An AVS Module to Convert Geographic Coordinates to Cartesian Coordinates Using Map Projection Functions

by

Evelyn L. Wright

U.S. Geological Survey Open File Report #95-593

July 1995
St. Petersburg, FL 33701

This report is preliminary and has not been reviewed for conformity with U.S. Geological Survey editorial standards (or with the North American Stratigraphic Code). Any use of trade, product, or firm names is for descriptive purposes only and does not imply endorsement by the U.S. Government.

ABSTRACT

This report describes a software module for the Application Visualization System (AVS) that converts geographic longitude and latitude coordinates to cartesian coordinates by using a wide variety of cartographic projection functions. The module was written to facilitate visualization of geographic data covering a wide region without spatial distortion. The C language source code, Makefile, and help file are included.

INTRODUCTION

At the U.S. Geological Survey we are using the Application Visualization System (AVS) software environment developed by AVS Inc. to visualize the results of an ocean circulation and sediment transport model for Massachusetts Bay [Signell and Wright, 1993]. Because the data covers a wide geographic region, spatial distortion is a problem unless map projections are used. Therefore, we found it necessary to write an AVS module to use map projections to convert geographic longitude and latitude coordinates to cartesian coordinates.

Initially written to satisfy our needs, module "Proj_coords" was made as general as possible so that it might be useful to the visualization community at large. "Proj_coords" uses the cartographic projection software system PROJ.4 [Evenden, 1990]. About 70 cartographic projections and inverse projections are provided as well as conversions of State Plane Coordinate Systems to and from geographic coordinates.

Module "Proj_coords" provides simple character-string type-in widgets for specifying the choice of projection and accompanying cartographic parameters. On-line help is provided in the form of a text browser for viewing help files that users can create in advance. The help files may contain whatever information the user considers most relevant for his application, such as which projections to use and the correct parameters to specify.

The C source code, the Makefile for generating the executable program, and the help file for the module are available via anonymous ftp from the International AVS Center (IAC) at the North Carolina Supercomputing Center (avs.ncsc.org). They are also included in the appendices of this report. The module requires PROJ.4 code, which can be acquired from anonymous ftp site charon.er.usgs.gov.

REFERENCES

1. G.I. Evenden. Cartographic Projection Procedures for the Unix Environment--A User's Manual, U.S. Geological Survey Open-File Report 90-284, 64 pages, 1990.

2. R.P. Signell and E.L. Wright. Circulation and Plume Tracking in Massachusetts Bay. In Proceedings of the Second Annual International AVS Conference, pages 74-86, Lake Buena Vista, Florida, 1993.

