there is a permanent file in our directory which contains information about our backup tape, and what it contains. To back-up current files, enter the command

DUMP.

System will ask for Master password.

To retrieve selected files, enter the command

LOAD, file1, file2, ...

You will receive a message when they have been loaded.

5. Copying Files to Tape on the Front-End

This appears to be fairly easy, at least for source files. Use

XTWRITE

which is documented in one of their manuals (IO MAGTAPE).

6. Creating Files on the 205

This is very messy, but the instructions included in the job-file for FINDIF appear to work. Use 6250 BPI because the files are large.

7. Tape Backup on the Back-End

To "dump" on backend submit a file like:

RESOURCE, NT = 1, JCAT = 53, TL = 1200.

ATTACH, file 1.

ATTACH, file 2.

DUMPF, file 1, file 2, DEV = NT, D = 6250, SLOT = sn/vpk/vsn.

To retrieve to backend:

RESOURCE, NT = 1, JCAT = 53, TL = 1200.

LOADPF, file 1, file 2, SEL = R, DEV = NT, D = 6250 SLOT = sn/vph/vsn.

DEFINE, file 1.

DEFINE, file 2.
Figure 4
Sample of DISPLAY, with record numbers

+++GET GETIM
GETIM  72 WORDS
+++DISPLAY GETIM

1.000= PROGRAM GETIM ( TAPE12, TAPE15 )
2.000=HALF PRECISION XOFT(15), XOPT2(2), DATA(2000)
4.000= INTEGER IOPT1(29), IOPT2(3), IOPT3(4)
5.000=C
6.000= OPEN ( UNIT=12, FILE='DOR01T', STATUS='OLD' )
7.000=READ(12,1000) IOPT1
8.000=1000 FORMAT( 812, 2115 )
9.000=READ(12,1010) IOPT2, XOFT
10.000=1010 FORMAT( 12,218,9F7.4,F6.5,2F5.4,F11.4,F8.4,F12.5 )
11.000=C
12.000=READ(12,1020) IOPT3, XOPT2
13.000=1020 FORMAT( 4110, 3F12.5 )
14.000=KCNT = IOPT3(3)
15.000=READ(12,1030) ( DATA(I),I=1,KCNT )
16.000=1030 FORMAT( 12E11.4 )
17.000=C
18.000=OPEN ( UNIT=15, FILE='TMSE1', STATUS='NEW' )
19.000=WRITE(15,1000) IOPT1
20.000=WRITE(15,1010) IOPT2, XOPT
21.000=WRITE(15,1020) IOPT3, XOPT2
22.000=WRITE(15,1030) ( DATA(I),I=1,KCNT )
23.000=C
24.000=CLOSE ( UNIT=15 )
25.000=STOP
26.000=END
27.000=$END
Figure 5
Sample of QED

+++QED GETIM

*4F3*

4.000= INTEGER IOPT1(29), IOPT2(3), IOPT3(4)
5.000=C
6.000= OPEN ( UNIT=12, FILE='DOR01T', STATUS='OLD' )
#4F9

4.000= INTEGER IOPT1(29), IOPT2(3), IOPT3(4)
5.000=C
6.000= OPEN ( UNIT=12, FILE='DOR01T', STATUS='OLD' )
7.000= READ(12,1000) IOPT1
8.000= 1000 FORMAT( BI2, 2I5 )
9.000= READ(12,1010) IOPT2, XOPT
$10.1I$

10.010=C THIS IS AN EXAMPLE OF HOW TO USE QED
10.020=C IT IS PRETTY SIMPLE-MINDED.
10.030= $9F5$

9.000= READ(12,1010) IOPT2, XOPT
10.000= 1010 FORMAT( I2,2I8,2F7.4,F6.5,2F5.4,F11.4,F8.4,F12.5 )
10.010=C THIS IS AN EXAMPLE OF HOW TO USE QED
10.020=C IT IS PRETTY SIMPLE-MINDED.
11.000=C

$10D3$

$9F4$

9.000= READ(12,1010) IOPT2, XOPT
11.000=C
12.000= READ(12,1020) IOPT3, XOPT2
13.000= 1020 FORMAT( 4I10, 3F12.5 )

$5$
Figure 6
Sample of DISPLAY, without record numbers

+++ DISPLAY GETIM,1,999,SUF
PROGRAM GETIM ( TAPE12, TAPE15 )
C
HALF PRECISION XOPT(15), XOPI2(2), DATA(2000)
INTEGER IOPT1(29), IOPT2(3), IOPT3(4)
C
OPEN ( UNIT=12, FILE='DOR01T', STATUS='OLD' )
READ(12,1000) IOPT1
1000 FORMAT( BI2, 21I5 )
READ(12,1010) IOPT2, XOOP
C
READ(12,1020) IOPT3, XOPTO
1020 FORMAT( 4I10, 3F12.5 )
KCNT = IOPT3(3)
READ(12,1030) ( DATA(I),I=1,KCNT )
1030 FORMAT( 12E11.4 )
C
OPEN ( UNIT=15, FILE='TMSER', STATUS='NEW' )
WRITE(15,1000) IOPT1
WRITE(15,1010) IOPT2, XOPT
WRITE(15,1020) IOPT3, XOPTO
WRITE(15,1030) ( DATA(I),I=1,KCNT )
C
CLOSE ( UNIT=15 )
STOP
END
$E0R
Steps for running FINDIF on Cyber 205

1. Create a model parameter file, which can be free field format, except for the first two records. The contents of the records:

   Record 1: LABEL, up to 80 characters

   Record 2: Model identifier, 5 characters, must be in first 5 characters

   Record 3: values of IRECT, IDENS, IFLAT, IKELLY, IEXPL, IVERT, ITRAN, ISORB, ISNST

   Record 4: values of MM, NN, KK, KSTRT, DELT, DELR, DELZ

   Record 5: values of KOUTST, KOUTEN, MOUTST, MOUTEN, NOUTST, NOUTEN

   Record 6: values of KINC, MINC, NINC, KMARK, KMINC

   Record 7: values of VPl, VS1, RO1, VP2, VS2, RO2, NA, NB, VPT, VST, ROT

   Record 8: values of ND, MSW, NSW

   Record 9: values of NSORCE, PLSWID, TSWAVE

Figure 7 is a sample model parameter file. Record numbers are not part of the file.

