Defence Research and Development Canada
Recherche et développement pour la défense Canada

DEFENCE R&D DÉFENSE

SAPLOT Maintenance and Enhancement

C. Calnan

Prepared By:
C. Calnan
xwave
38 Solutions Drive
Halifax, N.S.
B3S 1N2

Contractor's Document Number: 1012063 PWGSC
Contract Number: W7707-06-3594 11cs
CSA: J. Theriault, 902-426-3100 ext 376

Defence R&D Canada - Atlantic
Contract Report
DRDC Atlantic CR 2007-132
August 2007

Canada

SAPLOT Maintenance and Enhancement

C. Calnan

Prepared By:
C. Calnan
xwave
38 Solutions Drive
Halifax, N.S. B3S 1N2

Contractor's Document Number: 1012063
PWGSC Contract Number: W7707-06-3594 11cs
CSA: J. Theriault, 902-426-3100 ext 376

Defence Research and Development Canada – Atlantic

Contract Report

DRDC Atlantic CR 2007-132

August 2007

IMPORTANT INFORMATIVE STATEMENTS

The scientific or technical validity of this Contract Report is entirely the responsibility of the Contractor and the contents do not necessarily have the approval or endorsement of the Department of National Defence of Canada.

Abstract

Previous contracts resulted in the design, creation, and implementation of the IDL program **SAPLOT**, which is based on earlier versions in FORTRAN and C. Continuing use of this program has uncovered a few bugs and some enhancements that would make the program even more useful.

In addition, the program suite *DMOS* is known to have a problem that affects the calculation of reverberation time series, a problem that has existed within the code since its creation.

The current contract was let in order to solve the problem in *DMOS*, fix the **SAPLOT** bugs, and to add some new capabilities to **SAPLOT**.

The change to *DMOS* was accompanied by a change in the *DMOS* User's Guide, a document that is still under production. The fixes and enhancements to **SAPLOT** are documented in this Contractor's Report alone since that program doesn't have a current User's Guide.

Résumé

Dans le cadre de contrats précédents, nous avons conçu, élaboré et mis en œuvre le programme SAPLOT à l'aide d'anciennes versions des langages FORTRAN et C. Nous avons depuis relevé certains bogues, ainsi qu'établi des améliorations qui en augmenterait l'utilité.

Il est également connu que la suite de programmes DMOS comporte un problème sur le plan du calcul des séries de temps de réverbération, problème qui existe dans le code depuis sa création.

Le contrat actuel vise à corriger ledit problème et les bogues de SAPLOT, de même qu'à intégrer de nouvelles capacités dans celui-ci.

La modification de DMOS se reflète dans le guide de l'utilisateur connexe, un document qui est toujours en cours de production. Toutefois, les corrections et les améliorations apportées à SAPLOT ne sont consignées que dans le rapport de l'entrepreneur, car il n'existe actuellement aucun guide de l'utilisateur pour ce programme.

This page intentionally left blank.

Executive Summary

SAPLOT Maintenance and Enhancement

C. Calnan; DRDC Atlantic CR 2007-132; Defence Research and Development Canada – Atlantic; August 2007.

Introduction

The DRDC Atlantic Scientific Plotting software package, SAPLOT has been in existence in various forms for decades. The package has been designed to enable the easy production of photo-ready graphics meeting common journal standards. SAPLOT has undergone many transitions, initially having been written in Fortran, then rewritten in C, and most recently in IDL (Interactive Data Language). Each major rewrite has introduced significantly enhance capability. For example, with the earlier IDL versions, the ability to deal with images has been introduced.

Results

Previous contracts resulted in the design, creation, and implementation of the IDL program SAPLOT. Continuing use of this program has uncovered a few bugs and some enhancements that would make the program even more useful. This current contract was undertaken in order to solve graphics problems in DMOS (DRDC Model Operating System), fix the SAPLOT bugs, and to add some new capabilities to SAPLOT.

Significance

SAPLOT is able to greatly ease the burden of producing consistent photo-ready figures for the publication of scientific data. It increases the efficiency of defence scientists, technologists, and computer scientists in preparing documentation of results.

Calnan, C. 2007. *SAPLOT Maintenance and Enhancement*, DRDC Atlantic CR2007-132. Defence R&D Canada – Atlantic.

Sommaire

SAPLOT Maintenance and Enhancement

C. Calnan ; DRDC Atlantic CR 2007-132 ; Recherche et développement pour la défense Canada – Atlantique ; août 2007.

Introduction

Le progiciel de traçage scientifique SAPLOT de RDDC Atlantique existe depuis des décennies sous diverses formes. Il a été conçu pour faciliter la production de graphiques compatibles avec des photographies et conformes aux normes courantes s'appliquant aux revues scientifiques. SAPLOT a connu de nombreuses transitions; initialement écrit en Fortran, il a ensuite été réécrit en C, puis, plus récemment, en IDL (Interactive Data Language). Chaque principale réécriture en a amélioré considérablement les capacités. À titre d'exemple, il est possible de gérer des images dans les versions plus récentes reposant sur IDL.

Résultats

Dans le cadre de contrats précédents, nous avons conçu, élaboré et mis en œuvre le programme SAPLOT. Nous avons depuis relevé certains bogues, ainsi qu'établi des améliorations qui en augmenteraient l'utilité. Le présent contrat vise à corriger des problèmes graphiques touchant DMOS (DRDC Model Operating System) et des bogues dans SAPLOT, de même qu'à intégrer de nouvelles capacités dans SAPLOT.

Portée

SAPLOT facilite grandement la production de figures uniformes compatibles avec des photographies aux fins de publication de données scientifiques. Il accroît l'efficacité des scientifiques de la défense, des technologues et des informaticiens dans la préparation de documents de résultats.

Calnan, C. 2007. *SAPLOT Maintenance and Enhancement*, DRDC Atlantic CR 2007-132. Defence R&D Canada – Atlantic.

Table of Contents

List of Tables

List of Figures

List of Listings

This page intentionally left blank.

1. Introduction

Contractor Reports [1] and [2] describe the creation and enhancement of the IDL program **SAPLOT**, a general purpose "script run" program that produces publication quality graphics. Users of the program have discovered a few bugs and deficiencies in the code and have also come up with a number of desirable expansions to the program.

Another task included in the contract was tracking down the cause of an anomaly in the results of the reverberation analysis program suite *DMOS*, specifically in the normal mode reverberation program. Once found, the cause was to be examined to determine whether the anomaly was due to a coding error, a misinterpretation of the theory, etc. The steps to be taken after that would depend on the cause of the anomaly. As it turned out, the source of the problem was not a bug, but was due to the way in which bottom scattering was implemented. Since the method was valid, an option was written into the program that would allow a user to choose between the current calculation method and a newly implemented one that would "smooth" the results more. Instructions on how to choose between the options were added to the *DMOS* user's guide, a manual that is still in draft form.

This Contractor Report describes the work done with both *DMOS* and **SAPLOT** and includes, for **SAPLOT**, instructions on how to use the new capabilities and examples of their output.

Besides common English typographic conventions, the following conventions are used in this document:

- **bold text** is used for filenames (e.g. **test.pro** or **/local/files/test.pro**)
- ***bold italics text*** is used for directories (e.g. ***/usr/tmp***)
- *italics* text is used for computer and program suite names (e.g. *Tessie* and *DMOS*)
- **Bold Arial text** is used to indicate program names (e.g. **SAPLOT, MONOGO**)
- Arial text is used to indicate function and subroutine names (e.g. PPR_SETUP)
- *italic Arial text* is used for variables' names in computer programs or associated with operating systems (e.g. *IDL_PATH*)
- `Courier text` is used for text to be typed on the keyboard, **SAPLOT** commands, code or file listings, etc. (e.g. enter "`idl`")

2. DMOS

Although the contract task that involves investigating what was assumed to be a bug in the *DMOS* program **MONOGO** was the last one specified on the contract, due to its critical nature it was the first task performed. Therefore its resolution will be described before the work on **SAPLOT**.

2.1 Description of the Problem

Figure 1 presents a sample reverberation time series created by the *DMOS* program **MONOGO**.

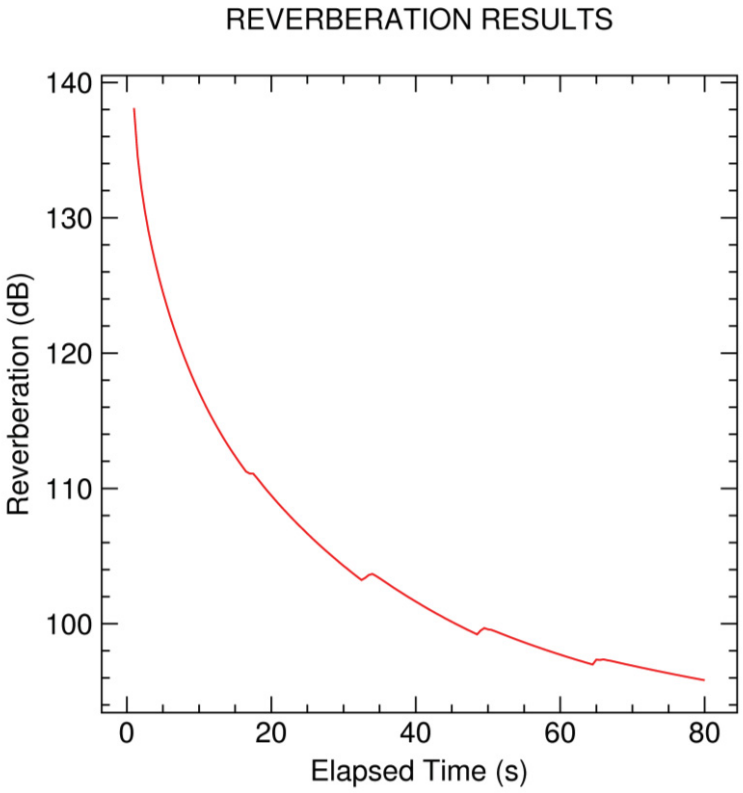

Figure 1. Original Reverberation Time Series

The data used to produce this curve were those from a BASE 04 trial, and the problem can be seen as a series of "bumps" on an otherwise smooth curve. These bumps occurred at multiples of about 16 seconds. At the assumed speed of sound, 1500 m/s, this time corresponds to a distance multiple of about 12 km, the distance step size used in setting up the data for the analysis.

2.2 Problem Cause

The measured data used for the analysis run varied over the distance of the radial for which the calculations were made. Among the parameters that varied with distance were water depth, scattering coefficients, and normal mode parameters. It was not immediately apparent which parameter or combination was responsible for the bumps on the curve.

To try and isolate the parameters responsible for the bumps a series of tests was run. By varying the parameters individually and in combination and then checking the results, it was eventually discovered that changes in the surface scattering coefficient were the cause of the bumps. Listing 1 is that of the input parameter file used by *DMOS* component **MONOGO** to calculate the reverberation time series.

Listing 1. Original MONOGO Input File

```
CAA "measured data"
      45.0000      57.5000      50.0000      90.0000
      45.0000      57.5000      50.0000      90.0000      40.0000
 GRADIENT = R1dprof.dat
Radial Information
      0.00000   centre.bin
      0.00000        -29.3     0  0  999  0  0
  -1
     12.04000   r01p02.bin
     12.04000        -28.7     0  0  999  0  0
  -1
     24.08000   r01p03.bin
     24.08000        -27.6     0  0  999  0  0
  -1
     36.12000   r01p04.bin
     36.12000        -26.8     0  0  999  0  0
  -1
     48.16000   r01p05.bin
     48.16000        -26.2     0  0  999  0  0
  -1
     60.20000   r01p06.bin
     60.20000        -25.3     0  0  999  0  0
  -1
```

In this listing the first two values on the lines under the lines containing "`.bin`" filenames are the distance in km for those lines and the Lambert surface scattering coefficients in dB. (The program actually converts these values in dB to dimensionless numbers via $10^{dB/10}$ before using them in calculations. For example the 0.0 km Lambert coefficient, -29.3, is converted to 0.001175 before being used.) It was found that the program started using a particular coefficient's value when its distance was reached and then used the same value until a new distance range and Lambert coefficient was reached. For example, the value 0.001175 was used until range calculations reached a distance of 12.04 km. At that point the value $10^{-28.7/10}$, or

0.001349, began to be used. It was the jump from one discrete value to another that was causing the bumps in the reverberation time series data.

