Plotc User's Manual

Version 2.6

Charles E. Towne
NASA Glenn Research Center
Cleveland, Ohio

November, 2005

Plotc User's Manual
Version 2.6

Charles E. Towne

National Aeronautics and Space Administration
Glenn Research Center
Cleveland, Ohio

Abstract

Plotc is a two-dimensional plotting program that allows the user to create various plots without becoming an expert in the use of, or even knowing about, the baseline graphics routines. Plots can be created consisting of multiple curves and/or sets of symbols. The curves can be plotted as solid lines, lines with symbols (vector-symbol plots), or in a sequence consisting of different line types (solid, dashed, dot-dashed, etc.) An option is provided to allow a multiple curve plot with each curve offset in the x-direction by some user-specified increment. Plots can be created with or without axes, labels, and legends. Axes can be either standard or logarithmic. The scaling of the axes can be done automatically, or specified by the user. Colors may be specified for all elements of the plot. A special-case version of *plotc*, called *plotc_p3d*, can be used to create two-dimensional line plots from the xyz and q files created by many computational fluid dynamics codes for use with the three-dimensional plotting program *Plot3d*. *Plotc* is written in Fortran 90, and is normally run on a Linux workstation. The plot may be displayed on the graphics monitor using OpenGL graphics routines, and a PostScript file is created that may be printed on a PostScript printer or viewed with a PostScript previewer. The OpenGL and PostScript plots are created separately; i.e., the PostScript plot is not merely a screen capture of the OpenGL plot. The OpenGL plot is intended as a "preview". The PostScript plot allows a wider variety of fonts to be used, with subscripts/superscripts, special symbols, etc.

Contents

1 Introduction

Plotc is a two-dimensional plotting program that allows the user to create various plots without becoming an expert in the use of, or even knowing about, the baseline graphics routines. Plots can be created consisting of multiple curves and/or sets of symbols. The curves can be plotted as solid lines, lines with symbols (vector-symbol plots), or in a sequence consisting of different line types (solid, dashed, dot-dashed, etc.) An option is provided to allow a multiple curve plot with each curve offset in the x-direction by some user-specified increment. Plots can be created with or without axes, labels, and legends. Axes can be either standard or logarithmic. The scaling of the axes can be done automatically, or specified by the user. Colors may be specified for all elements of the plot.

A special-case version of *plotc*, called *plotc_p3d*, may also be used. *Plotc_p3d* can be used to create two-dimensional line plots from the xyz and q files created by many CFD codes for use with the three-dimensional plotting programs *Plot3d* and FAST (Walatka, Buning, Pierce, and Elson, 1990). Plots may be created showing the variation of a function along any of the coordinate lines in the three-dimensional flow field. *Plotc_p3d* is described in detail in Appendix A.

Plotc is written in Fortran 90, and is normally run on a Linux workstation. The plot may be displayed on the graphics monitor using OpenGL graphics routines, and a PostScript file is created that may be printed on a PostScript printer or viewed with a PostScript previewer. The OpenGL and PostScript plots are created separately; i.e., the PostScript plot is not merely a screen capture of the OpenGL plot. The OpenGL plot is intended as a "preview". The PostScript plot allows a wider variety of fonts to be used, with subscripts/superscripts, special symbols, etc.

A slightly earlier version of *plotc* is available for SGI systems, with the on-screen plot created using SGI's gl routines. Much earlier versions of *plotc* have also been run on IBM, Sun, and DEC workstations, and a Convex mainframe.

2 General Description

In this section the basic characteristics and capabilities of *plotc* are described. The input parameters are described in greater detail in Section 3, and example cases are presented in Section 4.

2.1 Input/Output

When *plotc* is run, a variety of input parameters are available to specify the types of curves and symbols to be drawn, information about the axes, the placement of the plot on the page, the text strings to be written, the legend to be used, the fonts and sizes to be used for text, colors, etc. These parameters are specified in a namelist input file supplied by the user, and all of them have default values. Additional input is needed to supply the coordinates of the curves and symbols to be plotted. These coordinates may be specified in a variety of ways, as described in Section 3.3 through Section 3.5. Multiple plots may be created with one input file by simply stacking sets of input data.

The plot is normally displayed on the graphics monitor, using OpenGL graphics routines, and a PostScript file is created that may be printed on a PostScript printer or viewed with a PostScript previewer. The OpenGL and PostScript plots are created separately; i.e., the PostScript plot is not merely a screen capture of the OpenGL plot. The OpenGL plot is intended as a "preview". The PostScript plot allows a wider variety of fonts to be used, with subscripts/superscripts, special symbols, etc.

2.2 Curves and Symbols

Plots can be created consisting of multiple curves and/or multiple sets of symbols. The input parameter `NCURV` specifies the number of curves, and its sign determines whether each curve is to be plotted as a solid line, or in a sequence consisting of different line types (solid, short-dashed, dot-dashed, long-dashed, and dotted.) The input array `ICTYPE` may be used to change this sequence. The input parameter `OFFSET` may be used in a multiple curve plot to offset each curve in the x-direction. This parameter is useful, for example, when plotting a family of velocity profiles from a marching analysis at different streamwise stations.

Similarly, the input parameter `NSYMB` specifies the number of sets of symbols, and its sign determines whether each set is to be plotted as circles, or in a sequence consisting of different symbol types (circle, square, triangle, diamond, inverted triangle, lower right quadrant circle, upper semicircle, rhombus, pentagon, and house.) The input array `ISTYPE` may be used to change this sequence, and to specify filled instead of open symbols. The size of the symbols is set by the array `ISIZE1`. Error bars may be included with the symbols, as controlled by the flag `IEBAR`.

The total number of curves plus the total number of symbol sets is limited to 1000. The total number of points describing the curves and symbol sets is limited to 100,000. These limits may be changed by modifying the file *params.h* and re-installing the code.

In addition to plotting the curves as *vectors* (i.e., lines), as described above, the input flag `ITYPE` may be used to specify a *point* or *vector-symbol* plot. In a point plot, each coordinate defining the curve is plotted as a single point. In a vector-symbol plot, each curve is plotted as a line, with symbols distributed every `IFREQ` points along the curve. A different symbol will be used for each curve in a vector-symbol plot, in the same sequence as described above for plotting sets of symbols. The size of the symbols on a vector-symbol curve is set by `ISIZE`.

When curves and symbols appear on the same plot, the curves are drawn first, followed by the symbols. When "open" symbols are being drawn the symbols are actually filled in with the background color, then outlined with the color used for lines. Material behind the symbols, such as curves, axes, grid lines, etc., will thus be hidden. This behavior may be changed using the flag `IMASK(1)`.

2.3 Subroutines `COORS` and `POINTS`

Plotc can be viewed as a plotting program surrounding two key subroutines — `COORS` and `POINTS`. The plotting program takes care of all the boring stuff, like scaling and labeling axes, centering the plot, doing the mapping between user's units and relative (i.e., plotting frame) units, etc. The fun stuff is done by the `COORS` and `POINTS` routines.

To get the coordinates of the curves to be plotted, a subroutine named `COORS` is called. Similarly, to get the coordinates of the points to be plotted as symbols, plus the size of any error bars, a subroutine named `POINTS` is called. The *plotc* program contains versions of these subroutines that simply read in coordinates from either the namelist input file or from separate files. This input is described in Section 3.3 and Section 3.4.

The supplied versions of `COORS` and `POINTS` allow a fair amount of variation in how coordinates are specified. However, the real power and flexibility of *plotc* becomes clearer when you start customizing the code for specific applications by writing your own subroutines `COORS` and/or `POINTS`. Details on the parameters to be returned by user-written versions of subroutines `COORS` and `POINTS` are presented in Section 3.5.

2.4 The Plotting Frame

Plots are created within a plotting frame measuring `FWID` × `FHGT` relative units. The default values for these parameters are 8.5 × 11.0 in portrait mode, and 11.0 × 8.5 in landscape mode. The choice of portrait or landscape mode is determined by the input flag `IROT`. The values of `FWID` and `FHGT` were chosen, obviously, to match the paper size in inches normally used to print the PostScript version of the plot.

Note that the values of `FWID` and `FHGT` do not affect the physical size of the window on the graphics monitor used to display the plot. The size of the window depends on the size of the monitor. The shape is determined by the ratio `FWID/FHGT`.[1] The PostScript plot, of course, is always printed on paper, with default measurements of 8.5 × 11 inches (in portrait mode), or 11 × 8.5 inches (in landscape mode). If necessary, the PostScript plot will be rescaled to fit on the page.

It's important to note the distinction between *relative* and *user's* units. *Relative units* are determined by `FWID` and `FHGT`, and are arbitrary units used to specify the limits of the plotting frame. *User's units* are determined by `XMIN`, `XMAX`, `YMIN`, and `YMAX`, the starting and ending values for the x- and y-axes. These parameters are described briefly in the next section, and in greater detail in Section 3.2. Some of the input parameters may be specified in either user's or relative units.

Normally, the plot will be centered both horizontally and vertically within the plotting frame. The exact location may be specified, however, using the input parameters `XORGN` and `YORGN`.

[1]The maximum dimension of the window will be 80% of the minimum dimension of the monitor. For the default values of `FWID` and `FHGT`, and for a 1280 × 1024 monitor, the window will thus be 633 × 819 pixels in portrait mode, and 819 × 633 pixels in landscape mode.

2.5 Axes

Plots may be created with or without axes, as specified by the input flags IXAXIS and IYAXIS. For plots with axes, these flags also control whether tick marks or grid lines are drawn. Logarithmic axes may be specified by the flags ILOGX and ILOGY.

The axis limits may be specified by the user using the input parameters XMIN, XMAX, YMIN, and YMAX. These limits may also be computed automatically for either or both axes, as determined by the flag ISCALE. The input parameters XLEN and YLEN specify the actual lengths of the x- and y-axes in relative units. Note that even if a plot is being created without axes, all of these values are still used by *plotc* to transform user's units into relative units.

The number of intervals on each axis (i.e., one less than the number of tick marks or grid lines) is given by the parameters NINTX and NINTY. Numeric labels are normally placed at each tick mark or grid line. This may be overridden by specifying the number of labels using the parameters NLABX and NLABY.

The axes and plot titles are specified by the variables XTITLE, YTITLE, and PTITLE. These may specified as part of the *plotc* input, or supplied by subroutine COORS.

By default, the x- and y-axes will intersect at the point (XMIN,YMIN). The intersection point may be changed using the parameters XORGU and YORGU. The input flag IFRAME may be used to put a box around the plot by drawing horizontal lines at YMIN and YMAX, and vertical lines at XMIN and XMAX, with or without tick marks.[2]

2.6 Text String Characteristics

Text strings are used in several places when creating a plot. They may be written at user-specified locations on the plot, as described in the next section. A legend table may also be created, as described in Section 2.8. Text strings are also used when writing the axis and plot titles, and the numeric axis labels.

The flag ITITLE controls whether or not titles should be written for the axes and for the plot itself. For the plot title, ITITLE also controls whether it is placed at the top of the plot or the frame. Normally the plot title is left-aligned, and the axis titles are centered. This may be changed using the input character array TALIGN.

For the PostScript version of the plot, a variety of fonts may be used for any of the text strings, as specified by the input character array FONTS. The character size for titles and axis labels may be specified using the parameters ISIZET and ISIZEL. In the OpenGL version of the plot, the fonts used currently may not be modified by the user.

In the PostScript plot, special characters (various symbols, Greek letters, etc.), subscripts, and superscripts may be used in text strings. Special characters are included using the notation \nnn, where *nnn* is the three-digit octal code for the character. Note that a backslash is used, not an ordinary slash. For all fonts except Symbol, *nnn* is the same as the PostScript octal code for the symbol. For the Symbol font, *nnn* is the PostScript octal code +400. The values of *nnn* for all the characters in the standard and Symbol fonts are shown in Table 1 and Table 2. These tables were modeled after those in Appendix A of the *PostScript Language Reference Manual* (Adobe Systems, Inc., 1985).

Subscripts are included in text strings using the notation \sub[whatever-you-want]. Superscripts

[2]The name of this input parameter, IFRAME, was a poor choice. It has nothing to do with the size of the plotting frame.

are done the same way, but with `\sup` instead of `\sub`. For example, `Re\sub[\561]` will appear as Re_θ.

You can also change fonts in the middle of a text string. The available fonts are listed in Section 3.1.4 under the description of the input variable `FONTS`. For fonts with one-character names, like Helvetica or Italic, use the notation `\ft`*x*, where *x* is the one-letter symbol for the font. For fonts with two-character names, like Helvetica Oblique, use the notation `\ft(`*xx*, where *xx* is the two-letter symbol for the font. This is useful for displaying mathematical variables in an italic or oblique font, as is normally done in technical literature, when the rest of the string is in some other font. For instance, the string "`\ftHReynolds Number, \ft(HORe`" will appear as Reynolds Number, *Re*.

As noted earlier, all these special symbols, etc., only work for the PostScript version of the plot. In the OpenGL version of the plot, text strings will appear strictly as specified by the user, with all the ugly \'s.

See Section 4.2 for an example using some of these features.

2.7 Text Strings at Arbitrary Locations

Up to 50 text strings may be written at arbitrary locations on the plot. The maximum number of characters allowed in each string, including those used to specify special characters, subscripts, etc., is 100. The text strings are specified by the input character array `ASTR`, and their locations by the arrays `XSTR` and `YSTR`. The locations may be specified in either relative or user's units, as determined by the input flag `NSTR`. Each string may be left-aligned, centered, or right-aligned at the specified locations, as determined by the input character array `SALIGN`. The character size for each string is set using the array `ISIZES`.

Normally, the area of the plot in which the strings are to appear is masked out before the strings are written. Material behind the strings, such as curves, axes, grid lines, etc., will thus be hidden. This behavior may be changed using the flag `IMASK(2)`.

2.8 Legend Table

A two-column legend table may be included on the plot.[3] The first column will show each distinct line style and symbol used on the plot, with the lines shown first, followed by the symbols. The second column will consist of user-specified text strings describing the significance of each line style and/or symbol. These text strings are specified by the input character array `ALEG`. A header may also be written above each column. Its contents are specified by the input character array `HDLEG`. The character size for the text strings in the legend is set using the parameter `ISIZEG`.

The location of the legend is specified by the input parameters `XLEG` and `YLEG`. The flag `ILEGND` is used to specify that a legend is to be included, and to determine whether its location is being specified in relative or user's units. A box may be drawn outlining the legend table, as specified by the input flag `ILEGBX`.

Normally, the area of the plot in which the legend is to appear is masked out before the legend is written. Material behind the legend, such as curves, axes, grid lines, etc., will thus be hidden. This behavior may be changed using the flag `IMASK(3)`.

[3]The input described in Section 3.1.6 also describes how to create a more general legend, with more than two columns.

2.9 Colors

Colors may be specified independently for any element of the plot — individual curves and/or symbol sets, axes, grid lines, titles, labels, etc. Some colors (`BLACK`, `RED`, `GREEN`, `YELLOW`, `BLUE`, `MAGENTA`, `CYAN`, and `WHITE`) may be specified by name using an input character variable. Other colors may be used by specifying the RGB triplet defining the color. In the RGB triplet, each of the values must be in the range 0.0–1.0, where 0.0 represents no contribution of the color, and 1.0 represents maximum intensity of the color. E.g., setting the RGB values to 1.0, 0.0, 0.0, results in red, and 1.0, 0.0, 1.0, results in magenta. The colors resulting from various combinations of RGB values, in 0.1 increments, are shown in Table 3.[4]

[4]Obviously, these pages must be printed on a color PostScript printer to be meaningful. A separate four-page PostScript file, *colors.ps*, containing this table is also distributed with *plotc*.

Table 1: Octal Codes for Characters in the Standard Fonts

Code	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*
00x	■	■	■	■	■	■	■	■
01x	■	■	■	■	■	■	■	■
02x	■	■	■	■	■	■	■	■
03x	■	■	■	■	■	■	■	■
04x		!	"	#	$	%	&	’
05x	(	)	*	+	,	-	.	/
06x	0	1	2	3	4	5	6	7
07x	8	9	:	;	<	=	>	?
10x	@	A	B	C	D	E	F	G
11x	H	I	J	K	L	M	N	O
12x	P	Q	R	S	T	U	V	W
13x	X	Y	Z	[	\	]	^	_
14x	‘	a	b	c	d	e	f	g
15x	h	i	j	k	l	m	n	o
16x	p	q	r	s	t	u	v	w
17x	x	y	z	{	|	}	~	■
20x	■	■	■	■	■	■	■	■
21x	■	■	■	■	■	■	■	■
22x	■	■	■	■	■	■	■	■
23x	■	■	■	■	■	■	■	■
24x	■	¡	¢	£	⁄	¥	ƒ	§
25x	¤	'	“	«	‹	›	ﬁ	ﬂ
26x	□	–	†	‡	·	□	¶	•
27x	‚	„	”	»	…	‰	□	¿
30x	□	`	´	ˆ	˜	¯	˘	˙
31x	¨	□	˚	¸	□	˝	˛	ˇ
32x	—	□	□	□	□	□	□	□
33x	□	□	□	□	□	□	□	□
34x	□	Æ	□	ª	□	□	□	□
35x	Ł	Ø	Œ	º	□	□	□	□
36x	□	æ	□	□	□	ı	□	□
37x	ł	ø	œ	ß	□	□	□	■

■	ASCII control character
□	not assigned

Table 2: Octal Codes for Characters in the Symbol Font

Code	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*
40x	▨	▨	▨	▨	▨	▨	▨	▨
41x	▨	▨	▨	▨	▨	▨	▨	▨
42x	▨	▨	▨	▨	▨	▨	▨	▨
43x	▨	▨	▨	▨	▨	▨	▨	▨
44x		!	∀	#	∃	%	&	∍
45x	(	)	∗	+	,	−	.	/
46x	0	1	2	3	4	5	6	7
47x	8	9	:	;	<	=	>	?
50x	≅	Α	Β	Χ	Δ	Ε	Φ	Γ
51x	Η	Ι	ϑ	Κ	Λ	Μ	Ν	Ο
52x	Π	Θ	Ρ	Σ	Τ	Υ	ς	Ω
53x	Ξ	Ψ	Ζ	[	∴	]	⊥	_
54x	‾	α	β	χ	δ	ε	ϕ	γ
55x	η	ι	φ	κ	λ	μ	ν	ο
56x	π	θ	ρ	σ	τ	υ		ω
57x	ξ	ψ	ζ	{	|	}	~	▨
60x	▨	▨	▨	▨	▨	▨	▨	▨
61x	▨	▨	▨	▨	▨	▨	▨	▨
62x	▨	▨	▨	▨	▨	▨	▨	▨
63x	▨	▨	▨	▨	▨	▨	▨	▨
64x	▨			≤	⁄	∞	ƒ	♣
65x	♦	♥	♠	↔	←	↑	→	↓
66x	°	±		≥	×	∝	∂	•
67x	÷	≠	≡	≈	…	⏐	⎯	↵
70x	ℵ	ℑ	ℜ	℘	⊗	⊕	∅	∩
71x	∪	⊃	⊇	⊄	⊂	⊆	∈	∉
72x	∠	∇	®	©	™	∏	√	⋅
73x	¬	∧	∨	⇔	⇐	⇑	⇒	⇓
74x	◊	〈	®	©	™	∑	⎛	⎜
75x	⎝	⎡	⎢	⎣	⎧	⎨	⎩	⎪
76x	░	〉	∫	⌠	⎮	⌡	⎞	⎟
77x	⎠	⎤	⎥	⎦	⎫	⎬	⎭	▨