Appendix A. Source Code (Proj_coords.c)

```
/* Module Name: "Proj_coords" (Filter) (Subroutine)                    */
/* Author:  Evelyn Wright
         U.S. Geological Survey
         Branch of Atlantic Marine Geology
         Woods Hole, MA 02543
         (508) 457-2332
         ewright@usgs.gov                                              */
#include <stdio.h>
#include <avs/avs.h>
#include <avs/port.h>
#include <avs/field.h>
#include <ctype.h>
#include <string.h>
#include "projects.h"

#define MAX_PARGS 100
#define MIN(a,b)  (((a) < (b)) ? (a) : (b))
#define MAX(a,b)  (((a) > (b)) ? (a) : (b))

/* ***************************************/
/*  Module Description                   */
/* ***************************************/
int Proj_coords_desc()
{
        int in_port, out_port, param, iresult;
        extern int Proj_coords_compute();

        AVSset_module_name("Proj_coords", MODULE_FILTER);

        /* Input Port Specifications                */
        in_port = AVScreate_input_port("coords",
                "field irregular", REQUIRED);

        /* Output Port Specifications               */
        out_port = AVScreate_output_port("proj_coords",
                "field irregular");

        /* Parameter Specifications                 */
        param = AVSadd_parameter("option_01", "string", "", "", ":");
        AVSconnect_widget(param, "typein");
        AVSadd_parameter_prop(param,"width","integer",4);
        param = AVSadd_parameter("option_02", "string", "", "", ":");
        AVSconnect_widget(param, "typein");
        AVSadd_parameter_prop(param,"width","integer",4);
        param = AVSadd_parameter("option_03", "string", "", "", ":");
        AVSconnect_widget(param, "typein");
        AVSadd_parameter_prop(param,"width","integer",4);
        param = AVSadd_parameter("option_04", "string", "", "", ":");
        AVSconnect_widget(param, "typein");
        AVSadd_parameter_prop(param,"width","integer",4);
        param = AVSadd_parameter("option_05", "string", "", "", ":");
        AVSconnect_widget(param, "typein");
        AVSadd_parameter_prop(param,"width","integer",4);
        param = AVSadd_parameter("option_06", "string", "", "", ":");
        AVSconnect_widget(param, "typein");
        AVSadd_parameter_prop(param,"width","integer",4);
        param = AVSadd_parameter("option_07", "string", "", "", ":");
        AVSconnect_widget(param, "typein");
        AVSadd_parameter_prop(param,"width","integer",4);
        param = AVSadd_parameter("option_08", "string", "", "", ":");
        AVSconnect_widget(param, "typein");
        AVSadd_parameter_prop(param,"width","integer",4);
```

```
        param = AVSadd_parameter("option_09", "string", "", "", ":");
        AVSconnect_widget(param, "typein");
        AVSadd_parameter_prop(param,"width","integer",4);
        param = AVSadd_parameter("option_10", "string", "", "", ":");
        AVSconnect_widget(param, "typein");
        AVSadd_parameter_prop(param,"width","integer",4);
        param = AVSadd_parameter("option_11", "string", "", "", ":");
        AVSconnect_widget(param, "typein");
        AVSadd_parameter_prop(param,"width","integer",4);
        param = AVSadd_parameter("option_12", "string", "", "", ":");
        AVSconnect_widget(param, "typein");
        AVSadd_parameter_prop(param,"width","integer",4);

        param = AVSadd_float_parameter("X scale factor", 1.00000,
                FLOAT_UNBOUND, FLOAT_UNBOUND);
        AVSconnect_widget(param, "typein_real");
        AVSadd_parameter_prop(param,"width","integer",4);
        param = AVSadd_float_parameter("Y scale factor", 1.00000,
                FLOAT_UNBOUND, FLOAT_UNBOUND);
        AVSconnect_widget(param, "typein_real");
        AVSadd_parameter_prop(param,"width","integer",4);
        param = AVSadd_parameter("inverse", "boolean", 0, 0, 1);
        AVSconnect_widget(param, "toggle");
        param = AVSadd_parameter("browse file", "string", "", "", ":");
        AVSconnect_widget(param, "browser");
        param = AVSadd_parameter("file contents", "string", "", "", ":");
        AVSconnect_widget(param, "text_browser");
        AVSadd_parameter_prop(param,"height","integer",10);

        AVSinitialize_output(in_port,out_port);

        AVSset_compute_proc(Proj_coords_compute);
        return(1);
}

/* ****************************************/
/* Module Compute Routine                 */
/* ****************************************/
int Proj_coords_compute( inp, out,
option_01, option_02, option_03, option_04, option_05,
option_06, option_07, option_08, option_09, option_10,
option_11, option_12, X_scale_factor, Y_scale_factor,
inverse, browse_file, file_contents)
        AVSfield *inp;
        AVSfield **out;
        char *option_01;
        char *option_02;
        char *option_03;
        char *option_04;
        char *option_05;
        char *option_06;
        char *option_07;
        char *option_08;
        char *option_09;
        char *option_10;
        char *option_11;
        char *option_12;
        float *X_scale_factor;
        float *Y_scale_factor;
        int inverse;
        char *browse_file;
        char *file_contents;
```

```
{
        int i, nelem, nxdim;
        float min_extent[AVS_FIELD_NSPACE]; /* minimum extent of coordinates */
        float max_extent[AVS_FIELD_NSPACE]; /* maximum extent of coordinates */

        static char *pargv[MAX_PARGS];
        static int pargc = 0;
        static PJ *Proj;
        UV PXY;

        /* calculate number of data elements in the field */
