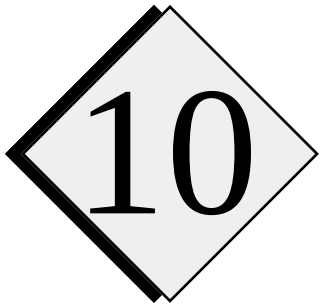

Data I/O and Post-Processing Utilities

10.1. ARPS History Data Format and I/O

As described in Section 3.7, ARPS supports a number of history data formats. They include the unformatted binary, ASCII, NCSA HDF, packed binary, NetCDF, Packed NetCDF, GrADS and Savi3D. The parameters that control the choice of these formats and the selection of the variables to be dumped out are set in the run time input file *arps40.input* (see Chapter 4, Runtime/Configuration Reference Guide). At run time, history dumps in one of these formats can be generated at a specified time interval. All except the Savi3D data can be converted to other formats using a data converter provided (see Section 10.2). Savi3D data are designed to be read by the corresponding visualization software.

The output in most of these formats includes at least two files (except for the Savi3D and GrADS formats, which require data to be contained in a single file at all times). One file, called *runname.fmtgrdbas*, contains the time-independent data arrays (base state and grid arrays). The other files contain the time-dependent arrays at different times, and are named *runname.fmtnnnnnn*. Here *fmt* indicates the format of the data, *nnnnnn* denotes the time of the data in seconds, and *runname* is a character string specified in *arps40.input* and is used to construct unique names for output files of each individual experiment (see Chapter 4).

ARPS, when trying to write out a history file, checks for the existence of files of the same name. If such a file already exists, a two digit version number is appended to the name to make it unique. ARPS also has a file compression function built in. When the compression flag is set, *filcmprs* =1, the output files will be compressed using the UNIX compression command **compress** or **gzip**. These files will be automatically uncompressed before they are read in.

The format indicator *fmt* in the files names is one of the following: *bin, asc, hdf, pak, bn2, svi, net, npk,* and *gad*, each representing one of the following data formats:

- *bin* - Unformatted write using the default precision of the computer, *e.g.*, real arrays are written as unformatted 32 bit words on the IBM RISC/6000, and as unformatted 64 bit words on the DEC Alpha. However, on the Cray, IEEE standard 32 bit words are written instead of the Cray-native 64 bit words. This is made possible by the Cray library routines ASNCTL and ASNFILE. The Cray-native binary format data are not completely portable.

- *asc* - Formatted ASCII files. These files are completely portable, but are very large in size. This is the most inefficient format in both storage usage and CPU time consumption.

- *hdf* - Output by calling the Hierarchical Data Format (HDF) library. HDF is pubic domain software, designed for multi-platform portability. HDF was developed and is supported by the National Center for Super-computing Applications (NCSA) at the University of Illinois. NCSA has also developed a set of tools for manipulating and viewing the HDF data sets on various platforms. One of them is the X-window application Ximage, that animates the image files in HDF format. Equivalent software is also available for IBM compatible PCs or Apple Macintoshes. The HDF library source code (mostly in C) can be obtained from the anonymous FTP server *anonymous@ftp.ncsa.uiuc.edu.* Further information on HDF can be obtained by contacting the NCSA at (217) 244-1144. By default, the make script for ARPS, **makearps**, assumes that the HDF library is not available. Linking with the HDF I/O routines and the HDF library can be invoked by supplying an I/O option **-io hdf** while linking, *i.e.*, **makearps -io hdf**.

- *pak* - Written as 16 bit integers after being packed from 32 bit reals. Data in this format are more compact, because of the shorter word length used. However, this format works only on machines that support 32 bit IEEE real and 16 bit integer numbers, *e.g.*, it will not work properly on a Cray or a DEC Alpha.

- *bn2* - A variation of the unformatted binary format. It allows for the writing of a portion of the entire 3-D arrays by skipping certain grid points. The additional control parameters for the format are set by calling subroutine BDMPSKIP and BDMPDOMN before calling the data dump routine DTADUMP.

