Getting Started

This chapter provides you with the information on the essential steps necessary to access, compile and run 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-77, and
- file transfer protocol (FTP).

3.2. Accessing ARPS Code via Anonymous FTP

ARPS model is accessible via anonymous FTP as follows (all computer commands to be entered by the user are shown in *italics*):

- Log onto the machine where you wish to place ARPS code and get into an appropriate directory.
- Type **ftp** *ftpcaps.uoknor.edu* [open FTP session to CAPS anonymous FTP server]
- Log in as **anonymous** and use your complete e-mail address as your password

- **cd** *pub/ARPS* [change to public access ARPS directory]
- **ascii** [establish ASCII format for data transfer]
- **get** *README_arps* [copy ARPS readme file to your home account]
- **bin** [set binary mode for data transfer]
- **get** *arps_current.tar.Z* [copy latest ARPS source code]
- **get** *arps40.docs.tar.Z* [copy ARPS 4.0 document files (this guide in postscript form)]
- **get** *arps40.data.tar.Z* [copy ARPS 4.0 terrain and surface data files]
- **quit** [terminate the FTP session]

You should first try to read *README_arps*, which provides a brief description of the current version of ARPS. The other fours files ending with *tar.Z* are the compressed tape archive (tar) file of the model code, postscript version of this User's Guide, terrain and surface characteristic data sets and files used by ARPS validation test suite. If you already have a copy of this guide and want to examine ARPS code and perform a few idealized tests only, *arps_current.tar.Z* maybe sufficient for you at first. Please note that *get arps_current.tar.Z* is actually a soft link an actual file, which is usually named *arps4.0.nn.tar.Z*. Here *nn* represents a subversion number. After you complete command ***get*** *arps_current.tar.Z*, you will actually find *arps4.0.nn.tar.Z* on your disk instead of *arps_current.tar.Z*.

To decompress and expand the code, type the following:

- **uncompress** *arps4.0.nn.tar.Z* [expand the compressed tar file]
- **tar** *-xvf arps4.0.nn.tar* [split the tar file into separate files that are placed into a subdirectory called *arps4.0.nn*]
- **cd** *arps4.0.nn* [change directory to arps*4.0.nn*]
- If you now type **ls** [list files], you should see the following (approximately):

```
./                celtrk3d.f        joinfiles.f       r2hdims.inc
../               chksym3d.f        july21_ice.snd    raydmp3d.f
CHANGES           cmix3d.f          lapsparms.cmn     rdextfil.f
COPYRIGHT         craylib3d.f       linkxlC*          rdextfil_laps.f
README            cumucst.inc       load1deg.file     rdextfil_ruc.f
advct3d.f         cumulus3d.f       load30sec.file    read3d.f
arps40.data/      diffdims.inc      load5min.file     read_obs.f
arps40.docs/      dims.inc          make.arps40       readinput.F
arps40.f          dir1deg.f         makearps*         rst3d.f
arps40.input      dir30sec.f        maproj3d.f        setbdt3d.f
arps40.tree       dir5min.f         may20.snd         setupmpi.f
arps40.validate/  dump3d.f          may20_calm.snd    setuppvm.f
arpscvt.input     energy3d.f        meraf.inc         setuppvmsp2.f
arpscvt11.f       exbc.inc          micro3d.f         setuppvmt3d.f
arpsdiff.input    exbc3d.f          micro_ice3d.f     sfcphy3d.f
arpsdiff12.f      exbcio3d.f        mpdims.inc        sfcphycst.inc
arpsextsnd.f      ext2arps.f        mpisubs.f         smooth3d.f
arpsextsnd.input  extdims.inc       mpitranssubs.c    soildiag3d.f
arpsgrads.gs      extlib3d.f        mptrans.c         soilebm3d.f
arpslib3d.f       fjoindumps.f      mthermo.f         solve3d.f
arpsplt.input     fjoinload.file    ncarg3d.f         spcounty.mapdata
arpsplt47.f       force3d.f         netcdf.inc        splitfiles.f
arpspltlib.f      fsplitdump.f      netio3d.f         stable.snd
arpspltmax.f      fsplitexbc.f      neutral.snd       subroutine.list
arpspltmax.input  fsplitrestart.f   nohdfio3d.f      sviio3d.f
arpsprt.input     fsplitsoil.f      noncarg3d.f       t3dtranssubs.c
arpsprt12.f       fsplitsoilini.f   nonetio3d.f      terrain.inc
arpsprtlib.f      fsplitterrain.f   nopakio3d.f       testmkarps*
arpsr2h30.f       genlib3d.f        nosviio3d.f       thermo3d.f
arpsread.f        globcst.inc       nqs.arps*         timelib3d.f
arpssfc.inc       gradsio3d.f       operat3d.f        tinteg3d.f
arpssfc12.f       grdtrns3d.f       out3d.f           tke3d.f
arpssfclib.f      hdfio3d.f         outlib3d.f        tmix3d.f
arpssoil.f        ibmlib3d.f        pakio3d.f         usstate.mapdata
arpstern.input    img3d.f           par.inc           wirfrm.f
arpstern12.f      indtflg.inc       par1d.inc         wirfrmstub.f
awkinput          inibase3d.f       par2d.inc         xncar.f
barnes3d.f        init3d.f          phycst.inc        xpost.f
bc3d.f            initpara3d.f      pltgrid.f
bcdif3d.f         intfield.f        pvmsubs.f
bndry.inc         iolib3d.f         pvmtranssubs.c
```