▨	ASCII control character
░	not assigned

Table 3: RGB Color Combinations

RGB	*0.0*	*0.1*	*0.2*	*0.3*	*0.4*	*0.5*	*0.6*	*0.7*	*0.8*	*0.9*	*1.0*
0.0 0.0 x	■	■	■	■	■	■	■	■	■	■	■
0.0 0.1 x	■	■	■	■	■	■	■	■	■	■	■
0.0 0.2 x	■	■	■	■	■	■	■	■	■	■	■
0.0 0.3 x	■	■	■	■	■	■	■	■	■	■	■
0.0 0.4 x	■	■	■	■	■	■	■	■	■	■	■
0.0 0.5 x	■	■	■	■	■	■	■	■	■	■	■
0.0 0.6 x	■	■	■	■	■	■	■	■	■	■	■
0.0 0.7 x	■	■	■	■	■	■	■	■	■	■	■
0.0 0.8 x	■	■	■	■	■	■	■	■	■	■	■
0.0 0.9 x	■	■	■	■	■	■	■	■	■	■	■
0.0 1.0 x	■	■	■	■	■	■	■	■	■	■	■
0.1 0.0 x	■	■	■	■	■	■	■	■	■	■	■
0.1 0.1 x	■	■	■	■	■	■	■	■	■	■	■
0.1 0.2 x	■	■	■	■	■	■	■	■	■	■	■
0.1 0.3 x	■	■	■	■	■	■	■	■	■	■	■
0.1 0.4 x	■	■	■	■	■	■	■	■	■	■	■
0.1 0.5 x	■	■	■	■	■	■	■	■	■	■	■
0.1 0.6 x	■	■	■	■	■	■	■	■	■	■	■
0.1 0.7 x	■	■	■	■	■	■	■	■	■	■	■
0.1 0.8 x	■	■	■	■	■	■	■	■	■	■	■
0.1 0.9 x	■	■	■	■	■	■	■	■	■	■	■
0.1 1.0 x	■	■	■	■	■	■	■	■	■	■	■
0.2 0.0 x	■	■	■	■	■	■	■	■	■	■	■
0.2 0.1 x	■	■	■	■	■	■	■	■	■	■	■
0.2 0.2 x	■	■	■	■	■	■	■	■	■	■	■
0.2 0.3 x	■	■	■	■	■	■	■	■	■	■	■
0.2 0.4 x	■	■	■	■	■	■	■	■	■	■	■
0.2 0.5 x	■	■	■	■	■	■	■	■	■	■	■
0.2 0.6 x	■	■	■	■	■	■	■	■	■	■	■
0.2 0.7 x	■	■	■	■	■	■	■	■	■	■	■
0.2 0.8 x	■	■	■	■	■	■	■	■	■	■	■
0.2 0.9 x	■	■	■	■	■	■	■	■	■	■	■
0.2 1.0 x	■	■	■	■	■	■	■	■	■	■	■

Continued on next page

Table 3: RGB Color Combinations (*Continued*)

RGB	*0.0*	*0.1*	*0.2*	*0.3*	*0.4*	*0.5*	*0.6*	*0.7*	*0.8*	*0.9*	*1.0*
0.3 0.0 x	⬛	⬛	⬛	⬛	🟪	🟪	🟪	🟪	🟪	🟦	🟦
0.3 0.1 x	⬛	⬛	⬛	⬛	🟪	🟪	🟪	🟪	🟪	🟦	🟦
0.3 0.2 x	🟫	🟫	⬛	⬛	🟪	🟪	🟪	🟪	🟦	🟦	🟦
0.3 0.3 x	🟫	🟫	⬛	⬛	⬛	🟪	🟦	🟦	🟦	🟦	🟦
0.3 0.4 x	🟩	🟩	🟩	⬛	⬛	🟦	🟦	🟦	🟦	🟦	🟦
0.3 0.5 x	🟩	🟩	🟩	🟩	🟩	🟦	🟦	🟦	🟦	🟦	🟦
0.3 0.6 x	🟩	🟩	🟩	🟩	🟩	🟩	🟦	🟦	🟦	🟦	🟦
0.3 0.7 x	🟩	🟩	🟩	🟩	🟩	🟩	🟦	🟦	🟦	🟦	🟦
0.3 0.8 x	🟩	🟩	🟩	🟩	🟩	🟩	🟦	🟦	🟦	🟦	🟦
0.3 0.9 x	🟩	🟩	🟩	🟩	🟩	🟩	🟦	🟦	🟦	🟦	🟦
0.3 1.0 x	🟩	🟩	🟩	🟩	🟩	🟩	🟦	🟦	🟦	🟦	🟦
0.4 0.0 x	🟥	🟥	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪
0.4 0.1 x	🟥	🟥	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪
0.4 0.2 x	🟫	🟫	🟫	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪
0.4 0.3 x	🟫	🟫	🟫	🟫	🟪	🟪	🟪	🟪	🟪	🟪	🟪
0.4 0.4 x	🟫	🟫	🟫	⬛	⬛	🟪	🟪	🟪	🟦	🟦	🟦
0.4 0.5 x	🟩	🟩	🟩	🟩	🟩	⬛	🟦	🟦	🟦	🟦	🟦
0.4 0.6 x	🟩	🟩	🟩	🟩	🟩	🟩	🟦	🟦	🟦	🟦	🟦
0.4 0.7 x	🟩	🟩	🟩	🟩	🟩	🟩	🟦	🟦	🟦	🟦	🟦
0.4 0.8 x	🟩	🟩	🟩	🟩	🟩	🟩	🟦	🟦	🟦	🟦	🟦
0.4 0.9 x	🟩	🟩	🟩	🟩	🟩	🟩	🟦	🟦	🟦	🟦	🟦
0.4 1.0 x	🟩	🟩	🟩	🟩	🟩	🟩	🟦	🟦	🟦	🟦	🟦
0.5 0.0 x	🟥	🟥	🟥	🟥	🟪	🟪	🟪	🟪	🟪	🟪	🟪
0.5 0.1 x	🟥	🟥	🟥	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪
0.5 0.2 x	🟧	🟫	🟫	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪
0.5 0.3 x	🟫	🟫	🟫	🟫	🟪	🟪	🟪	🟪	🟪	🟪	🟪
0.5 0.4 x	🟫	🟫	🟫	🟫	🟫	🟪	🟪	🟪	🟪	🟪	🟪
0.5 0.5 x	🟨	🟨	🟫	🟫	⬜	⬜	⬜	🟪	🟪	🟦	🟦
0.5 0.6 x	🟩	🟩	🟩	🟩	🟩	🟩	⬜	🟦	🟦	🟦	🟦
0.5 0.7 x	🟩	🟩	🟩	🟩	🟩	🟩	🟦	🟦	🟦	🟦	🟦
0.5 0.8 x	🟩	🟩	🟩	🟩	🟩	🟩	🟦	🟦	🟦	🟦	🟦
0.5 0.9 x	🟩	🟩	🟩	🟩	🟩	🟩	🟦	🟦	🟦	🟦	🟦
0.5 1.0 x	🟩	🟩	🟩	🟩	🟩	🟩	🟦	🟦	🟦	🟦	🟦

Continued on next page

Table 3: RGB Color Combinations (*Continued*)

RGB	*0.0*	*0.1*	*0.2*	*0.3*	*0.4*	*0.5*	*0.6*	*0.7*	*0.8*	*0.9*	*1.0*
0.6 0.0 x	🟥	🟥	🟥	🟥	🟪	🟪	🟪	🟪	🟪	🟪	🟪
0.6 0.1 x	🟥	🟥	🟥	🟥	🟪	🟪	🟪	🟪	🟪	🟪	🟪
0.6 0.2 x	🟫	🟫	🟥	🟥	🟪	🟪	🟪	🟪	🟪	🟪	🟪
0.6 0.3 x	🟫	🟫	🟫	🟫	🟪	🟪	🟪	🟪	🟪	🟪	🟪
0.6 0.4 x	🟫	🟫	🟫	🟫	🟫	🟪	🟪	🟪	🟪	🟪	🟪
0.6 0.5 x	🟫	🟫	🟫	🟫	🟫	🟫	🟪	🟪	🟪	🟪	🟪
0.6 0.6 x	🟩	🟩	🟩	🟩	⬜	⬜	⬜	⬜	🟦	🟦	🟦
0.6 0.7 x	🟩	🟩	🟩	🟩	🟩	🟩	⬜	⬜	🟦	🟦	🟦
0.6 0.8 x	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟦	🟦	🟦
0.6 0.9 x	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟦	🟦	🟦
0.6 1.0 x	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟦	🟦	🟦
0.7 0.0 x	🟥	🟥	🟥	🟥	🟪	🟪	🟪	🟪	🟪	🟪	🟪
0.7 0.1 x	🟥	🟥	🟥	🟥	🟪	🟪	🟪	🟪	🟪	🟪	🟪
0.7 0.2 x	🟥	🟥	🟥	🟥	🟪	🟪	🟪	🟪	🟪	🟪	🟪
0.7 0.3 x	🟧	🟥	🟥	🟥	🟪	🟪	🟪	🟪	🟪	🟪	🟪
0.7 0.4 x	🟧	🟧	🟧	🟫	🟫	🟪	🟪	🟪	🟪	🟪	🟪
0.7 0.5 x	🟧	🟧	🟧	🟫	🟫	🟫	🟪	🟪	🟪	🟪	🟪
0.7 0.6 x	🟨	🟨	🟫	🟫	🟫	🟫	🟫	🟪	🟪	🟪	🟪
0.7 0.7 x	🟨	🟨	🟨	🟨	🟨	⬜	⬜	⬜	🟦	🟦	🟦
0.7 0.8 x	🟩	🟩	🟩	🟩	🟩	🟩	🟩	⬜	🟦	🟦	🟦
0.7 0.9 x	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟦	🟦	🟦
0.7 1.0 x	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟦	🟦	🟦
0.8 0.0 x	🟥	🟥	🟥	🟥	🟥	🟪	🟪	🟪	🟪	🟪	🟪
0.8 0.1 x	🟥	🟥	🟥	🟥	🟥	🟪	🟪	🟪	🟪	🟪	🟪
0.8 0.2 x	🟥	🟥	🟥	🟥	🟥	🟪	🟪	🟪	🟪	🟪	🟪
0.8 0.3 x	🟧	🟥	🟥	🟥	🟥	🟪	🟪	🟪	🟪	🟪	🟪
0.8 0.4 x	🟧	🟧	🟧	🟥	🟥	🟥	🟪	🟪	🟪	🟪	🟪
0.8 0.5 x	🟧	🟧	🟧	🟧	🟥	🟥	🟪	🟪	🟪	🟪	🟪
0.8 0.6 x	🟧	🟧	🟧	🟧	🟧	🟫	🟫	🟪	🟪	🟪	🟪
0.8 0.7 x	🟨	🟨	🟨	🟨	🟨	🟫	🟫	🟫	🟪	🟪	🟪
0.8 0.8 x	🟨	🟨	🟨	🟨	🟨	🟨	⬜	⬜	⬜	🟦	🟦
0.8 0.9 x	🟨	🟨	🟨	🟩	🟩	🟩	🟩	🟩	⬜	🟦	🟦
0.8 1.0 x	🟨	🟨	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟦	🟦

Continued on next page

Table 3: RGB Color Combinations (*Continued*)

RGB	*0.0*	*0.1*	*0.2*	*0.3*	*0.4*	*0.5*	*0.6*	*0.7*	*0.8*	*0.9*	*1.0*
0.9 0.0 x	🟥	🟥	🟥	🟥	🟥	🟪	🟪	🟪	🟪	🟪	🟪
0.9 0.1 x	🟥	🟥	🟥	🟥	🟥	🟪	🟪	🟪	🟪	🟪	🟪
0.9 0.2 x	🟥	🟥	🟥	🟥	🟥	🟪	🟪	🟪	🟪	🟪	🟪
0.9 0.3 x	🟧	🟥	🟥	🟥	🟥	🟪	🟪	🟪	🟪	🟪	🟪
0.9 0.4 x	🟧	🟧	🟧	🟥	🟥	🟥	🟪	🟪	🟪	🟪	🟪
0.9 0.5 x	🟧	🟧	🟧	🟧	🟧	🟥	🟪	🟪	🟪	🟪	🟪
0.9 0.6 x	🟧	🟧	🟧	🟧	🟧	🟧	🟪	🟪	🟪	🟪	🟪
0.9 0.7 x	🟨	🟨	🟨	🟧	🟧	🟧	🟧	🟪	🟪	🟪	🟪
0.9 0.8 x	🟨	🟨	🟨	🟨	🟨	🟨	🟨	⬜	⬜	🟪	🟪
0.9 0.9 x	🟨	🟨	🟨	🟨	🟨	🟨	🟨	⬜	⬜	⬜	⬜
0.9 1.0 x	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	⬜	⬜	⬜
1.0 0.0 x	🟥	🟥	🟥	🟥	🟥	🟪	🟪	🟪	🟪	🟪	🟪
1.0 0.1 x	🟥	🟥	🟥	🟥	🟥	🟪	🟪	🟪	🟪	🟪	🟪
1.0 0.2 x	🟥	🟥	🟥	🟥	🟥	🟪	🟪	🟪	🟪	🟪	🟪
1.0 0.3 x	🟧	🟧	🟥	🟥	🟥	🟥	🟪	🟪	🟪	🟪	🟪
1.0 0.4 x	🟧	🟧	🟧	🟥	🟥	🟥	🟪	🟪	🟪	🟪	🟪
1.0 0.5 x	🟧	🟧	🟧	🟧	🟧	🟥	🟪	🟪	🟪	🟪	🟪
1.0 0.6 x	🟧	🟧	🟧	🟧	🟧	🟧	🟥	🟪	🟪	🟪	🟪
1.0 0.7 x	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟥	🟪	🟪	🟪
1.0 0.8 x	🟨	🟨	🟨	🟨	🟧	🟧	🟧	🟧	🟥	🟪	🟪
1.0 0.9 x	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	⬜	⬜	⬜
1.0 1.0 x	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	⬜	⬜	⬜

3 Input Description

This section describes in detail the input to the *plotc* program, and to the supplied versions of `COORS` and `POINTS`. It also describes the parameters that must be computed by user-written versions of `COORS` and `POINTS`. Input to *plotc* is through namelists `PTYPE` and `AXES`. Input to the supplied versions of `COORS` and `POINTS` is through various types of read statements. Input to any user-supplied version of `COORS` and/or `POINTS` is, of course, determined by the user. Multiple plots may be created with one input file by simply stacking sets of input data.

Except for the input to `COORS` and `POINTS`, all of the parameters have default values and do not need to be specified by the user unless some other value is desired. Unless stated otherwise, the type of the parameters (real or integer) follows standard Fortran convention. (I.e., those starting with `I`, `J`, `K`, `L`, `M`, or `N` are integer, and the remainder are real.)

3.1 Namelist `PTYPE`

This namelist is used to specify various things about the type of plot to be produced. An asterisk is used to mark those that a casual user is most likely to change from their default values.

3.1.1 Curves and Symbols

The following parameters deal with the number and type of curves and symbol sets to be plotted.

* `NCURV`	Number of curves to be plotted. If `NCURV` > 0, each curve will be drawn using the line style specified by the input parameter `ICTYPE(1)`. The default for `ICTYPE(1)` gives a solid line. If `NCURV` < 0, a different line style will be used for each curve, as specified by the input array `ICTYPE`. The sum `NCURV` + `NSYMB` must be 1000 or less. The default value for `NCURV` is 1.
* `NSYMB`	Number of sets of points to be plotted using symbols. If `NSYMB` > 0, each set will be plotted using the symbol specified by the input parameter `ISTYPE(1)`. The default for `ICTYPE(1)` gives a circle. If `NSYMB` < 0, a different symbol will be used for each set, as specified by the input array `ISTYPE`. The size of the symbol is specified by the parameter `ISIZE1`. The sum `NCURV` + `NSYMB` must be 1000 or less. The default value for `NSYMB` is 0.
`ISIZE1`	An array of `NSYMB` integers specifying the symbol sizes for the sets of points plotted using symbols. The size is in 1/72's of a relative unit for the OpenGL plot, and 1/72's of an inch for the PostScript plot. The maximum value allowed is 999. This parameter may also be specified for each set by subroutine `POINTS` (see Section 3.4 and Section 3.5). If so, the corresponding value in the namelist is ignored. The default values are all 10.
`ITYPE`	0 for a vector (i.e., line) plot. 1 for a point plot. 2 for a vector-symbol plot. This parameter applies to the curves being plotted. In a vector plot, each curve is drawn as a line, as described above for `NCURV`. In a point plot, each coordinate defining the curve is plotted as a single point. In a vector-symbol plot, each curve is drawn as a line, as described above for `NCURV`, but with symbols distributed

along the curve. A different symbol will be used for each curve, as specified by the input array `ISTYPV`. The defaults are: circle, square, triangle, diamond, inverted triangle, lower right quadrant circle, upper semicircle, rhombus, pentagon, and house. If more than ten curves are being plotted, this sequence is repeated. The size of the symbol is specified by the parameter `ISIZE`. The color of the symbol will be the same as the color of the curve. Note that for a purely symbol plot, you should set `NCURV` = 0 and use a non-zero value for `NSYMB`. The default value is 0.

`IFREQ` In a vector-symbol plot, a symbol will be plotted every `IFREQ` points. The default value is 10.

`ISIZE` Size to be used for the symbols on a vector-symbol curve. The size is in 1/72's of a relative unit for the OpenGL plot, and 1/72's of an inch for the PostScript plot. The maximum value allowed is 999. Note that in a vector-symbol plot, all curves will use the same size symbol. The default value is 10.

`ICTYPE` An array of integers defining the line styles to be used when plotting curves. The default values for `ICTYPE(1-5)` are 1, 2, 3, 4, 5, corresponding to the sequence solid, short-dashed, dot-dashed, long-dashed, and dotted. This sequence is repeated to give the default values for the rest of the array.

`CLRCRV` An array of character variables, specified as `CLRCRV(ICURV)`, defining the colors to be used for each curve. The subscript `ICURV` is the curve number. E.g., setting `CLRCRV = 'RED', 'CYAN'` in the input file results in the first curve being red, and the second being cyan. If specified, these colors override those set using `RGBCRV`, described below.

`RGBCRV` A two-dimensional array of RGB values, specified as `RGBCRV(I,ICURV)` defining the colors to be used for each curve. The first subscript varies from 1 to 3, for the red, green, and blue color components, respectively, and the second subscript is the curve number. E.g., setting `RGBCRV(1,1) = 1.0, 0.0, 0.0` and `RGBCRV(1,2) = 0.0, 1.0, 1.0` in the input file results in the first curve being red, and the second being cyan. The default color for the first curve is black, and the default color for the remaining curves is the first curve color. See Section 2.9 for a bit more information about specifying colors in *plotc*.

`ISTYPE` An array of integers defining the symbol types to be used when plotting sets of points as symbols. The default values for `ISTYPE(1-10)` are 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, corresponding to the sequence circle, square, triangle, diamond, inverted triangle, lower right quadrant circle, upper semicircle, rhombus, pentagon, and house. This sequence is repeated to give the default values for the rest of the array. If `ISTYPE` > 0, an open symbol is drawn. If `ISTYPE` < 0, a filled symbol is drawn.

