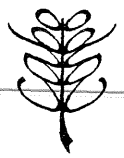
Simulation Reports CABO-TT

PCSMP on IBM PC-AT's or PC-XT's and compatibles

D.M. Jansen, R.T. Dierkx, H.H. van Laar & M.J. Alagos

Simulation Report CABO-TT nr. 15

A joint publication of



Centre for Agrobiological Research (CABO)

and

Department of Theoretical Production Ecology, Agricultural University

Wageningen 1988

Simulation Reports CABO-TT

Simulation Reports CABO-TT is a series giving supplementary information on agricultural simulation models that have been published elsewhere. Knowledge of those publications will generally be necessary in order to be able to study this material.

Simulation Reports CABO-TT describe improvements of simulation models, new applications or translations of the programs into other computer languages. Manuscripts or suggestions should be submitted to:
H. van Keulen (CABO) or J. Goudriaan (TPE).

Simulation Reports CABO-TT are issued by CABO and TPE and they are available on request. Announcements of new reports will be issued regularly. Addresses of those who are interested in the announcements will be put on a mailing list on request.

Address

Simulation Reports CABO-TT
P.O. Box 14
6700 AA Wageningen
Netherlands

Authors' affiliations

D.M. Jansen[1]
R.T. Dierkx[2]
H.H. van Laar[2]
M.J. Alagos[3]

[1] Centre for Agrobiological Research (CABO, Wageningen)
[2] Department of Theoretical Production Ecology (Agricultural University, Wageningen)
[3] International Rice Research Institute (IRRI, Los Baños, Philippines)

Table of contents

1. INSTALLATION of PCSMP on PC

1.1 System requirements

The present version 4.0 of PCSMP is suited for PC's with the following specifications:

512 KB working memory (RAM), but preferably more
10 MB hard disk or more
at least 1 disk drive for 360 KB or 1.2 MB
DOS 2.11 or later versions

Before starting PCSMP, be sure that there is a PATH to the subdirectory where the DOS commands are located, and to the COMMAND.COM file. This can be tested by typing the DOS command PATH.

A coprocessor is not needed, but the performance of PCSMP is faster if a 80287 mathematical coprocessor is added to the CPU.

At this moment, PCSMP is known to run on:

-- IBM AT (with and without 80287 coprocessor)
-- IBM XT (without 8087 coprocessor)
-- ICM AT (without 80287 coprocessor)
-- MITAC COMPUTITAN (with and without 80287 coprocessor)
-- Olivetti M24 (with and without 80287 coprocessor)
-- SWIFT 1200 plus - AT-2586-A (without 80287 coprocessor)
-- TRIGEM 286 III (without 80287 coprocessor)
-- UPTRON PC/AT (unknown about 80287 coprocessor)
-- TANDY 3000 (with and without 80287 coprocessor)

The PLOTCSM program that enables plotting of output made while running a PCSMP program, is available for CPU's with a HERCULES MONOCHROME GRAPHICS CARD (or compatible), or with an EGA card. To enable plotting on the screen, also a MONOCHROME MONITOR is needed. A graphics printer (e.g. EPSON FX-85, or LX-800) is required to obtain hard-copies of the plot-output.
The plot program is not required for adequate running of PCSMP on the CPU, it merely makes its output more enjoyable. Thus, even if the system in use does not fulfill the plot requirements, the user can still run PCSMP.

1.2 DESCRIPTION OF THE REQUIRED FILES.

8087.COM
command file to switch the 80287 chip (if present) on or off

8087ONLY.LIB
MS-FORTRAN 8087ONLY library (see EMULATOR.LIB)

ANSI.SYS
System file to control the video display. This file should be present in the root directory.

AUTOEXEC.BAT
The general command procedure. One of its actions is to set the system path. This file should be present in the root directory.

C8087.EXE
executable file to make the linking procedure depending on whether the 80287 chip is enabled (linking with 8087ONLY.LIB) or not (link with EMULATOR.LIB)

CONFIG.SYS
The system configuration file, which should be present in the root directory. It should contain the commands 'FILES=20', 'BUFFERS=40', 'DEVICE=C:\ANSI.SYS', 'FCBS=8,0', and 'BREAK=ON'

EMULATOR.LIB
The MS-FORTRAN EMULATOR library. If a 80287 mathematical coprocesso is available (and enabled), most of the calculations are performed this coprocessor. If not available or disabled, the coprocessor is emulated in software. If the 80287 chip is enabled, also the 8087ONLY.LIB could be used.

EXECUT.LIB
The PCSMP library, containing OBJECT code modules for various subroutines and functions that are typical for PCSMP.

FOR1.EXE, FOR2.EXE, FOR3.EXE
Together these files are the IBM MS-DOS FORTRAN compiler version 2. This compiler supports the subset of the FORTRAN-77 ANSI standard. is a 3 stage compiler. The first (FOR1) and second (FOR2) stage generate the object code, the third stage (FOR3) generates an assembly listing. In everyday use of PCSMP, this third stage is not needed.

LIB.EXE
the FORTRAN library manager. It enables combining of OBJECT MODULES into an OBJECT LIBRARY. It is used during creation of a user-made subroutine library.

LINK.EXE
The MS-DOS linker, version 2.10. It is a one stage linker. Therefor the order of presenting the object code and libraries is important. In the batch file PCSMP.BAT, the correct order is provided.

MAIN2.OBJ
The PCSMP main program in OBJECT code. It takes care of a correct sequence of sections in the program made by the user. Also it updat the program variable TIME (depending on the method of integration) this part. It always is LINKED before the OBJECT code of the progra made by the user. This is done by commands in PCSMP.BAT.

PCSMP.BAT
The command procedure controlling the process of translating, compiling, linking and executing.

PCSMPFIG.EXE
The routine producing the PCSMP logo on the screen

PLOT.BAT
The command procedure to invoke plotting of output generated by PCSMP. It should not be called on subdirectory C:\PCSMP, as it also deletes the files STPL.FNT, SROM.FNT, and TPLGSA.DAT (which are stored on C:\PCSMP for use on other directories). Protecting these files against deleting (with ATTRIB +R) will cause errors during plotting.

PLOT.TXT
File containing some information about PLOTCSM

PLOTCSM.EXE
The routine producing the plots. Invoked by PLOT.BAT

README.TXT
File containing some information about PCSMP.

STOP.BAT
A command procedure to cause ending of another command procedure with a beep to notify the user.

TRANS.EXE
The PCSMP translator from PCSMP to FORTRAN. During TRANS, the following files are made: FOR03.DAT, UPDATE.FOR, DATA.FOR, TABLE.TMP. The latter three are used for further compilation and execution.

ZLINK.BAT
Batch file to facilitate linking with right subroutine library

1.3 Installation of PCSMP

In APPENDIX B, a procedure is described to install PCSMP on AT or XT (or compatible). The command files PCSMP.BAT and PLOT.BAT that invoke PCSMP and PLOTCSM respectively, are used to execute a PCSMP program. In these files, various command files are assumed to be found on two subdirectories: C:\PCSMP and C:\FORTRAN.

On directory C:\PCSMP the following files should be found:

1. TRANS.EXE
2. EXECUT.LIB
3. MAIN2.OBJ
4. PCSMP.BAT
5. STOP.BAT
6. ZLINK.BAT
7. C8087.EXE
8. README.TXT
9. PLOT.BAT
10. PLOT.TXT
11. PLOTCSM.EXE
12. STPL.FNT
13. SROM.FNT
14. TPLGSA.DAT
15. TEST.BAT
16. TEST2.BAT
17. PCSMPFIG.EXE
18. SUCROS.CSM
19. LARGE.CSM
20. SUBR.FOR

On directory C:\FORTRAN the following files should be found:

1. FOR1.EXE	4. LINK.EXE	7. IBMFOR.LIB
2. FOR2.EXE	5. EMULATOR.LIB	8. LIB.EXE
3. FOR3.EXE	6. 8087ONLY.LIB	9. 8087.COM

This directory structure is of course just an example. Some users mig prefer to put the PCSMP and FORTRAN programs on different subdirector (e.g. C:\SYS\PCSMP and C:\SYS\MSFOR2). This can easily be achieved by copying the files from C:\PCSMP and C:\FORTRAN to the respective subdirectories. However, in the batch files PCSMP.BAT and PLOT.BAT, t references to C:\PCSMP and C:\FORTRAN should be changed to the direct names preferred by the user. Also the PATH statement in the AUTOEXEC.I should be changed accordingly.

1.4 IN CASE OF TROUBLE AFTER FIRST INSTALLATION

Many error messages (e.g. Bad Command or file name) can occur if there is no PATH to the COMMAND.COM, or when this COMMAND.COM is of a DOS version older than version 3.2. Check this first before going into details on the error messages.
At the same time it should be checked whether the path is set to the directories where the PCSMP files and the FORTRAN files are found.

To find out in which phase an error occurs, the user can type in the various commands manually (Chapter 2. The consecutive stages of PCSMP and intermediate files).

2. THE CONSECUTIVE STAGES OF PCSMP AND INTERMEDIATE FILES

2.1 Introduction

After invoking PCSMP (i.e. the PCSMP.BAT file) translation, compilation, linking and execution take place. For each of these phases, an explanation is given of what happens, which files are being made, and what commands the user has to give to see what happens during any of the 4 phases.

2.2 TRANSLATION PHASE

The PCSMP file is translated in FORTRAN, and additional files are created:

FOR03.DAT : a copy of the source file, containing error messages (if during translation a PCSMP error was detected), a list of type and number of parameters and variables in the model, and a list with the order of calculation of the variables.
UPDATE.FOR : the sorted FORTRAN version of SUCROS.CSM, containing the statements of the program in the form of a subroutine.
DATA.FOR : a list of names and values of common variables
CONTRO.SYS : a list of the INPUT (names plus values) and the execution control and output control statements.
TABLE.TMP : the dimensions of arrays in the source program.

FOR03.DAT file is not used in the following phases. DATA.FOR and UPDATE.FOR will be compiled into OBJECT files (see next phase) and CONTRO.SYS and TABLE.TMP will be used as input files during the execution phase.

The command to invoke translation SUCROS.CSM is: **TRANS SUCROS.CSM<ENTER>**

2.3 COMPILATION PHASE

In this phase, the files UPDATE.FOR and DATA.FOR are compiled using the IBM-FORTRAN compiler (version 2.0). This results in the files:

DATA.OBJ : containing the object code of some data statements
UPDATE.OBJ : containing the object code of the program structure
UPDATE.LST : a copy of UPDATE.FOR, with all lines numbered, and (if errors were found) error and warning messages (only when the D option was used when invoking PCSMP).

The commands to achieve this are:

FOR1 UPDATE,,UPDATE;<ENTER>
FOR2<ENTER>
FOR1 DATA;<ENTER>
FOR2<ENTER>

2.4 LINKER PHASE

An executable version of the program is made (called UPDATE.EXE) by combining the following OBJECT code files in the same order as given here:

MAIN2.OBJ : the PCSMP main program, containing the calls for several PCSMP subroutines to check integration (method and step size) and a call for UPDATE, the subroutine containing t program lines. MAIN2.OBJ is the same for all PCSMP files It is stored on C:\PCSMP. In case it is stored on anothe subdirectory, the user has to change the LINK command in PCSMP.BAT accordingly.
UPDATE.OBJ : see under COMPILATION PHASE
DATA.OBJ : -do-

In these OBJECT modules, both PCSMP and FORTRAN subroutines are calle for, and sometimes also user supplied subroutines. These subroutines found in subroutine LIBRARIES:

SUBR.LIB : where the user made subroutines are stored in object for Stored on the subdirectory C:\MACROS\SUBR
EXECUT.LIB : the PCSMP subroutines in OBJECT modules. Stored on C:\PC
EMULATOR.LIB:the FORTRAN subroutines in OBJECT modules. It is the MS- FORTRAN EMULATOR library. Stored on C:\FORTRAN
8087ONLY.LIB:as EMULATOR.LIB to be used exclusively with 80287 mathematical coprocessor

The complete link command (to be typed as **one continuous line**) is:

LINK C:\PCSMP\MAIN2+UPDATE+DATA,UPDATE,,C:\MACROS\SUBR\SUBR+ C:\PCSMP\EXECUT+C:\FORTRAN\EMULATOR<ENTER>

In case no user-made subroutine library is needed, the command is (ag one continuous line; In both cases EMULATOR can be replaced by 8087ONLY):

LINK C:\PCSMP\MAIN2+UPDATE+DATA,UPDATE,,C:\PCSMP\EXECUT+ C:\FORTRAN\EMULATOR<ENTER>

2.5 EXECUTION PHASE

The actual commands in the program are evaluated, and if no errors (e division by zero) are found, the required output will be calculated a written into the following files:

FOR06.DAT : Contains a list of all INPUT and the requested output in tables (and/or figures). If an error was found in the inp no output will be found, and a remark will be written und the location of the error (preceded by ***).
PREP1.DAT : The maxima and minima for the output variables that were made available for plotting through the PREPARE statement. It is used by PLOT, to calculate the maxima and minima of the X and Y axes of the plots.
PREP2.DAT : The values of the output variables that were made availabl for plotting through the PREPARE statement. It is input t PLOT.

The command to execute the program (after translation, compilation an linking): **UPDATE<ENTER>**

3. EVERY DAY USE OF PCSMP

3.1 In general

After installation of PCSMP the user will be able to run PCSMP from any subdirectory (although it is advisable **NOT** to run it on the subdirectory where the PCSMP program files are stored (default on C:\PCSMP), as some PLOT files will be deleted).

The commands involved are simple and straightforward:

PCSMP.BAT is invoked by typing **PCSMP**

The file that the user want to run (e.g. SUCROS.CSM) should follow the PCSMP command: **PCSMP SUCROS.CSM**<ENTER>

In doing so, all the intermediate files discussed in chapter 4 will be made. However if the user would check his/hers subdirectory, many of the intermediate files appear to be deleted. This is taken care of by the PCSMP.BAT file. Upon successful or unsuccessful completion of the 4 phases, it will delete the following files:

UPDATE.FOR	DATA.FOR
UPDATE.OBJ	DATA.OBJ
UPDATE.MAP	TABLE.TMP
UPDATE.EXE	CONTRO.SYS

3.2 Options

Especially when errors are encountered during one of the phases (mostly during compilation or linking), it can be helpful to save some files. To achieve this, the user could type in manually all the commands described in paragraph 4. However, to make life a little easier, there are some options for using PCSMP, that provide the same result. Sometimes the user would like to save all the intermediate files, at other occasions he/she also would want to call the FORTRAN DEBUG.
These two options are abbreviated to S and D and should follow the filename:

PCSMP SUCROS.CSM S<ENTER>	will result in execution of SUCROS.CSM, and all the intermediate files will be Saved.
PCSMP SUCROS.CSM D<ENTER>	will result in the same as the S option, plus that the files UPDATE.LST and UPDATE.MAP will be created. Especially UPDATE.LST can be helpful in tracing FORTRAN ERRORS. If such an error occurs, the number of the line in UPDATE.LST will be written on the screen. In UPDATE.LST the type of error can be found

If the intermediate files are saved (either with option S or D), the following options can be used to skip part of the process from translation to execution.
When using the following options, there is no need for a filename. A possible command could be:

PCSMP L<ENTER>

The possible options (i.e. to type instead of L) are:

C: start with COMPILER phase (i.e. skip the TRANSLATION phase)
L: start with LINKER phase (i.e. skip TRANSLATION and COMPILER phase)
E: start with EXECUTION phase (i.e. skip TRANSLATION, COMPILER and LINKER phase). This is equivalent to the command UPDATE

Option C is to be used when changes have been made in UPDATE.FOR or DATA.FOR.
Option L is to be used after UPDATE.FOR is changed and a new UPDATE.OB is created (with the commands: FOR1 UPDATE,,UPDATE;<ENTER> FOR2<ENTER>). The user does not have to re-compile DATA.FOR.
Option E is to be used when the user wants to run the program with oth parameter values or constants. The values of all this kind of input is stored in CONTRO.SYS. The user can change these valu (by using an editor or word processor) and can calculate the output with the new input, without having to wait for the translation, compilation and linking.

Be aware that each time that PCSMP is invoked (with whatever option), creates intermediate and output files that always have the same name. there are already files with these names on the subdirectory where PCS is invoked, the contents of these files will be replaced by the new contents. And there is no method of retrieving the old information oth than by invoking PCSMP with the original file.
Example:
If you invoke PCSMP L (after changing the value of a parameter in CONTRO.SYS), the existing FOR06.DAT, PREP1.DAT and PREP2.DAT will be overwritten. To avoid this, you should first rename these files if you want to keep the information in it.