2.3 Problem Resolution

It was determined that the current method of using the coefficients was not a bug or error, just one possible implementation of a solution. However, since other parameters were being linearly interpolated between different ranges (water depth, for example), it was decided to add code to the analysis that would cause the converted parameters to be linearly interpolated between stated values as well. Figure 2 shows the results once this interpolation technique was implemented. The figure also contains the original, "Not Interpolated" curve from Figure 1 for comparison.

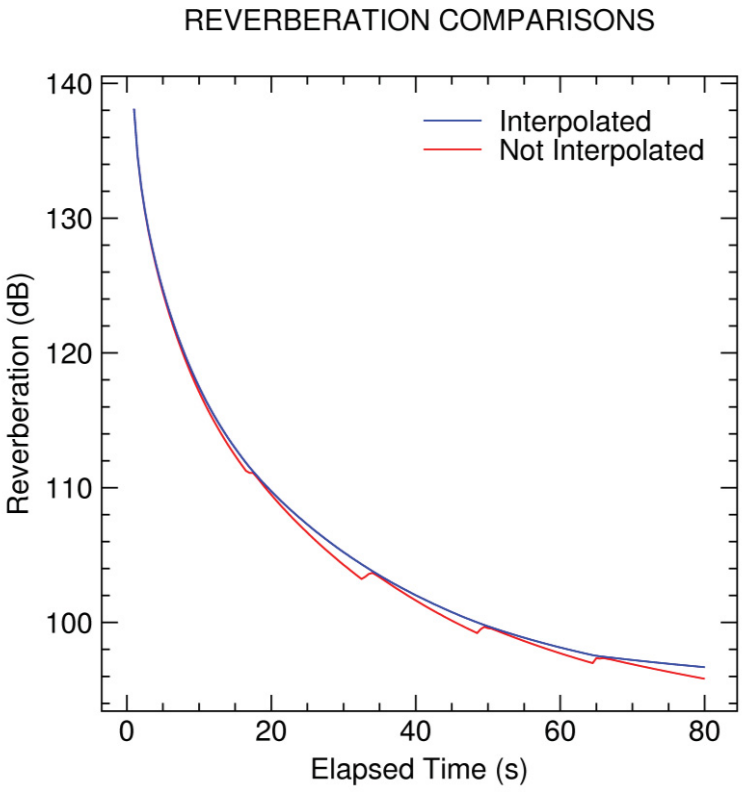

Figure 2. Reverberation Time Series Comparisons

It can be seen that the curves are closest together at times (and so ranges) where new coefficients are defined. This occurs since at these times both sets of calculations are using the same data. The interpolation scheme can also be seen to produce a smoother overall time series, something noticed in both reality and in other models' results.

Since the original scheme was not actually incorrect and it was determined that at times it might even be preferable to the interpolation scheme, it was decided to allow the program to be able to perform the calculations in both ways, depending on a user's wishes. Accordingly, the program that reads the Listing 1 files was modified so that it could be told to produce interpolated results as a special case. The original method would be kept as the default in case old input files ever had to be rerun and their original results were desired.

Listing 2 is a copy of the first eight lines of Listing 1 with one change, highlighted in bold text on line five.

Listing 2. MONOGO Input File with the Interpolation Option

```
CAA "measured data"
45.0000      57.5000      50.0000      90.0000
      45.0000      57.5000      50.0000      90.0000      40.0000
 GRADIENT = R1dprof.dat
Radial Information   lambert interp
      0.00000   centre.bin
      0.00000        -29.3     0  0  999  0  0
  -1
```

Users can opt to use Lambert surface scattering linear interpolation by including the text string "`Lambert Interp`" on the "`Radial Information`" line. This string is converted to upper case before processing so the case of the line is irrelevant, but there must be one only space between "`lambert`" and "`interp`". If either word is misspelled or there is not one space between them, the string will not be interpreted properly and the interpolation will not be performed.

2.4 Bottom Scattering Option

While the above investigation was being performed it was realized that although users could select a surface scattering method, there was not an equivalent option for bottom scattering. By default both are calculated using the Lambert scattering model, but users could opt to have Chapman-Harris scattering calculated at the surface. This is accomplished by entering "`SURFACE SCATTERING = CHAP`" anywhere after line three and before the "`RADIAL INFORMATION`" line.

It was decided that the program should also allow users the option of using the Chapman-Harris model for bottom scattering as well. Accordingly, the program now allows this by recognizing the string "`BOTTOM SCATTERING = CHAP`" when it is located between line three and the "`RADIAL INFORMATION`" line.

The same rules apply for both surface and bottom scattering instructions:

- The complete strings "`SURFACE SCATTERING`" or "`BOTTOM SCATTERING`" must appear with those exact spelling and one space between the two words.
- The two-word phrase must be followed by "=", although any number of spaces may be between the words and the "=".
- The "=" must be followed by "`CHAP`", with that exact spelling, although any number of spaces may be between "=" and "`CHAP`".
- The text strings may be in any combination of upper or lower case letters since the lines are all converted to upper case before processing.
- Lambert scattering may be explicitly chosen by entering either "`SURFACE SCATTERING = LAMBERT`" or "`BOTTOM SCATTERING = LAMBERT`", depending on where the scattering is desired.

2.5 User Guide Update

The *DMOS* User's Guide, which is unpublished as this Contractor's Report is being written, has been updated to include instructions on how to perform Lambert surface scattering interpolation and how to calculate bottom scattering via the Chapman-Harris model.

3. SAPLOT

SAPLOT is usually run by giving it the name of an input file. This file contains data to be plotted as well as instructions on the size and location of the plot, labels, legends, the colour and type of lines used to "connect the dots" of a data line, and so on. Each different type of data or instruction is accompanied by what is referred to in this and other **SAPLOT** reports as a "command." The fact that **SAPLOT** input files contain much more than just data to be plotted makes the files, in effect, programs used to run **SAPLOT**. Because of that, this report refers to **SAPLOT** input files as "scripts" in the same manner that interpreted languages such as Perl and Python are run by programs called scripts.

The main purpose of the current contract was for maintenance and expansion of **SAPLOT**. The contract listed eight tasks, which were then sorted into the following list by order of priority:

1. Fix known bugs and a problem with the "help" file.
2. Add the capability to have images inserted into plots.
3. Add the capability for "external" files, where the script can name files external to itself.
4. Add the capability to plot polar data.
5. Add the capability that would allow non-ASCII characters in plotted text.
6. Remove code relating to Encapsulated PostScript (EPS) files.
7. Allow the definition of values or maximum inter-value deltas that could be used to permit gaps in data, and define a date/time format for `CURVE` data.
8. Ensure that the code that works properly under the development system, which is command line driven under Linux, also works under the IDL Linux and Windows IDE.

Due to time constraints only the first four tasks on the list were completed. This was largely due to the complexity involved in tasks 2, 3, and 4. As it turned out there were a number of difficulties to be overcome in getting images plotted, and the writing of the code for polar plots required incorporating data from a large number of other **SAPLOT** commands. The four tasks that were completed are discussed in Sections 3.1 and 3.2, while the remaining four are discussed in Section 3.3.

In this section several notational conventions are used when **SAPLOT** commands are described. Since these descriptions are written to mimic the format of the existing **SAPLOT** user's guides [3] and [4], these conventions are mainly copied from those documents.

The following sample user's guide usage line helps describe the conventions:

```
CMD <a value> b <"X"|"Y"> [c,[d,[e]]]
```

In this sample:

- The two parameters "`a value`" and `b` are mandatory; angle brackets may be used if a parameter's summary consists of more than one word.
- Square brackets ("`[`" and "`]`") indicate optional parameters.
- A pipe ("`|`") indicates the "OR" condition. In the above example one of the literal strings `X` or `Y` must be present, as indicated by the angle brackets. Quotes may be optional; the write-up on command parameters will indicate if the quotes are needed.
- Parameters `c`, `d`, and `e` are all optional.
- Parameter `c` may appear by itself.
- Parameter `d` can only be provided if a value is also given for the optional parameter `c`, which must precede it.
- Parameter `e` can only be provided if values are also given for the optional parameters `c` and `d`, which must precede it.

3.1 Maintenance

For the current contract, maintenance involved tracking down and fixing bugs discovered by users and making some changes in the way that several commands were executed.

3.1.1 Bug Fixes

The following bugs were found by users:

- The `XLABEL` command did not work correctly with a log *x* axis.

  In tracking down this error it was discovered that the `XLABEL` command only did work correctly with linear *x* and *y* axes, and didn't work for $\log_{10}$, $\log_2$, or probability axes. The `XLABEL` command now works correctly.

- Probability axis data were plotted incorrectly.

  In fact, it was discovered that everything about probability plotting was incorrect, but only slightly so. This was only realized after a detailed comparison of plotted **GSM** output was made with plots of the same results in the **GSM** manual. The problem was then tracked down and fixed.

- If there were more than six lines on a plot, all the lines beyond the sixth one were plotted with the same line type – a solid line.

  A minor typo caused this error. Once fixed, the line types now cycle through the six line types that IDL contains.

- A maximum of about 32,000 data points was allowed on a curve.

  Curve data are read from the script and stored in arrays. Before plotting, the data are moved in part by way of an integer counter that loops through the values. An oversight allowed the creation of the integer counter as a "small" integer (i.e. a two-byte integer), which has a maximum value of 32,767. The counter was changed to a "long" four-byte integer, which has a maximum value of over 2.1×10^9, and the problem was cleared up. However, if more than that number of points are desired to be plotted in a single curve, then any data points with indices above that number will be lost. Should this happen, a new tack will have to be taken with reading, storing, and plotting the data.

3.1.2 Other Changes

There were four other changes made under the maintenance umbrella. The first of these was listed under task 1, but others changes were made when work on **SAPLOT** uncovered problems with them. These changes are listed in this section.

1. Help File Display

The problem with the "help" file **idl-saplot.help** was that it could only be accessed if it was located in the directory from which IDL was started. This restriction was overly confining, especially since that file was normally located in the directory that holds the code. The restriction was removed by making **SAPLOT** look for the help file in the current directory, and then in all the directories contained in its *!PATH* variable. Once a file with the correct name is found the program prints it out and aborts the search.

2. `LABEL` and `LEGEND`

These commands cause, respectively, one or a specified number of lines of data to be read and written to the plot. These lines of data will be used as they appear, which means that the command `EXTERNAL` (Section 3.2.2), comment lines, or inline comments will all be written to the plot as they were entered. That is, these lines' actual purposes are ignored. Blank labels and legend lines can be created by leaving the appropriate lines without text.

Inline comments are not allowed in `LABEL` or `LEGEND` lines.

3. LEGEND

This command must now be followed by an integer giving the number of legend lines to follow. If there are more lines of LEGEND data than plotted curves, then the LEGEND lines greater than the number of curves will be written as entered, but the text won't have a sample curve line in front of them.

4. PATPLT and PTPLT

Formerly only the last pattern defined for a page would be plotted, but now all of them are. It must be remembered that all plottable data (CURVE, PATPLT, PTPLT, POLAR, IMAGE) are stacked and plotted in the order of last defined to first defined. This results in the first specified plottable data being on top of all the others, should they overlap.

3.2 Enhancement

The **SAPLOT** enhancement resulted in the addition of three new capabilities to the program under four new commands, capabilities that didn't exist in any known previous versions of the program. This section presents the information on those new capabilities under their command names.

3.2.1 IMAGE

This command allows for any number of images to be drawn on a plot. The first image listed for a plot will be the last one drawn and so will be the topmost image in case of overlap. The last image listed for a plot will be the first one drawn and so will be overlain by all other images in case of overlap. Images can not be drawn on polar plots, but **SAPLOT** allows new plots to be created (the PLOT command) and set at the same location. One use of the IMAGE command can allow any number of images to be placed on the same plot, with the end of image data being the appearance of a line that does not start with either a single or double quotation mark.

Usage:

```
IMAGE
"fname" Xo Yo code <code related data> [Xop Xep Yop Yep]
'fname' Xo Yo code <code related data> [Xop Xep Yop Yep]
```

where:

"fname" 'fname'	-	is the name of an image file. Since filenames may contain blank spaces, the filename must be enclosed in either single or double quotes. Filenames may not contain the character "!".

IDL determines the type of the image data by `fname`'s extension, but if reading the file under that assumption causes an error to occur, IDL will check the contents of the file. If the file contains data that do not correspond to an image type recognized by IDL, then **SAPLOT** will write an error message and skip the line containing the image data.

`Xo` - is the *x* position of the image's lower left display position in the plot's data units.