- *svi* - The MeRAF (Meta Random Access File) data file compatible with Savi3D Version 1.2 and later. Savi3D is a 3-D visualization program developed by the Supercomputer Systems Engineering and Services Company (SSESCO), Minneapolis, Minnesota. Please note that since data at all time levels are written into a single file, the time indicator *nnnnnn* in the file name represents the time at which the data was first written. Again, by default, Savi3D is assumed to be unavailable. To link with the Savi3D dump routines and the library, use option -*io savi*. For example, for ARPS40, do **makearps** -*io savi*. For information on running Savi3D visualization software, see Section 10.7. The contact address for SSESCO is: *SSESCO, 511 11th Ave. S., Box 212, Minneapolis, MN, 55415. Tel. (612) 342-0003, E-mail: info@ ssesco.com.*

- *net* and *npk* - The data are written in NetCDF and packed 16 bit NetCDF format, respectively. NetCDF is the Network Common Data Format developed and supported by Unidata, University Corporation for Atmospheric Research (UCAR). NetCDF has a word length of 32 bits, while the packed NetCDF has half of that length, therefore requires half of the disk storage. Similar to HDF, NetCDF is platform independent, but requires the NetCDF library. The source code can be obtained from *anonymous@ftp.unidata.ucar.edu*. For further information on NetCDF, a user can contact : *UCAR Unidata Program Center, P.O. Box 3000, Boulder, Colorado, USA 80307, (303) 497-8644*, Internet: *support@unidata.ucar.edu*, or World Wide Web (WWW) server at *http://www.unidata.ucar.edu*. By default, **makearps** assumes that NetCDF is not available. Option -*io net* must be included in order to link with the NetCDF I/O routines and the library.

- *gad* - Data files written in unformatted binary to be read by the interactive graphic program GrADS. The GrADS data-writing does not require any external library. The object code of the GrADS for most popular platforms can be obtained from its author Brian Doty, University of Maryland (*doty@cola.umd.edu*, Tel: 310-405-5356). For information on displaying GrADS data set, see Section 10.8.

Finally, the option for the history data format, *hdmpfmt*, is set in the NAMELIST block OUTPUT in file *arps40.input* for ARPS40, and in a similar manner for other post-processing programs. The choices of options for *hdmpfmt* can be found in Chapter 4. The output files will be written into a separate directory if *dirname* specified in the input file is not the current work directory.