`CLRSYM` An array of character variables, specified as `CLRSYM(ISYMB)`, defining the colors to be used for each symbol set. The subscript `ISYMB` is the symbol set number. If specified, these colors override those set using `RGBSYM`, described below.

`RGBSYM` A two-dimensional array of RGB values, specified as `RGBSYM(I,ISYMB)` defining the colors to be used for each symbol set. The first subscript varies from 1 to 3, for the red, green, and blue color components, respectively, and the second subscript is the symbol set number. The default color for the first symbol set is black, and the default color for the remaining symbol sets is the first symbol set color. See Section 2.9 for a bit more information about specifying colors in *plotc*.

`IEBAR`

0 for no error bars on symbols.
±1 for error bar lengths defined as a fraction of the y-coordinate value.
±2 for error bar lengths specified explicitly in user's units.

Positive values imply the error bar lengths above and below the symbol are the same. Negative values imply they're different. See Section 3.4 for details on how to specify the error bar lengths, and Section 4.2 for an example. The length of the endcaps on the error bars will be the same as the length of the tick marks on the axes, specified by the parameter `TKLEN` in namelist `AXES`. The color of the error bars will be the same as the color of the symbols. The default value for `IEBAR` is 0.

`ISTYPV`

An array of integers defining the symbol types to be used when plotting vector-symbol curves (`ITYPE` = 2). This array is used in exactly the same way as the `ISTYPE` array described above.

`ICLIP`

0 for no clipping.
1 to clip curves and symbols at the ends of the axes.

The default value is 0.

`IMASK(1)`

0 to allow material behind open symbols, such as curves, axes, etc., to show.
1 to mask out material behind open symbols.

The default value is 1.

`OFFSET`

Number of x-axis intervals each curve is to be offset in the x-direction. Note that this need not be an integer value. This option is not available if `ILOGX` = 1. If `OFFSET` > 0., the x-axis will automatically be extended to account for the offset of curves 2 through `NCURV`. The user should specify `XMAX`, `NINTX`, and `XLEN` (see namelist `AXES`) for the first curve and not worry about the offset. The default value is 0.0 (no offset.)

3.1.2 Coordinate Definition Procedure

These parameters determine how the curve and symbol set coordinates are being defined.

The following parameters determine how the coordinates of the curves are being specified when the supplied `COORS` routine is being used.[5] For more detail, see Section 3.3.

* `MODEC`

0 to read all the x-coordinates, followed by all the y-coordinates, with an `8E10.0` format.
1 to read only the y-coordinates, with equally spaced x-coordinates computed from input values of x_0 and Δx.
2 to read x-y coordinate pairs.
4 to read all the x-coordinates, followed by all the y-coordinates. Note that this is the same as the `MODE` = 0 option, but without the restriction of a fixed format.
20 to read two specified columns of a table as the x and y-coordinates.
30 to get the curve coordinates from a Genplot file.

The default value is 0, for compatibility with earlier versions of *plotc*.

[5]Users of the *plotxy* plotting package will note a remarkable similarity between some of these options and those in *plotxy*.

`IFILEC`	0 to read the coordinates of the curves from the namelist input file. 1 to read the coordinates from a separate file. The file name must be specified via the `-a` option when *plotc* is run. If `MODEC` = 30, `IFILEC` is automatically set to 1. The default value is 0.
`FMTXYC`	A character variable of up to 100 characters defining the format to be used for reading the coordinates of the curves. This can be: (1) the single character "`*`", denoting Fortran list-directed (i.e., free-form) input; (2) the single character "`b`" (or "`B`"), denoting Fortran unformatted input; or (3) an ordinary Fortran format specifier, enclosed in parentheses. If list-directed input is being used, the values must be separated by whitespace (i.e., spaces, tabs, and/or new line characters) and/or commas. If unformatted input is being used, the coordinates must be read from a separate file (i.e., `IFILEC` = 1). The default value is "`*`".
`ISKIPC`	The number of records to skip before starting to read the coordinates of the curves. This option is most likely to be useful when using the output from another program as input to *plotc*. The default value is 0.
`IREWC`	0 for no rewinding between curves. 1 to go back to beginning of the file for each new curve. This option only applies when `MODEC` = 20 and `IFILEC` = 1. With this option, coordinates for multiple curves may be specified from a single table. The default value is 0.

The following parameters determine how the coordinates of the symbol sets and the error bar dataare being specified when the supplied `POINTS` routine is being used. Except for the error bar data, these parameters are exactly analogous to those defined above for the curves being plotted. For more detail, see Section 3.4.

`* MODEP`	0 to read all the x-coordinates, followed by all the y-coordinates, followed by the error bar data, with an `8E10.0` format. 1 to read only the y-coordinates, followed by the error bar data, with equally spaced x-coordinates computed from input values of x_0 and Δx. 2 to read x-y coordinate pairs, followed by the error bar data. 4 to read all the x-coordinates, followed by all the y-coordinates, followed by the error bar data. Note that this is the same as the `MODE` = 0 option, but without the restriction of a fixed format. 20 to read specified columns of a table as the x and y-coordinates, and the error bar data. The default value is 0, for compatibility with earlier versions of *plotc*.
`IFILEP`	0 to read the coordinates of the symbol sets and the error bar data from the namelist input file. 1 to read the coordinates and the error bar data from a separate file. The file name must be specified via the `-b` option when *plotc* is run. The default value is 0.
`FMTXYP`	A character variable of up to 100 characters defining the format to be used for reading the coordinates of the symbol sets and the error bar data. This can be: (1) the single character "`*`", denoting Fortran list-directed (i.e., free-form) input;

(2) the single character "b" (or "B"), denoting Fortran unformatted input; or (3) an ordinary Fortran format specifier, enclosed in parentheses. If list-directed input is being used, the values must be separated by whitespace (i.e., spaces, tabs, and/or new line characters) and/or commas. If unformatted input is being used, the coordinates and error bar data must be read from a separate file (i.e., IFILEP = 1). The default value is "*".

ISKIPP — The number of records to skip before starting to read the coordinates of the symbol sets and the error bar data. This option is most likely to be useful when using the output from another program as input to *plotc*. The default value is 0.

IREWP —
0 for no rewinding between symbol sets.
1 to go back to beginning of the file for each new symbol set. This option only applies when MODEP = 20 and IFILEP = 1. With this option, coordinates for multiple symbol sets may be specified from a single table.

The default value is 0.

The following parameters may be used to modify the coordinates returned by the COORS and/or POINTS routines. Note that these apply to coordinates returned by user-written routines, as well as the supplied routines. They can be very handy when existing subroutines COORS and/or POINTS are being used, and you would like to modify the returned values. The x-coordinates may be transformed using the equation $x_{\text{plot}} = a_x x + b_x$. Similarly, the y-coordinates may be transformed using the equation $y_{\text{plot}} = a_y y + b_y$. And finally, the x and y coordinates may be switched.

AXTR — An array specifying the constants a_x. The first NCURV elements (i.e, AXTR(1) through AXTR(NCURV)) apply to the curves being plotted, and the following NSYMB elements (i.e., AXTR(NCURV+1) through AXTR(NCURV+NSYMB) apply to the symbol sets being plotted. The default values are all 1.0.

BXTR — An array specifying the constants b_x. The first NCURV elements (i.e, BXTR(1) through BXTR(NCURV)) apply to the curves being plotted, and the following NSYMB elements (i.e., BXTR(NCURV+1) through BXTR(NCURV+NSYMB) apply to the symbol sets being plotted. The default values are all 0.0.

AYTR — An array specifying the constants a_y. This is exactly analogous to AXTR. The default values are all 1.0.

BYTR — An array specifying the constants b_y. This is exactly analogous to BXTR. The default values are all 0.0.

NPLIM — The maximum number of points to use in defining a curve. If the number of points NP returned by subroutine COORS is greater than NPLIM, the points will be "thinned" by the minimum integer factor large enough to limit the number of points to NPLIM. The original first and last points will always be included. This parameter may be useful, for example, when the curve coordinates are defined by a machine-generated input file, such as a Genplot file using the MODEC = 30 option. The default is the value of the Fortran parameter NPMAX in the file *params.h*, which is 100,000 as distributed.

IREV —
1 to switch the x- and y-coordinates returned by subroutines COORS and POINTS. The switch is done *after* the coordinates have been transformed using the parameters AXTR, etc., described above. The x- and y-axis titles returned by subroutine COORS are also reversed. Note that this only affects the parameters

returned by `COORS` and `POINTS`. The parameters describing the axes in namelist `AXES` are not reversed.

0 for no switching.

The default value is 0.

3.1.3 The Plotting Frame

The following parameters control the placement of the plot on the plotting frame.

* `IROT`	0 to display plot in portrait mode. 1 to display plot in landscape mode. The default value is 0.
`XORGN`	The x-location on the plotting frame of the lower left corner of the plot, in relative units. The default is to center the plot horizontally on the frame, but with a minimum `XORGN` value of 1.5 to allow room for the y-axis labels and title.
`YORGN`	The y-location on the plotting frame of the lower left corner of the plot, in relative units. The default is to center the plot vertically on the frame, but with a minimum `YORGN` value of 1.0 to allow room for the x-axis labels and title.[6]
`PGHGT`	Paper height in inches for the PostScript plot. The default value is 11.0 in portrait mode, and 8.5 in landscape mode.
`PGWID`	Paper width in inches for the PostScript plot. The default value is 8.5 in portrait mode, and 11.0 in landscape mode.
`FHGT`	Frame height in relative units for the OpenGL plot, and in inches for the PostScript plot. The default value is 11.0 in portrait mode, and 8.5 in landscape mode.
`FWID`	Frame width in relative units for the OpenGL plot, and in inches for the PostScript plot. The default value is 8.5 in portrait mode, and 11.0 in landscape mode.[7]
`CLRBK`	An character variable defining the color to be used for the background. If specified, this color overrides one set using `RGBBK`, described below.
`RGBBK`	An array of three RGB values defining the color to be used for the background. For the OpenGL version of the plot, the default is a white background; for the PostScript plot, the default is to not fill the background at all.[8] See Section 2.9 for a bit more information about specifying colors in *plotc*.

[6]If `IYAXIS` = 3 in namelist `AXES`, suppressing the plotting of the y-axis and its labels, defaulting `XORGN` will center the plot horizontally with no minimum `XORGN` value. Similarly, if `IXAXIS` = 3, defaulting `YORGN` will center the plot vertically with no minimum `YORGN` value.

[7]Note that the values of `FHGT` and `FWID` do not affect the physical size of the window on the graphics monitor. The size of the window depends on the size of the monitor. The shape is determined by the ratio `FWID`/`FHGT`. The PostScript plot is printed on paper measuring `PGWID` $\times$ `PGHGT` inches. If necessary, the PostScript plot will be rescaled to fit on the page.

[8]Note, however, that when the `IMASK` input parameters indicate that material behind symbols, strings, and the legend is to be masked out, the area is actually filled with the background color. If no background color is specified, the area is filled with white. This may not be desirable if the PostScript plot is printed on non-white paper.

3.1.4 Text String Characteristics

The following parameters determine the character fonts and sizes to be used for titles, labels, etc.

`* ITITLE`	A 3-element array, specified as `ITITLE(I)`, indicating whether or not titles should be included on the plot. The subscript `I` = 1 for the plot title, 2 for the x-axis title, and 3 for the y-axis title. The font to be used is specified by `FONTS(2)`. Valid values for `ITITLE(I)` are: 0 for no title. 1 to include a title. `ITITLE(1)` may be specified as ±1. A value of +1 puts the plot title at the top of the plot, and −1 puts it at the top of the frame. The default values are 0, 1, 1.
`TALIGN`	An array of character variables, given as `TALIGN(I)`, specifying how the plot and axis titles are to be aligned. The subscript `I` = 1 for the plot title, 2 for the x-axis title, and 3 for the y-axis title. Valid values are `LEFT`, `CENTER`, or `RIGHT`, resulting in left-aligned, centered, and right-aligned titles, respectively. All alignment will be with respect to the plot, with one exception. If `ITITLE(1)` = −1, the plot title will be aligned with respect to the frame. The default values are `LEFT`, `CENTER`, `CENTER`, resulting in a left-aligned plot title, and centered x- and y-axis titles.
`FONTS`	An array of character variables, given as `FONTS(I)`, specifying the fonts to be used on the plot.[9] The subscript `I` = 1 for the axis labels, 2 for the axis and plot titles, 3 for the legend, and 4 for user-specified text strings. Valid values for `FONTS(I)` are: `H` Helvetica. `HB` Helvetica Bold. `HO` Helvetica Oblique. `HD` Helvetica Bold Oblique. `R` Times Roman. `B` Times Bold. `I` Times Italic. `BI` Times Bold Italic. `C` Courier. `CB` Courier Bold. `CO` Courier Oblique. `CD` Courier Bold Oblique. `S` Symbol. The default values are all `H`.
`* ISIZET`	Size of characters in axis and plot titles. The size is in 1/72's of a relative unit for the OpenGL plot, and 1/72's of an inch for the PostScript plot.[10] The default value is 18.
`* ISIZEL`	Size of characters in axis labels. The size is in 1/72's of a relative unit for the OpenGL plot, and 1/72's of an inch for the PostScript plot.[11] The default value is 16.

[9] This parameter is currently supported only for the PostScript version of the plot.
[10] This parameter is currently supported only for the PostScript version of the plot.
[11] This parameter is currently supported only for the PostScript version of the plot.

3.1.5 Text Strings at Arbitrary Locations

The following parameters may be used to write strings of text at user-specified locations on the plot. Up to 50 strings may be written. The maximum number of characters allowed in each string, including those used to specify special characters, subscripts, etc., is 100. The font to be used is specified by `FONTS(4)`.

`NSTR`	1 if the values of `XSTR` and `YSTR`, described below, are in user's units. −1 if the values of `XSTR` and `YSTR` are in relative units. The default value is 1.
`XSTR`	An array, given as `XSTR(ISTR)`, specifying the x-locations of the strings. The subscript `ISTR` is the string number. `XSTR` may be specified in either user's or relative units, depending on the value of `NSTR`. The default values are all 0.
`YSTR`	An array, given as `YSTR(ISTR)`, specifying the y-location of the baseline of the strings. The subscript `ISTR` is the string number. `YSTR` may be specified in either user's or relative units, depending on the value of `NSTR`. The default values are all 0.
`ASTR`	An array of character variables, given as `ASTR(ISTR)`, specifying the contents of the strings. The subscript `ISTR` is the string number. Each string must be 100 characters or less. The default values are all blanks.
`ISIZES`	An array, given as `ISIZES(ISTR)`, specifying the sizes of the characters to be used in the strings. The subscript `ISTR` is the string number. The size is in 1/72's of a relative unit for the OpenGL plot, and 1/72's of an inch for the PostScript plot.[12] The default values are all 12.
`CLRSTR`	An array of character variables, specified as `CLRSTR(ISTR)`, defining the colors to be used for each string. The subscript `ISTR` is the string number. If specified, these colors override those set using `RGBSTR`, described below.
`RGBSTR`	A two-dimensional array of RGB values, specified as `RGBSTR(I,ISTR)` defining the colors to be used for each string. The first subscript varies from 1 to 3, for the red, green, and blue color components, respectively, and the second subscript is the string number. The default for the first string is the color of the x-axis title; the default for the remaining strings is the color of the first string. See Section 2.9 for a bit more information about specifying colors in *plotc*.
`SALIGN`	An array of character variables, given as `SALIGN(ISTR)`, specifying how strings are to be aligned with respect to the coordinates given by `XSTR` and `YSTR`. The subscript `ISTR` is the string number. Valid values are `LEFT`, `CENTER`, or `RIGHT`, resulting in left-aligned, centered, and right-aligned strings, respectively. The default values are all `LEFT`.
`IMASK(2)`	0 to allow material behind the strings, such as curves, axes, etc., to show. 1 to mask out material behind the strings. The default value is 1.

[12]This parameter is currently supported only for the PostScript version of the plot.

3.1.6 Legend Table

The following parameters control the placement and contents of a legend table on the plot. The font to be used is specified by `FONTS(3)`.

* `ILEGND`	0	for no legend.
	±1	for a standard two-column legend. The first column will list the distinct types of curves that were plotted, followed by the distinct types of symbols that were plotted. In determining "distinctness," curves and symbols drawn with different colors are considered distinct. In addition, two sets of the same symbol plotted at different sizes are considered distinct. The user must specify the contents of the second column in the input variable `ALEG`, described below.
	±2	for a generalized legend. The user must specify the contents of each column in the input variable `ALEG`, described below.
		If `ILEGND` > 0, the values of `XLEG` and `YLEG`, described below, must be in user's units. If `ILEGND` < 0, they must be in relative units. The default value is 0.
`XLEG`		The x-location of the upper left corner of the legend table, in either user's or relative units, depending on the sign of `ILEGND`. The default is to put the legend table in the upper right corner of the plot if `ILEGND` > 0, or the frame if `ILEGND` < 0.
`YLEG`		The y-location of the upper left corner of the legend table, in either user's or relative units, depending on the sign of `ILEGND`. The default is to put the legend table in the upper right corner of the plot if `ILEGND` > 0, or the frame if `ILEGND` < 0.
`NLEGC`		Number of columns in the legend table. This parameter is only needed if `ILEGND` $= \pm 2$. The maximum allowed value is 10. The default value is 2.
`NLEGR`		Number of rows in the legend table, not counting the header row. The maximum allowed value is 50. This parameter is only needed if `ILEGND` $= \pm 2$. There is no default value.
`HDLEG`		An array of character variables, specified as `HDLEG(ILEGC)`, giving the header for column `ILEGC` in the legend table. The subscript `ILEGC` thus varies from 1 to `NLEGC`. Each header must be 100 characters or less. The default values are all blanks.
`CLRLHD`		A character variable defining the color to be used for the headers. If specified, this color overrides one set using `RGBLHD`, described below.
`RGBLHD`		An array of three RGB values defining the color to be used for the headers. The default is the color of the x-axis title. See Section 2.9 for a bit more information about specifying colors in *plotc*.
`ALEG`		An array of character variables, specified as `ALEG(ILEGR,ILEGC)`, giving the contents of row `ILEGR`, column `ILEGC`, in the legend table. The subscript `ILEGR` thus varies from 1 to `NLEGR`, and `ILEGC` varies from 1 to `NLEGC`. Within each column, each element must be 100 characters or less. If `ILEGND` $= \pm 1$, only `ALEG(ILEGR,2)` (i.e., the contents of the second column) should be specified. Lines and symbols representing the distinct types of curves and symbols being plotted will automatically be drawn in column one.

If `ILEGND` = ±2, and if `ALEG(ILEGR,ILEGC)` = `'\CRV`*n*`'` or `'\crv`*n*`'`, where n is a 1 or 2 digit integer, a straight line will be plotted in row `ILEGR`, column `ILEGC`, representing curve number n. Similarly, if `ILEGND` = ±2, and if `ALEG(ILEGR,ILEGC)` = `'\SYM`*n*`'` or `'\sym`*n*`'`, where n is a 1 or 2 digit integer, a symbol will be plotted in row `ILEGR`, column `ILEGC`, representing symbol set number n.

The default values are all blanks.

`CLRLTX` An array of character variables, specified as `CLRLTX(ILEGR)`, defining the colors to be used for the *text* in each row of the legend.[13] The subscript `ILEGR` is the row number. If specified, these colors override those set using `RGBLTX`, described below.

