N92-30013

AN INTRODUCTION TO USING THE FORTRAN PROGRAMS PROVIDED WITH

COMPUTATIONAL NUCLEAR PHYSICS 1
NUCLEAR STRUCTURE
K. LANGANKE, J.A. MARUHN AND S.E. KOONIN (EDS.)
SPRINGER-VARLAG BERLIN GERMANY 1991

Matthew A. Boytos and John W. Norbury
Department of Physics
Rider College
Lawrenceville NJ 08648

INTRODUCTION

The authors of Computational Nuclear Physics have provided along with their text an excellent set of well-written, ready-to-run Fortran programs that should prove useful in many disciplines of theoretical nuclear physics.

The purpose of this document is to provide, simply, a synopsis of the programs and their use for those who wish to begin working on the computer immediately. We will attempt to provide some background on each program before going into the specific details of how to get the program running and make its results useful.

A separate section is devoted to each chapter (and program set) in the text. Within each section, there are five headings. A brief description of what will be found in each follows.

Abstract — A short summary of what the program(s) will do, and brief instructions on their use.
Files — A listing of the files provided by the authors and their content and use.
Compiling, linking and running — A comprehensive set of instructions giving the specifics on installing the code(s).
Obtaining results — A section of hints, notes and procedures to help users make effective use of the code(s).
Tutorial — A detailed, step-by-step procedure for installing the code and running an example calculation.

This guide is not meant to be a replacement for the text, and thus we will not present information (such as tables, charts) that is present in the text except where necessary.

All on the procedures given are general with the exception of the tutorial which is specific to VAX/VMS. The particular examples we give were checked for accuracy on a VAX 4000 using VAX/VMS 5.3 and VAX Fortran.

When we refer to a specific file path, the characters ' ... ' mean the file specification necessary to reach the level where the specific files we refer to have been installed in your system.

A suggestion: It is useful to have a separate subdirectory for each chapter in the text. An easy way to do this is with a system something like this:

chapter 1 [...KOONIN.CHAPTER_ 1]

and so on for each of the 10 chapters. This will simplify keeping track of the many files that most of the programs use. Also , keep in mind that as a package, the programs require at least 10.0 MBytes disk space to be used effectively. Be sure that this amount is available before beginning to avoid delays. We will point out when a particular program uses either large amounts of disk space or cpu time. Finally we have ensured that all programs run without errors on a Vax.

CHAPTER 1

THE NUCLEAR SHELL MODEL

• ABSTRACT

The codes consist of four separate programs. The first two, FDSMCFP and PD are used to calculate coefficients needed by the two main codes, FDSM and FDTR. FDSMCFP and PD need only be run one time and as long as the data files are retained, one need only generate an appropriate input file and run the program (FDSM or FDTR) of interest.

• FILES

FDSMCFP.FOR - This is the Fortran code for the segment that generates the coefficients of fractional parentage (CFPs).

PD.FOR - Fortran code to generate the Hamiltonian operator matrix elements

LIB.FOR - A Fortran library of often used subroutines

FDU0.FOR - Part 1 of the actual shell model code

FDTR.FOR - Part 2 of the shell model code (computes transitions)

FDSM.INP - A sample input file

FDTR.INP - A sample input file for transition calculations

• COMPILING, LINKING AND RUNNING

Note: In order to use the programs in this chapter, 5.0 Mbytes of disk space are required. The IMSL Fortran library is also required.

Begin by compiling the above five Fortran source codes separately. Next, link FDSM.OBJ, FDSM-CFP.OBJ, PD.OBJ and FDTR.OBJ to LIB.OBJ and the IMSL library. (They should *not* be linked to each other. See the tutorial.)

The first time these programs are used, FDSMCFP and PD will need to be run. These take only a few minutes of cpu time, with the exception of FDSMCFP's second run, which will take about 120 minutes of cpu time to complete.

