User’s Guide to

McPlot

Version 4.5
August, 1993

by James McEnnan

User's Guide to McPlot
August, 1993

TABLE OF CONTENTS

INTRODUCTION

McPlot is a general-purpose plotting utility which is designed for scientists and engineers who wish to produce high-quality annotated plots from previously generated data stored in files. It is written in a portable dialect of the C Programming Language specifically for UNIX platforms. Plot descriptions are interpreted by a YACC generated parser which provides for a free-format, natural input language. The program incorporates a fan data compression algorithm to reduce plotting time of dense plot data. The user may also invoke built-in cubic spline interpolation routines for data smoothing, as well as a full function scientific calculator (RPN) for general transformations of input data.

In addition to conventional two-dimensional Cartesian plots, McPlot provides the capability to display three-dimensional data in spherical perspective geometry. Input options and the interpretation of data elements are automatically adjusted for each format.

McPlot may be run interactively, with on-line help, from terminals which support graphics output or as a background process executing previously created plot commands. The program currently supports the X Window System, DEC VT102/VT220 and Tektronix 4010/4014 compatible terminals (xterm) and PostScript laser printers.

McPlot accepts data files produced by Matlab or ACSL, as well as user generated ASCII, Fortran and C Language (binary) files in a general matrix format. There are no built-in limits on the size of allowed data files; memory for plot data storage is dynamically allocated, as needed, up to the limits of available machine memory.

McPlot also supplies a basic C Language utility plot library, similar in function to the standard CalComp library, which may be linked with user generated subroutines for specific applications requiring graphics output. The library provides graphics output capability for all of the devices currently supported by McPlot. Manual pages for each function in the library are also included for reference.

Other features include:

- Program defaults may be customized for each user by means of specified *environment* variables.
- The user has the option to log input, including comments for documentation.
- System commands may be executed from within the program, providing the capability to examine or edit data, start jobs, search for files, etc., without exiting McPlot.
- Device independent output, within the limits of screen size and resolution.
- User controlled page layout: portrait or landscape mode, optional date/time stamp, number and size of plots, over-all scale factor, character size and annotation format, etc.

- Multiple axes on one plot and any number of plots per page.
- Linear and logarithmic grids and axes with automatic scaling or user prescribed axis scale and offset; polar plots.
- User selected viewpoint, as well as the size and grid style of the reference sphere used for spherical perspective geometry plots.
- User control over the placement of curve legends, titles, annotation of maximum, minimum and final values, etc.
- Option to clip outliers as well as select segments of curves for plotting.
- Option to place tick marks as a function of input data parameters; user determination of interval spacing and annotation.
- Option to draw error bars around specified points using input data values.
- 9 different types of dashed lines and 18 centered symbols to distinguish curves.
- Five user selectable fonts with scalable characters may be freely mixed for annotation. Additional mathematical symbols, Greek letters, superscripts and subscripts are integrated with the full ASCII character set.
- Arbitrary placement of character strings, centered symbols, data values, arcs, lines, circles, boxes or arrows, etc., to supplement annotation.

How to Use This Guide

The McPlot User’s Guide is divided into seven main sections:

- **Installation**: instructions for installing McPlot on the system.
- **Running the Program**: explains the command line arguments which select the output plot device and other options.
- **Input Plot Data Formats**: describes the allowed formats and options for plot data files.
- **Tutorial**: gives an overview of the input syntax for McPlot and general rules for the formation of the input statements which describe a plot.
- **Examples**: provides a number of examples describing various program options.
- **Customizing the Program**: shows how to set environment variables to specify default values for input parameters.
- **Reference**: provides a complete description of the possible McPlot actions.

New users should read the sections describing how to run the program and the plot data formats, the tutorial and examples. After gaining some experience with the program, the user may wish to customize the program to his own particular requirements. The reference section may be consulted as necessary to provide more detailed information on specific McPlot options.

1. *INSTALLATION*

McPlot is distributed as a ZIP file containing the source code for McPlot *per se*, a general purpose C Language graphics support library which may be linked with user generated routines for specific applications which require graphics output, a user's guide and manual pages for McPlot and its associated plot library and data and input files for sample plots.

The total disk space required is approximately 5 Mbytes.

To install McPlot, create a directory (either in your home directory or /usr/local or wherever you wish to install it), move the ZIP file to that directory and type

```
unzip mcplot.zip
```

1.1 Contents

The McPlot directory should now contain a copy of the GNU General Public License, *gpl.txt* and the following subdirectories.

/bin	This directory contains the McPlot executable, *mcplot* and an executable shell script, *mcprint*, which may be used to send plots to a PostScript laser printer.
/Doc	This directory contains the user's guide as well as the troff source required to create the PostScript file.
/jplot	This directory contains the source code for the McPlot graphics support library, a *Makefile* for creating the library and the compiled library, *plotlib.a*. This library, which is functionally equivalent to the standard Fortran CalComp library, may be linked with any C Language application in which graphics output is desired for any of the devices supported by McPlot. In particular, it is required for the creation of the McPlot executable.
/mcplot	This directory contains the source code for the McPlot program *per se* and a *Makefile* for creating the source library, *mclib.a*, and McPlot executable.
/Examples	This directory contains the data and McPlot input required to produce the example plots shown in the User's Guide. The file, *plot.laser*, contains the McPlot input required to generate the completely annotated example plots, while the file, *plot.screen*, will reproduce the same plots without the additional annotation describing the McPlot input.

/Man	This directory contains a sample manual page for McPlot. The file, *mcplot.l*, comprises the appropriate nroff or troff input to produce the manual page. In addition, the subdirectory, **/plot**, contains sample manual pages for each of the functions in the McPlot graphics support library.

1.2 Making McPlot

If your machine is not a Linux workstation, you will have to create an appropriate executable. Each source directory contains a *Makefile* which may be used to create the appropriate library and/or executable. The graphics support library should be made as the first step. After creating the library, *plotlib.a*, in the subdirectory, */jplot*, the McPlot executable, *mcplot*, may be generated by executing the make utility in the subdirectory, */mcplot*. The new executable will be placed in the mcplot */bin* directory.

As a final step, it may be desirable to create a symbolic link to the McPlot executable in */usr/local/bin* (or */usr/bin* or wherever) so that users do not have to change their environment PATH variable or set up an alias in their *.login* or *.cshrc* files in order to execute McPlot.

To create the symbolic link, change /usr/local/bin, for example, to the current directory. Then, make the symbolic link,

ln -s *McPlot_directory_name*/bin/mcplot mcplot

where *McPlot_directory_name* is the name of the McPlot home directory. With this, McPlot may be invoked by any user whose PATH includes /usr/local/bin by simply typing "mcplot".

1.3 Windows Setup

McPlot does not maintain an internal image of the contents of a plot window; once a plot element is drawn, the information required to create it is discarded. Thus, unless the underlying window system provides a backup, the contents of the plot window may be lost if the plot window is concealed and subsequently restored to the foreground. It may be noted, however, that all of the window systems currently supported by McPlot have the capability to provide backup storage. Whether it is actually implemented or not depends on the particular system and device used to display the plots.

Backing store is enabled by default on current Sun workstations. If, for some reason it is not enabled, see your system administrator.

If Exceed is used to access a server from a PC, backing store may be enabled from the *performance* options of the configuration menu. Select "enable backing store" and "when mapped" from the list of options.

On a Linux workstation under the Gnome Desktop, root access is required to enable backing store. After logging in as *root*:

start *gdmsetup*
under the *Security* tab there is a button (*Configure X Server*)
in the window that comes up, edit *Command* to read "/usr/bin/Xorg -audit 0 +bs -wm"

Alternatively, edit /etc/gdm/custom.conf and add *+bs* and *-wm* to the command line options.

(Note: *+bs* is an undocumented option for XFree86.
Current documentation implies that backing store enabled is the default and the option, *-bs*, turns it off.
Hence, it may be that for older implementations of the X Window System, simply removing the *-bs* and adding the *-wm* option will enable backing store as documented.)

Finally, it should be noted that McPlot output is intended to be device independent so that plots produced in X windows appear the same as plots sent to PostScript printers, for example, within the limits of device resolution. Hence, the default size of an X plot window is 8½"×11", corresponding to standard U.S. letter size paper. For monitor screens smaller than 19" (diagonal), the entire default window cannot be displayed and, as a result, some part of the plots may be lost. In this case, it may be desirable to open a smaller window.

In order to accommodate the wide variety of X terminals, McPlot allows the user to specify the default position and size of the window by assigning appropriate values to the *environment* variables, JPXWXP, JPXWYP, JPXWXD, JPXWYD, which determine, in order, the initial (*x*, *y*) position of the upper-left corner and the *X*- and *Y*-dimensions (in *pixels*) of the window. See the section, "Customizing the Program", for a complete list of environment variables recognized by Mcplot.

To utilize this feature, the user may edit his *.profile*, *.login* or *.cshrc* file and add the appropriate *setenv* or *set* commands. In the C Shell, for example, these have the form,

```
setenv JPXWYD 690
```

while, in the Bourne or Bash shell the same result is obtained by

```
JPXWYD=690
export JPXWYD
```

For a 17" monitor screen, 690 is the appropriate value for the window vertical (Y) dimension; smaller screens will require smaller values.

It is also possible to resize the plot window using the mouse. However, the procedure is rather cumbersome and is not recommended. It is included here only for completeness.

Because each plot element is drawn only once, interactive resizing of the plot window is not effective until a new page is started. If it is desired to resize the window before any plots are drawn, the following input sequence may be used:

```
> put;            % Opens the plot window and displays it on the screen.
>
>                 % At this point, the plot window may be resized.
>
>                 % Next, read data, choose layout and select curves as usual.
>
> plot page ... ;        % The plot flag, page, forces a new page.
>
```

If a smaller window is necessary, it may also be desirable to change the overall scale factor used by McPlot. This is done using the *layout* parameter, *factor* (see Section 7.5). A factor of 0.67 will allow a full page of data to be plotted in a window sized as indicated above.

Finally, for convenience, it may be desirable to select window preferences which make a window active when a mouse pointer moves over them, but do not raise the window. This allows review of a series of plots without having to toggle between screens.

This completes the installation of McPlot.

1.4 Graphics Support Library

The C Language graphics support library included with the McPlot program provides most of the functionality of the basic CalComp plot library for all of the graphics terminals and printers currently supported by McPlot. These routines may be used by any C Language application which requires graphics output.

Assuming that the user's source routines have been compiled and the resultant relocatable object modules have been placed in a library, *yourlib.a*, a typical link command would have the form:

```
cc -o prog yourlib.a /usr/local/mcplot/plotlib.a -lX11 -lm
```

on a Linux workstation, while on a Sun workstation the command would be:

```
cc -o prog yourlib.a /usr/local/mcplot/plotlib.a -L$OPENWINHOME/lib -lX11 -lm
```

See the manual pages and McPlot source for more information about using the Graphics Support Library.

2. RUNNING THE PROGRAM

User input to McPlot is read from the standard input, *stdin*. Input may be entered interactively from a terminal or the user may redirect *stdin* so that the program reads from a previously prepared file. In either case, the input is identical; there is no difference between batch and interactive mode.

Error messages, prompts and *help* output are written to *stderr*. If an error is encountered during batch operation, the user is notified, but the program will continue execution until the end of file is reached.

2.1 Invocation

The program is invoked with the following command line.

mcplot [-q] [-l [*logfile*]] [-i [*datafile*]] [-m [*mode*]] -w | -x | -p [*printfile*]]

The various options are explained in the following table.

-q	Quiet option. This option turns off all text output except for error messages. In particular, the user is not prompted to continue with the next plot. This option should be selected for batch jobs or to quickly review a series of plots.
-l	Log user input. The next argument, if present, is assumed to be the name of the log file. If no name is present, the log file is given a name of the form McPlot.XXXXXX, where XXXXXX is a unique string which depends on the user's job number (see also mktemp(3C)).
-i	This option may be used to specify the default data *file* parameter used in the *read* command. The next argument, if present, is assumed to be a data file name. If no file is specified, the *environment* variable, JPFILE, is used. If the file name is unspecified, the default, is "data" in the current directory.

The following options determine the output mode and device type.

-m Output mode. The next argument, if present, is assumed to be an integer specifying the output mode. The effect, if any, depends on the selected device. For PostScript printers, *mode* = 0 specifies encapsulated PostScript output which may be directly imported into FrameMaker or other document preparation utilities. Any other value of *mode*, or if the output mode is not specified, produces regular PostScript output. For other output devices, *mode* currently has no effect.

-w Output is directed to a window which is opened under Sun OpenWindows or the X Window System. The default position and size of the window may be specified by assigning appropriate values to the *environment* variables, JPXWXP, JPXWYP, JPXWXD, JPXWYD, which determine, in order, the initial (*x*, *y*) position of the upper-left corner and the *X*- and *Y*-dimensions (in *pixels*) of the window. The window may be moved, resized or iconized using the mouse.

-x Output is directed to *stdout* and comprises the appropriate graphics commands (escape sequences) for X terminal emulation (Xterm) under the X Window System.

-p Output is appropriate for a PostScript laser printer. The next argument, if present, is assumed to be a file name for the print output. If the file name is –, output is directed to *stdout* and may be piped directly to lpr. If no file name is specified, the default file name used is "McPlot.PS" (or "McPlot.EPS if *mode* = 0). The print file may be printed using the command, lpr.

If no option is specified by the user on the command line, the environment variable, TERM, is read to determine the user's device type. If the device type is unspecified, the **-w** option is assumed. Only one output device may be specified at a time.

In most cases it is sufficient to simply type the program name to run McPlot interactively. However, if it is desired to create a McPlot input file interactively, for later batch submission, the invocation,

```
mcplot -l logfile
```

will save subsequent user input in the file, *logfile*, in the current directory.

User input to McPlot may also be redirected to read from a previously prepared file and the resultant plots reviewed interactively; e.g.,

```
mcplot < user_input_file
```

where *user_input_file* contains the McPlot input statements required to produce the desired plots.

Normally, the following prompt is written to *stderr* after each *plot* or *put* action.

Press return to continue, q to quit.

This prompt and the user's response, which is read from */dev/tty*, are entirely separate from the input parser and allow the user to pause after each plot is generated. The command line option, **-q**, suppresses this prompt, as well as other program text output, and should be used for all batch operation.

To obtain plots on a PostScript laser printer using a previously prepared file of McPlot input statements, the command line typically has the form,

```
mcplot -q -p- < user_input_file | lpr
```

In this invocation, the PostScript output is piped directly to lpr.

If it is desired to exercise McPlot from a shell script; e.g., to avoid long command lines, the following should be noted. In some cases, the Shell will block a command which attempts to read */dev/tty*. Thus, the *exec* command must be used to start McPlot from a shell script in this case whenever the **-q** option is not invoked. A sample script to send output to a PostScript laser printer is shown below.

Sample McPlot Shell Script

```
#! /bin/sh
# mpr: Shell script to execute McPlot and pipe PostScript output to lpr.
#
if test $# -eq 0
then
  echo "No input file given."
  exit
fi
if test -f $1
then
  exec mcplot -q -p- < $1 | lpr
else
  echo "file $1 does not exist."
fi
```

This script may be invoked as follows:

mpr *user_input_file*

where *user_input_file* contains the McPlot input statements required to produce the desired plots.

A more elaborate shell script, *mcprint*, which utilizes additional McPlot command line options and input capabilities is provided with this distribution.

The *mcprint* shell script allows the user to determine the printer to which the postscript output is sent.

In addition, the name of the file containing the data to be plotted may also be specified on the *mcprint* command line. This is useful if a number of similar plots are to be generated from different data sets, for example.

The *mcprint* script also allows plots to be printed full sized, even if the screen is too small to display a complete page. To use this feature, the plot description must include the following comment after the normal plot *layout* description:

```
%layout factor=1;
```

On option, the *mcprint* script will remove the initial *%* (percent sign), signifying a comment, so that the resultant plots are printed full size. See the Tutorial and Reference Sections for more information about comments and the *read* and *layout* actions.

Finally, typing

```
mcprint -h
```

displays the following:

```
usage: mcprint [ -h ] [ -m ] [ -i data_file ] [ -{[P]printer} ] input_file

  printers:
          -lp
          -P{other} (note: no space between P and printer name)

  other options:
              -m -> modify input_file:
                  if -i option, remove lines containing "file";
                  otherwise, remove lines starting with "%read"
                  remove '%' character in column 1 (only)

              -i -> use input data file name: data_file
                  see McPlot User's Guide for description
                  note: there must be a space between
                  -i and the name, data_file
```

As with any X application, operation of McPlot under the X Window System also provides additional pop-up menus which may be invoked with the mouse. In this case, the user may move or resize the plot window, bring it to the foreground or put it in the background or shrink it to an icon. The various options are selected in the same manner as in other X applications.

3. *PLOT DATA FORMAT*

The model for plot data in McPlot is a matrix in which each row comprises one data record and each column is referred to as a plot vector. There are no limits on the number of plot vectors or records other than those imposed by the available machine memory.

3.1 Rectangular Cartesian Coordinates

In rectangular Cartesian coordinates, any pair of plot vectors may be used to describe a plane curve. The coordinates of successive points on the curve are given by the corresponding components of the plot vectors; one for the abscissa (x-coordinate), the other for the ordinate (y-coordinate).

In a typical application, data output by a simulation comprises the state of the system at successive times. These data are written to a file as a series of records; each record containing the simulation time variable and selected elements of the state vector. After the file is written, the elements of the state vector may be plotted as a function of time using McPlot. In this case, the sequence of time values would comprise one plot vector, the abscissa values, and successive values of one of the elements of the state vector would comprise the ordinate of the plotted curve.

3.2 Spherical Perspective Geometry

In spherical perspective geometry, input plot data is interpreted in terms of the azimuth (*az*) and elevation (*el*) in decimal degrees, and radial distance (*d*) in units of the reference sphere radius, of a point in space. Any sequence of such points may be used to define a curve which is drawn in perspective relative to the reference sphere. The nominal range of azimuth angles is $0 \le az < 360$, while the range of elevation angles is $-90 \le el \le 90$. The distance, *d*, must be greater than 1 if the point is to be visible on the plot.

Examples of acceptable azimuth and elevation variables are right ascension and declination (on the celestial sphere) or longitude and latitude (geodetic data).

An arbitrary vector may be defined in terms of *az* and *el* by

$$\vec{r} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = r \begin{pmatrix} \cos(az)\cos(el) \\ \sin(az)\cos(el) \\ \sin(el) \end{pmatrix}$$

where $r = \sqrt{x^2 + y^2 + z^2}$ is the magnitude of $\vec{r}$. Hence, in general,

$$az = RTD \times Tan_2^{-1}(y, x)$$

$$el = RTD \times Tan^{-1}(z/\sqrt{x^2 + y^2})$$

$$d = r/RADIUS$$

where $RTD = 180/\pi$ is the conversion from radians to degrees and $RADIUS$ is the radius of the reference sphere. For the particular case in which $\vec{r}$ is a unit vector ($d = 1$), the expression for the elevation angle simplifies to

$$el = RTD \times Sin^{-1}(z)$$

3.3 Data File Formats

The particular format of a plot data file varies, in general, from one application to another. The specific formats currently accepted by McPlot are:

- ASCII data. In this case, the data is produced in a form which is readable using a normal text editor and is directly printable on standard printers or terminals. Distinct data values must be separated by white space (blank, tab or newline) or a comma (,) or a slash (/); other separators are not allowed. If a comma or slash is used, repeated separators may be employed to signify missing data values which are assigned the value zero. In keeping with the spirit of the UNIX operating system, successive plot records are not marked in any way; e.g., by a newline (carriage return). Hence, the user must specify to McPlot the number of variables in a record (= the number of plot vectors) and McPlot will read across physical lines in a file to fill up a record.

- Fortran binary (unformatted write). In this format, each plot data record is produced by executing a Fortran statement of the form

  ```
  write(iunit) (plotrec(k), k=1,nvar)
  ```

  In this case, the user must specify to McPlot both the number of variables in a record (nvar) and the precision or length of a variable (single or double precision). All of the variables in a record must be of the same length (all single or all double precision). It is assumed that all input variables are in floating point form; integer variables are not allowed.

  Unfortunately, there is no uniform standard in Fortran for end-of-record delimiters or record format; each Fortran compiler is free to define the end-of-record indicator and record format arbitrarily. For example, Fortran files written on a Sun workstation by a program compiled using the Sun Fortran compiler are not readable by a Fortran program compiled using GNU Fortran on the same machine. In order to accommodate this situation, McPlot has the capability to read files generated by several common Fortran varieties: Sun Fortran, GNU Fortran and Microsoft/Digital Fortran. The specific Fortran version may be specified by the user when the data is read in. However, McPlot does not perform byte re-ordering, so Fortran files written on different machines may still be unreadable.

- C Programming Language binary write. In this format, a plot data record is produced by executing a C Language statement of the form

  fwrite(buf,sizeof(*variable*),nvar,fp);

  In this case, the user must specify to McPlot both the number of variables in a record (nvar) and the precision or length of a variable (single or double precision). All of the variables in a record must be of the same length (all single or all double precision). It is assumed that all input variables are in floating point form; integer variables are not allowed.

- Matlab (*.mat) file. In this case, the number of rows equals the number of records and each column comprises one plot vector (variable). It is not necessary to specify the number of variables in this case, since the Matlab header record contains the necessary information about the size of the matrix; nor is it necessary to specify the precision of the input data, since Matlab variables are always double precision. The input matrix may be either real or complex; McPlot will indicate which variables correspond to the real part and which variables correspond to the imaginary part of the matrix. By default, McPlot reads the first item in a Matlab file. However, the user may optionally specify the name of the Matlab variable (matrix) to be read in and McPlot will search the file to find the appropriate name.

  It should be noted, however, that versions of Matlab later than Matlab 4 use a proprietary encoding scheme which has not been published. As a result, McPlot is unable to read arbitrary Matlab files. Nevertheless, Matlab 4 files may be read and written by all current versions of Matlab.

- ACSL (*.rrr) file. As with Matlab data, it is not necessary to specify the number of variables in this case, since the ACSL file includes a header record which contains the necessary information about the record length; nor is it necessary to specify the precision of the input data, since ACSL output files are always single precision. McPlot will report the number of variables as well as the number of records actually read in.

3.4 Memory Management

In order to provide for manipulation of the data to be plotted without affecting the user's original data and to allow for rapid access to the data for scaling and plotting, it is necessary to copy the data into memory. In order to insure that files of arbitrary size can be accommodated, memory for data storage is dynamically allocated as required. The only limits are those of available machine memory.

Memory for data storage is allocated in blocks; one for each variable. Each data block is a structure which contains a pointer to the next data block, statistical information about the data in the block, such as the first and last, maximum and minimum values, and the number of data items, etc., as well an array containing the actual data items. The statistics are compiled as the data is read in and simplify the process of scaling and retrieval, since only the statistics of preceding blocks need be accessed in order to locate the block containing a particular value. If enough data is read in to fill the first block, more memory is allocated and the pointer is set to the start of the newly allocated memory. The result is a linked list of data blocks in which at most one block need be examined in detail to find a given record. The size of each data block is large enough that only one block is required for small plots (≤ 506 points). Because all values are stored internally as double precision real numbers, 4K bytes of memory are required per data block for each plot variable.

In addition to the memory required to store data, some memory is allocated for intermediate storage of plot vectors during plot execution. In general, two working vectors are allocated after the data is read in and these are used to hold properly scaled data values for plotting. Additional memory may required by the *calc* action to implement the stack. This memory is allocated only if the *calc* action is entered and is deallocated when the action is terminated. Similarly, if data smoothing is selected, additional working vectors may be allocated as needed. All of this is usually transparent to the user. However, if, at any time, there is insufficient machine memory to provide the required storage, an error message is printed on *stderr* and, depending on where the allocation fails, the input data in memory may be lost. It should be noted, however, that it is unlikely that insufficient memory will be found on current systems.

4. TUTORIAL

The purpose of this section is to provide an overview of the user input to McPlot. For most users, this section together with the examples should be enough to get started. A complete description of the possible McPlot actions is given in the reference section.

McPlot is designed to facilitate interactive specification of the features of an output plot. A built-in help command and context sensitive prompting provide additional assistance in this regard. At the same time, the program may be used in batch mode by simply redirecting a file containing previously created McPlot commands to the standard input. The user input log of an interactive session, which is produced, on option, by the program, is one source of such command files. In any case the input syntax is identical in either batch or interactive mode.

User input to McPlot is controlled by a parser which accepts free format input and performs specified actions when an instance of a grammatical rule is recognized. Generally, the action is to select a member of the current list of possible input variables and assign a value to it. It is not necessary to understand how the parser works in order to use McPlot. However, it is necessary to have some idea of the grammatical structure in order to take full advantage of the input features.

The grammar of a language is simply a set of rules or definitions which describe the syntactically correct structures of the language. Accordingly, it is appropriate to start a description of the input syntax of McPlot with some definitions.

These include:

- Token: an element of the input which is recognized by the parser as a distinct entity. One of the functions of the *help* command is to provide a list of currently acceptable tokens. In general, tokens are delimited by white space (blank, tab, newline). White space is otherwise ignored; in particular, newlines (carriage returns) are not seen by the parser and have no significance except to make the input more legible.
- Action keyword: a token which initiates an action.

4.1 Actions

In McPlot, the current action keywords are:

Keyword	Description
input	This action allows plot description elements (user input) to be inserted into the input stream from a specified file. When an *end-of-file* is reached, input is resumed from the previous input stream. *Input* actions may be nested. The current maximum is 5 levels.
read	This action is used to read into memory the matrix of data to be plotted.
calc	This action is used to transform data before plotting. It employs Reverse Polish Notation (RPN) which is similar to that of the Hewlett-Packard line of scientific calculators.
map	The *map* action determines whether data is interpreted in terms of rectangular Cartesian or spherical coordinates and adjusts plot options and the output format accordingly.
layout	The *layout* action is used to set the default composition of a page. It allows specification of the default size, number and position of plots on a page and well as the default character size, grid style, etc.
select	*Select* actions specify the curves (data elements) which will be drawn on a particular plot.
plot	The *plot* action controls the composition of the current plot on the page and causes it to be drawn on the selected device.
put	The *put* action allows the placement of additional annotation on the current plot.

The above list also indicates the normal sequence of actions which is used to generate a plot, although it is not necessary to exercise all actions for every plot. There are no built-in constraints, however, so that actions in McPlot may be entered in any order, and may be repeated as many times as desired.

Each action has a certain number of variables (or operators in the case of the *calc* action) associated with it. The user may specify as many of these variables as necessary and in any order. Any variable not explicitly prescribed assumes its default value. In this way, simple plots may be obtained with a minimum of input but the user has the freedom to specify as many aspects of the resultant plot as desired.

The general form of an action is:

> *action* { *specification_list* } ;

where *action* is one of the action keywords listed above, the optional *specification_list* prescribes the variables (or operators) associated with the action, and the semi-colon terminates the action. Note that the semi-colon is always required, even if the specification list is empty.

There are, in general, two types of variables which may appear in a specification list. These are:

- Parameter: a variable which may appear on the left-hand side of an assignment statement.

A parameter specification must have the form, *parameter* = *value*, where *value* may be nothing, an integer, a real number, a string, or a comma separated list of values. Thus, valid parameter assignment statements might include:

factor =

The result of this is that the value of *factor* is unchanged.

type = 5

The integer parameter *type* is set to 5.

start = 4.35

The real parameter *start* is set to 4.35.

legend = "1st curve"

The string parameter *legend* is set to "1st curve".

y = , , 27, , 2

The first, second and fourth elements of the integer array, *y*, are unchanged, the third element is set to 27 and the fifth element is set to 2.

- Flag: a token which assigns a predetermined value to an internal variable when it appears in the input stream. For example, the *read* flag, *ascii*, causes an internal flag variable to be set to 0, while the flag, *gnufor*, causes the same internal variable to be set to 2. Generally, flags occur in pairs; e.g., *grid*, *nogrid*, etc. In the help screen, the default flag value is generally the first one shown.

Note that it is an error to attempt to assign a value to a flag variable, while a parameter variable must be followed by an equal sign.

In the *calc* action, tokens appearing in the specification list must be one of the following:

- Operator: token which invokes a function to perform some transformation on the input data.
- Constant: a literal number entered by the user or a predefined constant such as *pi* (= 3.1415...), *dtr* (conversion from degrees to radians) or *rtd* (conversion from radians to degrees).

Certain parameters and flags associated with an action may have *attributes*.

- Attributes: variables associated with a particular parameter or flag and which further define it. Attributes may, themselves, be either parameters or flags.

The syntax for attributes is as follows:

parameter (*attribute_specification_list*) = *value*

flag (*attribute_specification_list*)

Within an attribute specification list, the parameter and flag attributes may be prescribed in any order. Any attribute not explicitly referenced assumes its default value. Note that the parentheses are necessary to mark the beginning and end of the attribute specification list. If no attributes are specified, the parentheses may be omitted.

For example, the *layout* flag, *time*, specifies that the date and time are drawn, by default, on all plots. Its attribute, *font*, determines the font used for the date/time annotation. This parameter may be set to zero by the user input,

time(font=0)

Similarly, the *plot* parameter, *title*, determines the title of the current plot. In the simplest case, a plot title may be specified by input of the form,

title = "This is the title"

If it is desired to place the plot title at some position other than the default, the attributes of *title* may be employed to specify the location. For example, the input,

title(x=1.0 y=5.0) = "This is the title"

places the beginning of the title one inch to the right and five inches above the current plot origin. In either case, the actual plot title is given by the string enclosed in double quotes.

Attributes may be nested to several levels so that attributes may, themselves, have attributes. This allows infrequently used variables to be effectively hidden during normal use, but remain available if required.

4.2 Commands

The final class of tokens which may appear in the input to McPlot are *commands*.

- Command: one of a class of tokens which may be invoked to exit the program, obtain help, save a log of the input, examine or edit files, start jobs, search for files, etc.

Commands may be invoked at any point in the input, in any order and as often as desired. The currently available commands are:

Command	**Description**
quit	Typing *quit* initiates the normal exit from McPlot; it is synonymous with End-of-File (Ctrl-D).
help	The command, *help*, results in a display of the currently available commands and actions or parameters and flags or operators and constants, depending on the parser state. The *help* command uses the same look-up tables as the parser. This means that an action must be invoked before its help screen can be seen. Thus, for example, the sequence to obtain help for a particular action is > *action help* { *specification_list* }; Similarly, a complete description of the attributes of a parameter or flag may be obtained by invoking the *help* command *after* entering the (left) parenthesis marking the start of the attribute specification list for that parameter or flag. In general, the *help* command may be employed at any point within an action to indicate the currently available parameters and flags (or operators and constants), or at the top level to list the commands and actions.
save	The *save* command causes a log of the user's input to McPlot to be saved in a file if the -l option has not been invoked; otherwise, it has no effect. If no log file name has been specified on the command line, the log file name is of the form McPlot.XXXXXX, where XXXXXX is a unique string which depends on the user's job number.

!system command	When an exclamation point, !, is encountered in the input stream, all input following it up to the first newline is sent to the operating system as a system command. Thus, the user may examine or edit files, start jobs, search for files, etc., without exiting McPlot. Note that system commands are not echoed in the log file.

McPlot actions are only performed when the terminal semi-colon is seen by the parser. Commands, on the other hand, are executed immediately, regardless of the parser state. Within a specification list, input is interpreted by the parser in the order it is entered so that the last value assigned to a parameter in a specification list is the one which is ultimately used. Similarly, a flag may be canceled by subsequently invoking its opposite within the same specification list. In the *calc* action, operations are performed immediately in the order specified.

4.3 Defaults

Any variable which is not explicitly specified by the user is given a *default* value by the program. Proper selection of defaults allows the user to describe the desired characteristics of a plot with minimum input.

In McPlot, there is a hierarchy of defaults.

At the top level, the user may specify nearly all of the program defaults by assigning appropriate values to certain *environment* variables. See the section, "Customizing the Program", for more information.

At the next level, certain defaults may be given on the command line which invokes the program. These include the default input plot data file name, the device to which the plots are directed, the log file name, etc.

The next level comprises the built-in program defaults. Defaults not specified on the command line or in the environment retain their programmed values.

