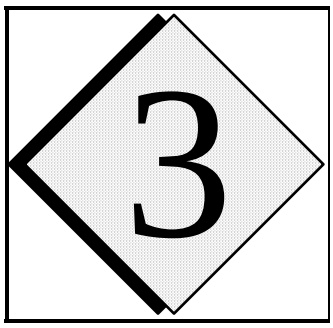

# Getting Started

This chapter provides you with the information on the essential steps necessary to access, compile and run the ARPS model and to examine the model output. Details regarding the model's theoretical and numerical formulation, definition of model parameters, *etc*. may be found elsewhere in this User's Guide.

## 3.1. Assumptions

We assume that the users of this document have a working knowledge of:

- UNIX operating system,
- FORTRAN-90, and
- file transfer protocol (FTP).

## 3.2. Downloading the ARPS On-line

The ARPS model is accessible via direct download on-line. Go to *http://caps.ou.edu/arps.htm* and follow the detailed instructions as provided. (The ability to download the code is contingent upon the agreement to certain terms of use that are related, in particular, to commercial application. In general, there are no restrictions for the use of the ARPS for purely academic research purposes).

Having retrieved the ARPS compressed datafile *arps5.xxxx.tar.gz* (where '*xxxx*' defines a specific version of the code), place it in an appropriate

directory on your local system. To decompress and expand the code, type the following:

| | |
|---|---|
| **gunzip** *arps5.xxxx.tar.gz* | [the original file was compressed using the GNU Free Software Foundation **gzip**, and is expanded using **gunzip**, the source code for which can be obtained from many anonymous FTP sites on the internet] |
| **tar** –xvf *arps5.xxxx.tar* | [split the tar file into separate files that are placed into a subdirectory called *arps5.xxxx* ] |
| **cd** *arps5.xxxx* | [change directory to *arps5.xxxx*] |
| **ls** | [to view the contents of the main ARPS directory] |

You will see subdirectories inside the ARPS root directory. The source code for all ARPS modules (except for include files) are located in the directory *src* (see next section). If you wish to locate a file that contains a specific Fortran subroutine (or a specific string), type the following inside the *src* directory:

**grep** *-i "subroutine name" */*.f90*

If you encounter problems with the above procedure, send an e-mail message to *arpssupport@ou.edu.*

## 3.3. Examining the Files

The ARPS main directory consists of a set of subdirectories, which are organized according to various types of datasets:

| | |
|---|---|
| *bin* | the subdirectory in which the executable is placed upon successful compilation of a given ARPS program (arps, adas, arpstrn, arpssfc, ext2arps, etc.). |
| *data* | contains a number of static data files used by the ARPS system. Large data files, including those of terrain and surface characteristics have to be downloaded separately as explained in section 3.7.1. |

*docs* documentation files pertaining to the ARPS and associated utilities. In particular, *docs/*.tree* contains calling trees of select programs that can be very helpful for understanding the structure of these programs and *docs/ARPSQuickGuide_v#.doc* provides a quick guide for using various programs in the ARPS package.

*include* contains include files used by various programs.

*input* contains the input files that provide the control parameters of ARPS and associated utility programs.

*lib* the subdirectory in which several libraries of shared subroutines and functions are placed. The libraries will be built automatically by *makearps* when needed.

*scripts* contains several script files (mainly in Perl) that allow for automated runs of ARPS for select test cases.

*sounding* contains **.snd* files which represent the 1D vertical environmental profile for select study cases.

*src* contains the source code (mainly FORTRAN-90) by subdirectories for each of the programs that are included in the full ARPS package. [The ARPS main driver program *arps.f90* is located in *src/arps*].

Other files in the ARPS root directory include:

*BUGS* lists known bugs in the current version.

*HISTORY* a history/log of modifications to the ARPS system of various versions.

*LICENSE* a copy of the license agreement that specify the terms under which the ARPS code may be used or distributed.

*TERMS_of_USE* reiteration of the terms by which a user may obtain access to and use the ARPS code.

*makearps* an UNIX C-shell script for compiling ARPS and other utility programs on common Unix platforms, the specific type of which is automatically detected.

*Makefile* a make description file used by makearps.

*MANIFESTS* a list by subdirectory of all the files contained in the ARPS root directory.

*README* provides basic information on how to compile and run the ARPS and its associated utilities.

*UPDATE* modifications to the ARPS prior to version 5.0.

## 3.4. List of ARPS Utilities

In addition to the atmospheric prediction model, the ARPS consists of a data analysis system, ADAS, as well as various pre- and post-processing utilities that allow for observational data preparation, forecast initialization, plotting and interpretation of forecast output, and forecast verification. Certain modules also assist in the generation of forecast ensembles and that allow for the execution of the ARPS using multiple processors on either shared- or distributed-memory machines.

The listing below is a general overview of the different utilities. Detail regarding the control parameters and range of capabilities of each utility is provided in subsequent chapters. A Quick Guide to the modules is given in Appendix E, which presents basic information on the location of the source code, compilation procedures, MPI issues, and dependencies on external libraries, if any.

**Atmospheric Prediction Model**

*arps* The ARPS model is the core module of the system, which numerically predicts a future state of the atmosphere from a given initial state. The initial conditions may be specified analytically or using a single vertical sounding (which defines the horizontally homogenous environment) plus initial perturbations, by interpolated values from an external forecast model, or by a 3D analysis of current observations.

*arps_mpi* The ARPS executable that is appropriate for use on a distributed memory machine using Message Passing Interface (MPI).

*splitfiles* Splits the dataset that defines the initial and boundary conditions for use by ARPS MPI job, i.e., by *arps_mpi*. Each

data file will be split into *n* files, where *n* is the number of processors to be used by the MPI job.

*joinfiles* Joins the collection of data files written by individual processors by the ARPS MPI run.

Note that the *splitfile* and *joinfiles* steps are optional depending on the option settings in ARPS input file. *arps_mpi* can read and write data files for the entire model grid.

*arpsagr* ARPS with the capability of invoking adaptive grid refinement. [Note: available only in *arps4.5.1* and earlier versions].

**Data Analysis**

*adas* The ARPS Data Analysis System (ADAS) generates a 3D gridded analysis of the current atmosphere using a model background field and up-to-date observations from a myriad of *in situ* and remote sensing sources including NEXRAD radars, wind profilers, satellites, surface observation networks, and aircraft. ADAS datasets may be used to provide the initial conditions for ARPS simulations or forecasts. The analyses can also used to provide boundary conditions for simulation runs.

*arpsassim* Retrieve temperature and pressure fields given two time levels of 3D wind field.

*mci2arps* Remaps satellite data to an ARPS grid as appropriate for ingest by ADAS.

*88d2arps* Remaps NEXRAD Level-II radar data into a format appropriate for ingest by ADAS.

*nids2arps* Remaps NEXRAD Level-III (NIDS) radar data into a format appropriate for ingest by ADAS.

