An Example of Interfacing AVS and NetCDF

by

Evelyn L. Wright

U.S. Geological Survey Open File Report #92-332

May 1992
Woods Hole, MA 02543

ABSTRACT

This report describes a prototype software module for interfacing UCAR/Unidata Network Common Data Form (netCDF) data files with the Application Visualization System (AVS) from AVS Inc. The module read_netcdf is specific to netCDF data files created by an ocean circulation model that has three-dimensional, curvilinear spatial coordinates that vary with time; however, the module may be useful as a template for developers with similar needs. The module creates an AVS three-dimensional, irregular field containing scalar data. The C language source code, Makefile, help file, and listings of sample input and output data are included.

INTRODUCTION

An ocean circulation and sediment transport model for Massachusetts Bay produces large data sets of three-dimensional variables such as currents, salinity, temperature, turbulent intensity, momentum fluxes and dispersion coefficients computed for a curvilinear, three-dimensional spatial grid that varies with time. Because of the large quantity of data and the need for the data to be transportable to multiple computer systems, the data is stored in netCDF files. NetCDF (UCAR, 1991) is the UCAR/Unidata Network Common Data Form: a public-domain, hardware-independent, self-describing binary format. Because visualization of the data is required for interpretation purposes, it is desirable to import the data into AVS. AVS is the Application Visualization System software environment for scientific visualization developed by AVS Inc.

Retrieval of data from netCDF files requires use of the netCDF interface, and this in turn requires a special AVS module to read the data directly into AVS. Because no such module was available from either AVS Inc. or the International AVS Center at the North Carolina Supercomputing Center, a decision was made to write our own module. This report describes the results of our initial efforts.

The C source code for read_netcdf is listed in Appendix A, and the Makefile for generating the executable program is shown in Appendix B. Note that the netCDF software library is required. The module runs successfully on a DECstation 5000 under Ultrix 4.2 and AVS 3.0. Appendix C contains a listing of the AVS on-line help file (i.e., "man" file) for the module.

OVERVIEW

The netCDF data files read by the module read_netcdf are generated by the regional circulation model described above. All of the data variables in the files (except sea-surface elevation) are a function of three spatial dimensions and time. The spatial coordinates are irregular. The X and Y axes (in the horizontal plane) represent easting and northing geographical positions. An example of an X-Y coordinate grid for a data set is shown in Figure 1. The vertical axis represents percentage of total depth that has been adjusted for sea-surface elevation (i.e., tide).

An example of a netCDF file that can be read by read_netcdf is shown in Figure 2. Because the actual data file is binary, the netCDF utility ncdump was used to produce this human-readable listing of the contents of the file. The data in this file is a very small subset of an actual data set generated by the model. The variable salt (i.e., salinity) is the data variable read by module read_netcdf. The X and Y coodinates are the variables x and y. The Z coordinates are a function of the variables depth, elev (i.e., sea-surface elevation) and sigma (i.e., fractional portion of total depth). Notice that the variables elev and salt are declared as short integers. This is to conserve space since the data files are usually quite large. These variables have limited dynamic range and therefore can be scaled down. The actual values are obtained by multiplying the integer values by the indicated scale_factor and then adding the add_offset value.

The output from read_netcdf is an AVS three-dimensional, irregular field containing scalar data. The results of reading the example netCDF file described in Figure 2 are shown in Figures 3 and 4. Figure 3 shows the contents of the created AVS field as displayed by the AVS module print_field, and Figure 4 shows the results of running the AVS module volume_bounds (with all parameters turned on) on the created field. Figure 5 shows the results of running volume_bounds on a full data set. Once the netCDF data has been converted into an AVS field, the user is able to do whatever visualizations he desires using the functionality of AVS.

DETAILED DESCRIPTION

To be able to use the module read_netcdf, the executable program must be made available to AVS by invoking the AVS Network Editor and then selecting Module Tools, Read Module, and the filename of the executable program in the directory in which it resides. Successful importing of the module is indicated by a rectangular icon box labelled "read netcdf" appearing in the Data Input column of the Network Editor. To make the on-line help file listed in Appendix C available to AVS, the file must be named read_netcdf.txt and the directory in which it resides specified in the HelpPath command in the user's .avsrc file.