`RGBLTX` A two-dimensional array of RGB values, specified as `RGBLTX(I,ILEGR)` defining the colors to be used for the text in each row of the legend. The first subscript varies from 1 to 3, for the red, green, and blue color components, respectively, and the second subscript is the row number. The default is the color of the legend header. See Section 2.9 for a bit more information about specifying colors in *plotc*.

`ILCLR` For the standard two-column legend only, setting `ILCLR = 1` causes the text in the second column to automatically be colored the same as the corresponding line or symbol in the first column. The default value is 0.

`ISIZEG` Size of characters in the legend. The size is in 1/72's of a relative unit for the OpenGL plot, and 1/72's of an inch for the PostScript plot.[14] The default value is 12.

`ILEGBX`
- 0 for a legend without an outlining box.
- 1 to draw an outlining box around the legend table, with vertical lines between the columns and a horizontal line below the header.
- −1 to draw an outlining box around the legend table, without lines between the columns or below the header.

The default value is 0.

`CLRLBX` A character variable defining the color to be used for the outlining box. If specified, this color overrides one set using `RGBLBX`, described below.

`RGBLBX` An array of three RGB values defining the color to be used for the outlining box. The default is the color of the x- and y-axes. See Section 2.9 for a bit more information about specifying colors in *plotc*.

`IMASK(3)`
- 0 to allow material in the legend area, such as curves, axes, etc., to show.
- 1 to mask out material in the legend area.

The default value is 1.

3.1.7 Output Devices

The following parameters deal with the type of output device(s) to be used.

[13] Lines and symbols in the legend representing curves and symbol sets, for both the standard two-column legend and the generalized legend, will automatically be drawn using the colors of the corresponding curves and symbol sets.

[14] The maximum value allowed is 999. This parameter is currently supported only for the PostScript version of the plot.

`IDEV`
1 to display each plot in a window on the graphics monitor using OpenGL routines.
2 to create PostScript versions of each plot, suitable for printing on a PostScript printer.

More than one device may be specified by setting `IDEV` equal to the sum of the values for the desired devices. The default value is 3, corresponding to both OpenGL and PostScript output.

3.1.8 Debugging

The following parameter may be useful when nothing seems to work right.

`IDEBUG` A 20-element array used to turn on debug printout. This output is printed on Fortran unit 30. Except where noted, set `IDEBUG(I)` = 1 for printout number `I`. Those currently available are as follows:

Number `I`	*Printout*
1	Namelist `PTYPE` and `AXES` input.
2	Coordinates of each curve being plotted.
3	Coordinates of the `IDEBUG(3)`'th curve (only if `IDEBUG(2)` = 0).
4	Coordinates of each set of points being plotted as symbols.
5	Coordinates of the `IDEBUG(5)`'th set of points (only if `IDEBUG(4)` = 0).
6	The input to subroutines `LABLGL` and `LABLPS`.
7	The input to subroutines `PLOTGL` and `PLOTPS`.
8	The input to subroutines `XAXGL` and `XAXPS`.
9	The input to subroutines `YAXGL` and `YAXPS`.
11	Parameters used for plots with `OFFSET` > 0.

The default values are all 0.

3.2 Namelist `AXES`

The parameters in this namelist describe the types of axes to be used. An asterisk is used to mark those that a casual user is most likely to change from their default values.

`* ISCALE`
0 to get axis scales automatically.
1 to read in scales for both the x- and y-axes.
2 to read in x-axis scales and get y-axis scales automatically.
3 to read in y-axis scales and get x-axis scales automatically.
4 same as 0, but also computes the length of the x-axis `XLEN`, overriding any specified value, to make the x- and y-scales the same (e.g., for non-distorted geometry plots.) If necessary, the plot will be reduced to fit within the frame width.

The default value is 0.

`CLRAX`	A character variable defining the color to be used for the axes. If specified, this color overrides one set using `RGBAX`, described below.
`RGBAX`	An array of three RGB values defining the color to be used for the axes. The default is black. See Section 2.9 for a bit more information about specifying colors in *plotc*.
* `XMIN`	Starting value for the x-axis, in user's units. Not needed if `ISCALE` = 0, 3, or 4. The default value is 0.
* `XMAX`	Ending value for the x-axis, in user's units. Not needed if `ISCALE` = 0, 3, or 4. The default value is 1.
* `YMIN`	Starting value for the y-axis, in user's units. Not needed if `ISCALE` = 0, 2, or 4. The default value is 0.
* `YMAX`	Ending value for the y-axis, in user's units. Not needed if `ISCALE` = 0, 2, or 4. The default value is 1.
* `NINTX`	Number of x-axis intervals, or log cycles if `ILOGX` = 1. The default value is 5.
* `NINTY`	Number of y-axis intervals, or log cycles if `ILOGY` = 1.[15] The default value is 5.
* `XLEN`	Length of x-axis in relative units. If `XLEN` > `FWID`, the plot will extend over more than one frame. The default value is the number of intervals, `NINTX`.
* `YLEN`	Length of y-axis in relative units. `YLEN` must be less than or equal to `FHGT`. The default value is the number of intervals, `NINTY`.
* `XTITLE`	A character variable specifying the title for the x-axis (up to 100 characters.) If specified, this overrides the x-axis title supplied by subroutine `COORS`. The default value is all blanks.
`CLRXT`	A character variable defining the color to be used for the x-axis title. If specified, this color overrides one set using `RGBXT`, described below.
`RGBXT`	An array of three RGB values defining the color to be used for the x-axis title. The default is the color of the axes. See Section 2.9 for a bit more information about specifying colors in *plotc*.
* `YTITLE`	A character variable specifying the title for the y-axis (up to 100 characters.) If specified, this overrides the y-axis title supplied by subroutine `COORS`. The default value is all blanks.
`CLRYT`	A character variable defining the color to be used for the y-axis title. If specified, this color overrides one set using `RGBYT`, described below.
`RGBYT`	An array of three RGB values defining the color to be used for the y-axis title. The default is the color of the x-axis title. See Section 2.9 for a bit more information about specifying colors in *plotc*.
* `PTITLE`	A character variable specifying the title for the top of the plot (up to 100 characters.) If specified, this overrides the plot title supplied by subroutine `COORS`. The default value is all blanks.

[15] For logarithmic axes, the number of intervals and/or the starting and ending values for the axes may be modified by *plotc* to be self-consistent.

`CLRPT`	A character variable defining the color to be used for the plot title. If specified, this color overrides one set using `RGBPT`, described below.
`RGBPT`	An array of three RGB values defining the color to be used for the plot title. The default is the color of the x-axis title. See Section 2.9 for a bit more information about specifying colors in *plotc*.
`IXAXIS`	0 to plot the x-axis without tick marks or grid lines. 1 to plot the x-axis with tick marks above the axis. -1 to plot the x-axis with tick marks below the axis. 2 to plot the x-axis with grid lines. 3 to suppress plotting of the x-axis. In the PostScript plot, interior grid lines will be drawn in a lighter color. The default value is 1.
`IYAXIS`	0 to plot the y-axis without tick marks or grid lines. 1 to plot the y-axis with tick marks to the right of the axis. -1 to plot the y-axis with tick marks to the left of the axis. 2 to plot the y-axis with grid lines. 3 to suppress plotting of the y-axis. In the PostScript plot, interior grid lines will be drawn in a lighter color.
`TKLEN`	Length of tick marks and error bar endcaps in relative units. The default value is 0.1.
`CLRGRD`	A character variable defining the color to be used for tick marks or grid lines. If specified, this color overrides one set using `RGBGRD`, described below.
`RGBGRD`	An array of three RGB values defining the color to be used for tick marks or grid lines. The default is the color of the axes. See Section 2.9 for a bit more information about specifying colors in *plotc*.
`GRAYGR`	Interior grid lines are by default drawn in a lighter version of the color specified by `CLRGRD` or `RGBGRD`, according to the formula `rgb = rgbgrd + graygr*(1.-rgbgrd)` which is applied to each of the RGB values. The parameter `GRAYGR` thus specifies the “lightness” factor. Valid values range from 0.0 (interior grid lines will not be lightened at all) to 1.0 (interior grid lines will be white). The default value is 0.8.
`IFRAME`	0 for no frame. 1 to put a frame around the plot by drawing horizontal lines at `YMIN` and `YMAX`, and vertical lines at `XMIN` and `XMAX`. 2 to put a frame around the plot, with tick marks added. The frame will be drawn in the same color as the axes. The default value is 0.
`ILOGX`	0 for a standard x-axis. 1 for a logarithmic x-axis. The default value is 0.
`ILOGY`	0 for a standard y-axis. 1 for a logarithmic y-axis. The default value is 0.

`INOTX`	0 to use floating or fixed notation (depending on the values) for the x-axis labels. 1 to use scientific notation for the x-axis labels. For standard axes, the default value is 0. Logarithmic axes automatically use scientific notation.
`INOTY`	0 to use floating or fixed notation (depending on the values) for the y-axis labels. 1 to use scientific notation for the y-axis labels. For standard axes, the default value is 0. Logarithmic axes automatically use scientific notation.
`NLABX`	Number of numeric labels to be placed on the x-axis.[16] The font to be used is specified by `FONTS(1)`. The maximum value is 51. The default value is `NINTX` +1 (i.e., each tick mark or grid line is labeled.)
`CLRXLB`	A character variable defining the color to be used for the x-axis labels. If specified, this color overrides one set using `RGBXLB`, described below.
`RGBXLB`	An array of three RGB values defining the color to be used for the x-axis labels. The default is the color of the axes. See Section 2.9 for a bit more information about specifying colors in *plotc*.
`NLABY`	Number of numeric labels to be placed on the y-axis. The font to be used is specified by `FONTS(1)`. The maximum value is 51. The default value is `NINTY` +1 (i.e., each tick mark or grid line is labeled.)
`CLRYLB`	A character variable defining the color to be used for the y-axis labels. If specified, this color overrides one set using `RGBYLB`, described below.
`RGBYLB`	An array of three RGB values defining the color to be used for the y-axis labels. The default is the color of the x-axis labels. See Section 2.9 for a bit more information about specifying colors in *plotc*.
`XORGU`	x-coordinate of x-axis origin (i.e., the point where the plotted axes will cross) in user's units. The default value is `XMIN`.
`YORGU`	y-coordinate of y-axis origin (i.e., the point where the plotted axes will cross) in user's units. The default value is `YMIN`.

3.3 `COORS` Input

After namelists `PTYPE` and `AXES` are read, subroutine `COORS` is called by *plotc* `NCURV` times (i.e., once for each curve being plotted) to get the number of points and the coordinates for each curve. The first time it is called (i.e., for the first curve) it also returns titles for the axes and for the plot itself. These will be used if the titles are not specified in namelist `AXES`, as described in Section 3.2. If `NCURV` = 0, subroutine `COORS` is still called once, to get the titles.

This subroutine `COORS` may either be user-supplied, or the version supplied with *plotc* may be used. Details on the parameters to be returned by a user-written version of subroutine `COORS` are presented in the next section. If the supplied version of `COORS` is being used, the titles and coordinates

[16] The labels will be evenly spaced on the x-axis. Depending on the values of `NLABX` and `NINTX`, they may or may not be lined up with the tick marks or grid lines. This comment also applies to the values of `NLABY` and `NINTY`, and the placement of the y-axis labels.

are read using Fortran `read` statements. The exact form of the data depends on the value of the input parameter `MODEC`, as described below.

The coordinates may be included in the namelist input file immediately following the titles, or, if `IFILEC` = 1, they may be read from a separate file. If they are being read from a separate file, the file name must be specified via the `-a` option when *plotc* is run. Note that this separate file contains *only* the coordinates. Any other needed information, such as the number of points `NP`, the values of x_0 and Δx in the `MODEC` = 1 option, etc., are read from the namelist input file, following the titles. If `NCURV` = 0, you should *not* supply your own version of `COORS`, and the titles should be input as described below.

For all values of `MODEC`, the following three records must immediately follow the end of namelist `AXES`. However, if the titles are being specified in the namelist, those specified here will be overwritten.

Record 1

`XTITLE`	Title for the x-axis (100 characters maximum.)

Record 2

`YTITLE`	Title for the y-axis (100 characters maximum.)

Record 3

`PTITLE`	Title for the top of the plot (100 characters maximum.)

3.3.1 `MODEC` = 0

With this option, the coordinates of the curves are specified by reading all the x-coordinates, followed by all the y-coordinates, with formatted read statements.[17] The input is described in detail below. The value of `NP` is read from the namelist input file, immediately following the titles. The coordinates themselves may be read from the namelist input file, or, if `IFILEC` = 1, from a separate file. Note that records 4, 5+, and 6+ are repeated `NCURV` times.

Record 4 (Format `I6`*)*

`NP`	Number of points in this curve. The total number of points on the plot (i.e., the sum for all the curves and symbol sets) must be no more than 100,000.

Records 5+ (Format `8E10.0`*)*

`X(I)`	x-coordinates of curve, where `I` = 1 to `NP`.

Records 6+ (Format `8E10.0`*)*

`Y(I)`	y-coordinates of curve, where `I` = 1 to `NP`.

See Section 4.1 for an example using the supplied `COORS` routine with `MODEC` = 0.

[17] Note that this option is similar to the `MODEC` = 4 option, but with the restriction of a fixed format. The only cases that would require `MODEC` = 0 instead of `MODEC` = 4 are those in which the coordinates are not separated by whitespace or commas. It is included for compatibility with earlier versions of *plotc*.

3.3.2 `MODEC` = 1

With this option, the coordinates of the curves are specified by reading only the y-coordinates. The format is determined by the input parameter `FMTXYC`. The x-coordinates are equally spaced, and computed from input values of x_0 and Δx. The input is described in detail below. The values of `NP`, `XO`, and `DX` are read from the namelist input file, immediately following the titles. The coordinates themselves may be read from the namelist input file, or, if `IFILEC` = 1, from a separate file. Note that records 4, 5+, and 6+ are repeated `NCURV` times.

Record 4

`NP`	Number of points in this curve. The total number of points on the plot (i.e., the sum for all the curves and symbol sets) must be no more than 100,000.

Records 5+

`XO,DX`	Starting value x_0 and increment Δx. The x-coordinate is computed from $x_i = x_0 + (i-1)\Delta x$, for $i = 1$ to `NP`.

Records 6+

`Y(I)`	y-coordinates of curve, where `I` = 1 to `NP`.

3.3.3 `MODEC` = 2

With this option, the coordinates of the curves are specified by reading x-y coordinate pairs. The format is determined by the input parameter `FMTXYC`. The input is described in detail below. The value of `NP` is read from the namelist input file, immediately following the titles. The coordinates themselves may be read from the namelist input file, or, if `IFILEC` = 1, from a separate file. Note that records 4 and 5+ are repeated `NCURV` times.

Record 4

`NP`	Number of points in this curve. The total number of points on the plot (i.e., the sum for all the curves and symbol sets) must be no more than 100,000.

Records 5+

`X(I),Y(I)`	x- and y-coordinates of curve, where `I` = 1 to `NP`.

3.3.4 `MODEC` = 4

With this option, the coordinates of the curves are specified by reading all the x-coordinates, followed by all the y-coordinates. The format is determined by the input parameter `FMTXYC`. The input is described in detail below. The value of `NP` is read from the namelist input file, immediately following the titles. The coordinates themselves may be read from the namelist input file, or, if `IFILEC` = 1, from a separate file. Note that records 4, 5+, and 6+ are repeated `NCURV` times.

Record 4

`NP`	Number of points in this curve. The total number of points on the plot (i.e., the sum for all the curves and symbol sets) must be no more than 100,000.

Records 5+

`X(I)`	x-coordinates of curve, where `I` = 1 to `NP`.

Records 6+

`Y(I)`	y-coordinates of curve, where `I` = 1 to `NP`.

3.3.5 `MODEC` = 20

With this option, the coordinates of the curves are specified by reading a table of values, with specified columns corresponding to the x- and y-coordinates. The format is determined by the input parameter `FMTXYC`. The input is described in detail below. The values of `NP`, `ICOLX`, and `ICOLY` are read from the namelist input file, immediately following the titles. The table itself may be read from the namelist input file, or, if `IFILEC` = 1, from a separate file. Note that records 4 and 5 are repeated `NCURV` times. Normally, records 6+ are also repeated `NCURV` times. However, if `IFILEC` = 1 and `IREWC` = 1, the file will be rewound after each curve, thus allowing multiple curves to be specified from a single table. In this case records 6+ are not repeated.

Record 4

`NP`	Number of points in this curve. The total number of points on the plot (i.e., the sum for all the curves and symbol sets) must be no more than 100,000.

Record 5

`ICOLX,ICOLY`	The column numbers in the table that correspond to the x- and y-coordinates.

Records 6+

`TABLE(I,J)`	A table of `NP` rows and `NCOL` columns, where `I` = 1 to `NP` and `J` = 1 to `NCOL`, and `NCOL` $\leq$ 40.

See Section 4.2 for an example using the supplied `COORS` routine with `MODEC` = 20.

3.3.6 `MODEC` = 30

With this option, the coordinates of the curves are specified in a Genplot file.[18] The name of the Genplot file must be specified using the `-a` option when *plotc* is run. The input is described in detail below. If plot and axes titles are not specified in the namelist input file, the titles in the Genplot file will be used.[19] The individual curve titles in the Genplot file are ignored. If the absolute value of the *plotc* input parameter `NCURV` is inconsistent with the number of curves `NCURVG` specified in the

[18] Genplot files are created by some post-processing utilities used with the NASTD and Wind-US Navier-Stokes CFD codes. Currently, *plotc* can only process Genplot files with data for a single plot.

[19] The plot title, if needed, will be created by combining the main plot title and the plot sub-title from the Genplot file.

Genplot file, a warning message will be printed, and the `NCURV` value will be used. The line styles for the curves will still be controlled by the sign of `NCURV`.

There are two possible formats for Genplot files, determined by the sign of the parameter `NCURVG` in the file. The first has multiple arrays of x and y coordinates, and the second has a single array of x coordinates and multiple arrays of y coordinates. For both formats, the first five records in the file (numbered 4–8 here, since records 1–3 are the axes and plot titles in the *plotc* namelist input file) are as follows:

Record 4

`MTITLE`	Main plot title.

Record 5

`STITLE`	Plot sub-title.

Record 6

`XTITLE`	x-axis title.

Record 7

`YTITLE`	y-axis title.

Record 8

`NCURVG`	Parameter whose absolute value is the number of curves.

The format for the remainder of the file depends on the sign of `NCURVG`. If `NCURVG` > 0, records 9, 10, and 11+ are repeated as a set `NCURVG` times.

Record 9

`CTITLE`	Curve title (ignored by *plotc*).

Record 10

`NP`	Number of points in this curve. The total number of points on the plot (i.e., the sum for all the curves and symbol sets) must be no more than 100,000.

Records 11+

`X(I),Y(I)`	x- and y-coordinates of curve, where `I` $= 1$ to `NP`.

If `NCURVG` < 0, record 9 is repeated |`NCURVG`| times, followed by records 10 and 11+.

Record 9

`CTITLE`	Curve title (ignored by *plotc*).

Record 10

`NP`	Number of points in curves. The total number of points on the plot (i.e., the sum for all the curves and symbol sets) must be no more than 100,000.

Records 11+

`X(I),Y1(I),Y2(I),...,Y`n`(I)`
x- and y-coordinates of curves, where `I` = 1 to `NP`, and n represents the number of curves.

