# Ada Bindings for the PLplot Plotting Package

## 1 About the This Document

This document describes the Ada bindings to the PLplot technical plotting software, how to obtain the necessary software components, and how to use them together.

## 2 Overview

The Ada bindings for PLplot provide a way for Ada programmers to access the powerful PLplot technical plotting facilities directly from Ada programs while working completely in Ada—the Ada programmer never needs to know or worry that PLplot itself is written in another language.

There are a thin binding and two thick bindings provided. The thin binding presents the application programming interface (API) in a form very similar to the C API, although in 100% Ada. The thick bindings present the API in a form to which Ada programmers will be more accustomed and add some ease-of-use features. It is expected that the thick bindings will be preferred.

## 3 About PLplot

PLplot is a general purpose technical plotting package that can be accessed directly by the user's own program without first writing the data to be plotted to disk. The package is marked by speed, convenience, breadth, and flexibility.

The PLplot project is one of the most active on SourceForge, typically in the 99th percentile for activity.

The following is taken from the PLplot web site at plplot.sourceforge.net:

"PLplot is a library of functions that are useful for making scientific plots.

"PLplot can be used from within compiled languages such as C, C++, FORTRAN and Java, and interactively from interpreted languages such as Octave, Python, Perl and Tcl.

"The PLplot library can be used to create standard x-y plots, semilog plots, log-log plots, contour plots, 3D surface plots, mesh plots, bar charts and pie charts. Multiple graphs (of the same or different sizes) may be placed on a single page with multiple lines in each graph.

"A variety of output file devices such as Postscript, png, jpeg and others, as well as interactive devices such as xwin, tk, xterm and Tektronics devices are supported. New devices can be easily added by writing a small number of device dependent routines.

"There are almost 2000 characters in the extended character set. This includes four different fonts, the Greek alphabet and a host of mathematical, musical, and other symbols. Some devices supports its own way of dealing with text, such as the Postscript driver, or the png and jpeg drivers that uses the Freetype library."

## 4 The Bindings

The bindings are a re-expression and extension of the C-language API and as such are a kind of abstract layer between the user's code and the PLplot binary library. Additionally, there are a few capabilities not in the official API but nonetheless which are available to the C programmer which are included in the bindings and thus are directly available to the Ada programmer.

The thin binding is a layer between the thick bindings and the underlying C code. It is mainly a programming convenience for the developer of the bindings; this is a common implementation for foreign language bindings and for the most part, the user can ignore it.

There are two thick bindings provided for the convenience of the user. Either may be used and they both provide exactly the same functionality. The thick bindings are the user's main concern with programming for PLplot.

### 4.1 Thin Binding

The thin binding, in the files `plplotthin.ads` and `plplotthin.adb`, is mostly a direct and obvious mapping of the C application programming interface (API) to Ada. Thus, for example, where a C program such as `plcol0` requires a single integer argument, there is a corresponding Ada program also called `plcol0` which also requires a single integer argument. (`plcol0` happens to set the drawing color using a number which is associated with a set of colors.) Various constants from the C API are also included here. Numeric types as defined in PLplot are associated with numeric types in Ada in the thin binding by use of Ada's type system. Thus, the thin binding refers to the PLplot-centric type `PLFLT` for floating-point types while the thick binding uses the usual Ada type `Long_Float`.

Many of the comments from the C source header file (similar in purpose to an Ada specification file) have been retained in the thin binding, even when they are no longer sensical. These might be pruned at some point to facilitate reading the Ada source.

Also included in the thin binding are some other declarations which help the Ada binding to mesh well with C by emulating certain data structures which are needed in some rather specialized usages as well as providing certain subprogram pointer types.

The Ada programmer working with either of the thick bindings will have to refer to the thin binding relatively rarely, if ever, and mainly to examine the subroutine pointer declarations and the several variant record types which are used mostly for contour and three-dimensional plots. However, some of these have been `subtype`-ed or `renames`-ed in the thick bindings so even less reference to the thin binding will be necessary. The goal is to put everything of interest to the user in the thick bindings and the user need not `with` the thin binding.

