USER'S GUIDE FOR THE POST-PROCESSOR OF MESSAGE II

M. Strubegger

September 1984
WP-84-72

Preface

This report presents the post-processor to the MESSAGE-II energy model (WP-84-71a and WP-84-71b). The processor is a convenient tool for the manipulation of input data and output results, for the generation of tables and graphs and the organization of data flows between a number of models.

Hans-Holger Rogner
Leader
International Gas Study

CONTENTS

1 INTRODUCTION

The post processor of MESSAGE II, CAP, is a Calculator Program designed to handle data extracted from various data files and/or the flow of information between several computer models. All data processed by CAP can be documented in generalized tabulated forms. Moreover, CAP can be used to generate graphic output, however, this feature is dependent on the software available on the computer used and the code has to be adapted accordingly.

One of the most important features of CAP--when used to handle a set of models--is, that it does not only allow to transfer data between the single moduls, but can also handle the results of all moduls and/or more scenarios at one time. Thus this program is especially useful for checking the consistency of scenarios described by a set of models.

In order to make such checks as effective as possible CAP can also be used as an interactive program. In this way all data of interest can be extracted from the according input and output files. For documentation of data, however, CAP will usually be used in batch mode, as in most cases a great number of mathematical expressions will be evaluated to produce a sufficient documentation.

The calculating instructions used by CAP follow standard mathematical notation. In addition, some extensions have been made to account for certain model specific requirements. They include the calculation of growth rates or of normalized time series, for example. Besides, two interpolation routines (natural cubic spline and linear interpolation) are available, permitting the use of files with time periods of different length.

Besides the detailled description of CAP, this user's guide contains also the description of other supporting programs useful for the evaluation of results obtained with MESSAGE II.

The last of the attached appendices containes a complete example for input and output files that is consistent with the example used in the "User's Guide for the Matrix Generator of MESSAGE II" (S. Messner, IIASA 1984).

1.1 Introduction to the Application of CAP

Cap requires two input files for controlling the data handling. The first of those files, called Control Variable File thereafter, is used to indicate to the program which of the data files are to be used for a certain application. The second one, called Calculating Instruction File, contains information on how to process the data extracted from one of the data files assigned to CAP. For this purpose a variable name identification scheme was developed that enables the program to identify on which of the data files the indicated data can be found. That scheme is described in detail in Section 6 of this user guide. As CAP was especially designed towards the handling of time series, each of the variable names used in the data handling instructions represents actually a time series. The data handling instructions are described in subsection 3.2.2, but the format can best be seen from the examples given in chapter 8.

The following table serves as a quick guide to the files and units (i.e. the numbers of logical files used in CAP). A detailed description of the file contents and formats is given below.

Input Files

unit	3:	Control variable file (formatted, sequential)
unit	5:	Calculating instructions file (formatted, sequential)

Data files

unit	2:	MESSAGE II technology name file (only if CAP is used in interactive mode) (formatted, sequential)
unit	4:	Preprocessed LP solution file (unformatted, direct access)
unit	8:	MEDEE-2 dump file (unformatted, sequential)
unit	9:	Information on the LP solution file (unformatted, sequential)
unit	10:	MESSAGE II technology dump file (unformatted, sequential)
unit	13:	Supplementary file with time series (formatted, sequential)
unit	14:	MESSAGE II time series dump file (unformatted, sequential)
unit	16:	Variable location in MEDEE-2 dump file (formatted, sequential)
unit	17:	MESSAGE II user defined constraints dump file (unformatted, direct access)
unit	18:	(upwards) additional preprocessed LP solution files (unformatted, direct access)

Output files

unit	1:	tabular output
unit	6:	error messages
unit	7:	intermediate file for plotter program;
unit	12:	tabular output for interactive mode

Intermediary files

unit	11:	interpolated time series
unit	15:	input format conversion

2 DESCRIPTION OF INPUT FILES

2.1 Control Variable File (Unit 3)

This file contains some general information such as switches to be set for plotting, interpolation or extrapolation of time series and ones indicating the input files to be used. The file gives also one line of the header written to an output file, which can be used to indicate e.g. the name of the region or country for which tables and/or plots are to be produced. Usually one of these files can be be used together with several calculating instruction files.

The file is divided into three information blocks, each beginning with a new line. Within those blocks free format can be used, i.e.,

-- Numbers and strings are separated by at least one blank, and can be written in an optional number of lines;

-- comma (,) indicates repetition of the previous string;

-- semicolon (;) indicates that any information written behind it is ignored;

-- blank lines are ignored.

The first block contains 16 integer numbers. The first 11 yield information on the files to be processed and outputs to be produced. The remaining 5 integers comprise the number of time periods required in the output file, a switch for the output format required, and 3 control integers for plotted outputs. The second block gives labels and lengths of time periods to be considered. The third block contains, if required, the output format statement and the first line of the title of the output file.

Throughout this guide all entries described are identified by E(x,y,z) with

x number of the logical file containing the entry;

y number of the block in a file;

z number of the entry in a block.

2.1.1 Content of blocks

Block 1

E(3,1,1)		plotter control; see also comments on entry E(5,2,2), and 3 DESCRIPTION OF OUTPUT FILES, for possible table formats;
	0	for generation of tables only;
	1	for generation of tables and an intermediate file containing information to be transfered to the plotting program CPLOT (see section 7 for a description of that program);
	-1	as 1, but labeling of plots will be suppressed.
E(3,1,2)		output record length control;
	0	record length set to a maximum of 132 columns;
	n	record length set to a maximum of n columns.
E(3,1,3)		error message control;
	0	messages are written to unit 6 only;
	1	messages are also written to unit 1 (tabular output); messages concerning errors which result in program termination are written to both files in any case.
E(3,1,4)		interpolation control; if the period lengths of the time series read from the data files differ from those given in block 2 of this file, they are interpolated before being used further;
	0	no interpolation allowed;
	1	for spline interpolation;
	2	for linear interpolation.
E(3,1,5)		extrapolation control; if the time horizon of time series read from the data files is shorter than the time horizon required for the output (as indicated in block 2 of this file), they are extrapolated before being used further;
	0	no extrapolation allowed;
	1	for simple repetition of the last value;
	2	to invoke linear extrapolation routine.
E(3,1,6)		MEDEE-2 dump file control;
	0	if no data are required from the MEDEE-2 dump file;
	n	to open unit 8 for a file containing n MEDEE-2 dump files.
E(3,1,7)		MESSAGE dump file control;
	0	if no data are required from a MESSAGE dump file;
	1	unused;
	2	to open units 10, 14 and 17 for MESSAGE II matrix generator dump files;

E(3,1,8)		unused
E(3,1,9)		unused
E(3,1,10)		LP solution file control;
	0	if no data required from a LP solution file;
	n	to open n direct access files obtained through the LP solution and read the number and labels of periods from a file on unit 9;
E(3,1,11)		supplementary file control;
	0	if no supplementary file used;
	1	to open unit 14 for a preprocessed supplementary file;
	2	to open unit 14 for a preprocessed supplementary file if available, otherwise to open unit 13 for a new supplementary file;
	3	to open unit 13 for a new supplementary file;
E(3,1,12)		number of time periods in the output file, the maximum number of time steps possible, is to be set prior to compilation of the program (see section 7.1).
E(3,1,13)		output formatting control; this switch is only active if E(5,2,2) is set to 3 or -3;
	0	tables are produced using the format (';',10a1,/,8(f10.3)) or, if the name of a time series is left blank, with (8(f10.3));
	n	if the format required is to be read from the first n lines of block 3 of this input file; the format statement can be up to five lines long.
E(3,1,14)		plot labeling control; the number given is used to divide the total time horizon, the result gives the number of years between the labels on the time axes of plots.
E(3,1,15)		tick mark control; the number given is used to divide the total time horizon, the result gives the number of years between tick marks on the time axes of plots.
E(3,1,16)		time step control;
	0	if the time steps required in the tabular output are equidistant, the base year and the step width are to be given in block 2 of this file;
	1	if the time steps required are of variable length, the labels for all time steps (i.e. the first year of each period) plus the the last year of the total time horizon have to be given in block 2 of this input file.

Block 2

E(3,2,1)	labels of time periods; if entry E(3,1,16) is set to zero two entries are read, the first one being the label of the base year, and the second one the length of the time steps; if set to one, then E(3,1,11)+1 entries, representing the labels of all periods plus the last year of the total time horizon, are required as input.

Block 3

E(3,3,1)	formatting; if entry E(3,1,13) is set to zero, then no entry is required here and the standard output format will be used; if entry E(3,1,13) is set to one, the required format statement with up to 400 characters in up to 5 lines is read; the format statement is to be written in the usual FORTRAN FORMAT notation and has to be able to process the name of the time series (10a1) and E(3,1,11) real variables.
E(3,3,2)	two header lines; i.e. the first line, restricted to 60 characters, is used as a header line on printed tables, the second one , which may contain a maximum of 40 characters, is used as the first line of a three-line header for tables and plots. The complete title is split into two parts so that one common line can be used for several tables and plots. For example, if the same kinds of calculations are to be carried out for different scenarios only that line has to be changed for each scenario, and this input file can be used together with a number of calculating instruction files as described below.

2.2 Calculating Instructions (Unit 5)

This file contains switches more closely related to a given set of calculating instructions than those described under 3.1 as well as the actual instructions. The file contents are again divided into three blocks.

The first input block in this file contains the second and third lines of the three-line header. (The first is contained in the control variable on file unit 3.) The second block contains switches, information on the units in which tables and/or plots are to be labeled, as well as scaling factors. This block has to be ended by a commercial add (@) written to an extra line, separating in this way the second and the third input blocks from each other. The third block contains the actual calculating instructions.

2.2.1 Content of blocks

Block 1

This block contains two lines serving as second and third lines of the header written on tables and plots that are produced from this input file. Each header line is restricted to 40 characters.

Block 2

E(5,2,1)		plotted output control; the file (on unit 7) containing the information for the plotter software is produced only when entry E(3,1,1) is not set to zero and entry E(5,2,2) is set to an absolute value of 1 or 2; the values transferred to the plotter software are also printed as third table in the tabular output on unit 1 (even if one or both of the plotter control switches are set to zero);
	0	for normal plotting i.e. all Y-values are plotted relative to the base line;
	1	for cumulative plotting;
	2	for plotting the shares of the sum of Y-values;
E(5,2,2)		output quantity and format control; if the absolute value of this entry is less than 3, then entry E(3,1,1) is in control if plots are to be produced or not; in all other cases entry E(3,1,1) has no influence as no plots can be produced; in this context, see also the comments on entry E(3,1,1) and DESCRIPTION OF OUTPUT FILES, for various table formats;
	0	to generate tables only;
	1	to generate tables and plots, curves are interpolated;
	-1	as 1, but allows for fixed labeling of the Y-axes of plots; see also the description on entry E(5,2,9) below;
	2	as 1, but all curves are plotted as step functions;
	-2	as 2, but allows for fixed labeling of the Y-axes of plots; see also the description on entry E(5,2,9) below;
	3	to print transposed output matrix to simplify further processing of that file, in this case all lines containing text will be preceded by a semicolon; thus the resulting table can directly be used as input file to MESSAGE II, which interpret a semicolon as the begin of a comment.
	-3	as 3, but lines that contain only zeros are not printed;
	13	as 3, but adds one line containing the sum over all time series shown in that table;
	4	to print the number of the variable in the LP solution file and its status in extra columns;
	-4	to print the status of the requested variable in the LP-solution;

E(5,2,3)		labeling control;
	-1	no labeling of tables and plots;
	0	to label outputs 'MESSAGE II';
	1	to label outputs 'MESSAGE II - NL';
	2	unused;
	3	to label outputs 'MESSAGE II - MIP';
	4	to label outputs 'MEDEE-2';
E(5,2,4)		table header control;
	-2	to suppress certain control characters written to the tabular output file; these control characters are only meaningful if the output file is to be used as input file to CAP later;
	-1	to suppress the header lines, date, and time written into the table header, this is useful when the output file produced is to serve as an input file to another program;
	0	all labels are written to the tables.
E(5,2,5)		units of variables in outputs; this entry has no influence on the calculations as the program does not know anything about the units of the values read from the various input files, it is up to the user to convert the results to the unit required; the string may have up to 8 characters where blanks have to be padded by a tilde ( ); this string appears in the first (or only) table produced and/or as left-hand side label of plots; if no unit name is to be written to the output files, at least one tilde must be present indicating that entry.
E(5,2,6)		units of variables in outputs; this string is written to the second table and as right-hand side label on plots, again this entry has no influence on the calculations; if the first character of this string is a percent sign (%) the second table contains the shares of each time series, but the plotted output remains unchanged and its right-hand side is labeled with the string following the percent sign immediately; as above, at least one tilde has to be entered to indicate the entry.
E(5,2,7)		multiplicative conversion factor between the two units given in the previous entries, (Unit defined in E(5,2,5) * E(5,2,7) = unit defined in E(5,2,6)).
E(5,2,8)		multiplicative scaling factor for all values; all time series calculated within CAP are multiplied by that factor.
E(5,2,9)		upper label on left-hand side of plots, which is used for fixed labeling if E(5,2,2)=-1; (this entry can remain in the file even if unused); this option helps to produce plots with identical labeling so they can be easier compared with each other.

E(5,2,10)	end sign, this is a commercial add (@) which is to be entered in the first column of a separate line.
If E(5,2,2)	is set to -2 additional input is required as control for step functions; the first entry is to be the number of curves plotted as step functions the following ones are the numbers of the according curves which are derived from the consecutive number of the mathematical equation used in this file to produce the curve; curves not included in that sequence in are plotted as continuous functions.