**Pre-processing**

*arpstern* Creates the terrain data files for the ARPS domain as specified by the user (an older program).

*arpstrn* Creates the terrain data files for the ARPS domain as specified by the user (a newer program that allows for the use of terrain data down to a spatial resolution of 3 seconds over the U.S.).

*arpssfc* Prepares the soil type, vegetation type, and leaf area index data files for use by the ARPS model.

*arpssoil* Creates initial soil moisture and soil temperature fields for ARPS soil model.

*ext2arps* Extracts and interpolates pertinent fields from a National Weather Service model forecast dataset to an ARPS grid to provide an ADAS analysis background or initial conditions and boundary conditions for an ARPS forecast.

*arpsintrp* Interpolates ARPS history format gridded data to an ARPS grid of different resolution and writes the output in ARPS history and/or boundary condition format. The output can be used for a nested ARPS run, as well as the background for an analysis on the output grid.

*arpstintrp* Time interpolates between two ARPS history format data files.

*mergetrn* Generates a smooth transition in the terrain field at the boundary between the parent (usually an external model that that provides boundary conditions) and child forecast domains that are of different spatial resolutions. The modules *ext2arps* and *arpsintrp* now have on-the-fly terrain merging capability so this step may not be necessary.

**Post-processing**

*arpscvt* Converts an ARPS dataset between various formats (binary, ascii, HDF, NetCDF, GRIB) including formats required by certain visualization packages (GrADS)

*arpscvtobs* Converts an LSO-formatted observational data file to an HDF file

*arpsdiff* Reads in two ARPS history format datasets of same or different resolutions, calculates the difference between fields and writes the difference fields in ARPS history format on one of the selected grids.

*arpsextsnd* Extracts a vertical column of data from an ARPS history file and generates a sounding file.

*arpspltncar* Plots data values from an ARPS history file as well as selected diagnosed variables (such as shear, helicity, relative humidity, CAPE) as contour, vector and color-filled maps in horizontal or vertical 2D slices across the dataset domain as specified by the user. The output is provided in a format consistent with NCAR graphics. *arpspltncar can also plot vertical profiles at user-specified columns.*

*arpspltpost* Capabilities are identical to *arpspltncar* but the output is given in a PostScript format.

*arpsprt* Reads an ARPS history file and prints arrays of field values for a specified 2D slice.

*arpsverif* Verifies an ARPS forecast through comparison with point observations or gridded fields (such as provided by an analysis or an alternate model forecast)

*pltgrid* Plots a grid map along with any grid boxes for an ARPS run with multiple domains. Useful as a tool for configuring the ARPS domain.

**Ensemble Forecasts**

*arpsensbc* generates external boundary files for a suite of ensemble members

*arpsenscv* plots variables associated with an ensemble

*arpsensic* generates a set of perturbed initial conditions that identify a suite of ensemble members

## 3.5. Process Flowchart of ARPS Model System

To run a complete model system, a number of steps are involved. The preprocessing step prepares the terrain, surface characteristics data sets, and the objective analysis for model initialization. After the model time integration, post-analysis is performed to examine the model output. We illustrate the interconnection of these processes using a flow chart, as given in Figure 3.1.

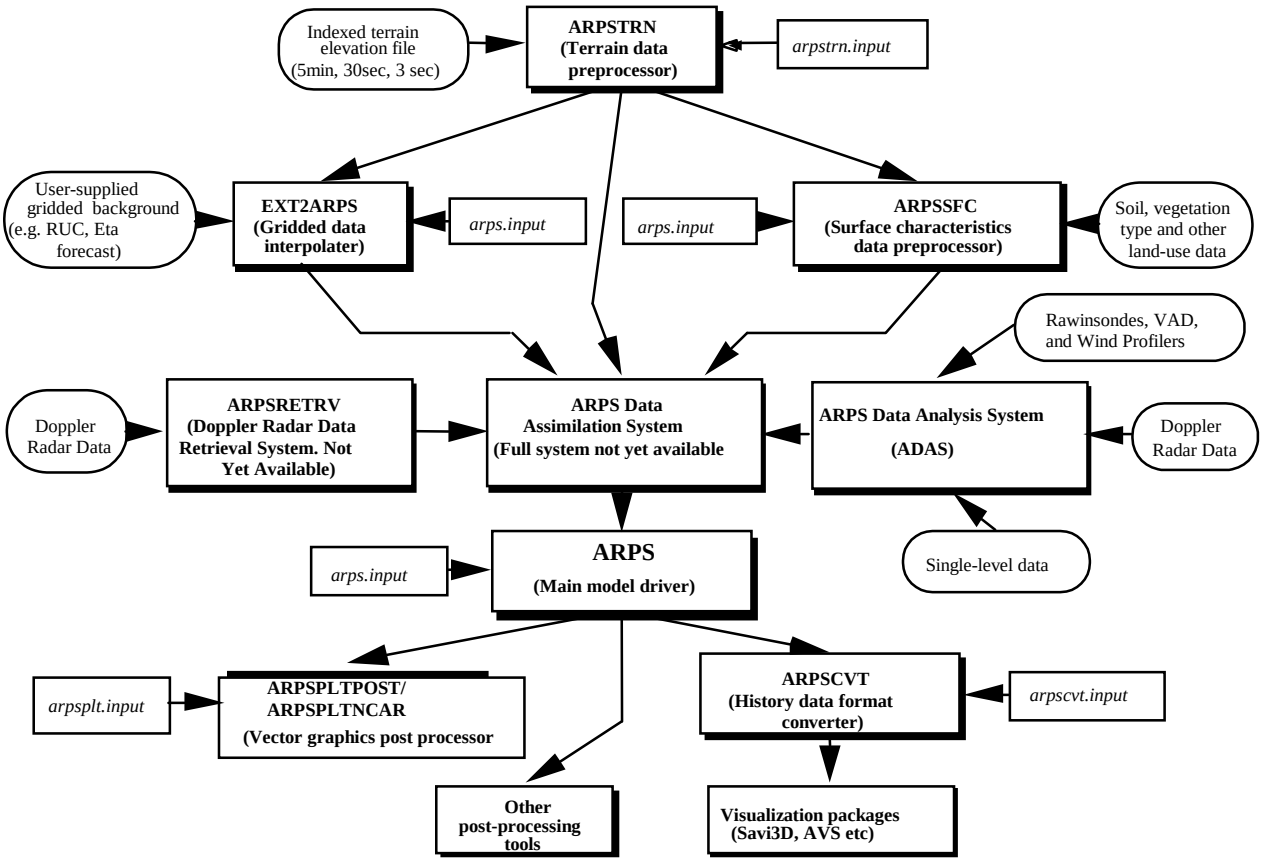


Figure 3.1. ARPS model process flow chart.