3.4 `POINTS` Input

At this point, subroutine `POINTS` is called by *plotc* `NSYMB` times (i.e., once for each set of points to be plotted as symbols) to get the number of points, the coordinates for each set of points, and the error bar data. As with subroutine `COORS`, subroutine `POINTS` may either be user-supplied, or the version supplied with *plotc* may be used. Details on the parameters to be returned by a user-written version of subroutine `POINTS` are presented in the next section. If the supplied version of `POINTS` is being used, the coordinates and error bar data are read using Fortran `read` statements. The exact form of the data depends on the value of the input parameter `MODEP`.

The coordinates and error bar data may be included in the namelist input file immediately following the coordinates of the curves, or, if `IFILEP` = 1, they may be read from a separate file. If they are being read from a separate file, the file name must be specified via the `-b` option when *plotc* is run. Note that this separate file contains *only* the coordinates and error bar data. Any other needed information, such as the number of points `NP`, the values of x_0 and Δx in the `MODEP` = 1 option, etc., is read from the namelist input file, following the coordinates of the curves.

The parameters `MODEP`, `IFILEP`, `FMTXYP`, `ISKIPP`, and `IREWP` determine how the coordinates of the symbol sets and the error bar data are being specified. Except for the error bar data, they are exactly analogous to the corresponding parameters described in the previous section for the `COORS` routine. The supplied version of subroutine `POINTS` is similar to the supplied version of subroutine `COORS`, except for the addition of the error bar data, and the fact that the axes and plot titles are not required.

3.4.1 `MODEP` = 0

With this option, the coordinates of the symbol sets and the error bar data are specified by reading all the x-coordinates, followed by all the y-coordinates, followed by the error bar data, with formatted read statements.[20] The input is described in detail below. The value of `NP` is read from the namelist input file, immediately following the curve coordinates. The coordinates of the symbol sets and the error bar data may be read from the namelist input file, or, if `IFILEP` = 1, from a separate file. Note that records 4–8+ are repeated `NSYMB` times.

Record 4 (Format `I6`*)*[21]

[20] Note that this option is similar to the `MODEP` = 4 option, but with the restriction of a fixed format. The only cases that would require `MODEP` = 0 instead of `MODEP` = 4 are those in which the values are not separated by whitespace or commas. It is included for compatibility with earlier versions of *plotc*.

[21] For compatibility with earlier versions of *plotc*, if `MODEP` = 0 the record specifying `NP` is actually read using a `2I6` format, with the second value being the symbol size. If non-zero this value will override the corresponding value of `ISIZE1` specified in namelist `PTYPE`.

`NP`	Number of points in this symbol set. The total number of points on the plot (i.e., the sum for all the curves and symbol sets) must be no more than 100,000.

Records 5+ (Format `8E10.0`*)*

`X(I)`	x-coordinates of symbol set, where `I` $=1$ to `NP`.

Records 6+ (Format `8E10.0`*)*

`Y(I)`	y-coordinates of symbol set, where `I` $=1$ to `NP`.

Records 7+ (Format `8E10.0`*, only if* `IEBAR` $\neq 0$*)*

`YERR1(I)`	Length of error bars above symbols, specified as either a fraction of the y-coordinate value (if `IEBAR` $= \pm 1$), or explicitly in user's units (if `IEBAR` $= \pm 2$).

Records 8+ (Format `8E10.0`*, only if* `IEBAR` < 0*)*

`YERR2(I)`	Length of error bars below symbols, specified as either a fraction of the y-coordinate value (if `IEBAR` $= \pm 1$), or explicitly in user's units (if `IEBAR` $= \pm 2$).

See Section 4.1 for an example using the supplied `POINTS` routine with `MODEP` $= 0$.

3.4.2 `MODEP` = 1

With this option, the coordinates of the symbol sets and the error bar data are specified by reading the y-coordinates, followed by the error bar data. The format is determined by the input parameter `FMTXYP`. The x-coordinates are equally spaced, and computed from input values of x_0 and Δx. The input is described in detail below. The values of `NP`, `X0`, and `DX` are read from the namelist input file, immediately following the curve coordinates. The coordinates of the symbol sets and the error bar data may be read from the namelist input file, or, if `IFILEP` $= 1$, from a separate file. Note that records 4–8+ are repeated `NSYMB` times.

Record 4

`NP`	Number of points in this symbol set. The total number of points on the plot (i.e., the sum for all the curves and symbol sets) must be no more than 100,000.

Records 5+

`X0,DX`	Starting value x_0 and increment Δx. The x-coordinate is computed from $x_i = x_0 + (i-1)\Delta x$, for $i = 1$ to `NP`.

Records 6+

`Y(I)`	y-coordinates of symbol set, where `I` $=1$ to `NP`.

Records 7+ (Only if `IEBAR` $\neq 0$*)*

`YERR1(I)`	Length of error bars above symbols, specified as either a fraction of the y-coordinate value (if `IEBAR` $= \pm 1$), or explicitly in user's units (if `IEBAR` $= \pm 2$).

Records 8+ (Only if `IEBAR` < 0*)*

`YERR2(I)`	Length of error bars below symbols, specified as either a fraction of the y-coordinate value (if `IEBAR` $= \pm 1$), or explicitly in user's units (if `IEBAR` $= \pm 2$).

3.4.3 `MODEP` = 2

With this option, the coordinates of the symbol sets are specified by reading x-y coordinate pairs, followed by the error bar data. The format is determined by the input parameter `FMTXYP`. The input is described in detail below. The value of `NP` is read from the namelist input file, immediately following the curve coordinates. The coordinates of the symbol sets and the error bar data may be read from the namelist input file, or, if `IFILEP` $= 1$, from a separate file. Note that records 4–7+ are repeated `NSYMB` times.

Record 4

`NP`	Number of points in this symbol set. The total number of points on the plot (i.e., the sum for all the curves and symbol sets) must be no more than 100,000.

Records 5+

`X(I),Y(I)`	x- and y-coordinates of symbol set, where `I` $= 1$ to `NP`.

Records 6+ (Only if `IEBAR` $\neq 0$*)*

`YERR1(I)`	Length of error bars above symbols, specified as either a fraction of the y-coordinate value (if `IEBAR` $= \pm 1$), or explicitly in user's units (if `IEBAR` $= \pm 2$).

Records 7+ (Only if `IEBAR` < 0*)*

`YERR2(I)`	Length of error bars below symbols, specified as either a fraction of the y-coordinate value (if `IEBAR` $= \pm 1$), or explicitly in user's units (if `IEBAR` $= \pm 2$).

3.4.4 `MODEP` = 4

With this option, the coordinates of the symbol sets and the error bar data are specified by reading all the x-coordinates, followed by all the y-coordinates, followed by the error bar data. The format is determined by the input parameter `FMTXYP`. The input is described in detail below. The value of `NP` is read from the namelist input file, immediately following the curve coordinates. The coordinates of the symbol sets and the error bar data may be read from the namelist input file, or, if `IFILEP` $= 1$, from a separate file. Note that records 4–8+ are repeated `NSYMB` times.

Record 4

`NP`	Number of points in this symbol set. The total number of points on the plot (i.e., the sum for all the curves and symbol sets) must be no more than 100,000.

Records 5+

`X(I)`	x-coordinates of symbol set, where `I` $= 1$ to `NP`.

Records 6+

`Y(I)`	y-coordinates of symbol set, where `I` $= 1$ to `NP`.

Records 7+ (Only if `IEBAR` $\neq 0$*)*

`YERR1(I)`	Length of error bars above symbols, specified as either a fraction of the y-coordinate value (if `IEBAR` $= \pm 1$), or explicitly in user's units (if `IEBAR` $= \pm 2$).

Records 8+ (Only if `IEBAR` < 0*)*

`YERR2(I)`	Length of error bars below symbols, specified as either a fraction of the y-coordinate value (if `IEBAR` $= \pm 1$), or explicitly in user's units (if `IEBAR` $= \pm 2$).

3.4.5 `MODEP` = 20

With this option, the coordinates of the symbol sets and the error bar data are specified by reading a table of values, with specified columns corresponding to the x- and y-coordinates and the error bar data. The format is determined by the input parameter `FMTXYP`. The input is described in detail below. The values of `NP`, `ICOLX`, `ICOLY`, `ICOLE1`, and `ICOLE2` are read from the namelist input file, immediately following the curve coordinates. The table itself may be read from the namelist input file, or, if `IFILEP` = 1, from a separate file. Note that records 4 and 5 are repeated `NSYMB` times. Normally, records 6+ are also repeated `NSYMB` times. However, if `IFILEP` = 1 and `IREWP` = 1, the file will be rewound after each symbol set, thus allowing multiple symbol sets to be specified from a single table. In this case records 6+ are not repeated.

Record 4

`NP`	Number of points in this symbol set. The total number of points on the plot (i.e., the sum for all the curves and symbol sets) must be no more than 100,000.

Record 5

`ICOLX,ICOLY,ICOLE1,ICOLE2`
The column numbers in the table that correspond to the x- and y-coordinates of the symbol set, and the error bar parameters `YERR1` and `YERR2` as defined for `MODEP` = 0, etc.

Records 6+

`TABLE(I,J)`	A table of `NP` rows and `NCOL` columns, where `I` $= 1$ to `NP` and `J` $= 1$ to `NCOL`, and `NCOL` ≤ 40.

3.5 User-Written Subroutines `COORS` and `POINTS`

As described in previous sections, after namelists `PTYPE` and `AXES` are read, subroutine `COORS` is called `NCURV` times (i.e., once for each curve being plotted) to get the number of points and the coordinates for each curve. Similarly, subroutine `POINTS` is called `NSYMB` times (i.e., once for each set of points to be plotted as symbols) to get the number of points and the coordinates for each set of points, and the error bar data.

User-written versions of `COORS` and/or `POINTS` must be compiled and linked into the executable during the build process, as described in Appendix B.3.[22]

If *user-supplied* `COORS` and/or `POINTS` routines are being used, any *input* they require is, of course, determined by the user. The current section describes the *output* parameters that must be computed by these subroutines, and passed back to *plotc* via the argument lists.

Again, subroutine `COORS` is called *once for each curve* in a plot. The rest of the *plotc* program doesn't care what goes on inside `COORS`, as long as it returns the number of points in the curve and their x-y coordinates. The first time it is called (i.e., for the first curve) it may also return titles for the axes and for the plot itself. These will be used if the titles are not specified in namelist `AXES`, as described in Section 3.2.

The argument list for subroutine `COORS` is (`X,Y,NP,ICURV,XTITLE,YTITLE,PTITLE`), where:

`ICURV`	An input parameter specifying the number of the current curve.
`X`	The output array of `NP` x-coordinates for curve `ICURV`.
`Y`	The output array of `NP` y-coordinates for curve `ICURV`.
`NP`	An output parameter specifying the number of points in curve `ICURV`. The total number of points on the plot (i.e., the sum for all the curves and symbol sets) must be no more than 100,000.
`XTITLE`	An output character variable specifying the x-axis title (up to 100 characters.) This is only needed when `ICURV` $= 1$, and only if `ITITLE(2)` $= 1$.
`YTITLE`	An output character variable specifying the y-axis title (up to 100 characters.) This is only needed when `ICURV` $= 1$, and only if `ITITLE(3)` $= 1$.
`PTITLE`	An output character variable specifying the plot title (up to 100 characters.) This is only needed when `ICURV` $= 1$, and only if `ITITLE(1)` $= 1$.

Just as subroutine `COORS` is called once for each curve, subroutine `POINTS` is called *once for each set of points* to be plotted as symbols. The argument list is (`X,Y,YERR1,YERR2,NP,ISYMB,ISIZED`), where:

`ISYMB`	An input parameter specifying the number of the current set of points.
`X`	The output array of `NP` x-coordinates for set `ISYMB`.
`Y`	The output array of `NP` y-coordinates for set `ISYMB`.

[22]In version 2.5 of *plotc*, for SGI systems, user-written versions of `COORS` and/or `POINTS` may be specified via command-line options at run time. Unfortunately, due to the variety of compilers available for Linux systems, and the variety of installation locations for OpenGL routines, that capability is not currently available under Linux.

`YERR1`	The output array giving the length of error bars above symbols, specified as either a fraction of the y-coordinate value (if `IEBAR` $= \pm1$), or explicitly in user's units (if `IEBAR` $= \pm2$).
`YERR2`	The output array giving the length of error bars below symbols, specified as either a fraction of the y-coordinate value (if `IEBAR` $= \pm1$), or explicitly in user's units (if `IEBAR` $= \pm2$).
`NP`	An output parameter specifying the number of points in set `ISYMB`. The total number of points on the plot (i.e., the sum for all the curves and symbol sets) must be no more than 100,000.
`ISIZED`	An output parameter specifying the size of the symbol to be used for set `ISYMB`. The size is in 1/72's of a relative unit for the OpenGL plot, and 1/72's of an inch for the PostScript plot.[23] If `ISIZED` is not specified, the value of `ISIZE1` in namelist `PTYPE` will be used.

See Section 4.3 for an example using a user-written `COORS` routine.

[23]See Section 2.4 for the definition of relative units.

4 Running The Code

The steps involved in running the code are best shown by looking at some examples. The first example uses the default versions of subroutines `COORS` and `POINTS` to create a very simple plot, consisting of a two curves and a set of symbols. The second example also uses the default versions of subroutines `COORS` and `POINTS`, but is more complex, illustrating some of the capabilities of *plotc*. It plots the same curves and symbols as the first example, but includes a legend and a couple of text strings, uses two different fonts, has user-specified axis scales, and is in color. The third example is a simple illustration of the use of a user-written `COORS` routine.

4.1 Example 1 — A Very Simple Plot

This example creates a very simple plot, consisting of a two curves and a set of symbols. Most of the namelist input is defaulted.

The namelist input file is listed below, along with explanatory notes for each line. Remember that with namelist input each line starts in column 2 or higher.

```
 &ptype
  ncurv=2, nsymb=1,                     Plot two curves and one set of symbols.
 &end
 &axes
 &end
Circumferential Coordinate              x-axis title.
Bogosity Function                       y-axis title.
Plotc Example 1                         Plot title.
    51                                  Curve 1 x and y coordinates.
       0.0       1.8       3.6       5.4       7.2       9.0      10.8      12.6
      14.4      16.2      18.0      19.8      21.6      23.4      25.2      27.0
      28.8      30.6      32.4      34.2      36.0      37.8      39.6      41.4
      43.2      45.0      46.8      48.6      50.4      52.2      54.0      55.8
      57.6      59.4      61.2      63.0      64.8      66.6      68.4      70.2
      72.0      73.8      75.6      77.4      79.2      81.0      82.8      84.6
      86.4      88.2      90.0
   0.00000   0.03141   0.06279   0.09411   0.12533   0.15643   0.18738   0.21814
   0.24869   0.27899   0.30902   0.33874   0.36812   0.39715   0.42578   0.45399
   0.48175   0.50904   0.53583   0.56208   0.58779   0.61291   0.63742   0.66131
   0.68455   0.70711   0.72897   0.75011   0.77051   0.79015   0.80902   0.82708
   0.84433   0.86074   0.87631   0.89101   0.90483   0.91775   0.92978   0.94088
   0.95106   0.96029   0.96858   0.97592   0.98229   0.98769   0.99211   0.99556
   0.99803   0.99951   1.00000
    51                                  Curve 2 x and y coordinates.
       0.0       1.8       3.6       5.4       7.2       9.0      10.8      12.6
      14.4      16.2      18.0      19.8      21.6      23.4      25.2      27.0
      28.8      30.6      32.4      34.2      36.0      37.8      39.6      41.4
      43.2      45.0      46.8      48.6      50.4      52.2      54.0      55.8
      57.6      59.4      61.2      63.0      64.8      66.6      68.4      70.2
      72.0      73.8      75.6      77.4      79.2      81.0      82.8      84.6
      86.4      88.2      90.0
   0.00000   0.00099   0.00394   0.00886   0.01571   0.02447   0.03511   0.04759
   0.06185   0.07784   0.09549   0.11474   0.13552   0.15773   0.18129   0.20611
   0.23209   0.25912   0.28711   0.31594   0.34549   0.37566   0.40631   0.43733
   0.46860   0.50000   0.53140   0.56267   0.59369   0.62434   0.65451   0.68406
   0.71289   0.74088   0.76791   0.79389   0.81871   0.84227   0.86448   0.88526
   0.90451   0.92216   0.93815   0.95241   0.96489   0.97553   0.98429   0.99114
   0.99606   0.99901   1.00000
   10                                      Symbol set 1 x and y coordinates.
       0.        10.        20.        30.        40.        50.        60.        70.
      80.        90.
   0.00000   0.02447   0.13552   0.28711   0.37566   0.62434   0.74088   0.90451
   0.96489   1.00000
```

The following terminal session shows how you would use the above input file to create your plot. Lines in slanted type are entered by the user. The remainder are printed by the computer. This example assumes the *plotc* namelist input file is *bogus1.plotin*.

```
plotc bogus1.plotin
Using namelist input file bogus1.plotin

PostScript output is in plotc.ps
```

The plot will appear in a window on the graphics monitor. This plot is drawn using OpenGL routines. To continue execution of the *plotc* program, with the cursor in the plot window, hit the Esc key, the Enter key, the space bar, or the left mouse button.

After *plotc* finishes executing, a new file called *plotc.ps* will appear in your current directory. This is the PostScript file for the plot, and can be printed using a a PostScript printer, or used as input to a PostScript previewer. Note that subsequent runs using *plotc* will overwrite this file, so change the file name if you want to keep it around. Or, alternatively, you can specify a name for this file from the command line with the `-o` option, as described in the *plotc* man page. The PostScript plot is shown in Figure 1.

4.2 Example 2 — A More Complex Plot

This example illustrates some of the additional capabilities of *plotc*. It draws the same two curves as in the first example, along with symbols representing experimental data with error bars. This time the `MODEC` = 20 option is used, and the coordinates of the curves are stored in a separate file. Text strings are drawn on the plot to label the two curves, a legend is added, and color is used.