Block 3

This block contains the actual calculating instructions applied. The format to be followed when setting up these instructions is explained in the next subsection. An example file illustrating the format specifications is included in Section 8. The number of mathematical expressions to be evaluated is limited to a number which has to be set via the dimensioning program CHDIM prior to compilation of CAP (see chapter 7.1).

2.2.2 Format of mathematical operations

The calculating instructions are to be written as mathematical equations of the format:

X = mathematical expression,

where X is the chosen name of the time series (with up to 10 characters) containing the result obtained by the mathematical operation. The mathematical expression is build up using standard mathematical notation and can be extended to more than one line by typing an ampersand (&) or a backslash (\) as last character in that line. The number of continuation lines possible has to be set in a data statement prior to compilation of the program (see section 7.1). The operands are the names of time series as they appear in the input files used. They are appended by an identifier (:id) defining for the program the input file containing the time series in question and, in cases when the given name refers to a set of data, additionally which of the time series is to be taken from that set. For further instructing the program what operation to perform on a time series, these names can be appended by one ore more operators (in the format ':operator'). These operators can be appended to all mathematical expressions valid in CAP, i.e., time series identifiers, parenthesis, constants or internal time series. For more information on identifiers and operators, see 6 Description of Variables and, in particular, 6.8 Operators.

CAP knows the basic mathematical operators +, -, *, /, ** and parenthesis () as well as their hierarchy. As the mathematical expressions are only evaluated when a closing parenthesis is found or the end of the expression is reached, one has to take care not to use too long expressions without any parenthesis. The maximum number of nested parenthesis and the maximum number of operands enclosed in parenthesis can be set prior to compilation of the program and thus be adapted to the specific requirements of the user (see section 7.1).

In addition, both constants and time series can be used by following the format requirements given below,

Constants — can be either used directly in a formula (identified as figures in square brackets []) or by using a stack. The stacking of constants is initiated by a ↑ in the first column. Each variable to be stacked has to be written in the format
STOii value
ii being the stack number (up to 20). The end of a stacking sequence has to be indicated by ↑ ↑. The use of a stacked constant is indicated by RCLii: which is used like a usual variable in a mathematical expression;

Time series — are identified by ↑n, with n being the numbers of the time series. This shorthand for 'read n-th time series' is to be used again like a normal variable in a mathematical expression. The time series themselves have to be entered one by one in the lines following the equation where they are used first;
if one of the time series already defined is to be used in another equation it can then be referred to by typing ~n, with n being the number used when that time series was entered; if ↑n with an n number used earlier is used again in another equation, the previous time series with that number will be overwritten; the number of entries per time series used must correspond to the number given in entry E(3,1,12); the maximum number of internal time series that can be handled by the program is to be set in a data statement prior to compilation (see section 7.1).

2.2.3 Control characters

The program recognizes a few control characters, provided they appear in the first column of a line.

#	indicates to the program to ignore that line;
"	the rest of the line will be copied to the tabular output preceded by a '#' in the first column. So the output can, if entry E(5,2,2) was set to 3 or -3, be again used as an input file to CAP.
'	this line is copied to the output without any control character in the first column.
@	end of input.

–	after '@' allows to copy the lines following that control character to the end of the tabular output file. The same control characters as described above are to be used to identify the lext lines. The text has to be ended by an '@' and one of the control characters described below. If E(5,2,2) is set to +/-3 this option has no effect as any lines preceeded by text identification control characters are copied to the output according to their position in the input file. In all other cases however all such lines are, if this option is not used, typed just after the header.
+	after '@' indicates that another calculating instruction file follows on the same input file. The resulting tables are separated by a '1' in the first column (FORTRAN printer control character for new page). The plotting information is also written to a single file and identified accordingly. It is recommended to use this option if more tables should be produced as it saves the repeated loading of the MESSAGE II dump files.
@	after '@' indicats 'END OF FILE'.

3 DESCRIPTION OF DATA FILES

3.1 MESSAGE II Technology Name File (Unit 2)

This file contains the names of and descriptions for all technologies, user defined constraints and energy forms included in the MESSAGE II input file. The infomation contained in that file is only used when CAP is used in interactive mode.

3.2 MEDEE-2 Dump File (Unit 8)

This dump file holds some general information and all the time series being inputs to or outputs from MEDEE-2. The information is stored in (1+2*NCAL) unformated FORTRAN records, with NCAL being the number of time steps for which MEDEE-2 was run.

The first record contains the following variables:

REGION	(character*8)	name of region or country;
CASE	(character*4)	number of scenario;
NCAL	(integer)	number of years;
YCAL(1:NCAL)	(integer field)	list of these years;
UNAME	(character*4)	MEDEE-2 output energy units.

Each of the following NCAL blocks consists of two FORTRAN records, the first containing

PARVAR(1:231) (real field) inputs to MEDEE-2;

and the second

DERVAR(1:206) (real field) outputs from MEDEE-2.

3.3 LP Solution File (Unit 4)

The actual LP solution file has to be preprocessed before being used as input to CAP. This preprocessing is performed by program RDSOL, that combines the results obtained for each time series into one record that are then sorted alphabetically to allow the use of an efficient search algorithm in CAP. The resulting file is written as an unformatted direct access file. For reasons of comparing different LP solutions or for aggregating various solutions it is possible to use up to 8 different solution files, where the files 2 to 8 are to be assigned to units 18 upwards. See chapter 6.4 on how to extract data from these files. For a more explicit description of RDSOL see chapter 7.2.

Besides the data file(s), another file (assigned to unit 9) is needed forwarding information on the content of the direct access file(s) file to the program. This file is produced by RDSOL (on unit 9) during preprocessing of the solution file. If more than one solution is being processed at the same time it is silently assumed that all files are of the same size and contain the same time series names.

3.4 MESSAGE II Dump Files (Units 10, 14 and 17)

These dump files hold a selection of the most important model inputs concerning technology data and user defined constraints. The files are created during matrix generation by program ROWS on the same logical units as unformatted, sequential access (units 10 and 14) or direct access (unit 17) file.

3.5 Supplementary File (Unit 13)

This file may contain additional time series, which are either supplied by the user or are obtained from a previous run of CAP with entry E(5,2,2) set to 3 or -3.

If the file is created by the user the following format specification have to be followed:

The file is divided into three blocks containing

-- comments,

-- number and labels of time steps, and

-- time series.

The comment block, made up of an optional number of lines, can be used to identify the data contained in that file.

The beginning of the actual data blocks is signaled by the string

NT YEARS

which must be written in the format (1x,a2,2x,a5) followed by a line containing the according figures in free format. The next line can be used to change default values on the file format. This line has to be typed using the format

(1x, a1, 1x, i2, 1x, i3, 1x, a40),

if the default values should be used, then the according entry has to remain blank. To this end, the following four values can be reset:

File content		if this entry is left blank (default) it is assumed that all of the time series names and values are read from the file assigned to the unit number specified by the next entry;
	n	indicates that no names of time series are given explicitly but that they are internally created as natural numbers in increasing order;
	a	indicates that the names of the time series are read continuing on this file (unit 13) but the time series are read from the file connected to the logical unit specified via the next entry.
Unit number		if the time series are not following below on this file then any unit number greater than 15 can be chosen, and the program continues to read from there; this option enables one to process files which are generated by any other computer program, provided the data file used satisfies the only conditions that time series are given row-wise and each one begins with a new line.
File width		the standard number is 80 columns, and the maximum 400.

Input format	the default considered is 'free format' (see description below); if the file to be read follows a defined format this can be specified here, resulting in a faster processing of that input file.

The line following the one used to reset default values is ignored by the program.

When the default input format is used the following rules have to be observed when setting up the file containing the actual set of data.

- -- The first column is reserved for control characters; number sign (#) indicates that this line is to be ignored and a commercial add (@) indicates the END OF FILE; but can be omited at the physical END OF FILE. All other characters in the first column are ignored.
- -- All variable names and values have to begin in or after column two.
- -- All values are assumed to be represented in real format. If integer numbers are used the input format has to be stated explicitely as Fn.0.
- -- All entries have to be separated by an optional number of blanks.
- -- Each time series has to start in a new line, but may continue over an optional number of lines.

3.6 Variable Location in the MEDEE-2 Dump File (Unit 16)

This file contains the names of all the variables used in MEDEE-2 and an index representing their location in the dump file produced by MEDEE-2. The location is defined by the number of bytes preceding a variable. Variables are ordered in the sequence in which they appear in the common blocks PARVAR (parameter variables) and DERVAR (derived variables) of the MEDEE-2 program. For a list of the variable names see Kahn, A., A. Hölzl; 'Evolution of Future Energy Demands...'.

The names of time series given in the calculating instructions file (on unit 5) are compared to those listed in the variable location file; and the given index enables the program to locate the respective time series in the MEDEE-2 dump file (on unit 8). These locations remain the same, unless the MEDEE-2 program is changed.

Retrieval of a well defined group of variables can be speeded up by setting up a new variable location file containing the names and locations of these variables only. This procedure is recommended if certain variables are frequently to be extracted from various dump files.

4 DESCRIPTION OF OUTPUT FILES

4.1 Tabular Output (Unit 1)

For E(9,1,1) * E(5,2,2) set to 0, +/-1, or +/-2

Up to three tables are produced. The first one is labeled with the unit given in the calculating instructions file (unit 5) by entry E(5,2,5). The second one gives them in the unit set by entry E(5,2,6) and using the conversion factor given as entry E(5,2,7). The values in the third table are those also used for plotting, following the instructions given by entry E(5,2,1). Tabels that would show identical figures are suppressed. An input file to the plotter routines (on unit 7) is produced only if the absolute value value of entry E(3,1,1) is set to 1.

For E(5,2,2) set to +/-3

Setting this switch to 3 results in a transposed matrix, which usually simplifies processing of the file produced. (Time series are written as rows rather than as columns.) The format of the file is such that it can again be used as an input file to CAP (see also description of unit 13 above). Each text line is preceded by a semicolon (;), in this way such a file can also directly be used as an input file to MESSAGE if, e.g., final or secondary energy demands are calculated from MEDEE-2. If the entry is set to -3, then all time series containing only zero values are not printed.

For E(5,2,2) set to 4

This table format can be used to check LP solutions for upper or lower limits on variables or rows and to get the number of the row or variable in question by that it is identified in the LP solution. The format is similar to that of the first table described above, but an additional column containing the number is printed before and one showing the status of the row or variable is printed behind the column containing the values required. The status is expressed as up, (at upper bound), lo (at lower bound), or bs (in basis). If a mathematical expression is used to calculate a time series it shows the number and status of the last variable in that expression.

For E(5,2,2) set to -4

While the format is similar to that obtained through E(5,2,2)=4 the column showing the number of that variable in the LP solution file is not printed in this case.

4.2 Graphic Output

As the program is set up now it produces (if entries E(3,1,1) and E(5,2,2) are set up accordingly) an output file on unit 7 containing all relevant information necessary for the plotter interface programs. The file is written in formatted records, in order to allow for a device independent postprocessing. The postprocessing; i.e.; creating an input file matching the format requirements of the plotter software used; is done with a program called CPLOT. This program is now written so that it matches the software used on VAX/IIASA, although the commands used are very similar or even similar to the ones used in CALCOMP software and is therefor easily to be adjusted to the required standards. The appendix to this document contains the description of the subroutines called from the plot library.

Another program called SPLOT can be used to produce simple graphical output on a line printer or a terminal.

The programs produce the image for one plot whose Y-axes is labeled with the two units specified. Additionally the time series names are numbered and written to the right of the plot frame containing the plotted time series. If entry E(3,1,1) is set to -1 all additional information, i.e., header and time series names are suppressed. See section 7 for a more detailed description of programs CPLOT and SPLOT.

5 DESCRIPTION OF INTERMEDIATE FILES

5.1 Interpolated Time Series Storage (Unit 11)

This file ia a prerequisite for CAP being used with E(3,1,11)=1. If E(3,1,11) is set to 2 and the file has zero length the values of the time series in the file on unit 13 are interpolated, as necessary, and written to unit 11. If E(3,1,11) is set to 1 it is assumed that the interpolation was performed in a previous run, and any values requested from the supplementary file are taken from the file on unit 11 instead from the one on unit 13. Setting E(3,1,11) to 3 indicates that the file on unit 13 is to be used as input file and is to be processed as required.

5.2 Input Format Conversion (Unit 15)

This file is an intermediate file, used for converting data given in free format to ones written in usual FORTRAN formats.

6 DESCRIPTION OF VARIABLES

Generally all variable names used in the calculating instruction file represent names of time series. If the values identified by a variable name is a single constant it is automatically expanded to a constant time series.

All variable names are written in the format below.

VARIABLE:DESCRIPTOR[:OPERATOR].

With VARIABLE representing the time series name to be searched for in one of the input files; DESCRIPTOR identifying the file of residence of the variable, and providing for a more exact variable specification if there exist more time series relating to the same variable (e.g. capital cost, efficiency, etc. related to a specific technology included in MESSAGE). In the cases of certain input files it is also possible to process time series for the same variables contained in different solution files; e.g., for variables read from the MEDEE-2 dump files, and for those read from LP solution files preprocessed by program RDSOL. The construction of variable names and their descriptors is explained in some detail in the following subsections. The description of OPERATORS is given in Subsection 6.8.