At the next level are *layout* parameters and flags which control the nominal layout of plots on a page, the size and spacing of plots, alpha-numeric character sizes, special centered symbol size and plotting interval, precision of numerical annotations, etc. These may be freely specified by the user and any *layout* parameter or flag which is not explicitly prescribed assumes its system default value according to the hierarchy noted above. The resultant values for these parameters and flags function, in turn, as defaults for *plot*, *put* and *select* parameters and flags (and their attributes) which describe the same characteristics.

At the next level are values assigned to *plot* parameters and flags which affect alpha-numeric character sizes, special centered symbol size and plotting interval, precision of numerical annotations, etc., within a given plot. These may be freely specified by the user and any *plot* parameter or flag which is not explicitly prescribed assumes its default value according to the hierarchy noted above. The resultant values for these parameters and flags function, in turn, as defaults for *select* parameters and flags (and their attributes) which describe the same characteristics for each curve on a plot.

In short, wherever appropriate, defaults flow down from the user to the program, to all plots in a given run and, finally, to components of a particular plot. If an exception to the default is desired, the user merely has to specify it at the appropriate level.

4.4 Morphology

The following rules are valid for all of the input to McPlot.

- Input may be either in upper or lower case. Except in quoted strings, all input is converted to lower case by the lexical analyzer.
- In general, a string is delimited by double quotes; however, anything which is not recognized by the lexical analyzer is returned to the parser as a string token. Hence, a string which contains no white space and does not begin with a sequence which might be recognized by the parser, as is the case with most UNIX file names, need not be delimited by double quotes, although it is always correct (and safe) to include them. Note, however, that, since all unquoted input to the lexical analyzer is converted to lower case, strings which are intended to include upper case characters must be quoted. If a double quote character is desired in a string, it must be escaped with a backslash, \".
- Additional mathematical symbols and Greek letters may be included anywhere within a quoted string. For this purpose, each symbol has a three-character name of the form, \(**x**, where **x** identifies the letter or symbol. In particular for Greek letters, **x** is an upper or lower case Roman letter which is reminiscent of the Greek; e.g., \(a is α, etc. As an example,

  $$\Sigma\ (\alpha\times\beta) = \tau$$

  is produced by the sequence,

  \(S (\(a\(*\(b) = \(t

  Any such three-character name which is not recognized by the lexical analyzer produces a blank space. A complete list of the identifying character names available in McPlot, together with the associated symbol, is shown below.

a	b	g	d	e	z	y	h	i	k	l	m	n	c	o	p	r	s	t	u	f	x	q	w
α	β	γ	δ	ε	ζ	η	θ	ι	κ	λ	μ	ν	ξ	o	π	ρ	σ	τ	υ	ϕ	χ	ψ	ω

A	B	G	D	E	Z	Y	H	I	K	L	M	N	C	O	P	R	S	T	U	F	X	Q	W
Α	Β	Γ	Δ	Ε	Ζ	Η	Θ	Ι	Κ	Λ	Μ	Ν	Ξ	Ο	Π	Ρ	Σ	Τ	ϒ	Φ	Χ	Ψ	Ω

!	#	$	*	?	,	/	:	;	<	=	>	.	0	1	2	3	4	5	6	7	8	9
$\neq$	∞	$\surd$	$\times$	$\pm$	∂	$\int$	$\equiv$	∇	$\leq$	$\approx$	$\geq$	$^\circ$	0	1	2	3	4	5	6	7	8	9

In addition, there are two non-printing characters which provide relative vertical motion within a string: \(+ raises the remainder of the string one-half character, while \(− lowers the remainder of the string one-half character. By appropriately pairing these raising and lowering characters, subscripts and superscripts may be accommodated within a quoted string.

Note that, if a three-character name sequence is desired in an output string, it must be escaped with a backslash, \\(**x**.

- Any token which begins with a digit, [0-9], a sign, + or −, or period, ., is assumed to be a number. If a decimal point is present, it is assumed to be a real number (double), otherwise it is an integer. A real number may have an optional exponent; e.g., 1.23e-45. The number ends at the first blank, tab or newline; hence, signed numbers must not have white space between the sign and the remainder of the number. The parser knows what the type of each variable is (double or integer) and will perform any necessary conversion. Thus, it is not necessary to include a decimal point when specifying whole real numbers and decimal points appended to integers are ignored.
- Certain tokens which may be described as punctuation, specifically, " ! = ; , ( ) [ ] (double quote, exclamation point, equal sign, semi-colon, comma, left and right parenthesis and left and right square bracket), do not require surrounding white space. Note, however, that all other tokens must be delimited by white space, including, for example, the / (division) or − (subtraction) tokens for the calculator.

4.5 Comments

Comments may be used to make the input more readable and allow selective use of parts of existing input files, without requiring creation of a separate file. In McPlot, anything between a % (percent sign) and the following newline is ignored by the parser. For example,

```
> %This is a comment
> select y=2;                    % this is another comment
>
```

4.6 Prompts

Whenever a newline is encountered in the input stream, a prompt is output by the lexical analyzer. This prompt reflects the state of the parser. At the top level, the prompt is just a right angle-bracket, >. When an action keyword is input, the prompt output in response to a newline will show the current action keyword followed by the angle-bracket; for example,

```
> read                       % this is the user’s input
read>                        % this is the output prompt
```

After starting an attribute list by entering the name of a parameter or flag and a left parenthesis, the prompt indicates the parser state by appending a dot and the name of the parameter or flag to the basic prompt. For example,

```
> select        x = 12 y = 26
select>         clip( y(max=13.47
select.clip.y>          min=-2.35))
select> ;
>
```

Each level of attributes (parentheses) acquires another suffix, which is removed when the parentheses are closed. Similarly, when an action is terminated by a semi-colon, the prompt reverts to its initial state form.

4.7 Error Recovery

Facilitating recovery from input errors is, in general, a difficult problem for the parser designer. In many instances, the appropriate action is to ignore all input after an error until the end of the line and then prompt the user to re-enter the line. In McPlot, this is not possible since the input is not line oriented and the parser does not, in fact, know when a newline is received.

The solution adopted is the following. If the parser is in its initial state so that an action keyword is expected, the parser will print a message on *stderr* indicating the problem and ignore all input after the error until a correct command or action keyword is entered.

Once an action has been started, the parser will print a message on *stderr* giving the approximate location of the error and ignore all subsequent input until a correct statement is encountered in the input stream. This may be either a command, a flag or a parameter assignment, or, in the case of the *calc* action, an operator or constant. For the most part, this does not cause any problems. The user simply re-enters the appropriate input starting from the indicated error. However, if the error is caused by the omission or misplacement of a final parenthesis or semi-colon, it is not sufficient to simply input the required parenthesis or semi-colon, since these do not qualify as statements. In this case, it is useful to remember that the *help* command is always a valid statement and does not cause any side effects. Thus, typing *help* followed by a carriage return is the simplest way to reset an error condition in these circumstances and the output help screen and prompt will precisely indicate the current state of the parser. At this point, the necessary terminating parenthesis or semi-colon will be accepted by the parser and the user may continue normal input to the program.

5. EXAMPLES

In the examples which follow, data from ten separate files of varying lengths and written in four different formats (ASCII, single and double precision C binary and single precision Fortran binary) are plotted in the same run. These data, as well as the input required to generate the example plots, are contained in the McPlot subdirectory, **/Examples**.

The examples are intended to illustrate the input syntax and some of the more commonly used features of McPlot. It would be difficult to demonstrate all of the features of the program, since, at last count, there are 256 different parameters and flags which may be prescribed by the user. However, as will be shown, quite elaborate plots may be obtained from relatively simple input.

The actual input statements which were used to generate each example are reproduced below the plots. New users may find it useful to run McPlot interactively and experiment with the examples. In this context, the *help* command may provide suggestions for other possible options. It should also be emphasized that the input format in McPlot is quite flexible. The user is encouraged to develop his own input style and apply it consistently.

The first step in generating a plot is to create a file containing the data to be plotted. Data for the first three examples was generated using the following Fortran program which writes to a file, "fort.14".

FORTRAN DEMO PROGRAM

```
      dimension p(3)
      data dtr/0.01745329252/

      do 100 i=1,361
        x = float(i - 1)
        p(1) = x
        x = x*dtr
        p(2) = sin(x)
        p(3) = cos(x)
        write(14) p
  100 continue

      stop
      end
```

The output is a series of 361 plot records in which x (in degrees) is the first variable, $sin(x)$ is the second variable and $cos(x)$ is the third variable in each record.

Cartesian Map

The first 13 examples apply to the Cartesian (*xy*) map (the default) in which input data points are interpreted as rectangular Cartesian coordinates (x,y) of a plane curve and are plotted conventionally using x for the abscissa and y for the ordinate.

5.1 Example 1

The first example shows a very simple plot with just two curves and no annotation such as might be used for a quick interactive examination of the data.

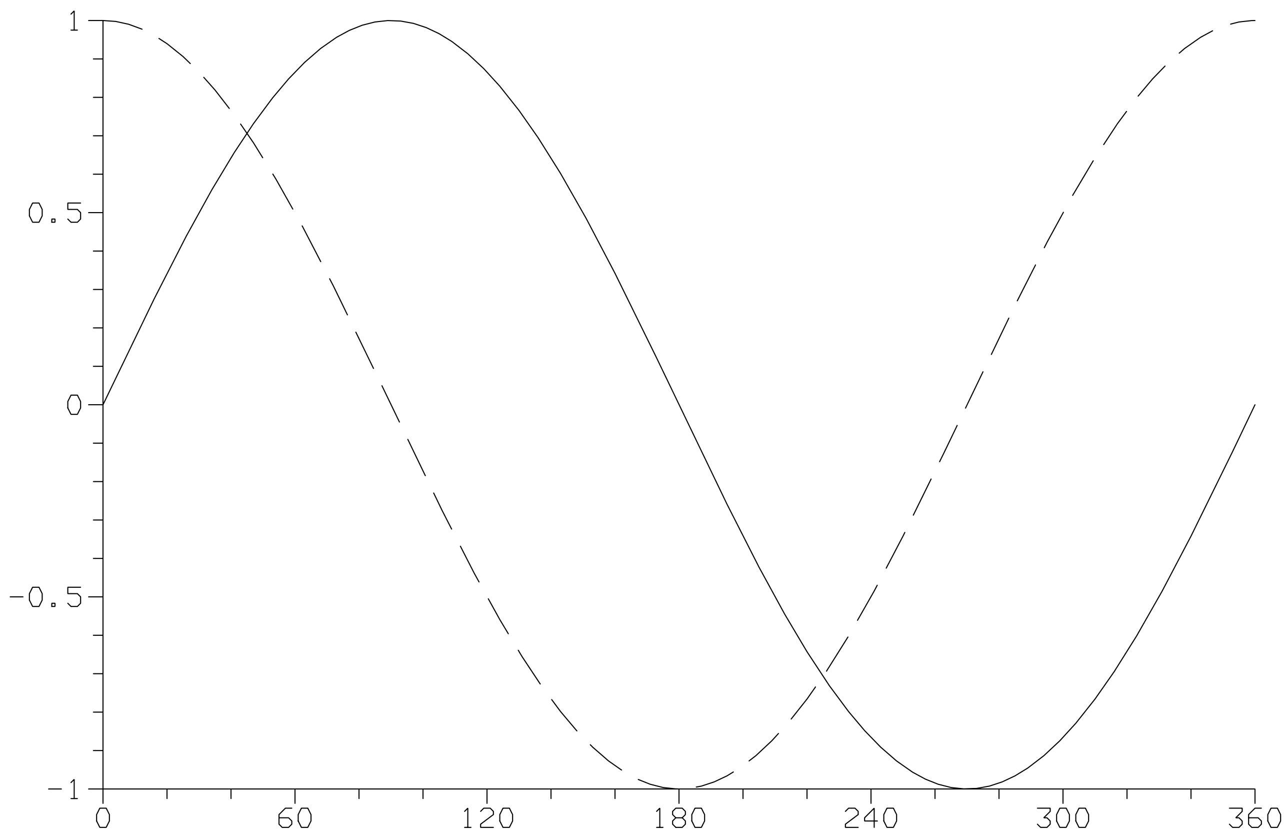
The *read* action is used to copy plot data into memory; the associated parameters and flags specify the data file name, the number of variables (= the number of columns in the plot data matrix) and the data format. Additionally, the parameter, *nextra*, is set equal to 1 in order to allocate space in memory for an additional plot vector which will be used in the next two examples.

The *select* action specifies which variables (columns in the plot data matrix) are to be used as the coordinates of each curve in the current plot, as well as other information. Normally, each curve requires a separate *select* action and all curves specified in consecutive *select* actions are scaled together. However, for simple plots with a minimum of annotation, it is possible to shortcut curve selection by simply assigning a comma-separated list of the appropriate data column numbers to the integer parameters *x* and *y*. In the *select* action for this example, the assignment, $y = 2, 3$, prescribes two curves in which the values of the dependent variable (y-coordinates) are taken from the second and third positions in the plot record respectively. By default, the values of the independent variable (x-coordinates) for each curve are assumed to occupy the first position in each record, so that *x* need not be specified in this case.

The *plot* action allows the user to specify the size and shape of the current plot, its position on the page, the scaling used to display the previously selected curves (linear or logarithmic), and other characteristics of the plot. In this example, linear axes with automatic scaling are provided by default. The *plot* action also causes the current plot to be displayed on the specified output device.

As will be seen, additional plots of data in the same file may be made by executing the appropriate *select* and *plot* actions. Once data is loaded, it remains in memory until the next *read* action.

Example 1



```
read gnufor single file="fort.14" nvar=3 nextra=1;
select y=2,3; plot;
```

5.2 Example 2

The second example is a typical annotated plot, including curve legends, axis labels and a title. In most cases, the general form of the input shown, together with the program defaults, is sufficient to produce an acceptable plot.

In the first step, the *calc* action is employed to form the average of the sine and cosine data values and the result is stored in the previously allocated fourth plot vector. The syntax here is strongly reminiscent of the Hewlett-Packard line of scientific calculators, and users familiar with Reverse Polish Notation should have little difficulty following the example.

Next, three curves are selected and a *legend* string is assigned to each curve. Note that a curve is specified by assigning an appropriate value (= the index of the variable in the plot record) to the integer parameters, *x* and *y*, in each *select* action. For the curves shown, the independent variable occupies the first position in the plot record ($x = 1$), which is the program default so that *x* need not be specified. However, it is always necessary to specify the index for the dependent variable, *y*, since its default value is undefined and curve plotting stops at the first undefined *y* value.

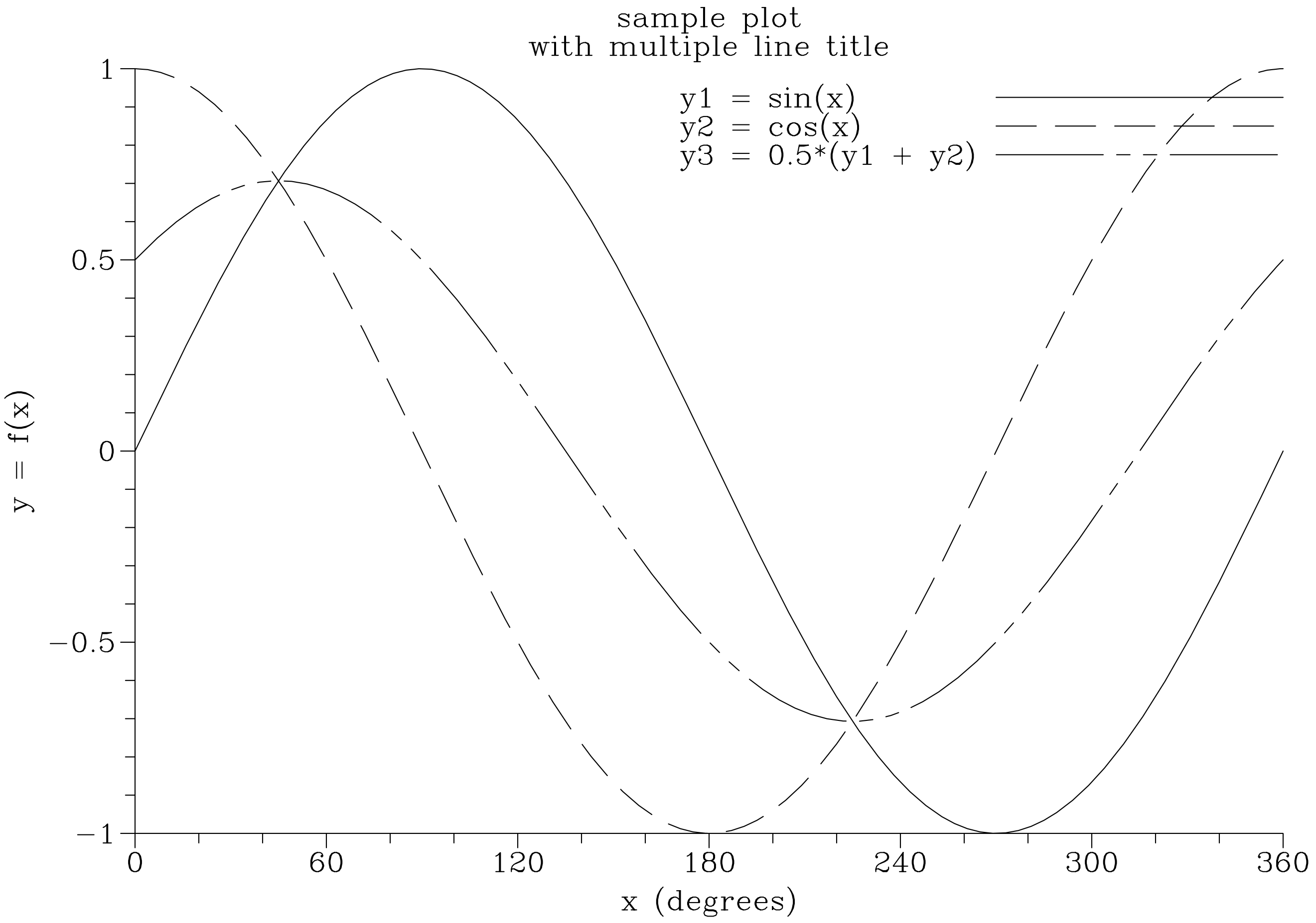
For reference, it may be noted that the only effect of a *select* action is to load internal array elements with the specified index values, legend string and other information concerning the curve; no manipulation of the data occurs. This means that, in general, the order of *calc* and *select* actions is not significant. In particular, the *calc* action for this plot could have been placed between any two successive *select* actions, or after all three curve were selected; the result would have been the same.

It is recommended that multiple *select* actions be used whenever more than one parameter is assigned for a curve; otherwise, it becomes difficult to keep track of which parameters go with which curve. In any case, multiple *select* actions are *necessary* whenever *select* flags are invoked. See the Reference Section for a more complete discussion of the *select* action.

Finally, the curves are plotted with labeled x and y axes and a title is provided. The *plot* parameter, *font*, determines the character set used for this plot (Roman). The *plot* flags, *x* and *y*, specify linearly scaled axes. The attribute, *label*, of these flags determines the axis label. Other *plot* parameters and flags may be invoked to specify other features of the plot, such as the position of the legend block and plot title and the size of the characters. In this plot, both the legend block and the title are placed at the default position, and the default character size is employed. Note that, for multiple line titles, the position is adjusted to accommodate all of the title lines.

It is important to remember that the assignment for the *label* attribute of the *plot* flags, *x* and *y*, must be enclosed in parentheses. Omission of the final parenthesis indicating the end of an attribute list is one of the more common input errors. Additionally, each action must be terminated by a semi-colon. A misplaced or omitted semi-colon will also cause an input error.

Example 2



```
calc rcl[2] rcl[3] + 0.5 * sto[4];

select y=2 legend="y1 = sin(x)";
select y=3 legend="y2 = cos(x)";
select y=4 legend="y3 = 0.5*(y1 + y2)";

plot font=1 x(label="x (degrees)")
            y(label="y = f(x)")

title="sample plot", "with multiple line title";
```

5.3 Example 3

In this example, the layout of the page is changed to accommodate 6 plots per page arranged in 2 columns of 3 plots each, and the axis lengths and overall plot scale factor are adjusted accordingly. Additionally, the default font is set to number 2 (Italic).

Next, one curve is selected and a *legend* string is assigned. In the subsequent *plot* action, the flag, *page*, is invoked to start a new page. In general, it is desirable to start a new page each time the plot layout is changed.

In the *select* action which follows, a new curve is prescribed with its own legend.* In the subsequent *plot* action, the flag, *same*, is invoked so that the second curve is drawn on the same plot as the first. Note that the default is to move to the *next* plot. At the same time, the *y* flag attribute, *new*, is invoked to obtain a new y-axis, with its own label for the second curve. The default is to use the *same* axis.

In the first plot, the axes were scaled automatically. For the second plot on this page, the x-axis scale (90 degrees/inch) is determined from user input by means of the parameter attribute, *scale,* which prescribes the number of plot units per inch.

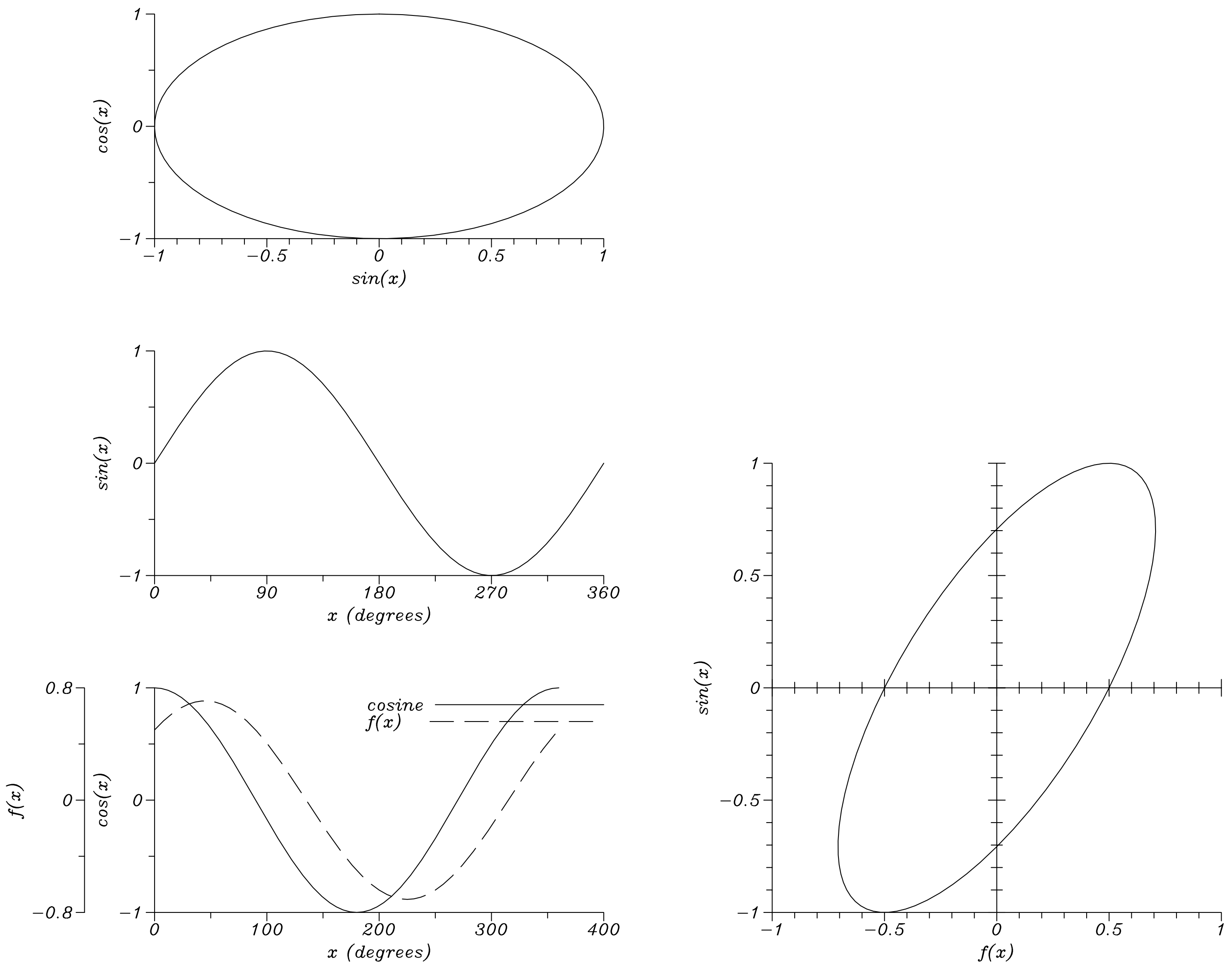
For the third plot in this example, the selected independent variable is in the second position in the input data record, while the dependent variable defining the curve is in position 3. In general, any plot vector may be selected to represent either the x- or y-coordinate of a curve.

Finally, the fourth plot in this example shows that the size of a plot may be changed at any time using the *plot* parameter, *axlens*. Additionally, the *plot* flag, *polar*, is invoked to produce the crossed axes at the center of the plot, and the offset and scale factor for each axis is prescribed to insure that x and y are plotted symmetrically.

Note that a subsequent plot on this page would be placed (by default) at its nominal position, one inch (scaled by *factor*) above the fourth plot.

* It may be noted that, in this case, when the second curve is plotted, the legend line samples do not precisely line up as in previous examples. The reason for this is quite simple. A legend block is sized by the length of the longest string in the *select* action(s) preceding a *plot* action so that the text and line samples are left and right justified respectively. Subsequent legends drawn on the same plot are left aligned with any previous legends. However, once a legend block has been drawn on a plot, the string length determining the width of the legend block is discarded so that the program can select a new maximum legend string length. (Note that the program has no way of knowing whether a subsequent legend will be drawn on the same plot or another plot.) In these circumstances, the line sample may not extend to the right edge of the legend block. If it is desired, the user may pad subsequent legend strings with blanks to make the strings have the same length. However, if a proportionally spaced font is used, the result may still exhibit a slight unevenness due to the differing character widths.

Example 3



```
layout stack=3,2 factor=0.7 axlens=4,2 font=2;

select y=3 legend="cosine";
plot page x(label="x (degrees)")
          y(label="cos(x)");

select y=4 legend="f(x)  ";
plot same y(new label="f(x)");

select y=2;
plot x(label="x (degrees)" scale=90) y(label="sin(x)");

select x=2 y=3;
plot x(label="sin(x)") y(label="cos(x)");

select x=4 y=2;
plot polar axlens=4,4
     x(label="f(x)"   start=-1 scale=0.5)
     y(label="sin(x)" start=-1 scale=0.5);
```

The next two examples use data generated by the following Fortran program, which writes to a file, "fort.15".

SECOND FORTRAN DEMO PROGRAM

```
      dimension p(3)

      do 100 i=1,9
        zeta = 0.05*(i + 1)

        do 10 n=1,11
          rn = 0.8 + 0.04*(n - 1)
          tau = 100.0*((1.0 - rn**2)**2 + (2.0*zeta*rn)**2)/(zeta*rn**4)
          p(1) = zeta
          p(2) = rn
          p(3) = tau
          write(15) p
   10     continue

  100 continue

      stop
      end
```

In this program, there is a double do-loop with a write to unit 15 in the innermost loop. Thus, the first 11 records correspond to the first value of zeta with varying values of tau and rn, which are functions of the inner loop index and zeta; the next 11 records correspond to the second value of zeta, *u.s.w.*

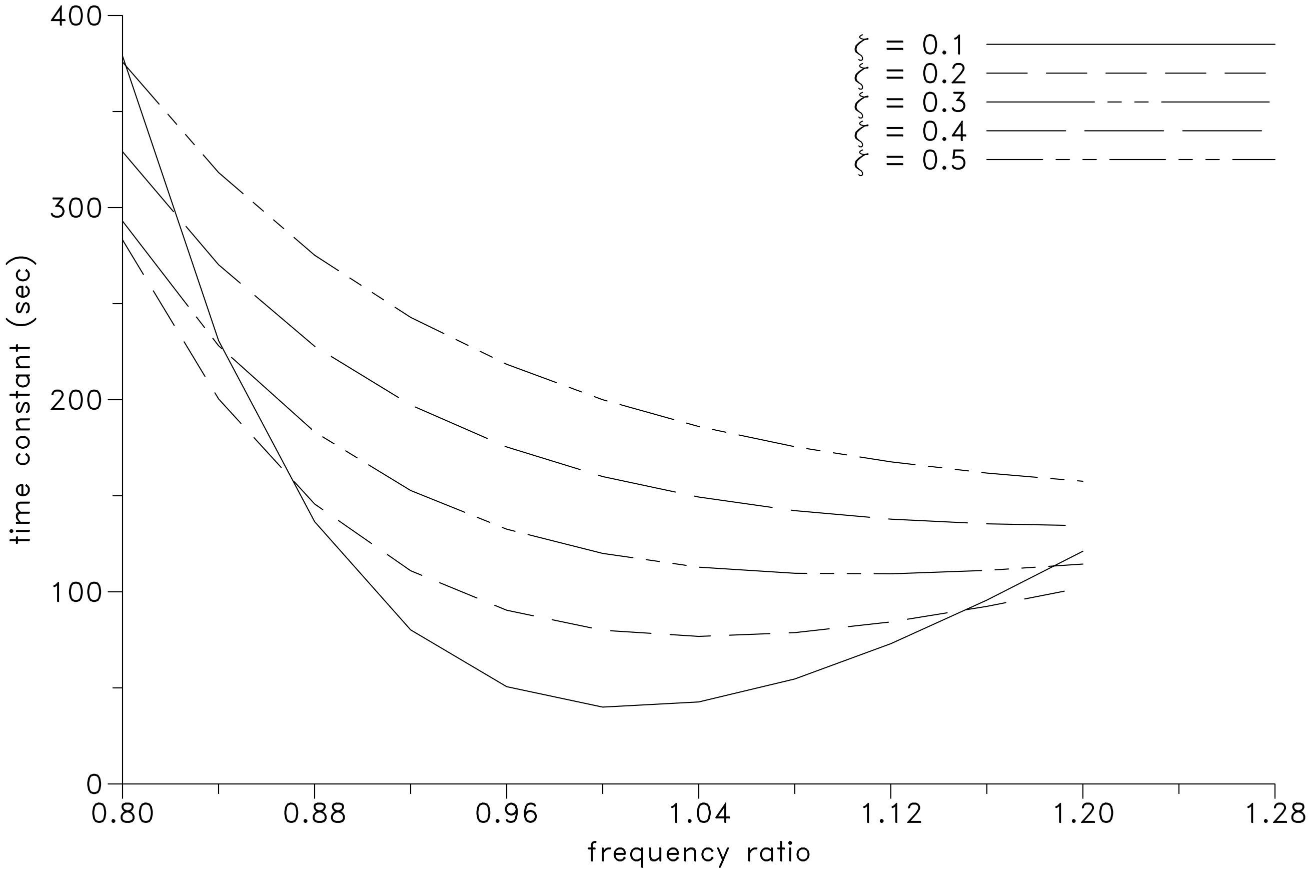
5.4 Example 4

In Example 4, the new data file is read in using the *read* action. Note that the file name must be specified since it differs from the previous name. The number of variables in the plot record is also specified, although it is not necessary in this case since it is the same as in the first *read* action. Also, the page layout is restored to its default configuration, except that the Sans-serif font (number 3) is chosen.

This example illustrates the use of the *select* parameters, *first* and *last*, to specify segments of the data for plotting. In this case, it is desired to plot values of the time constant, tau, as a function of the frequency ratio, rn, for selected values of zeta. For each curve, the independent variable is in position 2 and the dependent variable is in position 3. The particular values of zeta considered are obtained by selecting 11 consecutive records for each curve starting at records 1, 23, 45, 67, 89 respectively.

Note that the *plot* flag, *page*, is required in this case to force a new page since the *stack* in the previous example was not completed (only four plots were executed).