NOTE: In the current version the number of timesteps (KK) must not exceed 8192 and the total number of snapshots INT(((KK-KMARK)/KMINC)+1.) must not exceed 26.

+++DISPLAY CRS01

1.000= CRS01: ROSE SCATTERING - GRADIENT
2.000=CRS01
3.000=1, 1, 1, 1, 1, 0, 0, 1, 2
4.000=999, 300, 4000, 1000, 0.00125, 0.01, 0.01
5.000=1500, 4000, 1, 999, 48, 148
6.000=2, 50, 50, 1600, 320
7.000=1.5, 0.0, 1.0, 8.43, 3.71, 2.688, 73, 224, 4.3, 2.48, 1.88
8.000=-273, 0, 0
9.000=13, 657, 0, 0
10.000=#EDR

Figure 7
2. Be sure FDPREP is on the system, by entering the command

GET FDPREP

If the system responds by saying the FDPREP does not exist, then do

LOAD FDPREP

You must wait until the system says that FDPREP is loaded before proceeding.

3. Make DOPREP a local file, by entering the command

GET DOPREP

You will probably have to edit this file. Figure 8 is a copy of the current file:

```plaintext
1.000=60118,BJHL1500.
2.000=PFILIES(GET,FDPREP)
3.000=PFILIES(GET,CRS01)
4.000=M77,I=FDPREP,-GO.
5.000=LGO,CRS01.
6.000=PFILIES,PUT,CRS01N.
7.000=PFILIES,PUT,CRS01L1.
8.000=PFILIES,PUT,FINUPD.
9.000=PFILIES,PUT,SORUPD.
10.000=PFILIES,PUT,BNYUPD.
11.000=PFILIES,PUT,JOBLF1.
12.000=PFILIES,PUT,JOBLF2.
13.000=#EOR
14.000=CRLOOF
15.000=BHLOOF
16.000=2035
17.000=MHD
17.020=AIIO54
19.000=#EOR
```

Figure 8

You must change all occurrences of CRS01 to the identification of the model to be run. In addition, the last five records should be:

a. Name of file containing boundary loop code

b. Name of file containing time-step loop code

c. Slot where output tape is stored, 4 digits

d. Visual Protect Key of tape (VPK), 3 letters

e. Tape label, 6 characters

PUT the new file.
4. Now you can run FDPREP. This usually runs very quickly, so you might as well wait for it. FDPREP produces the following files:

- **BNYUPD** updated code for FDBNY
- **SORUPD** updated code for FDSORS
- **FINUPD** updated code for FINDIF

New copy of model parameter file. The file name is the model name followed by N.

Log file - name is model name followed by L.

Job file to run FDBNY and FDSORS. This file will need to be edited if you do not want to run both programs (see Step 5). File name is JOBFL1.

Job file to run FINDIF. File name is JOBFL2.

5. Figure 9 is a sample of JOBFL1. If you want to run only FDBNY, delete records 9, 10, and 11. To run only FDSORS, delete records 6, 7, and 8.

```
1.000=6011B,BJH,CY.
2.000=RESOURCE,JCAT=S4,TL=3600.
3.000=MFLINK,CRS01P,ST=LM3,DU=C6,JCS=**
4.000= *PPFILES(GET,CRS01N)*.
5.000=ATTACH,BJLIB.
6.000=MFLINK,BNYPROC,ST=LM3,DU=C6,JCS=*
7.000= *PPFILES(GET,BNYPROC)*.
8.000=BEGIN,BNYPROC,1023,CRS01B,902,CRS01L2,CRS01F.
9.000=MFLINK,SORPROC,ST=LM3,DU=C6,JCS=**
10.000= *PPFILES(GET,SORPROC)*.
11.000=BEGIN,SORPROC,1299,CRS01S,465,CRS01L3,CRS01F.
12.000=*EOR
```

Figure 9
Figure 10 is file UPDBLIB. In record 4, change CRLOOP to the name of the file containing the variable loop code to be used in FDBNY. Now you can incorporate the update code into FDBNY and FDSORS, by submitting jobs UPDBLIB and UPDSLIB.

+++DISPLAY UPDBLIB
1.000=6011B,BJH,
2.000=PFILES(GET,BNYLIB)
3.000=PFILES(GET,BNYUPD)
4.000=PFILES(GET,CRLOOP)
5.000=UPDATE(I=BNYUPD,F=BNYLIB,C=BNYFOR)
6.000=PFILES(PUT,BNYFOR)
7.000=#EDR

Figure 10

6. Now you are ready to submit JOBFL1, to run FDBNY and FDSORS. However, be sure you have at least 3000 words of local storage available, to hold the two log files.

7. When JOBFL1 has been completed, check to be sure it ran correctly. Look at the DAYFILE. Look at the two log files (CRSO1L2 and CRSO1L3, for example). You can use the hard-copy terminal at IPC to get listings of these files. You should then DELETE BNYFOR, SORFOR and the log files.

8. You can now prepare to run FINDIF. Figure 11 is a list of file UPDFLIB. Due to disk space limitations, FINDIF is stored as a basic source file. Before running UPDFLIB, you must run MAKFLIB. Then after you run UPDFLIB, you must delete FINLIB.