6.1 Variables Read from the MEDEE-2 Dump File

format: name[:sn.]:descriptor

name	variable name as used in the MEDEE-2 program, see Khan A., A. Hölzl; 'Evolution of Future Energy Demands..' for a full list of the variable names.	
:sn.	is used if more than one solution is processed at one time for aggregation purposes, etc.; in this case the dump files produced by several of MEDEE-2 runs just have to be combined into a single file; the dot (.) has to be set to the number of the solution file in question if a time series is to be extracted from one of the files; if it is set to zero, or the entry is omitted altogether, then the sum over the time series in question is calculated using all MEDEE-2 solution files available;	
:descriptor	med	to be appended to indicate that the time series is contained in a MEDEE-2 dump file.

6.2 Variables Read from the MESSAGE II Dump Files

6.2.1 Technology variables

Format: name:descriptor

name	for conversion technologies: 4 character variable name constructed according to the MESSAGE II user guide; the first character is the energy level identifier; the second one the main energy input identifier; the third one the additional technology identifier; and the fourth character the main output identifier;
	for storage technologies: 3-character variable name constructed according to the MESSAGE II user guide plus control characters; the first character is the level identifier, the second one the identifier of the fuel to be stored, the third one the additional technology identifier, the fourth character in this field has to be set to dot (.), the fifth character is to be set to **g** when referring to values related to the input/output part or to **v** to indicate values related to the volume part of the storage device; it remains unused (but can be set to **g**) when a storage device with a fixed relation between generation and volume parts is represented by this technology.

:descriptor

cap	capital cost per unit of total output;
var	variable operation and maintenance cost per unit of total output;
fix	fixed operation and maintenance cost per unit of total output;
ctm	delivers a value representing the multiplicative factor necessary for converting cost per unit of total output to cost per unit of main output (i.e., name:cap * name:ctm gives the capital cost per unit of main output);
pll	technical plant life;
plf	availability factor;
eff	main output to main input efficiency;
lev	calculates levelized cost of technology using the technical plant life as economic lifetime and setting the long term discount rate equal to the interest rate;
dec	parameter describing the decay of storage contents;
rgv	relation between generation and volume capacities of storage devices;

each of the following descriptors is valid for a group of entries and delivers inputs to or outputs from the technology relative to one unit of main input or one unit of installed capacity respectively, the dot (.) has to be substituted by the identifier given in the MESSAGE II input file used to produce the dump file,

ei.	energy input, if the dot (.) is set to star (*) instead of a specific fuel identifier it delivers the sum over all energy inputs;
eo.	energy output, setting dot (.) to star (*) results again in the sum over all outputs;
ri.	nuclear fuel requirement;
ro.	nuclear fuel retirement;
fr.	fraction of total input or output of the technology in the load region indicated (dot (.) set to the number of the load region), when the load duration pattern of a technology was fixed;
rn.	coefficient in user defined constraints, where r stands for 'o' or 'c' indicating a relation to one unit of output or one unit of new installed capacity respectively and n stands for the type of the constraint (1 or 2); the dot is used for identifying the number of loadregion this value is assigned to (set to dot (.) if the coefficient is not defined for a specific load region); the name of the constraint in question has to be added as ':name';

6.2.2 User defined constraints

Format: name:descriptor

name	name of the user defined constraint	
:descriptor	in.	coefficient in user defined constraints, where i stands for 'r' or 'v' indicating the right hand side of the constraints or the value of the objective function coefficient implied on all row entries and n stands for the type of the constraint (1 or 2); the dot is used for identifying the number of loadregion this constraint is defined for (set to dot (.) if the constraint is not defined for a specific load region or to get the sum over all load regions);

6.2.3 Demands

Format: name:loadregion

name	dm.	where the dot has to be set to the identifier of the demand sector in question.
:loadregion	lr.	gives the fraction of the demand falling into the load region indicated, if dot (.) is set to t this results in the total demand.

6.2.4 Other variables

Format: name:

lr.	length of load region specified in dot (.) as fraction of one.
lp.	length of period specified in dot (.) as given in entry E(3,2,2); if the dot is set to a specific period identifier (in this case represented by a letter rather than a number), then the whole time series is set to the value of the length of the period indicated; if the dot is set to zero the time series will contain the set of the period lengths as given in entry E(3,2,2).

6.3 Variables Read from the LP Solution File

Format: name[:sn.]:descriptor

name	first 7 characters of the 8 character time series name in the LP solution file where trailing dots can be omitted; with the character identifying the number of the load region in question (fifth character when MESSAGE I is used, or seventh character in the case of MESSAGE II), set to 0, the program calculates the sum over the load regions; otherwise the time series relating to the load region in question is extracted.
:sn.	is needed if more than one solution is to be processed at the same time for purposes of aggregation, etc.; the number of the file to be chosen is the consecutive number of the file in the order they are assigned to the unit numbers, where the first file is to be assigned to unit 4 and the rest is assigned to units 18 upwards; the dot (.) is to be set to the number of the solution file in question if a time series is to be extracted from one of the files; if it is set to zero, or the entry is omitted altogether, then the sum over the time series in question is calculated using all LP solution files;
:descriptor	the descriptor is used to identify the column in the LP output from that the time series is to be extracted;

act	activity column,
sla	slack (in rows section) or discounted objective function value (in columns-section),
obx	the objective function value is undiscounted relative to the center of the periods (to be used for all variables excluding capacities),
oby	the objective function value is undiscounted relative to the beginning of the periods (to be used for capacity variables),
lol	lower limit,
upl	upper limit,
dis	shadow price discounted,
shp	shadow price undiscounted.

6.4 Variables Read from the Supplementary Input File

Format: name:descriptor

name	variable name as given in the supplementary input file.
:descriptor fil	has to be appended to the variable name to indicate that the time series is to be read from the supplementary file.

6.5 Use of Internally Stored Values

The results of a given number of calculations are stacked and can be used in the following calculations by using the left-hand side name of the equation referred to. The number of time series to be stacked is to be set prior to compilation of the program, see section 7.1 on how to set that value.

Format: name:

The search procedure for the internally stored time series follows the 'last in - first out' principle.

6.6 Operators

Operators can be appended as ':operator' to any type of operand used in a mathematical expression; i.e., variable names of all types described above, internal constants and time series and closing brackets. It is also possible to add more than one operator when separated by colons. They are applied to the time series in their sequential order. Only the final result following the completion of all transformations appears as output. As some of the operators just copy values from one field element to another (e.g. :sh+, :sh-, :ex1) care has to be taken when time series with unequal time steps are used.

Format: :operator

:ind	the time-series is divided by the value of the first period (normalized);
:exp	calculates the exponential value of each member of the time series, (value(i) = exp(value(i));
:grp	calculates the total growth rate per period (the value for the last period is set to zero);
:prp	calculates the growth rate in percent per period (the value for the last period is set to zero);

:gra	calculates the annual average growth rate per period (the value for the last period is set to zero);
:pra	calculates the annual average growth rate in percent per year (the value for the last period is set to zero);
:prd	multiplies the values in the time series by the length of the period, thus allowing to calculate e.g., the total new installed capacity per period from the values given as annual averages (as usually the case in time series taken from LP solution files derived from matrices produced from either of the MESSAGE models);
:cum	calculates the cumulative value of the time series up to the end of each period, and thus can be used e.g., to calculate the total resource extraction; this operator can for instance also be used to calculate exponential growth functions; e.g., the expression [1.03] ** ([1.]:cum - 1p0:) delivers 1.000 1.159 1.344 1.558 2.094 if entry E(3,2,1) was set to 1975 1980 1985 1990 2000;
:log	calculates the natural logarithm of each member of the time series;
:lnp	calculates the natural logarithm of the quotient var(t+1)/var(t) (the last value of the resulting time series is set to one);
:lnt	calculates the natural logarithm of the quotient var(t+1)/var(1) (the last value of the resulting time series is set to one); these last two options are very useful for calculating elasticities;
:dif	calculates the difference between field elements, i.e., 1st element = 2nd element - 1st element, where the last element is set to zero;
:#nn	fills the whole time-series with its nn'th element; (e.g.; use '# 2', '#+2' or '#02' when referring to the 2nd element)
:0nn	sets the nn'th element of a time series to zero (0), all others remain unaltered;
:1nn	sets all elements but the nn'th element of a time series to zero (0);
:sh+	shifts the whole time series one element to the right, the first element is set to zero.
:sh-	shifts the whole time series one element to the left, the last element is set to zero.
:ex1	fills the last plus one element of a time series with the last element;

:ex2 fills the last plus one element with a value calculated by linear extrapolation from the last two values; this or the former operator can be used to avoid zeros written per default to the last field element e.g. when differences (:dif) or logarithms of quotients (:lnp, :lnt) are calculated; (the sequence of operators: :dif:sh+:ex2:sh- can be used to calculate the differences between the values for each time step and to get a linear extrapolation for the last time step, which is set to zero by the operator :diff);

:new this operator allows for a rough calculation of annual new installations on the basis of energy flows; if this operator is used four variables are read in free format from the first line after the formula including this operator; if it is used more than once in one formula a new line is required for each occurrence; these four variables are the plant life (to be given as integer variable), the plant factor, the energy flow in the base year and the historic growth rate of the flow (all to be given as real variables); using this information and the time series of the annual flow for each period, the program then calculates the annual new capacities needed to replace the ones going out of operation and the ones necessary to meet additional demands; this operator is usually used together with MESSAGE I, where the resource extraction and the transportation of energy carriers is not modeled explicitly;

:nii with ii being the plant life, also calculates the annual new installations on the basis of annual energy flows; the difference to :new being, that here the existing capacity is calculated as the annual flow in the first period divided by the plant life based on the assumption that the build-up of this capacity was equally distributed to the previous years; this leads, in the case of increasing growth rates for new installations during the periods prior to the modeled time frame to an overestimation of the needs for new installations during the first few periods of the modeled time frame; the resulting time series has to be divided by the plant factor explicitly.

To obtain the *total installed capacity* of a technology, it is necessary to calculte it explicitly by adding the slack of the capacity equation to the actual output divided by the length of the load region and dividing the result by the plant factor. See chapter 8.1.2 for examples for technologies with and without load regions.

7 SUPPORTING PROGRAMS

This section describes how to use the program needed to change the dimensions of CAP to the required size and how to run the programs necessary to convert output files from other programs to a format which can be interpreted by CAP.

7.1 The Dimensioning Program CHDIM

This program is common to all codes related to the MESSAGE II model and is used to adapt the array dimensions of the codes to the specific requirements of an explicit application. All codes handling the same type of information have to contain the same data statements and common blocks with arrays being of equal size for a complete model run. CHDIM works on blocks of information in the FORTRAN codes. The beginning of such a block is identified by three COMMENT lines, where the second line contains a four character block name. The end of the block is signalled by one more empty COMMENT lines. Thus such a block has the following pattern:

c
c *name*
c
Block of information to be changed, e.g.; dimension statements, common blocks or data statements.
c

Program CHDIM reads three switches from the standard input (unit 5). The first switch tells CHDIM under which operating system the programs are running. Currently there are two options, i.e.; U for UNIX or C for Control Data operating systems NOS or NOS/BE, but these options can easily be extended to other operating systems if necessary. The next two switches control the performance of the program where the first one is related to dimensions of arrays common to all programs and the second one to dimensions of arrays contained in CAP only.

If the according switch is set to zero, the program tries to read the required dimension from files assigned to unit 4 (in the case of dimensions relevant to all programs) and unit 3 (in the case of dimensions relevant to CAP only). If no input file is found, then the program uses default values when writing the new dimension statements. Here only the second input file is described in detail, the description of the first file can be found in Messner S.; 'User's Guide for the Matrix Generator of MESSAGE II'. The input file read from unit 3 contains the following variables (default values are shown in parenthesis):

ntm	- maximal number of time steps (13);
ncols	- maximal number of time series stored (30);
nstac	- maximal number of internal stacs (30);
nzr	- maximal number of internal time series (10);
ikl	- maximal number of nested brackets (10);
icont	- maximal number of continuation lines (4);

The entry ncols, controlling the maximum number of time series stored, at the same time, when entry E(5,2,2,) is set other than 3 or -3, represents the maximum number of time series to be written to the output files; nstac defines the maximum number of operands in a mathematical expression without being enclosed in brackets; nzr refers to the time series defined in the calculating instruction file by ↑n as described in section 2.2.2.

If the switch is set to 1, the program writes instead of complete blocks of code, containing dimension and common statements, depending on the setting of the first switch either

include 'com*name*'

in the case of U(NIX), or

*call com*name*

in the case of C(DC). These statements are used to indicate that a file called 'com*name*' is stored outside the program and has to be included during compilation (in the case of UNIX) or to be included when the program is extracted from an UPDATE library in the case of CDC operating systems. In case of other operating systems the write statement can be adapted easily to the specific requirements. In any of these cases (switch set to zero or one) the program to be changed is read from unit 1 and the new code is written to unit 2.

Setting the switch to -4 results in creating the blocks of information on separate files having the required names and format. This performance is required when using the UNIX operating system.

If the switch is set to -5, the program creates one file containing all blocks of information according to the format specifications required by CDC UPDATE libraries (i.e.; each block is preceded by *cd com*name*).

7.2 Program RDSOL

As indicated earlier (section 3.3) RDSOL is needed to convert a solution file produced by a commercial linear programming package to a format that can be interpreted by CAP. The present version of the program is able to process solution files produces by MINOS (Stanford University, Department of Operations Research). If other packages are used, then the according changes have to be done in SUBROUTINES FINDR and GETREC as described there. It is assumed that the solution file is written in two parts not containing any blank lines, the first block contains the rows section, the second one the columns section. SUBROUTINE FINDR is used to set the file pointer to the first row (=func), SUBROUTINE GETREC reads one line from the solution file and to set the file pointer to the first line of the column section after the rows section was read.