First, run FDSMCFP giving it 'SO8' input in response to 'symmetry' prompt. Then, run it again (in batch mode) using 'SP6' input, which will, as stated before, require about 2 hours of cpu time. Then, when the job has finished, run PD twice, once for 'SO8' and once for 'SP6'. These runs will produce several output files. Keep these files as they are required for all subsequent runs of FDU0 and FDTR.

The programs FDU0 and FDTR take input from files named FDU0.INP and FDTR.INP. All input is done in one block format. Output is in the form of a comprehensive output file named FDTR.OUT or FDU0.OUT.

• OBTAINING RESULTS / NOTES

As stated previously, all input and output to/from the FDU0 and FDTR codes is via data files using a text format. The input file is in block form. Note that the dollar sign must be in the second column or else an input conversion error will occur.

An effective way to keep track of files is to write a separate input file for each problem you will be solving, then copy this file to either FDU0.INP or FDSM.INP and execute the program. When finished, copy the output file to a separate output file for each project.

A comprehensive description of the input parameters is provided in the text. The author provides a sample input data file which can be used to verify the programs output and to serve as the basis for experimenting with the parameters.

• TUTORIAL

Compile the five Fortran source code files.

```
$ FORTRAN FDSMCFP.FOR
$ FORTRAN PD.FOR
$ FORTRAN LIB.FOR
$ FORTRAN FDU0.FOR
$ FORTRAN FDTR.FOR
```

Each of the four program object files is now linked to the library object file as well as the IMSL library. (They should *not* be linked to each other.)

```
$ LINK FDSMCFP.OBJ, LIB.OBJ, IMSL/LIBRARY
$ LINK PD.OBJ, LIB.OBJ, IMSL/LIBRARY
$ LINK FDU0.OBJ, LIB.OBJ, IMSL/LIBRARY
$ LINK FDTR.OBJ, LIB.OBJ, IMSL/LIBRARY
```

The FDSMCFP program is run first to generate a file of coefficients. Run it first for 'SO8' symmetry, then as a batch job for the 'SP6' symmetry as this will take about 2 hours cpu time.

```
$ RUN FDSMCFP.EXE
SO8
```

Note that SO8 must be entered in UPPERCASE letters. This will create 4 new files named SO8P.DAT, SO8JSIZE.DAT,SO8CFP.DAT and SO8JSIZE.TAB.

A typical batch file would be

```
$ RUN FDSMCFP.EXE
SP6
```

Note that SP6 must be entered in UPPERCASE letters.

This will create 4 new files named SP6P.DAT, SP6JSIZE.DAT,SP6CFP.DAT and SP6JSIZE.TAB. These are needed as input for FDU0 and FDTR.

When the job has finished running, run PD for both 'SO8' and 'SP6' symmetries.

To run the main codes using the sample input file, simply type

```
$ RUN FDU0.EXE
```

and then to compute the transitions

```
$ RUN FDTR.EXE
```

The output will be written to files named FDSM.OUT and FDTR.OUT in a text format.

CHAPTER 2

THE SKYRME-HARTREE-FOCK MODEL OF THE NUCLEAR GROUND STATE

• ABSTRACT

The Skyrme-Hartree-Fock method is implemented in a single Fortran program SKHAFO. The code uses an iterative solution. A sample input file for the 17 [O] nucleus is provided.

• FILES

SKHAFO.FOR - Fortran source code for the Hartree-Fock analysis
FOR005.DAT - Sample input file

• COMPILING, LINKING AND RUNNING

There are no special requirements; simply compile the single source code file, link the object file and run the program.

• OBTAINING RESULTS / NOTES

All input and output is done using files. The input file must be named FOR005.DAT. Output is written to files named FOR006.DAT and FOR011.DAT. As with several of the programs in the text, a convenient method of processing data sets is to rename input files to FOR005.DAT, run the program and then rename the output to another file for safe keeping.

Comprehensive descriptions of the input parameters are provided in the text.