`Yo` - is the *y* position of the image's lower left display position in the plot's data units.

`code` - is a code value from 1 to 6. Its value determines what code related data **SAPLOT** will try to read from the line.

`code related data` - The contents of this data type depends on the value of `code` in the following way:

`code`	contents	description
1	`Xlen`	width of the plotted image along the *x* axis in the plot's data units
2	`Ylen`	height of the plotted image along the *y* axis in the plot's data units
3	`Xlen`, `Ylen`	width and height of the plotted image along the *x* and *y* axes in the plot's data units
4	`Xpos`	*x* location of the plotted image's right side in the plot's data units
5	`Ypos`	*y* location of the top of the plotted image in the plot's data units
6	`Xpos`, `Ypos`	location of the upper right corner of the plotted image in the plot's data units

`code` values 1, 2, 4, and 5 will cause the other length or position to be calculated in data units for an image that maintains its aspect ratio. `code` values 3 and 6 allow users to dispense with the image's aspect ratio; images can be squished along either the *x* or *y* axes.

The axis lengths for `code` values 1, 2, and 3 can be negative. When they are this causes the images to be plotted in the opposite direction to the axes' numbering.

`Xop` `Xep` `Yop` `Yep` - A user must include either all or none of these values.

These four values are the *x* and *y* origin and end display pixels based on the original image before scaling or any type of

clipping. These values allow a subset of the image to be selected for display such that the stated pixel numbers are matched with the image positioning/sizing data that precede them. That is, pixel `Xop` occurs at position `Xorig`, pixel `Yep` is plotted at position `Ypos`, and so on. Therefore these values contain clipping information for the image. Position `Xorig` will be given to pixel `Xep`, and the value for `Xlen` will be based on `Xop` and `Xep` minus `Xop`.

Pixel (1,1) is the lower left pixel of an image, and pixel (`Xop`,`Yop`) is the lower left pixel to be displayed (unless it is clipped because it's outside the data window).

All four values may be omitted if the entire image is to be displayed, and if any of the values are less than 1, the limiting value for that item is used. That is, if `Xop` and/or `Yop` is less than 1, they are reassigned to be pixel 1, and if `Xep` and/or `Yep` are less than 1, they are reassigned to the number of pixels in that direction.

If axes' ranges are specified via the `RANGE` command and image positions are specified that cause the image to be drawn outside the defined data box, the images will be clipped.

Listing 3 presents a sample **SAPLOT** script that uses the `IMAGE` command to plot the following three images found on the Internet:

- **Atlantis.png** - a 1150 by 863 pixel PNG file
- **Quest.GIF** - a 100 by 55 pixel GIF file, and
- **quest.jpg** - a 753 by 481 pixel JPEG file.

The script also plots two `CURVE`s in order to help demonstrate the way in which plottable data are "stacked."

Listing 3. SAPLOT Script with the Command `IMAGE`

```
linclr 3        ! red

CURVE
  3.0  -3.4
  4.0   2.3
  5.0   3.3
  6.0   0.5
  7.7   9.1
  9.2   3.2
 15.3  -1.0
```

```
IMAGE
 "Atlantis.png"   4.0  -2.0   3  -9.  5. 575 800  1 300
 "Quest.GIF"      3.0   1.0   1   6.
 "quest.jpg"      7.0   7.0   6  15.  3.

linclr 4       ! green

CURVE
  4.0  -3.9
  5.0   2.8
  6.0   3.6
  7.0   0.1
  8.7   9.6
 10.2   3.9
 14.3  -1.9
```

Figure 3 is the full size graphic result of using the Listing 2 script with **SAPLOT**.

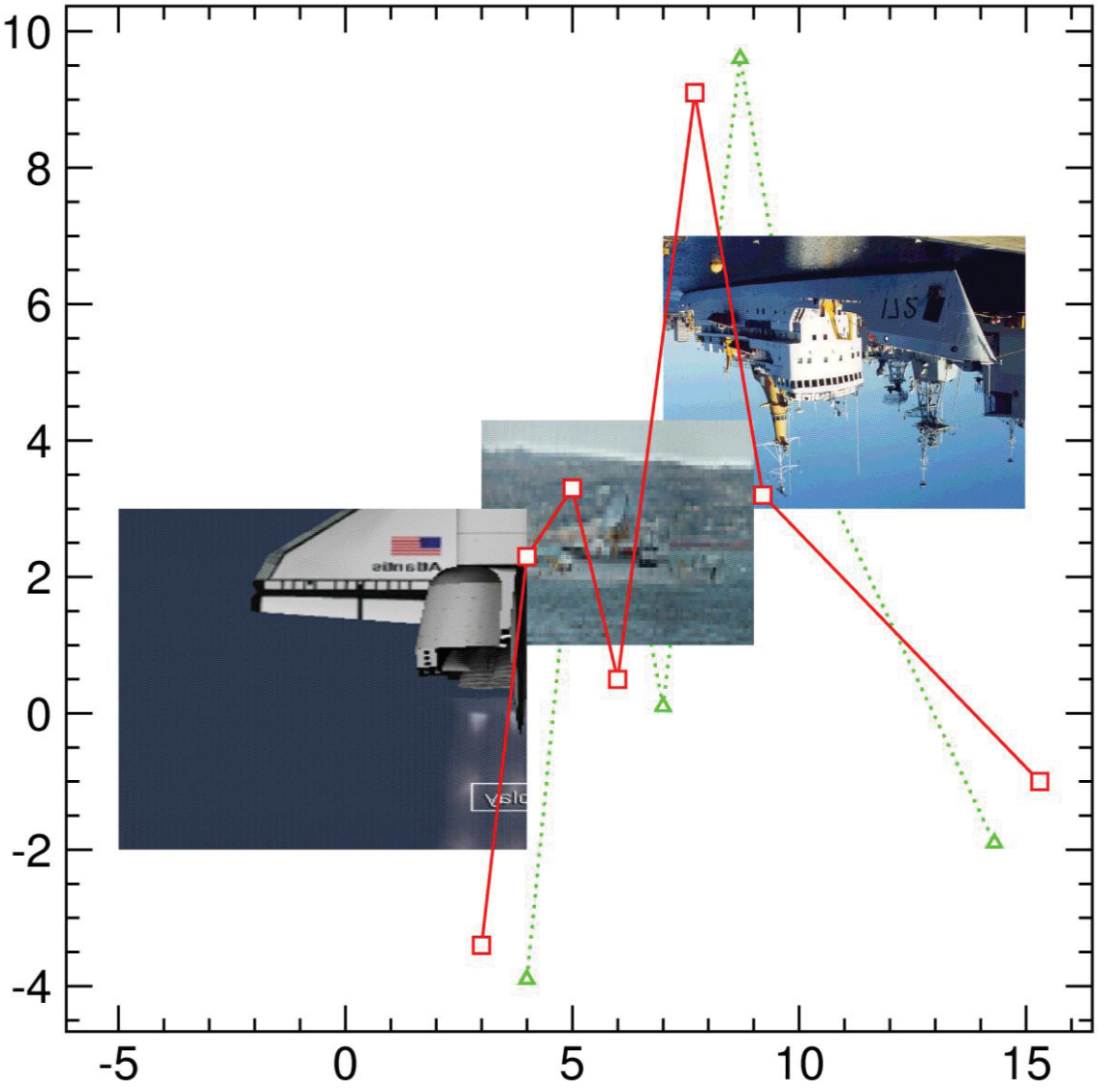

Figure 3. SAPLOT Results from the Command `IMAGE`

3.2.2 `EXTERNAL`

This command is used in **SAPLOT** scripts to indicate that the script is to continue in a separate file external to the current script. When it encounters this command **SAPLOT** simply shifts its "input mode" to the new file; when that file comes to an end **SAPLOT** closes the file and returns to the file that named the external file.

There is no limit to the depth of chained external files, but **SAPLOT** does keep track of the names of those files, only dropping the names once the files have been closed. If the program encounters an external file name that is already on the current chain of external files, an error message is written out and that particular EXTERNAL command is ignored. This is done to prevent an infinite loop in reading files. However a specific external file can legally be used any number of times as long as it is not contained in a circular list.

Usage: `EXTERNAL <"fname"|'fname'>`

where:

`"fname"` - is the name of an external file. Since filenames may contain
`'fname'` blank spaces, the filename must be enclosed in either single or double quotes. Filenames may not contain the character "`!`".

If **SAPLOT** can't open this file then an error message is written and the line containing that EXTERNAL command is skipped.

EXTERNAL may appear in the middle of a series of blank lines or comments, or following the completion of another command. In these cases the first data line in the EXTERNAL's file must be a valid **SAPLOT** command.

The command may also appear within certain data types as well. In this case the first line of the file may consist of:

- A blank line, comment line, or a **SAPLOT** command in order to indicate the end of the data.
- More data, to continue that of the "calling" script.
- The **SAPLOT** command EXTERNAL to move the input flow to another external **SAPLOT** script. The file so named will be treated the same as the file that named it, and so the rules of this paragraph apply to it as well.

The types of data that can continue within an EXTERNAL data file are:

- CURVE
- PATPLT
- PTPLT
- POLAR

An EXTERNAL data line will be treated as data and written on the plot for the **SAPLOT** commands:

- LABEL
- LEGEND
- PATLEGEND
- PTLEGEND

Listings 4 through 8 present a series of **SAPLOT** scripts that illustrate how EXTERNAL may be used. Following these scripts is Figure 4, which presents the results of using these scripts.

Listing 4. Main SAPLOT Script for a Test of the Command EXTERNAL

```
!  This is the main file for a test of EXTERNAL.
!  File "ext4.data" references "ext5.dat"

external "ext2.dat"

linclr 1   ! black

curve
external  "ext3.dat"

linclr 3   ! red

curve
10   5
20   8
external   "ext4.dat"
70  26
80  28
90  30
100 32
110 34
120 36
```

Listing 5. EXTERNAL Test File ext2.dat

```
!  This file is referenced from "ext1.sap"

label  1
X axis

linclr 4    ! green

curve
10   2
20   4
30   6
40   8
50  10
60  12
70  14
80  16
```

```
90  18

label 2
Y axis
```

Listing 6. `EXTERNAL` Test File ext3.dat

```
10   4
20   7
30  10
40  13
50  16
60  19
70  22
80  25
90  28
100 31

label 3
what's up?
```

Listing 7. `EXTERNAL` Test File ext4.dat

```
30  11
40  14
external 'ext5.dat'
80  26
90  29
100 32
110 35

label 7
Anybody know

linclr 5   ! Blue

curve
10  14
20  16
30  18
40  20
50  22
60  24
```