Example 4



```
read file=fort.15 nvar=3;
layout stack=1,1 axlens=6,4 factor=1.0 font=3;

select x=2 y=3 first=1  last=11 legend="\(z = 0.1";
select x=2 y=3 first=23 last=33 legend="\(z = 0.2";
select x=2 y=3 first=45 last=55 legend="\(z = 0.3";
select x=2 y=3 first=67 last=77 legend="\(z = 0.4";
select x=2 y=3 first=89 last=99 legend="\(z = 0.5";

plot page x(label="frequency ratio")
          y(label="time constant (sec)");
```

5.5 Example 5

In Example 5 the same data is used, but instead of plotting consecutive records, the *select* parameter, *skip*, is used to pick out specific points in the plot vector which are then connected to form the plotted curve.

Ordinarily, the function of the parameter, *skip*, is to force a reduction in the density of points actually plotted. This is particularly useful if special centered symbols are selected, since, in this case, the built in data compression algorithm is not invoked. However, it is also possible to accommodate certain cyclic arrangements of plot data by appropriate specification of the parameter, *skip*. In order to employ this feature effectively, it is important to know how the *skip* parameter is used in McPlot.

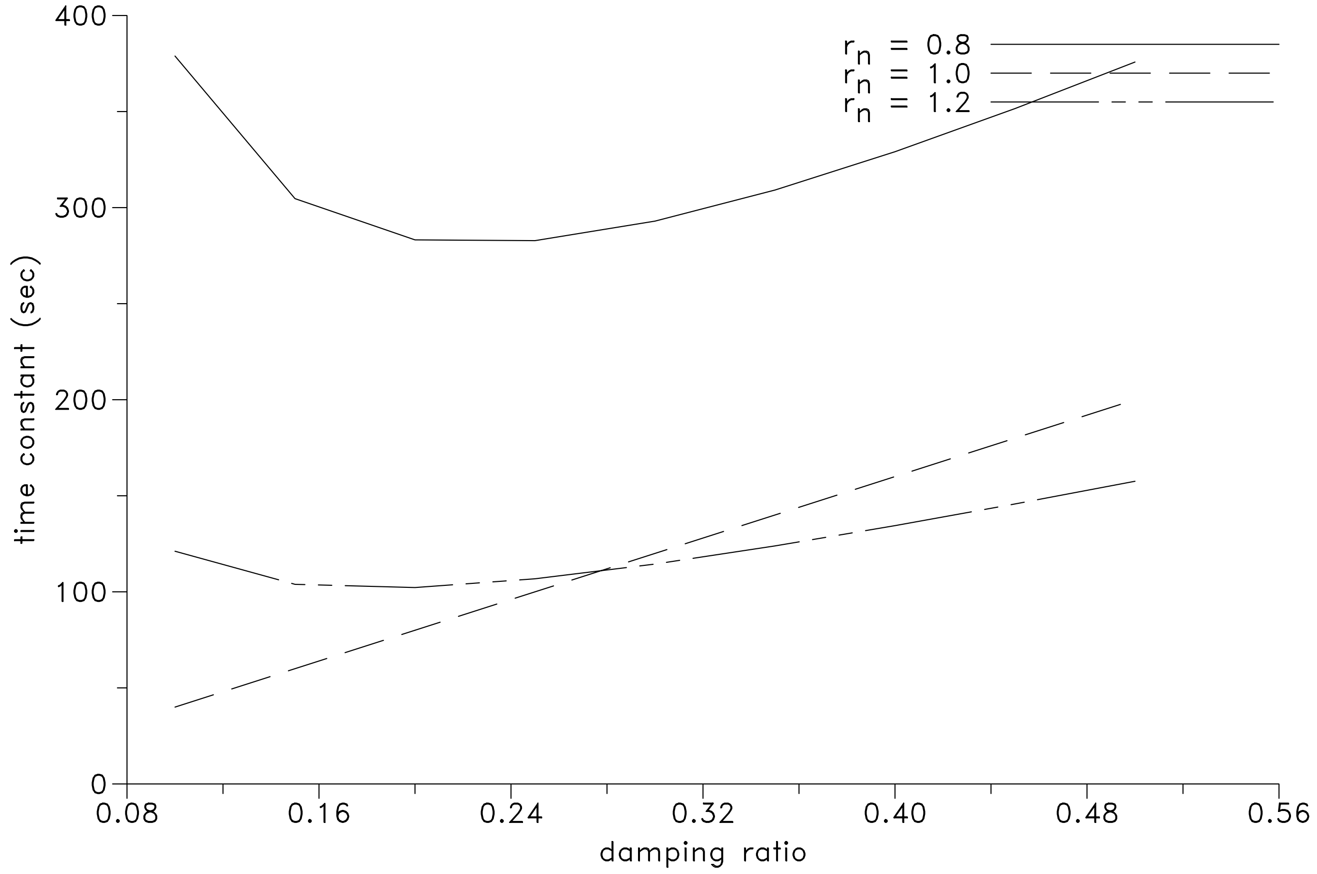
Once a set of curves has been selected, the maximum and minimum values are determined to obtain the appropriate scaling factors and the axes are drawn. Next, the data values are scaled and, if data smoothing is requested, a cubic spline interpolation is performed. It is only when the points in the transformed plot vector are actually being drawn on the plot that the parameter, *skip*, is finally used. As each point is processed, a special centered symbol and/or connecting line is drawn only for those points whose index, modulo *skip*, is zero.

Evidently, then, if, as in this example, the parameter, *skip*, is used to accommodate a cyclic arrangement of the data, some functionality may be lost. In particular, data smoothing may produce undesirable results and axis scaling and the maximum, minimum and final value annotations will be based on all of the data in the plot vector, not just the points selected by the value of *skip*. However, all other McPlot features are available in this case as for the nominal sequential arrangement of data.

In this plot, the selected independent variable is in position 1, the dependent variable is in position 3 and *skip = 11*, so that only every eleventh record is used to draw the curve. With this choice, referring to the Fortran program used to generate the data, one obtains a plot of the time constant, tau, as a function of the damping ratio, zeta, for specific values of the frequency ratio, rn. The particular value of the frequency ratio obtained depends on the initial record. By starting with records 1, 6 and 11 respectively one obtains the curves shown.

It may be noted that, for this plot, the curves are displayed in separate *plot* actions in order to illustrate the fact that the axes are scaled with respect to all of the data in a plot vector, including skipped records, Ordinarily, it is more efficient (and convenient) to define the curves in consecutive *select* actions and then display them in a single *plot* action. This procedure will also insure that the axis scaling is appropriate for all of the curves which are intended for the same plot.

Example 5



```
select x=1 y=3 skip=11          legend="r\(−n\(+ = 0.8";
plot x(label="damping ratio")
     y(label="time constant (sec)");

select x=1 y=3 skip=11 first=6  legend="r\(−n\(+ = 1.0";
plot same;

select x=1 y=3 skip=11 first=11 legend="r\(−n\(+ = 1.2";
plot same;
```

For the next three examples, the plot data was generated using the C Language program shown below.

C DEMO PROGRAM

```
#include <math.h>
#include <stdio.h>
main()
{
  FILE *fp;
  double a, b, t, r = 0.0, f[5];

  fp = fopen("c.1.dat","w");

  for(t = 0.0;t <= 6.0;t += 0.05) {
    if(t > 1.0 && t < 2.0) {
      a = 0.2;
      b = 0.1;
    }
    else if(t > 3.0) {
      a = -0.2;
      b = -0.3;
    }
    else {
      a = 0.0;
      b = 0.0;
    }

    r += 0.01;

    f[0] = t;
    f[1] = r*cos(t);
    f[2] = r*sin(t);
    f[3] = sin(t) - b;
    f[4] = 0.5*cos(t) - a;

    fwrite(f,sizeof(double),5,fp);
  }
}
```

The output in this case is written to the file, "c.1.dat", and consists of 121 unformatted binary records, each of which comprises 5 double precision real variables.

The first position in the record is time in seconds. The next two variables are the x- and y-coordinates of a spiral curve. The last two variables are essentially sine and cosine functions with discontinuous portions.

5.6 Example 6

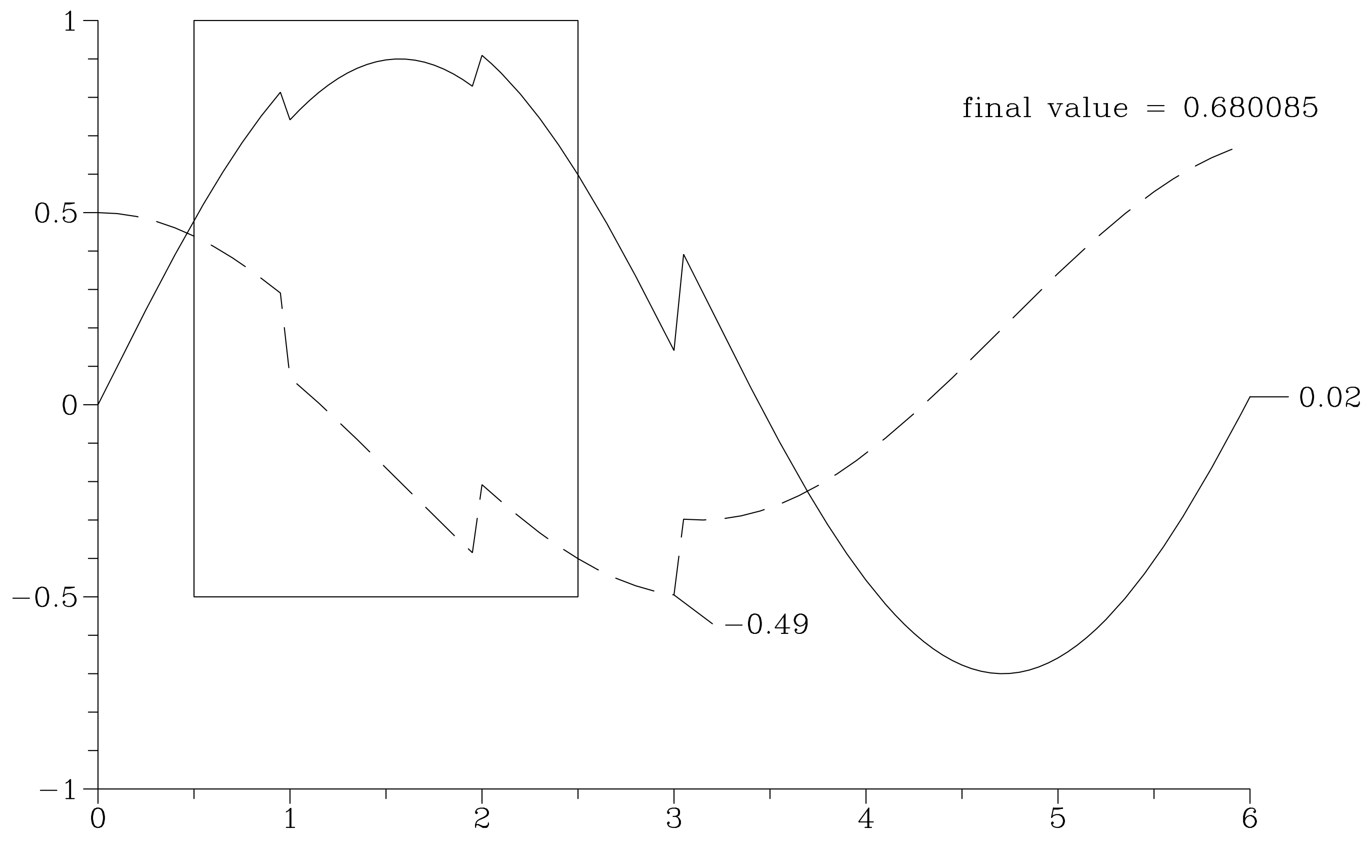
In Example 6, the new data file is read in using the *read* action. Note that a full specification of *read* parameters and flags is required.

In the *select* actions which follow, the flag, *note*, is invoked to obtain the final value of the first curve and the minimum and final values of the second curve. In the second case, the final value is labeled, the precision is set to 6 places after the decimal and the position of the annotation is prescribed. The attributes, *x* and *y*, define the position of the start of the annotation in inches relative to the current plot origin.

The curves are then plotted using the default linear axis scaling.

In the next step, a box is drawn around a segment of the plot using the *put* action. The position of the upper left-hand corner of the box is specified by the parameter attributes, *x* and *y*, while the width and height of the box are determined by the parameter attributes, *wid* and *ht*. In this case, it is desired to give the position coordinates and dimensions in plot units, so the *put* flag, *scale*, is invoked. The default is to interpret the input position and dimensions in *inches*.

Example 6



```
read c double file="c.1.dat" nvar=5;
layout font=1;

select y=4 note(fv);
select y=5
note(min fv(x=4.5 y=3.5 ndec=6 label="final value = "));
plot;

put box(x=0.5 y=1 wid=2 ht=1.5) scale;
```

5.7 Example 7

In the next example, the same curves are selected. However, by specifying the *plot* parameters, *start* and *end*, which determine the range of the independent variable for all of the selected curves, only the boxed region of the previous plot is displayed. Note that the plot is automatically scaled according the specified range of the independent variable.

In the next step, a segment of the first curve is boxed using the *put* action as in the previous example.

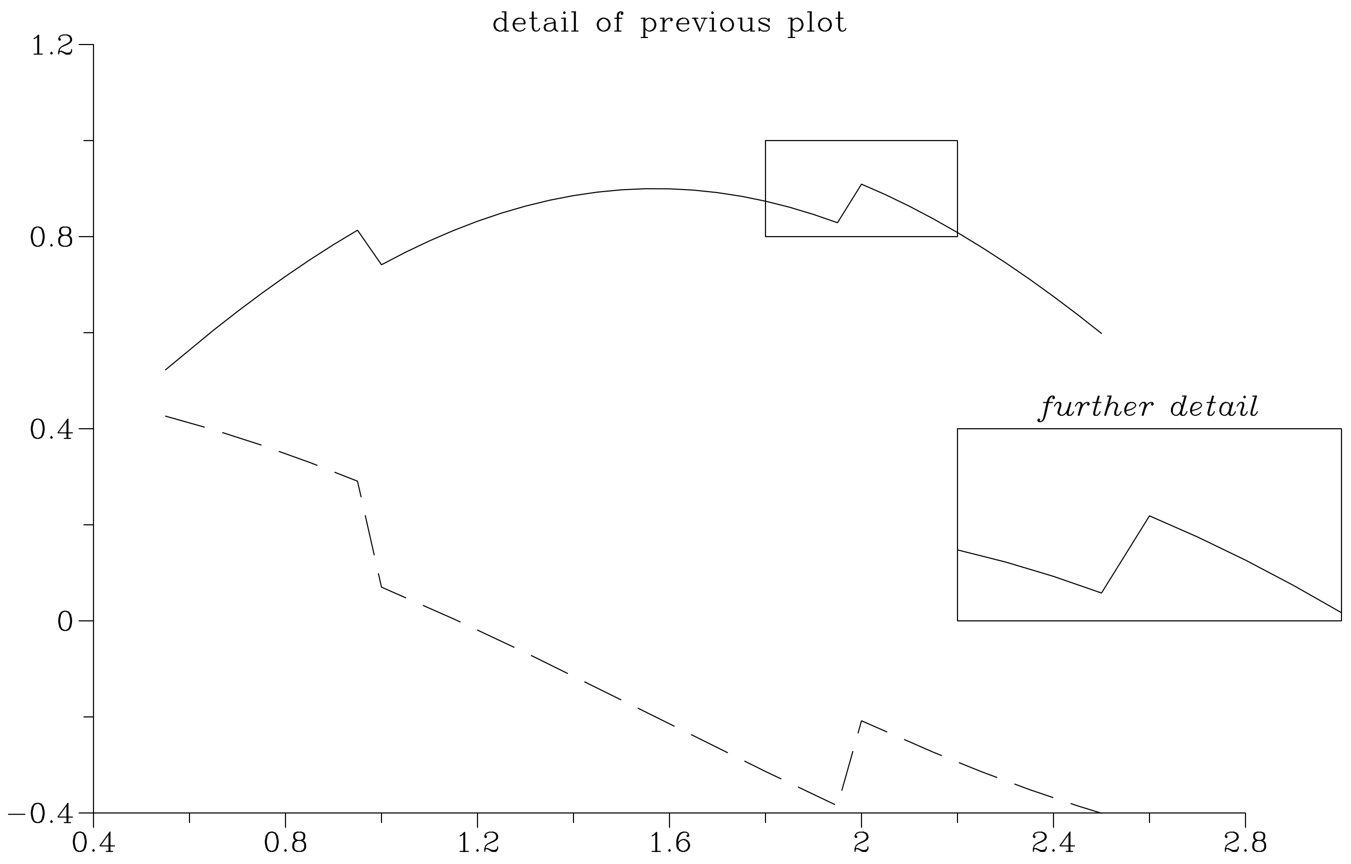
In the subsequent *select* action, the fourth plot vector is again specified and the parameters, *start* and *end*, are used to limit the range of the independent variable defining the curve to the new boxed region.

This curve segment is then plotted on the same page by invoking the *plot* flag, *nopage* (note that the current default is one plot per page).

In the *plot* action, the parameter, *origin*, specifies the (x,y) position of the origin of the current plot in inches relative to the origin of the previous plot. The x- and y-axis lengths (in inches) are determined by the parameter, *axlens*. The *plot* flags, *nox* and *noy*, specify linear scaling; however, the axes (and annotation, if any) are not actually drawn on the plot.

In the final action, a box is placed around the small plot to further define it. Note that, in this case, the position and dimensions of the box are given in *inches* (the default) and that the position of the box is specified relative to the origin of the current (small) plot.

Example 7



```
select y=4,5;
plot start=0.5 end=2.5 title="detail of previous plot";

put box(x=1.8 y=1.0 wid=0.4 ht=0.2) scale;

select y=4 start=1.8 end=2.2;
plot nopage origin=4.5,1 axlens=2,1 nox noy
     title(font=2)="further detail";

put box(x=0 y=1 wid=2 ht=1);
```

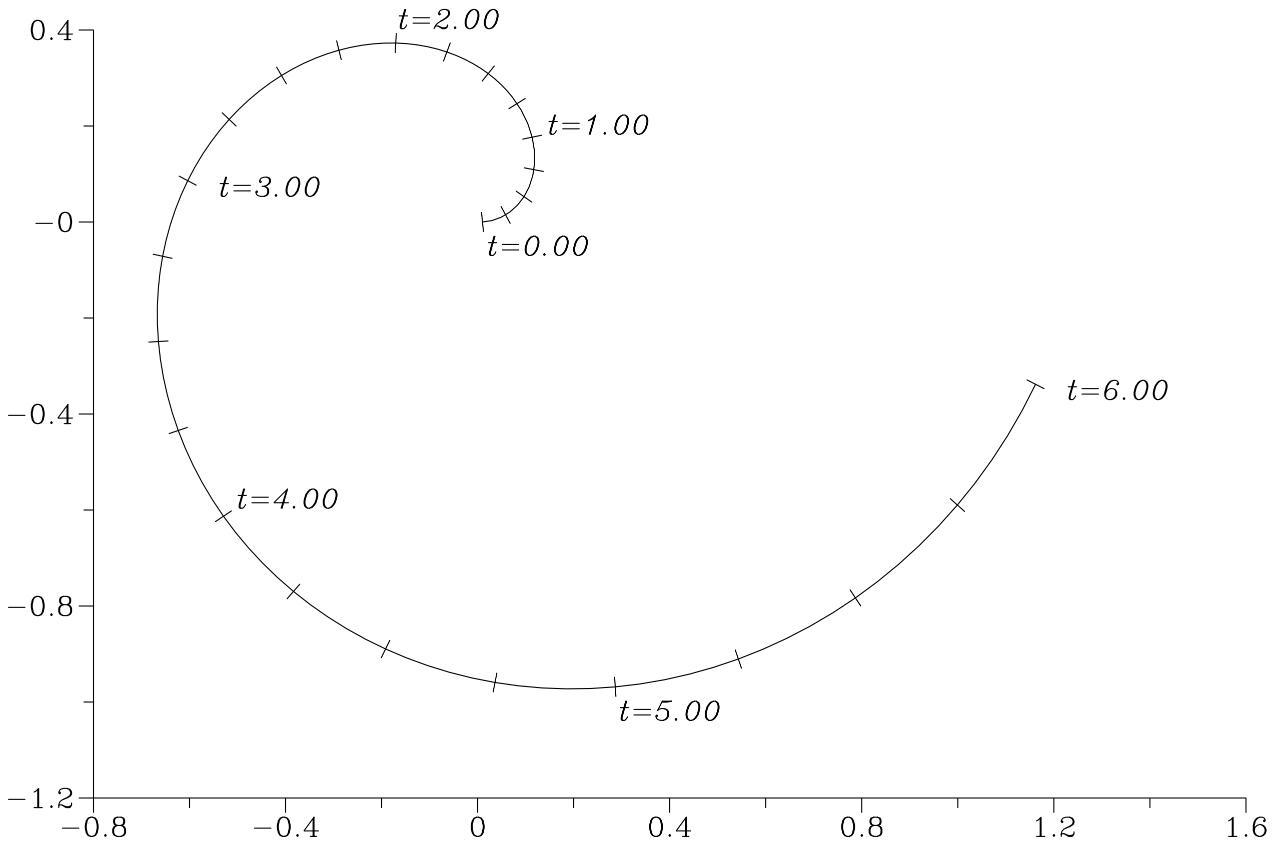
5.8 Example 8

Example 8 shows an application of the *select* flag, *ticks*, to draw annotated tick marks at intervals along the curve.

Tick marks may be drawn at specified intervals along the curve as a function of a tick control variable which is simply one of the input plot vectors whose column number is determined by the user (the default is $var = 1$). The tick marks may be annotated with the corresponding values of yet another variable (plot vector) which may be selected by the user (the default is to use the same variable number as specified for the tick control variable). The annotations may also be labeled and the interval between annotation values independently prescribed by the user.

In this case, the second and third positions in the plot record are selected to define the curve. The attributes of the flag, *ticks*, give the spacing, *delta*, between tick marks and annotation values, as well as the *label* string. Note that default values ($var = 1$) are used for both the tick control and annotation variables. See the Reference Section for a more complete description of the *ticks* option.

Example 8



```
select x=2 y=3
ticks(delta=0.25 note(delta=1.0 label="t=" font=2));
plot;
```

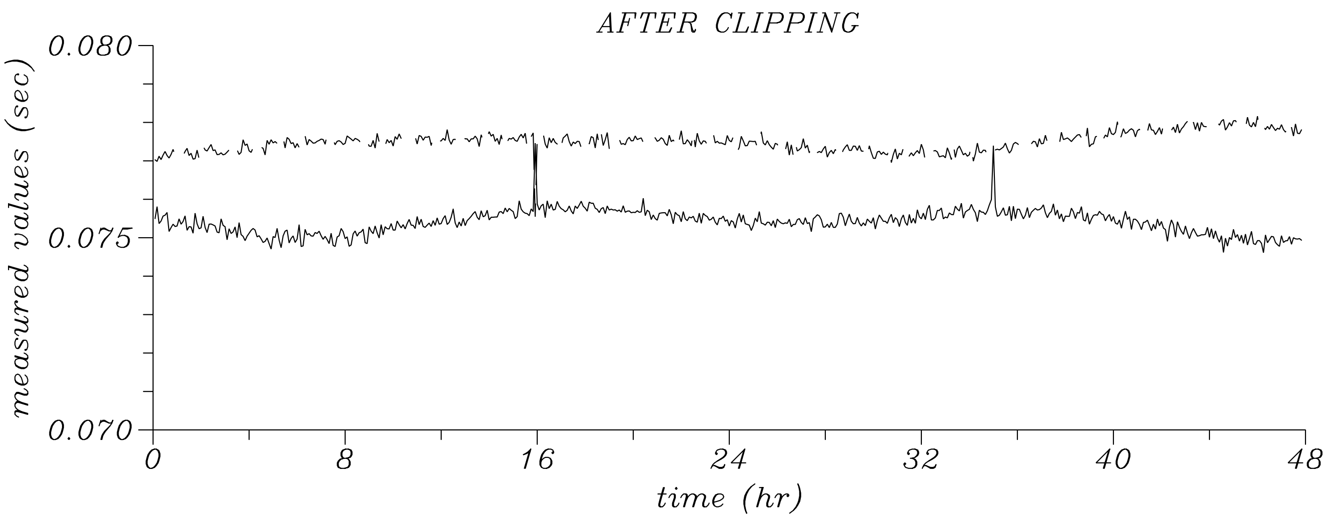
5.9 Example 9

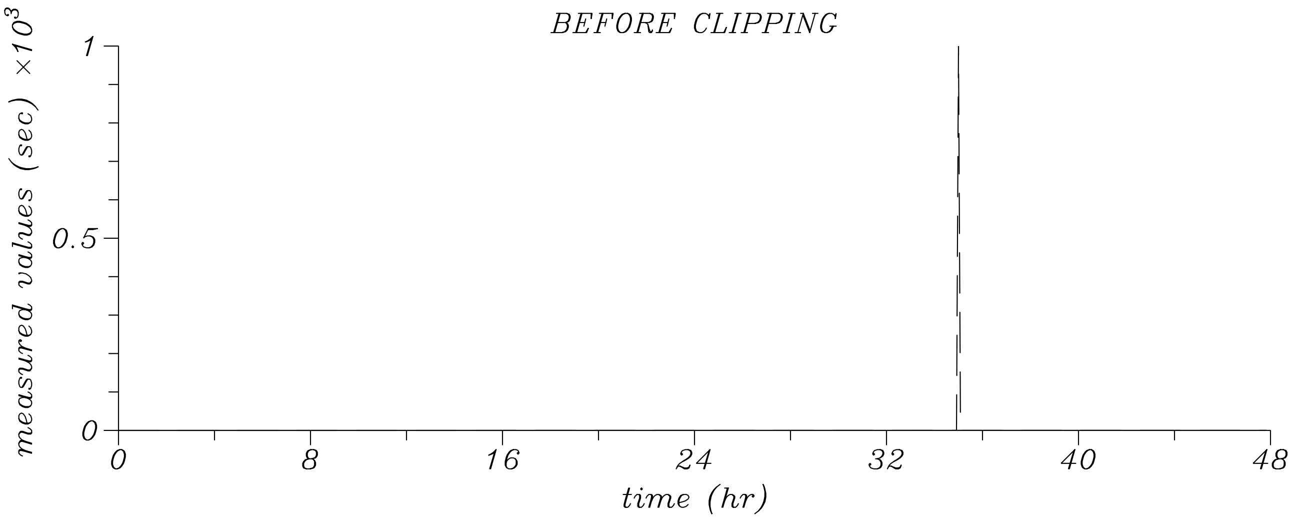
Example 9 illustrates the data clipping option in McPlot. It uses a set of measured time values taken over a 48 hour period. The data reside in a single precision Fortran binary file, "fort.16", comprising 575 records. The first variable in each record is the elapsed time (in seconds) relative to the start of the experiment, and the next two variables are the actual measurements (sec).

After reading in the data, changing the plot layout and converting the elapsed time to hours, the measurements are plotted with default scaling and no clipping. Evidently, there is an invalid data point about 35 hours after the start of the experiment.

In this case, it is known that the measured values should lie between 70 and 80 msec. Thus, in the next plot, the flag, *clip*, is invoked to "lift the drawing pen" whenever the y-coordinate of the curve falls outside the range, $0.07 \le y \le 0.08$. The result is a considerably improved representation of the data.

Example 9





```
read gnufor single file=fort.16 nvar=3;

layout stack=2 axlens=,2 font=2;

calc rcl[1] 3600 / sto[1];        % convert time to hours

select y=2,3;
plot page x(label="time (hr)") y(label="measured values (sec)")
     title="BEFORE CLIPPING";

select y=2,3;
plot clip(y(max=0.08 min=0.07))
     x(label="time (hr)") y(label="measured values (sec)")
     title="AFTER CLIPPING";
```

5.10 Example 10

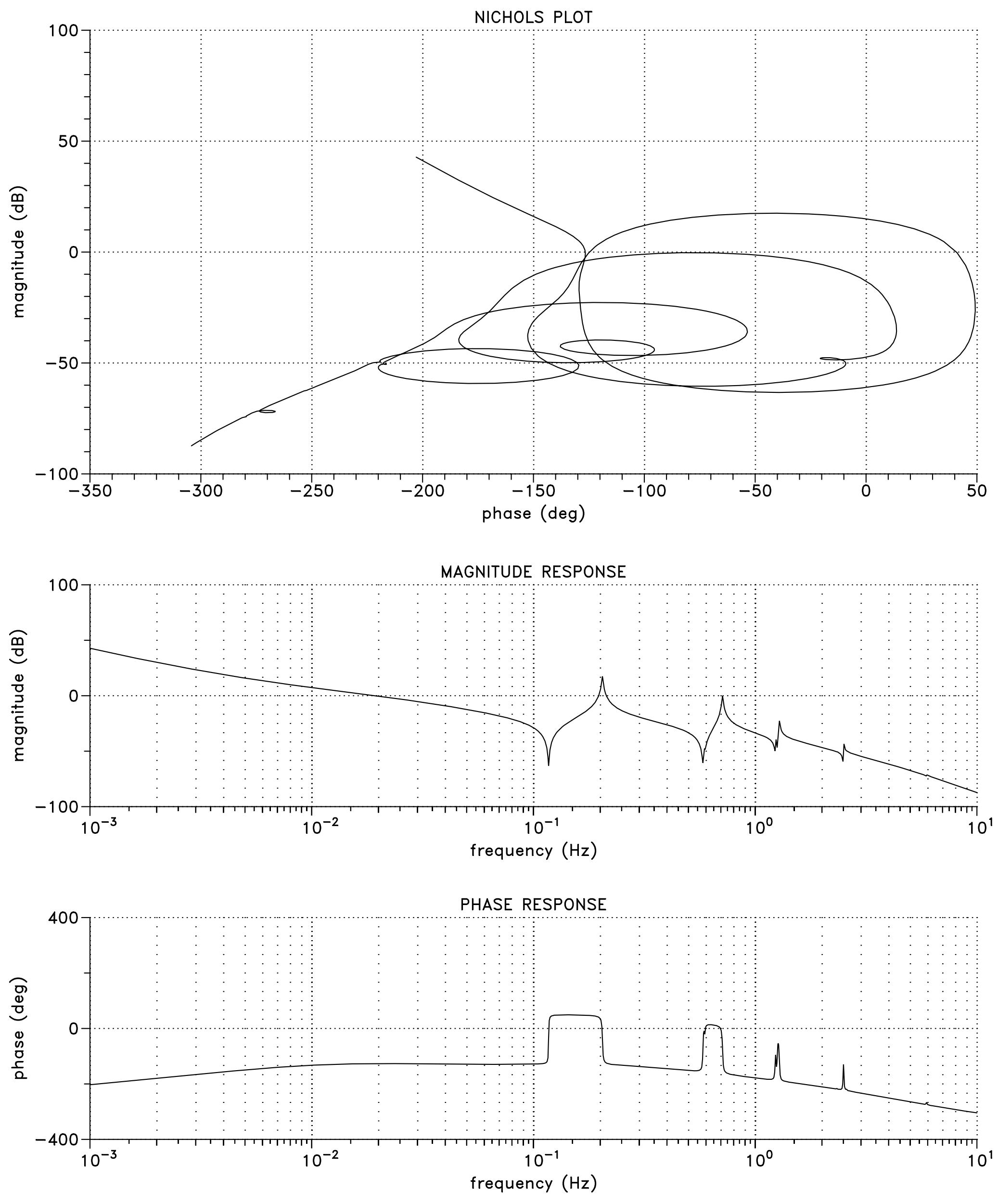
Example 10 illustrates the default gridding and logarithmic axis options. The data comprises 3322 records with three single precision variables in each record. The first position contains the frequency; the second and third hold magnitude and phase values. The data is stored in the C binary file, "c.2.dat".

After reading in the data, changing the plot layout and converting the magnitude values to dB, the phase and magnitude responses and a Nichols plot are generated.

In each case, the *plot* flag, *grid*, is invoked to provide the default gridding. It may be noted that the same result could have been achieved by invoking the *layout* flag, *grid*, which specifies gridding by default for all subsequent plots. See the reference section for a complete discussion of the *grid* option.