### 4.2 The Thick Bindings

The thick bindings provide most of the information that the Ada programmer needs. Normally, only one of the two thick bindings would be used per user program but it should be possible to include both but that scenario would be unusual.

There are three main aspects of the thick bindings: providing an alternative access to the PLplot API, extending the PLplot functionality with some easy-to-use features, and overlaying Ada data structures and types.

In the first aspect, the thick bindings provide a fully Ada interface to the entire PLplot library. Packages are `with`-ed and `use`-d as normal Ada code. Ada arrays can be passed as usual, not requiring the array length or start or end indices to be passed separately. All necessary Ada types are made to match the underlying C types exactly.

The second aspect of the thick bindings is to provide some simplified ways to get a lot of plotting done with only one or two subroutine calls. For example, a single call to `Simple_Plot` can display from one to five "*y*'s" as a function of a single "*x*" with default plot appearances chosen to suit many situations. Other simple plotters are available for three-dimensional and contour plots. Manipulating PLplot's colors is similarly made easy and some default color schemes are provided.

The third main aspect of the thick binding is to use Ada data structures and Ada's type system extensively to reduce the chances of inappropriate actions. For example, Ada arrays are used throughout (as opposed to C's pointer-plus-offset-while-carrying-along-the-size-separately approach). Quantities which have natural range limits are `subtype`-d to reflect those constraints. The hope is that program errors will result in more-familiar Ada compilation or run-time errors rather than error reports from the PLplot library or no reports at all. However, there remain a few instances where the typing could be improved and PLplot errors will still be reported from time to time.

Both the specification and body for the standard thick (and thin) binding contain the C subroutine name as a comment line immediately above the Ada procedure declaration; this should help in making the associations between "Ada" names and "PLplot" names. Also, the subroutine-specific comments from the C API have been retained verbatim.

### 4.3 Standard Thick Binding Using Enhanced Names

The distinguishing feature of this thick binding (the "standard" binding) is to provide more descriptive names for PLplot subroutines, variables, constants, arguments, and other objects. Most Ada programmers will be more comfortable using these names. For example, in the C API as well as the thin Ada binding and the other thick Ada binding, the procedure `plcol0(1)` sets the drawing color to red. In the standard thick binding, the same thing is accomplished by writing `Set_Color(Red)`. The Ada program may just as well write `Set_Color(1)` since the binding merely sets a constant Red to be equal to the integer 1. Many such numeric constants from the C API are given names in this thick binding. These renamed integers are discussed more fully in Section 7.2.

The disadvantage of this renaming is that it makes referring to the PLplot documentation somewhat awkward. There might be, at some time, a utility for easing this problem by providing an HTML file with links so that a "normal" PLplot name can be linked to the "Ada" name along with the appropriate entry in the Ada specification, as well as another HTML file with links from the "Ada" name directly to the PLplot web page that documents that name. It might also be possible to provide an alternate version of the documentation with the enhanced names used. (The developer of the bindings has a sed file prepared which makes most of the subroutine-name substitutions.) However, this thick binding retains the original C subprogram names as comments immediately above the `function` or `procedure` name in the code listing so it is relatively easy to locate the relevant item in the PLplot documentation.

One simple rule applies in reading the PLplot API documentation: the argument names are in the same order in Ada as in the PLplot documentation (the names are different) except that all array lengths are eliminated. The PLplot documentation, for each subroutine, shows a "redacted" version which should be correct for Ada as well as other languages which have proper arrays.

The standard bindings are in the Ada files plplot.ads and plplot.adb.

### 4.4 Thick Binding Using Traditional Names

This thick binding provides exactly the same functionality as the standard thick binding but retains the original names as used in the C code and the PLplot documentation.