When read_netcdf is invoked by moving its icon box to the Network Editor workspace, a file browser and a typein widget appear in the control panel on the screen. The file browser is labelled "Read netCDF File" and displays file names that have the extension ".cdf". It is assumed that netCDF files to be read by the module have that extension. The user makes his selection of an input file from the file browser. The typein widget is labelled "time record" and indicates which time step of the data is to be read. Only one time step of the data is read for each invocation of the module. The default time step record is zero, which means that the first time record is read.

The netCDF files to be read by the module must contain the following dimension names: xpos, ypos, zpos, and time, where the size of dimension time is declared as UNLIMITED. The following variables must be in the file: float variables sigma(zpos), x(ypos,xpos), y(ypos,xpos), and depth(ypos,xpos) and short variables elev(time,ypos,xpos) and salt(time,zpos,ypos,xpos). The variables elev and salt must have the following float attributes: scale_factor and add_offset, where the scale factor is to be multiplied first and then the offset added to the data.

The data portion of the AVS field to be created is the data from the variable salt. The coordinates are irregular and therefore an X, Y and Z coordinate value is required for each data point. All the X's are stored first, followed by all the Y's and then the Z's. The X and Y coordinates are scaled in the program by a factor of 1/4000 to change the values to a reasonable range. The Z locations are calculated as follows:

$$Z = sigma*(depth + elev) + elev$$

LIMITATIONS AND FUTURE PLANS

At present, read_netcdf is specific to the ocean circulation model described in this report. The only generalizations in the program are specification by the user of the netCDF input file and desired time step, and automatic determination by the program of the size of the netCDF data dimensions so that the proper amount of memory for storing the data variables can be allocated automatically.

The present module does not allow any flexibility on names of dimensions and variables, dimensionality and type of variables, required attributes (e.g., add_offset), and method of computing the coordinate values used for the AVS field data. Also, there is no choice on the type of AVS field created. The coordinates are assumed to be three-space and irregular (rather than uniform or rectilinear), and the data is assumed to be of type float, three-dimensional, and scalar (rather than vector).

Because reading netCDF data into AVS is of more general interest, effort is being made to make module read_netcdf more flexible and able to process a more general form of netCDF files. In the meantime, it is hoped that the source code of the present version will be useful as a template for writing AVS modules to read other netCDF data files.

REFERENCES

1. Unidata Program Center, 1991, NetCDF User's Guide: An Interface for Data Access, University Corporation for Atmospheric Research, Boulder, CO.