4. MATHEMATICAL OPERATIONS AND FUNCTIONS IN PCSMP

4.1 Introduction

PCSMP uses the same arithmetic rules as FORTRAN (see Appendix A.1). In a PCSMP-program, all the standard FORTRAN functions can be used (see Appendix A.2). In addition, PCSMP has 'functional blocks' or functions, that are used for more complex mathematical operations, such as integration, time delay, limiting the value of a variable. The general form is:

```
Y = FNAME(X1,X2,X3)
```

Where Y is the outcome/result of the mathematical operation in function FNAME with input X1, X2 and X3.

The PCSMP functions are listed in Appendix A.3, and described in detail in the IBM CSMP manual. Of the PCSMP the following are available on the mainframe computer (i.e. with the full CSMP language), but **NOT** on the PC: REALPL, LEDLAG, CMPXPL, TRANSF, MODINT, PIPE, ARRAY, and SCALAR.

In general, the PCSMP functions can directly be used in user-made subroutines. There are however a few exceptions: some cannot be used (INTGRL, DERIV, DELAY, ZHOLD), some can be used, but additional information should be passed on through the subroutine (AFGEN, NLFGEN, FUNGEN, TWOVAR; see paragraph 4.3).

When in a user-made subroutine a PCSMP function is to be used, that has a name starting with I, J, K, L, M or N, but should give REAL ouput, be sure to declare the PCSMP function as a REAL. E.g.:

```
SUBROUTINE SUSELF(X1,X2, ....
REAL LIMIT
...
X1=LIMIT(X2,X3,X4)
...
RETURN
END
```

Most PCSMP functions are rather straightforward, and easy to use and to understand. In the next paragraphs two (groups) of PCSMP functions will be discussed that deserve a little more attention.

4.2 the INTGRL function

FORM:

```
     Y = INTGRL(IY, RY)        (for a single element Y)
```

or

```
     Y = INTGRL(IY, RY, N)     (for array Y consisting of N elements)
```

The latter form is equivalent to:

```
     Y(1) = INTGRL(IY(1),RY(1))
     Y(2) = INTGRL(IY(2),RY(2))
     ...
     Y(N) = INTGRL(IY(N),RY(N))
```

Where Y is output, IY is the initial condition of Y and RY is the rate of change of Y during the interval between TIME TIME+DELT.

ACTION:
The INTGRL performs a numerical integration of Y over TIME:

$$Y_{TIME+DELT} = Y_{TIME} + DELT * RY_{TIME}$$

REMARKS:
Take notice, that TIME is an internal variable of PCSMP, and that after every complete evaluation of the expressions in the DYNAMIC part of a PCSMP program, TIME is increased with DELT.
With the TIMER TIME statement, an initial value can be given to TIM

Especially in systems with a positive feedback, the value of DELT should be carefully chosen: when it is too large, a large relative error in the calculation of Y will result; if DELT is too small, ti is unnecessarily spend on calculations.
PCSMP has various integration methods. Some calculate at each time step the largest possible DELT by given maximally allowed absolute and relative error in Y, others use a fixed DELT, to be given a val by the user of the model via the TIMER DELT statement. A more thorough explanation of the various integration methods is in the I CSMP manual (page 69 and further; the method STIFF can only be used on the mainframe, but not on the PC). For most crop growth models, the integration method RECT (i.e. rectangular) should be used, whic uses a fixed DELT. The statement METHOD RECT therefore should be included in these models.

It is advisable, **NOT** to use the INTGRL as part of an mathematical expression as this can create problems due to the fact that PCSMP treats the INTGRL statements in the sorting process different from other expressions and functions. Therefore, instead of using:

```
     Z = X + INTGRL(IY, RY)
```

it is better to use:

```
     Y = INTGRL(IY, RY)
     Z = X + Y
```

The INTGRL can not be used in user-made subroutines, as it requires some additional data that is very specific to PCSMP.

4.3 the AFGEN, NLFGEN, FUNGEN, and TWOVAR functions

FORM:

Y = AFGEN(FNAME,X)

Y = NLFGEN(FNAME,X)

Y = FUNGEN(FNAME,N,X)

Y = TWOVAR(FNAME,X,Z)

Where: Y is output or dependent variable, FNAME is the name of the table where the data are stored, X and Z are input, or independent variables, and N is the degree of curve used for interpolation.

ACTION:
These functions generators (IBM CSMP manual, page 85-94) calculate the value of Y belonging to X by interpolation of the data in the FUNCTION FNAME, which is a set of data that the user provided to the program (see paragraph 5.2.4). The data is stored in pairs of X,Y or in trio's of X,Z,Y.

AFGEN	performs	linear interpolation for pairs of X,Y
NLFGEN	"	quadratic interpolation for pairs of X,Y
FUNGEN	"	interpolation with degree N (ranging from 1 to 5) for pairs of X,Y
TWOVAR	"	linear interpolation for trio's of X,Z,Y

REMARKS:
When using FUNGEN, the number of X,Y pairs that is provided via a FUNCTION, should be at least one more than N (the degree of interpolation). FUNGEN with N=1 is equivalent to AFGEN, FUNGEN with N=2 is equivalent to NLFGEN.

Higher order interpolation doesn't always give 'better' results, especially when the data shows a discontinuity or a large relative change (Figure 4.1).

When the input to a function generator is outside the range of input data in the function table, a warning is written in the FOR06.DAT. In such situations, extrapolation of the data is done instead of interpolation. This doesn't necessarily result in erroneous calculations, but the outcome of the extrapolation should be checked, or the data in the function table should be extended.

When using one or more of these functions inside a user made subroutine, in the definition of the subroutine the names of the function-table(s) and the word NLOC (or of another INTEGER variable that is not used anywhere else in the program) have to appear. Be sure NLOC appears as the first in the input/output list after the name of the subroutine, and is declared integer in the subroutine:

```
SUBROUTINE SUEXMP(NLOC,FNAME1,FNAME2,X1,X2,Y1,Y2,Y3)
IMPLICIT REAL(A-Z)
INTEGER NLOC

Y1=FUNGEN(NLOC,FNAME2,5,X1)

Y2=AFGEN(NLOC+10,FNAME1,X2)

Y3=AFGEN(NLOC+15,FNAME1,X1+X2)

RETURN
END
```

In this example, FNAME1 and FNAME2 are function tables that have t appear in the main program:

```
FUNCTION FNAME1= ....
FUNCTION FNAME2= ....
```

Observe that in the subroutine, the word NLOC appears right after AFGEN(or FUNGEN(, and that the second and third time some values a added to it. This needs to be done as PCSMP requires 5 memory allocations for each call for AFGEN, 10 for FUNGEN or NLFGEN, and 1 for TWOVAR (IBM CSMP manual, page 100).

In the main program, the name of the subroutines where a function generator is called, should appear after a HISTORY label, with an indication of the total memory requirements. The latter should be calculated as 5 * (the number of calls for AFGEN in the subroutine) 10 * (the number of FUNGEN or NLFGEN) + 12 * (the number of TWOVAR)

```
HISTORY SUEXMP(20)
```

(more subroutine names can follow a HISTORY label).

In the main program, the call for the subroutine should show the na of the function-tables that are used, but not the NLOC variable:

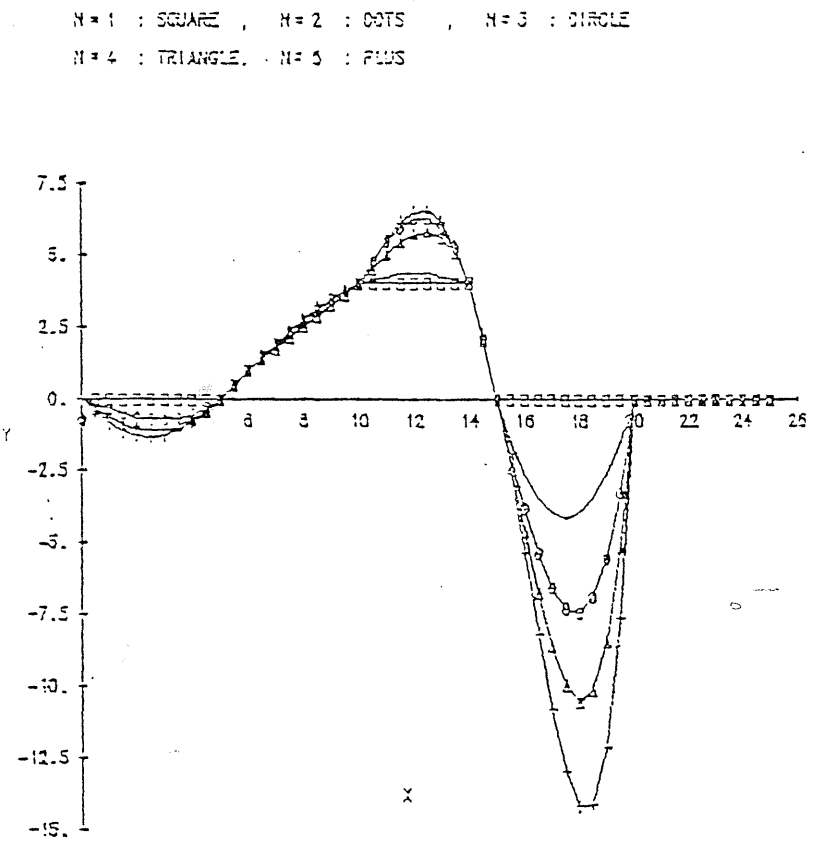
```
Y1,Y2,Y3=SUEXMP(FNAME1,FNAME2,X1,X2,Y1,Y2,Y3)
```

or in NOSORT sections:

```
CALL SUEXMP(FNAME1,FNAME2,X1,X2,Y1,Y2,Y3)
```

Some additional information can be found on pages 99-102 in the IBM CSMP manual.

Figure 4.1: Result of various degrees of interpolation of the same FUNCTION.
Where:
Y = FUNGEN(YFUTB,N,X)
FUNCTION YFUTB=-5.,0., 0.,0., 5.,0., 6.,1., 10.,4.,...
14.,4., 15.,0., 20.,0.



5. INPUT

5.1 INTRODUCTION

PCSMP programs contain three types of 'named symbols':

1. **commands** to run the program
2. **variables** that change during the dynamic phase of the program
3. **parameters** and **functions** that do not change during the dynamic phase of the program

When we talk about input, we refer mainly to the third group of symbols: those whose value does not change during the dynamic phase Secondly, also the initial values (if not zero) of some of the variables have to be input to the PCSMP program. A third type of input is needed to have the model run as it should (PCSMP-system input). In the following, when the word parameter(s) is used, reference is made to any type of input (the word PARAM however, is related to a specific type of input, that will be discussed later).

Before going into detail on the various types of parameters and how to address values to them, it is necessary to understand how information is stored in the computer.

The memory of the computer is divided into small parts. Each part c contain a limited amount of information. When running a PCSMP program, a certain part of the total memory (RAM) is claimed, depending on the size of the program. Each symbolic name (i.e. variable or parameter) is given some parts of that claimed memory. During the first part of the execution phase, even before the so called INITIAL phase (see paragraph 8 SORT/NOSORT), the contents of all the claimed memory is set at zero. This means that all the variables, parameters etc. have the value 0.

After this is done, the CONTRO.SYS file is read, and each parameter will be given the value given as written in the CONTRO.SYS file, which is made during translation phase with INPUT from the PCSMP program. Assume that in the program the following lines are found:

```
        WSH   =INTGRL(WSHI,GSH)
        WRT   =INTGRL(WRTI,GRT)
INCON WSHI=50., WRTI=50.
```

Then in CONTRO.SYS the following line will be found

```
INCON WSHI=50., WRTI=50.
```

Before CONTRO.SYS is read, WSHI AND WRTI will have the value 0., bu after reading this file, they will have the value 50. In other word WSHI AND WRTI are updated with the value 50.

On the monitor of the PC (but not on the monitor of a mainframe computer), it can be seen when updating of the parameters is completed. Assume that in the CONTRO.SYS the following TIMER statements occur:

```
TIMER FINTIM=100., DELT=1., PRDEL=5., OUTDEL=5.
```

Right after the start of the execution phase, the following line will be written on the screen:

```
TIME    .0000        FINTIM    .0000        DELT    .0000
```

When updating of parameters is over, this line will change to

```
TIME    .0000        FINTIM   100.00        DELT    1.000
```

Right after the line is changed, all the variables and all the parameters that were not given a value in CONTRO.SYS will still have the value 0. Only those that are found in CONTRO.SYS will get the value given to them in that file.

The next phase is the execution of the INITIAL section of the model (if it exists, otherwise immediately the next phase, the DYNAMIC section of the model, will start). All the equations that are found in this section are executed, using the values for the parameters as updated in the previous phase. All variables that are not updated before the moment they are needed in an equation, still will have the value 0. Only after they have been calculated (i.e. they appeared at the left side of the '=' sign in the INITIAL part of the model) they can have a value different from 0.

After the INITIAL phase, the execution moves to the DYNAMIC section of the program, in which the variables are updated each time step (DELT) according to the program lines. Parameters are not changed, unless the same name is given both to a parameter and to a variable:

```
INCON A=2.
PARAM C=5.
      B=C*A
      A=TIME/C
TIMER TIME=0.,FINTIM=100.,PRDEL=1.,OUTDEL=1.,DELT=1.
METHOD RECT
PRINT A,B,C
```

In this case, initially 'A' will be regarded as a parameter, and will be given the value 2., but it will get a different value every time the equations are evaluated (notice that two lines are repeated. This is not a typing error. In PCSMP, at the first time step, the complete model is evaluated twice. In this way the initial values for INTGRL statements can be calculated in the INITIAL phase such that the state variable has the right value at the start of the DYNAMIC part of the model):

MEMORY	SCREEN
A = 0. B = 0. C = 0.	TIME .000 FINTIM .000 DELT .000
A = 2. B = 0. C = 5.	TIME .000 FINTIM 100. DELT 1.00
A = .2 B = 1. C = 5.	TIME .000 FINTIM 100. DELT 1.00
A = .2 B = 1. C = 5.	TIME 1.00 FINTIM 100. DELT 1.00
A = .4 B = 2. C = 5.	TIME 2.00 FINTIM 100. DELT 1.00

etc.

In the following paragraphs the various types of input will be discussed.

5.2. PARAM, INCON and CONSTANT

use:
PARAM : to assign a value to a parameter
CONSTANT: to assign a value to a constant
INCON : to assign a value to an initial condition of a variable that is integrated in the program using the INTGRL statement.

Form:

```
PARAM PNAME1=1 , PNAME2 = 2.3
PARAM PNAME3=1.E7
CONSTANT CNAME1=-1.1 ,CNAME2= 2
INCON INAME1=.0001,INAME2=-2
```

Comments (see also the CSMP manual by IBM, page 37):
Integer and negative values are allowed; a comma is used to distinguish between to different parameters; blanks are allowed, except in the name and between the numbers of a value.
The labels PARAM, INCON and CONSTANT can be used indiscriminately. Use of different label names however makes it easier to recognize h a certain parameter is used in the model: it provides information t the user.

PARAM, CONSTANT and INCON can not be used to give values to an arra In that case, the label TABLE has to be used.

5.3. TABLE

Use:
assign values to parameters, constants, or initial conditions, that are DIMENSIONED.

Form:

```
TABLE TNAME1(1-5)=1., 2.,3. ,4., 5.
TABLE TNAME2(1-15)=15*1.
TABLE TNAME3(1-8)= 3*1., 0.5,77., 3*3.5
```

Comments: (see also page 38 and 42 of the IBM CSMP manual)
When using a TABLE, the parameter or initial condition after the TABLE statement, should be **DIMENSIONED** by using the **STORAGE** label:

```
STORAGE TNAME1(5), TNAME2(15), TNAME3(10)
```

The STORAGE label claims memory for the arrays. Instead of only one part of the memory (as for a symbol defined with the PARAM label), a number of memory parts is claimed. The actual number of claimed memory parts per array depends on the number that is typed between parenthesis after the array name in the STORAGE label. In the example above, 5 places are allocated to array TNAME1; 15 to TNAME2 and 10 to TNAME3. Note that in TABLE TNAME3(1-8) only 8 of the claimed memory parts are given a value. You can always claim more memory parts than you actually use (the parts not used will store the value 0), but you should **NEVER** claim less. If you do, calculation errors might occur, as the program takes the contents of other memory parts and uses them as if they were of the array you wished to use. Mostly, no error message is given, and if an error message appears in FOR03.DAT, FOR06.DAT, or UPDATE.LST, it is not always clear that it is caused by a incorrect dimensioning.