The traditional bindings are in the Ada files plplot_traditional.ads and plplot_traditional.adb.

### 4.5 Examples

An important part of the Ada bindings is the examples, some 30 of which demonstrate how to use many of the features of the PLplot package. These examples also serve as a testbed for the bindings in Ada and other languages by checking the Postscript files that are generated by each example against those generated by the C versions. These examples have been completely re-written in Ada (but retain a C flavor in the names that are given to objects). All of the Ada examples generate exactly the same Postscript as the C versions, Examples 14 and 17 excepted since those operate interactively and don't (normally) make Postscript. Two versions of each example are available, one calling the standard binding and the other the traditional binding. (In development, a sed script does almost all of the conversion automatically.)

## 5 Obtaining the Software

There are three software components that you will need: an Ada compiler, the PLplot library, and the Ada bindings.

### 5.1 Obtaining an Ada compiler

You will need an Ada compiler in order to use the Ada PLplot bindings. There are several compilers available. Here, we will focus on the free, open source compiler that is included with the GNU Compiler Collection, (gcc) which is at the center of much of the open source software movement. The gcc Ada compiler is known as GNAT, for GNU

NYU Ada Translator, where NYU stands for New York University. (Although GNAT was originally developed at NYU, it has for many years been developed and supported commercially by AdaCore with academic and pro versions available.)

Your computer may already have GNAT installed, or you can download it from gcc.gnu.org. Another route to obtaining GNAT is from the AdaCore page, libre2.adacore.com. There are versions for many operating systems and processors including Apple's OS X or its open source version Darwin, Linux, and Windows. The gcc and AdaCore versions differ in their licenses. Download the version that you need and follow the installation instructions.

### 5.2 Download and install PLplot

PLplot can be downloaded from the PLplot home page at sourceforge.net—plplot. Follow the installation instructions after downloading. The installation process is more involved than other open source software and requires that your computer has cmake installed. OS X users can try installing PLplot in its entirety from MacPorts but that activity is not officially supported by the PLplot developers. The advantage of using MacPorts is that all installation dependencies are automatically installed for you.

### 5.3 Download the Ada bindings to PLplot

The third major software component is the bindings themselves. Since they are currently included with the PLplot software itself, there is no need to download them from another place.

The bindings themselves are six Ada source files named (using GNAT filename extensions) plplot.ads, plplot.adb, plplot_traditional.ads, plplot_traditional.adb, plplothin.ads, plplotthin.adb. There are two additional files, plplot_auxiliary.ads and plplot_auxililary.adb which will be discussed later, in Section 9. These can be stored somewhere on your system's search paths for easy access.

## 6 How to use the Ada bindings

### 6.1 Ada 95 versus Ada 2005

The bindings will work for either Ada 95 or Ada 2005. The only difference that concerns PLplot users is that Ada 2005, in Annex G.3, provides declarations for real-valued vectors and matrices (along with some other functionality). These declarations make available `type Real_Vector` and `type Real_Matrix.`

The installation process for PLplot requires you to select Ada 95 or Ada 2005. After that, the correct bindings are generated automatically depending on your choice. The differences are very minor: If Ada 2005, the type declarations provided according to Annex G.3 are used; if Ada 95, similar type declarations are provided. For the most part, you don't need to think about this much. However, see Section 9, Compilation Notes, if you are using Ada 95 and need to declare vectors or matrices. The design goal in either case is to encourage users to use `Real_Vector` and `Real_Matrix` since these are the "official" versions of these entities as of Ada 2005. Someone using objects based on these type definitions in Ada 95 in their PLplot programs should expect their programs to work without modification if they should switch to Ada 2005.