The program reads the solution file from unit 1. Another input file, created by program ROWS on unit 3, is read from unit 3. This file contains the name of the LP package used, that can be used as a switch to select the according READ or FORMAT STATEMENTS in RDSOL. The output file is written to unit 4 and eventual error messages are written to unit 6. Two additional output files are produced. The first one (on unit 9) contains additional information on the solution file to be forwarded to CAP. The second one (on unit 8) contains information on how the file was sorted. That file can be used as an input file on the same unit when a solution containing the same row and column names is to be sorted later on. To indicate that a file is to be sorted according to previous information the switch read from unit 5 has to be set to 'NOSORT', otherwise to 'SORT'.

7.3 Program CPLOT

Program CPLOT reads the file produced by CAP on unit 7 from unit 8. Additionally it asks interactively for some additional input from the standards input file (unit 5). The program uses subroutines contained in the IIASA plotting library (see Appendix 1 for a description). The subroutines included in this library are very similar to CALCOMP plotter software. Even if other plotter software is used it should not be too difficult to adjust the according subroutine calls.

7.4 Program SPLOT

Program SPLOT works very similar to program CPLOT. The difference being that the output represents a printable file containing a graphical representation of the results produces by CAP.

7.5 Program NBF

Program NBF allows to scan an LP solution for all active constraints. This is specially very useful for analyzing a solution if one models long energy chains or uses genarally large models. Optionally the output file can show the same values as described in section 6.3. Additionally the status (ll, ul, fx) of the row or column can be shown. The output is produced in a way that all variables and rows concerning one technology, user defined constraint or energy form are contained in one group.

For this purpose the program reads from a number of dump and intermediary files produced during matrix generation and postprocessing the solution file. The following files are required; the technology dump file (PROGRAM ROWS unit 10) is read from unit 10, the technology name file (PROGRAM ROWS unit 2) is read from unit 2, the technology dump file (PROGRAM ROWS unit 11) is read from unit 11,the resource name file (PROGRAM ROWS unit 13) is read from unit 13. Additionally two files produced by program RDSOL (see above) are read from the same unit numbers as written to by RDSOL (unit 4 and unit 9). The output file is written to unit 1.

The control variables are read from unit 3. The first line contains the identifier for the requested output in the same notation as described in section 6.3 (act, sla, obx, oby, lol, upl, dis, shp). Setting this identifier to 'at ' results in printing the status of the variables and rows. Another entry is read from the 5'th column of the first line, setting this entry to 'f' tells the program to type numerical output in F10.4 FORMAT, any other character results in using E10.4 FORMAT. The next line is used to control the program input. If the first character is set to '@' NBF reads from the file assigned to unit 2 and prints information on all energyforms, user defined constraints and techologies. If the switch is set to '+' the program requires input from unit3.

This input is to be typed using format (a1,1x,a7,1x,a20); containing an identifier showing if the required row or column is an energyform (e), a user defined constraint (a), a technology (t), a resource (r) or an entry that does not match the usual notation used in MESSAGE II (h). In any case except 'h' the program attempts to find all rows and columns connected to the root name defined in the next entry (i.e. for technologies; activity, new installed capacity, capacity constraint, market penetration). The next entries contain the column or row name (omitting the time period identifier), the identifier for the requested output (see above, if this entry is left blank, then the identifier from the first line of input is used) and optionally a variable description.

8 HOW TO GET STARTED

This section shows a few examples of input files to CAP together with the resulting output files. These examples do not cover the full set of options provided, but sketch some basic information needed to get started with CAP.

8.1 Complete Sets of Input Files to CAP

8.1.1 Calculations processing MEDEE-2 results

The Control Variable File

The file shown is typed with one entry per line only, note that this format is, as indicated in section 2.1, not the only possible one, but a number of entries could be typed also in one single line, care has only to be taken, that each of the three blocks of entries described begin with a new line.

```
;Block 1
 1    ; switch set to allow for plotting
 0    ; switch reserved for later purposes
 1    ; error messages are written to the file assigned to unit 6
      ; and to the tabular output file on unit 1
 2    ; indicates that linear interpolation is to be used
 0    ; no extrapolation of time series allowed
 1    ; to read from a  MEDEE-2 dump file on unit 8; this implies that
      ; the file containing the variable locations has to be available on unit 2
 0    ; no variables to be read from MESSAGE II dump file
 0    ; unused switch
 0    ; no input to be read from IMPACT output file
 0    ; no input to be read from a LP solution file
 0    ; no input from a supplementary input file
 8    ; number of time steps required in the output file
 1    ; output formatting control, format is set so it fits onto the
      ; printed page, the format is given in Block 3
 5      ; plot labeling
10      ; tick marks
 1      ; time steps are given explicitly
;Blocks 2 and 3
;labels of periods, format, and first header line
1975 1980 1985 1990 2000 2010 2020 2030 2040
(';',10a1/5x,8f7.1)
```

EXAMPLE 1 FOR CAP

The calculating instruction file

This file is used to prepare the demand related input file to MESSAGE II. The switch controlling the output format is set so, that the output file can directly be read by the MESSAGE II matrix generator program.

The calculations performed result in the final energy demands for specific uses of certain energy carriers and for useful energy demands that can be met by a number of different final energy forms. The final energy demands are calculated for electricity in the household/service sector and for transportation purposes (ELECTR), specific electricity demand in the industrial sector (ELIND), the demand for coal used in the transportation sector and for steel production (CLSPU) and the demand for feedstocks (FEEDST). The useful energy demands comprise the demands for low and medium temperature process heat (PH/ml) and furnace heat (PH/h) needed in the industrial sector and the demands for low temperature heat in the household/service sector (HHS/l). Parts of the demand for specific uses of electricity have to be converted from TWhrs/yr to GWyr/yr as required by MESSAGE II.

```
DEMANDS FROM MEDEE 2                  2. title line
INPUTS TO MESSAGE II                  3. title line
0     ;  not effective in this case
-3    ;  to print transposed matrix and suppress zero lines
1     ;  to label outputs with MEDEE-2
0     ;  to write all headers
GWa   ;  main energy unit
-     ;  second energy unit (not effective in this case)
1.    ;  conversion factor between the units (not effective)
1.    ;  multiplicative factor
@     ;  end of second block
#
# setting of a constant
#
↑
sto01 8.76  factor for converting TWhrs to GWyrs
↑ ↑
#
# begin of calculations
#
ELECTR  = eltr:med + elac:med + (elap:med + elsv:med) / rcl01:
LIQUIDS = tmftr:med + mfind:med
FEEDST  = feed:med
```

```
CLSPU   = coke:med + tcltr:med
ELIND   = elsind:med / rcl01:
PH/ml   = usman(1):med + usman(3):med + flacm:med * [.65]
PH/h    = usman(2):med
HHS/l   = sh:med + thserv:med + hw:med
```

The variable location file

As the variables used in this calculations are to be extracted from the MEDEE-2 dump file, it is necessary to supply an input file indicating the location of the variables. The variable location file (a part of it is shown below) contains the names of the variables used and their relative location in the dump file. See section 3.1 for a description of the file contents.

```
eltr        1408
elac        1700
elap        1496
elsv        1552
tmftr       1400
mfind       1204
feed        1200
coke        1196
```

The tabular output file

```
;TABLE 1
;MEDEE-2                                          EXAMPLE 1 FOR CAP
;date: 21. 7.1982                                 DEMANDS FROM MEDEE-2
;time: 15:18  hrs                                 INPUTS TO MESSAGE II

;nt   years
; 8   1975   1980   1985   1990   2000   2010   2020   2030

;GWa
;ELECTR
         62.1    81.5   101.0   125.0   173.0   228.5   287.5   350.0
;LIQUIDS
        423.3   529.8   636.4   740.7   949.4  1145.5  1320.5  1474.3
;FEEDST
        126.4   154.9   183.3   215.2   279.0   338.0   392.5   442.7
;CLSPU
         79.1    94.7   110.2   125.9   157.3   182.8   204.3   221.8
;ELIND
        115.4   141.9   168.4   199.2   260.8   327.4   393.9   460.1
;PH/ml
        134.2   158.3   182.4   207.2   256.7   297.8   336.1   371.5
;PH/h
        134.2   158.3   182.4   207.2   256.7   297.8   336.1   371.5
;HHS/l
        240.4   284.6   328.8   357.5   414.7   454.0   488.0   516.8
```

6.1.2 Calculations processing a MESSAGE II dump and a LP solution file

The control variable file

The control variable file shown here is, as opposed to the one in the previous section, typed using the minimum amount of lines possible.

```
0 , 1 , 2 , 2 0 0 -1 0 6 0 10 5 0
1975 10

EXAMPLE 2 FOR CAP
```

The calculation instruction file

The mathematical expressions used this input file show a number of calculations useful to interpret the results of MESSAGE II. The usual way would be to set up a number of input files; e.g., for identifying where the various primary energy carriers are used, or by which energy forms a given demand is met. Here these calculations appear rather mixed, the aim of the calculations is described in the file. As in the previous example the reader is assumed to be familiar with the set of models and, in this specific case, with MESSAGE II.

```
CALCULATIONS  WITH
MESSAGE II RESULTS
 1 1 2 0
GWa/a
Quads
0.02989
1.
@
#
# amount of natural gas imported
#
NATGAS   =  iaga:act + iagb:act
#
# amount of electricity produced in hydropower stations
#
HYDRO    = (x.1e..0:act + x.2e..0:act)
#
# total installed capacity of LWRs
#
LWRcap   = (cxuue:sla + xuue:act * xuue:eff) / xuue:plf
#
# total installed capacity of coal power plants
#   (as this plant type is modeled with alternative operation
#    modes, the activities of both have to be added)
#
COALcap  = (cxc1e.1:sla &
         + (xc1e..1:act*xc1e:eff + xc2e..1:act*xc2e:eff) &
         / lr1:) / xc1e:plf
```

```
#
# electric power required in the first load region
#
elpow1    = dme:lr1
#
# amount of coal used during each time period
#
coaluse   = (rraa:act + rrab:act):prd
```

The tabular output file

```
TABLE 1

MESSAGE II                               EXAMPLE 2 FOR CAP
date:          23. 7.1982                CALCULATIONS WITH
time:          17:17 hrs                 MESSAGE II RESULTS

GWa/a
          NATGAS     HYDRO    LWRcap    COALcap    elpow1    coaluse
 1975     185.67      9.33      0.         1.00      6.14      34.42
 1985     199.58     13.28      2.61       8.61      9.82     172.22
 1995     111.72     23.16     12.95      38.12     12.80     707.23
 2005      78.58     27.61      8.61      44.76     14.32    1115.19
 2015      50.77     31.62      8.95      55.31     15.85    1261.45
 2025      56.08     29.73      6.97      57.18     17.11    1183.83
 2035      53.25     29.79      6.29      57.78     17.50    1200.00

Quads
          NATGAS     HYDRO    LWRcap    COALcap    elpow1    coaluse
 1975       5.55      0.28      0.         0.03      0.18       1.03
 1985       5.97      0.40      0.08       0.26      0.29       5.15
 1995       3.34      0.69      0.39       1.14      0.38      21.14
 2005       2.35      0.83      0.26       1.34      0.43      33.33
 2015       1.52      0.95      0.27       1.65      0.47      37.70
 2025       1.68      0.89      0.21       1.71      0.51      35.38
 2035       1.59      0.89      0.19       1.73      0.52      35.87

          NATGAS     HYDRO    LWRcap    COALcap    elpow1    coaluse
GWa/a

 1975     185.67    195.00    195.00     196.00    202.14     236.57
 1985     199.58    212.86    215.47     224.08    233.91     406.12
 1995     111.72    134.87    147.82     185.94    198.74     905.97
 2005      78.58    106.20    114.81     159.57    173.89    1289.08
 2015      50.77     82.38     91.33     146.65    162.50    1423.95
 2025      56.08     85.81     92.78     149.96    167.07    1350.90
 2035      53.25     83.04     89.33     147.11    164.60    1364.60
```

6.2 SAMPLES FOR MATHEMATICAL EXPRESSIONS USED IN CAP

As indicated above, this section will contain only single mathematical expressions followed immediately by the result obtained by CAP. For this examples the output file produced from the first set of input files is used as input file to CAP on unit 13. To indicate how that file was produced the control variable file and the beginning of the calculating instruction file are included.