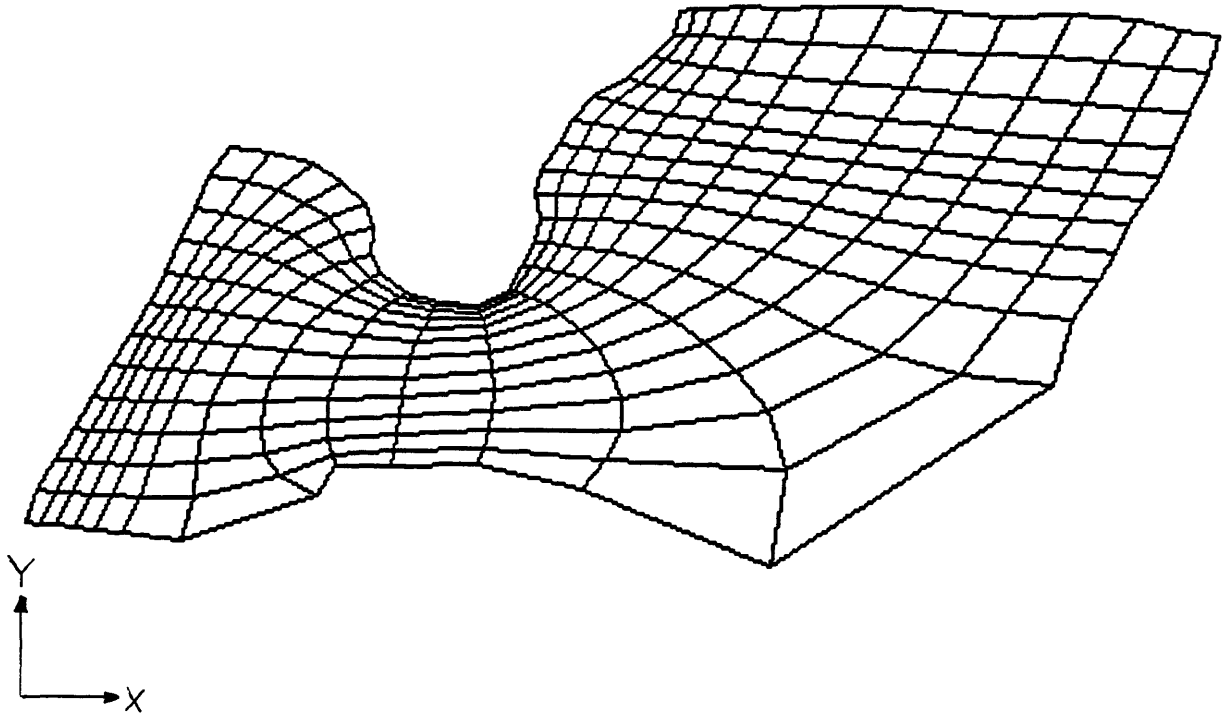


Figure 1. Example of an X-Y grid for a full data set.

```
netcdf small {
dimensions:
        xpos = 4 ;
        ypos = 3 ;
        zpos = 2 ;
        time = UNLIMITED ; // (1 currently)

variables:
        float sigma(zpos) ;
        float x(ypos, xpos) ;
        float y(ypos, xpos) ;
        float depth(ypos, xpos) ;
        short elev(time, ypos, xpos) ;
                elev:scale_factor = 0.00012207404f ;
                elev:add_offset = 0.f ;
        short salt(time, zpos, ypos, xpos) ;
                salt:scale_factor = 0.00061037019f ;
                salt:add_offset = 20.f ;

data:

 sigma = -0.5, -0.60000002 ;

 x =
  664700, 669700, 674700, 679700,
  664700, 669700, 674700, 679700,
  664700, 669700, 674700, 679700 ;

 y =
  262200, 262200, 262200, 262200,
  267200, 267200, 267200, 267200,
  272200, 272200, 272200, 272200 ;

 depth =
  7.5, 7.5, 7.5, 5,
  7.5, 5, 5, 5,
  5, 5, 2.5, 2.5 ;

 elev =
  400, 270, 152, -10,
  431, 280, 138, 21,
  425, 307, 85, -29 ;

 salt =
  8127, 8189, 8191, 8191,
  8069, 8168, 8187, 8190,
  7980, 8138, 8181, 8188,
  8127, 8189, 8191, 8191,
  8071, 8169, 8187, 8190,
  7981, 8138, 8181, 8188 ;
}
```

