Chapter 17
TIME SAVERS IN INTERACTIVE TSP

This chapter outlines interactive TSP features that save you time and make the program more convenient to use. Some of these, such as re-executing modified commands, you may use every interactive session; others, like automatic backup and recovery, are "there when you need it". See the *Reference Manual* for further information.

17.1. Revision and re-execution of commands

Interacting with a program implies that your commands may depend on past output. Often this only involves modifying your previous command. You might need to fix a typo, change a list of options, add a term to an equation, etc.... In any case, you are spared retyping the whole command. Interactive TSP offers several ways to do this. Note that the following sections apply to *commands* only; data should be modified with UPDATE.

17.1.1. Re-execution

There will be occasions when you want to execute a command or group of commands entered previously without performing a modification. The EXEC command is provided for this purpose.

```
EXEC 10
```

would re-execute line 10. If two line number arguments are given, the range of lines between them (including them) will be executed.

Some examples of when EXEC might be used:

1. Execution was initially suppressed with an EXIT command in collect mode or an input file.
2. Errors were discovered in the data and results need to be regenerated after an UPDATE has been performed.
3. Commands you use more than once and don't want to retype (for example, a SMPL with lots of gaps).
4. A modification has been performed separately.

The use of EXEC in collect mode is slightly different, and is discussed in the *Reference Manual*.

17.1.2. Modifying commands and fixing typos

In Chapter 4, we introduced the easiest way to revise commands and re-execute them when you are interacting with TSP: editing using the cursor keys. However, there are time when you would like to simply add or delete a variable name, or re-execute a whole set of commands as a group. For these purposes, you will find the interactive commands EDIT, RETRY, ADD, and DROP useful. In this section, we describe EDIT and RETRY. In the next section, we describe ADD and DROP, which are special cases of RETRY.

Use EDIT and RETRY when you want to change the arguments in a command. EDIT makes the changes without executing them. RETRY combines EDIT, which makes changes, and EXEC, which executes them. Syntax, "arguments" and editing rules are explained in the *Reference Manual*, and are identical for both EDIT and RETRY. Here are some examples of using EDIT:

```
EDIT 10
```

would request modification of line 10; the line will be displayed, and the edit prompt issued:

```
10. CONS C GNP
>>
```

If no line is requested, the previous line is assumed. Only one line may be edited at a time (i.e. only one line number argument will be accepted). The basic editing functions are insert, delete, and replace.

Possible uses for EDIT are:

1. Modifying something that is not executable (for example, an equation definition using FRML).
2. Modifying a range of lines before re-executing them as a group, or modifying a line within a range that will then be executed as a group.
3. Correcting mistakes made while entering commands in collect mode.

RETRY is appropriate when:

1. Correcting typos in interactive mode.
2. Re-executing a command with a modified list of options or variables.

The following example illustrates the relationship between EDIT and RETRY, as well as basic use of the editor:

```
 10? INT DP,DP1,LGNP,TIME,C INVR C,G,LM,TIME
<error message because procedure INST is misspelled>
 11? EDIT 10
  >> REPLACE INT INST 1
  >> EXIT
  10. INST DP,DP1,LGNP,TIME,C INVR C,G,LM,TIME ;
 12? EXEC 10
  10. INST DP,DP1,LGNP,TIME,C INVR C,G,LM,TIME ;
   <output from the INST command.>
```

This same change can be performed more easily with the RETRY command (note that we have used all possible abbreviations and omissions to save typing).

```
 11? RET
  >> R INT INST
  >>
  10. INST DP,DP1,LGNP,TIME,C INVR C,G,LM,TIME ;
   <output from the INST command.>
```

17.1.3. Adding and dropping variables

ADD and DROP are used to add or drop variables from the previous command and automatically re-execute it. They simplify the task of re-executing the previous estimation command with a modified list of series, but are not limited to that particular use.

ADD and DROP both make permanent modifications to the "previous" command. The "previous" command, is defined as the last TSP command that *is not itself* ADD or DROP.

```
ADD var1 var2
```

is identical to

```
RETRY
>> INSERT var1
>> INSERT var2
>> EXIT
```

Both insert var1 and var2 at the end of the previous command (presuming there was not a sequence of ADDs and DROPs) and execute it. Similarly,

DROP var1 var2

is identical to

RETRY
>> **DELETE var1**
>> **DELETE var2**
>> **EXIT**

Both delete the first occurrences of var1 and var2 in the previous command (same assumptions) and execute it.