Do not be concerned with what may be interpreted as error messages in the output file that refer to for the PAIR routine, indicating termination of the calculation without convergence in the first few iterations. There is no reason to be concerned about this as the PAIR routine converges well in the later iterations.

• TUTORIAL

The first steps are to compile and link the program.

```
$ FORTRAN SKHAFO.FOR
$ LINK SKHAFO.FOR
```

Since a sample input file named FOR005.DAT is provided, simply run the program by typing $ RUN SKHAFO.EXE

The output is written to two files, FOR006.DAT and FOR011.DAT.

CHAPTER 3

THE CRANKED NILSSON MODEL

• **ABSTRACT**

The main code for this chapter is NICRA.

• **FILES**

NICRA.FOR - The Nilsson Cranker Fortran source code
APPEN.TEX - Text file giving example output and hints
INPUT1.DAT INPUT2.DAT - Two example input files for the study of 160 [Yb]
NICRAPAR.FOR - Include file of parameters
NICRAINC.FOR - Include file of common block statements

• **COMPILING, LINKING AND RUNNING**

While the basic installation procedure is simple enough, there a few fine points that may need attention.

The first of these is the file NICRAPAR.FOR. This file is included in the code via an include statement to the compiler and determines the dimension of several variables. The value here to be concerned about is the maximum number of shells. The default value is set at 6 shells. For a different number of shells, see the table below.

Before you set the number of shells to the maximum, keep in mind the size of the executable module *that results!*

The value for the variable MAXDIM to change in the file is shown in the table.

MAX N	MAX DIM	MODULE SIZE
4	22	0.1 MBytes
5	34	0.2
6	50	0.3
7	70	0.4
8	95	0.6
9	125	0.9
10	161	1.5
11	203	2.2
12	252	3.2

Secondly, if your system is not a VAX you will need to rewrite the include statements in NICRA.FOR.

• **OBTAINING RESULTS / NOTES**

As the program is written. NICRA expects to receive input from the terminal. A much more effective method is to run the program as a batch job, inserting the batch commands into the input file. Output is written to a text file.

The best method to use to run the program for different data set is to write a small batch file, then add it to you data sets before submitting the job. Processing takes about 2-15 minutes of cpu time depending on the number of shells and other parameters.

The amount and type of output the program produces can be controlled by changing values in the input file of the variables IN_LEV and LEV_PRINT.

A listing of input parameters as well as a sample input file appear in the text for reference.

The authors provide two examples in the files INPUT1.DAT and INPUT2.DAT. The first example shows how to construct a single particle diagram that is a plot of single orbitals as a function of angular speed of rotation. About 5 minutes of cpu time is required for this calculation using the values for LEV_PRINT in INPUT1.DAT file. The plot in Fig. 3.3 in the text can be produced simply enough by making a copy of the output file, formatting the necessary data correctly with labels and commands for TELL-A-GRAF or another similar graphics package.

The second example is an investigation of the triaxiality of the nucleus of 160 [Yb] as a function of spin. This run takes approximately 25 minutes of cpu time to complete. The graphs appearing in Fig. 3.4 can be produced in the same manner as the previous example.

• TUTORIAL

In this example we will run the program using the default number of shells, MAXN=6. The instructions to use more or fewer shells are given above. First, set the default directory to chapter 3 and compile the Fortran file NICRA.FOR. Then link the resulting object file. There is no need to worry about including the files NICRAPAR.FOR and NICRAINC.FOR as this is done automatically for VAX systems via include statements in the main code.

```
$ FORTRAN NICRA.FOR
$ LINK NICRA.OBJ
```

Note that the other codes should *not* be compiled because they are done via an INCLUDE statement as mentioned above.

Because the programs need a large amount of input data entered, it is best to run them as a batch job with the command $ RUN NICRA placed at the top of INPUT1.DAT. Then change its name say to BAT.COM and submit it as a batch job.

The second example can be run in the same way as the first. It is a good idea to use a different set of files for each run.

CHAPTER 4

THE RANDOM-PHASE-APPROXIMATION FOR COLLECTIVE EXCITATIONS

• ABSTRACT

The main code for this chapter is RPA.