Figure 2. Sample Input. The above summary of the data in sample netCDF binary input file small.cdf was obtained by running netCDF program ncdump.

```
HEADER:
NDim: 3
NSpace: 3
Veclen: 1
Type: 2 (real)
Size: 4
Uniform: 2 (Irregular)
Dims: [ 4 3 2 ]
Min Extent: [ 166.175 65.55 -4.49258 ]
Max Extent: [ 169.925 68.05 -1.24481 ]

DATA:
(0,0,0) 24.9605  @(166.175,65.55,-3.72559)
(0,0,1) 24.9605  @(166.175,65.55,-4.48047)
(0,1,0) 24.9251  @(166.175,66.8,-3.72369)
(0,1,1) 24.9263  @(166.175,66.8,-4.47895)
(0,2,0) 24.8708  @(166.175,68.05,-2.47406)
(0,2,1) 24.8714  @(166.175,68.05,-2.97925)
(1,0,0) 24.9983  @(167.425,65.55,-3.73352)
(1,0,1) 24.9983  @(167.425,65.55,-4.48682)
(1,1,0) 24.9855  @(167.425,66.8,-2.48291)
(1,1,1) 24.9861  @(167.425,66.8,-2.98633)
(1,2,0) 24.9672  @(167.425,68.05,-2.48126)
(1,2,1) 24.9672  @(167.425,68.05,-2.98501)
(2,0,0) 24.9995  @(168.675,65.55,-3.74072)
(2,0,1) 24.9995  @(168.675,65.55,-4.49258)
(2,1,0) 24.9971  @(168.675,66.8,-2.49158)
(2,1,1) 24.9971  @(168.675,66.8,-2.99326)
(2,2,0) 24.9934  @(168.675,68.05,-1.24481)
(2,2,1) 24.9934  @(168.675,68.05,-1.49585)
(3,0,0) 24.9995  @(169.925,65.55,-2.50061)
(3,0,1) 24.9995  @(169.925,65.55,-3.00049)
(3,1,0) 24.9989  @(169.925,66.8,-2.49872)
(3,1,1) 24.9989  @(169.925,66.8,-2.99897)
(3,2,0) 24.9977  @(169.925,68.05,-1.25177)
(3,2,1) 24.9977  @(169.925,68.05,-1.50142)
    |       |              |
    |       |              X-coordinate,Y-coordinate,Z-coordinate
    |     data-variable
    |
X-index,Y-index,Z-index
```

Figure 3. Sample Output. The above summary of the AVS field created by module read_netcdf from netCDF file small.cdf was obtained by running AVS module write field to write the output from module read netcdf to disc, and then running AVS modules read field and print field on that disc file.