In the first two plots, a logarithmic x-axis is specified by means of the *plot* flag, *logx*. In general, either the x- or y-axis or both may be plotted using a logarithmic scale by invoking the *plot* flags, *logx* and/or *logy*.

Example 10



```
read c single file=c.2.dat nvar=3;
layout stack=3 axlens=8,2 factor=0.6 font=3;

calc rcl[2] log10 20 * sto[2];      % convert to dB

select x=1 y=3;
plot page grid title="PHASE RESPONSE"
logx(label="frequency (Hz)") y(label="phase (deg)");

select x=1 y=2;
plot grid title="MAGNITUDE RESPONSE"
logx(label="frequency (Hz)") y(label="magnitude (dB)");

select x=3 y=2;
plot grid axlens=,4 title="NICHOLS PLOT"
x(label="phase (deg)") y(label="magnitude (dB)");
```

For the next three examples, the data is contained in an ASCII file, "ascii.dat", which is reproduced below.

ASCII DATA

0	-0.09177
1	-0.09230
2	-0.09092
3	-0.08570
4	-0.10434
5	-0.11274
6	-0.11451
7	-0.11274
8	-0.10434
9	-0.08570
10	-0.09092
11	-0.09230
12	-0.09177
0	-0.09819

The last value in the second column is a time average, and is not, strictly, part of the data to be plotted. Its use will be indicated in Example 12.

5.11 Example 11

Example 11 illustrates the cubic spline interpolation of sparse data and some of the capabilities of the *put* action.

In the first step, the data is read in and extra space is allocated for an additional variable. Then, the plot layout is restored to the default configuration.

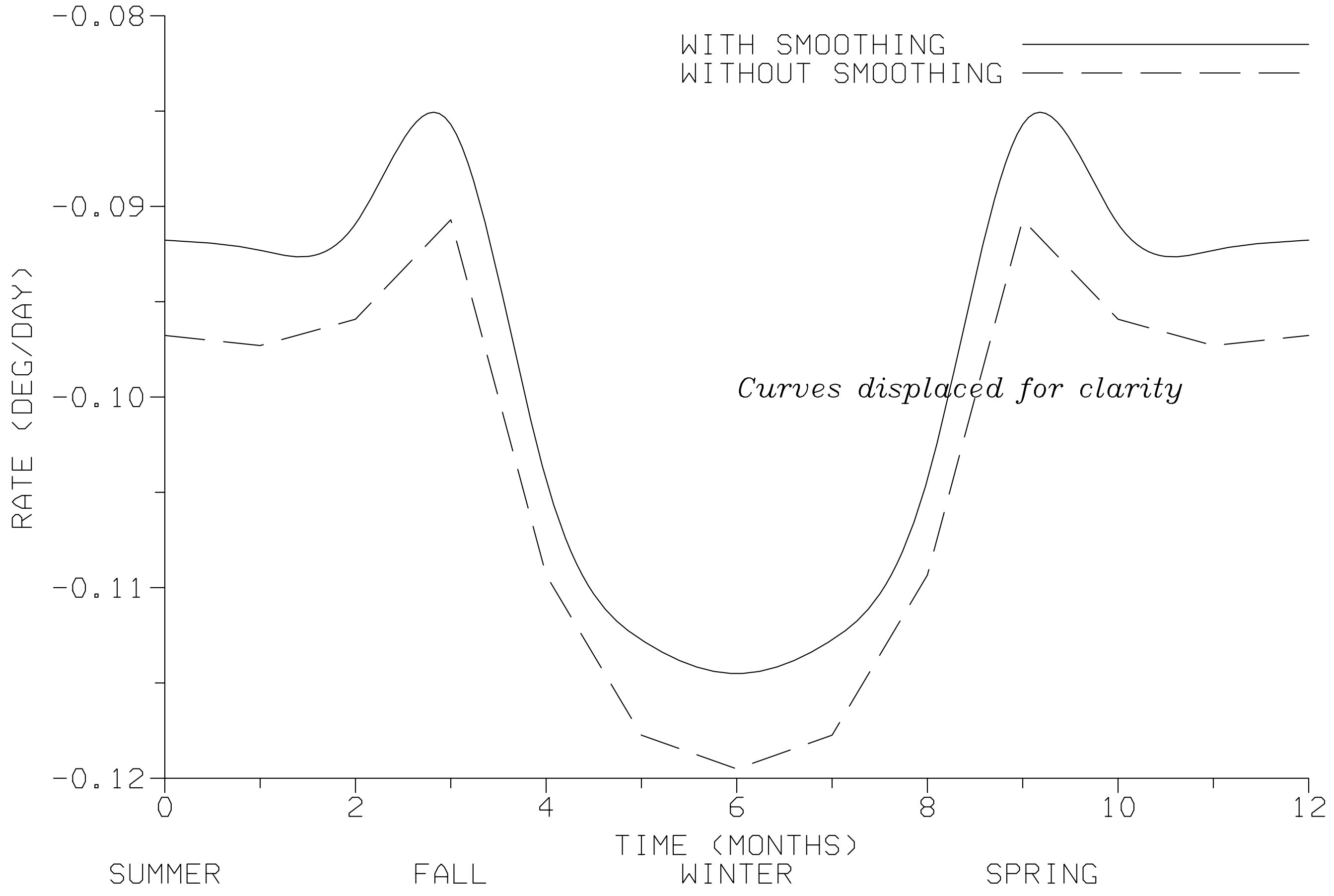
The first curve is selected and, in the subsequent *plot* action, the flag, *smooth*, is invoked to yield the curve shown.

The *calc* action is then used to create a new plot vector which is essentially the same as the original except that it is displaced in y.

This new plot vector is then selected and, in the second *plot* action, the displaced curve is drawn without smoothing for comparison on the same plot.

Finally, the *put* action is employed to provide additional annotation. Note that the position of the *put* strings is specified in inches relative to the origin of the plot. In general a string may be placed anywhere on the page, at any angle relative to the horizontal and with any character size. See the Reference Section for further discussion of the *put* action.

Example 11



```
read ascii file=ascii.dat nvar=2 nextra=1;

layout stack=1,1 factor=1 axlens=6,4;

select x=1 y=2 legend="WITH SMOOTHING   ";
plot smooth last=13                    % all but the last point
     x(label="TIME (MONTHS)") y(label="RATE (DEG/DAY)") page;

calc rcl[2] 0.005 - sto[3];           % move second curve down

select x=1 y=3 legend="WITHOUT SMOOTHING";
plot same last=13;                     % all but the last point

put str(x=3 y=2 font=2)="Curves displaced for clarity";
put str(x=-0.3 y=-0.55)="SUMMER";
put str(x= 1.3 y=-0.55)="FALL";
put str(x= 2.7 y=-0.55)="WINTER";
put str(x= 4.3 y=-0.55)="SPRING";
```

5.12 Example 12

In this example, the *put* action is used to draw a line representing the time average of the data and to annotate the average value. The last record of the input data file is used to place the annotated line automatically.

As before, a smooth curve is drawn through the data. In addition, the *plot* flag, *page*, is invoked with the attribute, *lands*, to place the x-axis of the plot along the long axis of the page. Note that the plot orientation may be changed on each new page.

In the next action, the original curve is selected with flags, *nosym* and *noline*. With this specification, a single dot is placed at each point defining the curve.

When the curve is drawn on the same plot, only the last record is included ($first = 14$). The result is that a single point is drawn on the plot with an abscissa (x) value of 0 and whose ordinate (y-coordinate) is the required average value given in the last record of the data. This procedure serves to set the default values for the *put* position attributes at the value of these coordinates.

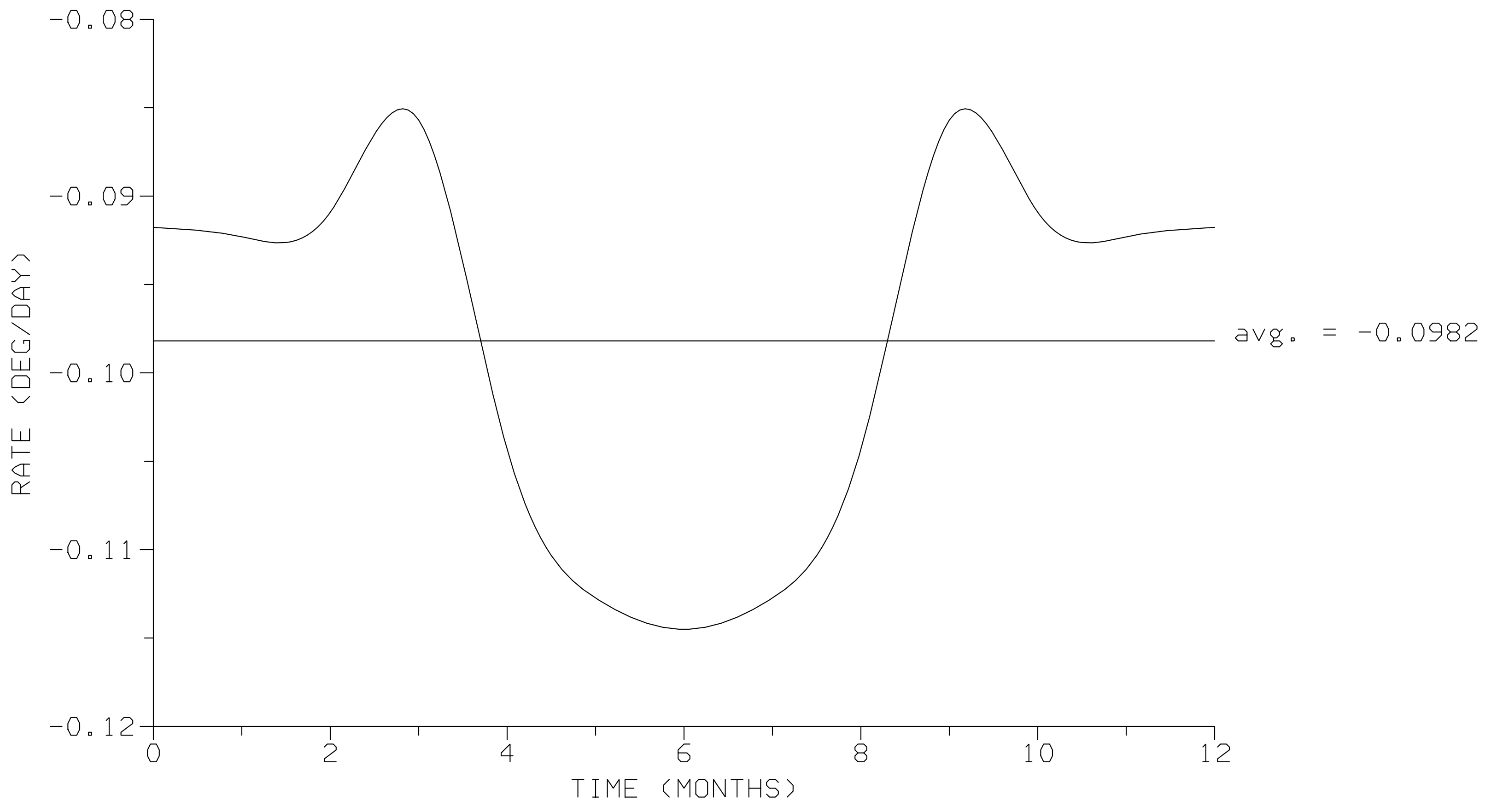
Having set the defaults, a line may be drawn across the plot at the required average value by specifying only the *from* and *to* x-coordinates. In the example, plot units (*scale*) are used to specify the coordinates. The same result could be obtained from the input,

```
> put line(from=0 to=6);
```

in which the coordinates are specified in inches.

Finally, the *put* flag, *data*, is used to place the annotated value contained in the last record at the default position, which is the last point plotted and corresponds to the right end of the previously drawn line.

Example 12



```
select x=1 y=2;
plot smooth page(lands) last=13       % all but the last point
     x(label="TIME (MONTHS)") y(label="RATE (DEG/DAY)");

select x=1 y=2 nosym noline;
plot same first=14;          % put a dot at the average value

put line(from=0 to=12 type=0) scale;      % then draw the line

put data(var=2 rec=14 label=" avg. = " ndec=4);
```

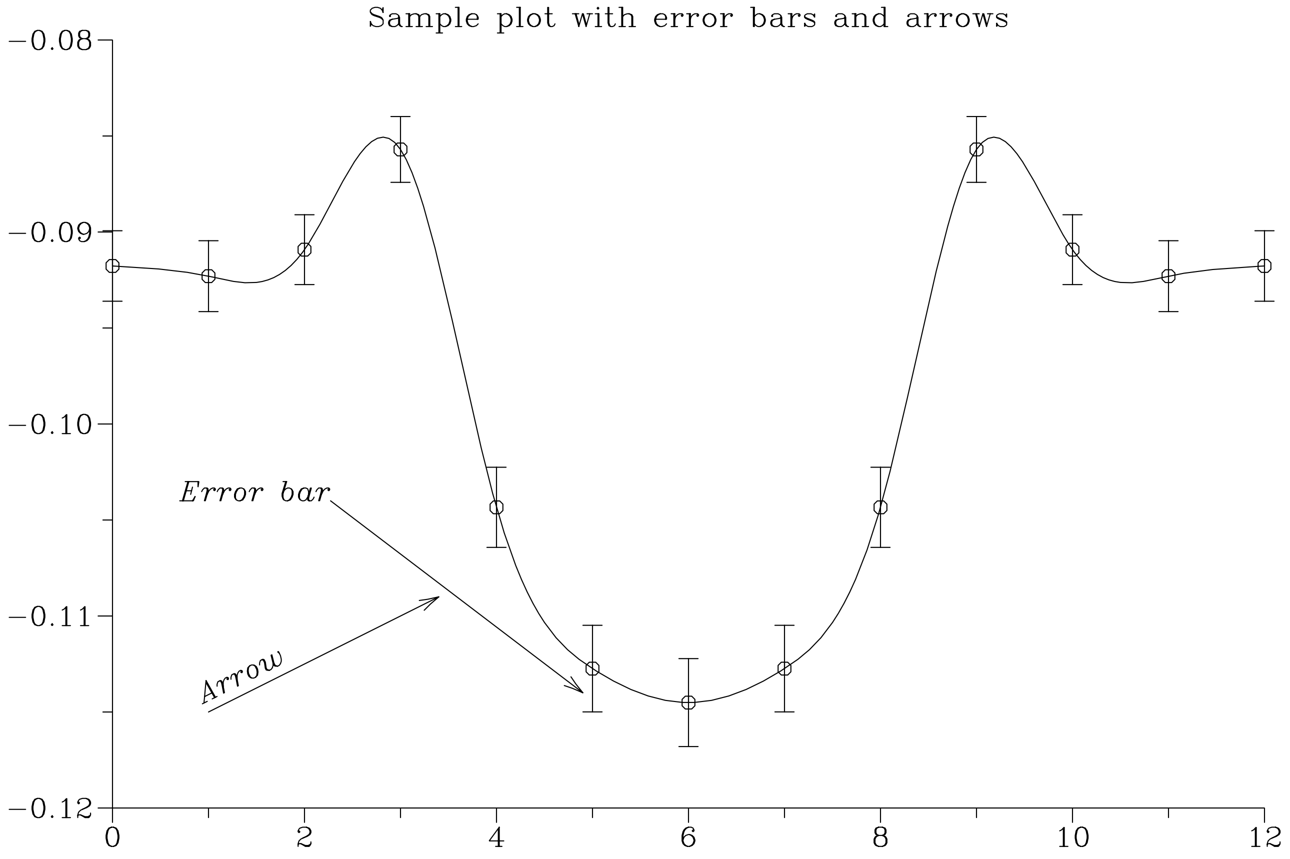
5.13 Example 13

This example shows an application of the *select* flag, *ebars*, to draw error bars for specified data points using error values contained in the data file. Additionally, the *put* flag, *arrow*, is illustrated. The same data is used as for the two previous examples.

In the *calc* action shown, the third plot vector is assigned possible error values which represent two percent of the ordinate of the original curve. In the subsequent *select* action, this plot vector is chosen to control the length of the error bars using the attribute, *var*, of the flag, *ebars*, and unconnected centered symbols are drawn at each point along the curve. Note that the possible error is assumed to be plus or minus the value contained in the specified plot vector. Finally, a smooth interpolation is drawn through the data points.

The *put* flag, *arrow*, may be used to draw attention to selected features of the plot. In the first *put* action, a string is placed on the plot and an arrow is drawn from the end of the string to a designated point in plot (scaled) coordinates. Note that only the *to* endpoint of the arrow need be specified in this case. In the final *put* actions, an arrow is first drawn between two specified points on the plot (in inches). Then, a string is drawn along the shaft of the arrow. In this case, the angle at which the string is drawn may be computed from the positions given for the endpoints of the arrow. The special vertical shift character, "\(+", is used to move the string away from the line segment in order to increase legibility.

Example 13



```
layout font=1;

calc rcl[2] 0.02 * sto[3];              % estimated uncertainty is 2 percent.

select y=2 sym noline last=13 ebars(var=3);   % plot points and error bars.
plot title="Sample plot with error bars and arrows";

select y=2 last=13;
plot same smooth;    % now draw a smooth curve through the data points.

put str(x=0.7 y=−0.104 font=2)="Error bar"
    arrow(to=4.9,−0.114) scale;

put arrow(from=0.5,0.5 to=1.7,1.1);
put str(x=0.5 y=0.5 ang=26.57 font=2)="\(+Arrow";
```

Spherical Map

The next 7 examples apply to the spherical map (*sphere*), in which input data points are interpreted in terms of spherical polar coordinates of a point in space and are plotted relative to a unit sphere as seen from the perspective of a specified observer.

5.14 Example 14

This example illustrates the definition of azimuth and elevation angles used in McPlot.

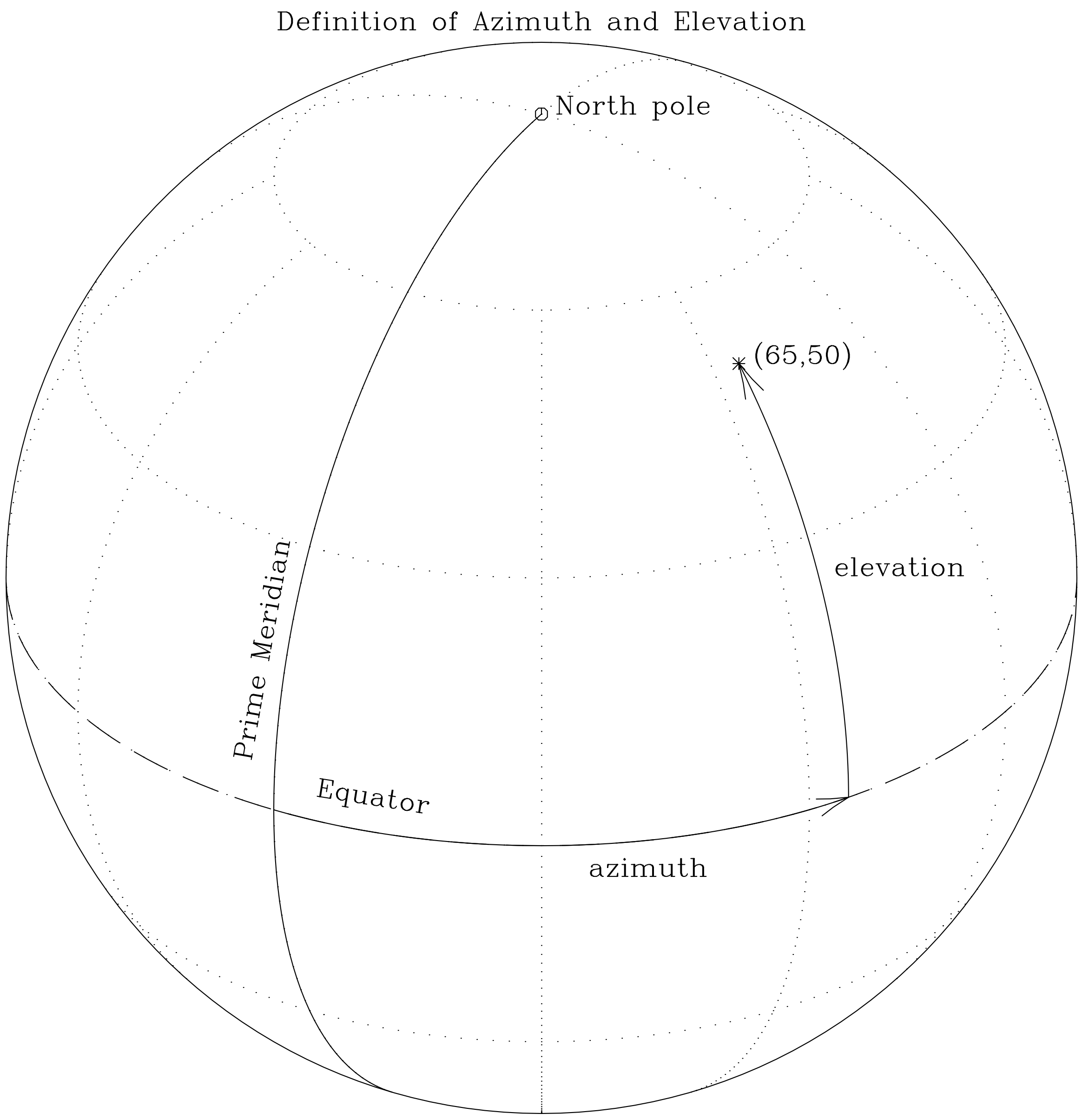
In the first two actions, the spherical map is selected, the radius of the reference sphere, as drawn on the page, is set to three inches and the viewpoint is specified relative to the perspective of an observer located at 30° azimuth, 30° elevation and infinitely far from the center of the sphere (default). The subsequent *plot* action causes the reference sphere to be drawn on the page. In this case, the program defaults are used for the grid style and spacing of latitude and longitude lines.

Next, a special centered symbol is placed at the North pole of the sphere. By default, the interpretation of the vector attribute, *pos*, giving the location of the symbol is in terms of azimuth and elevation values (in decimal degrees) relative to the sphere. The format is, *pos* = **az**, **el** [, **d**], where **az** and **el** are the azimuth and elevation angles and **d** is the (optional) radial distance from the center of the reference sphere in units of the sphere radius. If it is not specified, the default is **d** = 1 so that the position of the point is on the surface of the sphere. If *pos* is not specified, the default is the last point plotted relative to the sphere so that it is not necessary to specify the position of the string labeling this point.

In the subsequent actions, great circle lines are drawn along the prime meridian and equator using the *put* flag, *line*. The format used to specify the vector parameters, *from* and *to*, is the same as for *pos*. Note that, if the starting and ending points specifying a great circle are antipodes, the actual path is indeterminate. The program resolves this ambiguity by choosing a path through one of the poles. Hence, for the equator, the azimuth values given are slightly less than 180 degrees apart to insure that the intended path is followed.

Finally, arrows are drawn, with appropriate labels, indicating the azimuth and elevation components of a point at azimuth, 65°, elevation, 50°, and a special centered symbol is placed at the point.

Example 14



```
map sphere;                                   % choose spherical map
layout radius=3 azo=30 elo=30 font=1;
plot title="Definition of Azimuth and Elevation";

put sym(pos=0,90 type=0) str=" North pole";

put line(from=0,90 to=0,−90) str(pos=−3,5 ang=80)="Prime Meridian";

put line(from=−59,0 to=119,0 type=1) str(pos=5,3 ang=−9)="Equator";

put arrow(from=0,0 to=65,0 size=0.2);
put str(pos=35,−4)="azimuth";

put arrow(from=65,0 to=65,50 size=0.2);
put str(pos=67,25)="elevation";

put sym(pos=65,50 type=11) str=" (65,50)";
```

5.15 Example 15

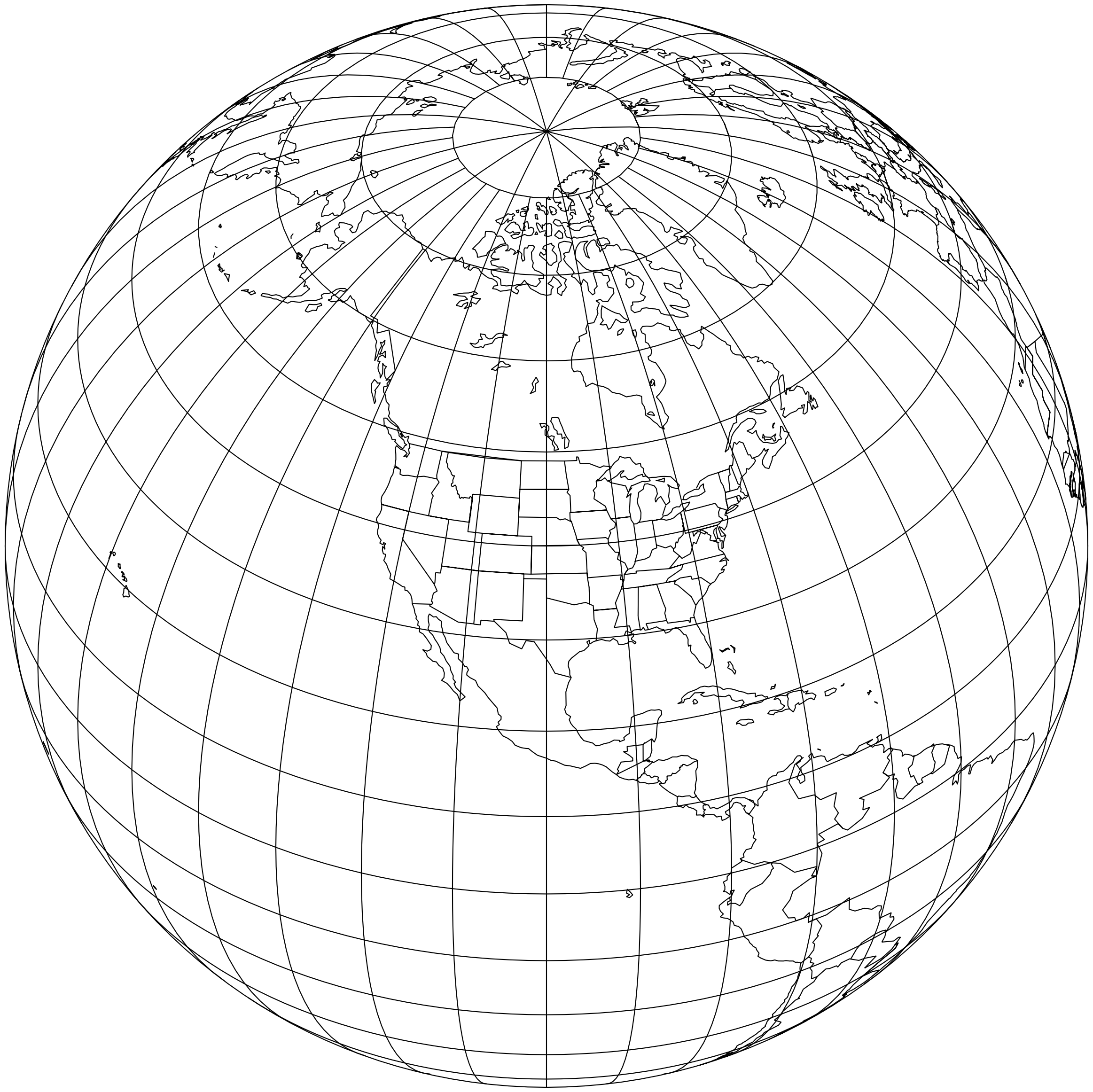
In this example, the *layout* action is used to specify a reference sphere in which the nominal angular separation between latitude and longitude grid lines is 10°. However, to avoid a confluence of longitude lines at the poles, the separation between longitude lines is increased to 30° for latitudes beyond ±80°. See the Reference Section for a more complete discussion of the possible *layout* options.

The viewpoint from which the sphere is observed is chosen to be the point at azimuth, -100°, elevation, 40°, and infinitely far from the center of the sphere (default). The subsequent *plot* action causes the reference sphere to be drawn on the page.

The *input* action is then used to overlay a political map of the Earth’s surface on the sphere. The file, "sphplt.map", contains the user input statements necessary to add this map to any previously defined reference sphere. See the Reference Section for a more complete description of the *input* action.

Example 15

Western Hemisphere
Viewpoint is 100° W longitude, 40° N latitude
as seen from "infinity"



```
layout solid intvl=10 pintvl=30 plat=80 radius=3
       azo=−100 elo=40;

plot title="Western Hemisphere",
"Viewpoint is 100\(. W longitude, 40\(. N latitude",
"as seen from \"infinity\"";

input file="sphplt.map";
```

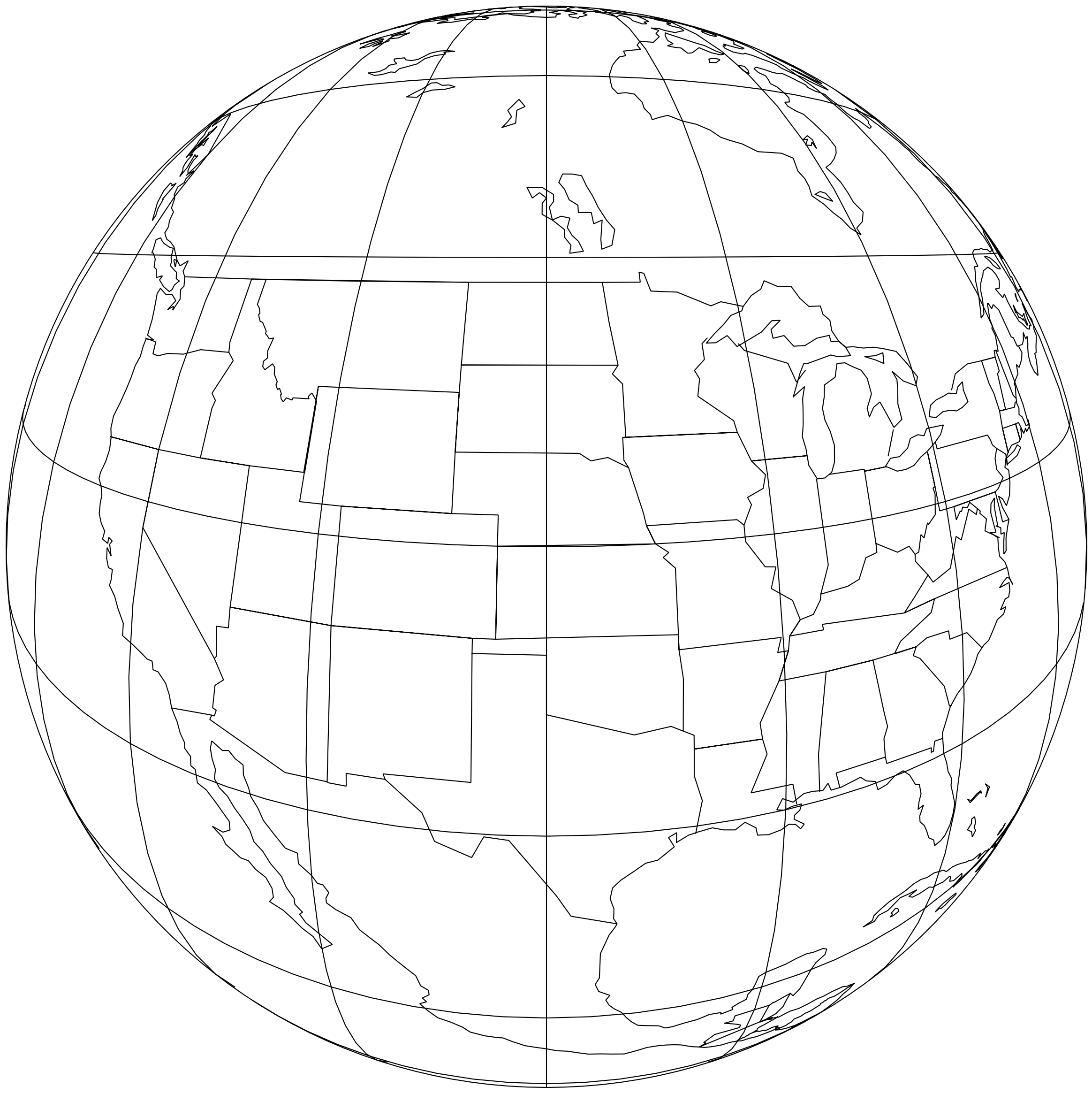
5.16 Example 16

In this example, although the layout is the same, the *plot* parameter, *dist*, is used to specify a viewpoint which is at a distance from the center of the sphere equal to 1.2 times the radius of the reference sphere. For the Earth, this would correspond to an altitude of approximately 1276 km. Otherwise, the plot is identical to the previous example.