## 3.6. Compiling ARPS and Supporting Programs

The compilation and linking of ARPS and all other ARPS-supported utility programs are orchestrated by a UNIX shell script, **makearps**. The script invokes a **make** command, which is a UNIX utility for managing code compilation based on object dependencies. The **make** utility compares the modification time of source files to that of object codes, and recompiles only those files that have been updated. The **make** command for ARPS uses a description file, *Makefile.*

The executable that is generated using **makearps** is placed in the *bin* subdirectory inside the ARPS root directory.

A help page will be listed when command **makearps** is entered without any arguments and provides information on the various parameters:

Usage: **makearps** [options] *cmd*

*cmd* is one of the following:

(Note names in *italic*, such as *arps*, are *cmd* parameters passed to makearps script, those in **bold**, such as **arps**, are the names of the executables produced by '**makearps** *cmd*', and those in capital letters, such as ARPS, refer to program names)

Executable commands:

| | |
|---|---|
| *all* | Make all executables except parallel and plotting programs |
| *adas* | Produce ADAS executable **adas** |
| *arps* | Produce ARPS executable **arps** |
| *arps_mpi* | Produce MP version ARPS executable **arps** (using MPI) |
| *arpsagr* | Produce executable **arpsagr** (available only in *arps4.5.1* and older versions) |
| *arpsassim* | Produce executable **arpsassim** |
| *arpscvt* | Produce executable **arpscvt** |
| *arpscvtobs* | Converts LSO surface data to HDF format files. |
| *arps2gem* | Convert ARPS history format data to Gempak format |
| *arps2ncdf* | Convert ARPS history format data to NetCDF format |
| *arpsdiff* | Produce executable **arpsdiff** for ARPSDIFF |
| *arpsenscv* | Generate 2-D plots for ensemble variables |
| *arpsextsnd* | Produce executable **arpsextsnd** for ARPSEXTSND |
| *arpsintrp* | Produce executable **arpsintrp** |
| *arpstintrp* | Produce executable **arpstintrp** |
| *arpspltmax* | Produce executable **arpspltmax** for ARPSPLTMAX |
| *arpspltncar* | Produce NCAR Graphics executable **arpspltncar** for ARPSPLT |
| *arpspltpost* | Produce PostScript executable **arpspltpost** for ARPSPLT |
| *arpspltpost_mpi* | Produce Message passing executable **arpspltpost** |
| *arpspltncar_mpi* | Produce Message passing executable **arpspltncar** |
| *arpsprt* | Produce executable **arpsprt** for ARPSPRT |
| *arpsensic* | Generate perturbation IC for ensemble fcst |
| *arpsensbc* | Generate perturbation BC for ensemble fcst |
| *arpsread* | Produce executable **arpsread**, a sample program |
| *arpssfc* | Produce executable **arpssfc** for ARPSSFC |
| *arpssoil* | Produce executable **arpssoil** for ARPSSOIL |
| *arpstern* | Produce executable **arpstern** for ARPSTERN |
| *arpstrn* | Produce executable **arpstrn** for ARPSTRN |
| *arpsverif* | Produce executable **arpsverif** for ARPSVERIF |
| *dir1deg* | Produce executable **dir1deg** for DIR1DEG |

| | |
|---|---|
| *dir5min* | Produce executable **dir5min** for DIR5MIN |
| *dir30sec* | Produce executable **dir30sec** for DIR30SEC |
| *mergetrn* | Produce executable **mergetrn** for MERGETRN |
| *pltgrid* | Produce executable **pltgrid** to plot ARPS grids |
| *splitfiles* | Produce executable **splitfiles** |
| *splitfiles_mpi* | Produce MPI version **splitfiles** executable |
| *splithdf* | Produce executable **splithdf** |
| *splithdf_mpi* | Produce MPI version **splithdf** executable |
| *joinfile* | Produce executable **joinfile** |
| *joinfiles* | Produce executable **joinfiles** |
| *joinbin2hdf* | Produce executable **joinbin2hdf** |
| *joinhdf* | Produce executable **joinhdf** |

Data conversion and display:

| | |
|---|---|
| *88d2arps* | Produce executable **88d2arps** |
| *ext2arps* | Produce executable **ext2arps** for user's external data |
| *ext2arps.laps* | Produce executable **ext2arps** for LAPS data |
| *ext2arps.gempak* | Produce executable **ext2arps** for GEMPAK data |
| *mci2arps* | Produce executable **mci2arps** |
| *nids2arps* | Produce executable **nids2arps** |
| *pltradcol* | Produce executable **pltradco**l |
| *pltsatfld* | Produce executable **pltsatfld** |
| *wtretcol* | Produce executable **wtretcol** |

Object library archive:

| | |
|---|---|
| *libarps* | Compile libarps.a |
| *libadasr* | Compile libadas.a |
| *arpssolver* | Produce an object library for ARPS solver |

Create tape archive (tar) packages:

| | |
|---|---|
| *Allfiles.tar* | Create tar files of the entire ARPS systems except for the documentation |
| *arps.tar* | Create tar files associated with ARPS |
| *adas.tar* | Create tar files associated with ADAS |
| *arps_mp.tar* | Create tar files associated with message passing |
| *arpsagr.tar* | Create tar files associated with ARPS Adaptive Grid Refinement |
| *arpsassim.tar* | Create tar files associated with ARPSASSIM |
| *arpscvt.tar* | Create tar files associated with ARPSCVT |
| *arpscvtobs.tar* | Create tar files associated with ARPSCVTOBS |
| *arpsdiff.tar* | Create tar files associated with ARPSDIFF |
| *arpsens.tar* | Create tar files associated with ARPSENS |
| *arpsextsnd.tar* | Create tar files associated with ARPSEXTSND |

| | |
|---|---|
| *arpsintrp.tar* | Create tar files associated with ARPSINTRP |
| *arpsplt.tar* | Create tar files associated with ARPSPLT |
| *arpsprt.tar* | Create tar files associated with ARPSPRT |
| *arpssfc.tar* | Create tar files associated with ARPSSFC |
| *arpssoil.tar* | Create tar files associated with ARPSSOIL |
| *arpstern.tar* | Create tar files associated with ARPSTERN |
| *arpstrn.tar* | Create tar files associated with ARPSTRN |
| *arpsverif.tar* | Create tar files associated with ARPSVERIF |
| *mergetrn.tar* | Create tar files associated with MERGETRN |
| | |
| *88d2arps.tar* | Create tar files associated with radar data remap |
| *ext2arps.tar* | Create tar files associated with EXT2ARPS |
| *mci2arps.tar* | Create tar files associated with satellite data remap |