+++DISPLAY UPDFLIB
1.000=6011B,BJH,
2.000=PFILES(GET,FINLIB)
3.000=PFILES(GET,FINUPD)
4.000=PFILES(GET,BHLOOP)
5.000=UPDATE(I=FINUPD,F=FINLIB,C=FINFOR)
6.000=PFILES(PUT,FINFOR)
7.000=#EDR

Figure 11

In record 4, change BHLOOP (if necessary) to the name of the file containing the variable loop code to be used in TSTEFL. BHLOOP contains the Bhasavanija (1983) code. STLOOP contains Stephen (1983) code. Then you can submit UPDFLIB to create the COMPILE file for FINDIF.
9. You will have to edit JOBFL2 if you are using a source or boundary file from a different model. For example, if you want to run model CRS02, and use the source file from CRS01. Figure 12 contains a list of JOBFL2. After record 13, insert the command

ATTACH CRS01S
SWITCH, CRS01, CRS02

This changes the name of file CRS01S to CRS02S, which is the file that FINDIF will be looking for.

10. Be sure you have plenty of file space available on the Front-End. You will need about 3000 words for the FINDIF log file. You can then submit JOBFL2. This will probably not run until evening, so you should ROUTE the job to WAIT.

11. Good Luck!

12. Send mail to Bill Whitson (AEX) to send tape that is specified slot # (line 30 in JOBFL2).

     MAIL, FILE = MSGS, TO = AEX

     Bill Whitson's phone number is (317)494-1787.

13. For accounting:

     PFILES LIST CHARGES

14. To read the Cyber tapes on the VAX 11-780 (WMS) use programs RDSNAP and RDTMSE. These move the snap-shot and time-series files from tape to disk. These programs are in [FIND.MMH1.VECI on the WHOI Blue VAX. Remember you must enclose file names on the tape in quotes:

     MFA0: "CRS01T"
Figure 12

Job-file to run FINDIF

1.000=6011B, BJOH, CY.
2.000=RESOURCE, NT=1, JCAT=S4, TL=3600.
3.000=ATTACH, DBJLIB.
4.000=REQUEST, CRS01T/210.
5.000=REQUEST, SNF1600/1648.
6.000=REQUEST, SNF1920/1648.
7.000=REQUEST, SNF2240/1648.
8.000=REQUEST, SNF2560/1648.
9.000=REQUEST, SNF2880/1648.
10.000=REQUEST, SNF3200/1648.
11.000=REQUEST, SNF3520/1648.
12.000=REQUEST, SNF3840/1648.
13.000=ATTACH, CRS01S.
14.000=ATTACH, CRS01B.
15.000=MFLINK, CRS01P, ST=LM3, DD=C6, JCS=",**
16.000= "PFILES(GET, CRS01N)"
17.000=MFLINK, FINPROC, ST=LM3, DD=C6, JCS=",**
18.000= "PFILES(GET, FINPROC)"
19.000=BEGIN, FINPROC, FINPROC, 5025, CRS01L4, CRS01P.
20.000=DEFINE, CRS01T.
21.000=DEFINE, SNF1600.
22.000=DEFINE, SNF1920.
23.000=DEFINE, SNF2240.
24.000=DEFINE, SNF2560.
25.000=DEFINE, SNF2880.
26.000=DEFINE, SNF3200.
27.000=DEFINE, SNF3520.
28.000=DEFINE, SNF3840.
29.000=REQUEST, TAPE, B=6250, ACCESS=W, TF=Y, RT=F, BT=K,
30.000= RLMAX=132, RB=40, SLOT=2035/MHD/A11054.
31.000=LABEL, A, MFn=TAPE, FID=CRS01T, LPROC=W, RT=F, BT=K,
32.000= RLMAX=132, RB=40, FSN=1.
33.000=LABEL, B, MFn=TAPE, FID=SNF1600, LPROC=W, RT=F, BT=K,
34.000= RLMAX=132, RB=40, FSN=9999.
35.000=LABEL, C, MFn=TAPE, FID=SNF1920, LPROC=W, RT=F, BT=K,
36.000= RLMAX=132, RB=40, FSN=9999.
37.000=LABEL, D, MFn=TAPE, FID=SNF2240, LPROC=W, RT=F, BT=K,
38.000= RLMAX=132, RB=40, FSN=9999.
39.000=LABEL, E, MFn=TAPE, FID=SNF2560, LPROC=W, RT=F, BT=K,
40.000= RLMAX=132, RB=40, FSN=9999.
41.000=LABEL, F, MFn=TAPE, FID=SNF2880, LPROC=W, RT=F, BT=K,
42.000= RLMAX=132, RB=40, FSN=9999.
43.000=LABEL, G, MFn=TAPE, FID=SNF3200, LPROC=W, RT=F, BT=K,
44.000= RLMAX=132, RB=40, FSN=9999.
45.000=LABEL, H, MFn=TAPE, FID=SNF3520, LPROC=W, RT=F, BT=K,
46.000= RLMAX=132, RB=40, FSN=9999.
47.000=LABEL, I, MFn=TAPE, FID=SNF3840, LPROC=W, RT=F, BT=K,
48.000= RLMAX=132, RB=40, FSN=9999.
B. FINDIF on the CRAY XMP-12

(Based on Notes by Mary M. Hunt, June 1986)

I. Introduction

This document describes briefly the CRAY XMP-12 super-computer system at NRL in Washington and the implementation of Ralph Stephen's Finite Difference software on the system. The CRAY system includes a cluster of three VAX-11/785's, running under VMS, which serves as a Front-End.

All editing and other preparation are done on the VAX, and jobs are submitted to the CRAY for execution. The use of the VAX for the Front-End saves lots of problems, since we are used to the VAX.

I had previously implemented this same software package on a Cyber 205 at Purdue. This document includes the differences between the two systems.

II. General System Description

How to Log-on

All work to date has been done by direct telephone line to the computer - no networks. Use the DIAL class on the PACX.

A. Direct call from a VT100

1. Set terminal to 1200 Baud.

2. Access the DIAL class on PACX and enter #1 K K85 K 1 202 767 2000

3. Follow instructions in CCF (NRL-Central Computer Facility) guide, page 1-4 to log-in (Appendix II). A sample is given in Figure 1.