A standard history data reader (subroutine DTAREAD) is provided with ARPS. This subroutine is called by post-analysis programs including those for plotting, data conversion and formatted printing to import the history format data. Since the history dump format is also used as the format for 3-D initialization of ARPS (*initopt=2*), DTAREAD is also called by the ARPS initialization routine to initialize the ARPS model arrays. A standard history data dump routine (DTADUMP) is provided with ARPS. DTADUMP handles the output of data in all ARPS-supported formats. An example program, *arpsread.f*, for reading the history format data is provided with the ARPS source code distribution and is reproduced here:

```
      PROGRAM ARPSREAD
c
c##############################################################################
c
c     PURPOSE:
c
c     Sample program to read history data file produced by ARPS 4.0
c
c     Link arpsread using the following command on a IBM RISC/6000
c     system, assuming HDF, NetCDF and Savi3D libraries are not available:
c
c     f77 arpsread.f read3d.o nohdfio3d.o nonetio3d.o nosviio3d.o \
c         gradsio3d.o pakio3d.o outlib3d.o ibmlib3d.o
c
c##############################################################################
c
      include 'dims.inc'
      include 'globcst.inc'
c
      real    x      (nx)       ! The x-coord. of the physical and
                                ! computational grid.
                                ! Defined at u-point.
      real    y      (ny)       ! The y-coord. of the physical and
                                ! computational grid.
                                ! Defined at v-point.
      real    z      (nz)       ! The z-coord. of the computational
                                ! grid. Defined at w-point.
      real    zp     (nx,ny,nz) ! The height of the terrain.
      real    hterain(nx,ny)    ! Terrain height.
      real    j1     (nx,ny,nz) ! Coordinate transformation
                                ! Jacobian -d(zp)/d(x)
      real    j2     (nx,ny,nz) ! Coordinate transformation
                                ! Jacobian -d(zp)/d(y)
      real    j3     (nx,ny,nz) ! Coordinate transformation
                                ! Jacobian  d(zp)/d(z)
      real    uprt   (nx,ny,nz) ! Perturbation u-velocity (m/s)
      real    vprt   (nx,ny,nz) ! Perturbation v-velocity (m/s)
      real    wprt   (nx,ny,nz) ! Perturbation w-velocity (m/s)
      real    ptprt  (nx,ny,nz) ! Perturbation potential temperature (K)
      real    pprt   (nx,ny,nz) ! Perturbation pressure (Pascal)
      real    qvprt  (nx,ny,nz) ! Perturbation water vapor specific
                                ! humidity (kg/kg)
      real    qc     (nx,ny,nz) ! Cloud water mixing ratio (kg/kg)
      real    qr     (nx,ny,nz) ! Rain water mixing ratio (kg/kg)
      real    qi     (nx,ny,nz) ! Cloud ice mixing ratio (kg/kg)
      real    qs     (nx,ny,nz) ! Snow mixing ratio (kg/kg)
```

```
      real    qh     (nx,ny,nz) ! Hail mixing ratio (kg/kg)
      real    km     (nx,ny,nz) ! The turbulent mixing coefficient for
                                ! momentum. ( m**2/s )
      real    ubar   (nx,ny,nz) ! Base state u-velocity (m/s)
      real    vbar   (nx,ny,nz) ! Base state v-velocity (m/s)
      real    wbar   (nx,ny,nz) ! Base state w-velocity (m/s)
      real    ptbar  (nx,ny,nz) ! Base state potential temperature (K)
      real    pbar   (nx,ny,nz) ! Base state pressure (Pascal)
      real    rhobar (nx,ny,nz) ! Base state air density (kg/m**3)
      real    qvbar  (nx,ny,nz) ! Base state water vapor specific
                                ! humidity (kg/kg)
      real    u      (nx,ny,nz) ! Total u-velocity (m/s)
      real    v      (nx,ny,nz) ! Total v-velocity (m/s)
      real    w      (nx,ny,nz) ! Total w-velocity (m/s)
      real    qv     (nx,ny,nz) ! Water vapor mixing ratio (kg/kg)
      real    tsfc   (nx,ny)    ! Temperature at surface (K)
      real    tsoil  (nx,ny)    ! Deep soil temperature (K)
      real    wetsfc (nx,ny)    ! Surface soil moisture
      real    wetdp  (nx,ny)    ! Deep soil moisture
      real    wetcanp(nx,ny)    ! Canopy water amount
      real    tem1(nx,ny,nz)    ! Work arrays
      real    tem2(nx,ny,nz)    ! Work arrays
      real    tem3(nx,ny,nz)    ! Work arrays
c
c#######################################################################
c
c     Misc. internal variables
c
c#######################################################################
c
      integer hinfmt, nchin
      character grdbasfn*80,filename*80
      real time
c
c#######################################################################
c
c     Get the name of the input data set.
c
c#######################################################################
c
      hinfmt = 1  ! Data format set to 1 for unformatted binary

      grdbasfn = 'arps40.bingrdbas'  ! File containing base state and
                                     ! grid arrays
      filename = 'arps40.bin003600'  ! History data file at 3600 s.
c
c#######################################################################
c
c     Call DTAREAD to read in history data.
c
c#######################################################################
c
      CALL dtaread(nx,ny,nz,
     :     hinfmt, nchin,grdbasfn(1:16),lengbf,
     :     filename(1:16),lenfil,time,
     :     x,y,z,zp, uprt ,vprt ,wprt ,ptprt, pprt ,
     :     qvprt, qc, qr, qi, qs, qh, km,
     :     ubar, vbar, wbar, ptbar, pbar, rhobar, qvbar,
     :     tsfc, tsoil, wetsfc, wetdp, wetcanp,
     :     ireturn, tem1,tem2,tem3)

      curtim = time
c
```

```
c     Analysis and plotting code
c
      STOP
      END
```

To read the history data, a user needs to specify the input data format, the file names of the grid and base state data and the time-dependent data. If the grid and base-state arrays are present in the time-dependent data sets, the grid and base state data file will be read just once.