Clean-up object files:

| | |
|---|---|
| *clean* | Delete all object files |
| *clean.exe* | Delete all executables |
| *clean.tar* | Delete all tar files |
| *clean.adas* | Delete all object files related to ADAS |
| *clean.arps* | Delete all object files related to ARPS |
| *clean.arps_mp* | Delete all object files and temporary source files for the message passing version of ARPS |
| *clean.arpsgar* | Delete all object files related to ARPSAGR |
| *clean.arpsassim* | Delete all object files related to ARPSASSIM |
| *clean.arpscvt* | Delete all object files related to ARPSCVT |
| *clean.arpscvtobs* | Delete all object files related to ARPSCVTOBS |
| *clean.arpsdiff* | Delete all object files related to ARPSDIFF |
| *clean.arpsextsnd* | Delete all object files related to ARPSEXTSND |
| *clean.arpsintrp* | Delete all object files related to ARPSINTRP |
| *clean.arpsplt* | Delete all object files related to ARPSPLT |
| *clean.arpsprt* | Delete all object files related to ARPSPRT |
| *clean.arpssfc* | Delete all object files related to ARPSSFC |
| *clean.arpssoil* | Delete all object files related to ARPSSOIL |
| *clean.arpstern* | Delete all object files related to ARPSTERN |
| *clean.arpstrn* | Delete all object files related to ARPSTRN |
| *clean.arpsverif* | Delete all object files related to ARPSVERIF |
| *clean.dirtern* | Delete *dir1deg.o, dir30sec.o, dir5min.o* |
| *clean.mergetrn* | Delete all object files related to MERGETRN |
| | |
| *clean.join* | Delete all object files joinfile |
| *clean.joins* | Delete all object files joinfiles |
| *clean.split* | Delete all object files splitfiles |
| *clean.splithdf* | Delete all object files splithdf |
| *clean.joinbin2hdf* | Delete all object files joinbin2hdf |
| *clean.joinhdf* | Delete all object files joinhdf |

| | |
|---|---|
| *clean.mptrans* | Delete all object files mptrans |
| *clean.88d2arps* | Delete all object files related to radar remap |
| *clean.mci2arps* | Delete all object files related to mci2arps |
| *clean.ext2arps* | Delete all object files related to ext2arps |

Directory option:

| | |
|---|---|
| *-re root_dir* | Specify *root_dir* as ARPS root directory |

Compiler options (machine dependent):

| | |
|---|---|
| *-opt n* | n = 0, 1, 2, or 3 for different optimization levels |
| *-d* | Turn on debug option, reset optimization level to 0 |
| *-p* | Automatic shared-memory parallelization (only available when the compiler supports automatic parallelization) |
| *-omp* | Shared-memory parallelization using explicit Open MP directives (only ADAS contains limited support for Open MP) |
| *-s* | Include compile options for generate performance statistics data at the end of run |
| *-c* | Communication platform |
| *-inline* | Inline small, frequently called routines |
| *-f fort_cmp* | Set the Fortran compiler (e.g., *-f pgf90*) |
| *-f90* | Use f90 compiler (default) |
| *-abi abi_opt* | Use –"abi_opt" (64, n32, o32) default compile option on SGI |
| *-l lib* | Use "-l lib" when linking |
| *-L linkdir* | Use "-L linkdir" when linking |
| *-x[iMKW]* | Specify the minimum set of processor extensions required. It applies for ifc on Intel processors (P, PII, PIII, and P4) |
| *-nomainopt* | Specify to compile several main programs, i.e., *adas*, *arps*, *ext2arps*, and *arpsplt*, without optimization. Certain compiles take too long and/or require too much memory when optimizing the main program while the optimization of the main programs often does not help much. Use this only when you experience problem compiling the main programs. |

.

History data (I/O) library support:

*-io opt1 [opt2 ...]* where options are:

| | |
|---|---|
| *nohdf* | No HDF support (HDF calls will be bypassed) |
| *hdf* | HDF (default) |
| *savi* | Savi3D |
| *v5d* | Vis5D |
| *net* | netCDF |
| *all* | all of the above |

NCAR graphics plotting option:

| | |
|---|---|
| *-ncarg* | Include graphic plotting using NCAR graphics for programs ARPSSFC and ARPSTERN. Default = no graphic plotting. |

ZXPLOT graphics plotting option for ARPSTERN:

| | |
|---|---|
| *-zxpost* | Use ZXPLOT to produce Postscript output |
| -zxncar | Use ZXPLOT to produce ncar graphics output. Default = no graphic plotting. |

Options passed to **make**:

| | |
|---|---|
| *-n* | Displays commands, but does not run them. |
| *-j nproc* | Execute make in parallel (only works on certain platforms). |
| *-p* | Execute make in parallel (on SGI setenv PARALLEL to # processors). |

Miscellaneous options:

| | |
|---|---|
| *-m mach* | Force machine type to *mach* (otherwise use tcsh HOSTTYPE to get the machine type). Recognized machine types include:<br><br>*t3e, cray, hi-ux, iris4d, linux, rs6000, sun4* and *alpha* (for Cray T3E, Cray, HP Unix, SGI IRIX, Linux, IBM AIX, Sun Solaris, HP/COMPAQ Tru64 Unix, respectively).<br><br>If none of the above matches, default type *unix* is used. |
| *-help* | Print this help page. |

The script uses the *csh* or *tcsh* environment variable HOSTTYPE to determine the machine that the code is compiled on. The main differences between the machines are the compiler and loader flags that are to be used.

## 3.7. A Quick Start with ARPS

When ARPS is used for idealized simulation studies, often real terrain data or 3-D analysis data are not required. In this case, fewer steps are needed to complete a model run as given below. It is assumed that the reference directory on your system is the ARPS root directory. The UNIX vi editor is used for illustrative purposes. The script in [] are comments.

**cp** *sounding/may20.snd* [copy example sounding file *may20.snd* into the current directory]
**vi** *may20.snd* [Prepare sounding if required]
**vi** *input/arps.input* [Set control parameters for the model run]
**makearps** *arps* [compile and link ARPS]
*bin*/**arps** < *input/arps.input* > *arps.output* & [Execute the model]

**vi** *input/arpsplt.input* [Set plotting control parameters including the names of history files to be processed]
**makearps** *arpspltncar* [Compile and link ARPSPLT with ZXPLOT and NCAR Graphics libraries]
*bin/***arpspltncar** < *input/arpsplt.input* & [Execute ARPSPLT program to generate graphic metafile output]
**idt** *gmeta* [Examine the graphic metefile using NCAR Graphics viewing tool]

(It is assumed that ZXPLOT and NCAR Graphics have been installed on your computer system. If the later is not installed, use *arpspltpost*.).

## 3.8. A Complete Run of ARPS

The steps required to make a complete ARPS run using real terrain, real surface characteristics, and a 3D analysis based on observational data are listed below.

### Set up terrain dataset

There are two programs available for the generation of the terrain dataset for a specified domain. The first is the original program **arpstern**, which has accompanied earlier versions of the ARPS code. It uses a Barnes analysis scheme to analyze the terrain data at the user-specified resolution. By setting various parameters, much flexibility is left to the user as to how the analysis is conducted. (Specifics on how to setup **arpstern** are given in a later section). The required terrain database is available for download on the CAPS anonymous ftp server. The terrain data for this method are available at a spatial resolution of either 5 minute (global) or 30 second (U.S. only).