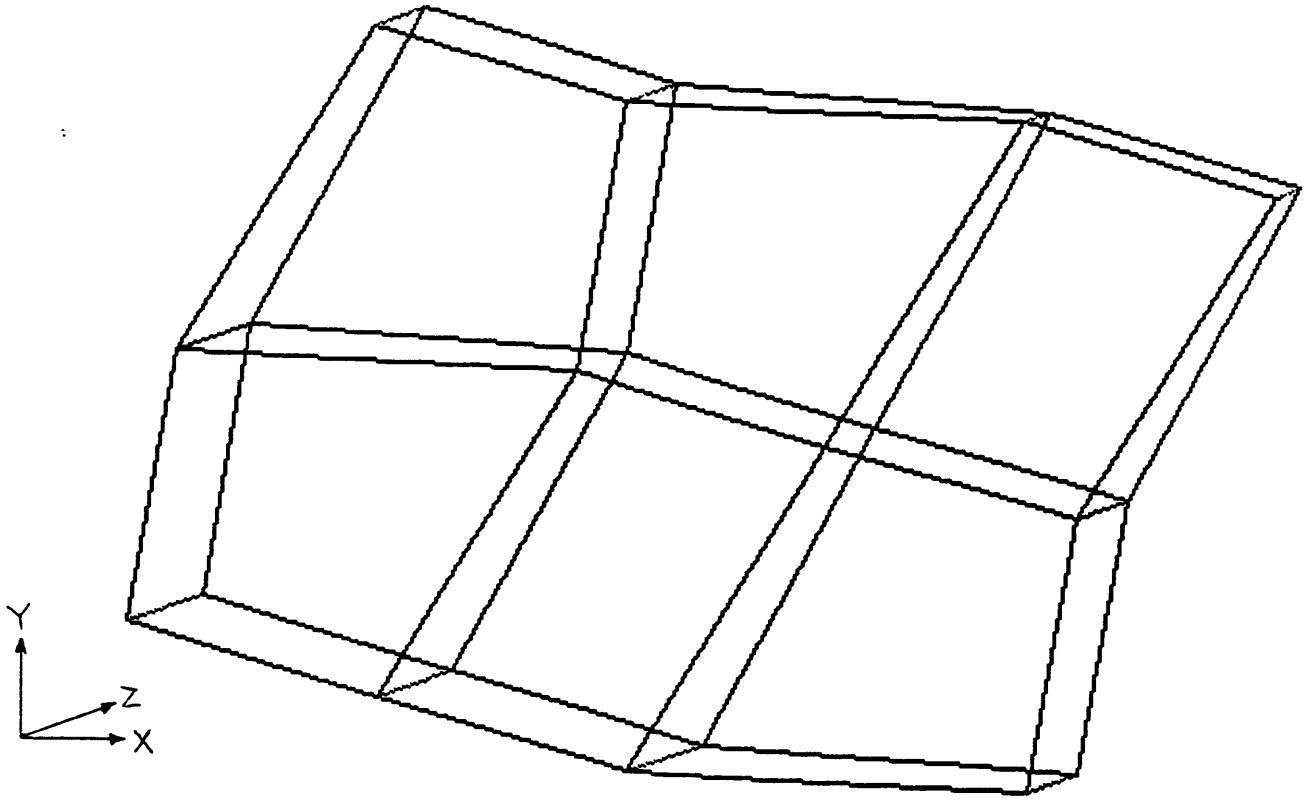


Figure 4. Volume bounds of the sample netCDF data file of Figure 2.