The TABLE label is also used to assign values to the initial values of INTGRL arrays:

```
WLV = INTGRL(WLVI,GLV,5)
TABLE WLVI(1-5)=5*0.4
```

These two lines are equivalent to:

```
WLV(1) = INTGRL(WLVI(1),GLV(1))
WLV(2) = INTGRL(WLVI(2),GLV(2))
WLV(3) = INTGRL(WLVI(3),GLV(3))
WLV(4) = INTGRL(WLVI(4),GLV(4))
WLV(5) = INTGRL(WLVI(5),GLV(5))
PARAM WLVI(1)=0.4,WLVI(2)=0.4,WLVI(3)=0.4,...
      WLVI(4)=0.4,WLVI(5)=0.4
```

When an INTGRL array is used, the STORAGE label need not to be used to reserve memory allocations for the state, the rate and the initial conditions. In the example above: WLV(5), GLV(5) and WLVI(5) should not put after the STORAGE label. This is due to the fact that PCSMP takes care of this when an array is integrated.

5.4. FUNCTION

Use:
The FUNCTION label is used for specification of pairs of x and y o
of trios of x, y and z. These coordinates are used by so called
FUNCTION GENERATORS (Paragraph 4.3).

Form:
a. Pairs of x and y:

```
FUNCTION FNAME1 = (X11,Y11), (X12,Y12), (X13,Y13),    etc
FUNCTION FNAME2 = X11,Y11, X12,Y12, X13,Y13,          etc
```

b. Trios of x, y and z:

```
FUNCTION FNAME3,Z1 = (X11,Y11), X12,Y12, (X13,Y13),   etc
FUNCTION FNAME3,Z2 = (X21,Y21), (X22,Y22), X23,Y23,   etc
FUNCTION FNAME3,Z3 = X31,Y31, (X32,Y32), (X33,Y33),   etc
```

Where:
Xij is the first independent variable at Zi
Zi is the second independent variable
Yij is the dependent variable at Xij and Zi

Comments: (see also IBM CSMP manual, page 38)
Parentheses and blanks are optional, but the X-Y pairs should be separated from each other by a comma, as should be the X from the Y in each pair.

The value of the first independent variable (X) should increase fro the leftmost pair onwards (i.e. X11 should have a lower value than X12, and X12 a lower value than X13). The second independent variab (Z) should increase from the top downward (Z1 should be less than Z Z3 should be larger than Z2).

The name of the FUNCTION table (in these examples FNAME1, FNAME2 an FNAME3) should not start with a character of the following list: I, J, K, L, M, N.
If the name of a FUNCTION table starts with one of these characters strange errors might occur, that can not be traced easily.

5.5. Input to the EXECUTION CONTROL STATEMENTS

Use:
Apart from data to run the program that you made, some information required so that the program will be evaluated in the way that you want (see also page 48.2 in the IBM CSMP manual).

Form:

```
TITLE DRY MATTER PRODUCTION
TIMER TIME=0., FINTIM=100., DELT=1., PRDEL=5., OUTDEL=5.
METHOD RECT
FINISH DS=2.
END
STOP
ENDJOB
```

Comments:

TITLE: (page 55 in the IBM CSMP manual). Each page of the output will be start with the text after TITLE. More than one TITLE can appear in a program. The first character of the text should be a capital.

TIMER: (page 48.2-49 in the IBM CSMP manual). PCSMP has an internal TIME during execution. The start of simulation can be indicated with the TIME after the TIMER. Instead of the value 0., any value can be used, as long as it is not larger than FINTIM, which indicates the TIME at which the program should stop. DELT sets the value of the time step used in the model to a certain value (in the example above, DELT will have the value 1.). PRDEL and OUTDEL are used to set the interval of preparing output. With 5 as the value of PRDEL and OUTDEL, at each TIME equal to n*5, the value of the variables after the PRINT, OUTPUT, and PRTPLOT labels are written to the FOR06.DAT (see the example of figure 17, page 53 in Penning de Vries and Van Laar, 1982). PRDEL is used for the PRINT label, and OUTDEL for the OUTPUT, PRTPLOT and PREPARE labels. PRDEL and OUTDEL can have different values (e.g PRDEL=1., OUTDEL=10.)

METHOD: (page 50 in the IBM CSMP manual, and paragraph 4.2) This label indicates the type of integration method that is used. METHOD STIFF can only be used on the mainframe, but not on the PC.

FINISH: (page 49 of the IBM CSMP manual). Sometimes evaluation of the model should stop when a certain condition is reached (e.g. if the crop is mature). The FINISH statement can be used to indicate that the model must stop if a variable equals a certain value, or if the difference between the value and a certain value changes sign. The form is: FINISH X=Y, where X is the variable, and Y is the value of X where the program has to stop.

END: (page 51 in the IBM CSMP manual). The END statement is used to indicate the end of a particular case or run. The first END statement should be put AFTER the whole of the structure of the program and the first complete set of the DATA input statements. After one END statement and before another END statement, changes in DATA input and some OUTPUT CONTROL statements can be made (See paragraph 7 RERUNS).

STOP: (page 45 of the IBM CSMP manual) This statement indicates the end of the PCSMP part of the model. After STOP, and before ENDJOB, you can write FORTRAN subroutines.

ENDJOB: (page 45 of the IBM CSMP manual) This statement indicates the end of the complete model.

Comments:

All these STATEMENTS (except STOP and ENDJOB) if used in the model, will be found in the CONTRO.SYS. The values and words that are input to these statements can be changed in the CONTRO.SYS (see paragraph 7 RERUNS).

5.6. Reading data from an external file

In some cases, reading data from another file might facilitate you work. This should be done using FORTRAN statements for opening and reading a file. It will not be discussed here.

6. OUTPUT

6.1 INTRODUCTION

Usually after a model is run, you want to see the outcome of the calculations. In general you don't want to see the results of all variables that are calculated. Also you might want to see the outc of some variables only once every so many time steps. Some output would like to get in tables, and of other output you would like to get a figure. PCSMP therefore has some options to create output of (limited) list of variables, some options regarding the form of the output, and options to modify the timing of output. The latter options are discussed in paragraph 5.5 EXECUTION CONTROL STATEMENTS the PRDEL and OUTDEL statements. Here therefore only the commands discussed that enable you to make a list of the output and to defi the form of the output:

```
PRINT TWT,WSH,WRT,GTW
OUTPUT TWT
PRTPLOT WSH,WRT
PREPARE WSH,WRT
PAGE MERGE,GROUP
```

All these statements appear in the CONTRO.SYS, and the user can change them in this file before execution of UPDATE.EXE (paragraph RERUN).

6.2 PRINT

(page 55 in the IBM CSMP manual, and Figure 17 in Penning de Vries and Van Laar, 1982)
The variables after the PRINT label will be put in the FOR06.DAT fi in TABLES versus time. Only 55 variables can be PRINTed. If there two or more PRINT statements used in the program, the LAST one wil be used.

6.3 OUTPUT and PRTPLOT

(page 57 in the IBM CSMP manual)
If 5 or less variables are listed after the OUTPUT or PRTPLOT statement, a figure is made in the FOR06.DAT (see the example of figure 17 in Penning de Vries and Van Laar, 1982). If more than 5 variables are listed, OUTPUT or PRTPLOT gives the same result as PRINT. The maximum number of variables after each OUTPUT or PRTPLO is 55, as is the case with PRINT. However unlike PRINT, more than

OUTPUT or PRTPLOT statement can be used in the model: all variables will be put out in figures or in tables.

6.4 PREPARE

(page 67 of the IBM CSMP manual)
When you want to make use of the PLOTCSM program to 'plot' your output on screen or on the printer, this statement should be used. Apart from the variables listed after PREPARE, also those after OUTPUT and PRTPLOT will be prepared for plotting, and written into the files PREP1.DAT and PREP2.DAT. PLOTCSM can handle only 25 variables (including those in the OUTPUT and PRTPLOT statements) and only 50 reruns. If more variables or reruns are used, no error message will be written to the screen or the FOR06.DAT, but PLOTCSM will not function properly.

6.5 PAGE

(page 63-66 of the IBM CSMP manual)
This statement is used to modify the figures produced by the OUTPUT and PRTPLOT labels. Many options are listed in the IBM CSMP manual. Of these, the GROUP and MERGE option are mostly used. GROUP indicates that a figure with one scale for all (or a group of) the variables should be produced. With MERGE (and the use of END RERUN, see paragraph 7 RERUNS) a figure can be made containing the values of one variable in various reruns. Of the PAGE options mentioned in the IBM CSMP manual, the WIDTH option does not work properly in PCSMP.

7. RERUNS

7.1 INTRODUCTION

If you have many years of weather data, or you are interested in th effect of the value of a certain parameter on the results of the model, you might want to calculate the outcome of the same model wi different values as input. Of course you can do this by running the model first with one set of input. Then you change the values, and translate, compile and execute the model again. This however, is no very efficient. The translation and compilation takes a lot of time Luckily, PCSMP provides some ways of making RERUNS more efficient, most of these will be discussed in the following paragraphs.

7.2 Using the MULTIPLE VALUE PARAMETER

Use: (page 120 in the IBM CSMP manual) When the only change you war to make to the input is in the value of ONE parameter.

Form:

```
PARAM PNAME1=(1.,2.,3.,4.,5.)
```

Comments:
Apart from PARAM, also INCON and CONSTANT can be used as above. In the example above, 5 reruns will be made with parameter PNAME havin first the value 1., then 2., etc.. If after this line the following line is included:

```
PARAM PNAME2=(10.,11.,12.)
```

then parameter PNAME1 will always have the value 1., and only 3 reruns are made, with parameter PNAME2 having the values 10., 11., and 12.

When using the PREPARE statement to obtain output for PLOTCSM, the MULTIPLE VALUE PARAMETER cannot be used to make reruns.

7.3. Using the END command

Use: (page 51 and 121 in the IBM CSMP manual).
This statement can be used if reruns have to be made with changes i more than one PARAM, CONSTANT or INCON, or in the values of a FUNCTION, or in the input to the TIMER, PRINT, OUTPUT or PREPARE statements. Also when a combination of the changes mentioned above required.

Form:

```
PARAM PNAME1=(1.,2.,3.,4.,5.)
PARAM PNAME2=10.
PARAM PNAME3=20.
END
PARAM PNAME2=11.
END
PARAM PNAME2=12.
PARAM PNAME3=2.
END
PARAM PNAME2=13.
END
STOP
```

Comments:
There should always be an END statement before the STOP statement. In addition to using the END, also the MULTIPLE VALUE PARAMETER can be used (paragraph 8.2). In the example above, for each rerun that precedes an END statement, 5 reruns are made with PNAME1 ranging from 1. to 5. In total, 4 * 5 = 20 reruns will be made if the statements above appear in a program.
Note that a PARAM (and all the other INPUT) keeps the value that is given to it during the last initialisation. In the example above, parameter PNAME3 will have the value 20. in the first 2*5 reruns, but it has the value 2. in the next 2*5 reruns.
In special cases, END RERUN is used. However, when you want to use PLOTCSM for plotting your results on screen or printer, the END RERUN statement CANNOT be used. Therefore, it is suggested to use only the END statement.

7.4 Using the CONTRO.SYS file

Use: As stated before, all the INPUT is written into the CONTRO.SYS file. When PCSMP is used with the options S or D (Paragraph 3.2), this file will remain on your directory. The values of parameters, functions, or tables can be changed, as can the input to the FINISH, PRINT, OUTDEL, PRTPLOT, and PREPARE statements. Execution with the updated CONTRO.SYS is invoked by typing UPDATE.

Comments:
When doing so, the output of the former run in the FOR06.DAT, and the PREP1.DAT and PREP2.DAT files, will be deleted and replaced by the values of the outcome for the changed input. If you want to keep the old output, you will have to rename the FOR06.DAT, the PREP1.DAT and the PREP2.DAT files. It is NOT possible to make a plot of the outcome of two reruns made by changing the CONTRO.SYS, as PLOTCSM can read only one PREP1.DAT and one PREP2.DAT at the time. Simply copying several PREP1.DAT or PREP2.DAT files together, will result in errors during running of PLOTCSM.

8. SORT and NOSORT in PCSMP

8.1 INTRODUCTION

SORTING is grouping the equations or lines of the program in a logical order, such that a variable is calculated before it is usec as input in another equation or line.

Assume a PCSMP program such as that of Figure 17 in Penning de Vrie and Van Laar (1986, page 53):

```
 1   TITLE DRY MATTER PRODUCTION
 2         TWT   =WSH+WRT
 3         WSH   =INTGRL(WSHI,GSH)
 4         WRT   =INTGRL(WRTI,GRT)
 5   INCON WSHI=50., WRTI=50.
 6         GSH   =0.7*GTW
 7         GRT   =0.3*GTW
 8         GTW   =(GPHOT-MAINT)*CVF
 9         MAINT =(WSH+WRT)*0.015
10         GPHOT =GPHST*(1.-EXP(-.7*LAI))
11         LAI   =AMIN1(WSH/500.,5.)
12   PARAM CVF=.7, GPHST=400.
```

This program is not sorted, as the variable LAI is used as input ir line 10, whereas it is calculated only in line 11.
A sorted version of the equations in this program would look like:

```
WSH =INTGRL(WSHI,GSH)
WRT =INTGRL(WRTI,GRT)
LAI=AMIN1(WSH/500.,5.)
GPHOT=GPHST*(1.-EXP(-.7*LAI))
MAINT=(WSH+WRT)*0.015
GTW=(GPHOT-MAINT)*CVF
GSH=0.7*GTW
GRT=0.3*GTW
TWT=WSH+WRT
```

This sorting of the equations in fact, takes place during translati of PCSMP to FORTRAN, because in this latter computer language only SORTED programs can be executed.
The advantages of PCSMP, where the user does not have to do the sorting him or herself are:
1. it is easy to program: lines can be put in any order
2. easy grouping of program lines to get nice lay-out

Of course there are also disadvantages:
1. it is impossible to use equations where the same variable occurs at both sides of the = sign.
2. the calculation order is sometimes hard to trace, and sometimes not as the user would like it to be. This especially occurs wher

variable has to be used first at its 'old' value and later in the same timestep, at its 'updated' value.

3. it is impossible to refer to labeled lines, because it is not clear where they will be found in the program.
4. sometimes algebraic loops occur in the program, when the value of a variable has to be known before it can be updated. An example of such an error is given by the following PCSMP program:

```
TITLE TEST 2
      A=B*C
      C=D+E
      F=A/C
      B=F+G
PARAM D=2.,E=3., G=1.5
PRINT A,B,C,D,E,F,G
TIMER TIME=1.,FINTIM=2.,DELT=1.,PRDEL=1.,OUTDEL=1.
END
STOP
ENDJOB
```

Sorting of this program can yield two sets of equations:

```
1 C=D+E                      1 C=D+E
2 A=B*C                      2 B=F+G
3 F=A/C          or:         3 A=B*C
4 B=F+G                      4 F=A/C
```

So either B (line 2 left) or F (line 2 right) has to be known before the whole set of equations can be calculated.

To overcome these disadvantages, one might want to have at least one part of the program not to be sorted by the computer. Still other parts of the PCSMP program should be sorted by the computer, and there should be some relation between the SORT and the NOSORT parts of the model.
PCSMP has some possibilities to do this, with the statement sets

1. **INITIAL-DYNAMIC-TERMINAL**
2. **NOSORT-SORT**
3. **PROCEDURE-ENDPROCEDURE**
4. **SUBROUTINE-END**

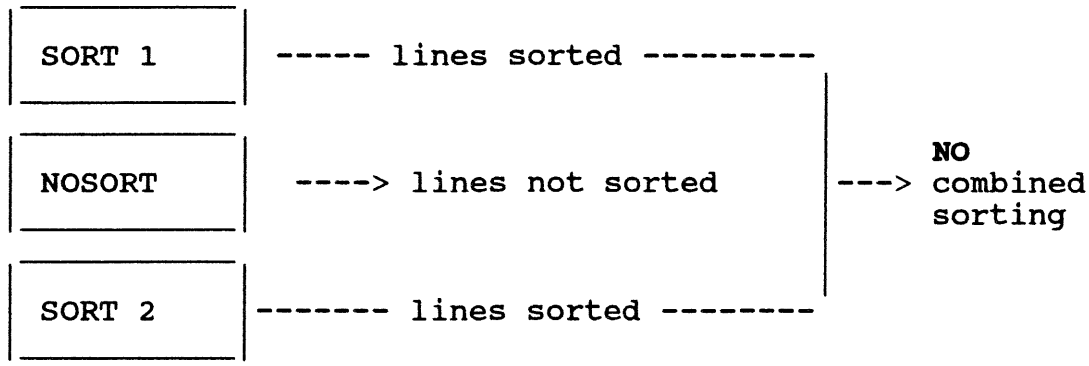
8.2. INITIAL - DYNAMIC - TERMINAL

(pp. 43-44 in the IBM CSMP manual)
In the INITIAL, lines are executed before the DYNAMIC is updated for the first time. Especially when PROCEDURE statements are used in the DYNAMIC, giving variables their initial value in the INITIAL can be helpful in directing the order of calculation. (see section about PROCEDURES). After calculating till a finish condition or FINTIM is reached, a TERMINAL can be invoked, e.g. to calculate the Harvest Index of the crop.