Listing 8. `EXTERNAL` Test File ext5.dat

```
50   17
60   20
70   23
```

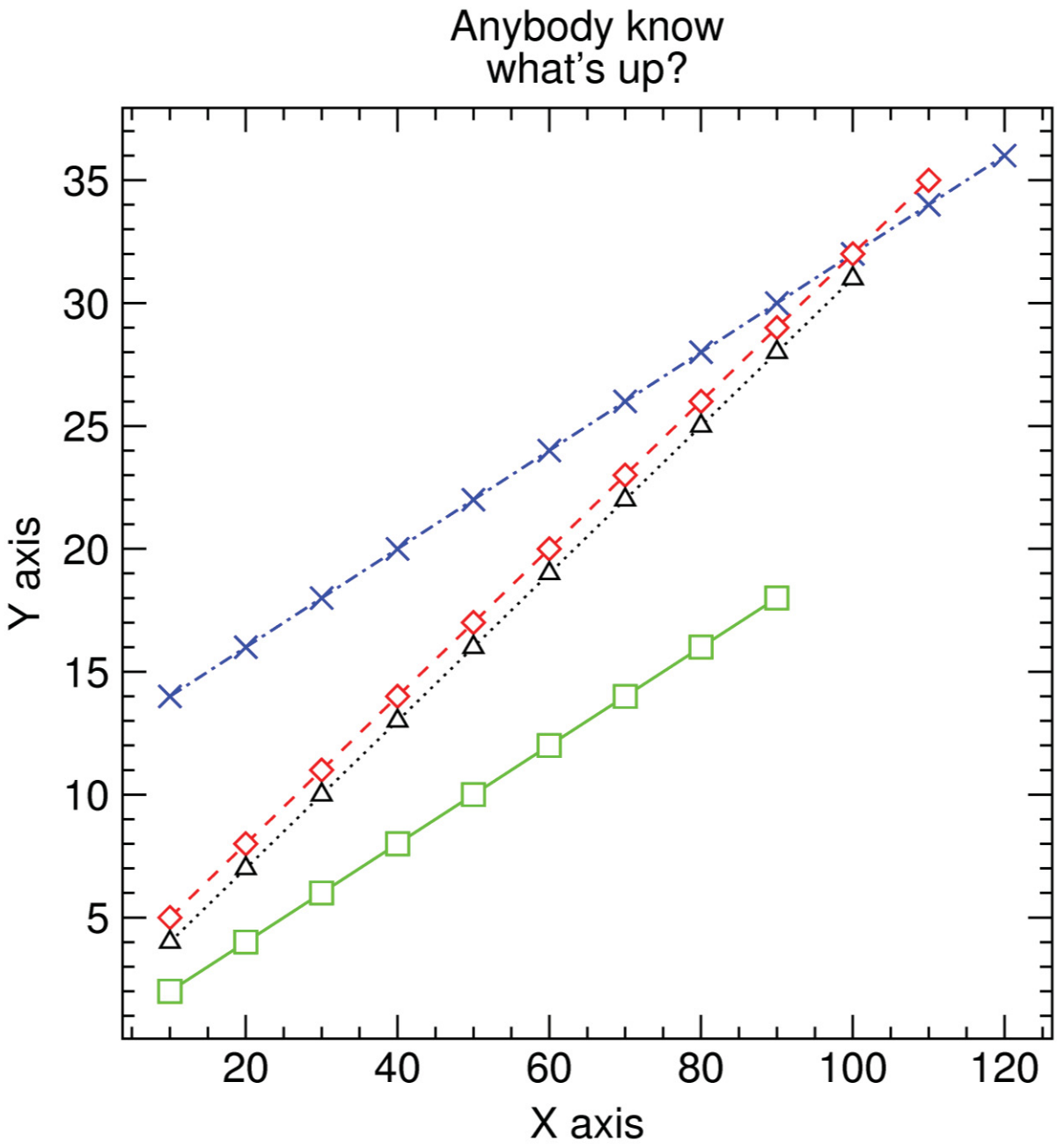

Figure 4. Plot Resulting from `EXTERNAL` Test

3.2.3 POLAR

The POLAR command is analogous to the CURVE command, except that where CURVE indicates that a series of (x,y) values follows the command, POLAR indicates that a series of (r,θ) values is to follow. As well, instead of plotting the data on a rectangular grid as CURVE does, POLAR causes the data to be plotted on a circular "grid."

Usage:

```
POLAR  [D|R]
 distance1     angle1
 distance2     angle2
 distance3     angle3
```

where:

D	-	indicates that the angles are in degrees and may be given as either d or D. This is the default if there is nothing following the command POLAR on the line.
R	-	indicates that the angles are in radians and may be given as either r or R.
distance*N*	-	is a distance in data units of the user's choice.
angle*N*	-	is an angle in the units specified or defaulted to on the POLAR command line.

A number of commands that set CURVE plot parameters are also used for POLAR plots, although some of them have different limitations and defaults. In those that allow axis numbers, only axis numbers 1 and 2 (distance and angle, respectively) are allowed. All axis 2 specifications must be in degrees regardless of the angles' units as specified on the line of the POLAR command. Specifying axis 3 or 4 (normally the top and right axes, respectively) for a POLAR plot will result in an error. The following commands affect POLAR data plots:

ADJUST	-	Normal usage.
ASET	-	Axis 1 describes the radial axis's type and can have values 1 through 4 (default = 1) Axis 2 describes the grid's orientation and can have the values: 1 – True (0º up, angles increasing clockwise; default) 2 – Cartesian (0º to the right, angles increasing counter-clockwise)
AXSET	-	Two values must be entered; the first will be used as the polar plot's diameter and the second will be ignored
CSET	-	Normal usage, but only curve types 0, 1, and 2 are allowed
DENSIT	-	Normal usage.
EXCH	-	Normal usage.

FORMAT - Normal usage, except that the centre of the POLAR circle will be located at the centre of the defined plot and its diameter will be the shortest side of the rectangle defined by its FORMAT value (e.g. FORMAT 2, which has the *x* origin at 2.5, the *y* origin at 1.5, an *x* axis length of 7.0, and a *y* axis length of 5.5 will result in a POLAR diagram with the centre at (6.00,4.25) and a diameter of 5.5)

FSET - Normal usage.

GRID - Normal usage. The default values for POLAR plots are:

```
GRID 1 0 -1
GRID 2 0 -1
```

which results in solid grid lines at major tick marks and none at the minor tick marks. This differs from the CURVE plot default, which is to have no grid lines drawn at all.

LINCLR - Normal usage.

LOG10 - Normal usage.

LSET - Normal usage.

MSET - Normal usage.

NCLIP - Normal usage. See **A Note Concerning Data Clipping:**, which follows this list of commands.

NSET - Axis 1: ignores this value and uses the PRSCALE data instead.
Axis 2: normal usage.

ORSET - Normal usage, except that the plot's origin is at its centre instead of the lower left.

RANGE - Axis 1: the values are the radial values at the centre and outer radius of the diagram
Axis 2: ignored

SSET - Axis 1: specifications are for circular lines at a constant radius.
Axis 2: specifications are for radial "spokes" and must be in degrees.

TSET - Normal usage.

WSET - Normal usage.

A Note Concerning Data Clipping:

IDL can perform data clipping by itself, which it does by defining a rectangular data window with sides parallel to the horizontal and vertical borders of the display and then only plotting data inside this area. **SAPLOT** produces polar plots by defining a square data window and then drawing a circular "bulls-eye" that touches the square data window at the centres of its sides. Data are then converted from (r,θ) to a local (x,y) and plotted inside this circle. However, there remain four areas at the corners of the square data window that are outside the bulls-eye. When polar data are to be clipped they should ideally stop at the outer edge of the bulls-eye. However IDL's built in clipping can't use anything but a rectangular clipping window and so can't clip polar plots properly. Therefore a software solution for this must be written into **SAPLOT**; unfortunately there was insufficient time in this contract to implement this solution.

Possible Problems with Related Commands:

Due to time constraints a number of plotting commands were not able to be worked into operation in conjunction with the POLAR command. Use of these commands will probably not produce exactly what a user desires, but in all cases there are alternate commands that can do essentially the same work as the dysfunctional commands. The commands that were not tested in detail or at all and probably will not work properly on a POLAR plot are: LABEL, XLABEL, XLSET, PVSET, THSET, and possibly other label or legend family commands.

Listing 9 presents the simplest script that can produce a POLAR plot. Following the listing is Figure 5, which shows the plot produced using all the defaults.

Listing 9. Simplest SAPLOT Script for the Command POLAR

```
polar
15    20
20    40
25    60
30    80
35   100
40   120
45   140
50   160
55   180
60   200
65   220
70   240
75   260
80   280
85   300
90   320
95   340
100  360
105   20
110   40
115   60
120   80
```

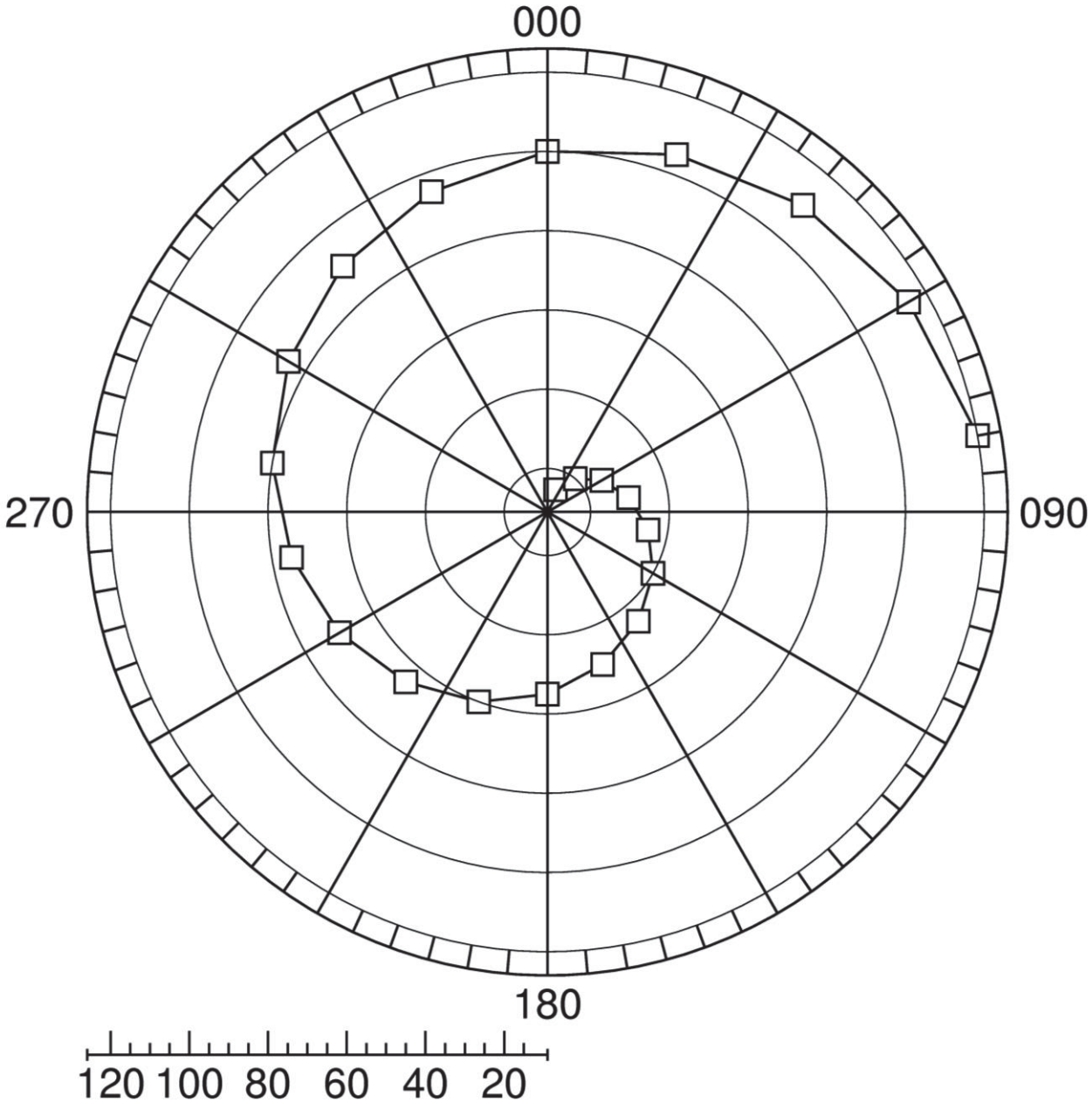

Figure 5. SAPLOT Plot from the Simplest POLAR Script

Listing 10 presents a sample POLAR plot script that contains a number of settings described above to show how they can affect the results. Following the listing is the resulting plot, Figure 6.