If you wish to locate a file that contains a specific subroutine (or a specific string), type the following:

grep *-i “subroutine name” *.f*

You may also find some files in the CAPS anonymous FTP directory, that have a suffix of *.gz*. These files have been compressed using the GNU Free Software Foundation software **gzip**, and can be expanded using command **gunzip**. The source code for **gzip** and **gunzip** can be obtained from many anonymous FTP sites on the internet.

If you encounter problems with the above procedure, send electronic mail to *arpsuser@uoknor.edu or* call CAPS at 405-325-0453.

3.3. Examining the Files

Note the following files now in your ARPS directory:

- *README* A file that contains the basic information on the content of this directory.
- *COPYRIGHT* Copyright notice for ARPS.
- *CHANGES* A file that documents the modification history of ARPS system.
- *makearps* A UNIX shell script for compiling ARPS and other utility programs.
- *make.arps40* A make description file used by *makearps.*
- **.f* FORTRAN-77 source files containing the model code.
- **.inc* FORTRAN include files used by the source code.
- **.input* Input files containing the control parameters of ARPS and other utility programs.
- *may20.snd* A sample sounding file.
- *arps40.data* A directory containing data sets used by ARPS
- *arps40.validate* A directory containing the files used by ARPS validation experiments.
- *arps40.docs* A directory containing the PostScript files of ARPS 4.0 User's Guide (Note: the actual files will be distributed separately from the main ARPS tar file, see Section 2.1).

Examine the ASCII source files as you wish, noting that file *arps40.f* contains the driver program, ARPS40, for ARPS. See Chapter 5 for further information on the program and subroutine functionality.

3.4. 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.