• FILES

RPA.FOR - Fortran source code for the program

• COMPILING, LINKING AND RUNNING

This is most likely the easiest to use program in the text. Simply compile it, link it and run it. Input is read from the keyboard, output goes to the terminal.

• OBTAINING RESULTS / NOTES

When the program has been started, it waits for input. Simply type the input data on you terminal using the format given in table 4.1 in the text. The output will appear on the terminal also. The session can be captured for later review/analysis of output by using set host to record in a log file.

• TUTORIAL

Set the default directory to chapter 4, compile and link the program RPA.

```
$ FORTRAN RPA.FOR
$ LINK RPA.OBJ
```

Then start the program and input the parameters. The values shown are for the example of 16 [O] given by the author.

```
$ RUN RPA
0.25,50
16,8
1,3,1,0
-1,0,0,0
10,0.5
-1100,15000,0.5,0.93
1,0,40,1,1
0
```

CHAPTER 5

THE PROGRAM PACKAGE PHINT FOR IBA CALCULATIONS

• ABSTRACT

In the IBA model two different bosons are considered: the s- and d-boson. The program PCIBAXW calculates excitation energies and wave functions; PCIBAEM calculates electromagnetic transitions; CFPGEN is the code to generate coefficients of fractional parentage (CFPs).

• FILES

PCIBAXW.FOR - Main program and some subroutines in Fortran for calculation excitation energies and wave functions

PCIBAEM.FOR - Electromagnetic transition matrix elements and probabilities main code

CFPGEN.FOR - Main Fortran code for generating CFP file

PCIBALIB.FOR - Library of subroutines commonly used by codes

ANGMOM.FOR - Routines for calculating angular-momentum recoupling brackets

DIAG.FOR - Routine for the diagonalization of a real symmetric matrix

PCIBAEM.OUT - Sample output files PCIBAXW.OUT

• COMPILING, LINKING AND RUNNING

The six Fortran source files, PCIBAXW.FOR, PCIBAEM.FOR, CFPGEN.FOR, PCIBALIB.FOR, ANGMOM.FOR and DIAG.FOR, should first be compiled separately resulting in six object files. Next link the object files as shown:

PCIBAXW.OBJ to DIAG.OBJ, PCIBALIB.OBJ and ANGMOM.OBJ

PCIBAEM.OBJ to PCIBALIB.OBJ and ANGMOM.OBJ

CFPGEN.OBJ to ANGMOM.OBJ

At this point the result should be three executable files.

• OBTAINING RESULTS / NOTES

All three programs are set up to accept input from the terminal and write their output to a file. The author of the code has provided full prompting for each input data item, making the programs easy to use. A table describing the input parameters is given in the text.

Program output is written to a file with the name of the program followed by .OUT . Before running the two main programs, run CFPGEN to create the file PHINT.CFP. For applications, run PCIBAXW to generate spectra followed by PCIBAEM to generate transitions rates. The authors provide sample output files.

Since output is written to the same file, the scheme of renaming the output file will be necessary to save results for future use.

How to calculate for a single nucleus:

1. Determine the number of bosons. The number of bosons is equal to the number of fermion pairs outside a closed shell. As an example, take 104 46 [Pd] 58. Here,

Neutrons: -4 fermions +2 bosons
Protons : +8 fermions +4 bosons
Total : +6 bosons

2. Determine the strategy: which limiting case? For a more rotational-like spectrum it is better to use multi-pole operators while for a vibrational case the appropriate method is not to use the multipoles but to

define the Hamiltonian in terms of HBAR, C, F and G.

3. Fit the parameters in the Hamiltonian Make a first guess of the parameters using the analytic formula given in the text.