Listing 10. SAPLOT Script for POLAR Using Extra Commands

```
!
!  POLAR plus settings

FORMAT 3
AXSET 4.5 2
ASET 2 2               ! Cartesian
RANGE 1 -10  130
Prscale 6

grid 1 0 -1      ! Circles
grid 2 0  2      ! Spokes

sset 2 0  30  3

linclr   5       ! blue

label 1
Slightly more than

label 5
a basic POLAR test

polar
15    20
20    40
25    60
30    80
35   100
40   120
45   140
50   160
55   180
60   200
65   220
70   240
75   260
80   280
85   300
90   320
95   340
100  360
105   20
110   40
115   60
120   80
```

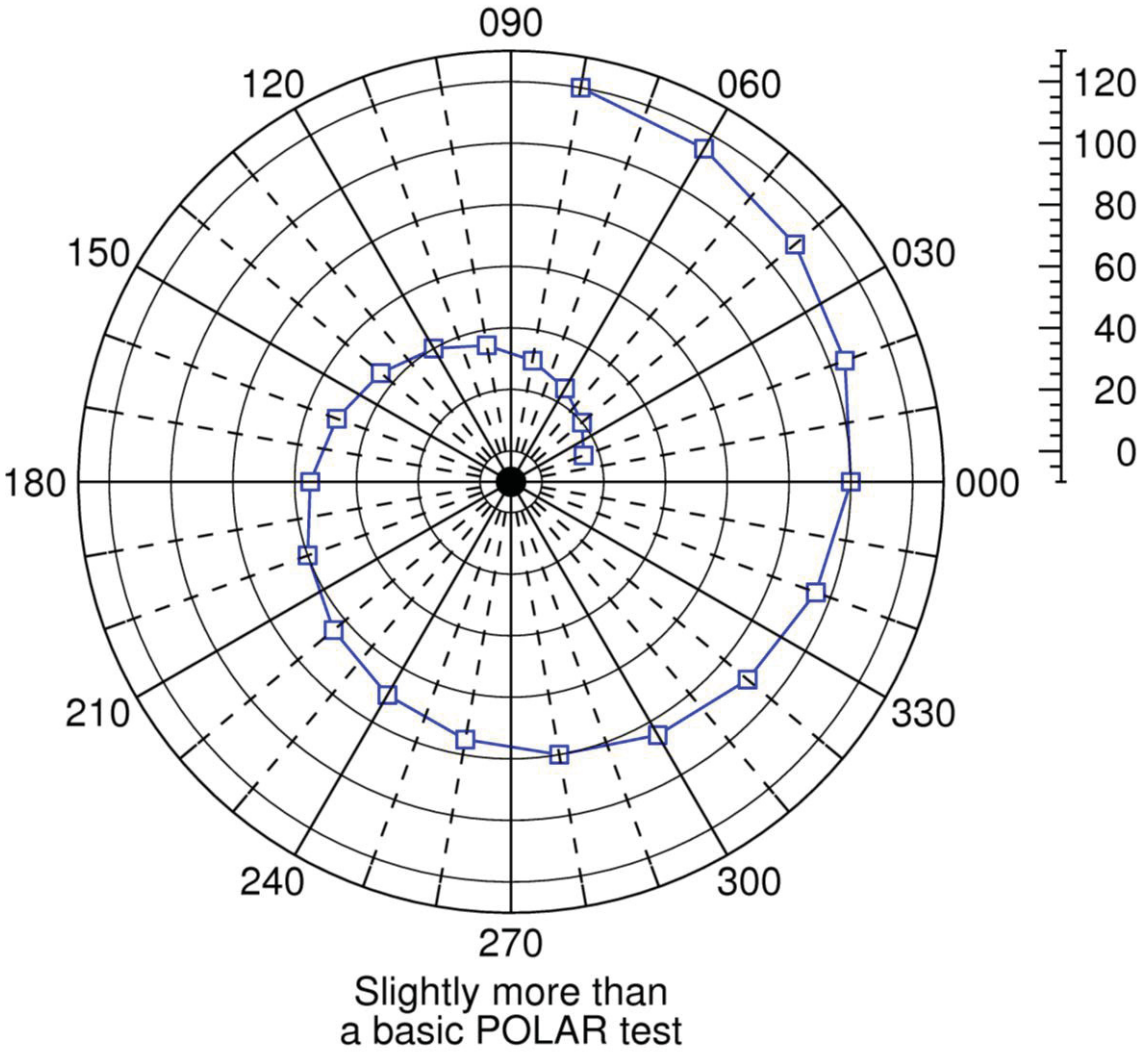

Figure 6. SAPLOT Plot from a POLAR Script with Options

3.2.4 PRSCALE

This command is used to position the polar radius scale on the plot. This scale is plotted as an alternative to drawing a numbered axis through the middle of the polar plot, where it was felt that a scale drawn outside the plot proper would provide for a much less cluttered diagram.

Usage: `PRSCALE location`

where:

`location` - specifies the location of the scale with respect to the plot itself. The valid values are:

- `0` - no radial scale will be drawn
- `1` - below the plot, left half (default if `PRSCALE` is not used)
- `2` - to the left of the plot, bottom half
- `3` - to the left of the plot, top half
- `4` - above the plot, left half
- `5` - above the plot, right half
- `6` - to the right of the plot, top half
- `7` - to the right of the plot, bottom half
- `8` - below the plot, right half

Figure 7 indicates the `PRSCALE` positions of several of the above values as a guide to its use.

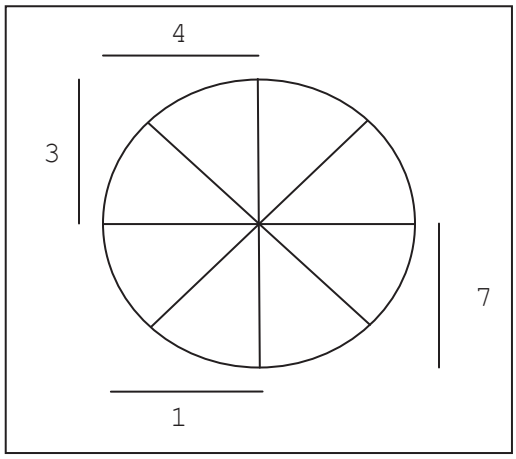

Figure 7. Positions of Selected `PRSCALE` Scales

3.4 SAPLOT Plotting Rules

During the work that resulted in Contractor Reports [1] and [2], a set of generalised rules for how **SAPLOT** produces plots has evolved. For the most part these rules were determined from examinations of the C code of a former version of the program, but one or two other rules were arbitrarily decided upon during the writing of the IDL version. For example, since the previous version did not produce screen output, it had no rules about that type of output. These rules were first explicitly stated in Section 2.4 of [2], but because of their importance it was thought worth while to reproduce them here. Users who are unaware of these rules may not always get exactly what is desired from the program.

The terminology used here is that of **SAPLOT**. For the sake of clarity several of the most potentially confusing terms will be explicitly defined:

page - By default the entire script is plotted on one piece of paper or screen, defined as a "page." The command `NEWPAGE` may be placed in the script in order to move plotting operations to a new piece of paper in PostScript output or to a refreshed and redrawn screen plot for screen output. Screen output will only move to the next page at the request of users.

plot - A page may contain any number of "plots", where a "plot" is a grouping of elements that share common parameters (data, character height, line thickness, etc.). A new plot is declared by way of the command `PLOT`.

curve - A "curve" is a series of data points that are to be joined by a line, indicated by the placement of symbols, drawn as histogram bars, etc. These data can be indicated in a **SAPLOT** script by the commands `CURVE` or `POLAR`.

The following plotting rules refer to the way in which **SAPLOT** actually produces plots, which is only done after almost all the script data have been read in and verified. The only exception is for `IMAGE` data: **SAPLOT** reads in the image filenames and verifies that they exist, but doesn't actually read in the images themselves until it is time to plot them. This is done to reduce memory requirements.

The plotting rules are:

1. The program runs an output device loop, first drawing all the data to the screen, if required, and then drawing all the data to the PostScript output file, if requested.

2. Inside the output device loop, the program runs a page loop, drawing one page at a time until all have been plotted.

3. Inside the page loop the program runs a plot loop, drawing all the plots on a page until the page is complete.

4. A plot is drawn in the following order:
   - plottable data: all the `CURVE`, `POLAR`, `IMAGE`, `PTPLT`, or `PATPLT` data, with the exception that a plot with `POLAR` data may not contain any other sort of plottable data
   - the databox is drawn; it is outlined and then axes are drawn, labelled, and tick marked – if requested by the script
   - all `LABEL`s are drawn in the order in which they appear in the script
   - all `LEGEND`s, `PTLEGEND`s, and `PATLEGEND`s are drawn in order of appearance

5. Plottable data require special mention. These items are plotted in reverse order of their appearance in the script. That is done because it is assumed that the first one encountered is the "most important" one and in plotting data in reverse order of

appearance, the first one encountered is plotted on top of all the others. Consequently, the last plottable item encountered on a page is plotted first, and all the others are plotted on top of it. The original C and FORTRAN **SAPLOT** plots `PATPLT`/`PTPLT` data first and then plots all `CURVE` data on top of it. If users desire, this can be accomplished in the IDL version by ensuring that the `PATPLT`/`PTPLT` data are the last plottable data on a plot in the script.

3.5 Suggestions for Further Work

Due to time constraints several **SAPLOT**-related tasks were not completed. These are tasks five through eight as listed in the introduction to Section 3:

5. Add the capability that would allow non-ASCII characters in plotted text.

6. Remove code relating to Encapsulated PostScript (EPS) files.

7. Allow the definition of values or maximum inter-value deltas that could be used to permit gaps in data, and define a date/time format for `CURVE` data.

8. Ensure that the code that works properly under the development system, which is command line driven under Linux, also works under the IDL Linux and Windows IDE.

These tasks are obvious candidates for completion, with the follow notes as to expected difficulty for each:

5. Only preliminary work was done for this task, and this involved ensuring that the fonts available in IDL contained at a minimum the extra characters that were able to be drawn in the older FORTRAN and C versions of **SAPLOT**. It's not known exactly how font changing would be implemented, but it appears that text strings may have to be plotted as groups of characters in the same font. When a string contains several font changes the placement of this overall string may be tricky to calculate since the lengths of each font grouping will apparently have to be computed individually.

6. All EPS code has been either commented out or otherwise removed from being used by **SAPLOT**. Therefore removing this code is a simple task that only involves finding the code and deleting it.

7. Some thought has been given to these problems during the current contract since they involve breaking plottable arrays of data into subsets. It was this technique that was considered in the problem of clipping `POLAR` data that went outside the circular grid but remained in the corners of the IDL clipping window. Implementation is not expected to be difficult.

8. Testing **SAPLOT** under the Windows and Linux IDEs will not be difficult, any difficulties involved would depend on what the test results are. Few to no problems are expected, however the tests have not been performed yet so this estimate is only a guess.

Other possible future work related to this program includes:

- **SAPLOT** only has 6 line types available to it, while the FORTRAN and C versions have 17 predefined line types. The IDL documentation could be checked and tests made to see if there is any way around this limitation. However since the older versions of **SAPLOT** "drew" directly to a PostScript file while the current version is filtered through IDL, it may not be possible (or at least simple) to expand the repertoire of line types.

- The data clipping mentioned for `POLAR` data in Section 3.2.3 and related to Task 7 above have the potential of solving another problem: clipping data in rotated (x,y) plots. The problem is that whereas IDL clips data based on a rectangular window in which data are plotted, **SAPLOT** allows plots to be rotated by arbitrary angles. This means that the two points that define an IDL clipping window will still mark the opposite corners of a rotated **SAPLOT** data window, but the other corners of the two windows won't coincide, as seen in Figure 8.

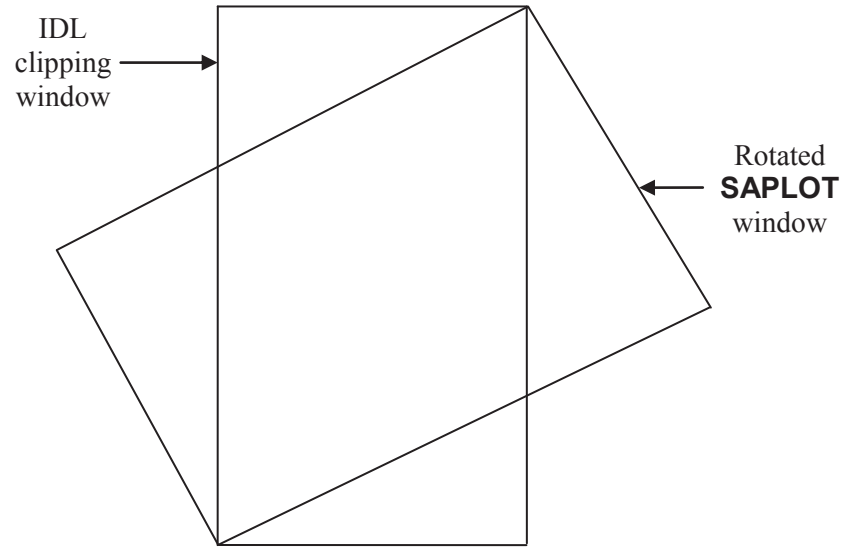

Figure 8. SAPLOT/IDL Clipping Window Problem

What happens when data are outside a rotated **SAPLOT** window and clipping is requested is that some data that a user wants to be clipped (above and below the rotated window) won't be and some data that shouldn't be clipped (to the left and right of the IDL clipping window) will not be plotted. The technique considered for clipping `POLAR` data examines each point and only plots data points within the **SAPLOT** data window. When consecutive points are inside and outside the **SAPLOT** data window an interpolated value is found on the window's edge and that point will be used for plotting.

- There has never been a **SAPLOT** user's guide written for the IDL version of the program. Up to the current time the manuals of [1] and [2] could be used with little problem, however the current version of **SAPLOT** contains capabilities beyond what the older versions possessed, and so users of those guides would be unaware of these capabilities. A user's guide should be written for the current version of **SAPLOT**.