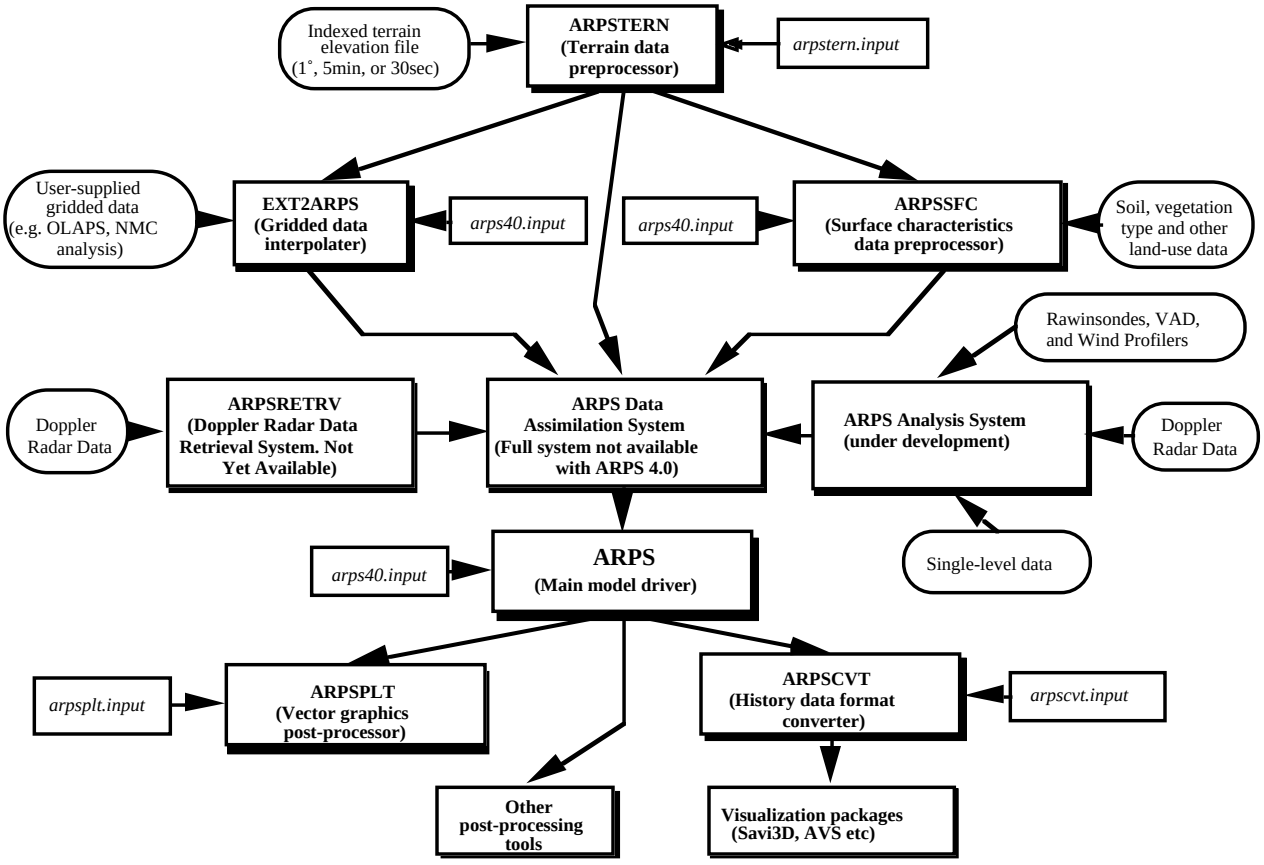

Figure 3.1. ARPS model process flow chart.

3.5. Compiling ARPS and Supporting Programs

The compilation and linking of ARPS40 and all other ARPS-supported utility programs are orchestrated by a UNIX shell script, **makearps**. The script invokes a **make** command based on the parameters provided to it. **make** is a UNIX utility for managing code compilation based on object dependencies. **make** 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 makes use of description file *make.arps40*. A help page similar to the following will be listed when command **makearps** is entered without any arguments:

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

cmd is one of the following [default is *arps40*]:

(Note names in *italic*, such as *arps40*, are *cmd* parameters passed to makearps script, those in **bold**, such as **arps40**, are the names of the executables produced by '**makearps** *cmd*', and those in capital letters, such as ARPS40, refer to program names)

Executable commands:

arps40	Produce executable **arps40** for ARPS40
arps40_pvmt3d	Produce PVM version executableof ARPS, **arps40_pvmt3d**, to run on a Cray T-3D.
arpscvt	Produce executable **arpscvt** for ARPSCVT
arpspltncar	Produce NCAR graphics version executable **arpspltncar** of ARPSPLT
arpspltpost	Produce PostScript version executable **arpspltpost** of ARPSPLT
arpspltmax	Produce executable **arpspltmax** for ARPSPLTMAX
arpsprt	Produce executable **arpsprt** for ARPSPRT
arpsr2h	Produce executable **arpsr2h** for ARPSR2H
arpsdiff	Produce executable **arpsdiff** for ARPSDIFF
arpsextsnd	Produce executable **arpsextsnd** for ARPSEXTSND
arpssfc	Produce executable **arpssfc** for ARPSSFC
arpstern	Produce executable **arpstern** for ARPSTERN
dir1deg	Produce executable **dir1deg** for DIR1DEG
dir5min	Produce executable **dir5min** for DIR5MIN
dir30sec	Produce executable **dir30sec** for DIR30SEC
ext2arps	Produce executable **ext2arps** for EXT2ARPS
ruc2arps	Produce executable **ruc2arps** for the RUC version of EXT2ARPS
laps2arps	Produce executable **laps2arps** for the OLAPS version of EXT2ARPS