10.2. Graphic Plotting Program - ARPSPLT

A vector graphics plotting program, ARPSPLT, is provided with ARPS. ARPSPLT reads in the ARPS history format, performs various analyses and generates graphic output as 2-D fields and 1-D profiles. ARPSPLT is based on graphic package ZXPLOT, which is developed locally. An introduction to ZXPLOT is given in Chapter 12. To use ARPSPLT, one must install ZXPLOT on a local computer system first. The source code of ZXPLOT is NOT distributed with ARPS, but the object library for most popular platforms can be obtained from its author Ming Xue, CAPS, University of Oklahoma (*mxue@uoknor.edu*) (see also Chapter 12 for the information on the anonymous FTP server for the object code).

The main program of ARPSPLT is contained in file *arpsplt47.f*. It resides in the same directory as the ARPS model code and shares with ARPS a number of include files and I/O modules. The array dimensions, *nx*, *ny* and *nz* are set in the same include file *dims.inc* that is used by ARPS40. The compilation and linking is controlled by the shell script *makearps*.

As explained in Chapter 12, two low-level interfaces for ZXPLOT are available. One calls a few low-level NCAR Graphics routines and produces CGM (computer graphics metafile) output compatible with the NCAR Graphics metafile and the related tools (*e.g.*, **ctrans**). This interface requires that the NCAR Graphics library be available. The other one, the PostScript (trademark of Adobe Systems, Inc.) interface, produces, independent of any other software package, PostScript files that can be viewed using PostScript viewers such as **Ghostview** (GNU free software foundation X-window based software, available via the internet) or printed on a PostScript printer.

Similar to the ARPS main program, the compilation and linking of ARPSPLT is handled by the UNIX shell script *makearps*. For ARPSPLT to produce CGM metafile output (NCAR Graphics required), the command is

makearps -*io io_options -mach machine_type arpspltncar*

and to produce PostScript graphic output, the command is:

makearps -*io io_options -mach machine_type arpspltpost*

where *io_options* are I/O format support options (see Section 3.5) and *machine_type* is one of *cray, rs6000, iris4d* and *sun4*. Both *io_options* and *machine_type* are optional. When the I/O option is not specified, it is assumed that no external data library is available. When the machine type is not given, the shell script tries to figure out the type by itself. Command

makearps -*help*

will list the help pages for **makearps**.

The above compilation and linking step produces executable commands **arpspltncar** or **arpspltpost**, which are then executed by entering:

arpspltncar < *arpsplt.input* > *arpsplt.output*

or

arpspltpost < *arpsplt.input* > *arpsplt.output,*

which generate a CGM metafile (named gmeta) or PostScript graphic output, respectively. The PostScript output is named *ps.outnnn*, where *nnn* is a sequential three digit number to avoid overwriting previous versions if they exist.

The control parameters for the plotting program ARPSPLT are set in the NAMELIST format input file *arpsplt.input*. Presently, ARPSPLT is capable of plotting wind vectors, contours, streamlines (NCAR Graphics version only), and color-filled fields in 2-D cross sections, and 1-D profiles in specified columns. 2-D HDF images (requires HDF library) can also be generated as an option. The input file provides flexible controls for overlaying fields, plotting multiple pictures in a frame, zooming the plotting domain, overlaying terrain and superposition of maps of political borders with the specified projection. When the map plotting option is turned on, a map data file has to be specified and the data available. Contact *arpsuser@uoknor.edu* for a sample map file. The map file is an ASCII table of (lat, lon) line coordinates. A more complete description of these control parameters can be found in the input file itself.

10.3. Data Conversion Program

A history data conversion program, ARPSCVT, is provided with ARPS. ARPSCVT converts history dump data between any two formats supported by ARPS. The main program is in file *arpscvt11.f* and is located in the main ARPS directory. Like most of the other utility programs, ARPSCVT shares with the model a number of include files and data I/O routines. The compilation and linking of this program is again controlled by shell script **makearps**. The command is

makearps -*io io_options* -*mach machine_type arpscvt*

The optional flags *io_options* and *machine_type* are defined in the same way as for graphic plotting program ARPSPLT (see Section 10.2).