The namelist input file is listed below, along with explanatory notes for each line. Remember that with namelist input each line starts in column 2 or higher.

```
 &ptype
  ncurv=-2, nsymb=1,                 Plot two curves and one set of symbols.
  rgbcrv(1,1)=0.5,0.0,0.0,           Colors "maroon" and red for curves.
  rgbcrv(1,2)=1.0,0.0,0.0,
  rgbsym(1,1)=0.0,0.0,0.5,           Use dark blue for symbols.
  ititle=1,1,1,                      Include plot and axis titles.
  fonts='H','R','R','R',             Use Helvetica for labels, Times-Roman otherwise.
  ilegnd=1,                          Add a legend.
```

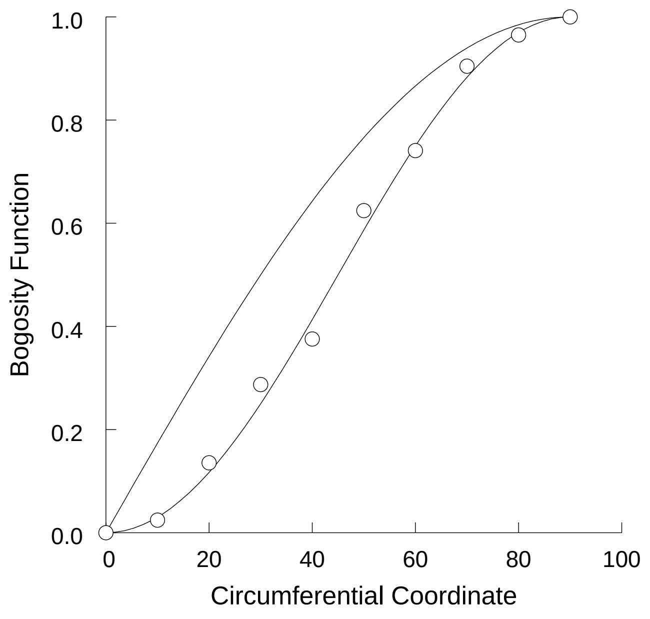

Figure 1: Resulting plot from example 1.

```
     xleg=7.5, yleg=.9,                 Legend position.
     aleg(1,2)='Theory A',              Legend contents, column 2.
               'Theory B',
               'Experiment',
     ilegbx=1,                          Draw a box around legend.
     ilclr=1,                           Use curve/symbol colors for text.
  astr='\ftIPr \ftR = 1.0',             Add these text strings to plot.
       '\ftIPr \ftR = 0.7',
     xstr=25.,35., ystr=.45,.28,        Locations for strings.
     isizes=16,16,                      Point sizes for strings.
     salign(1)='RIGHT',                 Right-align first string.
     rgbstr(1,1)=0.5,0.0,0.0,           Colors ``maroon'' and red for strings.
     rgbstr(1,2)=1.0,0.0,0.0,
  modec=20,                             Get curve coordinates from a table.
     ifilec=1, irewc=1,                 Separate file for table, rewind between curves.
  modep=4,                              Get symbol sets as x, then y, free-form.
  iebar=1,                              Add centered error bars, length is fraction of y value.
 &end
 &axes
  iscale=1,                             Specify scales for both axes.
  xmin=0., xmax=90., xlen=6., nintx=9,  x-axis information.
  ymin=0., ymax=1.,  ylen=6., ninty=5,  y-axis information.
  ixaxis=2, iyaxis=2,                   Draw grid lines.
  rgbax=0.0,0.0,0.5,                    Dark blue for axes.
 &end
Circumferential Coordinate, \561        x-axis title.
Bogosity Function, \ftIF\sub[\542]      y-axis title.
Plotc Example 2                         Plot title.
```

```
51                                      Number of points, curve 1.
1 2                                     Use col. 1 for x, col. 2 for y.
51                                      Number of points, curve 2.
1 3                                     Use col. 1 for x, col. 3 for y.
10                                      Symbol set 1 x and y coordinates, error bar data.
0. 10. 20. 30. 40. 50. 60. 70. 80. 90.
0.00000 0.02447 0.13552 0.28711 0.37566 0.62434 0.74088 0.90451 0.96489 1.00000
0.      0.9     0.4     0.2     0.1     0.075   0.05    0.05    0.02    0.01
```

The table defining the curve coordinates is stored in a separate file, listed below.

```
 0.0   0.00000   0.00000
 1.8   0.03141   0.00099
 3.6   0.06279   0.00394
 5.4   0.09411   0.00886
 7.2   0.12533   0.01571
 9.0   0.15643   0.02447
10.8   0.18738   0.03511
12.6   0.21814   0.04759
14.4   0.24869   0.06185
16.2   0.27899   0.07784
18.0   0.30902   0.09549
19.8   0.33874   0.11474
21.6   0.36812   0.13552
23.4   0.39715   0.15773
25.2   0.42578   0.18129
27.0   0.45399   0.20611
28.8   0.48175   0.23209
30.6   0.50904   0.25912
32.4   0.53583   0.28711
34.2   0.56208   0.31594
36.0   0.58779   0.34549
37.8   0.61291   0.37566
39.6   0.63742   0.40631
41.4   0.66131   0.43733
43.2   0.68455   0.46860
45.0   0.70711   0.50000
46.8   0.72897   0.53140
48.6   0.75011   0.56267
50.4   0.77051   0.59369
52.2   0.79015   0.62434
54.0   0.80902   0.65451
55.8   0.82708   0.68406
57.6   0.84433   0.71289
59.4   0.86074   0.74088
61.2   0.87631   0.76791
63.0   0.89101   0.79389
64.8   0.90483   0.81871
66.6   0.91775   0.84227
68.4   0.92978   0.86448
70.2   0.94088   0.88526
```

```
72.0   0.95106   0.90451
73.8   0.96029   0.92216
75.6   0.96858   0.93815
77.4   0.97592   0.95241
79.2   0.98229   0.96489
81.0   0.98769   0.97553
82.8   0.99211   0.98429
84.6   0.99556   0.99114
86.4   0.99803   0.99606
88.2   0.99951   0.99901
90.0   1.00000   1.00000
```

The following terminal session shows how you would use the above input files to create your plot. Lines in slanted type are entered by the user. The remainder are printed by the computer. This example assumes the *plotc* namelist input file is *bogus2.plotin*, and that the file containing the coordinates of the curves is *bogus2.coords*.

```
plotc -a bogus2.coords bogus2.plotin
Using namelist input file bogus2.plotin
Using unit 8 input file bogus2.coords

PostScript output is in plotc.ps
```

As in the previous example, the OpenGL plot will appear in a window on the screen, and the PostScript plot will be written into the file *plotc.ps*. The PostScript plot is shown in Figure 2.

4.3 Example 3 — Using a User-Supplied COORS Routine

Suppose, for some strange reason, you wanted a plot of a square inscribed in a circle. You could create a file named *coors.f90*, containing the following COORS routine to be used with *plotc*. It computes coordinates for two curves. The first is the circle, centered at (0,0) with a radius of 1. The second curve is the inscribed square.

```
      subroutine coors (x,y,np,icurv,xtitle,ytitle,ptitle)
!
!          This subroutine is used as an example to illustrate the use
!     of the plotting program Plotc.  It returns x-y coordinates for a
!     plot of a square inscribed in a circle (not real useful, but this
!     is only an introductory example, after all.)
!
      dimension x(1),y(1)
      character*(*) xtitle,ytitle,ptitle
      save radius
      data pi /3.1415927/

!
!-----First curve, the circle
!
      if (icurv == 1) then
```

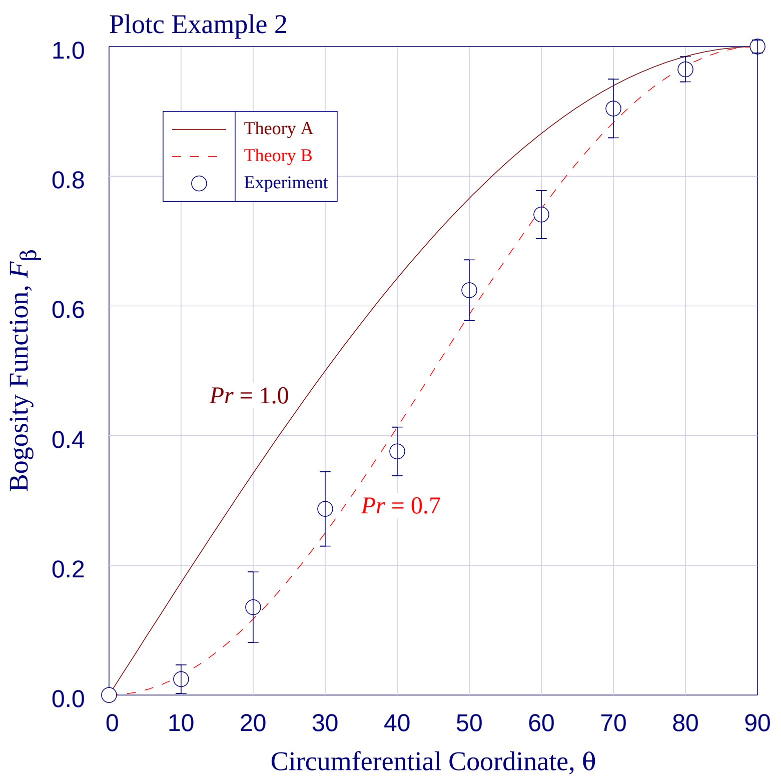

Figure 2: Resulting plot from example 2.

```
!--------Set number of points in curve
         np = 101
!--------Define x-y coordinates for circle
         radius = 1.
         dtheta = (2.*pi)/float(np-1)
         do i = 1,np
            theta = dtheta*float(i-1)
            x(i) = radius*cos(theta)
            y(i) = radius*sin(theta)
         end do

!
!-----Second curve, the inscribed square
!
      else if (icurv == 2) then
!--------Half-length of side
         side2 = sqrt(radius/2.)
!--------x-y coordinates of square corners
         x(1) = -side2
         y(1) =  side2
         x(2) =  side2
         y(2) =  side2
         x(3) =  side2
         y(3) = -side2
         x(4) = -side2
         y(4) = -side2
         x(5) = x(1)
         y(5) = y(1)
!--------Total number of points in curve
         np = 5
      endif

      return
      end
```

The above user-written file *coors.f90* must then be compiled and linked into the executable during the build process, as described in Appendix B.2.

You'll also need a namelist file for the *plotc* input. Remember that with namelist input each line starts in column 2 or higher. The input file might look like:

```
 &ptype
  ncurv=2,                                    Plot two curves (circle and square).
 &end
 &axes
  iscale=1,                                   Specify scales for both axes.
  xmin=-1., xmax=1., ymin=-1., ymax=1.,       Axes starting and ending values.
  xlen=6., ylen=6.,                           Axes lengths.
  ixaxis=3, iyaxis=3,                         Suppress plotting of both axes.
 &end
```

Note that even though we are suppressing the plotting of both axes, values for `XMIN`, `XMAX`, `XLEN`, etc., must still be specified to allow *plotc* to transform user's units into relative units.

The following terminal session shows how you would then create your plot. Lines in slanted type are entered by the user. The remainder are printed by the computer. This example assumes that the *plotc* namelist input file is *cirsq.plotin*.

```
plotc cirsq.plotin
Using namelist input file cirsq.plotin

PostScript output is in plotc.ps
```

As in the previous example, the OpenGL plot will appear in a window on the screen, and the PostScript plot will be written into the file *plotc.ps*. The PostScript plot is shown in Figure 3.

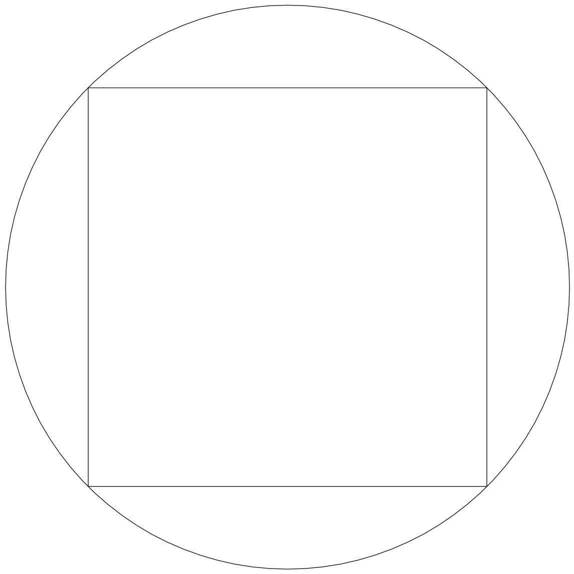

Figure 3: Resulting plot from example 3.

5 *plotc* Man Page

The following man page describes how to use the shell script *plotc*.

NAME

plotc — a 2-D plotting program

SYNOPSIS

plotc [**-a** *input8*] [**-b** *input9*] [**-o** *outps*] [**-g** *outdb*] [**-V**] *namelist*

DESCRIPTION

plotc is a Bourne shell script used to run the plotting program *plotc.ex*. The namelist input to *plotc.ex* must be supplied in the file *namelist*, and is described in detail in the *Plotc User's Manual*.

OPTIONS

-a	Links the file *input8* to *fort.8*. This is the file used for the coordinates of the curves being plotted when `IFILEC` = 1.
-b	Links the file *input9* to *fort.9*. This is the file used for the coordinates of the symbol sets being plotted when `IFILEP` = 1.
-o	Use *outps* as the name of the output PostScript file. The default is *plotc.ps*.
-g	Use *outdb* as the name of the output debug file. The default is *plotc.debug*.
-V	Print the current version number and usage information for *plotc*, and immediately exit.

NOTES

Please contact Charlie Towne (*towne@nasa.gov*) to report bugs and/or to suggest improvements.

BUGS

The input parameter `FONTS`, which allows the user to specify the font to be used for plot titles, labels, legend, and text strings, is supported only for the PostScript plot.

The input parameters `ISIZEL`, `ISIZET`, `ISIZEG`, and `ISIZES`, which allow the user to specify the size of characters in the labels, titles, legend, and strings, are supported only for the PostScript plot.

Probably others yet to be discovered.

FILES

plotc accesses or creates the following files:

plotc.ex	Executable.
plotc.debug	Debug output file, symbolically linked to *fort.30*.
plotc.ps	PostScript output file, symbolically linked to *fort.31*.
plotc.psprolog	Prolog used in PostScript output file.

SEE ALSO

Plotc User's Manual.

AUTHOR

Charlie Towne

Appendix A. *plotc_p3d*—2-D Plots from Plot3d Files

As mentioned in the introduction, a special-case version of *plotc*, called *plotc_p3d*, may also be used. *plotc_p3d* can be used to create two-dimensional line plots from the xyz and q files created by many CFD codes for use with the three-dimensional plotting program *Plot3d*[24](Walatka, Buning, Pierce, and Elson, 1990). Plots may be created showing the variation of a function along any of the grid lines in the three-dimensional flow field. This appendix includes some general information about *plotc_p3d*, an example showing how to run it, and the man page for the shell script used to run the code.

A.1 General Description

plotc_p3d is, basically, simply *plotc* with a specialized `COORS` routine. Since, except for the `COORS` routine, the *plotc_p3d* code is the same as the *plotc* code, namelists `PTYPE` and `AXES` described previously in Section 3.1 and Section 3.2 for *plotc* also apply to *plotc_p3d*. A third namelist, `P3D`, is used to specify the function to be plotted and the grid line or lines of interest. The standard `COORS` input described in Section 3.3, of course, is not used. However, sets of symbols may be plotted using the standard `POINTS` input, just as in *plotc*.

The type of *Plot3d* xyz and q files (i.e., whole, planes, or 2-D; unformatted, binary, or formatted; and with or without `IBLANK`'ing)[25] is specified via command-line options. The files may be single-block, or multi-block with at most 20 blocks. The number of grid points per block is limited to 400,000. The grid lines of interest (i.e., those along which the function being plotted is varying) must all be in the same block.

A.2 Namelist `P3D`

This namelist is used to specify the desired function, and to identify the grid line or lines along which the function will be plotted.

`IFCT` An integer specifying the function to be plotted. The functions currently available are:

1	x-velocity, u
2	y-velocity, v
3	z-velocity, w
4	Mach number, M
5	Speed of sound, a
6	Contravariant velocity normal to ξ surface, U
7	Contravariant velocity normal to η surface, V
8	Contravariant velocity normal to ζ surface, W
9	Total velocity magnitude, $\|\vec{V}\|$
10	x-momentum, ρu
11	y-momentum, ρv
12	z-momentum, ρw
13	ξ-velocity

[24] *Plot3d* is a plotting package written at NASA Ames Research Center, and is used primarily for displaying results from computational fluid dynamics codes.

[25] Currently, xyz files with `IBLANK`'ing may be read, but the `IBLANK` array is ignored.

14	η-velocity
15	ζ-velocity
16	Flow angle, α_v
17	Flow angle, α_w
20	Static density, ρ
21	Total density, ρ_T
30	Static pressure, p
31	Total pressure, p_T
32	Static pressure coefficient, c_p
33	Total pressure coefficient, c_{p_T}
34	Pitot pressure, p_p
35	Dynamic pressure, q
40	Static temperature, T
41	Total temperature, T_T
50	Total energy per unit volume, E_T
51	Total energy, E_T/ρ
52	Internal energy, e_i
53	Kinetic energy, e_k
60	Static enthalpy, h
61	Total enthalpy, h_T
70	x-vorticity, Ω_x
71	y-vorticity, Ω_y
72	z-vorticity, Ω_z
73	Total vorticity magnitude, $\lvert\vec{\Omega}\rvert$
80	Entropy, s
90	Laminar coefficient of viscosity, μ_l
91	Laminar second coefficient of viscosity, λ_l
92	Laminar coefficient of thermal conductivity, k_l
93	Specific heat at constant pressure, c_p
94	Specific heat at constant volume, c_v
95	Ratio of specific heats, γ
100	Turbulent coefficient of viscosity, μ_t
101	Turbulent second coefficient of viscosity, λ_t
102	Turbulent coefficient of thermal conductivity, k_t
103	Effective coefficient of viscosity, μ_e
104	Effective second coefficient of viscosity, λ_e
105	Effective coefficient of thermal conductivity, k_e
106	Turbulent kinetic energy, k
107	Turbulent dissipation rate, ϵ
108	Inner region coordinate, y^+
109	Inner region velocity, u^+
120	Shear stress, τ_{xx}
121	Shear stress, τ_{yy}
122	Shear stress, τ_{zz}
123	Shear stress, τ_{xy}
124	Shear stress, τ_{xz}

125	Shear stress, τ_{yz}
126	Heat flux, q_x
127	Heat flux, q_y
128	Heat flux, q_z
200	x-coordinate, x
201	y-coordinate, y
202	z-coordinate, z
210	Inverse Jacobian, J^{-1}
211	Metric coefficient, ξ_t
212	Metric coefficient, ξ_x
213	Metric coefficient, ξ_y
214	Metric coefficient, ξ_z
215	Metric coefficient, η_t
216	Metric coefficient, η_x
217	Metric coefficient, η_y
218	Metric coefficient, η_z
219	Metric coefficient, ζ_t
220	Metric coefficient, ζ_x
221	Metric coefficient, ζ_y
222	Metric coefficient, ζ_z

The definitions of all these functions are presented in the next section. Note that for turbulent flow, functions 100–107 and 120–128 require a turbulence data file, in addition to the usual xyz and q files. The format of the *Plot3d* xyz and q files is described by Walatka, Buning, Pierce, and Elson (1990). The format of the turbulence data file is described in Appendix A.4. The default value for `IFCT` is 13, corresponding to the ξ-velocity.

`IABSC` An integer specifying the abscissa for the plot. Note that the `IABSC` $= -n$ capability allows you to plot any supported function against any other supported function. For 3-D cases, the valid values are:

1	Cartesian coordinate, x
2	Cartesian coordinate, y
3	Cartesian coordinate, z
4	Index along the specified grid line
5	Distance along the specified grid line
$-n$	Function number n.

For 2-D cases, of course, 3 is an invalid value, as are negative values related to the z direction. The default value is 5, corresponding to the distance along the specified grid line.

`IDIR` An integer specifying the computational coordinate direction of the grid line(s) of interest. For 3-D cases, the valid values are:

1	For the ξ direction
2	For the η direction
3	For the ζ direction

For 2-D cases, of course, 3 is an invalid value. The default value is 1, corresponding to the ξ direction.