Object library archive:

arpssolver	Produce an object library for ARPS solver

Create tape archive (tar) files:

Allfiles.tar	Create tar files of the entire ARPS systems except for the documentation
arps40.tar	Create tar files associated with ARPS40
arpscvt.tar	Create tar files associated with ARPSCVT
arpsplt.tar	Create tar files associated with ARPSPLT
arpsprt.tar	Create tar files associated with ARPSPRT
arpsr2h.tar	Create tar files associated with ARPSR2H
arpsdiff.tar	Create tar files associated with ARPSDIFF
arpsextsnd.tar	Create tar files associated with ARPSEXTSND
arpssfc.tar	Create tar files associated with ARPSSFC
arpstern.tar	Create tar files associated with ARPSTERN
ext2arps.tar	Create tar files associated with EXT2ARPS
arpsdocs.tar	Create tar documentation files of ARPS
arpsdata.tar	Create tar data sets used by ARPS

Clean-up object files:

clean	Delete all object files
clean.arps40	Delete all object files related to ARPS40
clean.arpscvt	Delete all object files related to ARPSCVT
clean.arpsplt	Delete all object files related to ARPSPLT
clean.arpsprt	Delete all object files related to ARPSPRT
clean.arpsr2h	Delete all object files related to ARPSR2H
clean.arpsdiff	Delete all object files related to ARPSDIFF
clean.arpsextsnd	Delete all object files related to ARPSEXTSND
clean.arpssfc	Delete all object files related to ARPSSFC
clean.arpstern	Delete all object files related to ARPSTERN
clean.dirtern	Delete *dir1deg.o dir5min.o dir30sec.o*
clean.ext2arps	Delete all object files related to EXT2ARPS

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	Parallel option
-s	Performance statistics
-inline	Inline small, frequently called routines
-ice	(CRAY only)

History data (I/O) library support .i.makearps:

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

hdf	Include data I/O routines using HDF format

net	Include data I/O routines using NetCDF format
savi	Include data I/O routines using Savi3D format
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.

Options passed to **make**:

-*n*	Displays commands, but does not run them.

Miscellaneous options:

-*m mach*	Force machine type to *mach* (otherwise use tcsh HOSTTYPE to get the machine type). Recognized machine types include: *cray, iris4d, rs6000* and *sun4.* If none of the above matches, default type *unix* is used.
-*help*	Print this help page.

The script uses the 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. The -*io* option specifies the available external libraries. By default, no external library is assumed to be available.

3.6. 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. We list these steps in the following (we assume NCAR Graphics has been installed on user's computer system):

vi *dims.inc*	[Set the model grid dimensions]
vi *may20.snd*	[Prepare a sounding if required]
vi *arps40.input*	[Set control parameters for the model run]
makearps *arps40*	[compile and link ARPS]
arps40 < *arps40.input* > *arps40.output* &	[Execute the model]
vi *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]

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

ctrans -d X11 *gmeta* [Examine the graphic metefile using NCAR Graphics viewing tool **ctrans**]

3.7. A Complete Run of ARPS

The steps required to make a complete ARPS run using real terrain, real surface characteristic data, and 3-D initial data set are listed as follows:

- Steps to prepare the terrain data, assuming the terrain data base has been properly set up (UNIX vi editor is used for illustrative purposes, those in [] are comments. These steps are unnecessary if the analytic terrain option is chosen):

  vi *terrain.inc* [Set grid dimension, etc.]
  vi *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]

  dir1deg < *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]
  dir5min < *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]
  dir30sec < *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]
  arpstern < *arpstern.input* > *arpstern.output* &
  [Execute program ARPSTERN]
  ctrans -*d X11 gmeta*
  [Examine the graphic plotting of the terrain field]

  Terrain data file *arpstern.dat* is used by ARPS40.

- Steps to prepare the surface characteristic data, assuming the data base has been properly set up. 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* =3).

  vi *dims.inc* [Set the model grid dimensions]
  vi *arps40.input* [Set other control parameters for ARPSSFC and ARPS40]
  makearps *-ncarg arpssfc*
  [Compile and link program ARPSSFC]
  arpssfc < *arps40.input* > *arpssfc.output* &
  [Execute program ARSSSFC]

  A surface characteristics data file whose name is specified inside *arps40.input* is produced.