To run the program enter

arpscvt < *arpscvt.input* > *arpscvt.output*

where *arpscvt.input* is an input file in NAMELIST format that specifies the name and format of both the input and output data files. The output control parameters are the same as those used by ARPS. Their definitions can be found in chapter 4.

ARPSCVT is usually used to convert ARPS history data set into other format that can be fed into visualization packages like Savi3D.

10.4. Formatted Printing Program for Examining History Data Dumps

Another utility provided with ARPS reads in the history data and produces formatted tables of the given fields on selected slices. It is useful for quantitative examination of data. The program, ARPSPRT, is in file *arpsprt12.f* and is organized in the same way as the plotting and conversion programs. To compile and link, enter

makearps -*io io_options* -*mach machine_type arpsprt*

again the *io_options* and *machine_type* flags are optional.

To run the program, enter

arpsprt < *arpsprt.input* > *arpsprt.output*

where *arpsprt.input* includes the parameters for file specifications and printing controls.

10.5. Other Utility Programs

The capabilities of utility programs reflect a user's needs. A user can write his/her own programs for particular purposes and contribute to the reservoir of programs built around ARPS. The CAPS model development group will coordinate such efforts and selectively support certain utilities. At this time, ARPS supports, in addition to the utilities described in the previous sections, a number of programs that are commonly used by its users. They include: ARPSPLTMAX, a program that plots the time series of the maximum and minimum of model variables, ARPSR2H, a program that converts a restart file into history dump format and at the same time performs interpolation from the original grid to a new grid if necessary, ARPSDIFF, a program that prints out the difference between two sets of history data, and ARPSEXTSND, a program that extracts a column of data from a history data set and writes them as a sounding file.

The compilation and linking of these programs are also controlled by the make script, and the commands are, respectively:

makearps *arpspltmax*
makearps *arpsr2h*
makearps *arpsdiff*
makearps *arpsextsnd*

Similar to the make commands for other programs, optional flags can be included to indicate the existence of libraries and system configurations. The commands to execute them are:

arpspltmax < *arpspltmax.input* > *arpspltmax.output*
arpsr2h < *arpsr2h.input* > *arpsr2h.output*
arpsdiff < *arpsdiff.input* > *arpsdiff.output*
arpsextsnd < *arpsextsnd.input* > *arpsextsnd.output*

where the input files specify the control parameters and the data files to be processed.

10.6. ARPSTOOLS

To provide a mechanism for coordinating user supplied analysis tools that are useful for the ARPS users, a directory known as ARPStools is maintained. The programs found in this directory are not as well supported as those found in the main ARPS directory, and the documentation may not be as intensive. The original author is usually responsible for maintaining and upgrading these programs. One of the most used program is ARPSSKEWT, a program that plots a Skew-T diagram by reading in a sounding file generated by an ARPS run (usually named *runname.sound*).

The ARPStools programs can be obtained from *anonymous@ftpcaps.uoknor.edu,* in directory *pub/ARPStools*. If you'd like to contribute to this collection, contact *arpsuser@uoknor.edu*.

10.6.1 ARPSSKEWT

ARPSSKEWT is one of the ARPStools programs that are available from CAPS. ARPSSKEWT, developed by Dr. Richard Carpenter of the Center for Computational Geosciences at the University of Oklahoma, takes the sounding file and creates a skew-*T* plot. A hodograph can also be produced optionally to show the shear profile of the environment. This program can be obtained from the CAPS anonymous FTP server *ftpcaps.uoknor.edu*. A UNIX tape archive (tar) file, *ARPSskewt.tar*, can be found in directory *pub/ARPStools*. Please note that ZXPLOT graphic package and the NCAR Graphics library are required to use this program. See Chapter 12 for information on ZXPLOT.

Assuming the user has ZXPLOT and NCAR Graphics properly setup and is now inside the ARPSSKEWT directory, the command to compile and link ARPSSKEWT is:

make *-f Make-ARPSskewt,*

where *Make-ARPSskewt* is a make description file found in the ARPSSKEWT directory.