        nelem = 1;
        for (i=0; i < inp->ndim; i++)
            nelem *= inp->dimensions[i];
        nelem *= inp->veclen;

        /* Copy the input field data to output field */
        switch (inp->type) {
            case AVS_TYPE_BYTE:
                for (i=0; i < nelem; i++)
                    (*out)->field_data[i] = inp->field_data[i];
                break;
            case AVS_TYPE_INTEGER:
                for (i=0; i < nelem; i++)
                    (*out)->field_data_int[i] = inp->field_data_int[i];
                break;
            case AVS_TYPE_REAL:
                for (i=0; i < nelem; i++)
                    (*out)->field_data_float[i] = inp->field_data_float[i];
                break;
            case AVS_TYPE_DOUBLE:
                for (i=0; i < nelem; i++)
                    (*out)->field_data_double[i] = inp->field_data_double[i];
                break;
            case AVS_TYPE_SHORT:
                for (i=0; i < nelem; i++)
                    (*out)->field_data_short[i] = inp->field_data_short[i];
                break;
        }                               /* End of switch */
        AVSmodify_parameter("file contents",AVS_VALUE,browse_file,NULL,NULL);
        pargc = 0;
        if (strcmp(option_01,"")) pargv[pargc++] = option_01;
        if (strcmp(option_02,"")) pargv[pargc++] = option_02;
        if (strcmp(option_03,"")) pargv[pargc++] = option_03;
        if (strcmp(option_04,"")) pargv[pargc++] = option_04;
        if (strcmp(option_05,"")) pargv[pargc++] = option_05;
        if (strcmp(option_06,"")) pargv[pargc++] = option_06;
        if (strcmp(option_07,"")) pargv[pargc++] = option_07;
        if (strcmp(option_08,"")) pargv[pargc++] = option_08;
        if (strcmp(option_09,"")) pargv[pargc++] = option_09;
        if (strcmp(option_10,"")) pargv[pargc++] = option_10;
        if (strcmp(option_11,"")) pargv[pargc++] = option_11;
        if (strcmp(option_12,"")) pargv[pargc++] = option_12;

        if (pargc < 1) return(1);
        if (!(Proj = pj_init(pargc, pargv))) {
            AVSmessage("version 1.0",AVS_Error,NULL, "proj initialization",
                NULL,"Cause of error:  %s", pj_strerrno(pj_errno));
            return(0);
            }
```

```
    /* Get min and max extents of coordinates */
    AVSfield_get_extent(*out,min_extent,max_extent);

    /* Assume X is longitude & Y is latitude, in degrees */
    /* Assume AVS irregular field that's at least 2-space */
    nxdim = 1;
    for (i=0; i < inp->ndim; i++)
        nxdim *= inp->dimensions[i];
    for (i=0; i < nxdim; i++) {
        if (inverse) {
            /* Inverse projection of X and Y coordinates  */
            PXY.u = inp->points[i]*(*X_scale_factor);
            PXY.v = inp->points[i+nxdim]*(*Y_scale_factor);
            PXY = pj_inv(PXY,Proj);
            if (PXY.u == HUGE_VAL || pj_errno != 0){
                AVSmessage("version 1.0",AVS_Error,NULL,"proj computation",
                    NULL,"Cause of error:  %s", pj_strerrno(pj_errno));
                pj_free(Proj);
                return(0);
                }
            PXY.u *= RAD_TO_DEG;
            PXY.v *= RAD_TO_DEG;
            }   /* End of inverse projection */
        else {
            /* Forward projection of X and Y coordinates  */
            PXY.u = inp->points[i]*DEG_TO_RAD;
            PXY.v = inp->points[i+nxdim]*DEG_TO_RAD;
            PXY = pj_fwd(PXY,Proj);
            if (PXY.u == HUGE_VAL || pj_errno != 0){
                AVSmessage("version 1.0",AVS_Error,NULL,"proj computation",
                    NULL,"Cause of error:  %s", pj_strerrno(pj_errno));
                pj_free(Proj);
                return(0);
                }
            PXY.u *= (*X_scale_factor);
            PXY.v *= (*Y_scale_factor);
            }   /* End of forward projection */

        (*out)->points[i] = (float)PXY.u;
        (*out)->points[i+nxdim] = (float)PXY.v;
        /* Determine new min and max extents of coordinates */
        if (i==0) {
            min_extent[0] = (float)PXY.u;
            max_extent[0] = (float)PXY.u;
            min_extent[1] = (float)PXY.v;
            max_extent[1] = (float)PXY.v;
            }
        else {
            min_extent[0] = MIN(min_extent[0],(float)PXY.u);
            max_extent[0] = MAX(max_extent[0],(float)PXY.u);
            min_extent[1] = MIN(min_extent[1],(float)PXY.v);
            max_extent[1] = MAX(max_extent[1],(float)PXY.v);
            }
        }           /* End of projection loop */

    /* Set new min and max extents of coordinates */
    AVSfield_set_extent(*out,min_extent,max_extent);

    pj_free(Proj);
    return(1);
}
```

```
/* ************************************************************************/
/* Initialization for modules contained in this file.                     */
/* ************************************************************************/
static int ((*mod_list[])()) = {
        Proj_coords_desc
};
#define NMODS (sizeof(mod_list) / sizeof(char *))

AVSinit_modules()
{
        AVSinit_from_module_list(mod_list, NMODS);
}

/* ************************************************************************/
/* User-supplied subroutines, functions, utility routines                 */
/* ************************************************************************/

/* allocate and deallocate memory  --  proj routines for AVS */
        void *
pj_malloc(size_t size) {
        return(ALLOC_LOCAL(size));
}
        void
pj_dalloc(void *ptr) {
        FREE_LOCAL(ptr);
}
```