4. You are now ready to run PCIBAEM. Several recipes for these runs are given in the text.

• **TUTORIAL**

Begin by compiling the six Fortran source code files.

```
$ FORTRAN PCIBAXW.FOR
$ FORTRAN PCIBAEM.FOR
$ FORTRAN CFPGEN.FOR
$ FORTRAN PCIBALIB.FOR
$ FORTRAN ANGMOM.FOR
$ FORTRAN DIAG.FOR
```

After the six object files have been generated, link the files as shown here.

```
$ LINK PCIBAXW.OBJ, DIAG.OBJ, PCIBALIB.OBJ, ANGMOM.OBJ
$ LINK PCIBAEM.OBJ, PCIBALIB.OBJ, ANGMOM.OBJ
$ LINK CFPGEN.OBJ, ANGMOM.OBJ
```

Next, you will need to run CFPGEN to create the file PHINT.CFP

```
$ RUN CFPGEN.EXE
```

In this last section, a sample input for PCIBAXW and PCIBAEM is shown. The programs are completely self-prompting and therefore easy to use. Only the responses are shown. These examples can be found on pp93-97 of the text.

EXAMPLE 1 : USING PCIBAXW
[R] means press RETURN or ENTER

```
$ RUN PCIBAXW.EXE
N [R]
7 [R]
Y [R]
0.5 [R]
[R]
-0.1 [R]
[R]
[R]
[R]
[R]
[R]
[R]
[R]
Y [R]
4 [R]
```

EXAMPLE 2 : USING PCIBAEM

In this example note that E2 must be entered in UPPERCASE letters.

```
$ RUN PCIBAEM.EXE
E2 [R]
[R]
2 [R]
1 [R]
0 [R]
```

-2 [R]
1 [R]
0 [R]
100 [R]
S [R]

Note: Pressing return [R] in response to a prompt instructs the program to use the default value for that parameter (see the text for the default values).

CHAPTER 6

NUMERICAL APPLICATIONS OF THE GEOMETRIC COLLECTIVE MODEL

• ABSTRACT

The main code for this chapter is GCM.

• FILES

GCM.FOR - Fortran source file for main program

ANGP.DAT - Data file provided that contains the parameters for the Hamiltonian

ANGQ.DAT - Data file provided that contains corresponding values of matrix elements for quadrupole operator.

INPUT.DAT - Sample input file for calculations with 186 [Os]

PARA.DAT - Parameters data file of Hamiltonian

• COMPILING, LINKING AND RUNNING

To run the GCM code, the IMSL Fortran library is required.

Compiling and linking the code is fairly straightforward. There is one source file GCM.FOR which should be compiled and the object file linked to the IMSL library. The program is then ready to use.

• OBTAINING RESULTS / NOTES

The program expects to read the files ANGP.DAT and ANGQ.DAT from units 20 and 21. Thus, you should copy these files as shown below:

```
copy ANGP.DAT to FOR020.DAT copy ANGQ.DAT to FOR021.DAT
```

As the example is set up, the code expects to read Hamiltonian parameters from unit 40. These parameters, which for the example are provided in the file PARA.DAT, can be read or calculated depending on the value of the variable IFPARA in the first line of the input file. To use the example as

it stands, copy PARA.DAT to FOR040.DAT.

The program expects to read input from the terminal, and thus it is suggested to run the program as a batch job using the input from a file to avoid typing errors since the program doesn't provide much in the way of error correction. Create a batch file to run the program, then append the given input file to it.

Output from the program is written to a file named OUTPUT.DAT, which includes a rudimentary graph meant for line printers. However, data from the output file can be used with relative ease to create plots of the type shown in the text by using TELL-A-GRAF or a similar graphics package.

• TUTORIAL

We will begin by setting the default directory and compiling the Fortran code. Then the object file is linked to the IMSL library.

```
$ FORTRAN GCM.FOR
$ LINK GCM.OBJ, IMSL/LIBRARY
```

Next, the input data files must be copied into appropriate Fortran unit files.

```
$ COPY ANGP.DAT FOR020.DAT
$ COPY ANGQ.DAT FOR021.DAT
$ COPY PARA.DAT FOR040.DAT
```

As the final step before running the program, create a batch file and append the input file to it. This will allow you to run the code for the 186 [Os] example given in the text.