B. Direct call from DECwriter

1. Set terminal to 1200 Baud, and parity to code 2:

   a. Strike CTRL SET-UP
   b. Strike 0 (zero) key until display shows 1200
   c. Strike P key until display shows 2
   d. Strike SET-UP again

2. Then log-in as above.
enter class DIAL
class dial start
HELLO: I'M READY
+
NUMBER?
$1K K85K12027672000

DIALING...
RINGING...
ANSWER TONE
ONLINE

Digital Ethernet Terminal Server V2.0.1 - LAT V5.0

Enter username> STEPHEN
Local> CONNECT N

Username: STEPHEN
Password:

Last interactive login on Friday, 9-MAY-1986 11:04
Last non-interactive login on Monday, 3-MAR-1986 16:49

Visit 1/785
NRL3
Figure 2
Sample Log-on To Save The Session

-24-

Username: IPC-MMH1
Password:

WHOI RED system

Last interactive login on Thursday, 29-MAY-1986 14:37
Last non-interactive login on Thursday, 29-MAY-1986 15:07

== MESSAGE CHANGED 2-JUN-1986 07:51:07.00 BY ACC-DRCI

SPAK UNAVAILABLE: The SPAK-disk will be unavailable this morning, Monday, June 2, 1986.

QUEUE AVAILABILITY CHANGED: Low cost batch queues are now open more hours.
Please read new IPC Newsletter (#84) for details. Thanks. (14 May 2-JUN-1986 08:16:28)

$ ALLO PACX1
%DCL-I-ALLOC, _TXEO: allocated
$ SET TERM/SPEED=1200/MODEM PACX1
$ SET HOST/LOG/DTE PACX1

REM-I-TOEXIT, connection established, type "\" to exit

enter class DIAL
class dial start

HELLO:I'M READY
$D NUMBERT
$1K K85K12027672000

DIALING...
KINGING...
ANSWER TONE
ONLINE

Digital Ethernet Terminal Server V2.0.1 - LAT V5.0

Enter username: STEPHEN
Local: CONNECT N

Username: STEPHEN
Password: -
C. To get your entire session copied to a file on the VAX:

1. Log-on to RED, using a CRT terminal.

2. $ALLOCATE PACXL
   $SET TERM/SPEED=1200/MODEM PACX1
   $SET HOST/LOG/DTE PACX1

3. You will then get the 'enter class' prompt. Enter DIAL, and proceed as above. The session will be stored in a file called SETHOST.LOG. See sample in Figure 2.

D. If, during a session, you suddenly get the message LOCAL, do not panic. You can get reconnected by typing RESUME.

Inter-computer Communication

Communication between the VAX and the CRAY is very easy. On the VAX end, there are a set of 'station' commands for CRAY communication. There are only a few you will need:

1. To submit a job to the CRAY

   $CRAY SUBMIT file-spec

   where 'filespec' is the name of a file containing the desired CRAY JCL.

2. To see if the job has completed

   $CRAY STATUS/OWNER

   Any uncompleted jobs you have submitted to the CRAY will be listed.

3. To watch the progress of a CRAY job, you can issue the VAX command

   $CRAY SET TERMINAL INFORM

   before submitting a CRAY job. This will result in a message whenever a file transfer between CRAY and VAX (in either direction) occurs.

These are the only commands I have used.
On the CRAY end, the command to access a file from the VAX is

`FETCH,DN=cfile,TEXT='vfile'`

where

cfile is the CRAY local dataset name
vfile is the VAX file spec

To send a file from the CRAY to the VAX

`DISPOSE,DN=cfile,TEXT='vfile'`

**Some CRAY Control Commands**

Every CRAY control command must end with a period.

1. **JOB** - Every CRAY job must start with a JOB command. The form of this command is `JOB,JN=jobname,T=time,MFL=size,*TAPE=ntapes.`

   - **jobname** is the job-name. The output will be returned to the VAX in a file called `jobname.CPR`
   - **time** is the maximum job CPU time, in seconds. If not specified, the limit is 8 seconds.
   - **size** is the maximum memory used, in words. The default is 135000. The maximum available is 1,536,000.
   - **ntapes** is the number of magnetic tape drives required by the job. If no tapes are required, the *TAPE should be omitted.

   The minimum job card would be `JOB,JN=job-name`.

2. **ACCOUNT** - The second command must be ACCOUNT:

   `ACCOUNT,AC=acno,US=usno,UPW=pswd`

   - **acno** is account number
   - **usno** is user name
   - **pswd** is user password
3. **ACCESS** - In order for a job to use a permanent dataset (file), you must include an ACCESS command to make the dataset local:

\[
\text{ACCESS, DN=dname.}
\]

\(dname\) is the permanent and local name of the dataset

4. **ASSIGN** - This command is used to connect a Fortran logical unit to a file. It is not required if the program does an explicit OPEN including the local dataset name:

\[
\text{ASSIGN, DN=dname, A=FTnn.}
\]

\(dname\) is the local dataset name
\(nn\) is the Fortran unit number

There are many other parameters which can be included on an ASSIGN command. For the most part, they are inscrutable.

5. **DELETE** - To delete a file, you must first make it local to the job, with the unique access:

\[
\text{ACCESS, DN=dname, UQ.}
\]

Then you can delete the file:

\[
\text{DELETE, DN=dname.}
\]

Adding NA to either of these commands will prevent the job from being aborted if an error occurs.

**Tapes**

Tape usage is very awkward and poorly documented. They have written a Tape Guide for the CRAY, but it is not much better than the CRAY manual, and in some places is incorrect. There appear to be three possible ways to create a tape.

1. Create an ANS labeled tape directly on the CRAY. The Control command options appear to allow this. However, it cannot be done. The consultants had told me this, but I didn't believe it until I tried. You cannot create a labeled tape on the CRAY which can be read by the VAX.
2. Use the *TAPE option on the DISPOSE command which allows you to send a CRAY file directly to a VAX tape. This does not work for multiple files, and does not include the capability to tell the operator to use a write ring. You must send him (or her) a message somehow to let him (or her) know you need a write ring.

3. DISPOSE all the files from the CRAY to the VAX, and then copy them interactively, from the VAX to tape. When running large models, there might not be room in our disk allocation (10000 blocks) for all the output files. One of the consultants suggested DISPOSing and copying the files one at a time. This solution also means that you must log-on again after the CRAY job has completed, perhaps wait for a tape drive, copy the files, and send the operator a message to mail the tape.