The second method, **arpstrn**, uses a bi-linear or a bi-quadratic interpolation scheme to prepare the terrain dataset for a given ARPS domain. Global 5 minute and 30 second datasets plus a 3 second dataset covering continental US and Alaska are available for use for this program. The 30 second data are organized in files for 1 degree x 1 degree patches and the files necessary for your domain can be automatically downloaded from the CAPS ftp server when the server name, login (ftp) and password (your e-mail address) are setup in file *.netrc* located in your home directory (further details are given later). The necessary 3 second data patches can be automatically downloaded from a USGS ftp server. You can also pre-download the necessary data files on your own disk and **arpstrn** will look at this location first before trying to remotely access the terrain datasets via ftp.

As to which program you should use, in general, if you prefer better control on the smoothness of your terrain field (via Barnes analysis) and the supported data sets are good enough for you, you may want to use **arpstern**. If linear or bilinear interpolation plus selected passes of smoothing satisfy you, **arpstrn** would be your choice. If you need 30 second resolution outside continental US or resolution higher than 30 second over US, then **arpstrn** is your only choice. Chapter 4 has more information on these programs.

#### ***Option 1****: Set up terrain dataset using* **arpstern**

The terrain data base is available at the CAPS ftp site. Retrieve the terrain data using *anonymous* ftp. It is assumed that you are in the ARPS root directory. (Preparation of terrain file is not necessary if the analytic terrain option for ARPS is chosen):

| | |
|---|---|
| **cd** data | [move to the *data* subdirectory] |
| **ftp** caps.ou.edu | [connect to the CAPS ftp site (using **anonymous** as your login and your e-mail address as the password] |

**cd** *pub/ARPS/ARPS.data/arpstern.data*
[go to the terrain subdirectory]
**get** *dma_elev.dat.Z* [retrieve the terrain data base]
**get** *dma.dat.Z*
**uncompress** *dma_evel.dat.Z* [uncompress terrain datafiles]
**uncompress** *elev.dat.Z*

Steps to prepare the terrain data, assuming the terrain data base has been properly set up, are as follows:

**cd** .. [Go back to ARPS root directory]
**vi** *input/arpstern.input* [Set parameters for model grid configuration and terrain data analysis]
**makearps** *dir1deg* [Compile and link program DIR1DEG]
**makearps** *dir5min* [Compile and link program DIR5MIN]
**makearps** *dir30sec* [Compile and link program DIR30SEC]

*bin/***dir1deg** < *input/arpstern.input*> *dir1deg.out* &
[Convert 1 degree terrain data to direct access data. This needs to be done only once if the output data file *dir1deg.dat* is saved]
*bin/***dir5min** < *input/arpstern.input*> *dir5min.out* &
[Convert 5 minute terrain data to direct access data. This needs to be done only once if the output data file *dir5min.dat* is saved]
*bin/***dir30sec** < *input/arpstern.input*> *dir30sec.out* &
[Convert 30 second terrain data to direct access data. This needs to be done only once if the output data file *dir30sec.dat* is saved]
**makearps** *-ncarg arpstern*
[Compile and link terrain analysis program ARPSTERN]
*bin*/**arpstern** < *input*/*arpstern.input* > *arpstern.output* &
[Execute program ARPSTERN]
**idt** *gmeta* [Examine the graphic plotting of the terrain field]

Terrain data file *arpstern.dat* is placed in the ARPS root directory.

***Option 2****: Set up terrain data set using* **arpstrn**

The global 5 min terrain data is available on the CAPS ftp server. The commands are:

**cd** data [move to the *data* subdirectory]

**ftp** caps.ou.edu [connect to the CAPS anonymous ftp server]
**cd** *pub/ARPS/ARPS.data/arpstern.data*
[go to the terrain subdirectory]
**get** tbase_global_5min.data.gz [retrieve the terrain data base]
**gunzip** tbase_global_5min.data.gz [uncompress terrain datafiles]

If using 30 second or finer resolution data, **arpstrn** will automatically download the necessary data files from a remote ftp server. Certain permissions, however, must be specified by your system to allow for the remote transfer of these files. (For details, reference the instructions in the *arpstrn.input* file). To allow for automated connection to the remote server(s), create a *.netrc* file in your home directory that contains the lines:

machine edcftp.cr.usgs.gov login anonymous password [your email address]
machine ftp.caps.ou.edu login anonymous password [your email address]

The *.netrc* file can only be readable by yourself. Do **chmod go-rxw *.cshrc*** to remove read/write/execute permission for everyone else.

The 30 second data is located at ftp://caps.ou.edu/ pub/ARPS/ARPS.data/arpstopo30.data and you can choose to pre-download the files in your own local directory.

Assuming that you are in the ARPS root directory, perform the following:

**makearps** *–ncarg arpstrn* [Compile and link program **arpstrn**. The *-ncarg* argument allows for the generation of a *gmeta* file that shows the resultant terrain upon execution of the program]
**vi** input/arpstrn.input [Set parameters for program **arpstrn**. (Be sure to set the full path for the existing directory in which the terrain datasets from the remote server will be placed (dir_trndata) as well as the path for the USGS dataset, *../src/arpstrn/usgs_dem.index*)]

*bin*/**arpstrn** < *input/arpstrn.input* > *arpstrn.out &*

A **.trndata* terrain file is produced in the main directory and named with a prefix according to *runname* specified in *arpstrn.input*.

**Set up surface data**

Download the surface characteristics data base from the CAPS ftp site. [Note: these steps are necessary only when the soil model is turned on (*sfcphy* = 3 or 4) and when the surface characteristics data from the data base are used (*sfcdat* = 2 or 3)].

**cd** data [move to the *data* subdirectory]
**ftp** caps.ou.edu [connect to the CAPS ftp site (using **anonymous** as your login and your e-mail address as the password]
**cd** *pub/ARPS/ARPS.data* [go to ARPS data subdirectory]
**get** *arpssfc.data.tar.gz* [retrieve the tar file containing the surface and soil characteristic data]

**gunzip** *arpssfc.data.tar.gz* [uncompress surface data files]
**untar** *-xvf arpssfc.data.tar* [untar surface data files, which are placed in an *arpssfc.data* subdirectory]

Modify the control parameters associated with ARPSSFC in *arps.input*. Be sure to correctly specify the full path names for the surface data files, which have been moved to your local machine. Compile and then run **arpssfc**.

**vi** *input/arps.input* [Set control parameters for ARPSSFC and ARPS]
**makearps** *-ncarg arpssfc* [Compile and link program ARPSSFC]
*bin/***arpssfc** < *input/arps.input* > *arpssfc.output* & [Execute program ARPSSFC]

**idt** *gmeta* [Examine the graphic plotting of the terrain field]

A **.sfcdata* file is generated and named with a prefix according to the *runname* parameter specified in *arpstrn.input*. This file is placed in the directory as given by *dirname*. The *gmeta* file is placed in the ARPS root directory.