### 6.2 GNAT versus non-GNAT

The bindings were made using the GNAT compiler and there is a slight dependence on that compiler. Specifically, the `Unrestricted_Access` attribute of GNAT was used in making the function `Matrix_To_Pointers` in plplotthin.adb and in a few callbacks. `Matrix_To_Pointers` is called whenever an Ada matrix (2D array) is passed to a PLplot subroutine. For more about `Unrestricted_Access` attribute, see Implementation Defined Attributes in the GNAT Reference Manual. This dependency shouldn't be difficult to remove by either incorporating the GNAT code which implements it, by following the TO-DO comment near the function definition in plplotthin.adb, or by providing the proper aliasing.

Another GNAT dependency is used to parse command line arguments in a C-like way.

Most of the GNAT dependencies can be found by searching the source code for "`GNAT`" and "`Unrestricted_Access`."

The GNAT dependence, though slight, will no doubt frustrate users of other Ada compilers. We welcome comments from those users, especially comments with specific suggestions on how to remove any GNAT-specific usages.

### 6.3 Sample command line project

It is instructive to present a simple example that can be compiled and run from the command line. Although this example is specific to one installation, it should be fairly straightforward to adapt it to another installation. Toward that end, it is helpful to understand the PLplot lingo of "build directory" and "installation directory."

Here is a simple program that will generate a plot of part of a parabola.

```
with
    PLplot_Auxiliary,
    PLplot;
use
    PLplot_Auxiliary,
    PLplot;

procedure Simple_Example is
    x, y : Real_Vector(-10 .. 10);
begin
    for i in x'range loop
        x(i) := Long_Float(i);
        y(i) := x(i)**2;
    end loop;

    Initialize_PLplot; -- Call this only once.
    Simple_Plot(x, y); -- Make the plot.
    End_PLplot;        -- Call this only once.
end Simple_Example;
```

Next is a bash script that will compile, bind, and link it. It is installation-specific in that paths to the GNAT compiler, PLplot libraries, and BLAS (Basic Linear Algebra System) and LAPACK (Linear Algebra Package) are hard-coded. You will have to adjust the paths to fit your installation. Some Linux installations which have GNAT 4.3 or later (Ada 2005) pre-installed might have already set the paths to the BLAS and LAPACK libraries.

(Note that the G.3 Annex of Ada 2005, in the GNAT version, depends heavily on BLAS and LAPACK. These packages are tried-and-true packages that are available from several places in either C or Fortran versions. The present example is specific to OS X which has them pre-installed.)

```
#!/bin/bash
/usr/local/ada-4.3/bin/gnatmake simple_example.adb \
-aI/usr/local/plplot_build_dir/bindings/ada \
-aL/usr/local/plplot_build_dir/bindings/ada/CMakeFiles/
plplotadad.dir \
-largs \
/usr/local/plplot/lib/libplplotd.dylib \
/Developer/SDKs/MacOSX10.4u.sdk/usr/lib/libblas.dylib \
/Developer/SDKs/MacOSX10.4u.sdk/usr/lib/liblapack.dylib
```

Beware of inadvertent line wraps in the above code.

The resulting binary program can be run by typing

```
./simple_example
```

## 7 Unique Features of the Ada bindings

The Ada bindings have been augmented with a number of features that are not present in other languages which work with PLplot. These features are intended to greatly simplify the use of PLplot; many users will find that they can do most of their work using these "Simple" plotters. Also included are facilities for easily manipulating the PLplot color tables

### 7.1 High-level features for simplified plotting

**Foreground-background control**

`Draw_On_Black, Draw_On_White`

The default for PLplot is to draw its graphics on a black background. A white background can be used instead with `Draw_On_White` or reset to the original mode with `Draw_On_Black`. Each of these manipulates color map 0 by swapping black and white so that e.g.with `Draw_On_White`, formerly white lines on a black background autotmatically become black lines on a white background.

**Simple Plotters**

Several high-level but flexible plotters are available, and more might be added in the future. It is expected that many users will find that these high-level routines are adequate for most of their day-to-day plotting.