- Steps to produce a three-dimensional initialization data set and the data files used to force the lateral boundaries of ARPS. The data are interpolated from data sets from another model or observational analysis grid. In general, file *rdextfile.f* has to be edited, and usually additional libraries have to included, so that the program can read in the user-supplied data set (see Section 8.5).

  vi *dims.inc* [Set the model grid dimensions]
  vi *extdims.inc* [Set the dimensions of external data arrays]
  vi *arps40.input* [Set control parameters, including the input data file names, for EXT2ARPS]
  makearps *ext2arps* [Compile and link program EXT2ARPS]
  ext2arps < *arps40.input* >! *ext2arps.out* &
  [Execute program EXT2ARPS]

  If ARPS is initialized with a single sounding, a sounding file has to be prepared instead of a 3-D initial data set. Other options using analytic base states are also available (see Section 8.4).

- Steps to run ARPS40 when all necessary data have been prepared:

  vi *dims.inc* [Set the model grid dimensions]
  v i *exbc.inc* [Set the dimensions of external boundary condition arrays. Set all three dimensions to 1 to collapes the arrays when external boundary forcing is not used.]
  vi *arps40.input* [Set control parameters for ARPS40]
  makearps *arps40* [Compile and link ARPS main program ARPS40]
  arps40 < *arps40.input >! arps40.output* &

arps40.input is a input file containing the control parameters for ARPS40 and other supporting programs, and is in the NAMELIST format. The detailed parameter settings for *arps40* are described in Chapter 4.

A set of output data will be produced. Instructions on how to examine them are presented in the next subsection.

3.8. Model Output

ARPS model provides options for a number of history data formats. History data are data written by the model and contains all necessary model variables for analysis. The history data formats can also be used to initialize the model. In this document, a history data file is sometimes referred to as a history dump. The output directory is specified in *arps40.input. runnam* refers to a character string not longer than 6 characters that is chosen (set in *arps40.input)* to identify your simulation.

The main output files from ARPS40 are described below.

- *arps40.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, *etc.*, can be found in this file. The user is strongly encouraged to examine this file during and after the job execution.

- *runnam.log* - A record of all input parameters in a NAMELIST format ready to be re-used as a input file for ARPS40 execution.

- *runnam.maxmin* - The maximum and minimum values of various fields at time intervals specified by the user in the file *arps40.input*. This file is read by program ARPSPLTMAX to plot time series of certain model variables.

- *runnam.rstnnnnnn* - A set of 'restart' binary data files produced at an interval specified by the user in *arps40.input*. Here *nnnnnn* are six 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. A restart file can also be read by program ARPSR2H, which creates a one-time-level history data.

- *runnam.fmtgrdbas* - history data containing only the time independent 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, net, npk, grds*, indicating the format of the data. A description on these formats and their use can be found in Section 10.1.

- *runnam.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 digits representing the data time in seconds.

3.9. Vector Graphics Analysis of ARPS Data

CAPS distributes a vector graphics plotting program, ARPSPLT (see Section 10.2), 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. The object codes for most popular systems can be obtained from *anonymous@ftpcaps.uoknor.edu: pub/ZXPLOT*. More detailed information on ZXPLOT can be found in Chapter 12. 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. For a quick start up, the steps to run ARPSPLT are:

vi *dims.inc* [Set the grid dimensions to match those of ARPS40]

vi *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]

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

or

makearps *arpspltpost*
[Compile and link ARPSPLT with ZXPLOT and ZXPLOT PostScript driver]

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

3.10. The Parallel Version of ARPS

The parallel version of ARPS 4.0 was created to support high resolution prediction research as well as conduct MPP research. A significant part of the serial code remains unchanged and most of the changes have been made to the data I/O routines and those related to the lateral boundary conditions. The translators converts code markers, which have been embedded into the official ARPS version, into calls to the communication routines.