Create a batch file named, for instance, GCM_FIRST_TRY.BAT, using a editor It should contain

```
$ RUN GCM
$ APPEND INPUT.DAT GCM_FIRST_TRY.BAT
```

Now run the program and the output is written to OUTPUT.DAT Remember that we are running it in batch mode.

```
$ SUBMIT/LOG_FILE=[...KOONIN.CHAPTER_6]GCM.LOG/NOPRINTER GCM_FIRST.BAT
```

The SUBMIT qualifiers used should be familiar by now.

CHAPTER 7

THE RELATIVISTIC IMPULSE APPROXIMATION

• ABSTRACT

The main codes for this chapter are TIMORA, FOLDER and HOOVER.

• FILES

TIMORA.FOR - Fortran code for the first section of the procedure that generates scalar and bayeron densities for neutrons and protons.

FOLDER.FOR - The Fortran code for the second section of the procedure that processes the densities into Dirac scalar and vector optical potentials.

HOOVER.FOR - Fortran code for the final program segment that takes input from FOLDER and adds coulomb potentials and computes the observable scattering.

TIMORA.INP FOLDER.INP - Example Input and output files provided by the author

• COMPILING, LINKING AND RUNNING

The three Fortran codes should be compiled and linked separately. To ensure correct results it is suggested that the three programs be run as described below.

• OBTAINING RESULTS / NOTES

First, run the program TIMORA. It will display the status of the run on the terminal. Next, run FOLDER, which will advance the solution a second step. Then, as a final step, run HOOVER. The total cpu time required to complete the run using the sample input data in the text was about 5 minutes.

It is up to the user to decide whether to run the codes in a batch file or interactively. It might be beneficial to run the programs all as a single batch job when large input data sets are to be processed or else there are large calculations to be done as these can require anywhere from 15 minutes to about 4 hours cpu time for any reasonable calculations that might be desired.

• TUTORIAL

As usual, set the default directory, then compile and link the three *separate* code segments.

```
$ FORTRAN TIMORA.FOR
$ FORTRAN FOLDER.FOR
$ FORTRAN HOOVER.FOR
$ LINK TIMORA.OBJ
$ LINK FOLDER.OBJ
$ LINK HOOVER.OBJ
```

The second step after the programs have been compiled and linked is to run them in order. Results are displayed on the terminal as the programs run to let the user know the status of the programs, any final output is written to data files in a text format.

So, now simply run the programs.

```
$ RUN TIMORA.EXE
$ RUN FOLDER.EXE
$ RUN HOOVER.EXE
```

CHAPTER 8

THREE-BODY BOUND-STATE CALCULATIONS

• ABSTRACT

The main code for this chapter is TRIMOD. This is the only chapter where we have modified our codes. We have done this so that they run with the IMSL library rather than the NAG library.

• FILES

TRIMOD.FOR - Source code for the Fortram program

• COMPILING, LINKING AND RUNNING

As the code was orginally written, the NAG Fortran library is required. Since it is apparantly not widely-used in the United States, we have modified the code to use IMSL procedures instead. The basic modifications included substituting the IMSL Gaussian quadrature subroutine for the NAG version used in the original code. The other NAG routine used was one that solved a linear system with multiple right hand sides. Since there was no directly corresponding routine in the IMSL library, we used IMSL's LU-factorization routine first, then used a loop to solve each right-hand-side using a single RHS linear system solve procedure. This is implimented with the subroutine MRHSLS added to the end of the code.

• OBTAINING RESULTS / NOTES

The program writes output to the screen. In order to save this for future use, it is convenient to use the SET HOST command with the qualifier /LOG_FILE= in order to capture the output in a specific file.