`Multiplot_Pairs`

Plot up to five x-y pairs with easy labeling, coloring, line width and styles, justification, and zooming.

`Simple_Plot`

Plot up to five *y*'s against a single *x* with easy labeling and automatic line colors and styles.

`Simple_Plot_Log_X`

Same as `Simple_Plot` but with logarithmic *x*-axis.

`Simple_Plot_Log_Y`

Same as `Simple_Plot` but with logarithmic *y*-axis.

`Simple_Plot_Log_XY`

Same as `Simple_Plot` but with logarithmic *x*- and *y*-axes.

`Simple_Plot_Pairs`

Plot up to five *x*–*y* pairs with easy labeling and automatic line colors and styles.

`Single_Plot`

Plot a single *x*–*y* pair with flexible labels, axis styles, colors, line width and style, justification, and zooming.

`Simple_Contour`

Make a contour plot with labels

`Simple_Mesh_3D`

Easy 3D mesh plot with labels, zooming, and perspective controls

`Simple_Surface_3D`

Easy 3D surface plot with labels, zooming, and perspective controls

**Simple color map manipulations**

PLplot provides extensive manipulation and control of two separate color maps, color map 0 and color map 1. The Ada binding makes basic manipulations easier and also adds facilities for making snapshots of color map 0 so that any state of the map can easlily be restored later. An initial snapshot is taken when the package is initialized so that the default color settings can always be restored after having been changed.

Another set of features lets the user reset the 16 individual colors in color map 0 after a color definition has been changed. It is important to note that while `Set_Pen_Color(Red) (plcol0` in the traditional binding`)` normally does what it says, `Red` simply has the value `1`. If the user changes the color map so that `1` corresponds to another color, then `Set_Pen_Color(Red)` will draw in that color instead of red. To always assure that red is drawn even if the color map has been changed for integer 1, use `Set_Pen_Color(Reset_Red)` instead. These 16 "reset" functions return the appropriate default integer for the specified color but also reset that slot in the color table so that a subsequent call such as `Set_Pen_Color(Red)` will also cause drawing in red.

Color map 1 also gets a easy-to-use makeover for Ada users. There are several pre-built color themes that are useful for quickly making surface and mesh plots, `Color_Themes_For_Map_1_Type`. These color themes can be quickly applied with `Quick_Set_Color_Map_1.`

Miscellaneous other Ada features include a pre-built mask function for `Shade_Regions` that does no masking; perhaps the most useful purpose is to provide a template for writing mask functions that do mask. And there is a handy function for calculating the contour levels for making contour plots.

Color table snapshots

- `Make_Snapshot_Of_Color_Map_0`
- `Restore_Snapshot_Of_Color_Map_0`
- `Restore_Default_Snapshot_Of_Color_Map_0`

Color resetting functions for the 16 colors of color map 0

- `Reset_Black`, `Reset_Red`, … `Reset_White`

Easy manipulation of color map 1

- Pre-built color themes for color map 1: `Color_Themes_For_Map_1_Type`
- Quick application of pre-built color themes: `Quick_Set_Color_Map_1`

Other features

- A pre-built mask function for `Shade_Regions` that does no masking: `Mask_Function_No_Mask`
- An easy way to calculate an array of contour levels for contour plots: `Calculate_Contour_Levels`

## 7.2 Integer-options Given Ada Names

The C version of PLplot uses a number of integers to mean specific things. Unfortunately, the meaning is lost when it it consigned to being a mere integer with no name. The Ada binding partially rectifies this situation by giving names to these integer constants—the integer can still be used if desired. (A more complete and safer rectification would use enumerated types.)

Below is a listing of at least the contexts in which these "re-namings" have been applied. In some cases the entire range of values is listed, but if there are more than about four such values for each context, only a sampling is given.

**Instances**

Colors: `Plot_Color_Type`

- 0 is Black, 1 is Red, etc.