In general, the reference sphere may be drawn in perspective from any point outside the sphere.

Example 16

Continental United States
Viewpoint is 100° W longitude, 40° N latitude
Distance from the center is 1.2 Earth radius



```
plot dist=1.2 title="Continental United States",
"Viewpoint is 100\(. W longitude, 40\(. N latitude",
"Distance from the center is 1.2 Earth radius";

input file="sphplt.map";
```

5.17 Example 17

Example 17 plots the results of a simulation of the precession of a spinning spacecraft using axial thrusters. The data is contained in a C binary file, "reor.dat", comprising the time (in seconds from start) and the corresponding right ascension and declination angles defining the direction of the spacecraft angular momentum vector.

In the initial actions, the data is read in and the plot *layout* parameters and flags are restored to their default values. Additionally, the simulation time values are converted to minutes using the *calc* action.

The *select* action specifies that azimuth values are to be taken from the second position in each record in the data file, while elevation angles defining the curve are obtained from the third plot vector. Since the radial distance parameter, *d*, is unspecified, the resultant curve is drawn on the surface of the reference sphere by default.

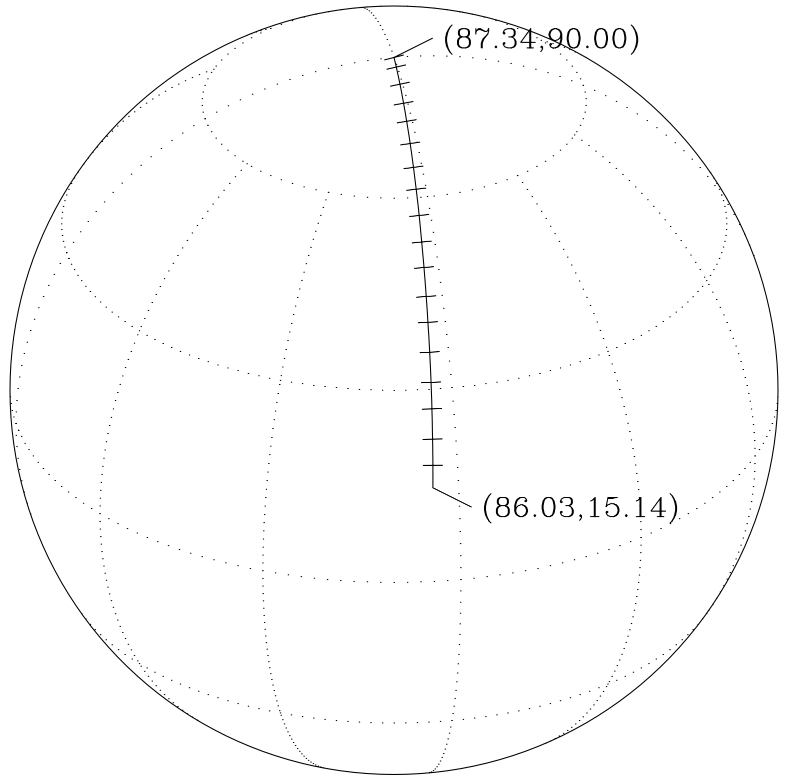
The *select* action also specifies that the initial and final points of the curve be annotated and that tick marks be placed at 5 minute intervals along the curve.

The *plot* action causes the selected curve to be plotted on the specified reference sphere with an appropriate title.

The final action indicates the use of the *put* flag, *inches*, in the spherical map. If this flag is specified, then position attributes are interpreted in terms of the horizontal (x) and vertical (y) coordinates (in inches) relative to the center of the reference sphere. In this example, the lower left corner of the first character of the string is placed at the same x-coordinate as the center of the sphere and 2.5 inches below it.

Example 17

Motion of the spacecraft angular momentum vector
during 75° tip maneuver



time ticks are at 5 minute intervals

```
read c double file="reor.dat" nvar=3;
layout dotted intvl=30 pintvl=90 plat=60 radius=2
       azo=80 elo=30;

calc rcl[1] 60 / sto[1];                        % convert time to minutes

select az=2 el=3 note(ip fp) ticks(var=1 delta=5);

plot title="Motion of the spacecraft angular momentum vector",
"during 75\(. tip maneuver";

put str(pos=0.0,-2.5)="time ticks are at 5 minute intervals" inches;
```

5.18 Example 18

This example illustrates the capability to draw curves which extend above the surface of the sphere.

It should be noted that the program does not scale the size of the reference sphere according to the data. It is the user’s responsibility to insure that the specified radius of the reference sphere is appropriate for the desired size of the resultant plot.

For this example, the data are contained in the file, "orbits.dat". Each record contains the true anomaly, azimuth, elevation and radial distance of each of four spacecraft.

First, the data is read in and the scale factor is set to 0.8 to reduce the plot to a more convenient size. Then, four curves are selected by assigning the appropriate column numbers (position in the plot record) to the parameters, *az*, *el* and *d*, and a legend string is assigned for each curve.

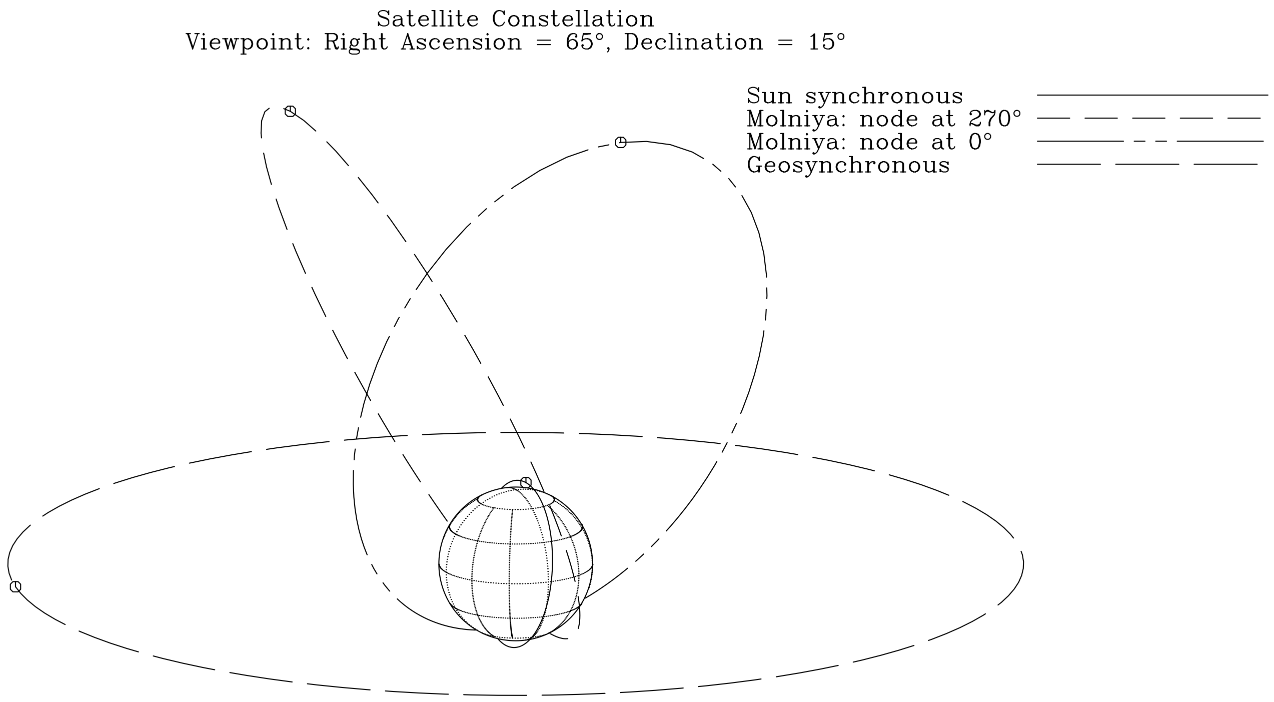
In the *plot* action which follows, the radius of the reference sphere is set equal to 0.5 inches. Since it is known that the radius of a geosynchronous orbit is approximately 6.61 Earth radii, this choice will result in a plot which is appropriately sized for the page.

The attributes of the *plot* flag, *legend*, are used to place the legend block at a convenient point away from the reference sphere. Note that the x and y coordinates of the upper-left corner of the legend block are described in inches relative to the center of the sphere.

Additionally, the attributes of the *plot* parameter, *title*, are used to increase the vertical space between the title and the center of the sphere.

Finally, the first point (only) of each curve is plotted on the same reference sphere using a specified special symbol to indicate the initial position of each spacecraft.

Example 18



```
read c single file="orbits.dat" nvar=16;
layout factor=0.8;

select az=2 el=3 d=4 legend="Sun synchronous";
select az=6 el=7 d=8 legend="Molniya: node at 270\(.";
select az=10 el=11 d=12 legend="Molniya: node at 0\(.";
select az=14 el=15 d=16 legend="Geosynchronous";

plot azo=65 elo=15 radius=0.5
legend(x=1.5 y=3.1)
title(y=3.5)="Satellite Constellation",
"Viewpoint: Right Ascension = 65\(., Declination = 15\(.";

          % put symbols at apogee

select az= 2 el= 3 d= 4 last=1 noline sym(type=0);
select az= 6 el= 7 d= 8 last=1 noline sym(type=0);
select az=10 el=11 d=12 last=1 noline sym(type=0);
select az=14 el=15 d=16 last=1 noline sym(type=0);
plot same;
```

5.19 Example 19

This example illustrates the transformation of input data points from rectangular Cartesian to spherical coordinates.

The file, "sts.dat", contains successive time values and x, y and z inertial position coordinates (km) corresponding to a typical Space Shuttle orbit. In the *read* action, three extra plot vectors are allocated to store the corresponding azimuth, elevation and radial distance values which are computed from these position coordinates.

In the first *calc* action, for convenience, the time values are converted from seconds to minutes from start. In the subsequent *calc* action, the following relations are used to determine the azimuth, elevation and radial distance from the rectangular coordinate values.

$$az = RTD \times Tan_2^{-1}(y, x)$$

$$el = RTD \times Sin^{-1}(z/r)$$

where

$$r = \sqrt{x^2 + y^2 + z^2}$$

and

$$d = r/RADIUS$$

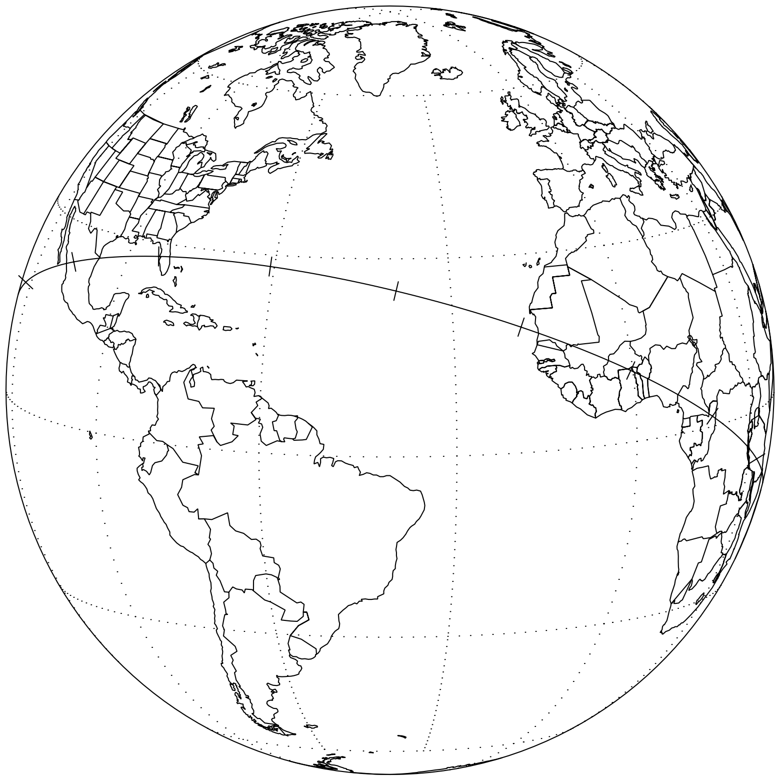
where *RADIUS* is the radius of the reference sphere in the same units as the position coordinates. For this plot, the appropriate value is the equatorial radius of the Earth, 6378.140 km.

In the *select* action, the radial distance parameter, *d*, is unspecified so that the orbital path is drawn on the surface of the reference sphere. Additionally, tick marks are specified at five-minute intervals along the curve.

Finally, the plot is drawn from the perspective of a viewpoint over the Atlantic Ocean and appropriate annotation and the world geopolitical map is added for reference.

Example 19

Space Shuttle Ground Track



time ticks are at 5 minute intervals

```
read c single file="sts.dat" nvar=4 nextra=3;
layout factor=1.0;

calc rcl[1] 60 / sto[1];           % convert time to minutes

calc rcl[3] rcl[2] atan2 rtd * sto[5]               % azimuth
rcl[2] xsq rcl[3] xsq + rcl[4] xsq + sqrt sto[7]  % radial distance (km)
rcl[4] rcl[7] / asin rtd * sto[6]                  % elevation
rcl[7] 6378.14 / sto[7];                           % radial distance (radii)

select az=5 el=6 ticks(var=1 delta=5);

plot azo=−40 elo=10
title="Space Shuttle Ground Track";

put str(pos=0.0,−2.5)="time ticks are at 5 minute intervals" inches;

input file="sphplt.map";
```

5.20 Example 20

This example illustrates some of the *put* options in the spherical map. First, a reference sphere is drawn. This is required for all *put* actions which refer to the sphere.

The *put* flag, *box*, produces, by default, a quadrilateral whose sides are formed by segments of great circles. The minimum angular separation of the vertical sides of the box is determined by the parameter attribute, *wid*, while *ht* gives the minimum angular separation of the horizontal sides. The resultant figure circumscribes, for example, the field-of-view of a scanning instrument mounted on a double gimbal mechanism with elevation as the inner gimbal degree-of-freedom and azimuth as the outer.

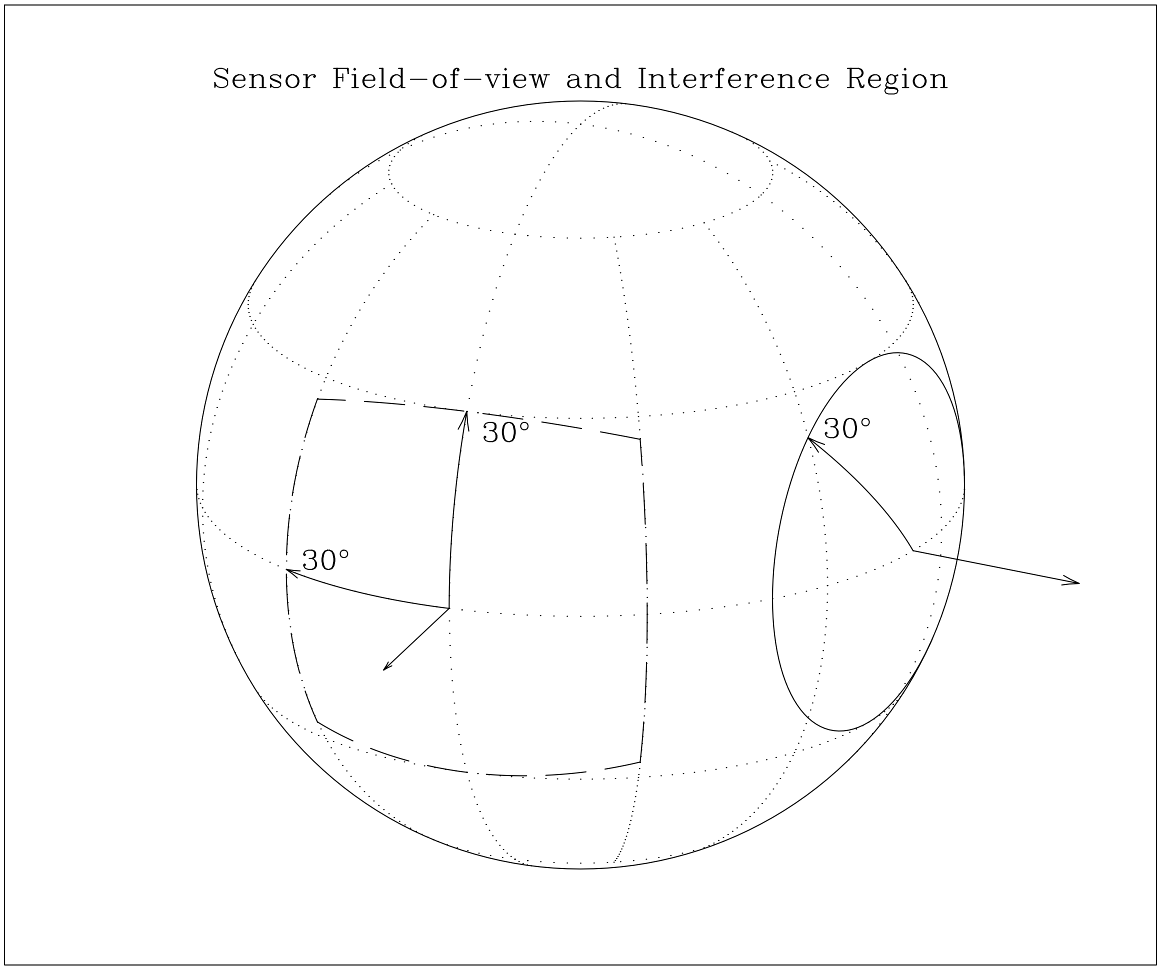
The center of the box is determined by the vector parameter attribute, *pos*. The input is of the form, *pos* = **az**, **el** [, **d**], where **az** and **el** are the azimuth and elevation values in decimal degrees and **d** is the radial distance. If **d** is not specified, the box is drawn on the surface of the sphere (**d** = 1).

The *put* flag, *arrow*, may be used to draw an arrow between two points relative to the sphere or, if the *put* flag, *inches*, is specified, any two points on the page. The flag attributes, *arc* (default) and *ray*, determine whether the arrow shaft is a segment of a great circle or a straight line. Note that if a great circle (*arc*) is selected, the radial distance values of the *from* and *to* attributes must be the same.

If the *put* flag, *deg*, is specified (default), the *put* flag, *circle*, causes a small circle to be drawn on the reference sphere. The position of the center of the circle is specified by the vector parameter attribute, *pos*, and the angular radius is determined by *rad*. The result is drawn in perspective relative to the selected viewpoint.

If the *put* flag, *inches*, is specified, then the result of *put* actions is essentially the same as in the Cartesian map. Moreover, in this case, the position attributes (*pos*, *from*, *to*) are interpreted in terms of horizontal (x) and vertical (y) coordinates in inches relative to the center of the sphere and the input format is of the form, *pos* = **x**, **y**. See the Reference Section for a more complete discussion of *put* options in the spherical map.

Example 20



```
plot azo=20 elo=20
title="Sensor Field−of−View and Interference Region";

put box(pos=0,0 wid=60 ht=60 type=1);

put arrow(from=0,0,0 to=0,0,1.5 ray);
put arrow(from=0,0,1 to=−30,0,1 arc);

put str=" 30\(.";
put arrow(from=0,0 to=0,30);
put str=" \(−\(−\(−30\(.";

put circle(pos=80,0,1 rad=30);

put arrow(from=80,0,0 to=80,0,1.5 ray);
put arrow(from=80,0,1 to=60,22.8381,1);
put str=" 30\(.";

put box(pos=−3,2.5 wid=6 ht=5) inches;
```

5.21 Character Sets

Data for four of the fonts used in McPlot, specifically, the Roman, Italic, Sans-serif and Script, as well as the Compound Greek/mathematics font, were obtained from a distribution of the Hershey fonts by the Usenet Font Consortium.

The format of the font data in this distribution, which differs from the format distributed by the U.S. NTIS, was originally created by

James Hurt
Cognition, Inc.
900 Technology Park Drive
Billerica, MA 01821

The Hershey fonts are digitized fonts originally created by Dr. A. V. Hershey at the U. S. National Bureau of Standards. They are suitable for typographic quality output on any vector device, when used at an appropriate scale. A complete listing of the fonts, which are in the public domain, may be obtained by ordering NBS Special Publication 424, "A contribution to computer typesetting techniques: Tables of Coordinates for Hershey's Repertory of Occidental Type Fonts and Graphic Symbols".

For use in McPlot, minor modifications were made to certain non-alphanumeric characters, the character set was re-ordered and the format of the data was restructured so that it differs from both the NTIS and Usenet Font Consortium distributions.

The next two pages provide samples of the five McPlot fonts and an example of the ASCII character set, additional mathematical symbols, Greek letters, special centered symbols and line types currently available in McPlot.

Sample McPlot Fonts

This is McPlot font number 0 (Simple, equal spacing).

Each character is formed on a basic 5×7 matrix.
This font requires the fewest number of strokes and
provides the greatest clarity on low resolution devices.

This is McPlot font number 1 (Roman, proportional spacing).

A more complex font, it requires approximately three times as
many strokes as the simple pen plotter–type font, but the
result is somewhat more pleasing to the eye.

This is McPlot font number 2 (Italic, proportional spacing).

*Proportional spacing provides a better appearance; however,
one is never sure where a sentence will end up!*

This is McPlot font number 3 (Sans–serif, proportional spacing).

Assertive, but not pushy, this font allows a clear, unpretentious
expression of ideas.

This is McPlot font number 4 (Script, proportional spacing).

"Dear, dear John, I'm sorry to have to be the one to tell you this...."

Example of the Simple Greek/mathematics font associated
with font number 0:

Εκτιμω την αληθεια. $E_0 = mc^2$. Μιλατε ελληνικα, βλεπω.

Example of the Compound Greek/mathematics font associated
with fonts 1 – 4:

Εκτιμω την αληϑεια. $E_0 = mc^2$. Μιλατε ελληνικα, βλεπω.

McPlot ASCII character set (Font 1):

!"#$%&'()*+,–./0123456789:;<=>?@

ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_‘

abcdefghijklmnopqrstuvwxyz{|}~⌧

Additional mathematical symbols:

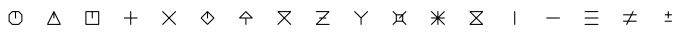
$\neq$ ∞ $\surd$ $\times$ $\pm$ ∂ $\int$ $\equiv$ ∇ $\leq$ $\approx$ $\geq$ $^\circ$ 0123456789

Greek alphabet:

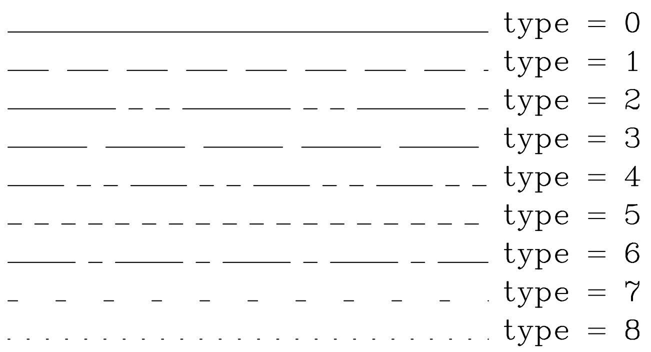
$\alpha\beta\gamma\delta\varepsilon\zeta\eta\vartheta\iota\kappa\lambda\mu\nu\xi o\pi\rho\sigma\tau\upsilon\varphi\chi\psi\omega$

ΑΒΓΔΕΖΗΘΙΚΛΜΝΞΟΠΡΣΤΥΦΧΨΩ

Special centered symbols (0 – 17):



Line types:



6. CUSTOMIZING THE PROGRAM

Proper selection of default values for input parameters allows the user to generate relatively complex plots with a minimum of input. McPlot allows the user to specify nearly all of the program defaults through the environment.

To utilize this feature, the user may edit his *.login* or *.cshrc* file and add the appropriate *setenv* commands. For example, the following C-Shell commands will make the default data file name, "mydata", the default record length, 12, and cause the date/time of the plot to be drawn automatically:

```
setenv JPFILE mydata
setenv JPRECL 12
setenv JPTIME yes
```

In the Bourne Shell, the same result is produced by the sequence of commands,

```
JPFILE=mydata
JPRECL=12
JPTIME=yes
export JPFILE JPRECL JPTIME
```

In order to avoid confusion, all of the *environment* variables which McPlot recognizes begin with the letters, JP, and must be written using upper case characters only. The values assigned to them must be given in lower case.

Table I gives the names of the environment variables which are accepted by McPlot, their default values and a brief description. Permitted alternatives are shown wherever appropriate.

Table I. McPlot Environment Variables

Name	Default	Description
JPFILE	data	Default plot data file name.
JPRECL	2	Default number of variables in a plot data record.
JPTYPE	ascii	Default type of input plot data file (= ascii | c | gnufor | sunfor | ftn | matlab | acsl).
JPPREC	double	Default precision of variables in the plot data file (= single | double).
JPXVAR	1	Default position of the independent (x) variable in a plot record (Cartesian map).
JPTIME	no	Put the date/time on each plot page by default (= yes | no)?
JPTIMX	0.35	Default x-coordinate of the start of the date/time annotation in inches relative to the lower left-hand corner of the page.
JPTIMY	0.5	Default y-coordinate of the start of the date/time annotation in inches relative to the lower left-hand corner of the page.
JPTSIZ	0.1	Default character size of the date/time annotation (in).
JPTANG	90.0	Default angle of the date/time annotation in degrees relative to horizontal. Positive angles result in counter-clockwise rotations, negative angles yield clockwise rotations.

Table I. (Continued)

Name	**Default**	**Description**
JPTFNT	0	Default font number for date/time annotation (0 - 4).
JPGRID	no	Draw a grid on a (Cartesian) plot by default (= yes | no)?
JPGDOT	yes	Use dots (vs. dashed lines) to form grid lines (= yes | no)?
JPGMIN	no	Grid minor axis subdivisions as well as major divisions (= yes | no)?
JPGTYP	0	Default line type to use for gridding (0 - 8).
JPSINT	1	Nominal interval between special (centered) symbols in terms of points along the curve.
JPNDEC	2	Nominal precision (number of decimal places) in numerical annotations.
JPYSTK	1	Nominal number of plots stacked vertically on a page.
JPXSTK	1	Nominal number of vertical columns of plots on a page. (The default for the total number of plots on a page is the product of JPYSTK and JPXSTK.)
JPCHHT	0.1	Nominal character size (in) used for labels, strings, etc.
JPNUHT	0.1	Nominal character size (in) used for numerical annotations.
JPSSHT	0.1	Nominal special (centered) symbol size (in).
JPFONT	0	Nominal character font number (0 - 4).
JPFACT	1.0	Default plot scale factor.

Table I. (Continued)

Name	Default	Description
JPYDEL	1.0	Nominal vertical space between plots (in).
JPXDEL	1.5	Nominal horizontal space between plots (in).
JPXORG	1.5	Default x-coordinate of the origin of the first plot on a page in inches relative to the lower left-hand corner (Cartesian map).
JPYORG	1.0	Default y-coordinate of the origin of the first plot on a page in inches relative to the lower left-hand corner (Cartesian map).
JPXLEN	6	Nominal Cartesian x-axis length (in).
JPYLEN	4	Nominal Cartesian y-axis length (in).
JPAZDF	1	Default position of the azimuth variable in a plot record (spherical map).
JPXCEN	3.0	Default horizontal (x) coordinate of the center of the first reference sphere on a page (spherical map) in inches relative to the lower left-hand corner.
JPYCEN	3.0	Default vertical (y) coordinate of the center of the first reference sphere on a page (spherical map) in inches relative to the lower left-hand corner
JPSRAD	2.0	Default reference sphere radius in inches (spherical map).

Table I. (Continued)

Name	**Default**	**Description**
JPAZOB	0.0	Default azimuth (deg) from which the reference sphere is viewed (spherical map).
JPELOB	0.0	Default elevation (deg) from which the reference sphere is viewed (spherical map).
JPDIST	1.0e10	Default distance (in terms of the reference sphere radius) from which the sphere is viewed (spherical map).
JPINTV	30	Default interval between latitude and longitude lines on the reference sphere (deg). Note: JPINTV must be in the range, $0 < \text{JPINTV} \leq 180$, and divide evenly into 180.
JPITVP	90	Default interval between longitude lines near the pole of the reference sphere (deg). Note: JPITVP must be a multiple of JPINTV ($0 < \text{JPITVP} \leq 180$) and divide evenly into 180.
JPLATP	60	Default latitude (deg) at which the interval between longitude lines on the reference sphere becomes JPITVP. Note: JPLATP must be in the range $0 \leq \text{JPLATP} \leq 90$, a multiple of JPINTV and divide evenly into 180.
JPDOTG	yes	Draw longitude and latitude lines on the reference sphere grid using unconnected dots (= yes | no)? If *no* is selected, solid lines are used.
JPSPTS	201	Default number of points in cubic spline interpolation.
JPORPG	portr	Default orientation of plots on a page (= portr | lands).

7. REFERENCE

This section contains a description of the possible McPlot actions. A complete list of actions may be obtained by entering the *help* command at the top level prompt, >. The resultant *help* screen is shown below.

```
      McPlot: Version 4.5

      Copyright (C) 1989, 1991 - 1993 by James McEnnan

     McPlot is free software and may be redistributed
     under the terms and conditions of the GNU General Public License.
     McPlot comes with ABSOLUTELY NO WARRANTY.
     For complete terms and conditions, see the GNU General Public License
     included with this distribution.

> help

McPlot commands are:

      quit
      help
      save
      !system command

Possible actions include:

      input
      read
      calc
      map
      layout
      select
      plot
      put
```

This list also indicates the normal sequence of actions which are employed to generate a plot. There are no restrictions, however, so that actions in McPlot may be entered in any order, and may be repeated as many times as desired.

7.1 INPUT

The *input* action allows plot description elements to be inserted into the input stream from a named file. The help screen for this action is shown below.

```
input> help

For the action, input, current parameters (attributes) are:

                file

and current flags (attributes) are:

                NONE
```

7.1.1 Parameters

The *input* parameter, *file*, determines the name of the file from which user input will be read. This file may contain any valid McPlot commands or actions, including other *input* actions. Thus, *input* actions may be nested. At present, the maximum depth of *input* levels is five. If more than 5 *input* statements are nested, a warning is directed to *stderr* and user input is continued from the current input stream.

Upon reaching the *end-of-file*, control reverts to the input stream from which the *input* action was invoked.

7.1.2 Scope

The parameter, *file*, does not retain its value after the *input* action is completed. If no file name is specified, no action is performed.

7.2 READ

The *read* action is used to read into memory the matrix of data to be plotted. The help screen for the *read* action is shown below.

```
read> help

For the action, read, current parameters (attributes) are:

            file        nvar        nextra

and current flags (attributes) are:

            ascii       c
            gnufor      sunfor
            decfor      matlab( name )
            acsl        binary
            single      double
```

7.2.1 Parameters

The *read* parameter, *file*, gives the name of the plot data file to be read in; its default value is "data" in the user's current directory.

The integer parameter, *nvar*, is the number of variables (plot vectors or columns of the data matrix) in a record. Its default value is *nvar* = 2. As noted previously, *nvar* must be specified, if different from the default value, for all ASCII, C or Fortran plot data files since there is, in general, no way to determine the record length in these cases from the file itself. For Matlab or ACSL files, the length of a record is provided in a header record and the parameter, *nvar*, is ignored.

If additional space is desired to store the results of *calc* actions in memory, then the parameter *nextra* must be set equal to the number of extra variables desired. The default value for *nextra* is 0.

7.2.2 Flags

The *read* flags are used to select the appropriate plot data input format. The default is *double* precision ASCII. Other possible choices are: *c* (binary), *gnufor*, *sunfor*, *decfor*, *matlab* or *acsl*, and *double* precision (referring to C or Fortran binary). The flag, *binary* is a synonym for C binary.