The control variable file

```
0 0 1 0 0               ;  entries  E(3,1,1)   to   E(3,1,5)
0 0 0 0 0 1             ;  entries  E(3,1,6)   to   E(3,1,11)
5 1 10 5 0              ;  entries  E(3,1,12)  to   E(3,1,16)
1980 10                 ;  entry    E(3,2,1)
(1x,10a1,5f10.2)

EXAMPLE 3 FOR CAP
```

The calculating instruction file

```
SAMPLE EXPRESSIONS

 1  3  2  -1
 -  -  1.  1.
@

'
' USE OF OPERATORS AND STORED TIME SERIES
' ======================================
'
' Total  specific electricity demand:
' totel   = ELECTR:fil  +  ELIND:fil
totel     = ELECTR:fil  +  ELIND:fil
```

The tabular output file

```
 USE OF OPERATORS AND STORED TIME SERIES
 ======================================

 Total  specific electricity demand:
 totel     = ELECTR:fil  +  ELIND:fil
 totel           177.50     223.40     269.40     324.20     433.80

 Index of electricity demand using the previous result:
 elindex   = totel:ind
 elindex           1.00       1.26       1.52       1.83       2.44
```

```
Growth of industrial process heat demand in percent per year:
PHgr     = (PH/ml:fil + PH/h:fil):pra
PHgr           1.67      1.43      1.28      2.17      0.

Linear extrapolation of previous time series
(the time series is shifted to the right by one element,
 then extrapolated and shifted back):
extr     = PHgr:sh+:ex1:sh-
extr           1.67      1.43      1.28      2.17      3.05

Cumulative use of liquid fuels up to the end of each period:
liqu-cum = LIQUIDS:fil:cum
liqu-cum    4233.00   9531.00  15895.00  23302.00  32796.00


USE OF INTERNAL TIME SERIES AND CONSTANTS
=========================================

Multiplication of the total feedstock demand with a time
series (e.g., share of feedstocks supplied by liquid fuels)
and division by a constant (e.g., the refinery efficiency):
liuq     = FEEDST:fil * ↑1 / [.92]
liqu         123.65    165.00    159.39    140.35    136.47

Use of the previously defined time series:
rest     = FEEDST:fil * ( [1.] - 1 )
rest          12.64      3.10     36.66     86.08    153.45
```

8.3 Using CAP to Produce Tables

Although the formatting options are somewhat restricted, it is possible to use CAP to produce tables in a format that can be controlled by the user. The example given here uses again the file produced in the first example.

The control variable file

```
0 0 1 0 0     ;  entries  E(3,1,1)   to  E(3,1,5)
0 0 0 0 0 1   ;  entries  E(3,1,6)   to  E(3,1,11)
5 1 10 5 0    ;  entries  E(3,1,12)  to  E(3,1,16)
1980 10  ; entry     E(3,2,1)
(10a1,'|',5f9.1,' |')

EXAMPLE 4 FOR CAP
```

The calculating instruction file

```
FILE FOR TABLES

 1 3 2 -1
- - 1. 1.
@
'
'Table xx: Useful energy demands  [GWyr/yr]
'
'----------|----------------------------------------|
'Demand    |                 Years                  |
sectors    = [1980.] + [1.]:cum - 1p0:
'----------|----------------------------------------|
ELECTRIC   = ELECTR:fil  +  ELIND:fil
LIQUIDS    = LIQUIDS:fil
FEEDST     = FEEDST:fil
COAL       = CLSPU:fil
'----------|----------------------------------------|
Final E.   = ELECTRIC: + LIQUIDS: + FEEDST: + COAL:
'----------|----------------------------------------|
```

The tabular output file

```
Table xx: Useful energy demands  [GWyr/yr]

----------|----------------------------------------|
Demand    |                 Years                  |
sectors   |  1980.0   1990.0   2000.0   2010.0   2020.0 |
----------|----------------------------------------|
ELECTRIC  |   177.5    223.4    269.4    324.2    433.8 |
LIQUIDS   |   423.3    529.8    636.4    740.7    949.4 |
FEEDST    |   126.4    154.9    183.3    215.2    279.0 |
COAL      |    79.1     94.7    110.2    125.9    157.3 |
----------|----------------------------------------|
Final E.  |   806.3   1002.8   1199.3   1406.0   1819.5 |
----------|----------------------------------------|
```

8.4 Using CAP in Interactive Mode

Using CAP in interactive mode does not differ drastically from using it in batch mode. All input files besides the calculating instructions file on unit 5 are required as described above. Units 1 (output file), 5 (input file) and 6 (error messages) should be assigned to the terminal. To signal interactive mode '$$' has to be typed as first entry. The program then asks for the scaling factor (i.e. E(5,2,8)) and if the printing option should be set. In the following the same input as usually read from unit 5 is required, the results are written to the terminal in the format usually switched by setting E(5,2,2) to 3.

To get information about row or technology names type ')?' to enter the help mode where further information is supplied. To print interesting results (line by line) use the ')p' option and the last result is transferred to the output file on unit 12.

9 THE IMPLEMENTATION ON THE COMPUTER

This chapter gives some comments on the implementation and operation of the codes described so far. To a certain extent the information given here builds upon chapter 5 of the *User's Guide for the Matrix Generator of MESSAGE II*. The first section contains the names of the subroutines belonging to each of the programs and the *common blocks* that are to be included into each of the subroutines. How these *common blocks* are to be created is described in chapter 7.2 of this report. The second section describes the logical FORTRAN UNITS used by the programs and the files associated to these UNITS.

9.1 Description of FORTRAN PROGRAMS

In the following a list of SUBROUTINE names is provided for each of the programs described. The main program name is typed in *italic* letters. On the computer tape each of these SUBROUTINES resides on a separate file. The first file on the tape contains a list with all file names in the same order as they are written to the tape.

RDSOL

is the first program to be run after the LP package. It reads the LP solution file and produces a sorted unformatted direct access file with the same information. If other LP packages than MINOS are used some statements in SUBROUTINES FINDR.f and GETLN.f have to be adjusted to the output format used in that package. The other programs are not affected by the LP package used.

Findr.f	Getln.f	Mkint.f	Mkrec.f	*Rdsm.f*
Setzf.f	Wrrec.f			

CAP

If the spline interpolation option of the program is to be used, then this program has to be loaded together with the IMS-Library. If other mathematical libraries are to be used instead of the IMS-Library then SUBROUTINES SETSP.f and DOSPL.f have to be modified accordingly. A description of the SUBROUTINES used from the IMS-Library can be found in Appendix 2 of this report.

Alin.f	Cal.f	Calc2.f	*Capm.f*	Check1.f
Cheow.f	Ciin.f	Ciout.f	Clin.f	Clout.f
Const.f	Crin.f	Crout.f	Cvar.f	Dati.f

Dolin.f	Dospl.f	Dumpi.f	Filout.f	Fixst.f
Func.f	Getv.f	Klamm.f	Loop.f	Match.f
Match2.f	Matout.f	Matpol.f	Matsto.f	Medout.f
Minout.f	Newcap.f	Nofunc.f	Plout.f	Rd3.f
Rd5.f	Rdvar.f	Rhash.f	Seelp.f	Setli.f
Setsp.f	Setup.f	Shift.f	Show.f	Smat.f
Solch.f	Status.f	Stauch.f	Tables.f	Title.f
Vari.f	Yrchk.f			

CPLOT

This program uses SUBROUTINES from the plotting library at IIASA. These SUBROUTINES are described in Appendix 1.

Cplotm.f Draw.f

SPLOT

Splotm.f

NBF

Acrow.f	Alin.f	Ciin.f	Ciout.f	Clin.f
Clout.f	Condev.f	Crin.f	Crout.f	Demels.f
Densar.f	Dumpi.f	Enform.f	Kopeak.f	Match.f
Minout.f	Nbfdum.f	*Nbfm.f*	Nbfout.f	Rhash.f
Seelp.f	Select.f	Yrchk.f		

9.1.1 List of common block names

The following list shows what common blocks are used in each of the programs. The names used are the same as the ones used in the according INCLUDE STATEMENTS in the programs. PROGRAM CHDIM will produce these files (with the same names) when switch '-4' is used. See the more detailed description in chapter 7.1.

NAME	COMMON BLOCK			
Acccom.f:	comrepo			
Acclim.f:	comrepo	combl	commat	compol
Accprc.f:	comrepo	combl	commat	compol
Acrow.f:	comsplv	comcap	commat	
Bounds.f:	comrepo	combl	comdev	comsto
	comchar			
Cal.f:	comcap			
Capm.f:	comhelp	comtext	comcap	comcapd
Cmpv.f:	commpvr	commpio		
Condev.f:	comcap	comsplv		
Cplotm.f:	complot	comcap	comtext	comcapd
Cvar.f:	comvarc	comvar		

Dati.f:	comcap			
Deman.f:	comrepo	combl		
Dolin.f:	comlin2			
Dospl.f:	comcap	comspl2		
Draw.f:	complot	comcap	comtext	
Dumpi.f:	comioda			
Dumpr.f:	comiore			
Dynco.f:	combl	comdev		
Enform.f:	comcap	comsplv		
Filout.f:	comsplf	comcap		
Func.f:	comcap			
Genin.f:	comrepo	combl	comdev	comsto
Getv.f:	comvarc	comvar		
Klamm.f:	comcap			
Loop.f:	comhelp	comtext	comcap	
Matorp.f:	commpvr			
Matorp.f:	comrepo	combl	comdev	commat
	compol	comchar	commult	comend
	comdim			
Matout.f:	comsplv	comcap1	combl	comdev
	commat	compol	comchar	comhelp
	comvarc	comvar	comdim	
Matpol.f:	commpio			
Matsto.f:	comsto	comsplv	comcap1	combl
	comdev	commat	compol	comchar
	comhelp	comvarc	comvar	comdim
Medout.f:	comsplm	comcap		
Minout.f:	comhash	comsplv	comstat	comcap
Nbfm.f:	comcap1	comchar	combl	comsplv
Nbfout.f:	comhash	comsplv	comstat	comcap
Newcap.f:	comhelp	comcap		
Nofunc.f:	comcap			
Nucfl.f:	comrepo	combl		
Plout.f:	comcap	comtext		
Rd3.f:	comhelp	comtext	comcap	
Rd5.f:	comhelp	comtext	comcap	
Rdvar.f:	comvar			
Repm.f:	comrepo	combl	comdev	comsto
	commat	compol	comdim	comchar
Resour.f:	comrepo	combl	comres	comimp
	comexp	comdim		
Rhash.f:	comcap	comsplv	comhash	
Rimex.f:	combl	comrepo		
Seelp.f:	comhash			
Setli.f:	comlin	comcap		
Setmp.f:	combl	comrepo	commpio	
Setmpt.f:	comrepo	combl	commpio	
Setsp.f:	comspl	comcap		
Setup.f:	comtext			
Setz.f:	comrepo			
Shift.f:	comcap			
Shortd.f:	comrepo	comdev		

Show.f:	comcap	comtext		
Smat.f:	commpio			
Status.f:	comtext	comcap	comstat	
Stauch.f:	comcap			
Tables.f:	comcap	comtext		
Title.f:	comtext	comcap		
Variab.f:	comdev	combl	compol	commat
	comvarc	comvar	commpvr	commpio

9.2 Running the post-processing programs

In the following section a call of a program is indicated by typing its name. The logical FORTRAN UNIT numbers together with the files associated to them are described as 'UNIT=FILENAME'. For a description of the contents and formats of the various files in use see sections 2, 3, 5 and 7 of this report and the *User's Guide to the Matrix Generator of MESSAGE II.*

RDSOL.obj	*3=SOLSIZ*	*8=SORT*	*6=RECL*	*5=input*
	9=DIRSIZ	*1=LPSOL*	*4=DIRSOL*	
CAP.obj	*1=TABLES*	*2=VARLOC*	*15=intm*	*3=TITLE*
	5=CALCIN	*17=RELDMP*	*6=err*	*8=MEDDMP*
	13=SUPFIL	*11=INTMFIL*	*4=DIRSOL*	*14=VARDMP*
	10=TECDMP	*9=DIRSIZ*	*7=PLOT*	
CPLOT.obj	*8=PLOT*	*5=input*	*6=output*	*3=CMND*
SPLOT.obj	*5=input*	*6=output*		
NBF.obj	*1=NBF*	*2=TECID*	*9=DIRSIZ*	*4=DIRSOL*
	3=NBFIN	*10=TECDMP*		

APPENDIX 1

GRAPHICS LIBRARY

This appendix contains the description of the SUBROUTINES used in CPLOT that are called from the library of graphic subroutines. The complete description of this library can be found in DESCRIPTION OF THE LIBRARY OF GRAPHICS SUBROUTINES (B. Schweeger; IIASA, Nov 1982).

NAME

gropen - initialise graphics and open graphics output file

SYNOPSIS

call gropen(file,idescr,loc)

DESCRIPTION

file	character, name of file on which graphics is to be output
idescr	integer, output, file descriptor returned by *icreat* or *iopen*
loc	integer, output, starting address in file

Gropen creates the file *file* (on the first call) or opens it for appending (on subsequent calls). *Gropen* returns the "file descriptor" of the file (in *idescr*) and the location in the file where graphics output will start (in *loc*). A file descriptor of -1 indicates that the file was not successfully opened. Subsequent calls to graphics subroutines will not produce any output. The routine also initialises scaling (1 user unit = 1 inch), puts the writing device at the origin and sets certain terminal characteristics. The graphics file **must** be closed with a call to *wendgr*.

SEE ALSO

wgraph, wendgr, wflush

NAME

fplot - move writing device

SYNOPSIS

call fplot(ipen,x,y)

DESCRIPTION

ipen	integer, pen action (raise or lower pen) parameter 0...unchanged odd...pen up even...pen down > 0...before move < 0...after move
x	real, user units, x-coordinate of destination
y	real, user units, y-coordinate of destination

Fplot moves the writing device to the specified coordinates, while the pen action is controlled by the parameter *ipen*. Example: call fplot(3,2.,4.) will lift the pen and then move to (2.,4.). It is equivalent to a call wmove(2.,4.).

SEE ALSO

penup, pendn, wmove, wdraw, wmover, wdrawr, wdline, werase, wthick, xpen

NAME

pendn - lower writing device

SYNOPSIS

call pendn

DESCRIPTION

Pendn lowers the writing device (put it onto the paper, switch the electronic beam on), without affecting its current position. The command will nevertheless not produce a dot on the display.

SEE ALSO

penup, fplot

NAME

penup - lift writing device

SYNOPSIS

call penup

DESCRIPTION

Penup raises the writing device (take it off the paper, switch the electronic beam off), without affecting its current position.