I think the second solution is the least objectionable. It does mean we must combine all the output into one file, and make an arrangement with the operators always to use a write ring when mounting a tape for STEPHEN. We then, when the tape has been created, can make arrangements to mail it back here.

In addition to the difficulties mentioned above, there are two other anomalies in the tape usage:

1. There does not appear to be any way to run a batch job on the VAX using tapes. They do not appear to have the equivalent of our tape queues and GETDEV.

2. There is no way for the system to distinguish between an internal tape label and an external tape label, except for interactive usage on the VAX. Otherwise, the system assumes that the numeric external label, assigned by the NRL staff, is the same as the internal ANS label.

Comparison to Cyber

The CRAY and CYBER computers seem to me to differ very little in ease (or difficulty) of use. The systems as a whole, however, differ considerably.

In some ways, the NRL CRAY system is easier to use than the CYBER system at PURDUE. For one thing, the VAX Front-End is much easier to use than the CYBER 6600 at Purdue. Another advantage is that the log output (DAYFILE on CYBER) is automatically returned to the Front-End as soon as the job is completed.

Disadvantages of the CRAY system are relatively poor and badly documented tape handling. Theoretically the tape handling looks good, but it does not appear to work as advertised. Also, there is no way the people at NRL could match the superb support and help we got from the Purdue staff, especially Bill Whitson. The big problem at NRL is that you might get a different consultant each time you call, and they don't always agree.
The CRAY presents severe limitations on the size of model that can be run, as compared to the Cyber at Purdue.

1. The CRAY has only 2 million words of memory, the CYBER has 4 million.
2. The CRAY has no HALF PRECISION data type, so all arrays use the full 64 bits.
3. There is no virtual memory on the CRAY, and the extended memory option is not implemented on the NRL machine.

Other differences are that UPDATE cannot be run on the Front-End, as we did on the CYBER, and that the CRAY allows multiple versions (they call them editions) of the same file. I am not sure if these items make much difference in the ease of operation; they are just different.

The CRAY documentation at first looks good. When you try to use it however, it is confusing, incomplete, and sometimes contradictory. For example, see the description of ASSIGN (page 8-1) and ACCESS (page 9-5) in the CRAY-OS Reference Manual. There does not appear to be any documentation about the vectorization process. They teach a class about it, but there is no documentation.

III. Implementation of FINDIF Software

Programs

The package includes four programs. The first, Program FDPREP, runs on the VAX Front-End, and creates files needed to run the other programs on the CRAY. The only input to FDPREP is a model parameter file, describing the model and the output desired.

The next two programs, FDBNY and FDSORS, run on the CRAY, and each creates a file which is required by the final program, FINDIF.

In addition to the programs, there is another file of source code, containing subprograms used on the CRAY.

Implementing the INCLUDE

The procedure required to include COMMON files into a source deck, using UPDATE, is a two-step process. First, your basic source file is subdivided into COMDECK's and DECK's, as shown in Figure 3. COMMON blocks which do not change are completely specified, but variable blocks remain undefined. The first step is to incorporate the COMDECK's into the DECK's, to create what they call a Program Library. There is a job-file to do this step for each program, as shown in Table 1. These create, on the CRAY, the program library files BNYLIB, SORLIB, and FINLIB.
Figure 3
Form of Source File

```
*COMMON MODPAR
   COMMON /MODPAR/ TRECT, IDENS, IFLAT, IKel
   COMMON /MODPAR/ IEXPL, IVERT, ITRAN, ISORB, ISNST
   etc.
*COMMON IOUNIT
   COMMON /IOUNIT/ LULDG, LUINP, LUBNY, LUSRC
   COMMON /IOUNIT/ LUSNP, LUTST
*COMMON COMEDA
*DECK MODPAR
*DECK TSTEPL
SUBROUTINE TSTEPL
```

```
*DECK ABSSR
SUBROUTINE ABSSR
```

```
*CALL MODPAR
*CALL IOUNIT
*CALL COMEDA
```

```
END
```

```
*DECK TSTEPL
SUBROUTINE TSTEPL
```

```
*CALL MODPAR
*CALL COMEDA
```

```
END
```
### Table 1

Files used for UPDATE procedures

<table>
<thead>
<tr>
<th>Program</th>
<th>Basic Source File (VAX)</th>
<th>Job To Create Program File (VAX)</th>
<th>Program Library Library (VAX)</th>
<th>File Containing Update Code (VAX)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FDBNY</td>
<td>FDBNY.DCK</td>
<td></td>
<td>MAKLIB.JOB</td>
<td>BNYLIB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(CRAY)</td>
<td>BNYUPD.DCK</td>
</tr>
<tr>
<td>FDSORS</td>
<td>FDSORS.DCK</td>
<td></td>
<td>MAKSLIB.JOB</td>
<td>SORLIB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(VAX)</td>
<td>SORUPD.DCK</td>
</tr>
<tr>
<td>FINDIF</td>
<td>FINDIF.DCK</td>
<td></td>
<td>MAKFLIB.JOB</td>
<td>FINLIB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(VAX)</td>
<td>FINUPD.DCK</td>
</tr>
<tr>
<td>subroutines</td>
<td>LIBSUB.DCK</td>
<td></td>
<td>MAKLIB.JOB</td>
<td>OBJLIB</td>
</tr>
</tbody>
</table>

These files are already created, so this first step does not need to be done unless changes in the source code are required.

Program FDPREP creates COMMON files for each of the three other programs. The UPDATE procedure to incorporate these files into the program libraries is part of the job files to run the programs, so needs no separate job.