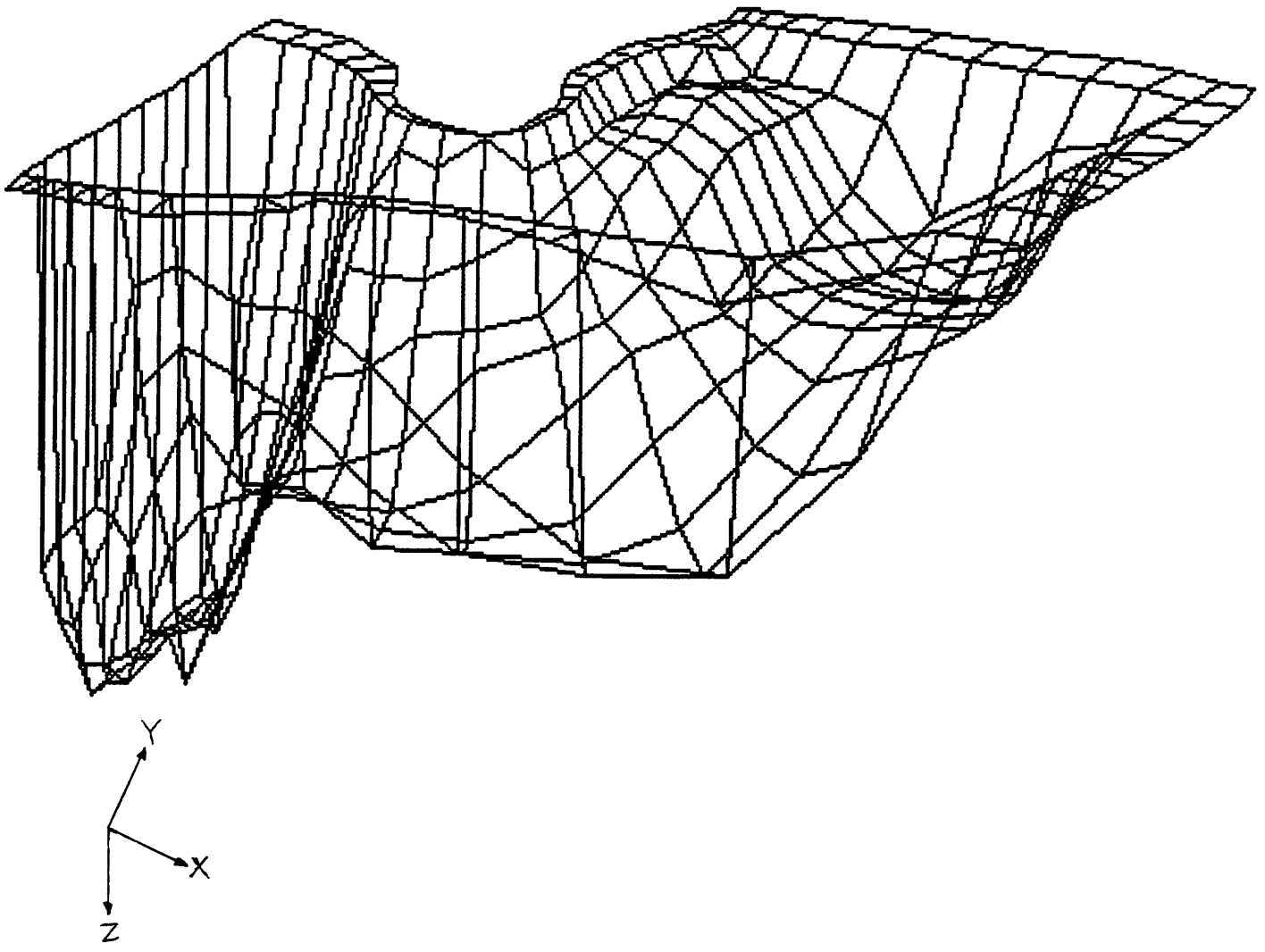


Figure 5. Volume bounds for a full data set.