SEE ALSO

pendn, fplot

NAME

txtbox - output text with simple vector-font

SYNOPSIS

call txtbox(x,y,iref,xs,ys,ang,hsep,vsep,text,maxchr,maxlin)

DESCRIPTION

x	real, user units, x-coordinate of the reference point of the string
y	real, user units, y-coordinate of the reference point of the string
iref	integer, reference point of (x,y)-coordinates to the textbox

```
1————5————9

2————6————10   center-line
3————7————11   base-line
4————8————12

iref....=0 nothing written
iref....< 0 same as above, but one extra character
      space between box and reference point
```

xs	real, inches, size of character box in x-direction
ys	real, inches, size of character box in y-direction
ang	real, radians, rotation angle of string around the reference point
hsep	real, inches, horizontal separation (spacing) of character
vsep	real, inches, vertical separation of lines
text	character, text string to be displayed
maxchr	integer, maximum number of characters of string to be displayed
maxlin	integer, maximum number of lines of string to be displayed

The simplest way of outputting text is the use of the subroutine *txtbox*. *X* and *y* are the coordinates of one of the 12 reference points of the virtual box containing the character string. For example iref=1 means that the box is addressed by its upper left corner, iref=6 centers the character string. If *iref* is negative and if the reference point is a peripheral one (i.e. all except 6 and 7), one extra space is left between the reference point and the virtual box. iref=0 suppresses any output. *Xs* and *ys* give the size of any single character in inches, *ang* is the angle in radians of the character string. *Hsep* is the intercharacter spacing in inches, *vsep* the space between lines of *text* in inches. *Maxchr* and *maxlin* are the respective maximum number of characters and lines to be output. The array *text* contains the character string; *text* will only be output in upper case. < line feeds> are encoded as '\' (backslash). *Txtbox* always stops scanning the array *text* whenever either *maxlin* or *maxchr* are exceeded. This routine uses the file *fort.7* (FORTRAN unit 7) for intermediate storage and must therefore not be used for any other purpose.

SEE ALSO

fchar, pflag

NAME

waxis - draw axis

SYNOPSIS

call waxis(x,y,wlngth,interv,ang)

DESCRIPTION

x	real, user units, x-coordinate of start of axis
y	real, user units, y-coordinate of start of axis
wlngth	real, user units, specifies the distance between tick marks
interv	integer, number of intervals
ang	real, degrees, angle of vector with x-axis

Waxis plots a vector with tick marks at regularly spaced intervals. Tick marks are 0.1 inch long, with one half of the mark on each side, and perpendicular to, the vector. Starting point and length are in user units and the direction angle between vector and x-axis is in degrees. The parameter *interv* specifies the desired number of tick marks minus one.

SEE ALSO

fgrid

NAME

wdline - draw dashed line

SYNOPSIS

call wdline(x,y,dl1,dl2,space)

DESCRIPTION

x	real, user units, x-coordinate of endpoint of line
y	real, user units, y-coordinate of endpoint of line
dl1	real, inches, lenght of first dash
dl2	real, inches, lenght of second dash
space	real, inches, lenght of interval between dashes

Wdline draws a dashed line from the current location to the coordinates specified. The three remaining parameters permit the specification of the pattern of the dashed line in inches (not user-units): *dl1* is the length of the first dash, *dl2* the length of the second dash, *space* is the length of the interval between the dashes. The pattern is repeated periodically and is not interrupted by direction changes of the vectors.

```
.... |---------|      |---------------|      |---------| ....

         dl1      space        dl2        space      dl1

      <............ one period ............>
```

The current location is updated to the endpoint of the vector.

SEE ALSO

wdraw, wdrawr, fplot

NAME

wdraw - draw absolute vector

SYNOPSIS

call wdraw(x,y)

DESCRIPTION

x	real, user units, x-coordinate of endpoint of vector
y	real, user units, y-coordinate of endpoint of vector

Wdraw draws a straight line between the current location and the destination specified by the parameters in absolute terms; the current location is updated to the endpoint of the vector.

SEE ALSO

wdrawr, wdline

NAME

wendgr - terminate graphics

SYNOPSIS

call wendgr

DESCRIPTION

Since the device independent output is heavily buffered for efficiency reasons, it is mandatory to terminate graphics output by a call to this routine. It will ensure that the last block of information will be output correctly, then close the graphics output file (generated with *gropen* or initialized with *wgraph*).

SEE ALSO

gropen, wgraph, wflush

NAME

wmove - move absolute

SYNOPSIS

call wmove(x,y)

DESCRIPTION

x	real, user units, x-coordinate of destination
y	real, user units, y-coordinate of destination

Wmove moves the writing device to the destination specified by the parameters in absolute terms without drawing a line.

SEE ALSO

wmover

NAME

worigd - define origin in centimeters

SYNOPSIS

call worigd(xo,yo)

DESCRIPTION

xo	real, centimeters, displacement of origin in the x-direction
yo	real, centimeters, displacement of origin in the y-direction

The user can reset the origin, the point addressed by (0.,0.) in user units, by using the routines *scalf*, *worigi* or *worigd*. The parameters of *worigd* give the displacement of the origin relative to the lower left corner in centimeters, i.e. in REAL UNITS, not in user units. For example call worigd (2.,3.) will set the origin 2 centimeters away from the left margin and 3 centimeters away from the bottom of the display. The origin is at the lower left corner of the display by default.

SEE ALSO

worigi, scalf

NAME

wscald - define scaling in centimeters

SYNOPSIS

call wscald(xs,ys)

DESCRIPTION

sx	real, centimeters per user unit, x-direction scaling factor
sy	real, centimeters per user unit, y-direction scaling factor

A scaling factor of 3.5 means that each user unit corresponds to 3.5 centimeters on the real device. The default scaling is one inch per user unit, i.e. 2.54 centimeters per user unit. Scaling can be different for the x and y axis; this might produce, mainly when rotations are used, unexpected results for the unexperiences user.

SEE ALSO

scalf, wscali

NAME

xadva - move paper forward

SYNOPSIS

call xadva(n)

DESCRIPTION

n	integer, centimeters, distance to feed paper in the x-direction

Xadva, still in the library for compatibility reasons, erases the display or moves paper forward to the next page for hardcopy devices. Only for the BBC pen plotter it specifically moves the paper *n* centimeters (1 < = n < = 34).

SEE ALSO

wclear

APPENDIX 2

IMS LIBRARY

This appendix contains the description of the SUBROUTINES used from the IMS-Library. These SUBROUTINE calls are to be modified accordingly in SUBROUTINES SETSP.f and DOSPL.f if other mathematical libraries are to be used for spline interpolation.

```
IMSL ROUTINE NAME   - ICSICU

PURPOSE             - INTERPOLATORY APPROXIMATION BY CUBIC SPLINES
                        WITH ARBITRARY SECOND DERIVATIVE END
                        CONDITIONS.

USAGE               - CALL ICSICU (X,Y,NX,BPAR,C,IC,IER)

ARGUMENTS    X      - VECTOR OF LENGTH NX CONTAINING THE ABSCISSAE
                        OF THE NX DATA POINTS (X(I),Y(I)) I=1,...,
                        NX. (INPUT) X MUST BE ORDERED SO THAT
                        X(I) .LT. X(I+1).
             Y      - VECTOR OF LENGTH NX CONTAINING THE ORDINATES
                        (OR FUNCTION VALUES) OF THE NX DATA POINTS.
                        (INPUT)
             NX     - NUMBER OF ELEMENTS IN X AND Y. (INPUT) NX
                        MUST BE .GE. 2.
             BPAR   - VECTOR OF LENGTH 4 CONTAINING THE END
                        CONDITION PARAMETERS. (INPUT)
                        2.0*SPP(1)+BPAR(1)*SPP(2) = BPAR(2),
                        BPAR(3)*SPP(NX-1)+2.0*SPP(NX) = BPAR(4),
                        WHERE SPP(I) = SECOND DERIVATIVE OF THE
                        CUBIC SPLINE FUNCTION S EVALUATED AT X(I).
             C      - SPLINE COEFFICIENTS. (OUTPUT) C IS AN NX-1 BY
                        3 MATRIX. THE VALUE OF THE SPLINE
                        APPROXIMATION AT T IS
                        S(T) = ((C(I,3)*D+C(I,2))*D+C(I,1))*D+Y(I)
                        WHERE X(I) .LE. T .LT. X(I+1) AND
                        D = T-X(I).
             IC     - ROW DIMENSION OF MATRIX C EXACTLY AS
                        SPECIFIED IN THE DIMENSION STATEMENT IN
                        THE CALLING PROGRAM. (INPUT)
             IER    - ERROR PARAMETER. (OUTPUT)
                      TERMINAL ERROR
                        IER = 129, IC IS LESS THAN NX-1.
                        IER = 130, NX IS LESS THAN 2.
                        IER = 131, INPUT ABSCISSA ARE NOT ORDERED
                          SO THAT X(1) .LT. X(2) ... .LT. X(NX).

PRECISION/HARDWARE  - SINGLE AND DOUBLE/H32
                    - SINGLE/H36,H48,H60

REQD. IMSL ROUTINES - UERTST,UGETIO

NOTATION            - INFORMATION ON SPECIAL NOTATION AND
                        CONVENTIONS IS AVAILABLE IN THE MANUAL
                        INTRODUCTION OR THROUGH IMSL ROUTINE UHELP
```

Algorithm

ICSICU computes an interpolatory approximation to a given set of points by cubic splines with arbitrary second derivative end conditions.

The tridiagonal system defining the second derivatives of the spline interpolate for (X,Y) is solved, producing the spline coefficients.

This routine requires input of four boundary condition parameters, BPAR(I), I=1,...,4. There are infinitely many cubic splines which interpolate a set of points and a particular one is chosen by setting these four parameters. If one has no such information about the curve then these parameters should be set to zero (in this case a natural cubic spline is obtained).

If the second derivatives of the curve are known at the points X(1), X(2), X(NX-), and X(NX), (call these derivative values F''_1, F''_2, F''_{NX-1}, F''_{NX}, where NX is the number of data points), then the boundary parameters should satisfy

$$2.0 * F''_1 + \mathrm{BPAR}(1) * F''_2 = \mathrm{BPAR}(2)$$

$$\mathrm{BPAR}(3) * F''_{NX-1} + 2.0 * F''_{NX} = \mathrm{BPAR}(4).$$

In particular, if F''_1 and F''_{NX} are known then it is appropriate to set

BPAR(1)	=	0.
BPAR(2)	=	$2.0 * F''_1$
BPAR(3)	=	0.
BPAR(4)	=	$2.0 * F''_{NX}$

These values will force the cubic spline to have the same second derivatives at X(1) and X(NX) (i.e., let S''_1 and S''_{NX} denote the second derivative of the cubic spline at the points X(1) and X(NX), respectively. Then $S''_1 = F''_1$ and $S''_{NX} = F''_{NX}$). If the first derivatives are known at the points X(1) and X(NX) (call these derivative values F'_1 and F'_{NX}), then the values

$$\mathrm{BPAR}(1) = 1.,\ \mathrm{BPAR}(2) = \frac{6.}{X(2)-X(1)}\left(\frac{Y(2)-Y(1)}{X(2)-X(1)} - F'_1\right)$$

$$\mathrm{BPAR}(3) = 1.,\ \mathrm{BPAR}(4) = \frac{6.}{X(NX)-X(NX-1)}\left(F'_{NX} - \frac{Y(NX)-Y(NX-1)}{X(NX)-X(NX-1)}\right)$$

will cause the cubic spline to have the same first derivatives at X(1) and X(NX) (i.e., let S'_1 and S'_{NX} denote the first derivative of the cubic spline at the points X(1) and X(NX), respectively. Then $S'_1 = F'_1$ and $S'_{NX} = F'_{NX}$).

See reference:

Ahlberg, J., Nilson, E., and Walsh, J., The Theory of Splines and Their Applications, Academic Press, New York, 1967.

Programming Notes

1. The routine requires that X(I) is less than X(I+1) for I = 1,2,..., NX-1. If X is not in ascending order, the user can reorder the vector by using IMSL sorting routines. See the Chapter I Introduction for details.

Example

Input:

```
        INTEGER  IC,NX,IER,I
        REAL     X(4),Y(4),BPAR(4),C(3,3)
        IC = 3
        NX = 4
        X(1) = 0.0
        X(2) = .33
        X(3) = 1.0
        X(4) = 2.0
C                                F(X(I))=((X(I)-2.0)*X(I)+3.0)*X(I)+4.0
        DO 5 I=1,NX
           Y(I) = ((X(I)-2.0)*X(I)+3.0)*X(I)+4.0
      5 CONTINUE
C                                F'(X(I)) = (3.0*X(I)-4.0)*X(I)+3.0
        BPAR(1) = 1.0
        BPAR(2) = 6.0/(X(2)-X(1))*((Y(2)-Y(1))/(X(2)-X(1))-3.0)
        BPAR(3) = 1.0
        BPAR(4) = 6.0/(X(4)-X(3))*(7.0-(Y(4)-Y(3))/(X(4)-X(3)))
        CALL ICSICU(X,Y,NX,BPAR,C,IC,IER)
           .
           .
           .
        END
```

Output:

IER = 0

$$Y = \begin{bmatrix} 4.0000 \\ 4.8081 \\ 6.0000 \\ 10.000 \end{bmatrix} \qquad C = \begin{bmatrix} 3.0000 & -2.0000 & 1.0000 \\ 2.0067 & -1.0100 & 1.0000 \\ 2.0000 & 1.0000 & 1.0000 \end{bmatrix} \text{(spline coefficients)}$$

Thus, for T in [.33,1.), the approximation provides

$$S(T) = ((D-1.01)D + 2.0067)D + 4.8081$$

where D = T-.33.