**Initialize soil fields**

The soil temperature and moisture fields are generally not observed therefore the initialization of soil models is an unsolved problem. For idealized experiments, ARPS allows the user to specify the initial condition via input parameters in *arps.input*. For soil moisture initialization, antecedent precipitation data can be used, with the API (antecedent precipitation index) method. When available, the forecast soil fields from an operational model such as the NCEP ETA may also be used as a hopefully reasonable guess. One does have to be careful about possible mismatch between the soil and/or

vegetation type definitions at the corresponding grid points of the source model and the ARPS. The API method is supported in soil model initialization program, ARPSSOIL. The use of ETA soil model state variables are supported by EXT2ARPS.

### Initialize the ARPS

The prognostic fields of the ARPS may be initialized using three basic options: 1) using a single vertical sounding plus optional initial perturbations (e.g., a thermal bubble); 2) using 3D gridded data interpolated from an external atmospheric model grid; 3) using a 3D gridded analysis from available observations and an analysis background (which can be from an external model or from ARPS itself). (More complex approaches to initialization, such as perturbed initial conditions for ensemble forecasting, forecast cycling, and data assimilation, are not discussed here).

***Option 1**: using a single sounding*

A vertical thermodynamic and wind profile of the environment is used to initialize the ARPS base-state variables. This base state will be horizontally homogeneous.

**vi** *sounding/***.snd* [Prepare a sounding (in the same exact format as provided in the example sounding file *sounding/may20.snd*)]

The ****.snd* file must be referenced in the *arps.input* file with *initopt=1*. Initial perturbations can be specified inside the initialization subroutine and a few types of thermal perturbations can be specified via input parameters, and the option parameter is *pt0opt*.

Note there are also additional options for specifying analytical soundings in the ARPS.

***Option 2**: using data from an external model*

Values for the initial fields are provided by an analysis or forecast from an external model, which is often of coarser spatial resolution and larger domain. In such case, the ARPS forecast is “nested” within this external model. The initial state in general is not horizontally homogeneous.

The module EXT2ARPS is used to extract the data from an external model and interpolate them to the ARPS grid.

**vi** *input/arps.input* [Set control parameters, including the input data file names, for EXT2ARPS]
**makearps** *ext2arps* [Compile and link program EXT2ARPS]
*bin/***ext2arps** < *input/arps.input* >! *ext2arps.out* & [Execute program EXT2ARPS]

The current set of external forecast models from NOAA's National Center for Environmental Prediction (NCEP), which may be used to provide ARPS initial data include:

- Rapid Update Cycle (RUC) model
- Eta model
- Global Forecasting System (GFS, formerly AVN)

Initial data may also be extracted from a (generally coarser resolution) ARPS analysis or forecast using **arpsintrp**.

**vi** *input/arpsintrp.input* [Set control parameters]
**makearps** *arpsintrp* [Compile and link program ARPSINTRP]
*bin/***arpsintrp** < *input/arpsintrp.input* >! *arpsintrp.out* & [Execute program ARPSINTRP]

The datasets that are produced by either EXT2ARPS or ARPSINTRP are of the same format as an ARPS history dump as is described in Section 3.9. These datasets may be used to initialize the ARPS by setting, in *arps.input*, the parameter *initopt = 3* as well as *inifile* and *inigbf* to the dataset names. . In addition, if parameter *exbcdump* is set to non-zero in *arpsintrp.input*, external boundary condition files for ARPS will be created.

***Option 3****: using an analysis from the ARPS Data Analysis System (ADAS)*

A 3D gridded analysis can be produced from available observations and an analysis background by the ADAS, and used to initialize the ARPS. The analysis background data (in ARPS history format) can be from either an external model (output of *ext2arps*) or ARPS. Details on how to prepare the observational data for analysis by ADAS and then also the list of ADAS parameters and functions are given in section _????___.

The resultant ADAS datasets are identical in format as an ARPS history dump as described in Section 3.9. To initialize the ARPS, these ADAS datasets must be referenced in *arps.input* using the parameters *inifile* and *inigbf*. Also *initopt = 3* must be set.

**Executing ARPS**

The ARPS may be easily compiled for execution on a single processor or across a set of processors on either shared-memory or distributed-memory machines.

**Option 1:** Steps to run non-parallel version of ARPS

Steps to run ARPS after all necessary data have been prepared:

**vi** *input/arps.input* [Set control parameters for ARPS]
**makearps** *arps* [Compile and link the ARPS main program]
*bin/***arps** < *input/arps.input >! arps.output* &

The *arps.input* file contains the control parameters for ARPS and other supporting programs, and is in the NAMELIST format. The detailed parameter settings for ARPS are described in Chapter 4.

The output data are dumped periodically or at specified times as specified in *arps.input.* The naming convention and format of these datasets are described in the next section.. Instructions on how to examine them are presented in Section 3.10.

**Option 2:** Steps to run shared-memory parallel version of ARPS

ARPS does not contain explicit share-memory-parallel compilation directives. To use more than one processor on share-memory multi-processor systems, the Fortran compiler has to support automatic parallelization. This capability is available with the latest versions of IBM, SGI, SUN, Intel ifc and efc, and Portland group pgf90 Fortran 90/95 compilers. For SGI, the support requires license for a separate package. On the Tru64 platform, automatic parallelization support requires a third party preprocessing package, called KAP Fortran (orderable from COMPAQ/HP). Additional compiler flags are required in all cases to invoke the parallelization. On most platforms, setting environmental parameters, PARALLEL and OMP_NUM_THREADS, tells the program (*arps*) how many CPUs to use.

To compile ARPS (or any other program in the ARPS package) with autoparallelization options, enter

**makearps –p arps**

Do the following before running ARPS (note that the environmental variables to set may be platform dependent).

**setenv** OMP_NUM_THREADS number_of_cpus_to_use
**setenv** PARALLEL number_of_cpus_to_use

**Option 3:** Steps to run distributed-memory parallel version of ARPS.

Executing the ARPS on a distributed-memory machine is slightly more complicated than on shared-memory machine. The procedure is given separately in section 3.11.

## 3.9. Running ADAS and Data Preprocessors

Steps required to generate a 3D gridded objective analysis for atmospheric state variables, whether for diagnostic studies, nowcasting or ARPS initialization purposes, are given below. Since the process involves the use of model forecast as the analysis background, and attempts to combine the background and observations in an optimal way, the process is often referred to as data assimilation. This section outlines the steps involved in producing such analyses using ADAS, the ARPS Data Analysis System.

### Create background field

Creating an analysis of the atmospheric state at a given time requires a reasonable "first-guess" value for the core meteorological variables at each gridpoint of a user-defined 3D domain. These background values may be provided using forecast data from an external model (such as the RUC or Eta models) or by the ARPS itself. Instructions on how to extract and interpolate external model data using EXT2ARPS are mentioned above in Section 3.8.