INITIAL	executed first (once only; before all the other calculations)
DYNAMIC	executed during each time step from start till end
TERMINAL	executed last (once only; after all the other calculations)

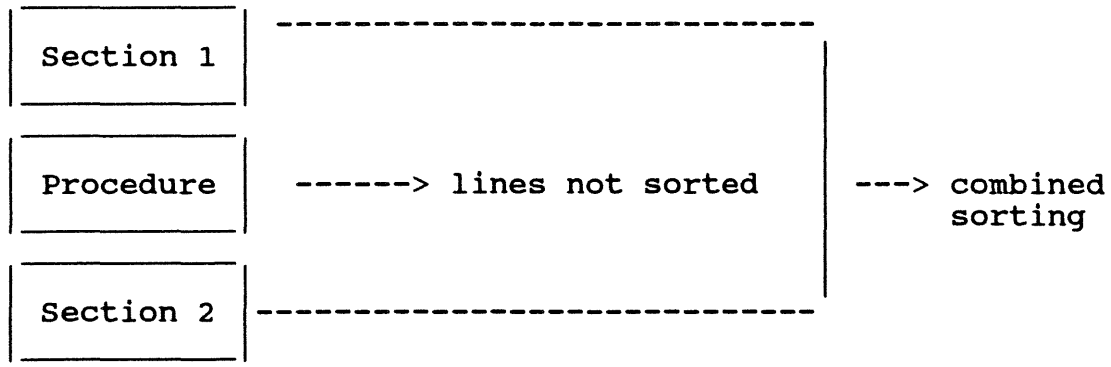
8.3. NOSORT - SORT

(p. 44 in the IBM CSMP manual)
All the lines between the statements NOSORT and SORT (in the following called a NOSORT section, in contrast to a SORT section) will not be sorted by the computer, but will be executed in the ord they were written by the user.
However, these statements are better not used, as they split the program into sections. If one SORT section is found before and one after a NOSORT section, they will be sorted only internally, but th lines of the two sections will not be sorted together:



8.4. PROCEDURE - ENDPROCEDURE

(pp. 95-97 in the IBM CSMP manual)
As with the NOSORT-SORT statement, lines between PROCEDURE and ENDPROCEDURE (hereafter called procedure) are not sorted by the computer, but executed in the order as provided by the user.
However, unlike the NOSORT block, all the other lines of the progra will be sorted together:



The procedure therefore has a clear advantage over the NOSORT-SORT statement. However, the form in which it appears is more complicated: a procedure should provide a list of output variables, that are calculated inside the procedure; it should have a name; and it should have a list of input variables that have to be calculated before the procedure can be used e.g.:

```
PROCED OUT1,OUT2,OUT3 = PRNAME(IN1,IN2,IN3)
       INTER= OUT2 * IN3
       OUT1 = INTER * IN2
       OUT3 = OUT1 * IN4
       OUT2 = OUT1 * OUT3
ENDPRO
```

In this procedure, OUT2 is used in the first line to calculate INTER (which is only used in the procedure, and therefore not put on the input or output list). OUT2 is not on the input list, so in the procedure its 'old' value will be used. In an INITIAL section, OUT2 could be given an initial value. IN4 is not put in the input list, but is used as an input in the procedure. This is possible, because all the variable names of the program are known in the procedure. Only if IN4 was updated before the procedure was evaluated, in the procedure the right value is used for IN4. This will always be the case when IN4 is a parameter (PARAM, INCON, CONSTANT, TABLE, FUNCTION), but not necessarily when IN4 is a variable.

PCSMP sorts the procedure according to the lists of inputs and outputs of it: first all the inputs will be calculated, and all the lines in the procedure are calculated before the line in which for the first time in the program, one of the outputs of the procedure is needed. The input list may not contain symbolic names that also occur in the output list. In fact, PCSMP treats the procedure as one equation: OUT1,OUT2,OUT3=f(IN1,IN2,IN3).

Any name may be given to the procedure, except 'reserved words', and names with illegal characters or starting with an non alphabetic character. The name should be used only once, thus no parameter or variable should have the same name. The amount of characters may not be more than six.

8.5. SUBROUTINE - END

(pp 98 -99 in the IBM CSMP manual)
In fact, these statements are similar to the PROCEDURE - ENDPROCEDU statements. However, unlike the procedure, by the computer it is nc considered to be part of the program itself, but merely something that the computer has to do before it can continue with the executi of the program.
It is like when delivering a speech, a person has to breath once in while. Taking breath is not part of the speech, but it has to be done, before the speaker can go on.

How to use the SUBROUTINE - END statements in PCSMP?

In the actual PCSMP program (also called the MAIN part of the program), a line is written with a list of outputs, a SUBROUTINE name; and a list of inputs:

```
OUT1,OUT2 = SUNAME(IN1,IN2,IN3)
```

In NOSORT sections or in procedures also the following line can be used:

```
CALL SUNAME(IN1,IN2,IN3,OUT1,OUT2)
```

After STOP, and before ENDJOB (at the end of the MAIN program), the lines of the SUBROUTINE are written. If more than one SUBROUTINE is needed, they all have to be placed before the ENDJOB statement:

```
STOP
      SUBROUTINE SUNAME(IN1,IN2,IN3,OUT1,OUT2)

            set of FORTRAN equations with IN1,IN2 and IN3 as
            input and OUT1 and OUT2 as output

      RETURN
      END

      SUBROUTINE SUNAM1(IN4,IN5,OUT3,OUT4,OUT5,OUT6)

            set of FORTRAN equations with IN4, and IN5 as inpu
            and OUT3, OUT4, OUT5 and OUT6 as output

      RETURN
      END
ENDJOB
```

Before each line of the subroutine, always six blanks have to be inserted (except when using digits to label certain lines, or when using a C at the leftmost side of a line, to indicate that the line contains comments).

At the end of the subroutine, always the statements RETURN and END have to be used. RETURN brings the program back to the line after tl

call for the subroutine was found, and END to indicate the end of the subroutine.
When using a subroutine, one should never forget that only the variables used in the input and output lists will be shared by the MAIN program and the subroutine. All the other variables and data of the MAIN are not know in the subroutine, and all the internal variables calculated in the subroutine are not known in the MAIN program (except when COMMON blocks are used, but these will not be discussed here; the reader is referred to a FORTRAN manual). This is a big difference with the PROCEDURE, that shares all the variables and parameters with the main program.

A special subroutine is the one in which only one variable is calculated. In that case, a FUNCTION should be used:

In the PCSMP program (the MAIN):

```
A = FUNAME(IN1,IN2,IN3)
```

Between STOP and ENDJOB as for SUBROUTINES:

```
STOP
      FUNCTION FUNAME(IN1,IN2,IN3)

            set of FORTRAN equations

      FUNAME=function of IN1,IN2,IN3
      RETURN
      END
ENDJOB
```

It is advisable that the variable FUNAME inside the FUNCTION is not used as input for an equation or evaluation, as a wrong value for FUNAME will be used. If the outcome of a function should be real, the FUNAME should not start with a character ranging from I - N, or the function should be declared REAL:

REAL FUNCTION IFUNC(.....) instead of **FUNCTION IFUNC(....)**

9. ERROR MESSAGES, DEBUG

9.1 Introduction

Three types of errors can occur after invoking PCSMP. First, PCSMP might not be invoked properly (e.g. because some files are lacking) secondly, the CPU might not be suited for running PCSMP, and thirdl errors might occur in the program that you would like to run.

When the CPU is not suited for PCSMP, often caused by a RAM that is too small, the error messages mostly appear during running of a program, and they are not easy to distinguish from error messages that occur because of errors in the simulation program, or because wrong installation of PCSMP. Therefore discussion of the errors caused by a CPU that is not suited for PCSMP are merged with the discussion of the errors caused by either a wrong installation of PCSMP or by errors in the model.

To detect whether enough RAM is available the following should be done:

After running the DOS command CHKDSK/F you might get on the screen something like (the numbers may be different):

```
8 lost clusters found in 8 chains
Convert lost chains to files (Y/N)?
```

After typing **Y**(es) to this question, the next lines appear on the screen (again: the numbers can be different):

```
21309440 bytes total disk space
   45056 bytes in 2 hidden files
   12288 bytes in 4 directories
 2689024 bytes in 91 user files
   16384 bytes in 8 recovered files
18546688 bytes available on disk

  655360 bytes total memory
  587392 bytes free
```

The last line is the important one in our case. If the number of fr bytes is less than 500000, you might expect problems with PCSMP. Yo should increase this number by buying RAM extension, or by preventi other programs to use part of the RAM. Ask your computer specialist how.

9.2 ERRORS DUE TO INCORRECT INSTALLATION OF PCSMP

In general: if it is not clear during which phase the problem occurs, type the required commands manually (as is explained in par. 2.).

I. **Problem:** After typing PCSMP the message **'bad command or file not found'** appears on the screen.
Cause & Solution: Most probably in C:\AUTOEXEC.BAT no PATH is set to C:\PCSMP and C:\FORTRAN. Add the following to the PATH statement in the C:\AUTOEXEC.BAT: ;C:\PCSMP;C:\FORTRAN. After changing the AUTOEXEC.BAT, reboot the system. If the message still appears after changing the AUTOEXEC.BAT and rebooting, check whether on C:\PCSMP and C:\FORTRAN all the required files are present (paragraph 1.4).

II **Problem:** When running a PCSMP file, an error occurs and the PCSMP.BAT stops before the execution phase.
Cause & Solution: Check whether the required files are found in C:\PCSMP and C:\FORTRAN. If not, do as is indicated in paragraph 1.4. If all the files are there, check the PATH statement in C:\AUTOEXEC.BAT (as described above). If the PATH is set, probably a PCSMP or a FORTRAN error occurred in the program (see paragraph 9.3). If no PCSMP error or FORTRAN error occurred, but still compilation was not completed successfully, the UPDATE.FOR might be too large for the FORTRAN compiler to be handled (more than 64 KBytes). If this is the case, make the main part of the program as small as possible by putting parts of the model in subroutines and compile each subroutine individually. Consult the FORTRAN manual (key word: lib or library) to make a subroutine library called SUBR.LIB on directory C:\MACROS\SUBR. If SUBR.LIB already exists on that subdirectory, add the object modules of the new subroutines. After doing so, invoke PCSMP with option D. When a large program is invoked with PCSMP option D, the resulting UPDATE.EXE might be too large; restart PCSMP but now with option S.

9.3 ERRORS DURING RUNNING OF A PCSMP PROGRAM

9.3.1 Introduction

PCSMP involves four phases:
1. TRANSLATION of a PCSMP program into a FORTRAN program
2. COMPILATION of the FORTRAN program into MACHINE CODE
3. LINKING of the MACHINE CODE program with PCSMP and FORTRAN subroutines (sometimes also with user supplied subroutines)
4. EXECUTION of the complete program

In **each** of these four phases errors can be detected that are due to wrong statements in the simulation program:

9.3.2 ERRORS during TRANSLATION of PCSMP into FORTRAN

9.3.2.1 INTRODUCTION

During TRANSLATION three files are created on basis of the original PCSMP program:

1. UPDATE.FOR
2. DATA.FOR
3. FOR03.DAT

A message is written to the screen when ERRORS are detected **DURING** translation:

?* PROBLEM CAN NOT BE EXECUTED**
Error during Translator Phase

HOW TO KNOW WHAT WENT WRONG?

In the file **FOR03.DAT**, a remark is placed **AFTER** each error (see the IBM CSMP manual page 154-162 for a complete list of error messages)

The error message is preceded by *** and often a $ is placed **under** the character where the error is located, e.g.:

```
AI = 0.5 */ BI
          $
*** ILLEGAL CHARACTER or DOUBLE OPERATOR
```

WHAT KIND OF ERRORS CAN BE EXPECTED DURING TRANSLATION?

1. Incorrect written statements
2. Illegal names of variables or functions
3. Wrong order of statements
4. Data input not correct
5. Dimensioning of variables is incorrect
6. Program is too big

9.3.2.2 INCORRECT WRITTEN STATEMENT

a. Too little or too many 'operators, i.e.: * / = , . + -) (

```
A = B *( C + D
        $
*** TOO MANY LEFT PARENTHESES
```

or

```
*** ILLEGAL CHARACTER or DOUBLE OPERATOR
```

b. An equation is split in more than 1 line without the continuatio 'card' (which consists of three dots ...).

```
*** CSMP STATEMENT INCORRECTLY WRITTEN
```

c. A statement is continued over too many line. E.g.:

```
      FIXED ...
*** TOO MANY CONTINUATION CARDS
```

d. A mathematical expression occurs after the following PCSMP labels: PARAM, INCON, FIXED, FUNCTION, TABLE, STORAGE, METHOD, TIMER.

```
*** CSMP STATEMENT INCORRECTLY WRITTEN
```

e. The output name of an equation is the same as one of the input names (only in a NOSORT block):

```
A = A + B
    $
*** INPUT NAME IS SAME AS OUTPUT NAME
```

9.3.2.3 ILLEGAL NAMES

a. At least one of the following illegal characters is used in a name: a 'blank', or + / ( ) $ = , * ' " ? ! # @ % ^ |

```
*** ILLEGAL CHARACTER OR DOUBLE OPERATOR
```

b. A name does not start with alphabetic character (A-Z)

```
*** SYMBOLIC NAME INCORRECTLY WRITTEN
```

c. A name has more than **six** characters:

```
*** SYMBOLIC NAME EXCEEDS SIX CHARACTERS
```

d. Lower case characters are used in a name instead of UPPER case:

```
X = xtb(IDATE)
    $
*** ILLEGAL CHARACTER OR DOUBLE OPERATOR
```

9.3.2.4 WRONG ORDER OF STATEMENTS

a. INITIAL, DYNAMIC, TERMINAL are not placed in the correct order

```
*** CSMP STATEMENT OUT OF SEQUENCE
```

b. The same variable name occurs more than once as an output in the same program part:

```
     A = B * C
     A = D /F
     $
*** OUTPUT NAME HAS ALREADY BEEN SPECIFIED
```

c. The STOP statement has been omitted; after ENDJOB the message will be:

```
*** CSMP STATEMENT INCORRECTLY WRITTEN
```

d. Statements are used in a NOSORT section, that only can be used in a SORT section: IF; GOTO; DO; WRITE; READ. This message can also appear in lines belonging to a IF or DO that contained errors

```
*** CSMP STATEMENT INCORRECTLY WRITTEN
```

9.3.2.5 DATA INPUT NOT CORRECT

a. The letter O is typed instead of the number 0 after declaration a PARAM, INCON, FUNCTION, TABLE:

```
*** CSMP STATEMENT INCORRECTLY WRITTEN
```

9.3.2.6 DIMENSIONING OF VARIABLES IS INCORRECT

a. A variable array that appears at the left side of and '=' sign i NOT dimensioned by a STORAGE statement:

```
STORAGE A(10)
A(IDATE) = DATE
B(IDATE) = DATE/2.
*** CSMP STATEMENT INCORRECTLY WRITTEN
```

9.3.2.7 PROGRAM IS TOO BIG

A PCSMP program can only be extended to certain limits. If it is to large, the following message appears in the FOR03.DAT:

```
*** PROBLEM INPUT EXCEEDS TRANSLATION TABLE nn
```

Where nn refers to the table on page 160 of the IBM-CSMP manual.