APPENDIX 3

Sample Input and Output Files

The CAP input files reproduced in this appendix represent a set of files consistent with the problem shown in the 'User's Guide for the Matrix Generator of MESSAGE II'. The simple graphs reproduced here where produced by the program SPLOT. This program asks interactively for the setting of various values and parameters. To obtain the same result the value for the *number of lines for plot* is to bet set to 30 (default is 15) and the *plot character* is to be set to *first character of name*. All other values and parameters remain at their defaults.

The files for are listed in the following order:

- *TITLE*, the control variable file,
- *CALCIN*, the calculating instructions,
- *TABLES*, the tables produced by CAP, and
- *PLOTOUT.DAT*, the graphs produced by SPLOT.

TITLE

CALCIN

```
 1 78  1 2 0   ; plot / record length / error messages / interpolation / extrapolation
 0  2  0 0 1 0 ; 1:read from: medee / message / dummy / dummy / solution / sup.file
 4  1 10 5 1   ; no of periods / format / plot labels / tick marks / time steps
1980 1985 1990 2000 2010
(1x,10a1,4f10.2)
                    STIM
BASE CASE
```

```
PRIMARY ENERGY USE
(thermal equivalent)
 1 1 ;    0:norm 1:cum 2:sh ; 0:tbl 1:t+p 2:stepf 3:transp 4:status <0:norm
 2 0 ;    -1:nolabl 0:MII   ; -1:noheader 0:header+1b
MWyr/yr
%
1.
1.
@
Oil         = iroa:act + iaga:act + iala:act
Coal        = rrca:act + rrcb:act
Hydro       = x.he:act / xffe:eff
@
+
SUPPLY OF LIQUID FUELS
(secondary energy)
 1 0 2 0
MWyr/yr
%
1.
1.
@
'
' The columns starting with 'i-' denote imports.
'
Petrol      = aong:act * ,:eff + afcg:act * ,:eff + af2g:act * ,:eff
i-petrol    = iaga:act
Fueloil     = aong:act * ,:eof + afcg:act * ,:eof + af2g:act * ,:eof
Lightoil    = aong:act * ,:eol + afcg:act * ,:eol + af2g:act * ,:eol
i-loil      = iala:act
@
+
ELECTRICITY GENERATION

 1 1 2 0
MWyr/yr
GWh/yr
8.76
1.
@
Fueloil     = xffe..0:act * ,:eff
Lightoil    = xlje..0:act * ,:eff
Hydro       = x.he:act * ,:eff
Coal cg     = xccd..0:act * ,:eoe
@
+
PRIMARY ENERGY USE
(thermal equivalent)
 1 1 ;   0:norm 1:cum 2:sh ; 0:tbl 1:t+p 2:stepf 3:transp 4:status <0:norm
 2 0 ;   -1:nolabl 0:MII   ; -1:noheader 0:header+lb
MWyr/yr
%
1.
1.
@
Oil         = iroa:act + iaga:act + iala:act
Coal        = rrca:act + rrcb:act
Hydro       = x.he:act / xffe:eff
@
+
SUPPLY OF LIQUID FUELS
(secondary energy)
 1 0 2 0
MWyr/yr
%
1.
1.
@
'
' The columns starting with 'i-' denote imports.
'
Petrol      = aong:act * ,:eff + afcg:act * ,:eff + af2g:act * ,:eff
i-petrol    = iaga:act
Fueloil     = aong:act * ,:eof + afcg:act * ,:eof + af2g:act * ,:eof
Lightoil    = aong:act * ,:eol + afcg:act * ,:eol + af2g:act * ,:eol
i-loil      = iala:act
@
+
ELECTRICITY GENERATION

 1 1 2 0
MWyr/yr
GWh/yr
8.76
1.
@
Fueloil     = xffe..0:act * ,:eff
Lightoil    = xlje..0:act * ,:eff
Hydro       = x.he:act * ,:eff
Coal cg     = xocd..0:act * ,:eoe
@
+
SYSTEM COSTS

 1 0 2 0
M~$(80)
-
1.
.001
@
Invest   = ccap:act
O+M *)   = ccur:act + cres:act
Import   = cimp:act
'
' *) O+M costs include domestic resource
'    extraction costs.
@
+
SHADOW PRICES and
EMISSIONS
 1 3 ;   0:norm 1:cum 2:sh ; 0:tbl 1:t+p 2:stepf 3:transp 4:status <0:norm
 -1 -1 ;   -1:nolabl 0:MIl   ; -1:noheader 0:header+lbl
-
-
1.
1.
@
↑
sto01 .11416
↑ ↑
'
'
' Shadow Prices, US mills / kWh
' ----------------------------
'
'                1980      1985      1990      2000
'
' Electricity per Load Region
L.R. 1   = xe....1:shp * rc101:
L.R. 2   = xe....2:shp * rc101:
L.R. 3   = xe....3:shp * rc101:
L.R. 4   = xe....4:shp * rc101:
'
' District Heat per Load Region
L.R. 1   = xd....1:shp * rc101:
L.R. 2   = xd....2:shp * rc101:
L.R. 3   = xd....3:shp * rc101:
L.R. 4   = xd....4:shp * rc101:
```

```
DISTRICT HEAT GENERATION

 1 0 2 0
MWyr/yr
GWh/yr
8.76
1.
@
Heatplant    = xffd..0:act * ,:eff
Cogen        = xccd..0:act * ,:eff
'
'
' Note: The heatplant is fueled by heavy oil,
'       the cogeneration plant by coal.
@
+
FINAL ENERGY

 1 1 2 0
MWyr/yr
%
1.
1.
@
Electr.      = feee..0:act * ,:eff
Distr.Ht     = fddd:act * ,:eff
Lightoil     = flll:act * ,:eff
Petrol       = fggg:act * ,:eff
@
+
HEATING STRUCTURE
(useful energy)
 1 0 2 0
MWyr/yr
%
1.
1.
@
Lightoil    = ullh:act * ,:eff
Dir.Elec    = ueeh:act * ,:eff
Distr.Ht    = uddh:act * ,:eff
El.Heatpmp  = uehh:act * ,:eff
@
+
'
' POWER USE IN 1st LOAD REGION, MW
' --------------------------------
'
'                  1980      1985      1990      2000
'
Fueloil      = xffe..1:act * ,:eff / lr1:
Lightoil     = xlje..1:act * ,:eff / lr1:
Hydro        = x.he:act * ,:fr1 / lr1:
Coal cg      = xccd..1:act * ,:eoe / lr1:
'1
'
'
' POWER USE IN 2nd LOAD REGION, MW
' --------------------------------
'
'                  1980      1985      1990      2000
'
Fueloil      = xffe..2:act * ,:eff / lr2:
Lightoil     = xlje..2:act * ,:eff / lr2:
Hydro        = x.he:act * ,:fr2 / lr2:
Coal cg      = xccd..2:act * ,:eoe / lr2:
'
'
'
' POWER USE IN 3rd LOAD REGION, MW
' --------------------------------
'
'                  1980      1985      1990      2000
'
Fueloil      = xffe..3:act * ,:eff / lr3:
Lightoil     = xlje..3:act * ,:eff / lr3:
Hydro        = x.he:act * ,:fr3 / lr3:
Coal cg      = xccd..3:act * ,:eoe / lr3:
'
'
'
' POWER USE IN 4th LOAD REGION, MW
' --------------------------------
'
'                  1980      1985      1990      2000
'
Fueloil      = xffe..4:act * ,:eff / lr4:
Lightoil     = xlje..4:act * ,:eff / lr4:
Hydro        = x.he:act * ,:fr4 / lr4:
Coal cg      = xccd..4:act * ,:eoe / lr4:
@
@
```

```
'
Lightoil   = al:shp * rc101:
Fueloil    = af:shp * rc101:
Petrol     = ag:shp * rc101:
Coal       = rc:shp * rc101:
Oil        = ro:shp * rc101:
'
'
'
'
' Emissions of Central Conversion Plants
' ---------------------------------------
'
' [Million Tons per Year]
'
'                  1980       1985       1990       2000
SO2 = pSO2:act
NOx = pNOx:act
@
+
POWERPLANTS
INSTALLED CAPACITY
 1 3 ;   0:norm 1:cum 2:sh ; 0:tbl 1:t+p 2:stepf 3:transp 4:status <0:norm
 -1 -1 ;   -1:nolabl 0:MII   ; -1:noheader 0:header+lbl
- - 1. 1.
@
'
'
' INSTALLED CAPACITY OF POWERPLANTS, MW
' -------------------------------------
'
'                  1980       1985       1990       2000
'
Fueloil     = ( cxffe.1:sla + xffe..1:act * ,:eff / 1r1: ) / xffe:plf
Lightoil    = ( cxlje.1:sla + xlje..1:act * ,:eff / 1r1: ) / xlje:plf
Hydro       = ( cx.he:sla + x.he:act * ,:eff * ,:fr3 / 1r3: ) / x.he:plf
Coal cg     = ( cxffe.1:sla / xccd:eff + xccd..1:act * ,:eoe / 1r1: ) * &
                xccd:eff / ,:plf
'
'
'
```

TABLES

TABLE 1 STIM

MESSAGE II
date: 21. 7.1984
time: 15:34 hrs

BASE CASE
PRIMARY ENERGY USE
(thermal equivalent)

MWyr/yr	Oil	Coal	Hydro
1980	6265.93	123.36	683.50
1985	5774.65	217.41	872.34
1990	5212.73	383.15	973.48
2000	4117.04	1190.00	973.48

%	Oil	Coal	Hydro
1980	88.59	1.74	9.66
1985	84.12	3.17	12.71
1990	79.35	5.83	14.82
2000	65.55	18.95	15.50

MWyr/yr		cumulative	absolute
	Oil	Coal	Hydro
1980	6265.93	6389.29	7072.79
1985	5774.65	5992.06	6864.40
1990	5212.73	5595.88	6569.36
2000	4117.04	5307.04	6280.52

```
TABLE  2                    STIM

MESSAGE II                  BASE CASE
date: 21. 7.1984            SUPPLY OF LIQUID FUELS
time: 15:34  hrs            (secondary energy)

The columns starting with 'i-' denote imports.


MWyr/yr
        Petrol   i-petrol Fueloil  Lightoil i-loil
1980     892.00     0.    2958.93  1956.83  607.31
1985     940.00     0.    2879.70  1897.80  358.61
1990     990.00     0.    2740.34  1820.10  211.75
2000    1060.00     0.    2475.12  1817.65   73.83

%
        Petrol   i-petrol Fueloil  Lightoil i-loil
1980      13.90     0.      46.12    30.50    9.47
1985      15.47     0.      47.39    31.23    5.90
1990      17.18     0.      47.56    31.59    3.67
2000      19.53     0.      45.61    33.50    1.36

MWyr/yr          cumulative absolute
        Petrol   i-petrol Fueloil  Lightoil i-loil

1980     892.00   892.00  3850.93  5807.76  6415.06
1985     940.00   940.00  3819.70  5717.50  6076.10
1990     990.00   990.00  3730.34  5550.44  5762.19
2000    1060.00  1060.00  3535.12  5352.78  5426.61
```

TABLE 3 STIM

MESSAGE II BASE CASE
date: 21. 7.1984 ELECTRICITY GENERATION
time: 15:34 hrs

MWyr/yr	Fueloil	Lightoil	Hydro	Coal cg
1980	134.39	38.65	254.26	20.30
1985	93.89	16.07	324.51	43.48
1990	76.04	19.06	362.13	88.12
2000	0.	28.31	362.13	321.30

GWh/yr	Fueloil	Lightoil	Hydro	Coal cg
1980	1177.23	338.58	2227.33	177.85
1985	822.46	140.75	2842.70	380.90
1990	666.09	166.98	3172.29	771.97
2000	0.	247.99	3172.29	2814.60

MWyr/yr		cumulative	absolute	
	Fueloil	Lightoil	Hydro	Coal cg
1980	134.39	173.04	427.30	447.60
1985	93.89	109.96	434.46	477.95
1990	76.04	95.10	457.23	545.36
2000	0.	28.31	390.44	711.74

```
TABLE  4                        STIM

MESSAGE II                      BASE CASE
date: 21. 7.1984                DISTRICT HEAT GENERATION
time: 15:34  hrs


Note: The heatplant is fueled by heavy oil,
      the cogeneration plant by coal.

MWyr/yr
        Heatplan Cogen
1980     1821.43    59.84
1985     1761.70   119.58
1990     1524.91   210.73
2000      813.12   654.50

GWh/yr
        Heatplan Cogen
1980    15955.76   524.20
1985    15432.48  1047.48
1990    13358.23  1846.01
2000     7122.94  5733.44

MWyr/yr         cumulative absolute
        Heatplan Cogen

1980     1821.43  1881.27
1985     1761.70  1881.27
1990     1524.91  1735.64
2000      813.12  1467.62
```

```
TABLE  5                        STIM

MESSAGE II                      BASE CASE
date: 21. 7.1984                FINAL ENERGY
time: 15:34  hrs


MWyr/yr
          Electr.  Distr.Ht Lightoil  Petrol
1980       402.84   1711.96  2426.09   892.00
1985       430.15   1711.96  2199.02   940.00
1990       490.82   1579.44  1963.78   990.00
2000       640.57   1335.54  1790.38  1060.00

%
          Electr.  Distr.Ht Lightoil  Petrol
1980         7.41     31.51    44.66    16.42
1985         8.15     32.42    41.64    17.80
1990         9.77     31.44    39.09    19.71
2000        13.27     27.67    37.09    21.96

MWyr/yr          cumulative absolute
          Electr.  Distr.Ht Lightoil  Petrol

1980       402.84   2114.80  4540.90  5432.90
1985       430.15   2142.11  4341.14  5281.14
1990       490.82   2070.26  4034.03  5024.03
2000       640.57   1976.11  3766.49  4826.49
```

TABLE 6 STIM

MESSAGE II BASE CASE
date: 21. 7.1984 HEATING STRUCTURE
time: 15:34 hrs (useful energy)

MWyr/yr

	Lightoil	Dir.Elec	Distr.Ht	El.Heatp
1980	1610.93	48.31	1540.76	0.
1985	1515.09	29.80	1575.00	30.10
1990	1419.37	15.55	1484.67	180.41
2000	1305.22	61.02	1268.76	465.00

%

	Lightoil	Dir.Elec	Distr.Ht	El.Heatp
1980	50.34	1.51	48.15	0.
1985	48.10	0.95	50.00	0.96
1990	45.79	0.50	47.89	5.82
2000	42.10	1.97	40.93	15.00

MWyr/yr cumulative absolute

	Lightoil	Dir.Elec	Distr.Ht	El.Heatp
1980	1610.93	1659.24	3200.00	3200.00
1985	1515.09	1544.89	3119.90	3150.00
1990	1419.37	1434.92	2919.59	3100.00
2000	1305.22	1366.24	2635.00	3100.00

TABLE 7 STIM

MESSAGE II BASE CASE
date: 21. 7.1984 SYSTEM COSTS
time: 15:34 hrs

*) O+M costs include domestic resource extraction costs.