The **make** command creates the executable *arpsskewt.*

An input file containing several parameters is required to run the program. An example input file, *ARPSskewt.sample.input*, is provided with the distribution, and the input file is in the NAMELIST format. The parameters in this example input files are:

```
&ARPS_SKEWT_INPUT
    file            =  'ARPS.sound.example',
    do_hodo         =  .true.,      [ Default is .false.]
```

```
    plot_sfc_parcel    =  .true.,      [ Default is .false.]
    print_info         =  .true.,      [ Default is .false.]
&END
```

where *ARPS.sound.example* is a sample sounding file. A file named *runnam.sound* is produced by every ARPS model run, and is in a format for ARPSSKEWT to read. Parameter *do_hodo* is an option switch for hodograph plotting, *plot_sfc_parcel* is a parameter that controls the plotting of a parcel trajectory lifted from the surface, and *print_info* the one that controls the printing of certain diagnostic information on the plot. Several other NAMELIST parameters can be set by the user in the input file. They are listed in the following, together with their default values:

helcont	= *.false.*	Plot helicity contours on hodograph.
hodo_denmwind	= *.false.*	Compute and plot density-weighted mean wind on hodograph.
plot_sfc_parcel	= *.false.*	Plot parcel ascending from surface on the skew-T diagram.
plot_tv	= *.false.*	Also plot virtual temperature.
caps_use_t	= *.true.*	Compute CAPE using temperature instead of virtual temperature (Note: virtual temperature is calculated including condensate loading of lifted parcels).
caps_use_irrev	= *.true.*	Compute CAPE for pseudo-adiabatic ascent (Note: pseudo-adiabatic assumes that all the water condensed in the parcel will fall out. *caps_use_t = .true.* and *caps_use_irrev = .true*. is the standard method for calculating CAPE.)
print_info	= *.false.*	Force printing of certain info. on plot
verbose	= *.false.*	Print extra diagnostic info.

After the input file is prepared, enter command

arpsskewt < *ARPSskewt.sample.input*

to execute the program.

The primary output of ARPSSKEWT is a graphic metafile (usually called *gmeta*). It contains the skew-T diagram and hodograph (optional) of the given sounding. Two other files are created at the same time, they are: *skt.1*, which contains a list of thermodynamic information and wind values as retrieved from ARPS sounding file, and, *skt.2*, which contains further information (virtual temperature, *etc*.) if the parameter *verbose='true'* is chosen.

10.7. Savi3D

Savi3D, a commercial 3-D visualization package, is available on the CAPS IBM RISC cluster and is supported by ARPS as one of the methods of visualizing 3-D model output. The package can perform wind vector and contour plotting, 3-D rendering as well as animation. ARPS, as well as the data conversion program ARPSCVT, can generate Savi3D compatible data sets that can be used by Savi3D. The command to invoke the package on the CAPS IBM cluster is

savi3D

A user then uses the window controls to load in the data set and perform all the interactive operations.

Savi3D is developed by Supercomputer Systems Engineering and Services Company (SSESCO). For further product information on Savi3D, please contact:

SSESCO
511 Eleventh Avenue South
Minneapolis, MN 55415
Tel: (621) 342-0003, FAX: (612) 344-1716
E-mail: info@ssesco.com

10.8. GrADS

GrADS data format is also supported by CAPS on the CAPS RISC workstation cluster. GrADS is an X-window based interactive data analysis program from the University of Maryland (See Section 10.1 for contact information). GrADS data can be generated by ARPS directly or by the data conversion program ARPSCVT. To invoke GrADS, enter:

grads

then type **open** *runname.gradscntl* at the prompt of the GrADS, where *runname.gradscntl* is a configuration file generated at the time the GrADS data set in produced. Other commands to manipulate the graphic display are then entered at the prompts of the GrADS window (by using the command line interface).

A GrADS script file is also available that provides a menu-based interactively interface for GrADS. To run the script, enter:

grads -*lc "run arpsgrads.gs" &*

where *arpsgrads.gs* is the name of the GrADS script file that is distributed with ARPS. Before running the script, you have to create a temporary file,

arpsgrads.tmp

which contains a single line with the data control file name (usually *runname.gradscntl* created at the same time the GrADS data set is written).