• TUTORIAL

First set the default directory, then compile the code and link it to the IMSL Fortran library.

```
$ FORTRAN TRIMOD.FOR
$ LINK TRIMOD,IMSL/LIB
$ RUN TRIMOD.EXE
```

CHAPTER 9

VARIATIONAL MONTE-CARLO TECHNIQUES IN NUCLEAR PHYSICS

• ABSTRACT

The main code for this chapter is VARMC.

• FILES

VARMC.FOR - Fortran source code file for the simulation program
VARMCH3.IN - A sample input file provided by the author
VARMCH3.OUT - An example output file provided by the author

• COMPILING, LINKING AND RUNNING

The actual procedure for getting the code installed is relatively simple: the source code is compiled to give an object code file which is then linked to give the executable program.

• OBTAINING RESULTS / NOTES

All input to the program is through the file VARMCH3.INP Program output is written to several data files in a text format. See the text for a full description of the input parameters.

• TUTORIAL

First, select the default directory and compile, then link the program file.

```
$ FORTRAN VARMC.FOR
$ LINK VARMC.OBJ
```

Now you are ready to use the program. It should be run in batch mode with the command $ RUN VARMC placed at the top of the VARMCH3.IN file. Also the header in this file MUST be removed in order for the program to run properly. Rename the whole file to JOB.COM and then submit as a batch job.

CHAPTER 10

ELECTRON-SCATTERING FORM FACTORS AND NUCLEAR TRANSITION DENSITIES

• ABSTRACT

The main codes for this chapter are ELHO, MAHO, WSAXE and WSAXM.

• FILES

The first four files here are source code for the four main programs

PROGRAM FILE	WAVE FUNCTION	TYPE OF TRANSITIONS
ELHO.FOR	HARMONIC OSC	ELECTRONIC
MAHO.FOR	HARMONIC OSC	MAGNETIC
WSAXE.FOR	WOODS-SAXON	ELECTRONIC
WSAXM.FOR	WOODS-SAXON	MAGNETIC

ELLIB.FOR - Library of math subroutines required by most of the code

The authors have also included several sample input files to allow the programs to be run right away. The user will need to supply data for WSAXE and WSAXM to run the example for these.

• COMPILING, LINKING AND RUNNING

The Fortran source code files are compiled separately and then the files are linked as follows:

ELHO.OBJ is linked to ELLIB.OBJ
MAHO.OBJ is linked to ELLIB.OBJ
WSAXM.OBJ is linked to ELLIB.OBJ
WSAXE.OBJ is linked to ELLIB.OBJ

Input is expected to be read from the terminal, which makes the use of batch processing convenient; output is done with data files in text format.

As we have described in previous section, the most convenient method to use in running the prograams is to append the input data set to a standard batch file.

• OBTAINING RESULTS / NOTES

Since the codes require that input come from the terminal, running the programs in the batch mode using the input files added (appended) to the batch file is the easiest method to obtain results with the minimum of fuss.

• TUTORIAL

Begin by setting the default directory, then compiling and linking the program segments.

```
$ FORTRAN ELHO.FOR
$ FORTRAN MAHO.FOR
$ FORTRAN WSAXE.FOR
$ FORTRAN WSAXM.FOR
$ FORTRAN ELLIB.FOR
$ LINK ELHO.OBJ, ELLIB.OBJ
$ LINK MAHO.OBJ, ELLIB.OBJ
$ LINK WSAXE.OBJ, ELLIB.OBJ
$ LINK WSAXM.OBJ, ELLIB.OBJ
```

Then run ELHO and MAHO First, you will need to create batch files for this purpose. Call them, for example, JOB1.COM and JOB2.COM

The file JOB1.COM should contain

```
$ RUN ELHO
```

and likewise, the file JOB2.COM should have the corresponding commands

```
$ RUN MAHO
```

To use the example data files, append them to the batch files.

```
$ APPEND ELHO.INP JOB1.COM
$ APPEND MAHO.INP JOB2.COM
```

Then submit each job to run ELHO and MAHO.

See the text for descriptions of the required input data for running WSAXE and WSAXM. They are run using the same procedures as shown above.