(Note: The code markers have a **cMP** prefix, and it is important that the location or the content of these markers is not disturbed)

The model assumes some understanding of parallel computing by the user, and the existence of a parallel programming platform (either PVM or MPI). All processes read the input file and the sounding file (if necessary). It is necessary that all programs have access to file system which contains these files. (CAPS has made available two guides which deal with the subject of understanding parallel computing, and using the parallel version of ARPS in more detail. Please send an email to *arpsuser@uoknor.edu* for more information).

Disclaimer: Although a lot of effort has been taken to make the parallel version more robust, problems might arise when using the code. Many of these problems could be due to the local implementation, or system related (machine failure *etc.*), but if they are neither, contact us at *arpsuser@uoknor.edu*.

The parallel version of ARPS is initialized slightly differently from the serial version. ARPS model configuration file, dims.inc has undergone philosophical change and a new file, par.inc, has been added to configure the model.

The values *nx* and *ny* now refer to the dimensions of the grid on a single processor (as opposed to the entire grid in regular ARPS). Two more variables, *nprocx* and *nprocy* are used to determine the number of processes/processors to be used in the run. The size of entire grid in the X and Y direction (the Z dimension has not been changed) can be determined by the following formulae

*global_x_size = (nx - 3) * nprocx + 3*
*global_y_size = (ny - 3) * nprocy + 3*

or alternately,

nx = (global_x_size - 3) / nprocx
ny = (global_y_size - 3) / nprocy

(Note: The variables *global_x_size* and *global_y_size* do not appear in the code but are used here as an aid to the calculation of the grid size).

The values of *nprocx*, and *nprocy* should be set equal to the number of processors needed, in the case of an MPP, or number of available machines times 2 or 3 depending on the amount of available memory, speed of the processor *etc*. in the case of a computer cluster.

The new file, *par.inc*, is used to set the size of the communication buffers. The values *nx_mx* and *ny_mx* should be set to *nx* and *ny*. The values could be larger than *nx* and *ny* but in no case should they be set to values smaller than *nx* and *ny*. The maximum number of processors too can be fixed

in this file, but they have been currently set to *nprocx* = 16 and *nprocy* = 16, which are far greater than a typical size.

To run the parallel version of the model, we list the following steps (assuming a run with PVM on the Cray T3D; also assuming that NCAR Graphics has been installed on the users computer system for post-analysis):

vi *dims.inc*	Set grid dimensions
vi *par.inc*	Set parallel version specific parameters
vi *arps40.input*	Set control parameters in input file
splitfiles_t3d < *arps40.input*	Split input data files into individual processor sections
echo *arps40.input* >! *input.name*	Store input file name into an intermediate file
makearps *arps40_pvmt3d*	Create the parallel version for the T3D using PVM
arps40_pvmt3d *-npes XX < input.name >! arps40.output* (on the T3D)	
	Execute the parallel version, specifying the number of processors.
joinfiles_t3d < *arps40.input*	Join the history dumps together
vi *arpsplt.input*	Set control parameters for ARPSPLT input file
makearps *arpspltncar*	Create a sequential version of arpspltncar
arpspltncar < *arpsplt.input*	Run arpspltncar
ctrans *-d X11 gmeta*	Display the graphic metafile

Each processor will write out its own history dump(s) during a run and utilities have been provided to split the input data files (terrain, surface *etc*.) and join the history dumps. The splitfiles utility will split all the necessary input data files, while joinfiles will join the all the history dumps together. Note: Once again, the program assumes that all processors have access to the file system on which the input files and the history dumps will be stored)

The parallel version of ARPS has been ported to two messaging platforms, PVM and MPI, and three different hardware platforms (Cray T3D, IBM SP2 and a cluster of workstations). The executable name indicates the message passing and hardware platform the program will execute on (a PVM version for the T3D is named **arps40_pvmt3d**. A MPI version for the IBM SP2 is named **arps40_mpisp2**). Executables for cluster environments do not have a hardware name extension (*e.g*. an executable using MPI and executing on a cluster of workstations is named **arps40_mpi**). For joinfile and splitfiles, similar rules apply; however, the message passing platform name is ignored. For a complete list of commands, enter **makearps -help**.

3.11. Where to Get Help

For help with general problems, use the electronic mail-based ARPS Information System described in Chapter 2. 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: 405-325-3041

Fax: 405-325-7614

E-mail: arpsuser@uoknor.edu