**Vectorization**

When this package was implemented on the CYBER, I used explicit vectorization commands. Such commands do not exist (or are not documented) on the CRAY, so each of these one-line commands had to be rewritten as a DO-LOOP. This tedious operation was done by Lee Gove before I became involved with the CRAY.
Permanent files on the VAX

1. File for FDPREP
   - FDPREP.FOR  source
   - FDPREP.OBJ  object
   - FDPREP.EXE  executable

2. Source files for CRAY
   - LIBSUBS.DCK  subprograms
   - FDBNY.DCK   source for FDBNY
   - FDSORS.DCK  source for FDSORS
   - FINDIF.DCK  source for FINDIF

3. Permanent job files
   - MAKLIB.JOB  creates subroutine object library on CRAY
   - MAKBLIB.JOB
   - MAKSLIB.JOB create UPDATE program libraries on CRAY
   - MAKFLIB.JOB

4. Procedure files
   - BNYPROC.PRC These files contain the
     constant part of the procedures
   - SORPROC.PRC to run each program
   - FINPROC.PRC

Permanent Files on the CRAY

Any other files are superfluous, and should be deleted.

1. Program library files
   - OBJLIB  subprogram object library
   - BNYLIB
   - SORLIB  Program library files from UPDATE
   - FINLIB

2. Files required during a model (say BNCH1)
   - BNCH1LB  boundary file
   - BNCH1LS  source file
IV. Steps to Run a Model

1. Create a model parameter file, which can be free field format, except for the first two records. The contents of the records:

   Record 1: LABEL, up to 80 characters

   Record 2: Model identifier, 5 characters, much be in first 5 characters

   Record 3: values of IRECT, IDENS, IFLAT, IKELLY, IEXPL, IVERT, ITRAN, ISORB, ISNST

   Record 4: values of MM, NN, KK, KSTRT, DELT, DELR, DELZ

   Record 5: values of KOUTST, KOUTEN, MOUTST, MOUTEN, NOUTST, NOUTEN

   Record 6: values of KINC, MINC, MINC, KMARK, KMNC

   Record 7: values of VP1, VS1, R01, VP2, VS2, R02, NA, NB, VPT, VST, ROT

   Record 8: values of ND, MSW, NSW

   Record 9: values of NSORCE, PLSWID, TSWAVE

Below is a sample model parameter file.

CRS01: ROSE SCATTERING - GRADIENT

CRS01
1, 1, 1, 1, 1, 0, 0, 1, 2
999, 300, 4000, 1000, 0.00125, 0.01, 0.01
1500, 4000, 1, 999, 48, 148
2, 50, 50, 1600, 320
1.5, 0.0, 1.0, 6.43, 3.71, 2.688, 73, 224, 4.3, 2.48, 1.88
-273, 0, 0
13, 657, , 0.0
2. ASSIGN the model parameter file to FOR055, and run FDPREP:

\$ASSIGN fspec FOR055
\$RUN FDPREP

The program asks for the tape number (6 digits) for the final output tape containing snap-shot files and time series file.

FDPREP creates the following files:

1. BNYUPD.DCK  
   SORUPD.DCK  
   FINUPD.DCK  
   These files contain update code  
   for COMMON for the three programs

2. JOBFL1.JOB  
   Job to run FDBNY and FDSORS
   JOBFL2.JOB  
   Job to run FINDIF

3. File containing log output. Name is FILEID and .LG1.  
   For example, if FILEID is BNCH1, filename is BNCH1.LG1.

   Name is FILEID and P.DAT. If FILEID is BNCH1, filename is BNCH1P.DAT

3. If any of the three programs needs editing, do the edits, and create a  
   new program library on the CRAY, by submitting

   MAKBLIB  for  FDBNY
   MAKSLIB  for  FDSORS
   MAKFLIB  for  FINDIF

4. Below is the job (JOBFL1.JOB) to run FDBNY and FDSORS.

```plaintext
JOB,JN=RNBNSR,T=3600,MFL-149878.
ACCOUNT,AC-28X127X6,US-STEPHEN,UFW-
   * ACCESS,DN=BNCH1B,UQ,NA.
   * DELETE,DN=BNCH1B,NA.
   * FETCH,DN=BNCH1P,TEXT=’BNCH1P.DAT’.
   * ASSIGN,DN=BNCH1P,A-FT55.
   * FETCH,DN=BNYFPROC,TEXT=’BNYFPROC,FRC’.
   * ACCESS,DN=OBJLIB.
   * CALL,DN=BNYFPROC.
   * DISPOSE,DN=BNCH1B,TEXT=’BNCH1B.LST’.
   * SAVE,DN=BNCH1B.
   * ACCESS,DN=BNCH1S,UQ,NA.
   * DELETE,DN=BNCH1S,NA.
   * FETCH,DN=BNCH1Q,TEXT=’BNCH1Q,DAT’.
   * ASSIGN,DN=BNCH1Q,A-FT55.
   * FETCH,DN=SORPROC,TEXT=’SORPROC,FRC’.
   * CALL,DN=SORPROC.
   * DISPOSE,DN=BNCH1S,TEXT=’BNCH1S.LST’.
   * SAVE,DN=BNCH1S.
```
To run only FDSORS, delete the records marked with *. To run only FDBNY, delete the records marked with 0. Then submit the job.

When the job has completed, make sure it has run correctly, by typing the output file, RNBNSR.CPR. Also the two log files, BNCH1LB.LST (from FDBNY) and BNCH2LS.LST (from FDSORS).