9.3.3. ERRORS DURING COMPILATION OF UPDATE.FOR

9.3.3.1 INTRODUCTION

When a PCSMP program is translated, two types of program parts are found: lines that should follow the PCSMP rules and lines that shou follow the FORTRAN rules (see paragraph 8. SORT/NOSORT).

During the translation phase, the lines that should follow the PCSM rules are translated correctly in FORTRAN (when no PCSMP error was detected; par. 9.3.2). The only errors that still might exist when the COMPILATION phase starts, are therefore FORTRAN errors in the FORTRAN parts of PCSMP. This can be in a PROCEDURE, a NOSORT block in a SUBROUTINE.

9.3.3.2 HOW TO FIND A FORTRAN ERROR

When the PCSMP.BAT file is invoked with the D(ebug) option, the fil UPDATE.LST will be made. In this file, errors are followed by

```
**** Error nnnn
```

Where nnnn is the Error number found in Appendix A of the IBM FORTR manual (pages A5-A11).

9.3.3.3. WHAT KIND OF FORTRAN ERRORS CAN BE EXPECTED?

Many FORTRAN errors can be found (including errors in mathematical expressions). In general, in NOSORT blocks, PROCEDUREs and SUBROUTINEs errors are found with DIMENSIONed variables (check the STORAGE statements), or with IF statements.

A special error can occur when UPDATE.FOR is larger than 64 Kbytes. The FORTRAN compiler can not handle such a big file and will stop before creating UPDATE.OBJ. The only way to escape this problem is by reducing the size of the UPDATE.FOR (paragraph 9.2).

9.3.4. ERRORS DURING LINKING

9.3.4.1 INTRODUCTION

During the LINKing phase, in the PCSMP and FORTRAN library (and if available, also in the library made by the user), those subroutines are located that are used in the PCSMP program, but that are not typed between the STOP and ENDJOB statement. These subroutines are then combined with the compiled version of the PCSMP program. The complete program is called the UPDATE.EXE.

9.3.4.2 HOW TO FIND THE ERRORS DURING THE LINKING PHASE?

When the PCSMP.BAT is used with the D(ebug) option, a file UPDATE.MAP is made. When an error is detected during linking, the following message will appear on the screen (when no floppy is in drive A)

```
Not ready error reading drive A
Abort, Retry, Ignore
```

If you type I(gnore), or when a floppy is in drive A:, the following will be typed on the screen:

```
Cannot find library A:FORTRAN.LIB
Enter new file spec.:
```

If you push the ENTER key, the program will continue with the LINKing and eventually start the EXECUTION. Stop this execution by pushing CTRL-C. In FOR06.DAT in general no errors can be detected. But in UPDATE.MAP, at the end of the file, the following message is written:

```
Unresolved externals

'name' in files(s)
   UPDATE.OBJ (UPDAT_)
```

where 'name' is the subroutine or array variable that was missing.

9.3.4.3 WHAT KIND OF ERRORS TO EXPECT DURING LINKING?

In general only two types of errors can be found:
a. one or more subroutines are invoked in the PCSMP program but are not in the EXECUT.LIB, nor in the EMULATOR.LIB (or 8087ONLY.LIB, or the FORTRAN.LIB), nor in the user made subroutine library (or your PC by default known as C:\MACROS\SUBR\SUBR.LIB). And they a not found between the STOP and ENDJOB statements. Check if two c more ENDJOB statements appear in the program, one placed before the definition of the subroutines that are missing.
b. in a FORTRAN part of the PCSMP program an array variable is used as input to a function (i.e. after the '=' sign), but it is not dimensioned with the STORAGE statement (or in subroutines with t DIMENSION statement).

9.3.5. ERRORS DURING EXECUTION

9.3.5.1 INTRODUCTION

Only during the EXECUTION phase, all the program lines will be evaluated, i.e. the output will be calculated. As all the input, program lines etc. at this stage is checked most thoroughly, still some errors can be detected. And many different types of errors can be found.

9.3.5.2 ERROR MESSAGES DURING EXECUTION AND CAUSES

1. After invoking of UPDATE.EXE, one of the following happens:
   -- On the screen is written: **EXEC failure**
   -- a message appears on the screen, with the word **HEAP** or **STACK**
   -- the screen turns dark and the program does not stop after typing CTRL-C
   -- strange characters are written on the screen, and the system needs to be rebooted to be able to work on it again

   All these things are caused by a too large program, or better sa to a RAM that is too small. Check with CHKDSK what the size of t free RAM of your computer is, and check whether other programs make use of this RAM. If you invoked PCSMP with the option D(ebug), invoke PCSMP again but now without that option and use option S(ave) instead.

   2. During execution an invalid mathematical operation is evaluat (e.g. division by zero). This is written on the screen. Invoke PCSMP with the option D(ebug) to find the line number at which this error occurs, and check the program carefully. Quite often parameters were not given a value, or a variable was spelled incorrectly. If this is the case, one can find in FOR03.DAT whic input was not given a value, and therefore still has the value 0 This message appears after the 'TRANSLATION TABLE CONTENTS'. If you don't find anything here, check carefully the value of

parameters and the calculation of the variables that could cause the erroneous mathematical operation. When the error according to the message on the screen appeared in line 0 (of UPDATE, or any user made subroutine), it might be due to multiplication or division by a very small number (in the order of 10^{-38}) in a PCSMP or FORTRAN function (e.g. AFGEN). Such a small number can occur as a result of rounding up errors. Check the input of all the PCSMP functions where multiplication or division occurs (i.e. INTGRL, AFGEN, TWOVAR, NFLGEN, EXP, ALOG, ALOG10, SQRT).

3. After invoking UPDATE, the execution stops without evaluating the program during a single time-step. This is most probably caused by an error in the INPUT. Check the FOR06.DAT file for error messages (marked with ***)

4. During EXECUTION, a message appears on the screen, similar to:

   ?Error DATA format error in file for06.dat
   Error code 1244, status 000E

   This error message can be caused by a missing FUNCTION table, that is called by an AFGEN or TWOVAR. Check all these statements.

5. Strange things happen during execution or the output shows strange results or even strange characters. This symptom can be caused by many errors, generally in the field of wrong dimensioning, or by calling subroutines with a different number of variables than they have in their definition. An example of this can be the use of the 'reserved words' (page 142 of the IBM CSMP manual).

6. The update seems to finish correctly, but the values of the output do not agree with the expected outcome. This can be caused by wrong input (check parameters etc. on values, and check the units/dimensions of both input and variables). Also the structure of the model can be wrong, or the order of calculation is not correct. The latter can be checked in the FOR03.DAT. To get the values of all the parameters and variables in the FOR06.DAT, one can use the PCSMP DEBUG subroutine. The following should be typed in the program just before the first END statement:

   ```
   NOSORT
   CALL DEBUG(n,x)
   SORT
   ```

   where n (integer) is the number of DELTs that all the variables are put in the FOR06.DAT, and x (real) is the first TIME where DEBUG is made active. Apart from doing this, the structure of the model should be checked (e.g. by making a flow diagram), and the assumptions should be checked, both in the model as well as those upon which your expectation for the output is based.

10. PLOTCSM

10.1 INTRODUCTION

PLOTCSM can make graphs from output of PCSMP-program where variable
are declared in the Output-Statement **PREPARE**. PLOTCSM uses two file
PREP1.DAT and **PREP2.DAT**. These files are created by the subroutine
PREPCSM which reads the results that have been saved by PCSMP in an
unformatted file called **FOR15.DAT** and writes the variables listed i
the PREPARE statement into the formatted files PREP1.DAT and
PREP2.DAT.

10.2 PRINTING DEVICES

The plots/graphs can be executed on three kinds of printing devices
1. Terminal - there are three versions of PLOTCSM available namely
   for an IBM-card, a Hercules-card and an EGA-card
2. Printer - EPSON (models FX-80, FX-100) and compatible printers
3. Plotter - HEWLETT PACKARD 7470A plotter

10.3 RESTRICTIONS

1. The total number of prepared variables must not exceed 25
2. The maximum prepared runs must not exceed 50.
3. A rerun must be preceded with the statement **END**. The statement **E**
   RERUN cannot be used.
4. The use of the statement **RESET PREPARE** is not allowed.
5. The use of multiple value parameters (Paragraph 7.2) is not
   effective when used with the PREPARE statement. Only one run, fo
   the last value in the list will be executed.

10.4 EXECUTION

To use PLOTCSM, the required command is **PLOT**. During the execution
PLOTCSM, the program will inquire how the user wants the resulting
graph to look like. The graph produced will be based only on the
options that the user chooses, by giving one of the following answe
to the questions regarding the types of the options:

- **Y** - yes
- **N** - no
- **S** - skip next **NUMBER** of runs
- **X** - a. default answer if first plot
  b. same answer as before for the same plot
- **D** - to change time interval that conforms with OUTDEL
- **M** - to change maxima and/or minima value(s)
- **E** - to put axes at the left corner bottom
- **O** - to force origin into the plot
- **P** - direct plot to printer
- **H** - direct plot to plotter
- **T** - direct plot to terminal (default)

or any combination of **D**,**M**,**E**, and **O**

Options to chose the run and/or variable(s) to be plotted:

a. 1-10 variables against time during one run in one plot
b. one variable against time during 1-10 runs in one plot
c. 1-10 variables against another during one run in one plot
d. one variable against another during 1-10 runs in one plot
e. one variable against one parameter at the end of each run

The type of line can be chosen:

line types	codes
cont. line	LSYM
no line	NSYM
broken line	DSYM

These types of lines can be accompanied by symbol terminators if distinction between runs or variables is desired:

symbol terminator	codes
dot	SYMB0
square	SYMB1
circle	SYMB2
triangle	SYMB3
plus	SYMB4
diamond	SYMB5
square with cross	SYMB6
elastic square	SYMB7
circle with cross	SYMB8
square with plus	SYMB9

When plotting up to six variables, and choosing the first six symbols (i.e. dot-diamond) to distinguish the lines made, the full names of these symbols will appear in the listing of the plot. If more than six variables are plotted, the codes for the line type and symbol terminator are printed. If the the 7^{th} to the 10^{th} symbol terminator (ie. square with cross - square with plus) are chosen, their codes will be printed instead of the full names.

To terminate PLOTCSM, **Ctrl-C** can be used to interrupt the program when a mistake has been made. However it is not advisable to do so in the phase that a plot is made on the screen, as the screen will not show the normal characters until after rebooting the whole system. After a plot has been made on the screen, the screen can be cleared by pressing the **ENTER** key only **once**. PLOTCSM will then ask whether you would like to make another plot, to which the default answer is No. Therefore, if after making a plot on the screen, the ENTER key is pushed more than once, PLOTCSM will stop after clearing the screen. After printing a plot (on printer or plotter), PLOTCSM will automatically come with the question for a new plot, thus if more plots should be made, the ENTER key should not be pushed after the printer reacted to the command to print the plot.

10.5 EXAMPLES

10.5.1. ONE VARIABLE VERSUS A PARAMETER, WITHOUT CONNECTING LINE

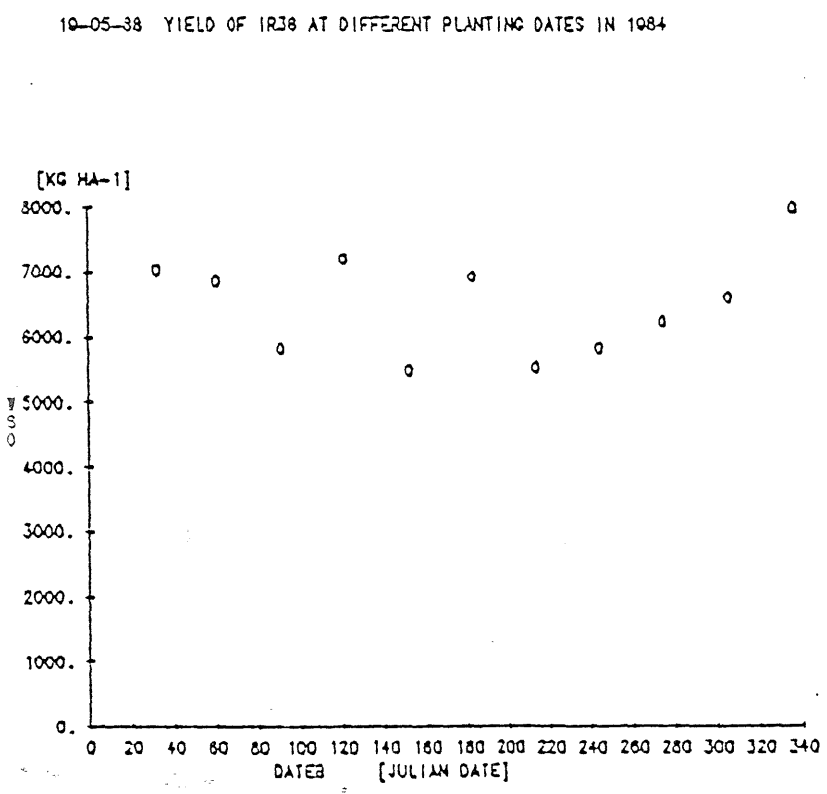
The data were generated with L1D.CSM, a model to calculate potenti
production of a crop in dependence of weather.

After giving the command **PLOT**, the following will be asked:

```
Which plot do you want?

Several variables against time during one run      [Y/N]: N

One variable against time during several runs      [Y/N]: N

Several variables against one during one run       [Y/N]: N

One variable against another during several runs   [Y/N]: N

one variable against one parameter at the end of each run [Y/N]: Y

Y variable:
WLV      [Y/N/S/X]: N
WST      [Y/N/S/X]: N
WRT      [Y/N/S/X]: N
WSO      [Y/N/S/X]: Y

One line style?      [Y/N//X]: Y
connect line         [Y/N]: N
no line              [Y/N]: Y

Symbols of own choice?     [Y/N/X]: Y
Symbol terminator for function 1

Dot      [Y/N]: N
Square   [Y/N]: N
Circle   [Y/N]: Y

Present date printed?  [Y/N]: Y

Header: YIELD OF IR36 AT DIFFERENT PLANTING DATES 1984

Do you want to print the units?  [Y/N/X]: Y
DATEB -unit: [JULIAN DATE]
WSO   -unit: [KG HA-1]

Plot on printer, plotter
or terminal?  (P/H/T): P
```

The resulting plot is shown in Figure 10.1

Figure 10.1: Plot of individual points of one variable versus a parameter.



10.5.2 ONE VARIABLE VERSUS A PARAMETER, WITH CONNECTING LINE

(After plotting the figure, PLOTCSM will ask the following:)

Do you want another plot? [Y/N]: **Y**

The same plot again? [Y/N/X]: **Y**

Any options? [DMEO/N]: **N**

One line style [Y/N/X]: **Y**
cont. line [Y/N]: **Y**

Symbols of own choice? [Y/N/X]: **X**

Present date printed? **Y**

Header: **YIELD OF IR36 AT DIFFERENT PLANTING DATES 1984**

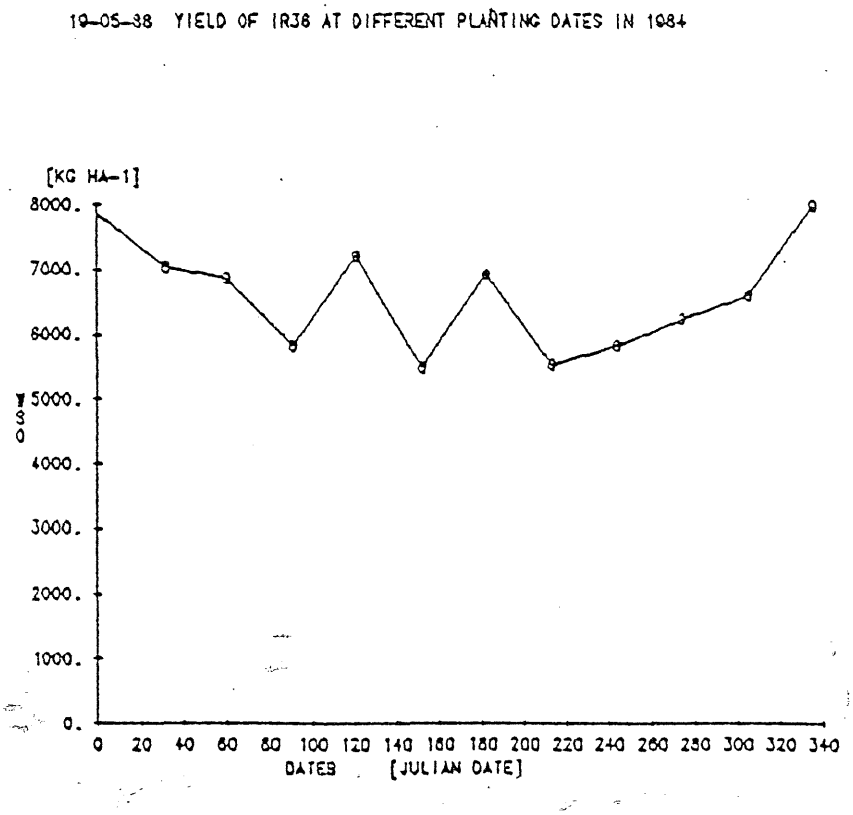
Do you want to print the units? [Y/N/X]: **X**

Plot on printer, plotter
or terminal (P/H/T): **P**

Do you want another plot? [Y/N] : **N**

The resulting plot is in Figure 10.2.