APPENDIX A. Source Code (read_netcdf.c)

```
#include <stdio.h>
#include <avs/avs.h>
#include <avs/field.h>
#include <avs/avs_data.h>
#include <netcdf.h>
#define MAXT 1000   /* Maximum time record number */
/* Used for AVS message() */
static char file_version[]="read_netcdf.c  Evelyn Wright,USGS,Woods Hole,MA  4/92";

AVSinit_modules()       /* Called by main() routine supplied by AVS */
{
        int read_netcdf_desc();
        AVSmodule_from_desc(read_netcdf_desc);
}
read_netcdf_desc()              /* Description function */
{
        int netcdf_compute();
        int out_port, p1, param;
        AVSset_module_name("read netcdf",MODULE_DATA);
        out_port=AVScreate_output_port("Output Field",
                                "field 3D scalar float 3-coord irregular");
        /* Create file browser that displays filenames with .cdf extension */
        p1 = AVSadd_parameter("Read netCDF File","string",0,0,".cdf");
        AVSconnect_widget(p1,"browser");
        param = AVSadd_parameter("time record","integer",0,0,MAXT);
        AVSconnect_widget(param,"typein_integer");
        AVSset_compute_proc(netcdf_compute);
        AVSautofree_output(out_port);
}
static
netcdf_compute(output,filename,time_rec)        /* Computation function */
AVSfield_float **output; /* AVS field output */
char *filename;          /* netCDF filename specified by user */
int time_rec;            /* time record number specified by user */
{
        AVSfield_float *tmp_field;
        int dims[3],i, kc, k, j ,cdfid;
        int sal_id,sigma_id;
        int depth_id, y_id, x_id, elev_id;
        int xpos_id, ypos_id, zpos_id;
        float tmp;
        char *malloc();
        static int one_start[]={0}, two_start[]={0,0};
        static int elev_start[]={0,0,0}, sal_start[]={0,0,0,0};
        static int two_count[2], elev_count[3], sal_count[4];
        static int xypos, xyzpos;

        int xpos, ypos, zpos;
        float *sigma_vals;
        float *depth_vals, *y_vals, *x_vals;
        short *elev_vals, *sal_vals;

        float sal_scale, sal_offset;
        float elev_scale, elev_offset;
```

```
if (!filename) return(1);                /* no filename yet */
if ((cdfid = ncopen(filename,NC_NOWRITE)) == -1) {
    AVSmessage(file_version,AVS_Warning,NULL,"netcdf_compute",
        "Ok","Can't open data file %s",filename);
    return(0);
}
/* Get ID # of each netCDF dimension */
xpos_id = ncdimid(cdfid,"xpos");
ypos_id = ncdimid(cdfid,"ypos");
zpos_id = ncdimid(cdfid,"zpos");

/* Get size of each netCDF dimension */
ncdiminq(cdfid,xpos_id,(char *) 0,&xpos);
ncdiminq(cdfid,ypos_id,(char *) 0,&ypos);
ncdiminq(cdfid,zpos_id,(char *) 0,&zpos);

/* Parameters for reading netCDF data for user-specified time step */
xypos = xpos*ypos;
xyzpos = xypos*zpos;
two_count[0] = ypos;
two_count[1] = xpos;
elev_start[0] = time_rec;
elev_count[0] = 1;
elev_count[1] = ypos;
elev_count[2] = xpos;
sal_start[0] = time_rec;
sal_count[0] = 1;
sal_count[1] = zpos;
sal_count[2] = ypos;
sal_count[3] = xpos;

/* Allocate memory for netCDF coordinate variables */
if ((sigma_vals=(float *)malloc(zpos*sizeof(float)))==NULL){
    AVSmessage(file_version,AVS_Fatal,NULL,"netcdf_compute",
        NULL,"Can't allocate memory for sigma_vals");
    return(0);
}
if ((depth_vals=(float *)malloc(ypos*xpos*sizeof(float)))==NULL){
    AVSmessage(file_version,AVS_Fatal,NULL,"netcdf_compute",
        NULL,"Can't allocate memory for depth_vals");
    return(0);
}
if ((y_vals=(float *)malloc(ypos*xpos*sizeof(float)))==NULL){
    AVSmessage(file_version,AVS_Fatal,NULL,"netcdf_compute",
        NULL,"Can't allocate memory for y_vals");
    return(0);
}
if ((x_vals=(float *)malloc(ypos*xpos*sizeof(float)))==NULL){
    AVSmessage(file_version,AVS_Fatal,NULL,"netcdf_compute",
        NULL,"Can't allocate memory for x_vals");
    return(0);
}
if ((elev_vals=(short *)malloc(ypos*xpos*sizeof(short)))==NULL){
    AVSmessage(file_version,AVS_Fatal,NULL,"netcdf_compute",
        NULL,"Can't allocate memory for elev_vals");
    return(0);
}
```

```
    /* Get ID # of each netCDF coordinate variable */
    sigma_id = ncvarid(cdfid,"sigma");
    depth_id = ncvarid(cdfid,"depth");
    y_id = ncvarid(cdfid,"y");
    x_id = ncvarid(cdfid,"x");
    elev_id= ncvarid(cdfid,"elev");

    /* Get coordinate data from netCDF file */
    ncvarget(cdfid,sigma_id,one_start,&zpos,sigma_vals);
    ncvarget(cdfid,depth_id,two_start,two_count,depth_vals);
    ncvarget(cdfid,y_id,two_start,two_count,y_vals);
    ncvarget(cdfid,x_id,two_start,two_count,x_vals);
    ncvarget(cdfid,elev_id,elev_start,elev_count,elev_vals);
    ncattget(cdfid,elev_id,"scale_factor",&elev_scale);
    ncattget(cdfid,elev_id,"add_offset",&elev_offset);

    /* Allocate memory for netCDF data variable salinity */
    if ((sal_vals=(short *)malloc(zpos*ypos*xpos*sizeof(short)))==NULL){
        AVSmessage(file_version,AVS_Fatal,NULL,"netcdf_compute",
            NULL,"Can't allocate memory for sal_vals");
        return(0);
    }
    sal_id = ncvarid(cdfid,"salt");        /* Get ID # of salinity data */
    ncvarget(cdfid,sal_id,sal_start,sal_count,sal_vals);    /* Get data */
    ncattget(cdfid,sal_id,"scale_factor",&sal_scale); /* Get attributes */
    ncattget(cdfid,sal_id,"add_offset",&sal_offset);

    ncclose(cdfid);      /* Close the netCDF file */

    /* Allocate space for field data */
    dims[0] = xpos;
    dims[1] = ypos;
    dims[2] = zpos;
    tmp_field = (AVSfield_float *)AVSdata_alloc(
                "field 3D scalar float 3-coord irregular",dims);
    /* Stuff the data into AVS field structure */
    for (i=0; i < xyzpos; i++)
      tmp_field->data[i] = sal_scale * (float)(*sal_vals++) + sal_offset;

    /* Stuff the real-world coordinates into AVS field structure */
    kc = 0;
    for (j=0; j < zpos; j++)
        for (i=0; i < xypos; i++)
            tmp_field->points[kc++] = *(x_vals+i)/4000.;

    for (j=0; j < zpos; j++)
        for (i=0; i < xypos; i++)
            tmp_field->points[kc++] = *(y_vals+i)/4000.;

    for (j=0; j < zpos; j++)
        for (i=0; i < xypos; i++) {
            tmp = elev_scale * (float)(*(elev_vals+i)) + elev_offset;
            if (*(depth_vals+i) < 0.) *(depth_vals+i) = 0.;
            tmp_field->points[kc++] = *(sigma_vals+j) *
                                      (*(depth_vals+i) + tmp) + tmp;
       }
    *output = tmp_field;       /* Set output pointer to new data */
    return(1);                 /* Indicates successful completion */
}
```