Files BNYPROC and SORPROC are given below.

```
FETCH, DN=BNYUPD, TEXT='BNYUPD.DCK'.
ACCESS, DN=BNYLIB.
UPDATE, P=BNYLIB, I=BNYUPD, N=0, C=BNYFOR, L=0, F.
CFT, I=BNYFOR, L=0, B=BNYBIN, OFF=V.
LDR, DN=BNYBIN, LIB=OBJLIB.

FETCH, DN=SORUPD, TEXT='SORUPD.DCK'.
ACCESS, DN=SORLIB.
UPDATE, P=SORLIB, I=SORUPD, N=0, C=SORFOR, L=0, F.
CFT, I=SORFOR, L=0, B=SORBIN, OFF=V.
LDR, DN=SORBIN, LIB=OBJLIB.
```

5. Below is the job (JOBFL2.JOB) to run FINDIF.

```
JNB, JN=BNCH1L, T=600, MCL=0150419.
ACCOUNT, AC=2AV1274A, US=STEPHENH.

ACCESS, DN=BNCH1L, TEXT='BNCH1UPD.DCK'.
UPDATE, P=BNCH1L, I=BNCH1UPD, N=0, C=BNCH1FOR, L=0, F.
CFT, I=BNCH1FOR, L=0, B=BNCH1BIN, OFF=V.
LDR, DN=BNCH1BIN, LIB=OBJLIB.
```

If you want to use a source or boundary file from a different model, you will have to edit this file. For example, in the job above, if you want to use the boundary file produced from model BNCH2, you would change the record with the arrow to

```
ACCESS, DN=BNCH1L, PDN=BNCH2B.
```

This gives the local name BNCH1L to the permanent file BNCH2B.

You can then submit this job to the CRAY:

```
$CRAY SUBMIT JOBFL2.JOB
```

The output file will be returned to the VAX Front-End as RUNFIN.CPR, and the log file will be BNCH1LB.LST. If they look correct, you can MAIL a message to the operator to send the tape.
C. ACKNOWLEDGEMENTS

We would like to thank Bill Whitson of the Purdue University Computing Center for his patient support of our work. The Cyber 205 portion of this research was funded by the National Science Foundation under Grants OCE-8117571 and OCE-8409155. The Cray-XMP development was funded by the Office of Naval Research under Contract #N00014-85-C-0001NR.

D. REFERENCES


I spent a week during August 1984 working at the Purdue University Computer Center (PUCC), to learn how to use their Operating System, and to try to get the FINDIF software operational on their Cyber 205 Super Computer. I was doing this work for Dr. Ralph Stephen of the Woods Hole Oceanographic Institution, who was a recipient of an NSF grant to use a Super Computer. I had previously had a small amount of experience using a CYBER 205. Since then I have spent about two months working on the project.

The FINDIF software consists of four programs, originally written by Dr. Stephen for the VAX 11/780 at W.H.O.I. These programs have all been modified by me, primarily to make them more transportable. Hunt, Gove and Stephen (1983) describes the programs in detail. The first program determines array sizes to be used by the other three programs, and checks parameter values for consistency and physical compatibility. This program, which runs on the Front-End system at Purdue, has been substantially modified, and now creates job files to run the other programs. The second and third programs create files on the 205 which are required by the final program in the series. Modifications to these programs has been minimal. Sections of the final program have been completely rewritten in vector code, in order to make the greatest possible use of the vector processor on the Cyber 205.

Some of the problems encountered have been:

1. The method employed by the MACE operating system to incorporate code into a source file is cumbersome. This is partly because the code is stored on the Front End, and compiled on the 205. This feature is used for COMMON specifications, and for some sections of code which change with different applications.

2. The operating system in general is very unforgiving; there are many things you must remember in order to get a model to run successfully.

3. The amount of disk storage allowed on the Front End is rather limited.

I hope these problems have been either circumvented or at least well enough documented so they will not cause problems in the future.

Some of the things I particularly like in the Purdue Computer Center Operating systems are their Dump/Load capability, which makes it very easy to back up and restore files on the Front End; the post-mortem dump on the Cyber 205, which can be a big help in determining exactly when and why a program aborts; and their system of documentation, which appears to be flexible and thorough.
Bill Whitson, who was given the task of advising NSF grant recipients, has been extremely helpful. He has responded quickly to questions via computer mail, initialized tapes, mailed tapes back to Woods Hole, and in general done everything possible to expedite things. The staff has also been responsive to the special requirements of long-distance researchers. Finally, when I encountered a bug in the Fortran run-time library, it was fixed very quickly.

We have so far succeeded in running a few small models for test purposes, and two large models. The times are incredibly fast; a model which took 7 hours 40 minutes on the VAX ran in 3 minutes on the Cyber 205. Now that our package has been debugged and documented, we should be able to process additional models with no more problems.

Appendix II

NRL-Central Computer Facility Log-on Procedures

CHAPTER 1

ORIENTATION

1.1 GENERAL INFORMATION

1.1.1 Introduction To The CCF

NRL's Research Computation Division (RCD) operates the Laboratory's Central Computer Facility (CCF) for the purpose of scientific research and development. CCF services are available to NRL employees and are conditionally available to the Department of Defense and other government agencies and their contractors.

This system consists of a Cray X-MP/12 computer front-ended by a cluster of three VAX 11/785s. The Cray is a powerful, general purpose machine that combines a single processing unit with 2 million 64-bit words of memory. It has an I/O subsystem with three interconnected I/O processors. The Cray X-MP/12 achieves extremely high processing rates by efficiently using the scalar and vector capabilities of the CPU combined with the system's random-access solid-state memory (RAM) and registers. Sustainable speeds of 50 to over 100 million floating point operations (MFLOPS) can be achieved by vectorizable Fortran codes.

The VAXcluster*, which has 8 megabytes of main memory per processor, supports electronic mail and file transfer. The front end provides communications via both terminal lines and local area networks. This configuration will allow many users to share the CCF concurrently. The front-end systems also provide such services as data base management systems, a document processor, and graphics support.

For an interim period, the RCD also operates a DECsystem-10 computer. There is a separate user guide for this computer, available in building 97, room 155.

* VAXcluster is a trademark of Digital Equipment Corporation
1.1.2 The Research Computation Division

The RCD has two branches: a User Support Branch and a Systems Support Branch. The User Support Branch provides CCF services including user consulting, problem solving, and training, as well as computer operations and local area network operations support. The Systems Support Branch provides for the software and engineering of the CCF, including acquiring or developing special system software for the CCF, the local NRL network (NICENET), and the host front-end to ARPANET/MILNET.