Justification for plots: `Justification_Type`

  - `User_Justified`
  - `Not_Justified`
  - `Justified`
  - `Justified_Square_Box`
- Axis styles: `Axis_Style_Type`
  - `Linear_Major_Grid`
  - `Linear_Minor_Grid`
  - etc.
- Font styles: `Font_Style_Type`
  - `Normal_Font`
  - `Roman_Font`
  - `Italic_Font`
  - `Script_Font`
- Character sets: `Character_Set_Type`
  - `Standard_Character_Set`
  - `Extended_Character_Set`
- Plot orientation: `Orientation_Type`
  - `Landscape`
  - `Portrait`
- Modes for parsing command line arguments: `Parse_Mode_Type`
  - E.g. `PL_PARSE_PARTIAL`
- Descriptions of map outlines (continents, states, etc.): `Map_Type`
  - `Continents`
  - `USA_and_States`
  - `Continents_and_Countries`
  - `USA_States_and_Continents`
- Various style and view options for 3D and surface plots
  - E.g. `Lines_Parallel_To_X`
- Kind of gridding algorithm for interpolating 2D data to a grid: `Gridding_Algorithm_Type`
  - E.g. `Grid_Bivariate_Cubic_Spline_Approximation`
- Flags for histogram style
  - E.g. `Histogram_Default`
- Flags for histogram binning
  - E.g. `Bin_Default`
- Names for color space models
  - Hue, Lightness, Saturation: `HLS`
  - Red, Green, Blue: `RGB`

### 7.3 One-offs

Convenient string handling for `Get_Device_Name` (`plgdev` in the traditional binding); a function version is provided that simplifies the string handling associated with this feature.

Overloaded `Set_Line_Style (plstyl` in the traditional binding) with a version that takes a single argument, `Default_Continuous_Line`. This replaces the awkward situation of calling the normal versions of these procedures with unused arguments simply to set the line style to the default, continuous, line.

The contour plotter `Contour_Plot_Irregular_Data` (`plfcont` in the traditional binding) is provided for making contour plots from irregularly spaced data. This feature is not documented in the PLplot API documentation.

## 8 Parts That Retain a C Flavor

There remains at least one area in the Ada bindings which is still affected by the C underpinnings. This might be cleaned up in future versions. There might be other residual C influence as well.

### 8.1 Map-drawing

`plmapform` as called by `Draw_Latitude_Longitude` (plmap) and `Draw_Latitude_Longitude` (`plmeridians`)

This is the only place in the PLplot bindings where a C subprogram calls an Ada subprogram while passing an array. If the array is unconstrained, there is no guarantee that it will work because C has no way of telling Ada what offset to use for the beginning of the array. But passing a constrained array is acceptable with the downside that the array size must be fixed within the bindings as being large enough to handle any situation; currently, it is sized as `0 .. 2000`. See Example 19 for how this is handled in by the user program. The constrained array is called `Map_Form_Constrained_Array`.

## 9 Known Issues

### 9.1 Stripchart labelling

In Example 17, all of the stripchart labels are the same regardless of the fact that the calling program  sets them to be different. This is likely to affect user programs in the same manner.

### 9.2 Documentation

In numerous places in the documentation, a feature is listed or described as "C only." Many of these features are actually available in Ada. For example, in `Contour_Plot` (`plcont` in the traditional binding), the transformation from array indices to world coordinates is mentioned as "C only" but is actually available in Ada.

## 10 Compilation notes

### 10.1 Ada 95 Versus Ada 2005

As discussed in Section 6.1, the bindings are made to work with Ada 95 but to also take advantage of the Annex G.3 (vector-matrix) features of Ada 2005. Actually, this adaptation takes place during the PLplot build process when `DHAVE_ADA_2007=OFF` or `DHAVE_ADA_2007=ON` is chosen; the appropriate binding source files are generated automatically. User programs will work with either compiler type without modification.