Appendix B. Makefile

```
# Edit the following for your site

# Path for AVS lib and include directories
AVS_PATH=/net/dogbert/chum1/avs/avs

# PROJ libraries and header directories
PROJ_LIB = /net/dogbert/chum1/ew/lib
PROJ_INC = -I/net/dogbert/chum1/ew/include

# End of editable site parameters

INC_FILE=$(AVS_PATH)/include/Makeinclude
include $(INC_FILE)

AVS_LIBS = $(AVS_PATH)/lib
AVS_INC = -I$(AVS_PATH)/include
BASELIBS=-lgeom -lutil -lm $(LASTLIBS)
CFLOWLIBS=-L$(AVS_LIBS) -lflow_c $(BASELIBS)
CSIMLIBS=-L$(AVS_LIBS) -lsim_c $(BASELIBS)
CFLAGS = -I./ $(AVS_INC) $(PROJ_INC)
CC = gcc

Proj_coords:    Proj_coords.o
 $(CC) $(CFLAGS) -o Proj_coords Proj_coords.o -L$(PROJ_LIB) -lproj $(CFLOWLIBS)
```

Appendix C. On-Line Help File (Proj_coords.txt)

AVS Local Modules
U.S. Geological Survey

Proj_coords
July 1995

NAME

Proj_coords - transforms geographic longitude-latitude coordinates to X-Y cartesian coordinates using standard cartographic projections

SUMMARY

Name Proj_coords

Author Evelyn Wright (ewright@usgs.gov)

Type filter

Input field irregular

Output field irregular

Parameter Name	Type	Default	Min	Max
option_01	typein			
option_02	typein			
option_03	typein			
option_04	typein			
option_05	typein			
option_06	typein			
option_07	typein			
option_08	typein			
option_09	typein			
option_10	typein			
option_11	typein			
option_12	typein			
X scale factor	typein_real	1.0	FLOAT_UNBOUND	FLOAT_UNBOUND
Y scale factor	typein_real	1.0	FLOAT_UNBOUND	FLOAT_UNBOUND
inverse	toggle	off		
browse file	file browser			
file contents	text browser			

DESCRIPTION

This module transforms geographic longitude and latitude coordinates to cartesian coordinates by applying a choice of approximately 70 cartographic projection functions. The inverse conversion is also available for many of the projections. The input AVS field must be irregular but may be any spatial dimension; the data may be vector or scalar, any valid AVS type, and any dimension. It is assumed that the X coordinate is longitude and the Y coordinate is latitude. This module requires the cartographic projection software system PROJ.4 available by anonymous ftp from charon.er.usgs.gov. Documentation for this system is also posted at the same site. It is necessary to refer to the PROJ documentation for details on specifying cartographic options.

APPLICATION

This module is useful for displaying data that covers a wide geographic region. In these cases, spatial distortion can be a problem unless map projections are used. The inverse conversion is useful whenever the user wants to manipulate geographic coordinates instead of X-Y cartesian coordinates. The cartesian coordinates computed by this module are a distance measure in meters.

INPUT

The input data is required to be an irregular field. The X and Y coordinates are assumed to be longitude and latitude, respectively, for forward projections. For inverse projections, the X and Y coordinates will be transformed to longitude and latitude, respectively.

OUTPUT

The output is the same type and size as the input.

PARAMETERS

option_01 . . . option_12	These typeins allow the user to specify the necessary PROJ cartographic parameters, e.g., choice of projection and associated parameters. See USAGE below.
X scale factor Y scale factor	These are multiplicative factors for scaling the cartesian coordinates, which are in meters unless you scale them (or specify different units with the PROJ "units=" parameter described under USAGE).
inverse	This toggle allows selection of the inverse conversion for many of the projections, i.e., X and Y cartesian coordinates to geographic longitude and latitude.
browse file	This file browser allows the user to select a file for viewing its contents. For example, users may wish to set up help files that explain cartographic options.
file contents	This text browser displays the contents of the "browse file" selected by the user.

USAGE

To use this module effectively, you must read and understand the portions of the PROJ documentation indicated under SEE ALSO. When reading the documentation, keep in mind that the "-" options do not pertain to this module; only the "+" options pertain to this module.