It is not possible to combine ADD and DROP into one step to perform a REPLACE function, or to make compound modifications. In these circumstances, the RETRY procedure or up-arrow editing must be used.

17.2. Reading commands from disk

An intermediate approach to running TSP programs which combines features of batch and interactive execution is provided by the INPUT command. This procedure reads a TSP command file from disk and executes it in TSP; following its execution you are left in interactive mode with all the data and results available for use.

The command

INPUT *filename*

will read filename.tsp in the current directory and execute it. All commands read are added to the session log, and may subsequently be REVIEWed, EDITed, etc....

You will also be given a choice as to whether you want the output directed to the terminal or to an output file. Creating an output file this way differs from the OUTPUT command in two ways: 1) if the output file already exists, a new version will be created (possibly overwriting the old), and 2) the file will automatically be closed at the end of execution of the input file (you may still append to it with the OUTPUT command).

An input file may be an entire TSP program prepared for batch execution or groups of TSP commands frequently executed as a unit. Advanced users might find it convenient to keep a library (i.e., directory) of TSP PROCs for use as input files. Since a PROC is not executed until it is "called", these files would produce no output when read -- adding a "SHOW procname" command at the end of each would provide quick information on the PROC just defined. Input files may be nested up to a depth of 5.

Reading an input file is exactly the same as using collect mode -- the difference is the source of the command stream. All commands will be read until execution is "requested"; for an input file this means encountering an END statement, or end-of-file. Just as in collect mode, execution may be suppressed by replacing the END statement with EXIT; you may then REVIEW what was read, and EXEC just the sections you need.

17.2.1. Your TSP login file

Every time you begin an interactive (or batch) session, TSP looks for a file called LOGIN.TSP before prompting you to begin entering commands. This file is a special case of an input file. The only difference is that execution is automatic at startup, and you will not be given the option of directing any resulting output to a disk file. If LOGIN.TSP does not exist in the current directory, your home or root directory will be searched as well.

If much of your work with TSP uses the same data, you may want to use LOGIN.TSP to set the SMPL and FREQ, load data, or open databanks. It is also convenient to place the OPTIONS command here.

17.3. Talking to the operating system (for DOS users)

A great deal of flexibility has been added to interactive TSP by allowing interaction with the operating system. SYSTEM allows you to suspend your session temporarily, take care of other business on the system, and pick up later right where you left off. SYSTEM takes no arguments, and simply produces the message

```
Enter commands.  Type CONTINUE to resume TSP session.
$
```

you may now create or modify files, or enter other commands. You may enter commands as long as you like;

```
$ CONTINUE
```

will resume your interactive session with no loss of continuity. There are many uses for this feature, one of the most apparent is the ability to fix data files or examine output files created during your session without halting the program. Please note that in order to use it in this way, the files in question must be closed first with the CLOSE, TERM, or OUTPUT command.

17.4. Automatic backup and recovery

Interactive TSP saves a copy of the input of your session when you exit the program, and if your session is terminated abnormally this copy provides the ability to recover the session.

During an interactive session, TSP automatically saves the command stream in a file named INDX.TMP. INDX.TMP is used by commands like EDIT which modify the command stream. INDX.TMP is written in a special format for this purpose. Upon normal exit from TSP, INDX.TMP is used to create the standard sequential file, BKUP.TSP, found in your directory and INDX.TMP is deleted.

BKUP.TSP is useful for reconstructing a session and can be run by TSP in batch mode, since interactive commands have been stripped from it. Note, however, that data entered with the ENTER and UPDATE is not saved in BKUP.TSP -- data should be saved in a databank or a SAVE file (see Chapter 13).

If you accidentally (or purposely) type CTRL-C, TSP will terminate abnormally. The same is true if the program encounters a fatal Fortran error or system failure. In these cases BKUP.TSP will not be created. However, if you reenter the program, TSP will be able to recover your session from the file INDX.TMP which still remains on disk. Only commands will be recovered -- you must EXEC portions whose results you want reinstated.

If TSP finds a file INDX.TMP on disk when you start the program, you will get the message:

```
WARNING> Your previous TSP session was terminated abnormally.
   Do you wish to recover it (y/n)? [y]
```

If you have changed directories or accounts since termination, or renamed INDX.TMP, TSP will not know you need to recover the session. You must then request recovery with the RECOVER command. Rules for specifying a filename are the same as those for the INPUT and OUTPUT commands.

Note: Do *not* try to use INDX.TMP as an INPUT file, or edit it (you will destroy the special format). Conversely, do not try to RECOVER any files that were not originally an INDX.TMP file created by TSP.