Figure 10.2: Plot of one variable versus a parameter, with a connecting line between the points.



10.5.3 PLOTTING ONE OR MORE VARIABLES VERSUS TIME

The data were generated with a program that read weatherdata from tables and prepared them for plotting.

COMMAND : **PLOT**

Which plot do you want?

Several variables against time during one run [Y/N]: **Y**

Which variable do you want to plot?]
MNT [Y/N/S/X]: **Y**
MXT [Y/N/S/X]: **Y**
VAP [Y/N/S/X]: N
WDSX [Y/N/S/X]: N
RAIN [Y/N/S/X]: N
RDTM [Y/N/S/X]: N
RDTC [Y/N/S/X]: N

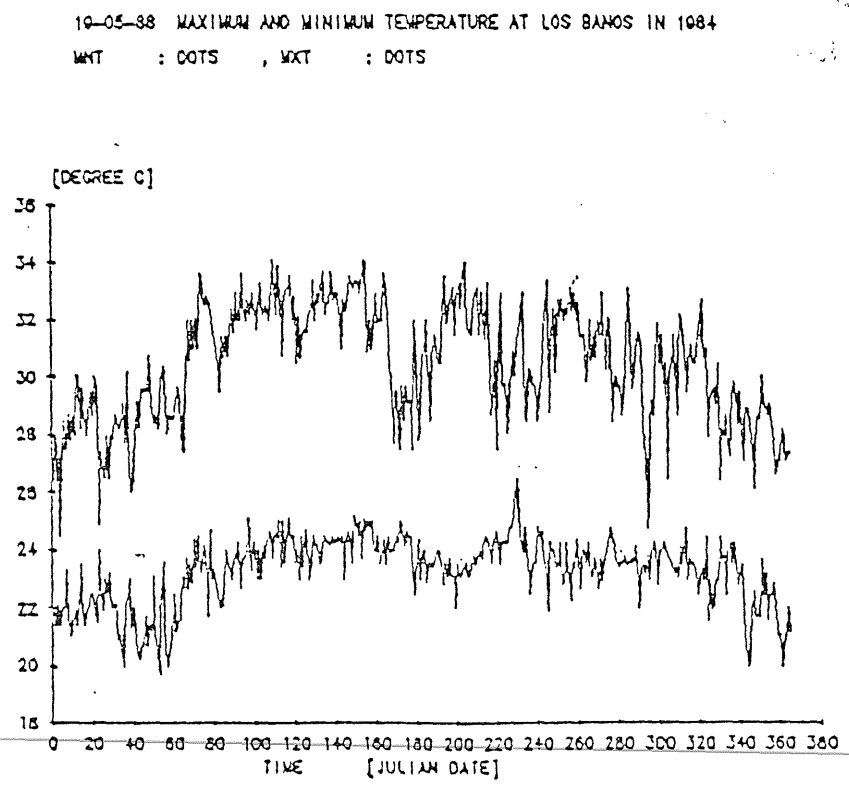
Plotting interval higher than 1.000 [Y/N]: N

One line style? [Y/N//X]: **Y**

```
cont. line              [Y/N]: Y
Symbols of own choice?      [Y/N/X]: Y
Symbol terminator for function 1
Dot      [Y/N]: Y
Symbol terminator for function 2
Dot      [Y/N]: Y

Present date printed?  [Y/N]: Y

Header: MAXIMUM AND MINIMUM TEMPERATURE AT LOS BANOS 1984

Do you want to print the units?  [Y/N/X]: Y
TIME -unit: [JULIAN DATE]
Y -unit: [DEGREE C]

Plot on printer, plotter
or terminal?  (P/H/T): P
```

The resulting plot is in Figure 10.3.

Figure 10.3: Plot of one variables versus time, with a connecting line between the points, and symbols of own choice.



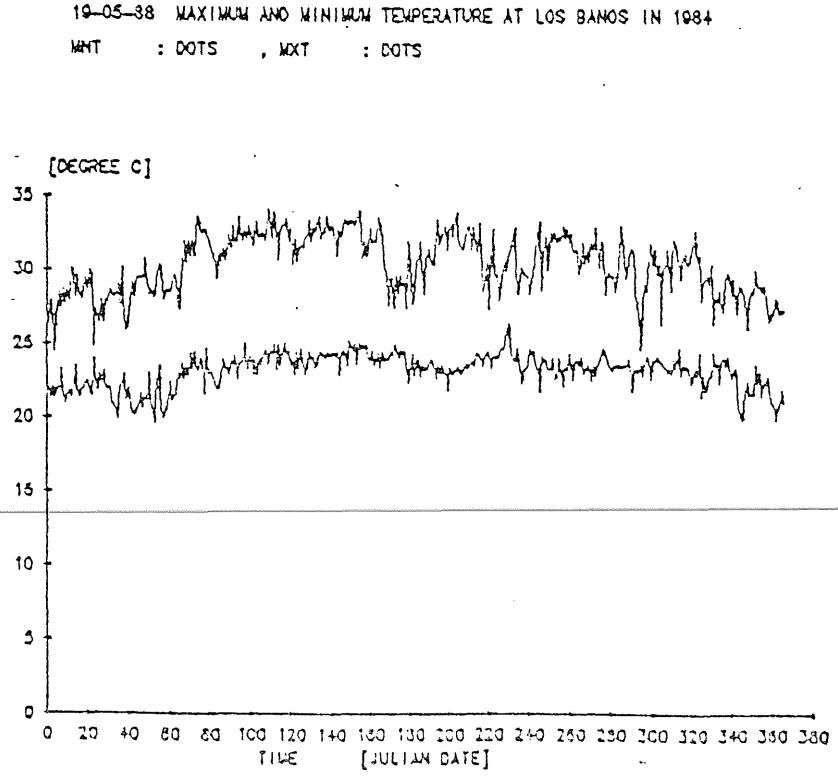
10.5.4 ADJUSTING THE MARGINS OF THE X AND/OR Y AXIS

After Figure 10.3 has been made, PLOTCSM asks:

```
Do you want another plot?      [Y/N]: Y

The same plot again?           [Y/N/X]: Y

Any options?                        [DMEO/N]: M
New X-minimum?    (xmin=       .100000D+01)      [Y/N] : N
New X-maximum?    (xmax=       .365000D+03)      [Y/N] : N
New Y-minimum?    (ymin=       .197000D+02)      [Y/N] : Y
0.
New Y-maximum?    (ymax=       .341000D+02)      [Y/N] : N

One line style              [Y/N/X]: X

Symbols of own choice?      [Y/N/X]: X

Present date printed? Y

Header: MAXIMUM AND MINIMUM TEMPERATURE AT LOS BANOS 1984

Do you want to print the units?    [Y/N/X]: X

Plot on printer, plotter
or terminal   (P/H/T): P
```

The result is in Figure 10.4

Figure 10.4: As Figure 10.3, after adjustment of margins of the Y-axis



10.5.5 SKIPPING SOME OPTIONS

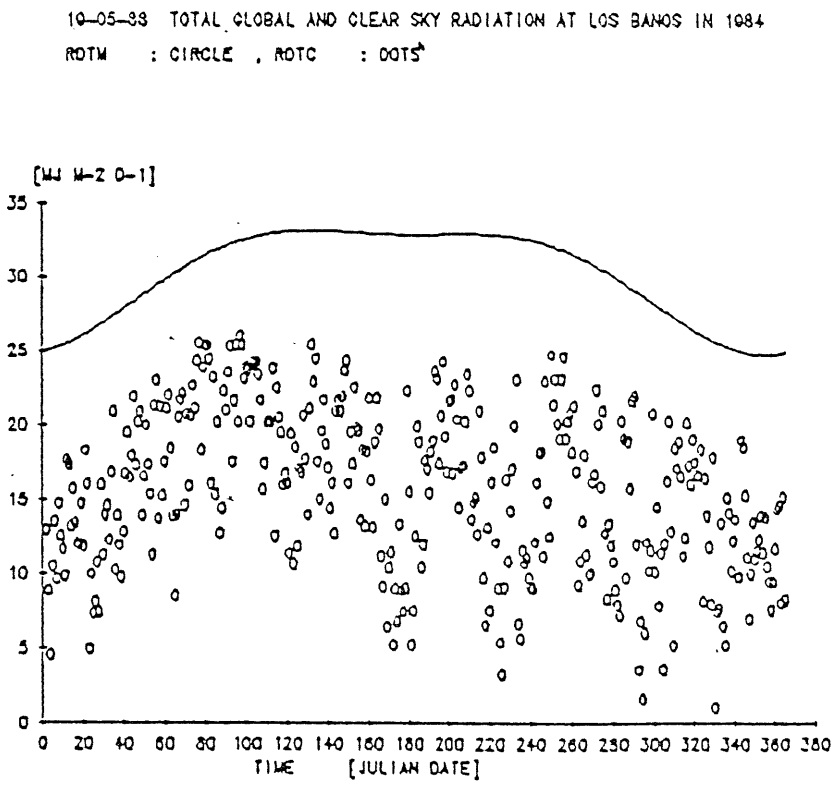
The same data of paragraph 10.5.3 are used. Now two variables will be plotted, that appear at the end of the list of variables that can be plotted. It will be shown how to SKIP several options to get quicker to the variables that you want to plot.
After the plotting of Figure 10.4, PLOTCSM will respond with:

```
Do you want another plot?     [Y/N]: Y

The same plot again?          [Y/N/X]: N

Several variables against time during one run     [Y/N]: Y

Which variable do you want to plot?
MNT      [Y/N/S/X]: N                  MNT      [Y/N/S/X]: S
MXT      [Y/N/S/X]: N                  5
VAP      [Y/N/S/X]: N                  RDTM     [Y/N/S/X]: Y
WDSX     [Y/N/S/X]: N    or            RDTC     [Y/N/S/X]: Y
RAIN     [Y/N/S/X]: N
RDTM     [Y/N/S/X]: Y                  (here, 5 variables are Skipped
RDTC     [Y/N/S/X]: Y                   from MNT onwards)

Plotting interval higher than 1.000  [Y/N]: N

One line style                [Y/N/X]: N

Symbols of own choice?        [Y/N/X]: Y

Line style of function 1
cont. line              [Y/N]: N
no line                 [Y/N]: Y
Symbol terminator for function 1
Dot      [Y/N]: N
Square   [Y/N]: N
Circle   [Y/N]: Y

Line style of function 2
cont. line              [Y/N]: Y
Symbol terminator for function 1
Dot      [Y/N]: Y

Present date printed? Y

Header: TOTAL GLOBAL AND CLEAR SKY RADIATION AT LOS BANOS 1984

Do you want to print the units?    [Y/N/X]: Y
TIME -unit: [JULIAN DATE]
Y -unit: [MJ M-2 D-1]

Plot on printer, plotter
or terminal  (P/H/T): P
```

Do you want another plot? [Y/N] : **N**

The result is in Figure 10.5.

Figure 10.5: Plot of one or more variables versus time, after skipping several variables.



APPENDIX A: ARITHMETIC RULES, FORTRAN AND CSMP FUNCTIONS

A.1. ARITHMETIC rules

Operator	Action	order
( )	grouping of calculations	1st
**	exponentiation	2nd
*	multiplication \	3rd
/	division /	
+	addition \	4rd
-	subtraction /	
=	replacement	5th

Functions and expressions within parentheses are always evaluated first. For operators of the same hierarchy, the component operations of the expression are performed from left to right. There is an exception for exponentiation, where the evaluation is from right to left. Thus, A**B**C is evaluated as A**(B**C).

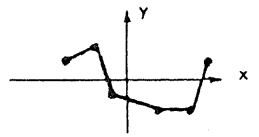
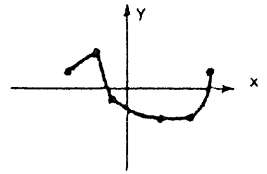
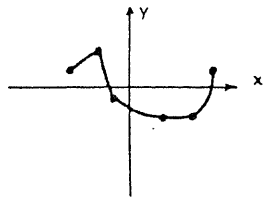
A.2 FORTRAN functions

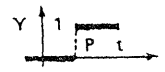
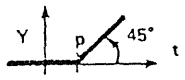
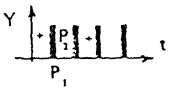
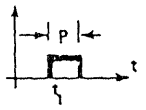
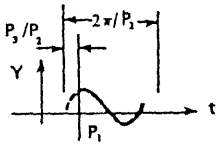
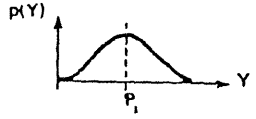
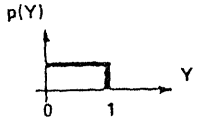
Name	Definition	Argument	Function
COS(X)	REAL*4 cosine	REAL*4	REAL*4
DCOS(X)	REAL*8 cosine	REAL*8	REAL*8
TAN(X)	REAL*4 tangent	REAL*4	REAL*4
DTAN(X)	REAL*8 tangent	REAL*8	REAL*8
ASIN(X)	REAL*4 arc sine	REAL*4	REAL*4
DASIN(X)	REAL*8 arc sine	REAL*8	REAL*8
ACOS(X)	REAL*4 arc cosine	REAL*4	REAL*4
DACOS(X)	REAL*8 arc cosine	REAL*8	REAL*8
ATAN(X)	REAL*4 arc tangent	REAL*4	REAL*4
DATAN(X)	REAL*8 arc tangent	REAL*8	REAL*8
ATAN2(X,Y)	REAL*4 arc tangent of X/Y	REAL*4	REAL*4
DATAN2(X,Y)	REAL*8 arc tangent of X/Y	REAL*8	REAL*8
SINH(X)	REAL*4 hyperbolic sine	REAL*4	REAL*4
DSINH(X)	REAL*8 hyperbolic sine	REAL*8	REAL*8
COSH(X)	REAL*4 hyperbolic cosine	REAL*4	REAL*4
DCOSH(X)	REAL*8 hyperbolic cosine	REAL*8	REAL*8
TANH(X)	REAL*4 hyperbolic tangent	REAL*4	REAL*4
DTANH(X)	REAL*8 hyperbolic tangent	REAL*8	REAL*8
LGE(C1,C2)	First argument lexically greater than or equal to second (8)	Character	Logical
LGT(C1,C2)	First argument lexically greater than second (8)	Character	Logical
LLE(C1,C2)	First argument lexically less than or equal to second (8)	Character	Logical
LLT(C1,C2)	First argument lexically less than second (8)	Character	Logical
EOF(I)	Integer end-of-file (9)	Integer	Logical