M $(80)	Invest	O+M *)	Import
1980	174.80	126.48	1147.87
1985	191.11	137.99	1051.25
1990	149.45	149.97	1043.65
2000	257.99	200.78	882.81

M $(80)	cumulative absolute		
	Invest	O+M *)	Import
1980	174.80	301.29	1449.16
1985	191.11	329.10	1380.35
1990	149.45	299.42	1343.07
2000	257.99	458.77	1341.58

Shadow Prices, US mills / kWh

	1980	1985	1990	2000
Electricity per Load Region				
L.R. 1	103.93	55.86	80.09	48.90
L.R. 2	103.93	55.86	80.09	19.51
L.R. 3	62.66	55.86	61.22	1.16
L.R. 4	0.	0.	0.	0.
District Heat per Load Region				
L.R. 1	49.54	44.76	45.77	46.72
L.R. 2	27.62	23.81	25.47	26.86
L.R. 3	27.62	23.81	25.47	26.86
L.R. 4	27.62	23.81	25.47	26.86
Lightoil	28.20	20.66	22.90	24.48
Fueloil	22.77	20.00	22.18	23.96
Petrol	4.98	29.30	31.81	34.15
Coal	-0.57	15.31	24.74	8.25
Oil	20.55	20.55	22.69	24.40

Emissions of Central Conversion Plants

[Million Tons per Year]

	1980	1985	1990	2000
SO2	41511.43	39763.66	36309.89	35780.23
NOx	14970.27	14155.11	15202.96	16345.50

INSTALLED CAPACITY OF POWERPLANTS, MW

	1980	1985	1990	2000
Fueloil	226.17	211.17	178.62	53.79
Lightoil	139.70	103.43	115.58	179.95
Hydro	347.43	443.42	494.84	494.84
Coal cg	16.72	36.23	73.44	383.03

POWER USE IN 1st LOAD REGION, MW

	1980	1985	1990	2000
Fueloil	203.55	190.05	160.76	0.
Lightoil	139.64	68.45	106.90	179.95
Hydro	215.25	274.72	306.57	306.57
Coal cg	26.27	57.98	117.50	524.83

POWER USE IN 2nd LOAD REGION, MW

	1980	1985	1990	2000
Fueloil	203.55	169.47	160.76	0.
Lightoil	80.85	30.98	30.28	39.62
Hydro	218.88	279.36	311.75	311.75
Coal cg	26.27	57.98	117.50	524.83

POWER USE IN 3rd LOAD REGION, MW

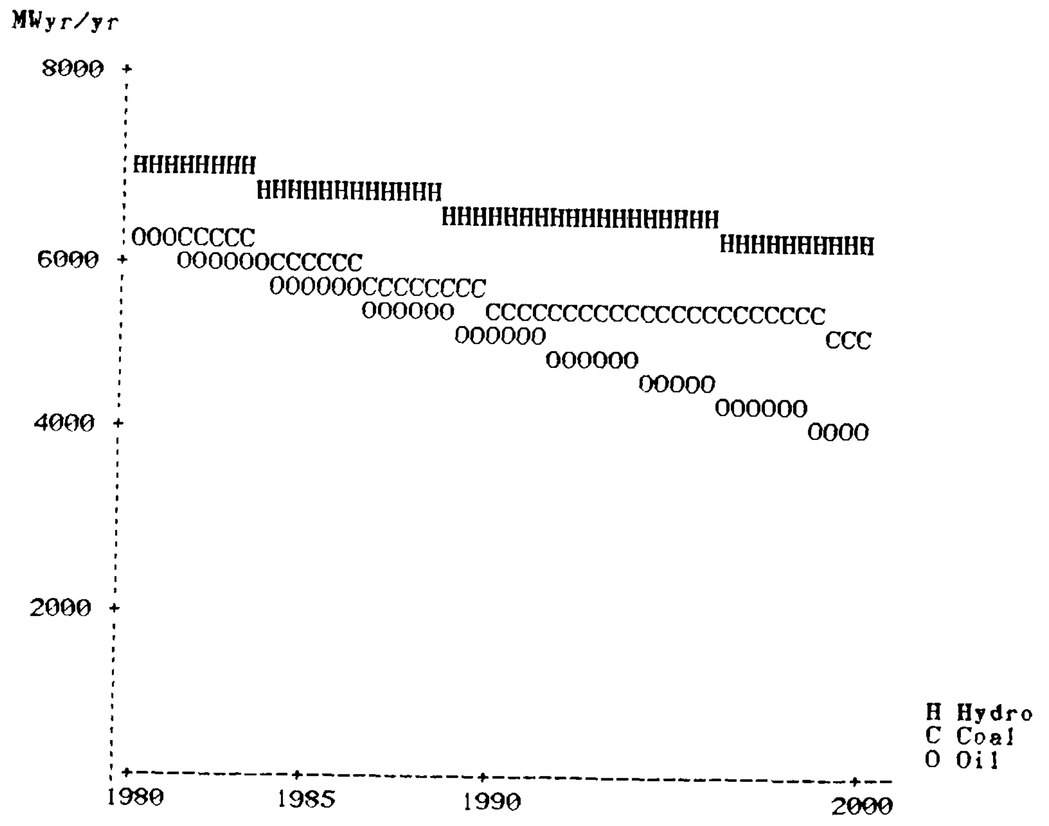
	1980	1985	1990	2000
Fueloil	148.67	64.54	27.03	0.
Lightoil	0.	0.	0.	0.
Hydro	218.88	279.36	311.75	311.75
Coal cg	26.27	57.98	117.50	307.65

POWER USE IN 4th LOAD REGION, MW

	1980	1985	1990	2000
Fueloil	0.	0.	0.	0.
Lightoil	0.	0.	0.	0.
Hydro	361.62	461.52	515.03	515.03
Coal cg	2.39	0.	0.	0.

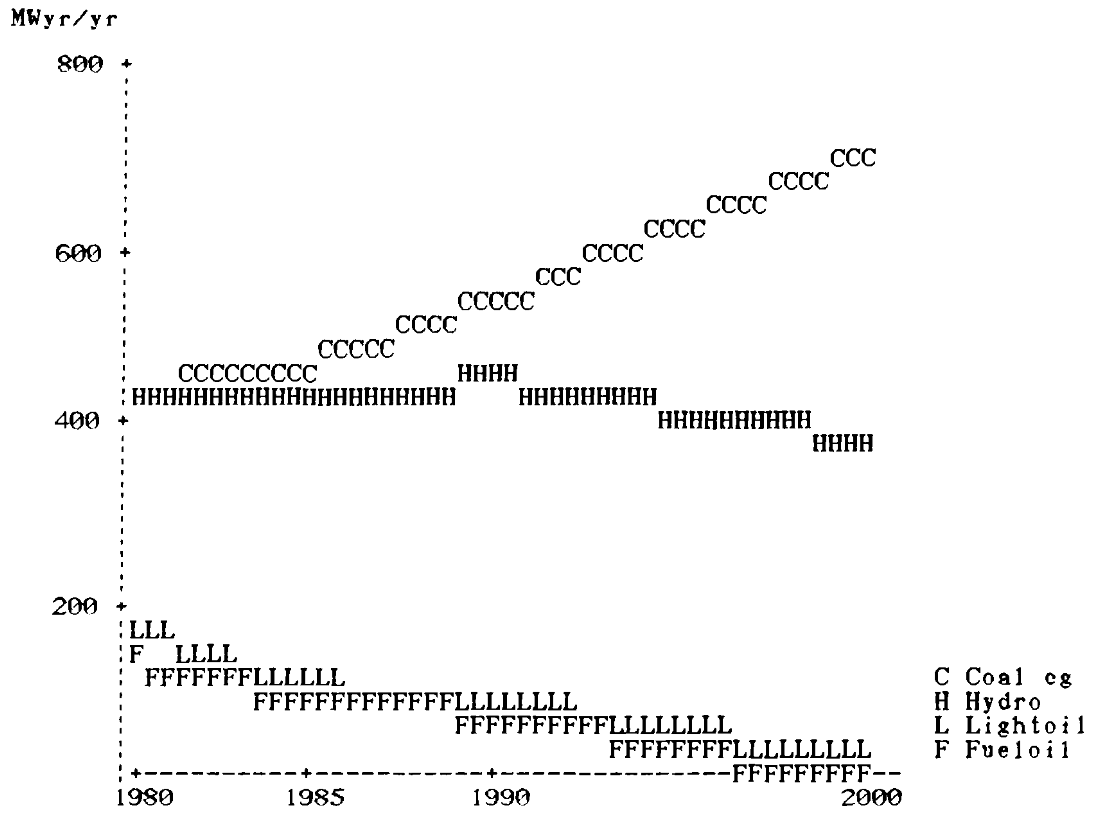
PLOTOUT.DAT

BASE CASE
PRIMARY ENERGY USE
(thermal equivalent)



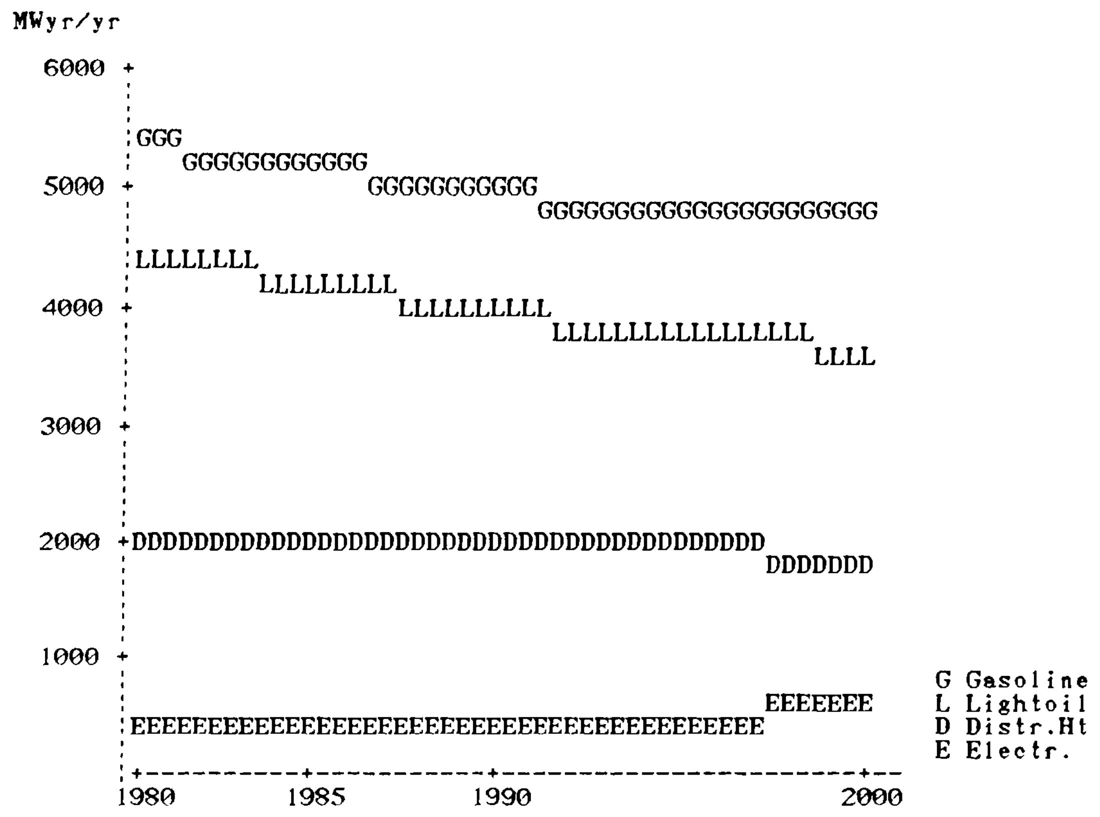
MESSAGE II   Date: 21. 7. 1984

BASE CASE
ELECTRICITY GENERATION



MESSAGE II Date: 21. 7. 1984

BASE CASE
FINAL ENERGY



MESSAGE II Date: 21. 7. 1984