`IND`	An array of up to 100 integers specifying exactly which grid line(s) in direction `IDIR` are of interest. For 3-D cases, each pair of values represents a separate curve in the plot. If `IDIR` $= 1$, corresponding to grid lines in the ξ direction, the first pair of values specifies the η and ζ indices of the first grid line of interest, the second pair specifies the η and ζ indices of the second grid line of interest, etc. Similarly, if `IDIR` $= 2$, corresponding to grid lines in the η direction, these values specify the ξ and ζ indices. And if `IDIR` $= 3$, corresponding to grid lines in the ζ direction, these values specify the ξ and η indices. For 3-D cases the default for the first pair of values is 1,1, and the default for the remaining values is 0. For 2-D cases, each value represents a separate curve in the plot. If `IDIR` $= 1$, corresponding to grid lines in the ξ direction, the first value specifies the η index of the first grid line of interest, the second value specifies the η index of the second grid line of interest, etc. Similarly, if `IDIR` $= 2$, corresponding to grid lines in the η direction, these values specify the ξ indices. For 2-D cases the default for the first value is 1, and the default for the remaining values is 0.
`INEAR`	An integer specifying the location of the solid wall for plots of y^+ or u^+. This parameter is only needed if `IFCT` $= 108$ or 109, or if `IABSC` $= -108$ or -109. For 3-D cases, the valid values are: 1 For a wall at $\xi = 0$; 2 For a wall at $\xi = 1$; 3 For a wall at $\eta = 0$; 4 For a wall at $\eta = 1$; 5 For a wall at $\zeta = 0$; 6 For a wall at $\zeta = 1$. For 2-D cases, of course, 5 and 6 are invalid values. The default value is 3, corresponding to a wall at $\eta = 0$.
`IBLOCK`	The block number for multi-block xyz and q files. The default value is 0, indicating that the xyz and q files are in single-block form.

A.3 Function Definitions

This section describes how the various functions that may be plotted are computed from information in the xyz and q files. For turbulent flow, functions 100–107 and 120–128 also require a turbulence data file. The equations in this section are written for 3-D flow. Those for 2-D flow are similar.

The xyz file contains the Cartesian coordinates x, y, and z at each grid point. The q file contains the static density q_1, the momentum components q_2 through q_4, and the total energy per unit volume q_5. In the xyz and q files, lengths are non-dimensionalized by the reference length L_r, density by the reference density ρ_r, velocity by the reference speed of sound a_r, and total energy per unit volume by $\rho_r a_r^2$. The q file also contains the reference Mach number $M_r = u_r/a_r$ and the reference Reynolds number $Re_r = \rho_r a_r L_r/\mu_r$, where u_r and μ_r are the reference velocity and the reference viscosity. The format of the xyz and q files is described by Walatka, Buning, Pierce, and Elson (1990).

The turbulence data file contains the turbulent viscosity coefficient μ_t, the turbulent kinetic energy k, and the turbulent dissipation rate ϵ. These three parameters are non-dimensionalized by

μ_r, u_r^2, and $\rho_r u_r^4/\mu_r$, respectively. This file also contains a "reference" Prandtl number defined as $Pr_r = \mu_r u_r^2/k_r T_r$, where k_r is the reference thermal conductivity. The format of the turbulence data file is described in Appendix A.4.

In the following equations, all parameters are non-dimensional, except for those with an overbar and reference conditions like ρ_r, etc. The parameter γ_r is assumed to be 1.4.

A.3.1 Velocities

The velocities in the Cartesian coordinate directions are defined as

$$u = \frac{q_2}{\rho M_r}$$
$$v = \frac{q_3}{\rho M_r}$$
$$w = \frac{q_4}{\rho M_r}$$

where the static density $\rho = q_1$. The division by M_r causes plotted velocities to be non-dimensionalized by u_r instead of a_r. The Cartesian momentum components are then simply ρu, ρv, and ρw.

The velocity magnitude is

$$|\vec{V}| = \sqrt{u^2 + v^2 + v^2}$$

The speed of sound is

$$a = \sqrt{\gamma_r R T}$$

Here T is the static temperature, defined by

$$T = \frac{\bar{T}}{T_r} = \frac{(\gamma_r - 1)}{R}\left(\frac{E_T}{\rho} - \frac{|\vec{V}|^2}{2}\right)$$

where E_T is the total energy per unit volume, defined by

$$E_T = \frac{q_5}{M_r^2}$$

The division by M_r^2 in the definition of the total energy per unit volume causes it to be non-dimensionalized by $\rho_r u_r^2$ instead of $\rho_r a_r^2$. The parameter $R = \bar{R}T_r/u_r^2 = 1/\gamma_r M_r^2$ is the dimensionless gas constant, and T_r is the reference temperature consistent with the reference speed of sound a_r.

The Mach number is then simply

$$M = \frac{|\vec{V}|}{a}$$

The contravariant velocities (i.e., velocities normal to constant ξ, η, and ζ surfaces) are given by

$$U = \xi_x u + \xi_y v + \xi_z w$$
$$V = \eta_x u + \eta_y v + \eta_z w$$
$$W = \zeta_x u + \zeta_y v + \zeta_z w$$

where the metrics ξ_x, ξ_y, etc., are computed numerically from the coordinates in the xyz file, as explained in Appendix A.3.13.

The velocities in the ξ, η, and ζ directions are

$$\begin{aligned} V_\xi &= \frac{x_\xi u + y_\xi v + z_\xi w}{(x_\xi^2 + y_\xi^2 + z_\xi^2)^{1/2}} \\ V_\eta &= \frac{x_\eta u + y_\eta v + z_\eta w}{(x_\eta^2 + y_\eta^2 + z_\eta^2)^{1/2}} \\ V_\zeta &= \frac{x_\zeta u + y_\zeta v + z_\zeta w}{(x_\zeta^2 + y_\zeta^2 + z_\zeta^2)^{1/2}} \end{aligned}$$

where the subscripts on x, y, and z denote partial differentiation. These terms are computed from the metrics, as follows:

$$\begin{aligned} x_\xi &= \eta_y \zeta_z - \eta_z \zeta_y \\ y_\xi &= \eta_z \zeta_x - \eta_x \zeta_z \\ z_\xi &= \eta_x \zeta_y - \eta_y \zeta_x \\ x_\eta &= \xi_z \zeta_y - \xi_y \zeta_z \\ y_\eta &= \xi_x \zeta_z - \xi_z \zeta_x \\ z_\eta &= \xi_y \zeta_x - \xi_x \zeta_y \\ x_\zeta &= \xi_z \eta_y - \xi_y \eta_z \\ y_\zeta &= \xi_x \eta_z - \xi_z \eta_x \\ z_\zeta &= \xi_y \eta_x - \xi_x \eta_y \end{aligned}$$

The flow angles are in degrees, and are defined by

$$\begin{aligned} \alpha_v &= \tan^{-1}(v/u) \\ \alpha_w &= \tan^{-1}(w/u) \end{aligned}$$

A.3.2 Densities

As noted in the previous section, the static density is simply

$$\rho = q_1$$

The total density is then

$$\rho_T = \rho \left(1 + \frac{\gamma_r - 1}{2M^2} \right)^{1/(\gamma_r - 1)}$$

A.3.3 Pressures

The static and total pressures are defined as

$$p = \frac{\bar{p}}{p_r} = \rho T$$

$$p_T = \frac{\bar{p}_T}{p_r} = p \left(1 + \frac{\gamma_r - 1}{2M^2} \right)^{\gamma_r/(\gamma_r - 1)}$$

where $\bar{p}$ is the dimensional static pressure, $\bar{p}_T$ is the dimensional total pressure, $p_r = \rho_r \bar{R} T_r$ is the reference pressure, and $\bar{R}$ is the dimensional gas constant.

The static pressure coefficient is defined as

$$c_p = \frac{\bar{p} - p_r}{\rho_r u_r^2/2}$$

This definition for c_p turns out to be equivalent to

$$c_p = 2R(p-1)$$

Similarly, the total pressure coefficient is defined as

$$c_{p_T} = \frac{\bar{p}_T - (p_T)_r}{\rho_r u_r^2/2}$$

where $(p_T)_r$ is the total pressure at the reference conditions. This definition for c_{p_T} turns out to be equivalent to

$$c_{p_T} = 2R[p_T - (p_T)_0]$$

where

$$(p_T)_0 = \left(1 + \frac{\gamma_r - 1}{2} M_r^2\right)^{\gamma_r/(\gamma_r-1)}$$

For $M \leq 1$ the pitot pressure $p_p = p_T$. For $M > 1$ the pitot pressure is

$$p_p = p\left(\frac{\gamma_r+1}{2}M^2\right)^{\gamma_r/(\gamma_r-1)}\left(2\frac{\gamma_r}{\gamma_r+1}M^2 - \frac{\gamma_r-1}{\gamma_r+1}\right)^{1/(\gamma_r-1)}$$

The dynamic pressure is defined as

$$p_{dyn} = \frac{\bar{p}_{dyn}}{p_r} = \frac{\rho|\vec{V}|^2}{2R}$$

where $\bar{p}_{dyn}$ is the dimensional dynamic pressure.

A.3.4 Temperatures

The static and total temperatures are defined as

$$T = \frac{\bar{T}}{T_r} = \frac{\gamma_r - 1}{R}\left(\frac{E_T}{\rho} - \frac{|\vec{V}|^2}{2}\right)$$

$$T_T = T\left(1 + \frac{\gamma_r - 1}{2M^2}\right)$$

A.3.5 Energies

The total energy per unit volume is

$$E_T = \frac{q_5}{M_r^2}$$

The division by M_r^2 causes it to be non-dimensionalized by $\rho_r u_r^2$ instead of $\rho_r a_r^2$. The total energy itself is then E_T/ρ.

The internal energy is defined as

$$e_i = \frac{\bar{c}_v \bar{T}}{u_r^2} = \frac{R}{\gamma_r - 1} T$$

where $\bar{c}_v$ is the specific heat at constant volume.

The kinetic energy is defined as

$$e_k = \frac{1}{2} |\vec{V}|^2$$

A.3.6 Enthalpies

The static enthalpy is defined as

$$h = \frac{\bar{c}_p \bar{T}}{u_r^2} = \frac{\gamma_r R}{\gamma_r - 1} T$$

Similarly, the total enthalpy is

$$h_T = \frac{\bar{c}_p \bar{T}_T}{u_r^2} = \frac{\gamma_r R}{\gamma_r - 1} T_T$$

A.3.7 Vorticity

The vorticity components in the x, y, and z directions are defined as

$$\begin{aligned}
\Omega_x &= \frac{\partial w}{\partial y} - \frac{\partial v}{\partial z} \\
\Omega_y &= \frac{\partial u}{\partial z} - \frac{\partial w}{\partial x} \\
\Omega_z &= \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}
\end{aligned}$$

These are actually computed using

$$\begin{aligned}
\Omega_x &= \xi_y w_\xi + \eta_y w_\eta + \zeta_y w_\zeta - \xi_z v_\xi - \eta_z v_\eta - \zeta_z v_\zeta \\
\Omega_y &= \xi_z u_\xi + \eta_z u_\eta + \zeta_z u_\zeta - \xi_x w_\xi - \eta_x w_\eta - \zeta_x w_\zeta \\
\Omega_z &= \xi_x v_\xi + \eta_x v_\eta + \zeta_x v_\zeta - \xi_y u_\xi - \eta_y u_\eta - \zeta_y u_\zeta
\end{aligned}$$

where the subscripts on u, v, and w denote partial differentiation. The velocity derivatives are computed using second-order central differences at interior points, and second-order one-sided differences at boundaries. The metrics are also computed numerically, as explained in Appendix A.3.13.

The total vorticity magnitude is simply

$$|\vec{\Omega}| = \sqrt{\Omega_x^2 + \Omega_y^2 + \Omega_z^2}$$

A.3.8 Entropy

The entropy is defined as

$$s = c_v \ln p - c_p \ln \rho$$

where c_v and c_p are the specific heats at constant volume and pressure, and are computed assuming the reference temperature $T_r = 519$ °R, as described in the next section.

A.3.9 Temperature-Dependent Parameters

The laminar viscosity coefficient is computed from the following formula, which is valid for air at temperatures from about 400 to 3400 °R (White, 1974):

$$\mu_l = T^{0.67}$$

The second coefficient of laminar viscosity $\lambda_l = -2\mu/3$.

The laminar thermal conductivity coefficient is computed from the following formula, which is valid for air at temperatures from about 375 to 1800 °R (White, 1974):

$$k_l = T^{0.81}$$

The specific heat at constant pressure is computed from the following formula, which is valid for air at temperatures from about 540 to 9000 °R (Hesse and Mumford, 1964):

$$c_p = \frac{1}{\gamma_r \bar{R} M_r^2} \left(8530 - \frac{3.12 \times 10^4}{\sqrt{\bar{T}}} - \frac{2.065 \times 10^6}{\bar{T}} + \frac{7.83 \times 10^8}{\bar{T}^2} \right)$$

where the dimensional temperature $\bar{T}$ is computed assuming a reference temperature of 519 °R, and the dimensional gas constant $\bar{R}$ is assumed to be 1716 ft^2/sec^2-°R. The specific heat at constant volume is then

$$c_v = c_p - R$$

and the ratio is

$$\gamma = \frac{c_p}{c_v}$$

A.3.10 Turbulence Parameters

Functions 100–107 require a turbulence data file, in addition to the usual xyz and q files. This file contains the turbulent viscosity coefficient μ_t, the turbulent kinetic energy k, and the turbulent dissipation rate ϵ. The second coefficient of turbulent viscosity $\lambda_t = -2\mu_t/3$.

The turbulent thermal conductivity coefficient is defined as

$$k_t = \frac{1}{k_r} \frac{\bar{c}_p \bar{\mu}_t}{Pr_t}$$

where the overbar denotes a dimensional quantity, and $Pr_t = 0.9$ is the turbulent Prandtl number. This turns out to be equivalent to

$$k_t = \frac{c_p \mu_t}{Pr_t} Pr_r$$

The effective viscosity coefficient, second coefficient of viscosity, and thermal conductivity coefficient are simply

$$\begin{aligned}\mu_e &= \mu_l + \mu_t \\ \lambda_e &= \lambda_l + \lambda_t \\ k_e &= k_l + k_t\end{aligned}$$

Computing the inner region variables y^+ and u^+ requires knowledge of the wall location. Since this information is not in the xyz, q, or turbulence data files, the user must supply it in namelist `P3D` when these parameters are being plotted. The inner region velocity is defined as

$$u^+ = \frac{|\vec{V}|}{u_\tau}$$

Here u_τ is the friction velocity, computed as

$$u_\tau = \left(\frac{\mu |\vec{\Omega}|}{\rho Re_r M_r} \right)_w^{1/2}$$

where the subscript w indicates wall conditions.

The inner region coordinate is defined as

$$y^+ = \frac{\rho_w u_\tau y_n}{\mu_w} Re_r M_r$$

where y_n is the distance to the wall.

A.3.11 Gradients

For turbulent flow, functions 120–128 require a turbulence data file, in addition to the usual xyz and q files. The shear stresses and heat fluxes are defined as

$$\begin{aligned}\tau_{xx} &= 2\mu_e \frac{\partial u}{\partial x} + \lambda_e \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) \\ \tau_{yy} &= 2\mu_e \frac{\partial v}{\partial y} + \lambda_e \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) \\ \tau_{zz} &= 2\mu_e \frac{\partial w}{\partial z} + \lambda_e \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right)\end{aligned}$$

$$\begin{aligned}\tau_{xy} &= \mu_e \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \\ \tau_{xz} &= \mu_e \left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \right) \\ \tau_{yz} &= \mu_e \left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \right)\end{aligned}$$

$$\begin{aligned}q_x &= -k_e \frac{\partial T}{\partial x} \\ q_y &= -k_e \frac{\partial T}{\partial y} \\ q_z &= -k_e \frac{\partial T}{\partial z}\end{aligned}$$

In the above equations, the derivatives of u with respect to x, y, and z are computed as

$$\begin{aligned} u_x &= \xi_x u_\xi + \eta_x u_\eta + \zeta_x u_\zeta \\ u_y &= \xi_y u_\xi + \eta_y u_\eta + \zeta_y u_\zeta \\ u_z &= \xi_z u_\xi + \eta_z u_\eta + \zeta_z u_\zeta \end{aligned}$$

where the subscripts on u denote partial differentiation. The derivatives of v, w, and T are computed similarly. The velocity derivatives are computed using second-order central differences at interior points, and second-order one-sided differences at boundaries. The metrics are also computed numerically, as explained below in Appendix A.3.13.

A.3.12 Coordinates

The Cartesian coordinates x, y, and z are obtained directly from the xyz file.

A.3.13 Metrics

The Jacobian of the generalized nonorthogonal grid transformation is defined as $J = 1/J^{-1}$, where

$$J^{-1} = x_\xi(y_\eta z_\zeta - y_\zeta z_\eta) + x_\eta(y_\zeta z_\xi - y_\xi z_\zeta) + x_\zeta(y_\xi z_\eta - y_\eta z_\xi)$$

and the subscripts denote partial differentiation. The metric coefficients themselves are defined as

$$\begin{aligned} \xi_x &= J(y_\eta z_\zeta - y_\zeta z_\eta) \\ \xi_y &= J(x_\zeta z_\eta - x_\eta z_\zeta) \\ \xi_z &= J(x_\eta y_\zeta - x_\zeta y_\eta) \\ \eta_x &= J(y_\zeta z_\xi - y_\xi z_\zeta) \\ \eta_y &= J(x_\xi z_\zeta - x_\zeta z_\xi) \\ \eta_z &= J(x_\zeta y_\xi - x_\xi y_\zeta) \\ \zeta_x &= J(y_\xi z_\eta - y_\eta z_\xi) \\ \zeta_y &= J(x_\eta z_\xi - x_\xi z_\eta) \\ \zeta_z &= J(x_\xi y_\eta - x_\eta y_\xi) \end{aligned}$$

The derivatives x_ξ, x_η, etc., are computed using second-order central differences at interior points, and second-order one-sided differences at boundaries.

A.4 Turbulence Data File

For turbulent flow, functions 100–107 and 120–128 require a turbulence data file, in addition to the Plot3d xyz and q files. This file is analogous to the q file in format. Like the q file, it may be whole, planes, or 2-D, and unformatted, binary, or formatted. It may also be single- or multi-block.

The following code fragments show how this file may be written, in unformatted or binary form, using Fortran write statements. A formatted file may be written by replacing `(iunit)` with `(iunit,*)`

The Fortran variable `nblock` is the number of grid blocks; `idim`, `jdim`, and `kdim` are the grid sizes in the ξ, η, and ζ directions for each block; `machr` is the reference Mach number M_r; `prr` is the "reference" Prandtl number $Pr_r = \mu_r u_r^2 / k_r T_r$; `rer` is the reference Reynolds number $Re_r = \rho_r a_r L_r / \mu_r$; `time` is an unused parameter; and `mut`, `ke`, and `eps` are the turbulent viscosity μ_t, the turbulent kinetic energy k, and the turbulent dissipation rate ϵ.