- Either in addition or as an alternative to a hardcopy **SAPLOT** user's guide, an online version could be written. This could be done either as straightforward text information, similar to UNIX/Linux `man` pages, or through a web browser with hyperlinked HTML information files. This latter form allows a much wider range of options than a plain text file, sample code and plots, for example.

4. Program File Locations

This section provides the locations of the main programs used in the course of this project.

DMOS

The current versions of the *DMOS* executables are located on *Spray* and *Pinta* in the directory ***/home/local/models/dmos/bin***, and on *Tessie* in the directory ***~calnan/projects/RevInv/dmos/bin***. On *Tessie*, however, the executables will be moved to ***/local/models/DMOS/bin*** once sufficient testing has been performed and the author and Scientific Authority are satisfied that *DMOS* is working properly.

Most *DMOS* executables were produced by the GNU **g77** compiler for Intel-based Linux, but if a user needs to compile the programs for a different platform the programs' source code is located in directories near the executables, with each program's code in a separate directory named after the program. The program **BellhopDRDC_S** is a Fortran 90/95 routine that may be compiled with the **gfortran** compiler. **BellhopDMOS**, however, must be compiled with **g95** due to binary file format considerations.

Speciation has also occurred with some source code files used by multiple *DMOS* programs, as it has in the IDL code and for the same reasons. Once again the intent is that ultimately the routines will be "rationalized," but for now if a user copies the source code in order to produce a new executable, care must be taken to use only the code in that program's subdirectory.

SAPLOT

The current version of **SAPLOT** is on *Tessie* in ***/local/models/IDL-SAPLOT*** and on *Tessie*, *Pinta*, and *Spray* in ***~calnan/IDL_code/SAPLOT***.

Note: As of the date of this writing (2007-02-05) new and updated code have not been placed on *Spray* since that computer is aboard the *M.V. Quest* for a sea trial.

Appendix

A.1 List of SAPLOT Routines

This section contains Table 1, an alphabetical list of the files that compromise the IDL version of **SAPLOT**. Prior to the current contract **SAPLOT** consisted of 51 IDL routines and one text "help" file. At this point the program now consists of 59 IDL routines, one utility IDL routine, and the help file.

As an aid to the following list, the main **SAPLOT** routine is named **saplot.pro**. As well, the names of all new routines have the string "New routine" below them, as is done for Filename 3 in the table.

Table 1. List of SAPLOT Files

Filename	Description
1. add_curve_info.pro	Called by READ_SAPFILE and READ_DATAMOD when a new curve is encountered in the input file. It adds array elements to an overall curve information structure for information on the new curve.
2. add_data_order.pro	Called by READ_SAPFILE when a command indicating plottable data is encountered: CURVE, PTPLT, PATPLT, IMAGE, or POLAR. It keeps track of the data's order so they can be plotted in the reverse order from which they are encountered.
3. add_image_info.pro New routine	Called by READ_IMAGE_DATA when the IMAGE command is encountered to store the commands' data in a temporary structure. The structure's contents are added to the complete variable later in PREP_PLOTS.
4. add_label_info.pro	Called by READ_LABELRC when one of the LABEL family of commands is encountered to store the commands' data in a temporary structure. The structure's contents are added to the complete variable later in PREP_PLOTS.
5. add_legend_info.pro	Called by READ_LEGENDRC when one of the commands LEGEND, ALSET, ILSET, PLSET, PSET, XLSET, LFSET, PATLEGEND, or PTLEGEND are encountered to store the commands' data in a temporary legend data structure, of which there are four.

Filename	Description
6. add_page.pro	Called by READ_SAPFILE when a new page is encountered in the input file to add the structure for the page to the overall data variable and fill it in with page-related defaults.
7. add_plot.pro	Called by READ_SAPFILE when a new plot is encountered in the input file in order to increment the number of plots in the space for the current page in the overall data variable and to add an empty plot structure to it.
8. checks.pro	Called by DRAW_LEGEND to check if curve data are inside a specified rectangle; used to find a data-free plot region for the legend.
9. colour_ci.pro	Called by DRAW_CURVES, DRAW_LABEL, DRAW_LEGEND, and DRAW_PATTERN to convert a **SAPLOT** colour code to a colour index that can be passed directly to a plotting routine.
10. comp_saplot.pro New routine	This is the utility routine. It is used to compile all the **SAPLOT** routines before running the program by way of the command `@comp_saplot`. In fact, this routine doesn't have to be run before running **SAPLOT** since the routines will compile anyway, but this allows a user to ensure that all the routines compile without error before starting a plotting run.
11. cpf.pro	Called by YERF to calculate a cumulative distribution function.
12. curve_pars.pro	Called by READ_SAPFILE to put the curve parameters, obtained from various curve related commands, into the overall data structure at the curve level. It is called once for each of the curve parameter structures.
13. databox_setup.pro	Called by PLOT_SAPDATA to set plotting parameters and "plot" the databox (actually drawing nothing) in order to "register" it for later overplotting of the data and the databox.
14. do_hatch.pro	Called by DRAW_HATCHPAT and DRAW_LEGEND to draw the hatch patterns of the `PATPLT` command.
15. do_rect.pro	Called to outline and/or fill in a rectangle by DRAW_CURVES (histogram bars), DRAW_LEGEND (boxes around the legend and legend items), and DRAW_GREYPAT (`PTPLT` pixels).

Filename	Description
16. draw_curves.pro	Called by PLOT_SAPDATA to draw data curves.
17. draw_databox.pro	Called by PLOT_SAPDATA to draw the data box, including tick marks and numbering.
18. draw_greypat.pro	Called by DRAW_PATTERN to draw greyscale data from the command `PTPLT`
19. draw_hatchpat.pro	Called by DRAW_PATTERN to draw hatch pattern data from the command `PATPLT`
20. draw_image.pro New routine	Called by PLOT_SAPDATA to plot one image.
21. draw_label.pro	Called by PLOT_SAPDATA to draw a label.
22. draw_legend.pro	Called by PLOT_SAPDATA to draw a legend.
23. draw_number.pro	Called by DRAW_DATABOX to write numbers on the data box's axes.
24. draw_pattern.pro	Called by PLOT_SAPDATA and recursively by DRAW_PATTERN to plot a line of `PATPLT` or `PTPLT` data; it calls further routines to do the actual drawing.
25. draw_symbols.pro	Called by DRAW_CURVES, DRAW_LEGEND, and recursively by DRAW_SYMBOLS to draw symbols on the output.
26. draw_tick.pro	Called by DRAW_DATABOX to draw tick marks and grid lines on the data box's axes.
27. drop_cmt.pro New routine	Called from READ_SAPDATA and READ_PTPAT_DATA to drop comments, including inline comments, from input lines before processing.
28. getword.pro	Called by PREP_STRING to get the next word (space delimited text) in a text string.
29. idl-saplot.help	Text file read by SHOW_FILE and written to the screen. It contains help information on IDL **SAPLOT** as well as information on program limitations, differences with the C version of **SAPLOT**, etc.
30. inch_data_conv.pro	Called by CHECKS and DRAW_LEGEND to convert measurements in inches to data units or vice versa.
31. int_check.pro New routine	Called from READ_SAPFILE, READ_IMAGE_DATA, READ_LABELRC, READ_LEGENDRC, and READ_PTPAT_DATA to ensure that a parameter string read from a script contains one integer and nothing else.

Filename	Description
32. legcom_check.pro	Called by READ_SAPFILE to check if the first word in a line in a legend is a **SAPLOT** command.
33. make_curve_struct.pro	Called by READ_SAPFILE to create a structure able to hold the data for one curve of known size.
34. make_ptpat_struct.pro	Called by READ_SAPFILE to create a structure to hold the data associated with the commands PATPLT and PTPLT.
35. num2str.pro	Called by DRAW_DATABOX and DRAW_NUMBER to write a number into a string with the requested number of digits after the decimal point and return the blank-free string.
36. numdec.pro	Called by DRAW_DATABOX to determine the number of digits required after a decimal point to maintain the accuracy of a written real number.
37. page_ptr.pro	Called by ADD_PAGE and ADD_PLOT to get a pointer to a page structure's contents.
38. plot_ptr.pro	Called by ADD_PAGE, ADD_PLOT, and PREP_PLOTS to get a pointer to a plot structure's contents with all the variables set to values indicating that defaults are to be used.
39. plot_sapdata.pro	Called by the main routine **SAPLOT** (if requested) to produce graphics.
40. polar_chk.pro New routine	Called by READ_SAPFILE to make sure that **SAPLOT** commands that are used by both CURVE and POLAR contain data valid for POLAR plotting.
41. prep_plots.pro	Called by READ_SAPFILE after the first run-through and the numbers of curves per page are known to redefine the plot structures so the correct numbers of curves are referenced. As well it moves LABEL data from a temporary structure into the main variable.
42. prep_string.pro	Called by DRAW_LABEL and DRAW_LEGEND to prepare a string for printing by converting certain IGL codes to IDL codes.
43. read_axisrc.pro	Called by READ_SAPFILE to check, validate, and store the axis-related commands ASET, DENSIT, NSET, RANGE, SSET, and TSET.

Filename	Description
44. read_datamod.pro	Called by READ_SAPFILE to check, validate, and store the data modification commands ADJUST, EXCH, and LOG10.
45. read_image_data.pro New routine	Called by READ_SAPFILE to read in IMAGE data lines.
46. read_labelrc.pro	Called by READ_SAPFILE to check, validate, and store the label-related commands LABEL, ALABEL, ILABEL, PLABEL, and XLABEL.
47. read_legendrc.pro	Called by READ_SAPFILE to check, validate, and store the legend-related commands LEGEND, ALSET, ILSET, PLSET, PSET, XLSET, LFSET, PATLEGEND, and PTLEGEND.
48. read_plotrc.pro	Called by READ_SAPFILE to check, validate, and store the plot-related commands AXSET, FSET, GRID, ORSET, PVSET, PWSET, and THSET.
49. read_ptpat_data.pro	Formerly named **add_ptpat_info.pro**, this function is called by READ_SAPFILE when a PTPLT or PATPLT command is encountered in order to get the specifics of the command for storage in a temporary structure.
50. read_sapfile.pro	Called by the main routine **SAPLOT** (if requested) to read the input file's contents into a variable.
51. reality_check.pro	Called by ADD_PTPAT_INFO, GETWORD, READ_AXISRC, READ_DATAMOD, READ_LABELRC, READ_LEGENDRC, READ_PLOTRC, and READ_SAPFILE to ensure that a passed-in variable contains one real number and nothing else.
52. rgb_ci.pro	Called by COLOUR_CI, DO_RECT, and PLOT_SAPDATA to convert an RGB colour specification to a colour index that can be passed directly to a plotting routine.
53. rotate_pts.pro	Called by DO_HATCH, DO_RECT, and DRAW_TICK to rotate points around specified points by specified angles.
54. saplot.pro	**Main routine**: checks user-supplied parameters and then calls READ_SAPFILE, PLOT_SAPDATA, and SHOW_FILE, if requested or required.

Filename	Description
55. saplot_defaults.pro	Called by PLOT_SAPDATA to puts all the defaults used by **SAPLOT** into a structure, which is returned.
56. saplot_umess.pro	Called by ADD_PTPAT_INFO, READ_AXISRC, READ_DATAMOD, READ_LABELRC, READ_LEGENDRC, READ_PLOTRC, and READ_SAPFILE to write error and warning messages to users, unless the capability is disabled in the main routine's parameter list.
57. show_file.pro	Called by the main routine **SAPLOT** (if requested) to read the help/information file **idl-saplot.help** and write it out to the screen.
58. string_len.pro	Called by DRAW_DATABOX and DRAW_LEGEND to return the length of a character string in screen units.
59. transfer_data.pro	Called by PLOT_SAPDATA to modify the data as read (as per the commands `ADJUST`, `EXCH`, and `LOG10`) and put the modified data into a temporary plotting structure.
60. ver_crv_pol.pro New routine	Called by READ_SAPFILE to verify the contents of a `CURVE` or `POLAR` command line.
61. yerf.pro	Called by DATABOX_SETUP, DRAW_CURVES, DRAW_LABEL, and INCH_DATA_CONV to calculate inverse cumulative probabilities when data are to be plotted on a probability axis or these axes are to be drawn.