RESEARCH COMPUTATION DIVISION

<table>
<thead>
<tr>
<th>TITLE</th>
<th>CODE</th>
<th>NAME</th>
<th>PHONE area</th>
<th>Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIVISION HEAD</td>
<td>2800</td>
<td>Rudi F. Saenger</td>
<td>767-2751</td>
<td></td>
</tr>
<tr>
<td>DEPUTY DIVISION HEAD</td>
<td>2801</td>
<td>Doris E. Gossett</td>
<td>767-6750</td>
<td></td>
</tr>
<tr>
<td>ADMINISTRATIVE OFFICER</td>
<td>2802</td>
<td>Beulah M. Thomas</td>
<td>767-6750</td>
<td></td>
</tr>
<tr>
<td>COMPUTER RESOURCES BRANCH HEAD</td>
<td>2810</td>
<td>George E. Perez</td>
<td>767-3884</td>
<td></td>
</tr>
<tr>
<td>USER SERVICES SECTION HEAD(acting)</td>
<td>2811</td>
<td>George E. Perez</td>
<td>767-3884</td>
<td></td>
</tr>
<tr>
<td>COMPUTER SERVICES &amp; OPERATIONS SECTION HEAD</td>
<td>2812</td>
<td>Jay S. Oberfield</td>
<td>767-3903</td>
<td></td>
</tr>
<tr>
<td>SYSTEMS SUPPORT BRANCH HEAD</td>
<td>2850</td>
<td>Harvey K. Brock</td>
<td>767-3887</td>
<td></td>
</tr>
<tr>
<td>SOFTWARE SECTION HEAD</td>
<td>2851</td>
<td>Dale A. Pfaff</td>
<td>767-3190</td>
<td></td>
</tr>
<tr>
<td>COMMUNICATIONS &amp; ENGINEERING SECTION HEAD</td>
<td>2852</td>
<td>Rufus D. McCulloch</td>
<td>767-3903</td>
<td></td>
</tr>
</tbody>
</table>

Phone numbers to call for various services:
- CCF 300/1200 baud dial-in line .. 767-2000
- CCF 2400 baud dial-in line ...... 767-1240
- CCF status recording .......... 767-3512
- CCF consultant's desk .......... 767-3542, 767-1374
- CCF Customer Service Counter ... 767-3993
- NRL Internet coordinator ...... 767-3884
- NRL-NFE 300/1200 baud dial-in ... 767-4830
- DECsystem-10 dial-in ........... 767-5333, 767-3816
- DECsystem-10 operations ....... 767-3794
1.1.3 Opening An Account

NRL employees and their contractors who wish to establish an account on the CCF should report in person to the Customer Service Counter (CSC) in building A49. There an application form must be filled out which will allow you to use both the VAXcluster and the Cray. The RCD staff will then verify that the job order number is valid. Each account on the CCF will be issued the following identification:

1) username
2) password
3) account number (NRL JON)

We don't recommend more than one person per account. If several customers wish to work on a project together, on the front-end we can issue each an account using the same account number and group number (UIC) with different usernames and programmer numbers. Files can then be transferred conveniently between several users while the computer usage of each is accounted for separately.

Non-NRL employees should make initial inquiries about the system by calling the CCF Outside Users Coordinator in User Services, (Autovon)297-3884 or (202)767-3884. Navy installations obtain a CCF account by submitting a Work Request (Form 2275) to the NRL Budget Officer, Code 1310, NRL, Washington, D.C. 20375-5000. Department of Defense (non-Navy) installations submit a Military Interdepartmental Purchase Request (Form DD 448). Government agencies and private companies submit cash in advance to the NRL Budget Officer.

1.1.4 Accessing The CCF

The CCF can be accessed via dial-up phone lines, via the NRL local area network (NICENET), via network connections with certain NRL VAX computers outside the RCD, and via DDN/MILNET. There are 48 dial-up lines and 16 local area network connections into the CCF. NICENET and DDN/MILNET connections are currently being worked out. Detailed information on these will be added later.
ORIENTATION
GENERAL INFORMATION

1.1.5 Dialing The Computer

The CCF front end will receive and transmit data at several different speeds which are measured by the baudrate (bits per second). You will have to:

1) know what baud rates your modem will handle
2) set your terminal to the speed you desire (see setting terminal parameters)
3) dial 767-2000 for 300/1200 baud, or 767-1240 for 2400 baud

As soon as you make the connection, type several carriage returns so that the terminal server can determine your baud rate. You will have a wait of up to 30 seconds, then you will see a pound sign prompt from the terminal server for the password: 

#

Type in "N" and press the carriage return key, as usual the password will not echo. Next the terminal server will prompt you for your username:

Enter username>

Type in your CCF VAX username and press carriage return. You will then be prompted by the ethernet:

LOCAL>

Since the CCF front end is actually several nodes in a VAXcluster, you can tell the terminal server to either choose a node for you by typing immediately after the ethernet prompt:

CONNECT N

or select one of the three VAXcluster nodes by typing:

CONNECT NRL1
or
CONNECT NRL2
or
CONNECT NRL3

A CCF VAX will then begin the timesharing dialogue. It will prompt you with "Username: ", indicating that you're ready to login.
1.1.6 CCF Initial Log-on/Log-off Procedures

You will type your username and password to log into the VAXcluster. Your "username" is generally your last name. Your "password" was chosen by you when you opened your account. The first time you log in to the VAX you will be required to change your password (the system will prompt you).

To submit a job to the Cray, you will use (on the ACCOUNT statement) your username as the user number (US), your "original" VAX password as the user password (UPW) and your job order number (no hypens) as your account number (AC). You will be required to change your Cray user password, by adding the NUPW (new user password) parameter on the ACCOUNT statement, the first time you submit a Cray job.

For your first Cray job, use JOB and ACCOUNT statements like:

JOB, JN=anynname.
ACCOUNT, AC=acntnmbr, US=username, UPW=originalpassword,
NUPW=newpassword.

For subsequent jobs, UPW now equals the new password set by NUPW.

To log off the system, you will need to type "LOG" twice. The first "LOG" will log you off the VAX; then you will see the LOCAL> prompt. Be sure to type LOG again to log yourself off the terminal server.
END
2-87
DTIC