Single-block, 2-D

```
   write (iunit) idim,jdim
   write (iunit) machr,prr,rer,time
   write (iunit) ((mut(i,j),i=1,idim),j=1,jdim), &
                 ((ke (i,j),i=1,idim),j=1,jdim), &
                 ((eps(i,j),i=1,idim),j=1,jdim)
```

Single-block, 3-D, whole

```
   write (iunit) idim,jdim,kdim
   write (iunit) machr,prr,rer,time
   write (iunit) (((mut(i,j,k),i=1,idim),j=1,jdim),k=1,kdim), &
                 (((ke (i,j,k),i=1,idim),j=1,jdim),k=1,kdim), &
                 (((eps(i,j,k),i=1,idim),j=1,jdim),k=1,kdim)
```

Single-block, 3-D, planes

```
   write (iunit) idim,jdim,kdim
   write (iunit) machr,prr,rer,time
   do k = 1,kdim
      write (iunit) ((mut(i,j),i=1,idim),j=1,jdim), &
                    ((ke (i,j),i=1,idim),j=1,jdim), &
                    ((eps(i,j),i=1,idim),j=1,jdim)
   end do
```

Multi-block, 2-D

```
   write (iunit) nblock
   write (iunit) (idim(ibl),jdim(ibl),ibl=1,nblock)
   do ibl = 1,nblock
      write (iunit) machr,prr,rer,time
      write (iunit) ((mut(i,j),i=1,idim(ibl)),j=1,jdim(ibl)), &
                    ((ke (i,j),i=1,idim(ibl)),j=1,jdim(ibl)), &
                    ((eps(i,j),i=1,idim(ibl)),j=1,jdim(ibl))
   end do
```

Multi-block, 3-D, whole

```
   write (iunit) nblock
   write (iunit) (idim(ibl),jdim(ibl),kdim(ibl),ibl=1,nblock)
   do ibl = 1,nblock
      write (iunit) machr,prr,rer,time
      write (iunit) (((mut(i,j,k),i=1,idim(ibl)),j=1,jdim(ibl)),k=1,kdim(ibl)), &
                    (((ke (i,j,k),i=1,idim(ibl)),j=1,jdim(ibl)),k=1,kdim(ibl)), &
                    (((eps(i,j,k),i=1,idim(ibl)),j=1,jdim(ibl)),k=1,kdim(ibl))
   end do
```

Multi-block, 3-D, planes

```
   write (iunit) nblock
   write (iunit) (idim(ibl),jdim(ibl),kdim(ibl),ibl=1,nblock)
```

```
do ibl = 1,nblock
   write (iunit) machr,prr,rer,time
   do k = 1,kdim(ibl)
      write (iunit) ((mut(i,j,k),i=1,idim(ibl)),j=1,jdim(ibl)), &
                    ((ke (i,j,k),i=1,idim(ibl)),j=1,jdim(ibl)), &
                    ((eps(i,j,k),i=1,idim(ibl)),j=1,jdim(ibl))
   end do
end do
```

A.5 Example

plotc_p3d was used to generate a plot comparing the computed and experimental static pressure distribution along the upper and lower surfaces of a two-dimensional transonic diffuser. The calculations were done using *Proteus*, an unsteady 2-D/axisymmetric Navier-Stokes computer code (Towne, Schwab, and Bui, 1993). The experimental data were taken by Hsieh, Wardlaw, Bogar, and Coakley (1987).

The *Plot3d* code, of course, was designed for 3-D problems. The unsteady 2-D *Proteus* code writes results into the *Plot3d* files with time stored in the z slot in the xyz file. This particular problem was a steady flow problem. Therefore only one time level, with the final solution, was written into the file.

The namelist input file was called *sajben.plotin*, and is listed below. Note that the coordinates of the points to be plotted as symbols, representing the experimental data, are read in using the default `POINTS` routine. These data are thus included in the namelist input file, as described in Section 3.4.

Note also that `NCURV` $= -1$ even though we want to plot 2 curves, showing the computed static pressure along the upper and lower surfaces of the diffuser. *plotc_p3d* will automatically determine the number of curves based on the input in namelist `P3D`. Therefore, only the sign on `NCURV` is significant.[26] As described in Section 3.1, the sign determines whether each curve is to be plotted using a solid line, or using different line types.

```
&ptype
 ncurv=-1, nsymb=-2,                     Use different line types for curves,
                                            and plot two sets of symbols.
 isize1=5,                               Size of symbols.
 ititle(1)=1,                            Include plot title.
 isizet=11, isizel=10,                   Size of characters in titles and labels.
 ilegnd=1, xleg=3.0, yleg=0.6,           Add a legend.
 isizeg=8,                               Size of characters in legend.
 ilegbx=-1,                              Put box around legend.
 aleg(1,2)='Bottom Wall - Proteus',      Legend contents, column 2.
           'Top Wall - Proteus',
           'Bottom Wall - Experiment',
           'Top Wall - Experiment',
&end
&axes
 iscale=1,                               Specify scales for both axes.
```

[26]This is only true if the input parameter `OFFSET` $= 0.0$. If `OFFSET` > 0.0, the number of curves to be plotted must be specified.

```
  xmin=-5., xmax=10., nintx=6, xlen=5.,     x-axis information.
  ymin=.3,  ymax=1.,  ninty=7, ylen=3.125,  y-axis information.
  iframe=2,                                 Draw frame around plot.
  ptitle='Transonic Diffuser Flow',         Title for top of plot.
 &end
 &p3d
  ifct=30,                                  Plot static pressure,
  iabsc=1,                                  vs. the x coordinate,
  idir=1,                                   along lines in the ξ direction,
  ind=1,1,51,1,                             with these η and ζ indices.
 &end
    38                                      Experimental values, lower surface.
   -4.0101   -3.3134   -2.4344   -1.9845   -1.6942   -0.9468   -0.6368   -0.5023
   -0.3008   -0.0504    0.0203    0.1383    0.3682    0.5759    0.7057    0.9142
    1.0316    1.1592    1.2952    1.4388    1.5355    1.5497    1.7481    1.7434
    2.0053    2.2696    2.4902    2.8756    3.3169    3.9725    4.4024    5.0671
    5.8317    6.3346    6.9666    7.4592    8.0546    8.6307
    0.8617    0.8621    0.8467    0.8218    0.7887    0.6771    0.6496    0.5823
    0.5685    0.5670    0.5496    0.5369    0.5259    0.4977    0.4965    0.4721
    0.4565    0.4448    0.4292    0.4939    0.5118    0.5797    0.5955    0.6175
    0.6265    0.6470    0.6799    0.7020    0.7250    0.7512    0.7637    0.7890
    0.8035    0.8100    0.8125    0.8142    0.8187    0.8202
    38                                      Experimental values, upper surface.
   -3.9303   -3.2281   -2.6103   -1.8224   -1.5536   -0.8820   -0.6452   -0.3465
   -0.1869    0.0125    0.2211    0.3900    0.5292    0.6399    0.7798    0.9111
    1.0512    1.2204    1.4180    1.5493    1.5157    1.7092    1.7050    1.9448
    2.1293    2.3701    2.6292    3.0134    3.4348    4.1133    4.5805    5.2556
    6.0164    6.5299    7.1564    7.6785    8.2383    8.8267
    0.8646    0.8646    0.8722    0.8740    0.8260    0.6671    0.5929    0.5552
    0.5428    0.5435    0.5423    0.5280    0.5045    0.4800    0.4621    0.4507
    0.4337    0.4213    0.4089    0.3975    0.5013    0.5291    0.5684    0.5877
    0.6193    0.6461    0.6672    0.7023    0.7307    0.7663    0.7825    0.7928
    0.8049    0.8127    0.8147    0.8159    0.8152    0.8153
```

A terminal session using the above namelist input file is shown below. The first line, in slanted type, is entered by the user. The xyz and q files are in whole binary form, and this information is specified via the `-r` and `-f` options on the *plotc_p3d* command line. The command *plotc_p3d* is the shell script used to run the plotting program *plotc_p3d*, and the man page is presented in the next section. This example assumes the *Plot3d* xyz and q files are named *sajben.px* and *sajben.pq*, respectively.

```
plotc_p3d -r w -f b sajben.px sajben.pq sajben.plotin
Using plot3d xyz file sajben.px
Using plot3d q file sajben.pq
Using namelist input file sajben.plotin
 Reading xyz and q files

PostScript output is in plotc_p3d.ps
```

As with *plotc*, the plot will appear in a window on the graphics monitor, drawn using OpenGL routines. The PostScript version of the plot will be stored in a file called *plotc_p3d.ps* in your current directory. Subsequent runs using *plotc_p3d* will overwrite this file, so change the file name if you want to keep it around. Or, alternatively, you can specify a name for this file from the command line with the `-o` option, as described in the *plotc_p3d* man page. The PostScript plot is shown in Figure 4.

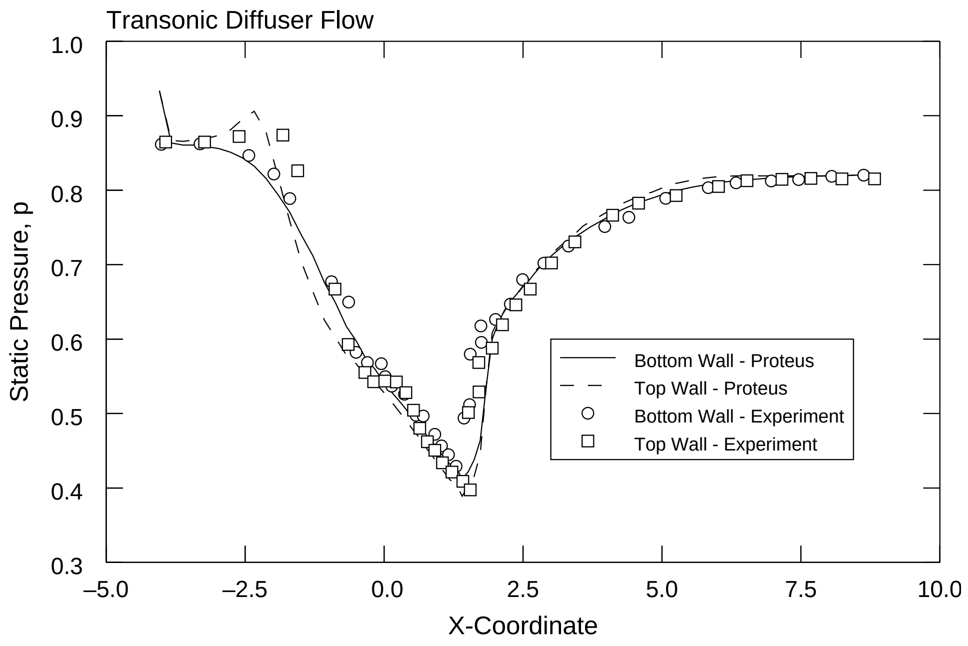

Figure 4: Resulting plot from *plotc_p3d* example.

A.6 *plotc_p3d* Man Page

The following man page describes how to use the shell script *plotc_p3d*.

NAME

plotc_p3d — a 2-D plotting program for *Plot3d* xyz and q files

SYNOPSIS

plotc_p3d [**-b** *input9*] [**-o** *outps*] [**-g** *outdb*] [**-t** *tfile*] [**-f** *u*|*b*|*f*] [**-r** *w*|*p*|*2*] [**-i**] [**-V**] *xyzfile qfile namelist*

DESCRIPTION

plotc_p3d is a Bourne shell script used to run the plotting program *plotc_p3d.ex.* This program can be used to create two-dimensional line plots from the xyz and q files created by many CFD codes for the three-dimensional *Plot3d* plotting program. Plots may be created showing the variation of various functions along specified grid lines in the three-dimensional mesh. The names of the xyz and q files are given by the arguments *xyzfile* and *qfile*, respectively. The namelist input to *plotc_p3d.ex* must be supplied in the file *namelist*, and is described in detail in the *Plotc User's Manual.*

OPTIONS

-b	Links the file *input9* to *fort.9.* This is the file used for the coordinates of the symbol sets being plotted when `IFILEP` = 1.
-o	Use *outps* as the name of the output PostScript file. The default is *plotc_p3d.ps.*
-g	Use *outdb* as the name of the output debug file. The default is *plotc_p3d.debug.*
-t	In addition to the xyz and q files, read the turbulence data file *tfile.*
-f	Specifies the file format for the xyz, q, and turbulence data files. The options are *u*, *b*, or *f*, corresponding to unformatted, binary, and formatted, respectively. The default is *b*.
-r	Specifies the record format for the xyz, q, and turbulence data files. The options are *w*, *p*, or *2*, corresponding to 3-D whole format, 3-D planes format, and 2-D format, respectively. The default is *w*.
-i	If specified, the xyx file includes the integer `IBLANK` array. The default is an xyx file without the `IBLANK` array.
-V	Print the current version number and usage information for *plotc_p3d*, and immediately exit.

NOTES

plotc_p3d.ex is simply a specialized version of *plotc.ex*, a general-purpose two-dimensional plotting program, with a `COORS` routine that reads the *Plot3d* xyz and q files and computes coordinates for various two-dimensional line plots.

Please contact Charlie Towne (*towne@nasa.gov*) to report bugs and/or to suggest improvements.

BUGS

The **-i** option allows xyz files with `IBLANK`'ing to be read, but currently the `IBLANK` array is ignored.

The input parameter `FONTS`, which allows the user to specify the font to be used for plot titles, labels, legend, and text strings, is supported only for the PostScript plot.

The input parameters `ISIZEL`, `ISIZET`, `ISIZEG`, and `ISIZES`, which allow the user to specify the size of characters in the labels, titles, legend, and strings, are supported only for the PostScript plot.

Probably others yet to be discovered.

FILES

plotc_p3d accesses or creates the following files:

plotc_p3d.ex	Executable for *plotc_p3d*.
plotc_p3d.debug	Debug output file, symbolically linked to *fort.30*.
plotc_p3d.ps	PostScript output file, symbolically linked to *fort.31*.
plotc.psprolog	Prolog used in PostScript output file.

SEE ALSO

Plotc User's Manual.
PLOT3D User's Manual, NASA TM 101067.

AUTHOR

Charlie Towne

Appendix B. Acquiring Plotc

Plotc is currently at Version 2.6, for use on Linux systems. A slightly earlier version (2.5) is available for SGI systems. Due to security enhancements in the ftp system at NASA Glenn, it is no longer permanently available via anonymous ftp. Contact me directly at *towne@nasa.gov* to acquire the program.

B.1 Unpacking

The file *plotc26.tar.gz* is a compressed tar file, and must first be uncompressed and un-tar'ed by doing

```
gunzip -c plotc26.tar.gz | tar xvf -
```

This will create a directory called *plotc26* in your current directory. In the *plotc26* directory will be the following files and subdirectories:

main/	Source code for non-device-specific routines
subsgl/	Source code for the OpenGL-specific routines
subsps/	Source code for the PostScript-specific routines
coors/	Source code for subroutines `COORS` and `POINTS`
coorp3d/	Source code unique to *plotc_p3d*
include/	Include files
plotc.psprolog	Prolog for PostScript output
plotc.sh	Basis for *plotc*, the shell script for running *plotc*
plotc_p3d.sh	Basis for *plotc_p3d*, the shell script for running *plotc_p3d*
plotc.1.trf	Nroff/troff source for *plotc* man page
plotc.1	Formatted on-line *plotc* man page
plotc.1.ps	PostScript version of *plotc* man page
plotc_p3d.1.trf	Nroff/troff source for *plotc_p3d* man page
plotc_p3d.1	Formatted on-line *plotc_p3d* man page
plotc_p3d.1.ps	PostScript version of *plotc_p3d* man page
plotc.namelist	Sample namelist input file for *plotc*
Makefile	Makefile for building/installing *plotc* and *plotc_p3d*
README	Introductory hype, unpacking and installation instructions, how to run sample input file, and change history
doc/.html*	HTML version of the *Plotc User's Manual*. Point your browser to the *index.html* file.
doc/manual.pdf	PDF version of the *Plotc User's Manual*
doc/quick.ps	PostScript version of *Plotc Quick Guide*

B.2 Installing *Plotc*

Plotc has been successfully built on a Linux system using version 6.3.2 of the Mesa 3-D graphics library, which is, in effect, an OpenGL clone. This version of Mesa includes GLUT (OpenGL Utility Toolkit) version 3.7.1, which provides the interface routines to the windowing system. Version 1.2.10 of *f90gl* was used for the Fortran bindings to the GLUT and Mesa graphics routines. None of these are provided with *plotc*, and they (or their equivalents) must be pre-installed on your system in order to build the *plotc* executable. (Unless, that is, you're building a PostScript-only version of *plotc*, as described below.)

First, *cd* into the *plotc26* subdirectory, and edit the *Makefile*.

- Check the definitions of `DIR1`, `DIR4`, and `DIR5`. These specify the directories to be used for storing various files, as described in the comments at the top of the file. They are set to directories commonly used at NASA Glenn, and assume you have root access. If you don't have root access, or if you'd rather store these files in other directories, change the definitions. The directory defined by `DIR1` must be in your search path.
- Check the definitions of the pre-processor, compile, and load commands and flags, and the Fortran libraries. These currently assume Intel Fortran is being used.
- Check the definitions of the locations for the OpenGL, GLUT, and *f90gl* include (*.h*) and module (*.mod*) files, and the libraries.
- Check the definitions of the X11 libraries.
- If OpenGL, etc., aren't available on your system, *plotc* can still be built, but only the PostScript plot will be available. To do this, in the definition of `OBJS`, add the line

  ```
  main/dumgl.o \
  ```

  somewhere among the other object files in *main*, and delete all the `subsgl/*.o` lines.

Finally, if you're installing both *plotc* and *plotc_p3d*, do

```
make install_p3d
```

If you're only installing *plotc*, do

```
make install
```

You may get some warning messages from *mkdir* about already-existing files. These can be ignored.

When finished, the following new files and subdirectories will appear:

main/.o*	Object code for non-device-specific routines
subsgl/.o*	Object code for the OpenGL-specific routines
subsps/.o*	Object code for the PostScript-specific routines
coors/.o*	Object code for subroutines `COORS` and `POINTS`
include/params.h	Include file containing Fortran parameters
DIR1/plotc.ex	Executable for *plotc*
DIR1/plotc	Shell script for running *plotc.ex*
DIR4/plotc.psprolog	Prolog for PostScript output
DIR5/plotc.1	Formatted on-line man page for *plotc*

where *DIR1*, *DIR4*, and *DIR5* are the directories defined in the *Makefile*.

If *plotc_p3d* was also installed, the following additional files and subdirectories will appear:

coorp3d/.o*	Object code unique to *plotc_p3d*
DIR1/plotc_p3d.ex	Executable for *plotc_p3d*
DIR1/plotc_p3d	Shell script for running *plotc_p3d.ex*
DIR5/plotc_p3d.1	Formatted on-line man page for *plotc_p3d*

B.3 Building With User-Written `COORS/POINTS` Routines

As noted in Section 3.5 user-written `COORS` and/or `POINTS` routines may be used to create specialized versions of *plotc*. To build a *plotc* executable with a user-written `COORS` and/or `POINTS` routine, after unpacking the compressed tar file, in the *coors* subdirectory simply replace the files *coors.f90* and/or *points.f90* with your own versions. Then follow the instructions in Appendix B.2 to install your specialized version of *plotc*.

Note that the subroutine names (`COORS` and `POINTS`) and the file names (*coors.f90* and *points.f90*) for your user-written versions must be the same as for the original supplied versions. You should therefore copy the original versions to a different name and/or location, in case you need to later re-build *plotc* using those routines.

References

Adobe Systems, Inc. (1985) *PostScript Language Reference Manual*.

Hesse, W. J., and Mumford, N. V. S. (1964) *Jet Propulsion for Aerospace Applications*, Pitman Publishing Corporation, New York.

Hsieh, T., Wardlaw, A. B., Jr., Collins, P., and Coakley, T. J. (1987) "Numerical Investigation of Unsteady Inlet Flow Fields," AIAA Journal, Vol. 25, No. 1, pp. 75–81.

Towne, C. E., Schwab, J. R., and Bui, T. T. (1993) "Proteus Two-Dimensional Navier-Stokes Computer Code — Version 2.0, Volumes 1–3," NASA TM's 106336, 106338, 106339.

Walatka, P. P., Buning, P. G., Pierce, L., and Elson, P. A. (1990) "PLOT3D User's Manual," NASA TM 101067.

White, F. M. (1974) *Viscous Fluid Flow*, McGraw-Hill Book Company, New York.