Understanding the concept of initialization and default files explained in the Interim Report referenced under SEE ALSO is also very important. Use of this module can be made easier by creating initialization and default files that satisfy your own unique requirements. The PROJ.4 system provides the initialization files nad27 and nad83 for U.S. State Plane Coordinate Systems and provides the default file proj_def.dat.

The option_xx typein parameters for this module are used for specifying the PROJ "+control" cartographic parameters explained in the PROJ documentation. The user enters one cartographic control parameter per option_xx typein, with or without the preceding "+" sign. These typeins are processed first, followed by the initialization file and then the default file. Reentry of an option is ignored with the first occurrence taking precedence.

A few of the most useful PROJ cartographic parameters explained in the PROJ documentation are as follows:

init=file:key	-	Names an initialization file containing cartographic control parameters associated with the keyword key.
proj=name	-	Selects the cartographic transformation function, where name is an acronym for the desired projection.
x_0=value	-	Specifies false easting; value is added to X cartesian coordinate. Used in grid systems to avoid negative grid coordinates. Default value is 0.
y_0=value	-	Specifies false northing; value is added to Y cartesian coordinate. Used in grid systems to avoid negative grid coordinates. Default value is 0.
lon_0=value	-	Specifies central meridian. Along with lat_0, determines the geographic origin of the projection.
lat_0=value	-	Specifies central parallel. See lon_0.
units=name	-	Selects conversion of cartesian values to the specified units. Default is meters. See PROJ documentation.

EXAMPLES OF USAGE

1. An example of converting longitude and latitude coordinates for mainland Massachusetts to State Plane Coordinates at a scale of 1:100,000 using initialization file nad27 supplied with PROJ.4:

```
option_01:  init=nad27:2001
option_02:  units=us-ft
X scale factor:  1e-05
Y scale factor:  1e-05
```

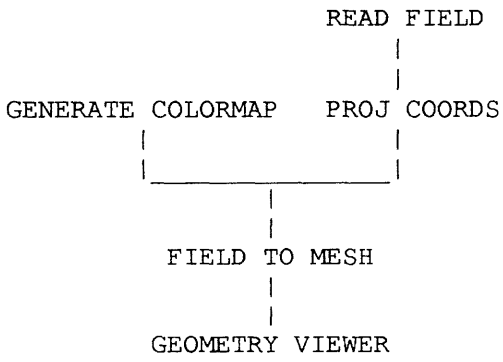
2. An example of converting longitude and latitude coordinates for the continental United States at a scale of 1:20,000,000 using Albers Equal Area projection with standard parallels 29.5 and 45.5 degrees north. Central meridian of 90 degrees west and central parallel of the equator determine the geographic origin of the projection. This projection is commonly used for maps of the United States.

```
option_01:  proj=aea
option_02:  lat_1=29.5n
option_03:  lat_2=45.5n
option_04:  lon_0=90w
option_05:  lat_0=0
X scale factor:  2e-07
Y scale factor:  2e-07
```

3. An example of converting longitude and latitude coordinates for the continental United States at a scale of 1:20,000,000 using Lambert Conformal Conic projection with standard parallels 33 and 45 degrees, central meridian 90 degrees west, and central parallel the equator. This is another common projection for the United States.

```
option_01:  proj=lcc
option_02:  lat_1=33n
option_03:  lat_2=45n
option_04:  lon_0=90w
option_05:  lat_0=0
X scale factor:  2e-07
Y scale factor:  2e-07
```

EXAMPLE AVS NETWORK



SEE ALSO

1. "Cartographic Projection Procedures Release 4 Interim Report",G.I. Evenden, Feb. 1994, pages 7 - 11, 17 - 22, 36 - 37 (ftp anonymous to charon.er.usgs.gov, file /pub/PROJ.4.1.3.ps.Z for 300dpi, black-on-white PostScript printers).

2. "Cartographic Projection Procedures for the Unix Environment--A User's Manual", G.I. Evenden, USGS Open-File Report 90-284, 1990, pages 10 - 57 (ftp anonymous to charon.er.usgs.gov, file /pub/proj.OF90-284.ps for 300dpi, black-on-white PostScript version).

3. "Map Projections - A Working Manual", John P. Snyder, U.S. Geological Survey Professional Paper 1395, 1987, 383 pages.

4. "An Album of Map Projections", John P. Snyder and Philip M. Voxland, U.S. Geological Survey Professional Paper 1453, 1989, 250 pages. (Visual presentation of the distortion of various map projections.)

Note: The above USGS Open-File Report and USGS Professional Papers can be ordered from USGS Books and Reports Sales, Federal Center, Box 25425, Denver, CO 80225, (303) 236-7476.