A.2 Main SAPLOT Data Structure

The main data structure used within **SAPLOT** is called *VAR*, for *var*iable. *VAR* contains all the data in the script (or scripts, if `EXTERNAL` is used) as well as related information. It is returned to users by **SAPLOT** if requested, although users may give the returned structure any name they desire

Table 2 lists the complete structure of the main variable. That table contains three main columns, which are:

Data Items - This has the names of the structures and their members. (The terms "structures" and "pointers" are used interchangeably since pointers point to structures.) All structure names are followed by a number indicating the number of members that they have; for example "*PAGE* (2)" indicates that the structure *PAGE* has two direct members. Structure members are listed in a column to the right of the one holding the name of the structure. As well, the last member of each structure is noting as being so.

One point apparent in the **Data Items** sub-columns is that the name of the first one is **Level 2**. There is indeed a Level 1, and it consists of the main variable, the structure *VAR*. This structure contains three direct members: the variables *.TYPE* and *.NPAGE*, and the pointer array *.PAGE*. Since a Level 1 column would only contain a single entry, it was omitted in order to permit a better display of the tabulated information.

Data Type - This either indicates that the data item is a structure/pointer or, if it isn't, gives its data type. The definition of a data items may be followed by one of four indicators:

- (*n*) - where *n* is a number this indicates that the data item is assigned this value upon creation; this value may be changed at a later time
- ("`STR`") - where `STR` is a character string this indicates that the data item is assigned this value upon creation
- (*) - means that a definite, appropriate value is assigned to the member based on other information, and
- (!) - means that no value is assigned to the data item

Comments - This describes the data items and gives useful information on it. It may contain the names of the **SAPLOT** commands that provide the items' data or causes their creations in parentheses, e.g. "(`NEWPAGE`)". When it is stated that a variable contains a code value, the values permitted and their meanings can be found in the **SAPLOT** user's guide under the **SAPLOT** commands located in that comment.

Table 2. Main SAPLOT Data Structure

Data Items				Data Type	Comments
Level 2	**Level 3**	**Level 4**	**Level 5**		
.TYPE				string "SAPLOT"	set as an identifier
.NPAGE				integer (*)	number of pages (NEWPAGE)
.PAGE[n] (2) (last *VAR* member)				pointer array of size *[.NPAGE]* (!)	pointers to each page's data; created in PAGE_PTR
	.NPLOT			integer (1)	number of plots on this page (PLOT)
	.PLOT[n] (31) (last *PAGE* member)			pointer array of size *[.NPLOT]* (!)	pointers to each plot's data; created in PLOT_PTR
		.FORMAT		integer (1)	format style of the page (FORMAT)
		.CLIP		string ("YES")	flag changed to "NO" if the command NCLIP is given (NCLIP)
		.COORD		string ("CARTESIAN")	coordinate system for CURVE data, changed to "POLAR" if a polar plot is to be produced (POLAR)
		.PRSCALE		integer (1)	reassigned to the value specified, if necessary (PRSCALE)
		.AXIS[4] (9)		fixed size structure array (!)	parameters for the 4 axes; created in PLOT_PTR
			.TYPE	integer (-1)	a code for the axis type (linear, log, etc.) (ASET)
			.LABSTYLE	integer (-99)	a code for the labelling style (NSET)
			.RNGST	string ("AUTO")	range determination flag; changed to "YES" or "INVAUTO" if users choose those options (RANGE)
			.RNGMN	real (-1.)	range minimum (RANGE)

Data Items				Data Type	Comments
Level 2	**Level 3**	**Level 4**	**Level 5**		
			.RNGMX	real (-1.)	range maximum (RANGE)
			.SSET	string ("NO")	flag changed to "YES" if there are SSET data for this axis (SSET)
			.TIC1	real (-1.)	tick mark start (SSET)
			.TICSP	real (-1.)	tick mark spacing (SSET)
			.TICSUB (last *AXIS* member)	integer (-1)	number of tick mark subdivisions (SSET)
		.XAXL		real (-1.)	• CURVE: *x* axis length in inches • POLAR: plot radius in inches (AXSET)
		.YAXL		real (-1.)	• CURVE: *y* axis length in inches • POLAR: plot radius in inches but set by **SAPLOT** (AXSET)
		.TICIN		real (-9999.)	major tick mark length inside the data window in tenths of inches (TSET)
		.TICOUT		real (-9999.)	major tick mark length outside the data window in tenths of inches (TSET)
		.SMTICIN		real (-9999.)	minor tick mark length inside the data window in tenths of inches (TSET)
		.SMTICOUT		real (-9999.)	minor tick mark length outside the data window in tenths of inches (TSET)
		.FSET		integer (-1)	flag: whether or not to draw the plot's frame (FSET)
		.SMTICDEN		integer (-1)	small tick mark density (DENSIT)

Data Items				Data Type	Comments
Level 2	Level 3	Level 4	Level 5		
		.GRID[4,2]		integer array 8*(-99)	major and minor grid line types for each of the 4 axes (GRID)
		.XORIG		real (-1.)	*x* origin: • CURVE: distance from the left side of the page to the plot in inches • POLAR: distance from the left side of the page to the centre of the plot in inches (ORSET)
		.YORIG		real (-1.)	*y* origin: • CURVE: distance from the bottom of the page to the plot in inches • POLAR: distance from the bottom of the page to the centre of the plot in inches (ORSET)
		.PIVOT		string ("NO")	flag changed to "YES" if PVSET is called (PVSET)
		.XPIV		real (-1.)	pivot point's *x* value from the left side of the page in inches (PVSET)
		.YPIV		real (-1.)	pivot point's *y* value from the bottom of the page in inches (PVSET)
		.LINWDTH		real (-1.)	line width in units of 1/300 inch (PWSET)
		.ROTN		real (-9999.)	the plot's Cartesian angle of rotation in degrees about the pivot point (THSET)
		.FNTSIZ		real (-1.)	label size in points (TXTSIZ)

Data Items				Data Type	Comments
Level 2	Level 3	Level 4	Level 5		
		.NDATS		integer (*)	number of plottable data sets in this plot (CURVE, PTPLT, PATPLT, IMAGE, POLAR)
		.DATORD[n]		string array of size *[.NDATS]* (!)	the type of data to plot in order of appearance: "CRV" – CURVE data "GRY" – greyscale PTPLT data "PAT" – pattern PATPLT data "IMG" – IMAGE data "POL" – POLAR data
		.NCURV		integer (*)	number of curves in this plot (CURVE, POLAR)
		.CURVES[n] (11)		pointer array of size *[.NCURV]* (!)	pointers to each curve's data; created in MAKE_CURVE_STRUCT (CURVE, POLAR)
			.INLINE	integer (-1)	input file line number of this CURVE command (CURVE, POLAR)
			.NPTS	integer (*)	number of points in the curve (CURVE, POLAR)
			.X[n]	real array of size *[.NPTS]* (!)	contains x or r data for the curve (CURVE, POLAR)
			.Y[n]	real array of size *[.NPTS]* (!)	contains y or θ data for the curve (CURVE, POLAR)
			.CTYPE	integer (-1)	curve type to use (CSET)
			.GREY	integer (-1)	greyscale value to use for a histogram's curve (GFILL)
			.ANG_UN	string ("X")	for POLAR plots this is the angle units, changed to "D" (degrees) or "R" (radians) (POLAR)

Data Items				Data Type	Comments
Level 2	Level 3	Level 4	Level 5		
			.COORD	string ("X")	for POLAR plots this is changed to "T" for plots in °True or "C" for plots in °Cartesian (ASET)
			.COLOUR	integer (0)	colour of the curve (LINCLR)
			.LSET	string ("NO")	flag changed to "YES" if LSET is used for this curve (LSET)
			.LINTYP	integer (-1)	line type for the curve (LSET)
			.SYMTYP	integer (-1)	symbol type for the curve (MSET)
			.LINWD (last *CURVES* member)	real (-1.)	line width for the curve (WSET)
		.NDMOD		integer (*)	number of data modification commands (ADJUST, EXCH, LOG10)
		.DATAMOD[n] (8)		pointer array of size *[.NDMOD]* (!)	pointers to the data modification info; created in PREP_PLOTS (ADJUST, EXCH, LOG10)
			.CMD	string ("DUMMY")	changed to the specific data modification command: "ADJUST", "EXCH", or "LOG10" (ADJUST, EXCH, LOG10)
			.CRV	integer (0)	curve number the data modification is intended for
			.XMULT	real (1.)	value to multiply the *x* or *r* array by (ADJUST)
			.XADD	real (0.)	value to add to the *x* or *r* array after being multiplied by *.XMULT* (ADJUST)
			.YMULT	real (1.)	value to multiply the *y* or θ array by (ADJUST)

Data Items				Data Type	Comments
Level 2	Level 3	Level 4	Level 5		
			.YADD	real (0.)	value to add to the y or θ array after being multiplied by *.YMULT* (ADJUST)
			.XLOG	string ("NO")	flag changed to "YES" if $\log_{10}$ of the x or r data are to be taken before plotting (LOG10)
			.YLOG (last *DATAMOD* member)	string ("NO")	flag changed to "YES" if $\log_{10}$ of the y or θ data are to be taken before plotting (LOG10)
		.NPTPAT		integer (*)	number of PTPLT and PATPLT commands on the plot
		.PTPATS[n] (13)		pointer array of size *[.NPTPAT]* (!)	pointers to PTPLT and PATPLT data; created in MAKE_PTPAT_STRUCT
			.TYPE	string ("XXX")	pattern type, changed to "GRY" for greyscale or "PAT" for pattern
			.INLINE	integer (-1)	input file line number of this PTPLT or PATPLT command (PTPLT, PATPLT)
			.XOFF	real (-1.)	the PTPLT or PATPLT x offset values
			.YOFF	real (-1.)	the PTPLT or PATPLT y offset values
			.XREP	integer (-1)	the number of times each pixel is to be repeated in the x direction as it is being plotted
			.YREP	integer (-1)	the number of times each pixel is to be repeated in the y direction as it is being plotted
			.R	real (-1.)	fraction of red values for PTPLT plots
			.G	real (-1.)	fraction of green values for PTPLT plots
			.B	real (-1.)	fraction of blue values for PTPLT plots

Data Items				Data Type	Comments
Level 2	Level 3	Level 4	Level 5		
			.XPPI	real (-1.)	*x* direction pixel density in pixels/inch
			.YPPI	real (-1.)	*y* direction pixel density in pixels/inch
			.NLINES	integer (*)	number of lines of data; initialized to the appropriate *PTPAT_INFO.NUMLIN* value
			.LINES[i] (last *PTPATS* member)	real array of size *[.NLINES]* (!)	the greyscale/pattern data
		.NLABS		integer (*)	number of labels in this plot (`LABEL`, `ALABEL`, `ILABEL`, `PLABEL`, and `XLABEL`)
		.LABELS[n] (9)		pointer array of size *[.NLABS]* (!)	pointers to each label's data; created in PREP_PLOTS
			.CMD	string (“`DUMMY`”)	changed to the specific label command used, if any: `LABEL`, `ALABEL`, `ILABEL`, `PLABEL`, or `XLABEL`
			.WHERE	integer (0)	the `LABEL` positioning parameter
			.X	real (0.)	the `*LABEL` (excluding `LABEL`) label's *x* position
			.Y	real (0.)	the `*LABEL` (excluding `LABEL`) label's *y* position
			.ANG	real (0.)	rotation of the `*LABEL` (excluding `LABEL`) label
			.XJUST	integer (0)	the `*LABEL` (excluding `LABEL`) label's *x* justification: 0 = left 1 = centre 2 = right

Data Items				Data Type	Comments
Level 2	Level 3	Level 4	Level 5		
			.YJUST	integer (0)	the `*LABEL` (excluding `LABEL`) label's *y* justification: 0 = bottom 1 = centre 2 = top
			.COLOUR	integer (0)	current text colour (`TXTCLR`)
			.LAB (last *LABELS* member)	string (“`DUMMY`”)	changed to the label itself
		.NLEGS		integer (*)	number of legends in this plot
		.LEGENDS[n] (18)		pointer array of size *[.NLEGS]* (!)	pointers to each legend's data; created in READ_SAPFILE
			.TYPE	string (*)	legend type, “`LEGEND`”, “`PATLEGEND`”, or “`PTLEGEND`”
			.NLINES	integer (*)	number of lines in the legend (`LEGEND`)
			.NN	integer (*)	number of `N*` values for `PATLEGEND` or `PTLEGEND`; maximum value is 5
			.N1	integer (*)	`N1` values for `PATLEGEND` or `PTLEGEND`
			.N2	integer (*)	`N2` values for `PATLEGEND` or `PTLEGEND`
			.N3	integer (*)	`N3` values for `PATLEGEND` or `PTLEGEND`
			.N4	integer (*)	`N4` values for `PATLEGEND` or `PTLEGEND`
			.N5	integer (*)	`N5` values for `PATLEGEND` or `PTLEGEND`
			.LINES[n]	string array of size `ABS`(*[.NLINES]*) (!)	legend's text lines