### 10.2 GNAT Dependence

There is a slight but significant dependence on the GNAT version of Ada. This is discussed more fully in Section 6.2

### 10.3 Compiler Warnings

During normal compilation of the Ada bindings, approximately a dozen warnings are generated, in pairs, of the following form:

```
bar.adb:46: warning: type of argument "foo" is unconstrained array
bar.adb:46: warning: foreign caller must pass bounds explicitly
```

These are normal and an unavoidable consequence of some of the callback routines interacting with the underlying C code.

### 10.4 PLplot_Auxiliary

The bindings include files PLplot_Auxiliary.ads and PLplot_Auxiliary.adb. These files are currently used to provide a few convenience subprograms that are used in the examples. However, they are also very tightly associated with the above-mentioned facility to accommodate either Ada 95 or Ada 2005 compilers. The current situation is such that if the user is using an Ada 95 compiler *and* requires the `Real_Vector` or `Real_Matrix` type definitions, then he/she should `with PLplot_Auxiliary`. If in doubt, `PLplot_Auxiliary` can always be `with`-ed without harm. In the future, this confusion might be removed and the need for PLplot_Auxiliary removed. (However, user programs that `with` it should still work without change.)

## 11 Notes for Apple Macintosh OS X users

The following comments apply to users of Apple Macintosh computers which run OS X. OS X users may use Apple's free integrated development environment (IDE) or may prefer other methods such as using a favorite editor and building from the command line.

OS X users should be aware that an excellent graphical terminal program is available and is highly recommended. It is called AquaTerm and is a full Cocoa program with window control. Performing a cut operation places a PDF of the front window on the clipboard, a convenience when working with other graphics or word processing programs.

### 11.1 Using Apple's Xcode IDE

The Macintosh Ada community has made a plug-in for Apple's free Xcode integrated development environment (IDE) that makes programming Ada in Xcode possible. The plug-in is included with the compiler that is available at www.macada.org. Since Xcode is based on gcc, it is possible to work in the various gcc languages as well as to incorporate binaries such as the PLplot library.

In order to make an Xcode project, drag-and-drop source files and the PLplot library file to the Groups & Files pane of an Ada project. There are a few idiosyncrasies that you may encounter so make sure to contact the very friendly Macintosh Ada mailing list at www.macada.org or study the FAQ at that same site if you have any difficulties.

### 11.2 A Note on AquaTerm

AquaTerm is a display option available on Macintosh computers using OS X and is supported by PLplot. It is a native Cocoa graphics "terminal" that is highly recommended. All output is antialiased and is easily cut-and-pasted in OS X's native PDF format. Get it at sourceforge.net—showfiles.php. It can also be installed from either the Fink www.finkproject.org or MacPorts www.macports.org—projects. projects.

### 11.3 X11

Apple supplies the X11 windowing system that is popular on some other Unix and Linux operations systems as part of the Developer Tools. All PLplot programs made with the Ada bindings will run on X11. In fact, some types of interactivity such as Examples 14 and 17 will not run on Apple's X11 (as of OS X 10.4 at least) and must be run on X11 (or some other output device such as TCL/TK).

### 11.4 GNAT for OS X

Apple Macintosh users will benefit from a pre-built version of GNAT that comes packaged using the usual Apple software installer and is strongly recommended. This compiler is available for both PowerPC and Intel Macintoshes at www.macada.org—Welcome.html.. This site is traditionally rather confusing but the mailing list is extremely helpful. The installer also includes an Ada-specific plug-in for Apple's Xcode IDE which is strongly recommended if you plan to work on this platform. Xcode is part of the Developer Tools and is available on the Apple system disks that also contain the operating system or it can be downloaded for free from developer.apple.com—xcode.

# Ada Bindings for the PLplot Plotting Package

Document version dated September 8, 2008
Author: Jerry Bauck

Distributed as part of the PLplot plotting software