APPENDIX B. Makefile

```
# Edit the following for your site

# AVS libraries and header directories
AVS_LIBS = $(ROOT)/usr/avs/lib
AVS_INC = -I$(ROOT)/usr/avs/include

# netCDF libraries and header directories
CDF_LIBS = $(LOCAL)/lib
CDF_INC = -I$(LOCAL)/include

# End of editable site parameters

# Select optional compiler
#CC = gcc
CC = cc

CFLAGS = -O $(AVS_INC) $(CDF_INC)
LDFLAGS = -L$(AVS_LIBS) -L$(CDF_LIBS)
BASELIBS = -lgeom -lutil -lm
FLOWLIBS = -lflow_c $(BASELIBS)
LIBS = -lnetcdf $(FLOWLIBS)

read_netcdf: read_netcdf.o
        $(CC) -o read_netcdf read_netcdf.o $(LDFLAGS) $(LIBS)
```

APPENDIX C. On-Line Help File (read_netcdf.txt)

AVS Local Modules read netcdf(6)

NAME

read netcdf - read a netCDF file that follows specific conventions

SUMMARY

Name	read netcdf				
Type	data				
Inputs	none				
Outputs	field 3D scalar float 3-coord irregular				
Parameters	Name	Type	Default	Min	Max
	Read netCDF File	browser	*.cdf		
	time record	typein	0	0	1000

DESCRIPTION

The read netcdf module is a working example of a module that can read netCDF files. This version can only read netCDF files that contain specific dimension, variable, and attribute names. Also, the coordinate values for the irregular field output are computed in a way that is data specific. See LIMITATIONS below for details.

PARAMETERS

Read netCDF File

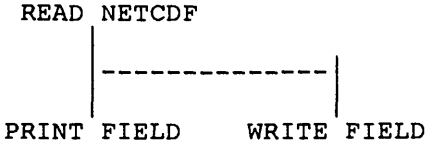
A file browser that displays filenames with the .cdf extension.

time record

Integer typein that defines which time record of data to read. By default, time record 0 (i.e., first record) is read. Only one time record is read. The maximum time record number is currently set to 1000.

EXAMPLE

The following network converts netCDF data into an AVS field, displays the contents of the new field, and gives the person the option of writing the new AVS field permanently to disk.



LIMITATIONS

The netCDF files to be read by this version of read netcdf must contain the following dimension names: xpos, ypos, zpos, and time, where the size of dimension time is declared as UNLIMITED. The following variables must be in the file: float variables sigma(zpos), x(ypos,xpos), y(ypos,xpos), and depth(ypos,xpos) and short variables elev(time,ypos,xpos) and salt(time,zpos,ypos,xpos). Variables elev and salt must have the following float attributes: scale_factor and add_offset, where the scale factor is to be multiplied first and then the offset added to the data.

The data portion of the AVS field output is the data from variable salt. The X and Y coordinates are variables x and y scaled by a factor of 1/4000. The Z coordinates are calculated from variables sigma, depth and elev as follows:

$$Z = sigma*(depth + elev) + elev$$

SEE ALSO

Unidata Program Center, 1991, NetCDF User's Guide: An Interface for Data Access, University Corporation for Atmospheric Research, Boulder, CO.