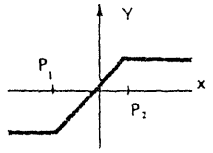
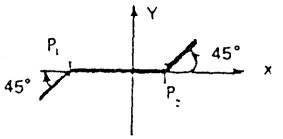
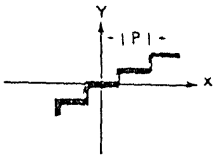
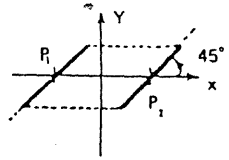
Name	Definition	Argument	Function
INT(X)	Conversion to integer (1)	REAL*4 Integer	Integer Integer
IFIX(X)	Conversion to integer (1)	REAL*4	Integer
IDINT	Conversion to integer (2)	REAL*8	Integer
REAL(X)	Conversion to REAL*4 (2)	Integer REAL*4	REAL*4 REAL*4
FLOAT(I)	Conversion to REAL*4	Integer	REAL*4
ICHAR(C)	Conversion to integer (3)	Character	Integer
CHAR(X)	Conversion to character (3)	Integer	Character
SNGL(X)	Conversion to REAL*4	REAL*8	REAL*4
DBLE(X)	Conversion to REAL*8 (4)	Integer REAL*4 REAL*8	REAL*8 REAL*8 REAL*8
AINT(X)	Truncation to REAL*4 (nearest whole number)	REAL*4	REAL*4
DINT(X)	Truncation to REAL*8 (nearest whole number)	REAL*8	REAL*8
ANINT(X)	Rounding to REAL*4 (nearest whole number)	REAL*4	REAL*4
DNINT(X)	Rounding to REAL*8 (nearest whole number)	REAL*8	REAL*8
NINT(X)	Rounding to integer (1)	REAL*4	Integer
IDNINT(X)	Rounding to integer	REAL*8	Integer
IABS(I)	Integer absolute value	Integer	Integer
ABS(X)	REAL*4 absolute value	REAL*4	REAL*4
DABS(X)	REAL*8 absolute value	REAL*8	REAL*8
MOD(I,J)	Integer remainder (5)	Integer	Integer
AMOD(X,Y)	REAL*4 remainder (5)	REAL*4	REAL*4
DMOD(X,Y)	REAL*8 remainder (5)	REAL*8	REAL*8
ISIGN(I,J)	Integer sign transfer (6)	Integer	Integer
SIGN(X,Y)	REAL*4 sign transfer (6)	REAL*4	REAL*4

Name	Definition	Argument	Function
DSIGN(X,Y)	REAL*8 sign transfer (6)	REAL*8	REAL*8
IDIM(I,J)	Integer positive difference (7)	Integer	Integer
DIM(X,Y)	REAL*4 positive difference (7)	REAL*4	REAL*4
DDIM(X,Y)	REAL*8 positive difference (7)	REAL*8	REAL*8
MAX0(I,J,...)	Integer maximum	Integer	Integer
AMAX1(X,Y,...)	REAL*4 maximum	REAL*4	REAL*4
AMAX0(I,J,...)	REAL*4 maximum	Integer	REAL*4
MAX1(X,Y,...)	Integer maximum	REAL*4	Integer
DMAX1(X,Y,...)	REAL*8 maximum	REAL*8	REAL*8
MIN0(I,J,...)	Integer minimum	Integer	Integer
AMIN1(X,Y,...)	REAL*4 minimum	REAL*4	REAL*4
AMIN0(I,J,...)	REAL*4 minimum	Integer	REAL*4
MIN1(X,Y,...)	Integer minimum	Real*4	Integer
DMIN1(X,Y,...)	REAL*8 minimum	REAL*8	REAL*8
DPROD(X,Y)	REAL*8 product X*Y	REAL*4	REAL*8
SQRT(X)	REAL*4	REAL*4	REAL*4
DSQRT(X)	REAL*8 square root	REAL*8	REAL*8
EXP(X)	REAL*4 e raised to power	REAL*4	REAL*4
DEXP(X)	REAL*8 e raised to power	REAL*8	REAL*8
ALOG(X)	Natural logarithm of REAL*4 argument	REAL*4	REAL*4
DLOG(X)	Natural logarithm of REAL*8 argument	REAL*8	REAL*8
ALOG10(X)	Common logarithm of REAL*4 argument	REAL*4	REAL*4
DLOG(X)	Common logarithm of REAL*8 argument	REAL*8	REAL*8
SIN(X)	REAL*4 sine	REAL*4	REAL*4
DSIN(X)	REAL*8 sine	REAL*8	REAL*8

CSMP III Statement	Equivalent Mathematical Expression
INTEGRATOR Y = INTGRL (IC, X) where: IC = $y\|_{t=t_s}$ Alternative 'Specification' Form: Y = INTGRL (IC, X, N) where: Y = output array IC = initial condition array X = integrand array N = number of elements in the integrator array (N must be coded as a literal integer constant)	$y(t) = \int_{t_s}^{t} x\,dt + y(t_s)$ where: t_s = start time t = time $\dot{y} = \int_{t_s}^{t} \dot{x}\,dt + \dot{y}(t_s)$ Equivalent Laplace Transfer Function: $\frac{Y(s)}{X(s)} = \frac{1}{s}$
DERIVATIVE Y = DERIV (IC, X) where: IC = $\frac{dx}{dt}\Big\|_{t=t_s}$	$y = \frac{dx}{dt}$ Equivalent Laplace Transfer Function: $\frac{Y(s)}{X(s)} = s$
IMPLICIT FUNCTION Y = IMPL (IC, P, FOFY) where IC = first guess P = error bound FOFY = output name from final statement in algebraic loop definition	$y = f(y)$ $\|y - f(y)\| < p\|y\|$
DEAD TIME (DELAY) Y = DELAY (N, P, X) where: P = delay time N = number of points sampled in interval p (integer constant) and must be ≥ 3, and $\leq 16,378$	$y = x(t-p)$; $t \geq p$ $y = 0$; $t < p$ Equivalent Laplace Transfer Function: $\frac{Y(s)}{X(s)} = e^{-ps}$
ZERO-ORDER HOLD Y = ZHOLD (X1, X2)	$y = x_2$; $x_1 > 0$ y = last output ; $x_1 < 0$ $y_{t=t_s} = 0$ Equivalent Laplace Transfer Function $\frac{Y(s)}{X(s)} = \frac{1}{s}(1 - e^{-st})$
DOUBLE PRECISION FLOATING-POINT TO SINGLE PRECISION Y = ZZRND (DNUMBR) where: DNUMBR is a double precision floating-point number. Y = rounded single precision value of DNUMBR	To convert the double precision floating-point number, DNUMBR, to single precision, rounding to hexadecimal digit.

CSMP III Statement	Equivalent Mathematical Expression
ARBITRARY FUNCTION GENERATOR (LINEAR INTERPOLATION) Y = AFGEN (FUNCT, X)	 y = f(x)
ARBITRARY FUNCTION GENERATOR (QUADRATIC INTERPOLATION) Y = NLFGEN (FUNCT, X)	 y = f(x)
FUNCTION GENERATOR WITH DEGREE OF INTERPOLATION CHOSEN BY USER Y = FUNGEN (FUNCT, N, X) where: FUNCT = function name N = degree of interpolation to be used. May be 1, 2, 3, 4, or 5 X = value of abscissa	 y = f(x)
ARBITRARY FUNCTION OF 2 VARIABLES Y = TWOVAR (FUNCT, X, Z)	y = f(x, z)
SLOPE OF A CURVE Y = SLOPE (FUNCT, N, X) where: FUNCT = name of curve N = degree of interpolation to be used X = value of abscissa	$y = \frac{df}{dx}$ at x
SAMPLING INTERVAL SWITCH Y = SAMPLE (P1, P2, {P3 / N}) where: P_1 = start time for sampling to occur P_2 = last time for sampling to occur P_3 = time interval between samples if entered as a floating-point number N = number of sampling intervals if entered as a fixed-point number	$y = 1.0$; TIME = $p_1 + k\,p_3 \leq p_2$ k = 0, 1, 2, . . . or $y = 1.0$; TIME = $p_1 + k(p_2 - p_1)/n \leq p_2$ k = 0, 1, 2, . . . y = 0.0 otherwise

CSMP III Statement	Equivalent Mathematical Expression
STEP FUNCTION Y = STEP (P)	$y = 0 \;;\; t < p$ $y = 1 \;;\; t \geq p$ 
RAMP FUNCTION Y = RAMP (P)	$y = 0 \;;\; t < p$ $y = t - p \;;\; t \geq p$ 
IMPULSE GENERATOR Y = IMPULS (P1, P2) where: P1 = time of first pulse P2 = interval between pulses	$y = 0 \;;\; t < p_1$ $y = 1 \;;\; (t - p_1) = kp_2$ $y = 0 \;;\; (t - p_1) \neq kp_2$ $k = 0, 1, 2, 3, \ldots$ 
PULSE GENERATOR (WITH X > 0 AS TRIGGER) Y = PULSE (P, X) where: P = minimum pulse width	$y = 1 \;;\; t_t \leq t < (t_t + p)$ or $x > 0$ $y = 0$; otherwise (t_t = time of trigger) 
TRIGONOMETRIC SINE WAVE WITH DELAY, FREQUENCY AND PHASE PARAMETERS Y = SINE (P1, P2, P3) where: P1 = delay P2 = frequency (in radians per unit time) P3 = phase shift in radians	 $y = 0$; $t < p_1$ $y = \sin(p_2 (t - p_1) + p_3)$; $t \geq p_1$
NOISE (RANDOM NUMBER) GENERATOR WITH NORMAL DISTRIBUTION Y = GAUSS (S, P1, P2) where: S = seed (S is first truncated to an integer if real valued, then its absolute value is used once to initialize the generator) P1 = desired mean of values of Y P2 = desired standard deviation of values of Y	Normal Distribution of Variable y p(y) = probability density function 
NOISE (RANDOM NUMBER) GENERATOR WITH UNIFORM DISTRIBUTION Y = RNDGEN (S) where: S = seed (S is first truncated to an integer if real valued, then its absolute value is used once to initialize the generator)	Uniform Distribution of Variable y p(y) = probability density function 

CSMP III Statement	Equivalent Mathematical Expression
LIMITER Y = LIMIT (P1, P2, X)	$y = p_1 \;;\; x < p_1$ $y = p_2 \;;\; x > p_2$ $y = x \;;\; p_1 \leq x \leq p_2$ 
DEAD SPACE Y = DEADSP (P1, P2, X)	$y = 0 \;;\; p_1 \leq x \leq p_2$ $y = x - p_2 \;;\; x > p_2$ $y = x - p_1 \;;\; x < p_1$ 
QUANTIZER Y = QNTZR (P, X)	 $y = kp \;;\; (k - \frac{1}{2})\, p < x \leq (k + \frac{1}{2})\, p$ $k = 0, \pm 1, \pm 2, \pm 3, \ldots$
HYSTERESIS LOOP Y = HSTRSS (IC, P1, P2, X) where: IC = $y|_{t_s}$	 $y = x - p_2 \;;\; (x - x(t - \Delta t)) > 0$ and $y(t - \Delta t) \leq (x - p_2)$ $y = x - p_1 \;;\; (x - x(t - \Delta t)) < 0$ and $y(t - \Delta t) \geq (x - p_1)$ otherwise: $y = y(t - \Delta t)$

CSMP III Statement	Equivalent Mathematical Expression	
NOT Y = NOT (X)	$y = 1;$ $y = 0;$	$x < 0$ $x > 0$
AND Y = AND (X1, X2)	$y = 1;$ $y = 0;$	$x_1 > 0, x_2 > 0$ otherwise
NOT AND Y = NAND (X1, X2)	$y = 0;$ $y = 1;$	$x_1 > 0, x_2 > 0$ otherwise
INCLUSIVE OR Y = IOR (X1, X2)	$y = 0;$ $y = 1;$	$x_1 < 0, x_2 < 0$ otherwise
EXCLUSIVE OR Y = EOR (X1, X2)	$y = 1;$ $y = 1;$ $y = 0;$	$x_1 < 0, x_2 > 0$ $x_1 > 0, x_2 < 0$ otherwise
NOT OR Y = NOR (X1, X2)	$y = 1;$ $y = 0;$	$x_1 < 0, x_2 < 0$ otherwise
EQUIVALENT Y = EQUIV (X1, X2)	$y = 1;$ $y = 1;$ $y = 0;$	$x_1 < 0, x_2 < 0$ $x_1 > 0, x_2 > 0$ otherwise

CSMP III Statement	Equivalent Mathematical Expression	
COMPARATOR Y = COMPAR (X1, X2)	$y = 0;$ $y = 1;$	$x_1 < x_2$ $x_1 > x_2$
OUTPUT SWITCH Y1, Y2 = OUTSW (X1, X2)	$y_1 = x_2, y_2 = 0;$ $y_1 = 0, y_2 = x_2;$	$x_1 < 0$ $x_1 > 0$
INPUT SWITCH RELAY Y = INSW (X1, X2, X3)	$y = x_2;$ $y = x_3;$	$x_1 < 0$ $x_1 > 0$
RESETTABLE FLIP-FLOP Y = RST (X1, X2, X3)	$y = 0;$ $y = 1;$ $y = 0;$ $y = 1;$ $y = 0;$ $y = 1;$	$x_1 > 0$ $x_2 > 0, x_1 < 0$ $x_1 < 0$, $x_3 > 0, y(t - \Delta t) = 1$ $x_1 < 0$, $x_3 > 0, y(t - \Delta t) = 0$ $x_2 < 0$, $x_3 < 0, y(t - \Delta t) = 0$ $x_2 < 0$, $x_3 < 0, y(t - \Delta t) = 1$
FUNCTION SWITCH Y = FCNSW (X1, X2, X3, X4)	$y = x_2;$ $y = x_3;$ $y = x_4;$	$x_1 < 0$ $x_1 = 0$ $x_1 > 0$

Appendix B. INSTALLATION procedure of PCSMP on PC using floppies PCSMP-V4-01 through PCSMP-V4-04

B.1. Acquisition of floppies PCSMP-V4

After receiving 4 (four) double sided, double density floppy disks, the floppies containing the installation procedure will be send to you by one of the following persons:

D.M. Jansen, IRRI MCD, P.O. Box 933, Manila, Philippines

or

H.F.M. ten Berge, CABO, P.O. Box 14, 6700 AA Wageningen, the Netherlands

B.2. System requirements

Before installation of PCSMP by using the command procedure INSTALL on floppy PCSMP-V4-01, be sure that there is a PATH to the subdirectory where the DOS commands are located, and to the COMMAND.COM file. This can be tested by typing the DOS command PATH.

On the root directory of the hard disk, a file CONFIG.SYS should be present which contains the following lines:

```
device=c:\ansi.sys
files=20
break=on
fcbs=8,0
buffers=40
```

When using this CONFIG.SYS, the file ANSI.SYS should be present on the root directory, to facilitate the interpretation of several commands to write characters on the screen. ANSI.SYS can be placed on another directory, but then the first line in the CONFIG.SYS shown here, has to be adapted.

The plot program that is on these floppies, needs a HERCULES MONOCHROME GRAPHICS CARD (or compatible). To enable plotting on the screen, also a MONOCHROME MONITOR is needed. A graphics printer (e.g. EPSON FX-85, or LX-800) is required to obtain hard-copies of the plot-output.
The plot program is not required for adequate running of PCSMP on the computer, it merely makes its output more appreciable. Thus, even if the system in use does not meet the plotting requirements, the user can still run PCSMP.
Furthermore, the following is required:
-- free hard disk space of 2MB or more
-- default startup from harddisk (OS resides on hard disk)
-- hard disk identification is C:
-- 5 1/4" diskette drive present, with identification A:

B.3. Description of the contents of the diskettes.

a. Files on diskettes:

Floppy:	PCSMP-V4-01	PCSMP-V4-02	PCSMP-V4-03	PCSMP-V4
04				
Files:	FOR1.EXE	LINK.EXE	TPLGSA.DAT	
PLOTCSM.EXE				
	FOR2.EXE	LIB.EXE	PCSMPFIG.EXE	
	INSTAL2.BAT			
	FOR3.EXE	PLOT.TXT	AUTOEXEC.BAT	TEST.BAT
	EMULATOR.LIB	SUCROS.CSM	SUBR.FOR	TEST2.BA
	IBMFOR.LIB	STPL.FNT	8087ONLY.LIB	LARGE.CS
	CONFIG.SYS	ZLINK.BAT	C8087.EXE	SROM.FNT
	ANSI.SYS	C8087.FOR	8087.COM	
	INSTALL.BAT	\EXEC	\TRANS	
	INSTAL1.BAT			
	README.TXT			
	PLOT.BAT			
	STOP.BAT			
	PCSMP.BAT			

b. File description:

Most files are discussed in chapter 1.2 of the handout: PCSMP on IB
PC-AT's or PC-XT's and compatibles. Here only the files needed duri
the installation and testing will be discussed:

\EXEC
This subdirectory contains the PCSMP source files for the EXECUT.LIB and the MAIN2.OBJ

\TRANS
This subdirectory contains the PCSMP source files for the TRANS.EXE.

C8087.FOR
The FORTRAN source code for C8087.EXE. When another directory-structure is used than the default of the installation procedure the indication of the path in the C8087.FOR has to be adapted accordingly. After compilation and linking of the adapted C8087.FOR, the newly made C8087.EXE should replace the existing file (by default on C:\PCSMP).

INSTALL.BAT
The command procedure to install PCSMP on the computer. It is to be invoked by putting floppy PCSMP-V4-01 in drive A:, and typing A:\INSTALL

INSTAL1.BAT
A command procedure invoked by INSTALL.BAT to facilitate installation of PCSMP.