In the case of Matlab files, McPlot will read the first matrix contained in the file by default. If there is more than one matrix in the Matlab file, the user may specify the name of the matrix to be read in by using the string parameter attribute, *name*, of the flag, *matlab*.

7.2.3 Scope

Only one file may be stored in memory at any time. However, there are no limits on the number of different files which may be accessed, so that the user may exercise the *read* action as often as required.

Read flags and parameters (except for *name*) retain their values from one invocation of the *read* action to the next. Thus, to access data from any number of files having the same format, only the file name must be specified (if different) after the first read action.

7.3 CALC

The *calc* action is used to transform data before plotting. Plot vectors (variables) may be added to or subtracted from, multiplied or divided by constants or other plot variables; in addition, trigonometric, and inverse trigonometric, exponential and logarithmic functions are available for more general transformations. Thus, the *calc* action provides essentially all features found in a full-function scientific calculator.

The operating logic of the *calc* action is very similar to that of the Hewlett-Packard line of calculators, with certain differences noted below. It is based on a mathematical logic known as "Polish Notation," developed by the noted Polish logician Jan Łukasiewicz (1876-1956). While conventional algebraic notation places the operators between the operands, Łukasiewicz's notation specified the operators before the operands. The current convention is to reverse the order so that the operators are specified after the operands; hence the term "Reverse Polish Notation" (RPN).

RPN permits complicated calculations in a straightforward manner, without parentheses or punctuation, by automatically retaining and returning intermediate results. The system is implemented by means of an internal memory stack with four registers, X, Y, Z and T. In McPlot, these registers are actually one-dimensional arrays which have been allocated enough space to hold an entire plot vector. Any number that is encountered in the input stream or results from the execution of a numeric function is placed in the X-register on an element by element basis. This placement may cause vectors already in the stack to lift, remain in the same register or drop, depending on the preceding and current operations. Vectors on the stack are stored on a last-in, first-out basis.

Note that, unlike the HP calculator, there is no separate ENTER operation. Thus, the input stream

```
calc> 1 2 3 4
```

will cause the T-register to be filled with a plot vector consisting of 1.0's (the number of elements equals the number of records read in), the Z-register with 2.0's, the Y-register with 3.0's and the X-register with 4.0's. If the above input is followed by

```
calc> /
```

then as a result of this operation, the X-register contains the quotient of the Y-register divided by the X-register, a vector consisting of 0.75's, the Y-register contains 2.0's, the Z-register contains 1.0's and the T-register contains 4.0's, which were the previous contents of the X-register. Note that, unlike HP calculator logic, the stack in McPlot is not popped after an operation; instead, the stack is rolled down. The fact that the stack is cyclic in McPlot is the major difference between the *calc* operation and the usual HP logic. In practice, however, this should not cause any apparent differences in operation.

7.3.1 Constants

At present, there are three predefined constants in the *calc* action: *pi* (= 3.1415...), *rtd* (= $180/\pi$), the conversion from radians to degrees, and *dtr* (= $\pi/180$), the conversion from degrees to radians. These may be used in the same manner as explicit constants (numbers) entered by the user. For example, the input

```
calc> rcl[1] rtd *
```

will recall the contents of the first plot vector and multiply each element by the conversion from radians to degrees.

7.3.2 Stack Operations

There are three operations for manipulating the stack, *rup* (roll up), *rdn* (roll down) and *exy* (exchange X- and Y-registers). These may be used to position vectors for subsequent operations. In the *calc* action, only the X- and Y-registers are utilized by the current set of unary and binary operations.

7.3.3 Changing Data

Operations on the stack are completely independent of the plot data stored in memory. The only way plot data can be changed is through the operations, *rcl* (recall) and *sto* (store). The input stream,

```
calc> rcl[4] 3600 / sto[4]
```

has the following effect. The fourth column of input plot data is copied into the X-register of the stack. Then, a vector consisting of 3600.0's is put into the X-register and the original contents of the X-register are pushed onto the Y-register. Next, the Y-register is divided by the X-register on an element-by-element basis and the result is placed in the X-register. Finally, the contents of the X-register are copied into the memory locations previously occupied by the fourth column of input data. The result is that every element of the fourth column of input data is divided by 3600.0. Note that, in this case, the original data values are changed as a result of the *sto* operation. If the *sto* operation is omitted, the original data is unchanged. Note also that, after a *sto* operation, the stack is rolled down so that the sequence

```
calc> sto[1] sto[2]
```

will store the contents of the X-register in plot vector 1 and the contents of the Y-register in plot vector 2. This differs from the usual HP *sto* operation, which does not affect the stack.

If extra space has been allocated in the data block by the *read* action (*nextra* > 0), then the results of calculations may be stored in these locations by the same mechanism. Thus, the input sequence

```
read file="data" nvar=4 nextra=1;
calc rcl[1] 3600 / sto[5];
```

will result in five plot vectors; the original four unchanged and a fifth which is obtained from the first by dividing each element by 3600.0. This fifth plot vector may subsequently be used for plotting on the same basis as the other four plot vectors which were originally read in.

The current help screen for the *calc* action is shown below and a summary description of the available operators is given in Table II.

```
calc> help

For the action, calc, associated operators are:

            rcl       sto       rup       rdn       exy
            +         –         *         /         sqrt
            indx      abs       chs       inv       xsq
            sin       cos       tan       log       log10
            asin      acos      atan      exp       ypowx
            atan2     xymax     xymin     upper     lower
            ymodx     floor     ceil      sum       mean

and available constants are:

            pi        rtd       dtr
```

Table II. McPlot *calc* Operators

operator	**description**
rcl	Syntax: rcl [*variable_number*] -- copies the plot vector specified by *variable_number* into the X-register.
sto	Syntax: sto [*variable_number*] -- copies the contents of the X-register into the plot vector specified by *variable_number*.
rup	Rolls the stack up so that X -> Y -> Z -> T -> X.
rdn	Rolls the stack down so that T -> Z -> Y -> X -> T.
exy	Exchanges the contents of the X- and Y-registers so that X <--> Y.
+	Adds the contents of the X-register to the contents of the Y-register on an element-by-element basis. The stack is then rolled down so that the result is placed in the X-register. The original contents of the Y-register are lost.
–	Subtracts the contents of the X-register from the contents of the Y-register on an element-by-element basis. The stack is then rolled down so that the result is placed in the X-register. The original contents of the Y-register are lost.
*	Multiplies the contents of the Y-register by the contents of the X-register on an element-by-element basis. The stack is then rolled down so that the result is placed in the X-register. The original contents of the Y-register are lost.
/	Divides the contents of the Y-register by the contents of the X-register on an element-by-element basis. The stack is then rolled down so that the result is placed in the X-register. The original contents of the Y-register are lost. If an element of the X-register is zero, the result is set to zero and an error message is written to *stderr*.

Table II. (Continued)

operator	**description**
sqrt	Replaces each element of the X-register by its square root. If an element of the X-register is negative, the result is set to zero and an error message is written to *stderr*.
indx	Rolls the stack up and replaces the contents of the X-register with the sequence: { 0.0, 1.0, 2.0, 3.0, ..., (*number_of_records* – 1) }
abs	Replaces each element of the X-register by its absolute value.
chs	Changes the sign of each element of the X-register.
inv	Replaces each element of the X-register by its inverse. If an element is zero, the result is set to zero and an error message is written to *stderr*.
xsq	Replaces each element of the X-register by its square.
sin	Replaces each element of the X-register (radians) by its sine.
cos	Replaces each element of the X-register (radians) by its cosine.
tan	Replaces each element of the X-register (radians) by its tangent.
log	Replaces each element of the X-register by its natural logarithm. If an element is non-positive, the result is set to zero and an error message is written to *stderr*.
log10	Replaces each element of the X-register by its common logarithm. If an element is non-positive, the result is set to zero and an error message is written to *stderr*.

Table II. (Continued)

operator	description
asin	Replaces each element of the X-register by the principal value of the inverse sine (radians). If the magnitude of an element is greater than one, it is normalized before the inverse sine is produced and an error message is written to *stderr*.
acos	Replaces each element of the X-register by the principal value of the inverse cosine (radians). If the magnitude of an element is greater than one, it is normalized before the inverse cosine is produced and an error message is written to *stderr*.
atan	Replaces each element of the X-register by the principal value of the inverse tangent (radians).
exp	Replaces each element of the X-register by its exponential.
ypowx	Replaces each element of the Y-register by the element raised to the corresponding X power. The Y-register is then rolled down so that the result is placed in the X-register. The original contents of the Y-register are lost. If both elements are zero, the result is set to zero and an error message is written to *stderr*.
atan2	Replaces each element of the Y-register by the appropriate value (radians) of the inverse tangent of (Y/X) depending on the quadrant and then rolls down the stack so that the result is placed in the X-register. The original contents of the Y-register are lost. If both elements are zero, the result is set to zero and an error message is written to *stderr*.

Table II. (Continued)

operator	**description**
xymax	Replaces each element of the Y-register by the larger of the corresponding elements in the X- and Y-registers. The stack is then rolled down so that the result is placed in the X-register. The original contents of the Y-register are lost.
xymin	Replaces each element of the Y-register by the smaller of the corresponding elements in the X- and Y-registers. The stack is then rolled down so that the result is placed in the X-register. The original contents of the Y-register are lost.
upper	Replaces each element of the X-register by the least upper bound of the elements in the X-register.
lower	Replaces each element of the X-register by the greatest lower bound of the elements in the X-register.
ymodx	Replaces each element of the Y-register by the remainder, f, with respect to division by the corresponding element of the X-register, such that $y = ix + f$ for some integer, i, $\lvert f \rvert < \lvert y \rvert$, and the sign of f is the same as the sign of y. If x is zero, y is unchanged. The Y-register is then rolled down so that the result is placed in the X-register. The original contents of the Y-register are lost.
floor	Replaces each element of the X-register by the largest integer not greater than X.
ceil	Replaces each element of the X-register by the smallest integer not less than X.
sum	Replaces each element of the X-register by the corresponding element of the cumulative sum, comprising the sum of the current element and the preceding element of the sum. For the first element, the preceding element is zero.
mean	Replaces each element of the X-register by the average value of the elements in the X-register.

7.4 MAP

The *map* action is used to specify the interpretation of data and the plot geometry. The help screen for the *map* action is shown below.

```
map> help

For the action, map, current parameters (attributes) are:

            NONE

and current flags (attributes) are:

            xy          sphere
```

7.4.1 Flags

The *map* flags are used to specify the interpretation of data and the plot geometry. The default is *xy*. In this mapping, hereafter referred to as the Cartesian map, input data points are interpreted as rectangular Cartesian coordinates of a plane curve and are plotted conventionally as *y* versus *x*.

Selection of the *map* flag, *sphere*, results in the interpretation of data elements in terms of spherical polar coordinates of a point in space. This will subsequently be referred to as the spherical map. The resultant plot is referenced to a unit sphere as seen from the perspective of a specified observer.

For each mapping, appropriate changes are made in the available *layout*, *select*, *plot* and *put* parameters and flags.

7.4.2 Scope

A selected mapping is retained until it is changed in a subsequent *map* action.

7.5 LAYOUT

The *layout* action is used to determine the default size, number and position of plots on a page, as well as the default font number, character size, etc. The particular parameters and flags which may be specified depend on the selected mapping.

7.5.1 Cartesian Map

The help screen for the *layout* action in the Cartesian map is shown below.

```
layout> help

For the action, layout, current parameters (attributes) are:

          factor    origin    axlens    stack     delta     ndec
          font      symht     numht     ssht      ssint

and current flags (attributes) are:

          portr     lands
          nogrid    grid( dot line major minor )
          notime    time( x y font symht ang )
```

7.5.1.1 Parameters

The parameter, *factor*, may be used to scale all plot dimensions uniformly, including the size of characters in annotations. All output drawing motions are effectively multiplied by the current value of *factor*. The scale factor may be any real number greater than zero; its default value is $factor = 1.0$. Although the scale factor may be set in the *layout* action, it does not take effect until the first *plot* action. In some cases, it may be desired to set the scale factor for a *put* action before plotting. In this case, the user input,

```
> plot nox noy;
```

without a preceding *select* action will effect the desired scale transformation immediately.

The next four parameters control the nominal size and position of plots on a page.

The vector parameter, *origin*, determines the x (horizontal) and y (vertical) position of the origin of the first plot (measured in inches) relative to the lower left-hand corner of the page. Note that the position specified by *origin* is absolute; it is not affected by the value of *factor*. The first component of *origin* specifies the x-coordinate and the second component gives the y-coordinate. To place the origin at (2.0, 3.0), for example, the input is

```
layout> origin = 2.0, 3.0
```

The comma between x- and y-values is required. The default value is *origin* = 1.5, 1.0.

The vector parameter, *axlens*, determines the default x- and y-axis lengths in inches. Only integer values are actually used. Fractional lengths are rounded to the nearest integer value. The first component of *axlens* specifies the x-axis length; the second component refers to the y-axis. The default value is *axlens* = 6, 4.

The next two parameters, *stack* and *delta*, are the only vector parameters in McPlot in which a vertical (y) direction is specified before a horizontal (x) direction. The parameter, *stack*, is an integer-valued vector which determines the number of plots stacked in a vertical column on a page by the number of columns. The default value is *stack* = 1, 1, or one plot per page. To obtain three plots on a page, one above the other, the input would be *stack* = 3 (note that it is not necessary to specify the number of columns in this case). To obtain three plots placed side by side on a page, the input would be *stack* = ,3 (note the comma). To obtain six plots on a page arranged in two columns of three plots each, the input is *stack* = 3, 2, etc. Plots are stacked from the bottom of the page up, and from the left-hand side towards the right.

The nominal spacing between plots (in inches) is specified by *delta*. The first component of *delta* determines the distance between the end of the y-axis of one plot and the start of the y-axis (origin) of the next vertical plot, while the second component of *delta* controls the distance between the end of the x-axis of one plot and the start of the x-axis (origin) of the next horizontal plot. The default value is *delta* = 1.0, 1.5. The actual distance between plots as displayed or printed is the input distance multiplied by the scale factor.

The parameter, *ndec*, sets the default precision (number of places after the decimal) used in annotating numerical values associated with a curve; e.g., the maximum, minimum or final value. Its default value is *ndec* = 2. The program will attempt to use an *f* format for these annotations; however, if the number is too small to be usefully represented with the specified value of *ndec* or if it is too large, an *e* format is used. If *ndec* is less than zero, the annotation is given in integer format with no decimal point.

The integer parameter, *font* (0 - 4), sets the default font number. Five different ASCII character sets are currently available: Simple, Roman, Italic, Sans-serif and Script. In addition, the Greek alphabet and special mathematical symbols are available in two forms: simple and compound. The compound form is automatically selected for the Roman, Italic, Sans-serif and Script fonts (1 - 4). The default is *font* = 0.

The parameter, *symht*, sets the default character size used in labels, legends and titles; annotations of maximum, minimum and final values; line tick annotations and *put* strings. The height from base to cap for upper case alphabetic characters is nominally *symht* inches. In proportionally spaced fonts, the width is character dependent. In the Simple font, except for descenders on lower case letters, each character occupies a square which is nominally *symht* inches on a side. The default value for *symht* is 0.1 inches.

The parameter, *numht*, sets the default size of axis tick marks and axis numerical annotations, as well as the character size used for *put num* and *data* values. Similarly, the nominal size of special centered symbols is determined by *ssht*. The default value for these parameters is also 0.1 inches.

The parameter, *ssint*, controls the default interval (in terms of points on the curve) between successive special centered symbols when these are selected to be drawn on the curve. The default value is *ssint* = 1, so that a centered symbol is drawn at every point.

7.5.1.2 Flags

The *layout* flags, *portr* and *lands*, control the nominal orientation of pages on devices which have the capability to produce output in either portrait or landscape mode; they have no effect otherwise. Portrait mode, the default, has the y-direction along the long axis of the page, while landscape mode sets the x-direction along the long axis of the page.

The flag, *grid*, causes a set of vertical and horizontal grid lines to be drawn on each plot automatically. The default is not to draw a grid (*nogrid*). If the axis scaling is linear, the corresponding grid spacing will be linear; if the axis is logarithmic, the grid will be logarithmic. The *help* screen for this flag is shown below.

```
layout.grid> help

For the action, layout, current parameters (attributes) are:

            NONE

and current flags (attributes) are:

            dot         line( type )
            major       minor
```

The *grid* flag attributes, *dot*, *line*, *major* and *minor*, determine the type and spacing of grid lines drawn, by default, on each plot.

The flag, *dot*, specifies that grid lines are composed of closely spaced dots and is the default. The dot spacing is currently 40 dpi for major axis divisions and 20 dpi for minor axis divisions. Alternatively, the grid may be made up of dashed lines by selection of the flag, *line*. In this case, the line type (0 - 8) may be specified by the parameter attribute, *type*. The default is a solid line (*type* = 0).

For linear axes, the flag, *major*, specifies that grid lines are drawn only at major axis divisions (one-inch intervals) and is the default. If the *grid* attribute, *minor*, is selected, grid lines are also drawn at minor axis divisions determined by the axis scaling. For logarithmic axes, grid lines are automatically drawn at minor axis divisions, determined by the axis scaling, and the flags, *major* and *minor*, have no effect.

The *layout* flag, *time*, if selected, causes the date and time of the plot to be written on each new page. The default is not to draw the date/time stamp (*notime*).

The attributes of the flag, *time*, allow the date/time annotation to be placed anywhere on the page at any angle and with any character size and font. The real parameters, *x* and *y*, give the x- and y-position of the start of the date/time string in inches relative to the lower left-hand corner of the page. The default position is $x = 0.35$, $y = 0.5$. The integer parameter attribute, *font*, determines the font number used for the date/time annotation, while the real parameter attribute, *symht*, sets the character size (in inches). The default font number is 0 and the character size is 0.1 inches. *ang* is the angle (in degrees) between the horizontal edge of the page and the date/time annotation. Positive values of *ang* produce a counter-clockwise rotation. The default value is 90.0 degrees.

Note: the date/time annotation is not affected by the plot scale factor.

7.5.2 Spherical Map

The help screen for the *layout* action in the spherical map is shown below.

```
layout> help

For the action, layout, current parameters (attributes) are:

          factor    center    radius    azo       elo       dist
          intvl     pintvl    plat      stack     delta     ndec
          font      symht     numht     ssht      ssint

and current flags (attributes) are:

          dotted( space )     solid
          portr     lands
          notime    time( x y font symht ang )
```

7.5.2.1 Parameters

The parameter, *factor*, may be used to scale all plot dimensions uniformly, including the size of characters in annotations. All output drawing motions are effectively multiplied by the current value of *factor*. The scale factor may be any real number greater than zero; its default value is $factor = 1.0$. Although the scale factor may be set in the *layout* action, it does not take effect until the first *plot* action. In the spherical map, there is no simple way to set the scale factor before the first plot.

The next eight parameters control the nominal size and appearance of the reference sphere used for plotting.

The parameter, *center*, is a vector which determines the x (horizontal) and y (vertical) position (in inches from the lower-left corner of the page) of the center of the first reference sphere drawn on a page. The default value is $center = 3.0, 3.0$. Note that the position specified by *center* is absolute; it is not affected by the value of *factor*.

The parameter, *radius*, specifies the radius (in inches) of the reference sphere on which the data is plotted. The default value is $radius = 2.0$.

The parameters, *azo* and *elo*, give the default azimuth and elevation angles (in degrees) of the point from which the reference sphere is observed. The default values are $azo = 0.0$ and $elo = 0.0$.

The parameter, *dist*, determines the apparent distance (in units of the sphere radius) from which the sphere is viewed. The default value is $dist = 1.0e+10$, which effectively places the viewpoint at infinity so that a complete hemisphere is shown. As the viewpoint is moved closer to the surface of the sphere, the perspective changes so that less of the surface is visible, with correspondingly more detail. Note, however, that values of $dist < 1.0$ are not allowed.

The integer parameter, *intvl*, determines the default spacing (in degrees) between longitude and latitude grid lines on the sphere. The default value is $intvl = 30$. Note that *intvl* must be a divisor of 180. The spacing determined by *intvl* is effective only between latitudes greater than $-plat$ and less than $+plat$. Outside this range, the spacing of longitude grid lines is determined by *pintvl* (see below) and no latitude grid lines are drawn.

In order to avoid a confluence of longitude grid lines at the poles, the user may specify a different spacing of longitude lines for latitudes greater than $+plat$ or less than $-plat$. The default spacing is given by the integer parameter, *pintvl*. The default for this parameter is $pintvl = 90$. Note that *pintvl* must also be a divisor of 180 and an integer multiple of *intvl*.

The integer parameter, *plat*, determines the latitude at which the spacing between longitude grid lines changes from *intvl* to *pintvl*. The default value is $plat = 60$. Note that *plat* must be in the range, $0 \leq plat \leq 90$, and must be an integer multiple of *intvl*.

The next two parameters, *stack* and *delta*, are the only vector parameters in McPlot in which a vertical (y) direction is specified before a horizontal (x) direction. The parameter, *stack*, is an integer-valued vector which determines the number of plots stacked in a vertical column on a page by the number of columns. The default value is *stack* = 1, 1, or one plot per page. To obtain three plots on a page, one above the other, the input would be *stack* = 3 (note that it is not necessary to specify the number of columns in this case). To obtain three plots placed side by side on a page, the input would be *stack* = ,3 (note the comma). To obtain six plots on a page arranged in two columns of three plots each, the input is *stack* = 3, 2, etc. Plots are stacked from the bottom of the page up, and from the left-hand side towards the right.

The nominal spacing between plots (in inches) is specified by *delta*. The first component of *delta* determines the vertical separation between the perimeter of one reference sphere and a subsequent sphere drawn on the same page, while the second component of *delta* controls the horizontal separation. The default value is *delta* = 1.0, 1.5. The actual distance between reference spheres as displayed or printed is the input distance multiplied by the scale factor.

The parameter, *ndec*, sets the default precision (number of places after the decimal) used in annotating numerical values associated with a curve; e.g., the initial or final coordinate values. Its default value is *ndec* = 2. The program will attempt to use an *f* format for these annotations; however, if the number is too small to be usefully represented with the specified value of *ndec* or if it is too large, an *e* format is used. If *ndec* is less than zero, the annotation is given in integer format, with no decimal point.

The integer parameter, *font* (0 - 4), sets the default font number. Five different ASCII character sets are currently available: Simple, Roman, Italic, Sans-serif and Script. In addition, the Greek alphabet and special mathematical symbols are available in two forms: simple and compound. The compound form is automatically selected for the Roman, Italic, Sans-serif and Script fonts (1 - 4). The default is *font* = 0.

The parameter, *symht*, sets the default character size used in labels, legends and titles; annotations of initial and final coordinate values; line tick annotations and *put* strings. The height from base to cap for upper case alphabetic characters is nominally *symht* inches. In proportionally spaced fonts, the width is character dependent. In the Simple font, except for descenders on lower case letters, each character occupies a square which is nominally *symht* inches on a side. The default value for *symht* is 0.1 inches.

The parameter, *numht*, sets the default character size used for *put num* and *data* values. Similarly, the nominal size of special centered symbols is determined by *ssht*. The default value for these parameters is also 0.1 inches.

The parameter, *ssint*, controls the default interval (in terms of points on the curve) between successive special centered symbols when these are selected to be drawn on the curve. The default value is *ssint* = 1, so that a centered symbol is drawn at every point.

7.5.2.2 Flags

The *layout* flags, *dotted* and *solid*, determine the characteristics of the longitude and latitude grid on the sphere. Selection of *dotted*, the default, results in a sphere grid made up of individual points placed at regular intervals along the lines of constant longitude and latitude. This grid style is convenient for terminal plots in which the small screen and low resolution limit the density of points which can be readily distinguished. The alternative, *solid*, results in conventional grid lines.

The integer parameter attribute, *space*, of the flag, *dotted*, determines the nominal spacing (in degrees) between points making up the latitude and longitude lines of the sphere grid. The range of *space* is from 0 to 3. *Space* = 0 results in a nominal spacing between points on major grid lines of ½ degree, while *space* = 3 gives a spacing of 3 degrees. The default value is *space* = 2.

The *layout* flags, *portr* and *lands*, control the nominal orientation of pages on devices which have the capability to produce output in either portrait or landscape mode; they have no effect otherwise. Portrait mode, the default, has the vertical (y) direction along the long axis of the page, while landscape mode sets the horizontal (x) direction along the long axis of the page.

The *layout* flag, *time*, if selected, causes the date and time of the plot to be written on each new page. The default is not to draw the date/time stamp (*notime*).

The attributes of the flag, *time*, allow the date/time annotation to be placed anywhere on the page at any angle and with any character size and font. The real parameters, *x* and *y*, give the x- and y-position of the start of the date/time string in inches from the lower left-hand corner of the page. The default position is $x = 0.35$, $y = 0.5$. The integer parameter attribute, *font*, determines the font number used for the date/time annotation, while the real parameter attribute, *symht*, sets the character size (in inches). The default font number is 0, and the character size is 0.1 inches. *ang* is the angle (in degrees) between the horizontal edge of the page and the date/time annotation. Positive values of *ang* produce a counter-clockwise rotation. The default value is 90.0 degrees.

Note: the date/time annotation is not affected by the plot scale factor.

7.5.3 Scope

The parameters, *factor*, *stack*, *delta*, *ndec*, *font*, *symht*, *numht*, *ssht* and *ssint*, as well as the flag, *time*, refer to the same variables in the spherical map as in the Cartesian map and may be specified in either *map* action.

All *layout* flags and parameters retain their values from one *layout* action to the next. Any parameter or flag which is not explicitly changed retains its previous value.

7.6 SELECT

The *select* action determines which variables will be used to define the curves for the current plot. Nominally, each curve requires a separate *select* action, so that the first invocation of *select* determines the first curve, the second invocation, the second curve, etc. Curves defined in consecutive *select* actions are scaled together so that they will all fit on the same plot. Currently, up to 25 curves may be selected at one time. It should be noted, however, that this limit only applies to the number of curves which may be considered simultaneously; following a *plot* action, additional curves may be selected and drawn on the same plot.

In general, all variables defining a curve must be specified in each *select* action. Thus, for the Cartesian map, both the independent variable, *x*, and dependent variable, *y*, must be specified; in the spherical map, the azimuth, *az*, elevation, *el*, and (optionally) the distance, *d*, are required. However, the program does provide default values for certain parameters; e.g., *x* in the Cartesian map and *az* in the spherical map, which are invoked if these parameters are not explicitly assigned values by the user. On the other hand, *y* in the Cartesian map and *el* in the spherical map remain undefined unless specified by the user. The program determines the number of curves to draw on the current plot by looking for the first undefined *y* or *el* value. Thus a *select* action in which these variables are not specified will result in a null plot.

Normally, an internal variable representing the current curve number is incremented each time the *select* action is invoked. After a *plot* action, this curve number is reset to 1. The user may also specify the curve number by following the *select* keyword with an integer denoting the number of the curve desired. For example, if the following series is input

```
> select y = 20;
> select y = 31;
> select y = 22;
```

and it is desired to change the dependent variable specified for the second curve, the subsequent input would have the form:

```
> select 2 y = 21;
```

Note that this mechanism resets the internal variable which contains the current curve number so that if it is desired to continue with the fourth curve in the above example, the next *select* should have the form:

```
> select 4 y = 17;
```

otherwise, the *y* parameter value for the *third* curve would be changed.

This construction is primarily intended to provide a mechanism for correcting mistakes in previous input and should be used with caution since curve plotting stops at the first undefined dependent variable, *y*, or elevation variable, *el*.

It is possible to shortcut curve selection for simple plots. The effect of multiple *select* actions is to increment an index into *select* parameter and flag arrays. The comma token has the same effect, locally, on the right-hand side of a parameter assignment, so that the result of the sequence:

```
> select az = 14 el = 27;
> select az = 15 el = 28;
> select az = 16 el = 29;
```

may also be obtained by the input

```
> select az = 14, 15, 16 el = 27, 28, 29;
```

Note, however, that the comma token may only appear on the right-hand side of parameter assignments. Hence, if a *select* action includes flags; e.g., *clip*, *note* or *ticks*, it is *necessary* to use multiple *select* actions to specify the curves.

7.6.1 Scope

The scope of *select* parameters and flags is the current *select* action only. Any parameter or flag which is not explicitly specified assumes its default value.

7.6.2 Data Compression

Normally, in the Cartesian map, the number of points in each selected curve is reduced before plotting by applying the Fan Data Compression Algorithm.* In this method, intermediate points which lie within a specified distance from a straight line connecting adjacent points are eliminated from the plot vector. The criterion for elimination is dependent on the resolution of the output device and is chosen automatically by the program. By construction, the resultant curve, as plotted, is indistinguishable from the original. The reduction in plotting time and size of output print files can be quite dramatic, particularly for dense plot data. However, certain options require omission of the data compression step. These will be indicated in the description of *select* parameters and flags which follows.

Note that data compression is not available for the spherical map.

* "Faster Plots by Fan Data Compression", R. A. Fowell and D. D. McNeil, IEEE Computer Graphics and Applications, March, 1989.

7.6.3 Cartesian Map

The main *help* screen for the *select* action in the Cartesian map is shown below.

```
select> help

For the action, select, current parameters (attributes) are:

        x         y         first     last      start     end
        skip      legend

and current flags (attributes) are:

        line( type )  noline
        nosym     sym( type intvl )
        noclip    clip( nox x noy y )
        nonote    note( nofv fv nomax max nomin min )
        noticks   ticks( var delta nonote note )
        noebars   ebars( var wid skip )
```

7.6.3.1 Parameters

For each curve, the independent variable is determined by assigning to the parameter, *x*, the index of the variable in the plot record (data column number). Similarly, the dependent variable is determined by assigning to the parameter, *y*, the appropriate column number. The default value for *x* is the last value assigned for the current plot or 1, while the default value for *y* is undefined. Curve plotting stops at the first undefined *y* parameter value, so that a null plot may be obtained by selecting no data and drawing no axes (see the discussion of the *plot* action).

In addition to the variable numbers, the user may specify, independently for each curve, the *first* and/or *last* record(s) to be plotted and/or the *start* and/or *end* value(s) of the independent variable. The default is to plot all of the data points in each curve. If the *first* and/or *last* record is specified, the axes will be scaled according to the data in the included records only. If a *start* and/or *end* value is input, the range of the independent variable will be less than or equal to the prescribed range, within the range specified by the *first* and/or *last* records, and the axes will be scaled accordingly. If the specified value of *start* is greater than the largest value of x, or *end* is less than the smallest value, a message will be printed on *stderr* and the entire curve (between the current *first* and *last* records) will be drawn.

If a portion of a curve is selected using the parameters, *first*, *last*, *start* or *end*, the maximum, minimum or final values used for annotation are relative to the selected segment. This allows the user to determine local extrema or the value of the dependent variable at a particular point.

The integer parameter, *skip*, determines the interval between points in the plot vector which are actually used to draw the curve. For example, if *skip* = 2, then only the first, third, fifth, ..., points in the input data are used to draw the curve. Note, however, that scaling of the axes, data smoothing and the maximum, minimum and final values used for annotation are determined from all of the points in the selected plot vectors and are not affected by the value of *skip*. The default value is $skip = 1$ so that every point is used. User input of $skip > 1$ results in the omission of data compression for that curve.

In general, unless there is a compelling reason (see, for instance, Example 5), use of the skip parameter should be avoided. The fan data compression algorithm, which is automatically invoked in McPlot, offers the simplest and most efficient means of reducing the number of points required to draw a particular curve.

Finally, a *legend* string may be associated with each curve for annotation. The position of the legend on the plot and other attributes may be specified in the *plot* action.