### Format data

The observational data to be used in the analysis need to be converted to a format acceptable by ADAS. A simple module may be created by the user to convert the data by referencing the FORMAT statements in ADAS that are associated with reading various data types. In particular, it is straight-forward to convert ASCII files for single-level observations (such as surface observations) and multi-level observations (such as rawinsonde and wind profiler data). In addition to ASCII format, ADAS also support HDF format observations. Guidelines for these data formats and suggestions for generating code are given in Appendix ____ . Sample data files for surface stations, upper-air rawinsonde and profiler data can be found at ftp://ftp.caps.ou.edu/pub/ARPS/sampleobs. A more complete data set for a test case, grouped in tar file data.25may1998.tar.gz can be found at ftp://ftp.caps.ou.edu/pub/ARPS. Converters for converting observations from

the Unidata IDD (Internet Data Distribution) data feed to the ADAS required format are available at user's request.

*Radar data*

ADAS can use WSR-88D Doppler radar data for cloud analysis and velocity adjustment, after the data are properly prepared. The conversion of radar data is slightly more complicated. The data, which are sequentially collected by scan elevation, must be averaged into vertical data "columns". The modules NIDS2ARPS (for Level-III NEXRAD data) and 88D2ARPS (for the raw Level-II NEXRAD data) are provided to prepare the radar data (Doppler winds and reflectivity) in an appropriate manner.

***NIDS2ARPS**: Level III (NIDS) radar data*

**vi** *Kxxxref.list* [for a given radar site 'Kxxx', create in the ARPS root directory a file containing a list of reflectivity files for individual elevation scans that are available at or near the same reference time (the reflectivity data files may exist in any directory on the local system, but the full path name must be given)]

Example *Kxxxref.list* file contents:

```
/scratch/data/Kxxxyyyymmddhhmm.N0R
/scratch/data/Kxxxyyyymmddhhmm.N1R
/scratch/data/Kxxxyyyymmddhhmm.N2R
/scratch/data/Kxxxyyyymmddhhmm.N3R
```

**vi** *Kxxxvel.list* [create a dataset for 'Kxxx' Doppler velocity fields for individual scans at or near the reference time that corresponds with the reflectivity datasets]

Example *Kxxxvel.list* file contents:

```
/scratch/data/Kxxxyyyymmddhhmm.N0V
/scratch/data/Kxxxyyyymmddhhmm.N1V
```

Execute NIDS2ARPS using the same *arps.input* file as to be used by ADAS, insuring that the domain configuration is the same as will be designated when running ADAS.

**vi** *input/arps.input* [set NIDS2ARPS-related parameters]

**makearps** *nids2arps* [compile NIDS2ARPS]
*bin/***nids2arps** *< input/arps.input > ! nids2arps.out &*

***88D2ARPS**: Level II (wideband) radar data*

**vi** *input/arps.input* [set 88D2ARPS-related parameters (those which define the physical domain configuration and the model background field to be used by ADAS)]

**makearps** *88d2arps* [compile 88D2ARPS]
*bin/***88d2arps** *Kxxx – diskf {radar file} < input/arps.input > ! 88d2arps.out &*
[where *Kxxx* denotes a specific NEXRAD radar site; *radar file* is the Level II radar dataset

The output dataset, with a prefix denoted by the radar site (*Kxxx*) is generated in the ARPS root directory, are to be used by ADAS. The Level II and Level III data from different radars can be used together in ADAS, but because of their better quality, the Level II data should be used whenever available.

*Satellite data*

The module MCI2ARPS converts raw satellite data into the format as is required by ADAS. Note that satellite data is used in ADAS only when the cloud analysis option is turned on.

**vi** *input/arps.input* [set MCI2ARPS-related parameters (which are primarily the physical configuration of the domain)]

*bin/***mci2arps** *{-hdf n } {sat file} < input/arps.input > ! mci2arps.out &*
[ *n* is the level of hdf compression (4 is often used); the *sat file* may be a visible, water vapor, or Ch4 (???) raw data file]

Example satellite raw data files:

| | |
|---|---|
| *goes12vis_ yyyymmdd_hhmm* | [visible data] |
| *goes12wv_yyyymmdd_hhmm* | [water vapor data] |
| *goes12ch4_yyyymmdd_hhmm* | [longwave IR data] |

MCI2ARPS must be rerun for each satellite data type. The output datafiles, which are generated in the ARPS root directory, must be referenced in the cloud analysis section of the *arps.input* file (parameters *ir_fname* or *vis_fname* ) in preparation for the execution of ADAS.

**Data error analysis and quality control**

*observation blacklist file*

The blacklist file is intended to be used to filter out stations with chronic instrument quality or citing problems that make the observations unrepresentative for the scale or feature being analyzed. An example *blacklist.sfc* file is provided in the subdirectory *data/adas*. Instructions regarding the setup of a blacklist file for your specific application are provided in Section __.___.

*background and observation error file*

To conduct an objective analysis, it is necessary to define an acceptable range of errors for each of the core field values (pressure, temperature, relative humidity, and winds) from both the various observation types and the background model forecast. Example error files are provided in the subdirectory *data/adas*. Recommendations on how to specify appropriate error values are given in Section __.___.

**Execute ADAS**

Once the chosen set of datafiles have been appropriately formatted and the background field identified, a 3D gridded analysis may be generated using ADAS.

**vi** *input/arps.input* [Set control parameters associated with ADAS]
**makearps** *adas* [Compile and link the ADAS main program]
*bin/***adas** < *input/arps.input >! adas.output* &

The resultant datasets are placed in the *dirname* subdirectory as is specified in *arps.input*. The datasets may be used to initialize the ARPS.

## 3.10. Model Output

ARPS model provides options for a number of history data formats. The model history datasets, which are sometimes referred to as history dumps, provide the values of the model core variables at a given run time. These datasets may be generated using various data formats and may be used to initialize the ARPS. The output directory is specified by the parameter *dirname* in *arps.input.*

The main output files from ARPS as well as their nomenclature are described below. For each, *runname* (as set in *arps.input*) is used as a prefix in order to uniquely identify the files associated with a given model run.

- *arps.output* - A file containing information from the standard output of model execution. Formatted printout of field arrays, run time diagnostic information, warnings of improper parameter settings, job aborting information, timing statistics of various packages, *etc.*, can be found in this file. The user is strongly encouraged to examine this file after the job execution.

- *runname.log* - A record of all input parameters in a NAMELIST format ready to be re-used as a input file for ARPS execution.

- *runname.maxmin* - The maximum and minimum values of various fields at time intervals specified by the user in the file *arps.input*. This file is read by program ARPSPLTMAX to plot time series of certain model variables.