INSTAL2.BAT
A command procedure invoked by INSTAL1.BAT to facilitate installation of PCSMP.

LARGE.CSM
A large PCSMP program, to test whether the internal memory of the computer is large enough.

SUBR.FOR
Subroutines that are invoked by LARGE.CSM

TEST.BAT
A batch file facilitating testing of SUCROS.CSM, the PLOT program, and LARGE.CSM (+SUBR.FOR).

TEST2.BAT
The batch file facilitating testing of LARGE.CSM (+SUBR.FOR) only.

B.4. Installation of PCSMP

Sequence of actions:
(<ENTER> means: push the ENTER key)

1. **Power up**: wait until prompt appears, and check whether there is a path to the directory with the DOS commands and to the DOS COMMAND.COM file (with the PATH command)

2. **Put floppy PCSMP-V4-01 in disk drive A:**

3. **A:**<ENTER>

4. **INSTALL** <ENTER>
   Do as is instructed. All 4 floppies will be used. The compilation of EXECUT.LIB and TRANS.EXE will take a long time, it varies between about half an hour on (fast; 10 MHz) AT compatibles and about 3 hours on (slow) XT compatibles.

5. After successful completion of the installation procedure, there are 2 options: to use the TEST program (see paragraph B.6 - B.9) or to do the testing by hand (following steps 6 to 11). To chose the latter option, you should push **CTRL-C** when the installation-procedure asks you whether you want to test the program

6. **Remove PCSMP-V4-04 from disk drive A:**

7. **Push CTRL-ALT-DEL keys all at the same time** (i.e. rebooting of the system)

8. Check whether all the files are copied to C:\PCSMP by typing:
   DIR C:\PCSMP

On directory C:\PCSMP the following files should be found:

1. TRANS.EXE	8. PLOT.BAT	15. PCSMPFIG.EXE
2. EXECUT.LIB	9. PLOTCSM.EXE	16. SUCROS.CSM
3. MAIN2.OBJ	10. TPLGSA.DAT	17. LARGE.CSM
4. PCSMP.BAT	11. STPL.FNT	18. SUBR.FOR
5. STOP.BAT	12. SROM.FNT	19. README.TXT
6. ZLINK.BAT	13. TEST.BAT	20. PLOT.TXT
7. C8087.EXE	14. TEST2.BAT	

9. Check whether all the files were copied to C:\FORTRAN by typing:
   DIR C:\FORTRAN

   On directory C:\FORTRAN the following files should be found:

1. FOR1.EXE	4. LINK.EXE	7. IBMFOR.LIB
2. FOR2.EXE	5. EMULATOR.LIB	8. LIB.EXE
3. FOR3.EXE	6. 8087ONLY.LIB	9. 8087.COM

10 Create a scratch subdirectory (e.g. C:\SCRATCH) by:
   MD C:\SCRATCH

11 change directory to C:\SCRATCH:
   CD \SCRATCH

12 copy SUCROS.CSM from C:\PCSMP to C:\SCRATCH:
   COPY C:\PCSMP\SUCROS.CSM C:\SCRATCH

13 Test PCSMP by running SUCROS.CSM:
   PCSMP SUCROS.CSM D

If step 13 is successful, the the following files are on C:\SCRATCH

1. SUCROS.CSM	6. UPDATE.EXE	11. FOR03.DAT
2. UPDATE.FOR	7. DATA.FOR	12. PREP1.DAT
3. UPDATE.OBJ	8. DATA.OBJ	13. PREP2.DAT
4. UPDATE.LST	9. CONTRO.SYS	14. FOR06.DAT
5. UPDATE.MAP	10. TABLE.TMP	

The directory structure that is created during installation, is jus an example. Some users might prefer to put the PCSMP and FORTRAN programs on other subdirectories than the default of the installati procedure (e.g. on C:\SYS\PCSMP and C:\SYS\MSFOR2). This can be achieved by copying the files from C:\PCSMP and C:\FORTRAN to the respective subdirectories. However, in the batch files used during running of PCSMP (i.e. PCSMP.BAT, PLOT.BAT) the references to C:\PCSMP and C:\FORTRAN should be changed to the directory names preferred by the user. Also the PATH statement in the AUTOEXEC.BAT should be changed accordingly. Furthermore, the file C8087.EXE (default on C:\PCSMP) should be adapted. This can be achieved by changing the reference to the FORTRAN directory in the file C8087.F (on floppy PCSMP-V4-02). After compilation and linking, the adapted C8087.EXE should be put on the new PCSMP subdirectory.

B.5. IN CASE OF TROUBLE DURING INSTALLATION

Many error messages (e.g. Bad Command or file name) can occur if there is no PATH to the COMMAND.COM, or when this COMMAND.COM is of a DOS version older than version 3.2. Check this first before going into details on the error messages.

I. **Problem:** the floppies can not be read.
Cause & Solution: either the operating system is not compatible with DOS 3.21 or the floppies have been maltreated. If you have an operating system that is not compatible with DOS, PCSMP will not run on the computer, unless the operating system is changed. If the operating system is compatible, send 4 formatted double density floppy disks to the person or organisation from whom/which you acquired the floppies so they can be sent back with the PCSMP files on it. Also send back the 4 installation floppies so that can be checked whether they are still in working order.

II. **Problem:** after invoking A:\INSTALL, not all the files are found in C:\PCSMP or C:\FORTRAN.
Cause & Solution: If the systems configuration is sufficient, the hard disk is called C:, the floppy drive is called A:, and the floppies can be read, but during running of INSTALL.BAT problems occur, remarks will be written to the file C:\PCSMP\INSTALL.LST (which is made during installation). If the installation was succesfull, in this file only remarks are found as ' file(s) copied' and ' press ENTER to continue'. If other remarks are found (e.g. 'file not found'), this is an indication of some error during installation. Check in the file C:\PCSMP\INSTALL.LST where an error message occured, or where a file could not be found. Check in the INSTALL.BAT, INSTAL1.BAT, and INSTAL2.BAT where this problem occurred, and try to solve the problem.

B.6. Test run of a small PCSMP program

One of the files that is copied to C:\PCSMP is a small PCSMP program called SUCROS.CSM. The TEST program will invoke translation, compilation and execution of the file. Start of the translation is shown by the line:

execution of SUCROS.CSM

After a while, when translation and compilation are through, and execution of the program starts, the screen will show the following lines (with some blank lines in between):

PCSMP execution program V4.0 October 1987 BC,DJ

TIME .0000 FINTIM .0000 DELT .0000

The lowest line will suddenly change into:

TIME 300.0 FINTIM 1000. DELT 2.000

The number after time will increase with steps of one unit till at successful completion of the run the line reads:

TIME 390.0 FINTIM 1000. DELT 2.000

If there was any error during translation to execution, a message will be written to the screen. If no errors appeared during running of the small program, you will be asked to test the PLOT program:

SUCCESSFUL COMPLETION OF THE TEST WITH A SMALL PROGRAM
IT RAN SUCCESSFULLY ON YOUR COMPUTER

now a plot program will be called. It will only be effective if you have a HERCULES or compatible MONOCHROME GRAPHICS CARD in your computer and MONOCHROME MONITOR attached to it.
Even if you do not have such a card, you can test the performance o a large PCSMP program (see option 2)

You have the following options:
1. PUSH CTRL-C, type Y to terminate the batch job and stop the testing
2. PUSH CTRL-C, type Y to terminate the batch job and type TEST2
3. PUSH ENTER and continue with the test of the PLOT program

If you want to skip testing of the plot program but want to continue with the testing of the large model, go to paragraph B.8. During testing of the plot program, several questions have to be answered, see paragraph B.7.

B.7. Testing the PLOT program

The next list will give you the questions plus the default answers provided by the PLOT program. Both will be written under the header **Question**. The answers that you have to type to reproduce Figure B.1, are given under the header **Answer**. After typing this one letter answer you have to push the ENTER key, to make the computer (the computer) accept the answers.

1. Question:
   Options PLOTCSM:
   Several variables against time during one run
   One variable against time during several runs
   Several variables against one during one run
   One variable against another during several runs
   one variable against one parameter at the end of each run

Which plot do you want?

Several variables against time during one run [Y/N]:
N

1. Answer: Y

2. Question (after each answer 'Y', a new variable will show up):
Which variables do you want to plot?
NWRT [Y/N/S/X]: N
WLV [Y/N/S/X]: N
WLVT [Y/N/S/X]: N
WVEG [Y/N/S/X]: N
TADRW [Y/N/S/X]: N
2. Answer to all the questions above: Y

3. Question:
Plotting interval higher than 2.000 [Y/N]: N
3. Answer: N

4. Question:
One line style [Y/N/X]: N
4. Answer: Y

5. Question:
cont. line [Y/N/X]: N
5. Answer: Y

6. Question:
Symbols of own choice [Y/N/X]: N
6. Answer: N

7. Question:
Present date printed [Y/N]: N
7. Answer: N

8. Question:
Header:
8. Answer: TEST

9. Question:
Do you want to print the units? [Y/N/X]: N
9. Answer: N

10. Question:
Plot on printer, plotter
or terminal? (P/H/T): T
10. Answer: T

Figure B.1 will be shown on the screen (the creation of the figure takes some time, but should not take more than a few minutes). After creation of this figure, the computer will wait for your sign to continue. If you have checked the figure on the screen with Figure

B.1, you should push the ENTER key (only once!). Then, the followir question will be asked:

11. Question:
Do you want another plot? [Y/N]: N
11. The answer depends on whether you want to test a printer or not
If not, you respond: N
If you want to test whether the printer connected to the comput can
print the figure you saw on the screen, you should respond with Y

If on question 11 your answer was N, the computer will continue wit the PCSMP - test, to check the performance of a large model (paragraph B.8). If you responded Y to question 11, the following questions will be asked:

12. Question:
The same plot again? [Y/N/X]: N
12. Answer: Y

13. Question:
Any options? [DMEO/N]: N
13. Answer: N

14. Question:
One line style [Y/N/X]: N
14. Answer: X

15. Question:
Symbols of own choice [Y/N/X]: N
15. Answer: X

16. Question:
Present data printed [Y/N]: N
16. Answer: N

17. Question:
Header:
17. Answer: TEST

18. Question:
Do you want to print the units [Y/N/X]: N
18. Answer: N

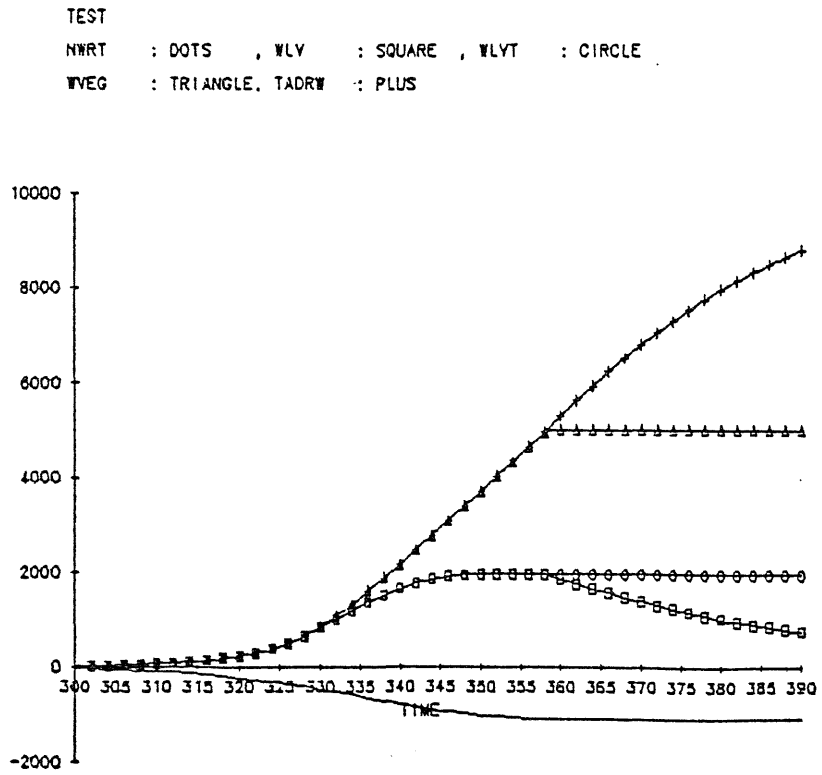
19. Question:
Plot on printer, plotter
or terminal? (P/H/T): T
19. Answer: P

After question 19, the printer (if 'on line') will respond with son movement of the printer head, and fall silent for several minutes. Then, bit by bit Figure B.1 will be printed. It can take some time before the printer really starts doing something. However if after

minutes there is still no figure being printed, or if some rubbish comes out of the printer, then you'll have to think about looking for another printer.
If something goes wrong during creation of the figure on screen or printer, you might have to reboot the computer. This means that the graphics did not work properly. Still you might try to test whether the computer can handle a large PCSMP program. If so, reboot the system, change to directory C:\SCRATCH, and continue the testing by typing TEST2.

After the printer has done its job, question 11 will be repeated. If you want to make another plot, repeat questions 12 - 19.

Figure B.1: Example of a plot made by PLOTCSM



B.8. Testing a large program

To check whether the free RAM on your computer really is sufficient a large PCSMP program is tested. Initialisation of this test is sho by:

Now a big PCSMP program will be tested, which will take some time (about 15 minutes). If you want to quit, push CTRL-C, otherwis push ENTER

The time needed depends on the configuration and clock speed. If yo can not spend the time, just push CTRL-C (and respond with Y to the question whether you want to terminate the batch job). If you want clean up the hard disk, you should delete all the files on the subdirectories:
C:\PCSMP, C:\PCSMP\EXEC, C:\PCSMP\TRANS, C:\FORTRAN, C:\SCRATCH.
Push the ENTER key to continue testing, and the following lines appear:

If your CPU has not enough RAM available (less than 500KB), you can expect one of the following to happen during the EXECUTION phase:

1. you get the message: EXEC FAILURE
2. your screen turns into a mess, and CTRL-C does not function anymore. Rebooting of the system is needed.
3. you get a message with the word HEAP or STACK in it (e.g. OUT OF STACK)

If such a problem occurs, either change the configuration of your CPU, or quit testing and try to find a more suitable CPU

press ENTER to continue

Simply push CTRL-C if you want to stop, (and terminate the batch job). If you continue the testing, and your system is too small, or not compatible with the ones on which we tested PCSMP, you might get other problems:

Another problem that might occur is that during execution the program will hang. I.e. execution will start, but stop at a certain point, without returning to the TEST program.

This is the case if the following line is shown on the screen without being changed for about 5 minutes:

TIME .0000 FINTIM 10.00 DELT 1.000

If this happens, your RAM is still too small, or the configuration that you have is not compatible with those that we tested the PCSMP on. In this case it is better to look for another computer to run PCSMP on

press ENTER to Continue

Again here you can quit by pushing CTRL-C.
When you pushed ENTER to continue the testing of the large program, you will have to wait quite some time before the following lines appear:

PCSMP execution program V4.0 October 1987 BC,DJ

TIME .0000 FINTIM .0000 DELT .0000

When execution is complete (and successful), the last line changes into:

TIME 10.00 FINTIM 10.00 DELT 1.000

Hereafter, a reconfirmation of the success is printed on the screen:

Your CPU made it !

You can use this CPU to run PCSMP on it.

B.9. Error messages during testing

Errors can occur during various stages of the testing. If they occu
your system configuration was most probably not suited for running
PCSMP, and you should try to change the configuration or forget abc
PCSMP on your system. Maybe there is no path to the COMMAND.COM fil
(check this).

Most errors will result in a message in the file INSTALL.LST on
C:\PCSMP. When the testing was not succesfull, first check the
contents of this file.

Errors during copying of files:

Message:
General Failure error reading drive A:
Abort Retry Ignore?
Cause:
The disk drive of your computer can't read double density flop
disks that are formatted under DOS 3.21.

Message:
Not ready error reading drive A:
Cause:
No floppy in drive A:, or disk drive is not called A: (the
latter can occur if you have more disk drives; if this is the
case, you will have to put the floppies into the other disk
drive attached to the computer.

Message:
A:\'name.ext' File not found
Cause:
The file 'name.ext' (here used as example) does not exist on
floppy A:. This error can only occur if files are deleted from
the floppy.

Message:
BAD COMMAND OR FILENAME
Cause:
Most probably the path is not set to the directory where the D
file COMMAND.COM is found.