Data Items				Data Type	Comments
Level 2	Level 3	Level 4	Level 5		
			.POSCMD	string ("NONE")	command used to position the legend, changed to the command used (ALSET, ILSET, PLSET, PSET, XLSET)
			.X	real (-1.)	legend's *x* position (ALSET, ILSET, PLSET, PSET, XLSET)
			.Y	real (-1.)	legend's *y* position (ALSET, ILSET, PLSET, PSET, XLSET)
			.ROT	real (-9999.)	legend's rotation in degrees (ALSET, ILSET, PLSET, PSET, XLSET)
			.XJUST	integer (-1)	legend's *x* justification: 0 = left 1 = centre 2 = right (ALSET, ILSET, PLSET, PSET, XLSET)
			.YJUST	integer (-1)	legend's *y* justification: 0 = bottom 1 = centre 2 = top (ALSET, ILSET, PLSET, PSET, XLSET)
			.COLOUR	integer (0)	current text colour (TXTCLR)
			.FRAME	integer (-1)	frame type to draw around the legend (LFSET)
			.FILL (last *LEGENDS* member)	integer (-1)	whether to blank-fill the legend box before writing the legend (LFSET)

A.3 Internal SAPLOT Plotting Parameter Structure

The structure *PL_PARS* contains plotting parameters for the current plot. It is created in PLOT_SAPDATA and is filled with plotting parameters in various routines as their values become known. The values are reset for each plot encountered in the input file. This structure is then passed to all the routines that do any drawing, serving as a sort of "common" area of these parameters. There is a certain amount of redundancy in the structure, but this was done to make it easier to use its contents.

Table 3. Internal Main Plotting Data Structure

Item	Data Type	Comment
PL_PARS	main structure, contains:	created in PLOT_SAPDATA
.PTYPE	string	initialized as "XY" for the default of an (*x*,*y*) plot, but will be reset to "PTRUE" (polar: True) or "PCART" (polar: Cartesian) if one of those plot types is specified
.FI	integer	default format index
.ORIENT	string	"P" or "L", depending on whether the plot is orientated as portrait or landscape
.ODEV	string	output device: "SCR" for the screen, "PS" for a PostScript file, and "EPS" for an EPS file
.DSCL	real	device scale (device units/inch; this is a fiction for screen output but is necessary to maintain aspect ratios)
.CCLIP	string	"YES" or "NO": whether or not to clip CURVE or POLAR data at the data boxes' borders
.ICLIP	string	"YES" or "NO": whether or not to clip IMAGE data at the data boxes' borders
.XAXTYP	integer	*x* axis type as per ASET
.YAXTYP	integer	*y* axis type as per ASET
.XS	real	output device's *x* axis size (device units)
.YS	real	output device's *y* axis size (device units)
.XMXPIX	integer	width of the screen plot in pixels
.YMXPIX	integer	height of the screen plot in pixels
.FNHT	real	plotted font height (inches)

Item	Data Type	Comment
.SYMHT	real	plotted symbol height (inches)
.NLEN	real	plotted length of a text string (device units)
.XAXL	real	data box's *x* axis length (inches)
.YAXL	real	data box's *y* axis length (inches)
.SCALE	real	scaling value for the plot (1. if the plot's format does not have font scaling)
.XORIG	real	data box's *x* origin (inches)
.YORIG	real	data box's *y* origin (inches)
.ROTN	real	data box's rotation (Cartesian degrees)
.XPIVOT	real	*x* point rotation is around (inches)
.YPIVOT	real	*y* point rotation is around (inches)
.XMIN	real	minimum *x* axis value (data units)
.XMAX	real	maximum *x* axis value (data units)
.XMINPU	real	minimum *x* axis value (plotting units)
.XMAXPU	real	maximum *x* axis value (plotting units)
.AXMIN	real	minimum *!X* axis value
.AXMAX	real	maximum *!X* axis value
.YMIN	real	minimum *y* axis value (data units)
.YMAX	real	maximum *y* axis value (data units)
.YMINPU	real	minimum *y* axis value (plotting units)
.YMAXPU	real	maximum *y* axis value (plotting units)
.AYMIN	real	minimum *!Y* axis value
.AYMAX	real	maximum *!Y* axis value
.NUMSP[4]	real	space taken by the numbers written to the four axes (inches)
.TICIN	real	longest interior tick mark (inches)
.NCUR	integer	number of curves on the plot
.NPLC	integer	number of points in the longest curve
.CTICO	real	column plot tick mark origin
.RTICO	real	row plot tick mark origin
.CTICSEP	real	column plot tick mark separation
.RTICSEP	real	row plot tick mark separation
.GSD	real	grey scale delta between histogram bars
.NCP[n]	integer array of size *[.NCUR]*	number of points in each curve
.X[n,m]	real array of size *[.NCUR,.NPLC]*	all the *x* data for the plot converted from input to output format
.Y[n,m]	real array of size *[.NCUR,.NPLC]*	all the *y* data for the plot converted from input to output format
.T3DTR[3]	real array	the translation array for IDL command `T3D`; used to pass `THSET`/`PVSET` data
.T3DROT[3]	real array	the rotation array for IDL command `T3D`; used to pass `THSET`/`PVSET` data

Item	Data Type	Comment
.ERRS	string	a flag: "YES" if error messages are to be written out and "NO" if they aren't
.WARN	string	a flag: "YES" if warning messages are to be written out and "NO" if they aren't
.LTHK	real	default line thickness for the format *PL_PARS.FI*
.LTHKSCL	real	current line thickness scale factor
.CLIPNO	integer	plot clipping code
.NCBARS	integer	number of column bars (histogram)
.CBARWDTH	real	column bar width (histogram)
.XCB1	real	initial column *x* position (histogram)
.COLSCL	real	column width scale factor (histogram)
.CRECT	*RECT* structure	column rectangle data one bar at a time (histogram)
.CBARNO	integer	column bar counter (histogram)
.NRBARS	integer	number of row bars (histogram)
.RBARWDTH	real	row bar width (histogram)
.YRB1	real	initial row *y* position (histogram)
.ROWSCL	real	row width scale factor (histogram)
.RRECT	*RECT* structure	row rectangle data one bar at a time (histogram)
.RBARNO	integer	row bar counter (histogram)
.LROTN	*LROTN* structure	structure needed by DO_RECT
.LCI	integer	last colour index used, needed when the IDL command TVLCT is used for PostScript output
.RDIF	real	RGB R difference between the maximum value and 100% on the 0-255 scale
.GDIF	real	RGB G difference between the maximum value and 100% on the 0-255 scale
.BDIF	real	RGB B difference between the maximum value and 100% on the 0-255 scale
.PLEFT	real	leftmost PTPLT or PATPLT data position
.PRIGHT	real	rightmost PTPLT or PATPLT data position
.PTOP	real	topmost PTPLT or PATPLT data position
.PBOTTOM	real	bottommost PTPLT or PATPLT data position

Item	Data Type	Comment
.PVI	integer	pattern vertical index from the top down; incremented by 1 every time a `PTPLT` or `PATPLT` line of data is drawn

References

[1] Calnan, Colin. *IDL SAPLOT Program Development*, DREA Contractor's Report DREA CR 2000-07, August 2000.

[2] Calnan, Colin. *Enhancements to IDL SAPLOT Graphics and Conversion of the OASES Acoustic Propagation Model to HP Computers*, DREA Contractor's Report DREA CR 2001-083.

[3] Young, Catherine A. *SAPLOT: Scientific Graphic Software User's Manual*, DREA Technical Communication 91/307, December 1991.

[4] Hughes, Stephen (assumed author). **saplot_ps.doc**. This is an untitled, undated, and unaccredited text file found on several computers accompanying the C code for an earlier version of **SAPLOT**. The document states that it's for version 1.0b21 and is a user's guide for **SAPLOT**.

Special thanks are extended to Bill Roger for help in trying to decypher some of the confusion surrounding IDL's handling of image files.

This page intentionally left blank.

DOCUMENT CONTROL DATA

(Security markings for the title, abstract and indexing annotation must be entered when the document is Classified or Designated)

1. ORIGINATOR (The name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Centre sponsoring a contractor's report, or tasking agency, are entered in section 8.)

C. Calnan
xwave
38 Solutions Drive
Halifax, N.S.
B3S 1N2

2a. SECURITY MARKING (Overall security marking of the document including special supplemental markings if applicable.)

UNCLASSIFIED

2b. CONTROLLED GOODS

(NON-CONTROLLED GOODS)
DMC A
REVIEW: GCEC DECEMBER 2012

3. TITLE (The complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S, C or U) in parentheses after the title.)

SAPLOT Maintenance and Enhancement

4. AUTHORS (last name, followed by initials – ranks, titles, etc. not to be used)

Calnan, C.

5. DATE OF PUBLICATION (Month and year of publication of document.)	6a. NO. OF PAGES (Total containing information, including Annexes, Appendices, etc.)	6b. NO. OF REFS (Total cited in document.)
August 2007	70	4

7. DESCRIPTIVE NOTES (The category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Contract Report

8. SPONSORING ACTIVITY (The name of the department project office or laboratory sponsoring the research and development – include address.)

Defence Research and Development Canada – Atlantic
9 Grove Street
P.O. Box 1012
Dartmouth, Nova Scotia B2Y 3Z7

9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant.)	9b. CONTRACT NO. (If appropriate, the applicable number under which the document was written.)
	W7707-06-3594

10a. ORIGINATOR'S DOCUMENT NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document.)	10b. OTHER DOCUMENT NO(s). (Any other numbers which may be assigned this document either by the originator or by the sponsor.)
1012063	DRDC Atlantic CR 2007-132

11. DOCUMENT AVAILABILITY (Any limitations on further dissemination of the document, other than those imposed by security classification.)

Unlimited

12. DOCUMENT ANNOUNCEMENT (Any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in (11) is possible, a wider announcement audience may be selected.))

Unlimited

13. ABSTRACT (A brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual.)

Previous contracts resulted in the design, creation, and implementation of the IDL program **SAPLOT**, which is based on earlier versions in FORTRAN and C. Continuing use of this program has uncovered a few bugs and some enhancements that would make the program even more useful.

In addition, the program suite *DMOS* is known to have a problem that affects the calculation of reverberation time series, a problem that has existed within the code since its creation.

The current contract was let in order to solve the problem in *DMOS*, fix the **SAPLOT** bugs, and to add some new capabilities to **SAPLOT**.

The change to *DMOS* was accompanied by a change in the *DMOS* User's Guide, a document that is still under production. The fixes and enhancements to **SAPLOT** are documented in this Contractor's Report alone since that program doesn't have a current User's Guide.

Dans le cadre de contrats précédents, nous avons conçu, élaboré et mis en œuvre le programme SAPLOT à l'aide d'anciennes versions des langages FORTRAN et C. Nous avons depuis relevé certains bogues, ainsi qu'établi des améliorations qui en augmenterait l'utilité.

Il est également connu que la suite de programmes DMOS comporte un problème sur le plan du calcul des séries de temps de réverbération, problème qui existe dans le code depuis sa création.

Le contrat actuel vise à corriger ledit problème et les bogues de SAPLOT, de même qu'à intégrer de nouvelles capacités dans celui-ci.

La modification de DMOS se reflète dans le guide de l'utilisateur connexe, un document qui est toujours en cours de production. Toutefois, les corrections et les améliorations apportées à SAPLOT ne sont consignées que dans le rapport de l'entrepreneur, car il n'existe actuellement aucun guide de l'utilisateur pour ce programme.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (Technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus, e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

FORTRAN; C; DMOS; IDL; surface scattering coefficient; PERL; PYTHON; scripts

Defence R&D Canada

Canada's Leader in Defence
and National Security
Science and Technology

R & D pour la défense Canada

Chef de file au Canada en matière
de science et de technologie pour
la défense et la sécurité nationale

www.drdc-rddc.gc.ca