- *runname.rstnnnnnn* - A set of 'restart' binary data files produced at an interval specified by the user in *arps.input*. Here *nnnnnn* are six (or more when needed) digits representing the model time in seconds. Each file contains two time levels of data and can be used to restart the model. After the restart, the model should run as if it had never stopped.

- *runname.fmtgrdbas* - history data containing only the time-invariant base-state and model grid arrays. Each run will generate only one such file. Here *fmt* is one of *bin, asc, hdf, pak, bn2, svi, grb, v5d, and grds,* indicating the format of the data. A description on these formats and their use can be found in Section 10.1.

- *runname.fmtnnnnnn* - a set of history data files that contain all necessary model variables for post-processing purposes. Here *fmt* represents the data format as before, and *nnnnnn* are six or more digits representing the data

time in seconds. Note that the fields contained in these files are total fields that are independent of the base-state fields in file *runname.fmtgrdbas*.

## 3.11. Vector Graphics Analysis of ARPS Data

A vector graphics plotting program known as ARPSPLT (see Section 10.2) is provided with ARPS package. It is based on graphics software known as ZXPLOT. ARPSPLT uses history data from ARPS. To use this package, you must install the ZXPLOT executable library on your machine first. The object codes for most popular systems can be obtained from *anonymous@ftp.caps.ou.edu:pub/ZXPLOT3*. More detailed information on the package can be found in Chapter 12 and at http://www.caps.ou.edu/ZXPLOT.

There are two versions of ARPSPLT executable, **arpspltncar** and **arpspltpost**, depending on the choice of output format of the graphics data. **arpspltncar** links ARPSPLT with ZXPLOT and NCAR Graphics package and the output will be NCAR Graphics metafiles. **arpspltpost** links with ZXPLOT only (NCAR graphics is not needed), and Postscript output will be produced. Naturally, **arpspltncar** requires that NCAR Graphics be installed on your system. NCAR Graphics is available at http://ngwww.ucar.edu/ngdoc/ng/.

Alternatively, a user can write his/her own analysis program using the history data read facilities supplied with ARPS. A template program for reading history data is provided in Section 10.1. The program file, *arpsread.f90*, is found in src/arps. For a quick start up, the steps to run ARPSPLT are:

**vi** *input/arpsplt.input* [Set plotting control parameters including the names of history files to be processed]

**makearps** *arpspltncar*
[ Compile and link ARPSPLT with ZXPLOT and NCAR Graphics libraries]

*bin/***arpspltncar** < *input/arpsplt.input >! arpsplt.output &*
[Execute ARPSPLT program to generate graphic metafile output]

**idt** gmeta [View graphics output]

or

**makearps** *arpspltpost*

[Compile and link ARPSPLT with ZXPLOT and ZXPLOT PostScript driver]

*bin/***arpspltpost** < *input/arpsplt.input >! arpsplt.output &*
[Execute ARPSPLT program to generate PostScript graphic output]

**gv** *runname.ps* [View Postscript output]

If you have some basic understanding of Postscript language, you can directly edit the PS file which is in ASCII format. You can also use graphics software such as Adobe Illustrator to interactively edit the pictures. Unfortunately, Illustrator can only import the first page in the PS file so you either have to manually split the pages (you need to have a copy of the header in each figure and need the file with SHOWPAGE and Q lines. SHOWPAGE separates which page in the PS file) or plot one page at a time. Another possibility is to convert the PS file to a PDF file (using Adobe Acrobat, the full version, not Reader, or using ps2pdf available on many Unix or Unix like platforms), then have Illustrator open the PDF file. You will be able to specify the page you want to open in the PDF file. You can save the edited figure in EPS format and directly insert it into word processors such as Microsoft Word. The vector resolution will be preserved when printed on a Postscript printer or to PDF Writer.

## 3.12. The Parallel Version of ARPS

Running ARPS on distributed-memory platforms such as a Linux cluster is supported by employing a 2D horizontal domain decomposition strategy (see Chapter 11) and using the Message Passing Interface (MPI) library. It is assumed that the MPI library is properly installed on your system and the MPI job environment properly setup.

Running the MPI version of ARPS is actually rather similar to the single processor version. In addition to all other parameters, several additional parameters need to be set in *arps.input.* They include *nproc_x* and *nproc_y*, which specify the *nproc_x* by *nproc_y* 'processor grid' for domain decomposition. Note that (*nx*-3)/*nproc_x* and (*ny*-3)/*nproc_y* must be integer. *nproc_x* × *nproc_y* is the total number of processors to use.

The actual command to run an MPI program depends on the platform used. The following gives the most common form, used by most Linux clusters, and on SGI Origin and HP/COMPAQ systems. Depending on the options set in *arps.input*, two additional steps that split the input and join the output data files may be needed, as outlined below.

**vi** *arps.input* | Set control parameters in input file

| | |
|---|---|
| **makearps splitfiles** | \| Create executable *splitfile.* |
| **splitfiles** < *arps.input* | \| Split input data files into individual<br>\| pieces to be read by individual processors.<br>\| This step is needed only when<br>\| *inisplited*=0.. |
| **makearps** *arps_mpi* | \| Create the parallel version if ARPS<br>\| using MPI |
| **bin/arps_mpi** *-nprocs ncpus <arps.input >! arps.output* | |
| | \| Execute *arps_mpi* specifying<br>the number of processors to use. |
| **makearps joinfiles** | \| Create executable *splitfile.* |
| **joinfiles** < *arps.input* | \| Join together the model output written<br>\| by individual processors into single ones<br>\| for the entire model domain. This step is<br>\| needed only when *dmp_out_joined*=0. |

When *inisplited* =0 and *dmp_out_joined* =0, each processor will read and write its own input and output data. The *splitfiles* utility will split all the necessary input data files, while *joinfiles* will join all output file together. The program assumes that all processors have access to the file system on which the input and output files will be stored. Whether one should read and write individual files or single files depending on the network speed among the processors and the speed of disk I/O. Unless this is a significant penalty, the latter option is recommended for simplicity. The latter option is, however, only available for binary and HDF I/O format, however.

## 3.13. Where to Get Help

For help with general problems, please first visit the Documentation and User Support pages of the ARPS web site at http://caps.ou.edu/ARPS. For your questions to be answered by CAPS staff, send e-mail to *arpssupport@ou.edu*. Mailing list arpsforum@lists.ou.edu can be subscribed to and used to post questions for peer ARPS users as well as developers. You may subscribe to mailing list *arpsnews@lists.ou.edu* can to receive periodic news posted by CAPS personnel. The searchable archives of past mails posted to all three mailing lists are accessible via the web and the links can be found at the ARPS User Support page. Other questions and comments may be directed to:

*ARPS Support Group*
*Center for Analysis and Prediction of Storms*
*University of Oklahoma*
*Sarkeys Energy Center, Rm 1110*
*100 East Boyd*
*Norman, OK 73019-0628, USA*

*Phone: 405-325-0453*
*Fax: 405-325-7614*
*E-mail: arpssupport@ou.edu*