7.6.3.2 Flags

There are four possible representations of curves in McPlot. These are determined by the flags, *line*/*noline* and *sym*/*nosym*. The permutations are:

- *line* and *nosym* (the default). In this case, solid or dashed lines are drawn connecting the points on the curve and data compression is employed. By default, the line type starts at 0 and is automatically incremented for each subsequent curve selected. The line type is reset to zero when a new plot is initiated.
- *line* and *sym*. In this case, one of the special centered symbols is placed at points along the curve and a line is drawn connecting the points. Data compression is omitted. The interval (number of data points) between special symbols may be specified by the user. By default, a solid line (*type* = 0) is used to connect the points. The symbol type starts at 0 and is automatically incremented for each subsequent curve selected. The symbol type is reset to zero when a new plot is initiated.
- *noline* and *sym*. One of the special centered symbols is placed at points along the curve and data compression is omitted. The interval (number of data points) between special symbols may be specified by the user. By default, the symbol type starts at 0 and is automatically incremented for each subsequent curve selected. The symbol type is reset to zero when a new plot is initiated.
- *noline* and *nosym*. In this case, a dot is placed at each point on the curve and data compression is not applied.

The line type of each curve may be specified by means of the integer parameter attribute, *type*, of the flag, *line*. If the line type is not specified by the user, the default value is employed as described above. Similarly, the attributes, *type* and *intvl*, of the flag, *sym*, allow the specification of the symbol type and the interval (number of points) between symbols. If the symbol type is not specified, the default value is employed as described above. The default for *intvl* is the current value of the *plot* parameter, *ssint*, if specified;

otherwise, the value of the corresponding *layout* parameter, if specified; otherwise the program default value (*intvl* = 1) is used.

7.6.3.2.1 Clipping

The *select* flag, *clip*, offers a means to eliminate outliers in the data. The *help* screen for this flag is shown below.

```
select.clip> help

For the action, select, current parameters (attributes) are:

        NONE

and current flags (attributes) are:

        nox     x( max min )
        noy     y( max min )
```

If clipping is selected, then as the data is plotted the "drawing pen" is "raised" whenever the value of a coordinate of a point on the curve is outside the range specified by the maximum and/or minimum values. The default is not to clip (*noclip*). It is possible to clip either *x* or *y* or both; the default is neither (*nox* and *noy*). Note that data compression is not applied if clipping is invoked.

If *clip* is selected, scaling of the curve takes into account the maximum and minimum values specified, independently for each axis, by the attributes, *max* and *min*. If no value is given for either *max* or *min*, then all of the data is plotted without clipping for that coordinate, unless the axis scale and offset are prescribed by the user in the *plot* action. In that case, the curve will be clipped at the boundary of the plot.

For example, to clip values of the dependent variable corresponding to a particular curve at a maximum value of 10.0, the input stream would include

```
select> clip( y(max = 10.0))
```

Note that the maximum, minimum or final values used for annotation of the curve are not affected by clipping; the actual values for the entire curve are produced, regardless of whether clipping is selected.

Either *clip(x)* or specification of the *first* and/or *last* record(s) and/or the *start* and/or *end* value(s) of the independent variable offer a means to select a particular segment of a curve for plotting. The difference lies in the following. When clipping is selected, the position of every point in the complete curve is computed and the point is plotted, except that points outside the clipping boundary are plotted with "pen up". Additionally, data compression is suppressed. On the other hand, if the parameters, *first*, *last*, *start* or *end*, are used to delimit the curve segment, only the specified records are considered, and data compression is applied. Hence, in general, this method of "windowing" the data is more efficient than clipping.

7.6.3.2.2 Maximum, Minimum or Final Value Annotation

The maximum and/or minimum and/or final value of a curve may be annotated by means of the *select* flag, *note*. The *help* screen for this flag is shown below.

```
select.note> help

For the action, select, current parameters (attributes) are:

            NONE

and current flags (attributes) are:

            nofv        fv( x y label font symht ndec yonly xandy )
            nomax       max( x y label font symht ndec yonly xandy )
            nomin       min( x y label font symht ndec yonly xandy )
```

For example, the input required to annotate the maximum and final values of a curve has the form:

```
select> note( max fv )
```

The default is not to annotate (*nonote*), and if *note* is selected, the desired annotation flag must be explicitly specified (defaults are: *nofv*, *nomax* and *nomin*).

For each of the annotation flags, *fv*, *max* and *min*, the parameter attributes allow the user to specify a *label* string using a given font number, *font*, character size, *symht*, and precision of the annotation value, *ndec*. If present, the label is drawn first, followed by a number giving the value. The program will attempt to employ an *f* format to represent the number. However, if the value is too small or too large, an *e* format will be used.

The default font number is the current value of the corresponding *plot* parameter, if specified; otherwise, the value of the corresponding *layout* parameter, if specified; otherwise, the program default font is used.

The defaults for *symht* and *ndec* are the current values of the corresponding *plot* parameters, if specified; otherwise, the values of the corresponding *layout* parameters, if specified; otherwise, the program default values are used.

The default position of the annotation on the plot is near the point being annotated. The user may also specify the *x* and/or *y* coordinates in inches relative to the current plot origin of the starting location of the annotation.

The pair of flags, *yonly* and *xandy*, control the content of the annotation. If *yonly* (the default) is specified, only the ordinate value is noted. If *xandy* is invoked, both the abscissa and ordinate values are given. In this case, the format of the annotation is: (**x**,**y**), where **x** and **y** are the coordinates of the point.

7.6.3.2.3 Line Ticks

The *select* flag, *ticks*, requires that annotated tick marks be drawn at specified intervals along the curve. The *help* screen for this flag is shown below.

```
select.ticks> help

For the action, select, current parameters (attributes) are:

          var          delta

and current flags (attributes) are:

          nonote       note( var delta label font symht ndec )
```

If this flag is specified, tick marks are placed at intervals along the curve. The default is *noticks*.

Among the attributes of the flag, *ticks*, *var* is an integer valued parameter which determines the number of the plot variable which controls tick placement along the curve. The default is $var = 1$. *delta* is a real parameter determining the spacing between tick marks as a function of the selected line tick control variable. Note that this parameter must be greater than zero.

The flag attribute, *note*, controls whether the ticks are annotated. If *nonote* is selected (the default), ticks are drawn without annotation. If *note* is selected, the ticks are annotated according to the specified attributes.

As an attribute of the flag, *note*, *var* is an integer parameter which determines the column number (position in the plot record) of the tick annotation variable. The default is the same as the tick control variable. Note that the annotation variable need not be the same as the tick control variable. For example, it is possible to plot the y- versus the x-component of velocity, place tick marks at specific time intervals and note the corresponding value of the distance traveled.

delta is a real parameter determining the interval between annotation values as a function of the annotation variable. The default value is the same as the tick spacing interval. *label* is a string parameter which may be used to label the annotations. The label string is drawn first, if specified, followed by the current value of the annotation variable. *font* and *symht* are parameters which determine the font and character size used for tick annotation. Finally, *ndec* is the precision to use for the number annotation.

The defaults for *font*, *symht* and *ndec* are the current values of the corresponding *plot* parameters, if specified; otherwise, the values of the corresponding *layout* parameters, if specified; otherwise the program default values are used.

7.6.3.2.4 Error Bars

The *select* flag, *ebars*, provides the capability to draw error bars at selected points on the curve using data values contained in a specified plot vector.

The integer parameter attribute, *var*, determines the number of the plot vector containing the possible error values. The default is $var = 1$. The integer parameter attribute, *skip*, determines the interval between the error values which are actually drawn relative to the points selected for the curve. The default value for *skip* is 1, so that error bars are drawn at every point used to plot the curve. Note that, as an attribute of *ebars*, *skip* is independent of the *select* or *plot* parameter, *skip*, which determines which points in the plot vector are used to plot the curve. In particular, error bar values may be skipped without affecting whether data compression is used to draw the curve.

If the flag, *ebars*, is specified (the default is *noebars*), then at each selected point a vertical line is drawn which is centered on the curve and whose length is proportional to twice the corresponding value contained in the plot vector of possible error values. The end of each error bar is capped by a horizontal line whose width (in inches) is determined by the real parameter, *wid*. The default value for *wid* is the same as the current value of *symht*, as specified in either the *plot* or *layout* actions, or by the system default (0.1 inches).

7.6.4 Spherical Map

The main *help* screen for the *select* action in the spherical map is shown below.

```
select> help

For the action, select, current parameters (attributes) are:

        az        el        d         first     last      skip
        legend

and current flags (attributes) are:

        line( type )  noline
        nosym         sym( type intvl )
        nonote        note( noip ip nofp fp )
        noticks       ticks( var delta nonote note )
```

7.6.4.1 Parameters

For each curve in the spherical map, the azimuth variable is determined by assigning to the parameter, *az*, the index of the variable in the plot record (data column number), while the elevation variable is determined by assigning to the parameter, *el*, the appropriate column number. The radial distance of each point on the curve, if required, is determined by assigning the appropriate data column number to the parameter, *d*. The default value for *az* is the last value assigned for the current plot or 1, while the default values for *el* and *d* are undefined. The number of curves on a plot is determined by examination of assigned *el* values, and curve plotting stops at the first undefined *el* parameter value.

If *d* is not assigned, the curve defined by the azimuth and elevation variables is drawn on the surface of the sphere. If the radial distance is defined by one of the variables in the plot record, it should be noted that the program does not check whether the curve will fit on the page. It is the user’s responsibility to insure that the reference sphere radius is appropriate for the desired size of the resultant plot.

In addition to the variable numbers, the user may specify, independently for each curve, the *first* and/or *last* record(s) to be plotted. The default is to plot all of the data points in each curve. If the *first* and/or *last* record is specified, line ticks and selected initial/final point annotations will apply to the data in the included records only.

The integer parameter, *skip*, determines the interval between points in the plot vector which are actually used to draw the curve. For example, if *skip* = 2, then only the first, third, fifth, ..., points in the input data are used to draw the curve. Note, however, that initial and final values used for annotation are determined from all of the points in the selected plot vectors, and are not affected by the value of *skip*. The default value is *skip* = 1 so that every point is used.

Finally, a *legend* string may be associated with each curve for annotation. The position of the legend on the plot and other attributes may be specified in the *plot* action.

7.6.4.2 Flags

There are four possible representations of curves in McPlot. These are determined by the flags, *line*/*noline* and *sym*/*nosym*. The permutations are:

- *line* and *nosym* (the default). In this case, solid or dashed lines are drawn connecting the points on the curve. By default, the line type starts at 0 and is automatically incremented for each subsequent curve selected. The line type is reset to zero when a new plot is initiated.
- *line* and *sym*. In this case, one of the special centered symbols is placed at points along the curve and a line is drawn connecting the points. The interval (number of data points) between special symbols may be specified by the user. By default, a solid line (*type* = 0) is used to connect the points. The symbol type starts at 0 and is automatically incremented for each subsequent curve selected. The symbol type is reset to zero when a new plot is initiated.
- *noline* and *sym*. One of the special centered symbols is placed at points along the curve. The interval (number of data points) between special symbols may be specified by the user. By default, the symbol type starts at 0 and is automatically incremented for each subsequent curve selected. The symbol type is reset to zero when a new plot is initiated.
- *noline* and *nosym*. In this case, a single dot is placed at each point on the curve.

The line type of each curve may be specified by means of the integer parameter attribute, *type*, of the flag, *line*. If the line type is not specified by the user, the default value is employed as described above. Similarly, the attributes, *type* and *intvl*, of the flag, *sym*, allow the specification of the symbol type and the interval (number of points) between symbols. If the symbol type is not specified, the default value is employed as described above. The default for *intvl* is the current value of the *plot* parameter, *ssint*, if specified; otherwise, the value of the corresponding *layout* parameter, if specified; otherwise the program default value ($intvl = 1$) is used.

7.6.4.2.1 Initial or Final Point Annotation

The initial and/or final points of a curve may be annotated by means of the *select* flag, *note*. The *help* screen for this flag is shown below.

```
select.note> help

For the action, select, current parameters (attributes) are:

                NONE

and current flags (attributes) are:

                noip        ip( x y label font symht ndec )
                nofp        fp( x y label font symht ndec )
```

For example, the input required to annotate the final point on a curve has the form:

```
select> note(fp)
```

The default is not to annotate (*nonote*) and if *note* is selected, the desired annotation flag must be explicitly specified (defaults are *noip* and *nofp*).

For each of the annotation flags, *ip* (initial point) or *fp* (final point), the attributes allow the user to specify a *label* string with a given character size, *symht*, and number of decimal places in the annotation value, *ndec*. The default position of the annotation on the plot is near the point being annotated. The user may also specify the horizontal (*x*) and vertical (*y*) coordinates in inches relative to the current plot center of the starting position of the annotation. Note that, in the spherical map, it is necessary to specify both *x* and *y*; otherwise, the initial/final point annotation will be placed at the default position. The annotation values drawn on the plot have the format: (**az**,**el**[,**d**]), where the **az** and **el** values are in decimal degrees and **d** is in units of the sphere radius. If **d** is unity, so that the point is on the surface of the sphere, the distance is omitted from the annotation.

7.6.4.2.2 Line Ticks

The *select* flag, *ticks*, requires that annotated tick marks be drawn at specified intervals along the curve. The *help* screen for this flag is shown below.

```
select.ticks> help

For the action, select, current parameters (attributes) are:

          var       delta

and current flags (attributes) are:

          nonote    note( var delta label font symht ndec )
```

If this flag is specified, tick marks are placed at intervals along the curve. The default is *noticks*.

Among the attributes of the flag, *ticks*, *var* is an integer valued parameter which determines the number of the plot variable which controls tick placement along the curve. The default is $var = 1$. *delta* is a real parameter determining the spacing between tick marks as a function of the selected line tick control variable. Note that this parameter must be greater than zero.

The flag attribute, *note*, controls whether the ticks are annotated. If *nonote* is selected (the default), ticks are drawn without annotation. If *note* is selected, the ticks are annotated according to the specified attributes.

As an attribute of the flag, *note*, *var* is an integer parameter which determines the column number (position in the plot record) of the tick annotation variable. The default is the same as the tick control variable. Note that the annotation variable need not be the same as the tick control variable. For example, it is possible to plot the locus of an orbit, place tick marks at specific time intervals and note the corresponding value of the true anomaly.

delta is a real parameter determining the interval between annotation values as a function of the annotation variable. The default value is the same as the tick spacing interval. *label* is a string parameter which may be used to label the annotations. The label string is drawn first, if specified, followed by the current value of the annotation variable. *font* and *symht* are parameters which determine the font and character size used for tick annotation. Finally, *ndec* is the precision to use for the number annotation.

The defaults for *font*, *symht* and *ndec* are the current values of the corresponding *plot* parameters, if specified; otherwise, the values of the corresponding *layout* parameters, if specified; otherwise the program default values are used.

7.7 PLOT

The *plot* action controls the composition of the current plot on the page and causes it to be displayed on the selected output device.

7.7.1 Cartesian Map

The main *help* screen for the *plot* action in the Cartesian map is shown below.

```
plot> help

For the action, plot, current parameters (attributes) are:

            factor      origin      axlens      font        symht       numht
            ssht        ssint       ndec        first       last        skip
            start       end         title( x y font symht )

and current flags (attributes) are:

            x( label start scale font symht numht same new )           nox
            y( label start scale font symht numht same new )           noy
            nologx      logx( label icyc ncycs font symht numht same new )
            nology      logy( label icyc ncycs font symht numht same new )
            nopolar     polar
            nogrid      grid( dot line major minor x nox y noy )
            legend( x y font symht )            nolegend
            noclip      clip( nox x noy y )
            nosmooth    smooth( points deriv k noperiod periodic )
            next        same
            nopage      page( portr lands )
```

7.7.1.1 Parameters

The *plot* parameters, *factor*, *origin*, *axlens*, and *font*, *symht*, *numht*, *ssht*, *ssint* and *ndec*, have essentially the same interpretation as the corresponding *layout* parameters except that their scope is the current plot only. Thus, the user may specify the plot scale factor, location of the origin and axis lengths, the default character font number and symbol sizes used for annotations, special symbols, etc., individually for each plot, independent of the values chosen in the *layout* action.

For the parameter, *origin*, the following should be noted. The reference point for determining the position of the origin of the first plot on a page is the lower left-hand corner of the page. For subsequent plots, the reference point is the origin of the previous plot on the page.

The physical location of the origin on a page is also affected by the effective scale factor.

Before the first *plot* action, the effective scale factor is 1.0. For subsequent plots, insofar as the determination of the origin is concerned, the effective scale factor is the value of the parameter, *factor,* in the previous plot. This allows the user to determine the position of the current plot, using the previous plot as a reference, without having to take the scale factor into account. Note that this only applies to the parameter, *origin*, so that, for example, the physical length of an axis drawn as the result of a *plot* action is the product of the value of the appropriate component of *axlens* and *factor* set within that action.

The *plot* parameters, *first*, *last*, *skip*, *start* and *end*, have essentially the same interpretation as the corresponding *select* parameters except that they apply to *all* curves on the current plot. Thus, the user may select default values to use for the *first* and/or *last* record(s) and/or the *start* and/or *end* value(s) of the independent variable or the interval between successive data points actually used to plot the curve which pertain to all of the curves on the current plot. Note, however, that values of *first*, *last*, etc. given for each individual curve in the *select* action take precedence over the values input in the *plot* action. If the user input for *skip* is greater than 1, data compression is omitted for all curves on the plot.

Finally, the *plot* parameter, *title*, may be used to provide a plot title. It is a vector, so that multiple line titles are obtained by a comma separated list of title strings; e.g.,

```
plot> title = "first line", "second line", "third line"
```

Nominally, each title line is centered on the plot and the spacing between lines is one-half the character size. Currently, up to 20 lines may be input in one *plot* action. If more title lines are required, the user may continue on the same plot using the *plot* flag, *same*, described below and the new title lines will be appended, by default, to the previous lines.

The title attributes, *x* and *y*, allow the user to input the position of the beginning of the first line of the title in inches relative to the current plot origin. If *x* is specified, subsequent title lines will be left justified; otherwise, the title lines are centered on the plot.

Note that an attribute list immediately follows the parameter name and that parameters require an equal sign. Accordingly, to specify a title position the input might have the form:

```
plot> title(x=1.0 y=5.0) = "This is the title"
```

The *title* parameter attributes, *font* and *symht*, may be used to determine the font number and size of characters used to draw the title. If they are not specified, the current values of the corresponding *plot* parameters, *font* and *symht*, are used, if defined; otherwise, the value of the corresponding *layout* parameters are used, if defined; otherwise, the program default values are used.

7.7.1.2 Flags

There are basically two types of axis scaling available in McPlot, linear and logarithmic (base 10). The *plot* flags, *x*, *nox*, *y*, *noy*, *logx*, *nologx*, *logy* and *nology*, determine the scaling used for the axes. As an example, the *help* screen for the axis flags, *x* and *logy*, are shown below.

```
plot.x> help

For the action, plot, current parameters (attributes) are:

            label       start       scale       font        symht       numht

and current flags (attributes) are:

            same        new( bottom top )
```

```
plot.logy> help

For the action, plot, current parameters (attributes) are:

            label       icyc        ncycs       font        symht       numht

and current flags (attributes) are:

            same        new( left right )
```

The defaults, *x* and *y*, prescribe linear scaling for the x and y axes respectively. Selection of *nox* and/or *noy* also results in linear scaling but, in this case, while the curves will be plotted inside the area defined by the axis origin and length parameters, the corresponding axis will not actually be drawn. Specification of *nox* and *noy* without a preceding *select* action is one means of obtaining a null plot.

If either *logx* or *logy* is invoked, the corresponding coordinate will be plotted using a logarithmic scale. Finally, *nologx* and *nology* are equivalent to *x* and *y* in that they result in linear axis scaling.

Note that the type of scaling used may be selected for each axis independently.

An axis label may be specified for either linear or logarithmic axes by means of the string parameter attribute, *label*, and the font number and character size used for the label may be given by the parameters, *font* and *symht*. Similarly, the character size used for numerical axis annotations is determined by the parameter attribute, *numht*. Each of these character sizes may be determined independently for each axis. Note, however, that the font used for axis numerical annotations is the same as for the label. If no values are input, then the current values of the plot parameters, *font*, *symht* and *numht*, are used, if defined; otherwise, the corresponding *layout* values are used, if defined; otherwise, the program defaults are used.

By default, the scale factor and starting value on the axis (offset) are determined automatically for each axis. However, the attributes of the axis flags may be used to completely prescribe the scaling. For a linear axis, the user may determine the starting axis annotation value and the number of plot units per inch by means of the parameter attributes, *start* and *scale*. Note that the final axis annotation value is equal to

$$start + scale \times axlen$$

where *axlen* is the length of the axis in inches. Similarly, the initial cycle and total number of cycles on a logarithmic axis may be set by using the integer parameters, *icyc* and *ncycs*.

The user may draw additional curves on the same plot using the same axes and labels or, on option, new axes and labels. Either linear or logarithmic scaling may be selected for the new axes. The axis flags, *new* and *same*, control whether a new axis is drawn or whether the same axis is used for the additional curves.

The attributes of the flag, *new*, allow placement of the new axes at either the left- or right-hand side of the plot (y-axis) or along the top or bottom edge of the plot (x-axis). For a new x-axis (either linear or logarithmic), the options are: *bottom* (default) or *top*. For a new new y-axis (linear or logarithmic), the options are: *left* (default) or *right*. Thus, the following input sequence might be used to draw a new y-axis along the right-hand side of the current plot.

```
> plot same y(new(right) label="This is a new axis on the right-hand side");
```

There is no limit to the number of new axes (of any type: *left*, *right*, *bottom*, *top*) which may be drawn on a single plot and the types may be freely mixed as desired.

Note, however, that if a new x-axis is drawn along the top of the plot it may interfere with any title lines which are present or may be drawn in a subsequent *plot* action. It is the user's responsibility to adjust the *y* position of the title accordingly. See the discussion below concerning the *plot* flags, *next* and *same*, for a further description of this option.

7.7.1.2.1 Polar

For linearly scaled plots, the flag, *polar*, causes an extra set of axes to be drawn which cross at the point (0.0, 0.0) in plot units if that point is included within the plot; otherwise, the polar axes are drawn as close to the zero point as possible without going outside the plot boundaries. This flag has no other effect; in particular, it does not affect the axis length or scaling. The default is *nopolar*.

7.7.1.2.2 Gridding

If desired, a grid may be drawn on the current plot by simply invoking the *plot* flag, *grid*. The spacing and line type of the horizontal and vertical grid lines may be specified by the attributes of the flag, *grid*. If these are not specified in the *plot* action, default values are determined by the corresponding *layout* parameters, if specified; otherwise, the program defaults are used. Other attributes of the grid may be specified by the user as shown in the *help* screen for this flag.

```
plot.grid> help

For the action, plot, current parameters (attributes) are:

            NONE

and current flags (attributes) are:

            dot         line( type )
            major       minor
            x( intvl lpc )          nox
            y( intvl lpc )          noy
```

The *grid* flag attributes allow the user to specify the type and spacing of grid lines drawn for each axis.

The flag, *dot*, specifies that grid lines are composed of closely spaced dots and is the default. The dot spacing is currently 40 dpi for major axis divisions and 20 dpi for minor axis divisions. Alternatively, the grid may be made up of dashed lines by selection of the flag, *line*. In this case, the line type (0 - 8) may be specified by the parameter attribute, *type*. The default is a solid line (*type* = 0).

For linear axis scaling, the corresponding grid lines are linearly spaced. For a logarithmic axis, the grid will be logarithmic. Note that it is not currently possible to draw a linear grid on a logarithmic axis or a logarithmic grid on a linear axis.

For linear axes, the flag, *major*, specifies that grid lines are only drawn at major axis divisions (one-inch intervals) and is the default. If the *grid* attribute, *minor*, is selected, grid lines are also drawn at minor axis divisions determined by the axis scaling. For logarithmic axes, grid lines are automatically drawn at minor axis divisions, determined by the axis scaling, and the flags, *major* and *minor* have no effect. In this case, the default number of lines per cycle will depend on the length of the axis and the number of cycles.

By default, both the x and y axes are gridded if the *grid* option is selected. The user may optionally suppress the grid for a particular axis by specifying either of the *grid* flags, *nox* or *noy*.

The parameter attributes of the *grid* flags, *x* and *y*, may be used to specify the grid spacing on the current plot, independently for each axis. The parameter attribute, *intvl*, determines the spacing, in plot (scaled) units between grid lines for a linearly scaled axis. Alternatively, the parameter attribute, *lpc*, specifies the number of grid lines per cycle (= 2, 5, 10; other values are ignored) for a logarithmic axis.

7.7.1.2.3 Legend Block

The *plot* flag, *legend*, specifies that a legend block, containing any previously selected curve legends, is to be drawn on the current plot (default). The flag, *nolegend* may be invoked to prevent display of previously selected legends.

By default, the legend block is placed near the upper-right corner of the plot, one character height from the top and with the end of the legend at the right margin.

The attributes of the flag, *legend*, may be used to specify the actual position of the legend block and the font and character size used to draw the legends. The real parameter attributes, *x* and *y*, specify the location, in inches relative to the current plot origin, of the upper left-hand corner of the legend block. The *legend* parameter attribute, *font*, determines the font number while *symht* may be used to specify the character size. The defaults are the current values of the *plot* parameters, *font* and *symht*, if specified; otherwise, the values of the corresponding *layout* parameters, if specified; otherwise, the program defaults are used.

The longest legend string given in the *select* action(s) immediately preceding a *plot* is used to size the legend block. Each string is left-justified and followed by a 1.5 inch line sample if the curve is drawn with connecting lines. If only symbols are used to plot the points, the corresponding special symbol is drawn preceding the legend string. If additional curves are selected with legends and they are drawn on the *same* plot, the new legends are left justified by default and appended at the bottom. Note, however, that the start of the line sample in these appended legends may not line up with the previously drawn line samples, since the length of the legend block is recomputed for each *plot* action.

7.7.1.2.4 Clipping

The *plot* flag, *clip*, has the same function as the corresponding *select* flag except that it applies to all curves in the current *plot* action. Note that if the clipping option is selected, data compression is suppressed. The default is *noclip*. However, if clipping is specified for a particular curve in the *select* action, it has precedence over the flag value given in the *plot* action.

The help screen for the *plot* flag, *clip*, is shown below.

```
plot.clip> help

For the action, plot, current parameters (attributes) are:

            NONE

and current flags (attributes) are:

            nox         x( max min )
            noy         y( max min )
```

As is the case for the corresponding *select* flag, the user may independently clip the x- or y-coordinate and assign maximum and/or minimum values to be retained on the plot. In general, the default values for *max* and *min* are the maximum and minimum values of the data being plotted. Thus, if automatic scaling is used and neither *max* or *min* is specified, no actual clipping will occur. On the other hand, if the axis scale and offset are prescribed by the user, then invocation of the *clip* flag and either attribute, *x* or *y*, will automatically clip data points which extend beyond the range of the specified axis, even if no maximum or minimum value is given.

7.7.1.2.5 Data Smoothing

McPlot incorporates a cubic spline interpolation routine to provide a smooth curve between sparse data points. It is invoked by means of the *plot* flag, *smooth*. The default is not to interpolate (*nosmooth*). Note that if *smooth* is specified, the cubic spline interpolation is applied to *all* selected curves in the current plot.

If an independent variable, *x*, associated with a curve is not monotonically increasing throughout its selected range, the cubic spline interpolation is omitted in that case and the curve is drawn without smoothing. In addition, an error message is directed to *stderr*.

The number of points in the interpolated curves may be specified by the parameter attribute, *points*. The default value is 201. The user may also specify the real parameter, *k*, which determines the end-point boundary conditions:

$$y_0'' = k y_1'' \quad \text{and} \quad y_n'' = k y_{n-1}''$$

The nominal value for *k* is 0.0, but in some cases a better fit results from $k = 0.5$. If it is required that the output be periodic, the flag, *periodic*, may be invoked. In this case, if the first and last values of the dependent variable are not the same, they will each be set equal to their average value and a message will be written on *stderr*. The default is *noperiod*.

It is also possible to obtain an estimate of the first or second derivative of a curve by setting the integer parameter attribute, *deriv*, equal to 1 or 2. Note, however, that in this case the scaling may not be correct since the derivative will be plotted using the scale and offset determined for the original curve. The default value for *deriv* is 0, which gives the curve itself.

7.7.1.2.6 Adding Curves to a Plot

The user may draw additional curves on the same plot by specifying the *plot* flag, *same*, in a subsequent *plot* invocation. As an example, the sequence:

```
> select y=15, 16, 17;
> plot;
  .
  .                    % There may be intervening read, calc,
  .                    % layout or put actions here.
  .
> select y=18, 19, 20;
> plot same;
```

will place six curves on the same plot. In this case, the first three curves will determine the plot scale and the next three curves will be drawn using the same scale and with the same axes.

Note that the default is to move on to the *next* plot.

Warning: the axis scale factors are determined by the initial set of selected curves. If the appropriate scale factor for curves which are subsequently added is significantly larger than for the original curves, the length of the lines to be drawn may be very great and the curves may lie off the page. If such a curve is drawn with a dashed line, it may take a number of minutes to complete it. In this case, the program will appear to "hang"; i.e., nothing will appear in the plot window and the program will not respond to user input. It is the user’s responsibility to insure that curves which are added by means of a *plot same* request are consistent with the original scaling.

If a new axis is desired for a subsequent set of curves on the same plot, the input sequence,

```
> select y=15, 16, 17;
> plot;
> select y=18, 19, 20;
> plot same y(new);
```

will result in a second y-axis displaced from the first by an amount which depends on the axis annotation and label position of the first axis. By default, this axis will be automatically scaled to the selected curves.

The location of this second axis is determined solely by the flag attributes, *left*, *right*, *top* and *bottom* of the axis flag, *new*; the parameter, *origin*, has no effect in this instance. However, the length of the new axis may be set to a value which differs from that of the first axis and, of course, the new axis may have its own label and user specified scale factor and offset independent of the first. There is no limit to the number of axes which may be drawn on a single plot and either the x- or y-axis or both may be re-specified in this way.

The default is to use the *same* axes for data placed on the *same* plot.

7.7.1.2.7 Paging

The user may elect to start a new page for the current *plot* action by invoking the flag, *page*. For each new page, the orientation of the plots (portrait or landscape) may be determined by specifying one of the *page* attribute flags, *portr* or *lands*. The default is determined from the preceding *layout* action, if the page orientation is specified there; otherwise, *portr* (portrait mode) is assumed. Note, however, that if the plot orientation is changed without going to a new page, the results will be unpredictable (and probably not very pretty!). By default, a new page is started whenever the *stack* specified in the *layout* action is completed. This may be over-ridden by specifying the *plot* flag, *nopage*.

It is recommended that a new page be started whenever a new map is selected, since the automatic page layout does not accommodate a mixture of Cartesian and spherical plots. If plots of both types are desired on the same page, the *origin* or *center* must be specified for each plot.

7.7.2 Spherical Map

The main *help* screen for the *plot* action in the spherical map is shown below.

```
plot> help

For the action, plot, current parameters (attributes) are:

          factor    center    radius    azo       elo       dist
          intvl     pintvl    plat      first     last      skip
          ndec      font      symht     numht     ssht      ssint
          title( x y font symht )

and current flags (attributes) are:

          dotted( space )     solid
          legend( x y font symht )      nolegend
          next      same
          nopage    page( portr lands )
```

7.7.2.1 Parameters

The *plot* parameters, *factor*, *center*, *radius*, *azo*, *elo*, *dist*, *intvl*, *pintvl*, *plat* and *font*, *symht*, *numht*, *ssht*, *ssint* and *ndec*, have essentially the same interpretation as the corresponding *layout* parameters except that their scope is the current plot only. Thus, the user may specify the plot scale factor, center position, viewpoint, radius, sphere grid, the default character font number and symbol sizes used for annotations, special symbols, etc., individually for each plot, independent of the values chosen in the *layout* action.

For the parameter, *center*, the following should be noted. The reference point for determining the position of the center of the first reference sphere plotted on a page is the lower left-hand corner of the page. For subsequent plots, the reference point is the center of the current reference sphere.

The physical location of the center of the reference sphere on a page is also affected by the effective scale factor. Before the first *plot* action, the effective scale factor is 1.0. For subsequent plots, insofar as the determination of the sphere center is concerned, the effective scale factor is the value of the parameter, *factor*, in the previous plot. This allows the user to determine the position of the current plot, using the previous plot as a reference, without having to take the scale factor into account. Note that this only applies to the parameter, *center*, so that, for example, the physical radius of the sphere drawn as the result of a *plot* action is the product of *radius* and *factor* set within that action.

The *plot* parameters, *first*, *last* and *skip*, have essentially the same interpretation as the corresponding *select* parameters except that they apply to *all* curves on the current plot.

Thus, the user may select default values to use for the *first* and/or *last* record(s), or the interval between successive data points actually used to plot the curve which pertain to all of the curves on the current plot. Note, however, that values of *first*, *last*, etc. given for each individual curve in the *select* action take precedence over the values input in the *plot* action.

Finally, the *plot* parameter, *title*, may be used to provide a plot title. It is a vector, so that multiple line titles are obtained by a comma separated list of title strings; e.g.,

```
plot> title = "first line", "second line", "third line"
```

Nominally, each title line is centered over the sphere and the spacing between lines is one-half the character size. Currently, up to 20 lines may be input in one *plot* action. If more title lines are required, the user may continue on the same plot, using the *plot* flag, *same*, described below and the new title lines will be appended, by default, to the previous lines.

The title attributes, *x* and *y*, allow the user to input the position of the beginning of the first line of the title in inches relative to the center of the current reference sphere. If *x* is specified, subsequent title lines will be left justified; otherwise, the title lines are centered on the plot.

Note that an attribute list immediately follows the parameter name and that parameters require an equal sign. Accordingly, to specify a title position the input might have the form:

```
plot> title(x=1.0 y=5.0) = "This is the title"
```

The *title* parameter attributes, *font* and *symht*, may be used to determine the font number and size of characters used to draw the title. If they are not specified, the current values of the corresponding *plot* parameters, *font* and *symht*, are used, if defined; otherwise, the value of the corresponding *layout* parameters are used, if defined; otherwise, the program default values are used.

7.7.2.2 Flags

The *plot* flags, *dotted* and *solid*, have the same interpretation as the corresponding *layout* flags except that their scope is the current plot only. If neither of these flags is selected, the default is determined from the preceding *layout* action, if specified there; otherwise, the program default (*dotted*) is used.

The integer parameter attribute, *space*, of the *plot* flag, *dotted*, has the same interpretation as the corresponding *layout* parameter except that its scope is the current plot only. If this parameter is not specified, the default is determined from the preceding *layout* action, if specified there; otherwise, the program default (*space* = 2) is used.

7.7.2.2.1 Legend Block

The *plot* flag, *legend*, specifies that a legend block, containing any previously selected curve legends, is to be drawn on the current plot (default). The flag, *nolegend*, may be invoked to prevent display of previously selected legends.

By default, the legend block is placed one character width (*symht*) to the right of the reference sphere projection and one character height down from the top.

The attributes of the flag, *legend*, may be used to specify the actual position of the legend block and the font and character size used to draw the legends. The real parameter attributes, *x* and *y*, specify the location, in inches relative to the current reference sphere center, of the upper left-hand corner of the legend block. The *legend* parameter attribute, *font*, determines the font number while *symht* may be used to specify the character size. The defaults are the current values of the *plot* parameters, *font* and *symht*, if specified; otherwise, the values of the corresponding *layout* parameters, if specified; otherwise, the program defaults are used.

The longest legend string given in the *select* action(s) immediately preceding a *plot* is used to size the legend block. Each string is left-justified and followed by a 1.5 inch line sample if the curve is drawn with connecting lines. If only symbols are used to plot the points, the corresponding special symbol is drawn preceding the legend string. If additional curves are selected with legends and they are drawn on the *same* plot, the new legends are left justified by default and appended at the bottom. Note, however, that the start of the line sample in these appended legends may not line up with the previously drawn line samples, since the length of the legend block is recomputed for each *plot* action.

7.7.2.2.2 Adding Curves to a Plot

The user may draw additional curves on the same plot by specifying the *plot* flag, *same*, in a subsequent *plot* invocation. As an example, the sequence:

```
> select el=2, 3, 4;
> plot;
  .
  .                 % There may be intervening read, calc,
  .                 % layout or put actions here.
  .
> select el=5, 6, 7;
> plot same;
```

will place six curves on the same reference sphere.

Note that the default is to move on to the *next* plot.

7.7.2.2.3 Paging

The user may elect to start a new page for the current *plot* action by invoking the flag, *page*. For each new page, the orientation of the plots (portrait or landscape) may be determined by specifying one of the *page* attribute flags, *portr* or *lands*. The default is determined from the preceding *layout* action, if the page orientation is specified there; otherwise, *portr* (portrait mode) is assumed. Note, however, that if the plot orientation is changed without going to a new page, the results will be unpredictable (and probably not very pretty!). By default, a new page is started whenever the *stack* specified in the *layout* action is completed. This may be over-ridden by specifying the *plot* flag, *nopage*.

It is recommended that a new page be started whenever a new map is selected, since the automatic page layout does not accommodate a mixture of Cartesian and spherical plots. If plots of both types are desired on the same page, the *origin* or *center* must be specified for each plot.

7.7.3 Scope

The scope of *plot* parameters and flags is the current *plot* action only. All *plot* parameters and flags revert to their default values when the next *plot* action is initiated.

7.8 PUT

The *put* action may be used to place additional annotation on a plot; e.g., strings, numbers, data values, symbols, arcs, lines, arrows, boxes and circles.

7.8.1 Cartesian Map

The *help* screen for the *put* action in the Cartesian map is shown below.

```
put>

For the action, put, current parameters (attributes) are:

            num( x y ang ndec numht label font symht )
            str( x y ang font symht )

and current flags (attributes) are:

            nosym       sym( x y type symht )
            nocircle    circle( x y rad type )
            nobox       box( x y wid ht ang type )
            noarc       arc( from to thru type )
            noline      line( from to type )
            nodata      data( var rec x y ang ndec numht label font symht )
            noarrow     arrow( from to type size )
            inches      scale
```

7.8.1.1 Origin

The reference point for all *put* position attributes (*x*, *y*, *from*, *to*, *thru*) is the current origin. Before any *plot* action is invoked, the current origin is the lower-left corner of each page. Subsequently, the current origin coincides with the origin of the last plot. Note that the current origin may be changed at any time by user input of the form,

> plot nopage nox noy origin=**x**, **y**;

without a preceding *select* action. In the above, **x** and **y** are the desired coordinates of the new origin.

7.8.1.2 Parameters

The *put* parameter, *str*, allows the user to place a string at any point on the page. For example, to place a string at the coordinates of the last point plotted, the input is simply:

> put str = "This is a string";

The user may specify the position of the string, the angle at which it is drawn and the font number and character size by means of the parameter attributes, *x*, *y*, *ang*, *font* and *symht*.

The position of the lower-left corner of the first character in the string is determined by the parameters, *x* and *y*. Their default values are the corresponding coordinates of the last point drawn on the page (either in a *plot* or a *put* action). Note that input values of *x* and *y* may be given in *inches* relative to the current plot origin or in plot units (*scale*); see the discussion of the *put* flags, *inches* and *scale*, below.

The angle at which the string is drawn relative to the x-axis is determined by the attribute, *ang,* and is input in degrees. A positive value of *ang* produces a counter-clockwise rotation, while a negative value results in a clockwise rotation. The default value is $ang = 0.0$.

The font number is determined by the integer parameter*, font*, while the size (in inches) of the characters used to draw the string is given by *symht*. If no values are input for these attributes, then the corresponding *layout* values are used, if defined; otherwise, the program defaults are used.

The *put* parameter, *num*, allows the user to place a number on the plot. The position of the number and the angle at which it is drawn may be input in the same manner as for the parameter, *str*. The user may also specify the precision, *ndec*, and character size, *numht*, used to draw the number. Normally, an *f* format is used to describe the number; however, if the number is too small to be usefully represented with the given precision, or if it is too large, an *e* format is used.

The number may be labeled by means of the attribute, *label*. If present, the label is drawn first, followed by the assigned number. The character size used for the label is given by *symht*, and may be specified independently of the number size. The integer parameter, *font*, determines the font number used for both the label and the number itself.

If any of the attributes, *font*, *symht*, *numht* or *ndec*, is not specified, then the corresponding *layout* value is used, if defined; otherwise, the program default is used.

7.8.1.3 Flags

The *put* flag, *sym*, may be used to place any one of the special centered symbols at any point on the current page. The default is *nosym*. The symbol type is determined by the integer parameter attribute, *type*. The default value is *type* = 0. The character size used for the symbol may be specified by the parameter attribute, *symht*. The default value is the value of *ssht* given in the *layout* action, if specified; otherwise, it is the program default. The attributes, *x* and *y*, may be used to specify the position of the centered symbol. The default location is at the coordinates of the last point placed on the page.

The *put* flag, *circle*, may be used to draw a circle of specified radius centered at any point on the page. The position of the center of the circle is determined by specifying the coordinates, *x* and *y*, in either inches or plot units relative to the current origin, depending on whether the *put* flag, *inches* or *scale*, is invoked. The default for *x* and *y* is the corresponding coordinate of the last point placed on the page in either a *plot* or a *put* action. The parameter, *rad*, specifies the radius of the circle in inches (only). The default is *rad* = 0.0. *type* specifies the line type used to draw the circle (the default is *type* = 0).

A rectangular box of specified width and height (measured in either inches or plot units) may be drawn anywhere on a page using the *put* flag, *box*. The default is *nobox*. The parameter attributes, *x* and *y*, refer to the location of the upper-left corner of the box. Their defaults are the corresponding coordinates of the last point placed on the page. The dimension of the box along the x-axis is specified by the parameter attribute, *wid*, while *ht* gives the dimension along the y-axis. The default value for these parameters is $ht = wid = 0.0$.

The parameter, *ang*, may be used to rotate the box about the point locating the upper left hand corner. Positive values of *ang* result in counter-clockwise rotations; negative values yield clockwise rotations. The default angle is *ang* = 0.0.

The line type used to draw the box is determined by the parameter, *type* (default = 0).

The *put* flag, *arc*, allows the user to draw an arc of a circle by specifying the beginning and end points of the arc and a desired intermediate point. The default is not to draw an arc (*noarc*).

The parameter attributes, *from*, *to* and *thru*, specify, respectively, the beginning, ending and intermediate points on the arc. These are vectors, (x, y), so that the input required to draw an arc from the point, (1.0, 2.0), to (3.0, 2.0) through the intermediate point, (2.0, 1.0), for example, would have the form:

```
> put arc(from=1,2 to=3,2 thru=2,1);
```

The default value of each component of *from*, *to* and *thru* is the corresponding coordinate of the last point plotted (whether in a *plot* or *put* action). For example,

```
> put arc(from=0 to=6 thru=3,0);
```

will draw an arc which starts at the y-axis and ends at a point six inches to the right of the y-axis at the y-coordinate of the last point plotted, while

```
> put line(from= ,0 to= ,6 thru=0,3) scale;
```

will result in an arc which starts at y=0 and ends at y=6 (in plot units) at the x-coordinate of the last point plotted.

Note: if the three points, *from*, *to* and *thru*, are collinear, a straight line is drawn between the points defined by *from* and *to*.

The (dashed) line type used to draw the arc is determined by the integer parameter attribute, *type*. The default value is *type* = 0.

The *put* flag, *line*, allows the user to place a straight line of a given type between any two points on the current page. It may be canceled by the flag, *noline*, which is the default.

The endpoints of the line are specified by the parameter attributes, *from* and *to* as for the flag, *arc*.

The (dashed) line type used is determined by the integer parameter attribute, *type*. The default value is *type* = 0.

The *put* flag, *data*, may be used to place the value of any plot data item stored in memory on the current plot. It may be canceled by the flag, *nodata*, which is the default.

The parameter attributes of the flag, *data*, are used to specify the data item and the associated annotation. *var* is the variable number (position in the plot record) corresponding to the data point to be put on the plot (default = 1). *rec* is the record number of the desired data value (default = 1). *x* and *y* give the position of the start of the annotation. The default is the position of the last point placed on the page. *ang* specifies the angle of the annotation in degrees relative to the horizontal axis of the page The default is *ang* = 0.0. Positive values of *ang* produce counter-clockwise rotations; negative values yield clockwise rotations. *label* is a string giving the annotation label. If a label is specified, it is drawn first, followed by the data value. *ndec* is the precision (number of decimal places) to be used for the number annotation. *symht* and *numht* are the label and number symbol sizes in inches. *font* is the number of the character font used to draw the data value and label. The defaults for these attributes are the values of the corresponding *layout* parameters, if defined; otherwise, the program defaults are used.

This option may be used to spot check selected values of the input data, or to allow certain annotation values to be passed to McPlot in the same file as the plot data (see Example 12).

Finally, the *put* flag, *arrow*, may be selected to place an arrow connecting any two points on the page. The endpoints of the arrow are specified by the parameter attributes, *from* and *to*, as for the *put* flags, *arc* and *line*. *type* determines the line type used to draw the shaft of the arrow, while *size* determines the size of the arrow head in inches. The default value for *type* is 0, while the default value for *size* is the value of *symht* given in the *layout* action, if specified; otherwise, it is the value of the program default symbol size (0.1 inches).

The *put* flag, *inches* (default), may be invoked if user input of positions and dimensions is desired in inches. Alternatively, the flag, *scale*, may be used to specify input values in plot units. These flags apply to the attributes, *x* and *y*, of *sym*, *circle*, *box*, *num*, *data* and *str*; to the attributes, *from* and *to*, of *arc*, *line* and *arrow*, as well as the attribute, *thru* of *arc*; and to the attributes, *wid* and *ht*, of *box*. Note that the default values for position attributes (either the x- or y-coordinate, or both) are the corresponding coordinates of the last point placed on the page, regardless of whether *inches* or *scale* is specified.

7.8.2 Spherical Map

The *help* screen for the *put* action in the spherical map is shown below.

```
put> help

For the action, put, current parameters (attributes) are:

        num( pos ang ndec numht label font symht )
        str( pos ang font symht )

and current flags (attributes) are:

        nosym       sym( pos type symht )
        nocircle    circle( pos rad type )
        nobox       box( pos ang wid ht type )
        noline      line( from to type arc ray )
        nodata      data( var rec pos ang ndec numht label font symht )
        noarrow     arrow( from to type size arc ray )
        deg         inches
```

7.8.2.1 Origin

In the spherical map, the reference point for all *put* position attributes (*pos*, *from*, *to*) is the current origin. Before any *plot* action is invoked, the current origin is the lower-left corner of each page. Subsequently, the current origin coincides with the center of the last plot. In the spherical map, it is not possible to change the current origin except through the *plot* action.

7.8.2.2 Location of Points

In the *put* action in the spherical map, the specification of the location of strings, symbols, numbers, data values, starting and ending points of lines, etc., to be placed as additional annotation on a plot is determined by the vector parameters, *pos*, *from* and *to*. The interpretation of these parameters and the form of the input depends on user selection of one of the *put* flags, *deg* or *inches*.

If *deg* (the default) is specified, the input is of the form, *pos* = **az**, **el** [, **d**], where **az**, **el** and **d** are the values of the azimuth and elevation (in degrees) and radial distance desired. If **d** is not specified, it is assumed that the point is on the surface of the sphere (**d** = 1). Thus, the input,

```
> put sym(pos=28, –51);
```

will place a special centered symbol at an azimuth of 28 degrees and an elevation of –51 degrees on the surface of the sphere.

If the flag, *inches*, is selected, the input position is of the form, *pos* = **x**, **y**, where **x** and **y** are the horizontal and vertical coordinates (in inches) relative to the center of the sphere. Thus, the input,

```
> put sym(pos=3.0, 2.5) inches;
```

will place a special centered symbol 3.0 inches to the right and 2.5 inches above the center of the sphere.

In all cases, the default values of *pos*, *from* and *to* are the coordinates of the last point plotted relative to the sphere if the flag, *deg*, is invoked, or the last point drawn on the page if the flag, *inches*, is invoked. Note, also, that both the azimuth and elevation components of *pos*, etc., must be specified; otherwise, the appropriate default values will be used.

7.8.2.3 Parameters

The *put* parameter, *str*, allows the user to place a string at any point on the page. For example, to place a string at the coordinates of the last point plotted, the input is simply:

```
> put str = "This is a string";
```

The user may specify the location of the string, the angle at which it is drawn and the font number and character size by means of the real parameter attributes, *pos*, *ang*, *font* and *symht*.

The position of the lower-left corner of the first character in the string is determined by the vector parameter, *pos*, as noted above. Whether the input value of *pos* is interpreted in terms of azimuth, elevation and distance from the center of the sphere, or in terms of the horizontal and vertical position in inches relative to the current sphere center is determined by the put flags, *deg* (default) or *inches*.

The angle at which the string is drawn relative to the horizontal is determined by the attribute, *ang*, and is input in degrees. A positive value of *ang* produces a counter-clockwise rotation, while a negative value results in a clockwise rotation. The default value is $ang = 0.0$.

The font number is determined by the integer parameter, *font*, while the size (in inches) of the characters used to draw the string is given by *symht*. If no values are input for these attributes, then the corresponding *layout* values are used, if defined; otherwise, the program defaults are used.

The *put* parameter, *num*, allows the user to place a number on the plot. The position of the number and the angle at which it is drawn may be input in the same manner as for the parameter, *str*. The user may also specify the precision, *ndec*, and character size, *numht*, used to draw the number. Normally, an *f* format is used to describe the number; however, if the number is too small to be usefully represented with the given precision, or if it is too large, an *e* format is used.

The number may be labeled by means of the attribute, *label*. If present, the label is drawn first, followed by the assigned number. The character size used for the label is given by *symht* and may be specified independently of the number size. The integer parameter, *font*, determines the font number used for both the label and the number itself.

If any of the attributes, *font*, *symht*, *numht* or *ndec*, is not specified, then the corresponding *layout* value is used, if defined; otherwise, the program default is used.

7.8.2.4 Flags

The *put* flag, *sym*, may be used to place any one of the special centered symbols at any point on the current page. The default is *nosym*. The symbol type is determined by the integer parameter attribute, *type*. The default value is *type* = 0. The character size used for the symbol may be specified by the parameter attribute, *symht*. The default value is the value of *ssht* given in the *layout* action, if specified; otherwise, it is the program default. The vector parameter attribute, *pos*, may be used to specify the location of the centered symbol as noted above.

The *put* flag, *circle*, may be used to draw a circle of specified radius centered at any point on the page. The default is *nocircle*. The geometrical character of the circle is determined by the *put* flags, *deg* and *inches*. If *deg* (default) is specified, the result is a small circle (locus of points having the same *angular* separation from the center of the circle), drawn in perspective; if *inches* is selected, the result is an ordinary circle. The interpretation of the parameters, *pos* and *rad*, are similarly affected.

The location of the center of the circle is determined by specifying the vector parameter, *pos*, as described above. The parameter, *rad*, specifies the radius of the circle. If *deg* is specified, the radius is in decimal degrees, while if *inches* is selected, the radius is assumed to be in inches. The default is *rad* = 0.0.

type specifies the line type used to draw the circle (default = 0).

A quadrilateral figure of specified width and height may be drawn anywhere on a page using the *put* flag, *box*. The default is *nobox*. The geometrical character of the quadrilateral is determined by the *put* flags, *deg* and *inches*. If *deg* (default) is specified, the quadrilateral is formed by the intersection of segments of four great circles, drawn in perspective, such that the angular separation of the sides is equal to the specified width and height of the box. On the other hand, if *inches* is selected, the result is an ordinary rectangle of specified width and height.

The interpretation of the parameter attributes, *pos*, *wid* and *ht*, are similarly affected by the choice of *deg* or *inches*. If *deg* (default) is specified, the vector parameter attribute, *pos*, refers to the location of the center of the box as described above. On the other hand, if *inches* is specified, the vector parameter attribute, *pos*, refers to the location of the upper left-hand corner of the box in inches relative to the center of the sphere. The default corresponds to the coordinates of the last point placed on the page.

Similarly, the dimension of the box along the horizontal direction (width) is specified by the parameter attribute, *wid*, while *ht* gives the dimension along the vertical direction (height). If *deg* is specified, *wid* prescribes the minimum angular separation of the vertical sides, while *ht* is the minimum angular separation of the horizontal sides. If *inches* is selected, *wid* and *ht* refer to the corresponding dimensions in inches. The default value for these parameters is $ht = wid = 0.0$.

The parameter *ang* may be used to rotate the box about the point defined by *pos*. Positive values of *ang* result in counter-clockwise rotations; negative values yield clockwise rotations. The default angle is *ang* = 0.0.

The line type used to draw the box is determined by the parameter, *type* (default = 0).

The *put* flag, *line*, allows the user to place a line of a given type between any two points on the current page. It may be canceled by the flag, *noline*, which is the default.

The (dashed) line type used is determined by the integer parameter attribute, *type*. The default value is *type* = 0.

The geometrical character of the line is determined by the *put* flags, *deg* and *inches*. If *deg* is specified, the line may be either a segment of a great circle (default) or a ray (straight line) connecting points relative to the sphere and drawn in perspective; if *inches* is selected, the result is a straight line which may connect any two points on the page. The interpretation of the parameters, *from* and *to*, are similarly affected.

If the *put* flag, *deg*, is specified, the *line* flag attributes, *arc* and *ray*, determine whether the line is a great circle or a straight line. If *arc* is specified (default), the result is a great circle segment. Selection of the flag, *ray*, produces a straight line. If the *put* flag, *inches*, is specified, *arc* and *ray* have no effect.

The endpoints of the line are specified by the vector parameter attributes, *from* and *to*, as described above. Thus, the input required to draw a segment of a great circle from the point defined by azimuth = 20.0, elevation = 0.0, to the point defined by azimuth = 30.0, elevation = –40.0, for example, would have the form:

```
> put line(from=20.0, 0.0 to=30.0, –40.0);
```

Note that, if the radial component of *from* and *to* is specified, it must be the same in each case if a great circle (*arc*) is selected. In addition, if the *from* and *to* values specified for a great circle are antipodes, the path is ambiguous. The program automatically resolves this ambiguity by choosing a route through one of the poles.

The *put* flag, *data*, may be used to place the value of any plot data item stored in memory on the current plot. It may be canceled by the flag, *nodata*, which is the default.

The parameter attributes of the flag, *data*, are used to specify the data item and the associated annotation. *var* is the variable number (position in the plot record) corresponding to the data point to be put on the plot (default = 1). *rec* is the record number of the desired data value (default = 1). The vector parameter, *pos*, gives the position of the start of the annotation as described above. *ang* specifies the angle, in degrees relative to the horizontal axis of the page, of the annotation. The default is *ang* = 0.0. Positive values of *ang* produce counter-clockwise rotations; negative values yield clockwise rotations. *label* is a string giving the annotation label. If a label is specified, it is drawn first, followed by the data value. *ndec* is the precision (number of decimal places) to be used for the number annotation. *symht* and *numht* are the label and number symbol sizes in inches. *font* is the number of the character font used to draw the data value and label. The defaults for these attributes are the values of the corresponding *layout* parameters, if defined; otherwise, the program defaults are used.

This option may be used to spot check selected values of the input data, or to allow certain annotation values to be passed to McPlot in the same file as the plot data.

Finally, the *put* flag, *arrow*, may be selected to place an arrow connecting any two points on the page. The default is *noarrow*. The geometrical character of the arrow is determined by the *put* flags, *deg* and *inches*. If *deg* is specified, the arrow shaft may be either a segment of a great circle (default) or a ray (straight line) connecting points relative to the sphere and drawn in perspective; if *inches* is selected, the shaft is a straight line which may connect any two points on the page. The interpretation of the parameters, *from* and *to*, are similarly affected.

If the *put* flag, *deg*, is specified, the *arrow* flag attributes, *arc* and *ray*, determine whether the arrow shaft is a segment of a great circle or a straight line. If *arc* is specified (default), the result is a great circle segment. Selection of the flag, *ray*, produces a straight line for the arrow shaft. If the *put* flag, *inches*, is specified, *arc* and *ray* have no effect.

The endpoints of the arrow are specified by the parameter attributes, *from* and *to*, as for the *put* flag, *line*. *type* determines the line type used to draw the shaft of the arrow, while *size* determines the size of the arrow head in inches. The default value for *type* is 0, while the default value for *size* is the value of *symht* given in the *layout* action, if specified; otherwise, it is the value of the program default symbol size (0.1 inches).

The *put* flag, *deg* (default), may be invoked if output appropriate for perspective geometry on a sphere is desired, with user input of positions and dimensions in degrees. Alternatively, the flag, *inches*, may be used to specify input values in inches relative to the current origin (*center*) of the sphere. These flags apply to the attribute, *pos*, of *sym*, *circle*, *box*, *num*, *data* and *str*; to the attributes, *from* and *to*, of *line*, and *arrow*, and to the attributes, *wid* and *ht*, of *box* and the attribute, *rad*, of *circle*. Note that the default value of any position attribute is given by the coordinates of the last point drawn relative to the sphere (*deg*) or the corresponding coordinate of the last point placed on the page (*inches*).

7.9 Scope

The scope of *put* flags and parameters is the current *put* action only. All *put* flags and parameters revert to their default values in subsequent *put* actions.

One of each object (symbol, circle, box, [arc,] line, number, datum, string or arrow) may be drawn in a single *put* action. The position of each item may be specified independently. If no position values are given, then the default values are used. Regardless of the order of input, objects in a single *put* action are always drawn in the order: symbol, circle, box, [arc,] line, number, datum, string, arrow. This allows the user to place a symbol or circle or box, extend an arc or line from it, write a labeled number or datum and follow the number with a string and then an arrow pointing to a feature of the plot, without having to specify any positions on the plot except for those of the initial symbol and line segment and the head of the arrow.

McPlot Quick Reference

Subject	**Action**	**Variable**	**Attribute**	**Reference Pages**
ACSL data files	read	acsl		80
add curves to plot	plot	same		117, 121
additional variables (memory)	read	nextra		80
annotation character font	layout	font		91, 96
annotation character size	layout	symht		91, 96
ASCII data files	read	ascii		80
arrow	put	arrow		126, 130
axis annotation character font	layout	font		91, 96
axis annotation number size	layout	numht		92, 96
axis label (linear)	plot	x, y	label	112
axis label (logarithmic)	plot	logx, logy	label	112
axis length (all plots)	layout	axlens		91
axis length (current plot)	plot	axlens		110
axis offset	plot	x, y	start	112
axis (origin of current plot)	plot	origin		110
axis (origin of first plot)	layout	origin		90
axis scale (linear)	plot	x, y	scale	113
azimuth angle selection	select	az		106
blank plot	plot	nox, noy		112
box	put	box		124, 129
C language data files	read	c		80
calculator	calc			82
Cartesian map	map	xy		89
center of first sphere	layout	center		94
center of current sphere	plot	center		119
centered symbol interval	select	sym	intvl	102, 107
centered symbol size	layout	ssht		92, 96
centered symbol type	select	sym	type	102, 107
change data	calc	rcl, sto		83
character font	layout	font		91, 96
character size	layout	symht		91, 96
circle	put	circle		124, 129
clip (all curves)	plot	clip	x, y	115
clip (current curve)	select	clip	x, y	102
cubic spline	plot	smooth		116

McPlot Quick Reference

Subject	Action	Variable	Attribute	Reference Pages
data file name	read	file		80
data record length	read	nvar		80
data value	put	data		125, 130
date of plot	layout	time		93, 96
dependent variable selection	select	y		100
derivative	plot	smooth	deriv	116
distance selection	select	d		106
double precision data	read	double		80
draw arrow	put	arrow		126, 130
draw box	put	box		124, 129
draw circle	put	circle		124, 129
draw line	put	line		125, 130
draw symbol	put	sym		124, 129
elevation angle selection	select	el		106
error bars	select	ebars		105
extra variables (memory)	read	nextra		80
factor (all plots)	layout	factor		90, 94
factor (current plot)	plot	factor		110, 119
final point annotation	select	note	fp	108
final value annotation	select	note	fv	103
font (all plots)	layout	font		91, 96
font (current plot)	plot	font		110, 119
Fortran data files	read	various		80
grid (all plots)	layout	grid		92
grid (current plot)	plot	grid		113
grid line type	layout	grid	type	92
independent variable selection	select	x		100
index of record (all curves)	plot	first, last		111, 119
index of record (current curve)	select	first, last		100, 106
initial point annotation	select	note	ip	108
input file	input	file		79
interpolation	plot	smooth		116
interval between symbols	select	sym	intvl	102, 107
landscape mode (current page)	plot	page	lands	118, 121
landscape mode (every page)	layout	lands		92, 96

McPlot Quick Reference

Subject	Action	Variable	Attribute	Reference Pages
legend for current curve	select	legend		101, 107
legend block	plot	legend		114, 120
line	put	line		125, 130
line ticks	select	ticks		104, 109
line type	select	line	type	101, 107
line type for grid	layout	grid	type	92
linear axis label	plot	x, y	label	112
linear axis	plot	x, y		112
linear grid (all plots)	layout	grid		92
linear grid (current plot)	plot	grid		114
logarithmic axis label	plot	logx, logy	label	112
logarithmic axis	plot	logx, logy		112
logarithmic grid (all plots)	layout	grid		92
logarithmic grid (current plot)	plot	grid		114
Matlab data files	read	matlab		80
Matlab variable name	read	matlab	name	80
maximum (clip all curves)	plot	clip	x, y (max)	115
maximum (clip current curve)	select	clip	x, y (max)	102
maximum value annotation	select	note	max	103
minimum (clip all curves)	plot	clip	x, y (min)	115
minimum (clip current curve)	select	clip	x, y (min)	102
minimum value annotation	select	note	min	103
multiple legends	plot	same		115, 120
multiple line titles	plot	title		111, 120
multiple linear axes	plot	x, y	new	113
multiple logarithmic axes	plot	logx, logy	new	113
name of Matlab variable	read	matlab	name	80
new linear axis	plot	x, y	new	113
new logarithmic axis	plot	logx, logy	new	113
new page	plot	page		118, 121
next plot	plot	next		118, 121
no axis	plot	nox, noy		112
number	put	num		124, 128
number of plots per page	layout	stack		91, 95
number size for axis annotation	layout	numht		92, 96

McPlot Quick Reference

Subject	**Action**	**Variable**	**Attribute**	**Reference Pages**
observation point (all plots)	layout	azo, elo, dist		93
observation point (current plot)	plot	azo, elo, dist		119
offset	plot	x, y	start	111
orientation of all pages	layout	portr, lands		92, 96
orientation of current page	plot	page	portr, lands	118, 121
origin of current plot	plot	origin		110
origin of first plot	layout	origin		90
plot factor (all plots)	layout	factor		90, 94
plot factor (current plot)	plot	factor		110, 119
plot data file name	read	file		80
plot spacing	layout	delta		91, 95
plot title	plot	title		111, 120
polar plot	plot	polar		113
portrait mode (current page)	plot	page	portr	118, 121
portrait mode (every page)	layout	portr		92, 96
precision of annotation values	layout	ndec		91, 95
radius of sphere (all plots)	layout	radius		94
radius of sphere (current plot)	plot	radius		119
radial coordinate selection	select	d		106
range of x (all curves)	plot	start, end		111
range of x (current curve)	select	start, end		100
read data	read			80
recall data	calc	rcl		83
record length	read	nvar		80
rectangle	put	box		124, 129
same plot	plot	same		117, 121
same page	plot	nopage		118, 121
scale factor for plot	plot	factor		110, 119
scale for axis	plot	x, y	scale	112
separation between plots	layout	delta		91, 95
single precision data	read	single		80
skip data points	select	skip		101, 106
smoothing	plot	smooth		116
spacing of plots	layout	delta		91, 95

McPlot Quick Reference

Subject	**Action**	**Variable**	**Attribute**	**Reference Pages**
special symbol interval	select	sym	intvl	102, 107
special symbol size	layout	ssht		92, 96
special symbol type	select	sym	type	102, 107
sphere longitude/latitude spacing	layout	intvl		95
sphere grid style	layout	dotted, solid		96
spherical map	map	sphere		89
store data	calc	sto		83
string	put	str		123, 128
symbol	put	sym		124, 129
symbol type for curve	select	sym	type	102, 107
symbols on a curve	select	sym		101, 107
tick marks on a curve	select	ticks		104, 109
time of plot	layout	time		93, 96
title of plot	plot	title		111, 120
viewpoint (all plots)	layout	azo, elo, dist		94
viewpoint (current plot)	plot	azo, elo, dist		119