Answers to Frequently-Asked Questions

In this chapter, we provide answers to some commonly asked questions about ARPS. Most of the questions are collected from the users. We will continue to collect users' questions and comments.

Q: How do I obtain ARPS code and documentation?

A: See section 3.2. The code and documents can also by accessed from the CAPS World Wide Web (WWW) page at *wwwcaps.uoknor.edu*. If you have problem connecting to the site or finding the files, e-mail *arpsuser@uoknor.edu*.

Q: When I typed the command **makearps** *arps40*, I got a message saying that the compilation flags were unknown. What could be the problem?

A: **makearps** tries to use suitable compilation options by determining the system type as one of IBM RS/6000, Cray, Sun, DEC Alpha or an undetermined UNIX system. It's possible that the compilation options prespecified in **makearps** are not compatible with the system version that you are using. In this case, you need to modify the options by editing **makearps.** For more information on the makefiles, see Chapter 3.

Q: The model stopped due to a floating point error in a microphysics routine, but I could not find anything wrong there.

A: The model can become unstable due to problems outside the microphyscis. Instabilities are often caught in the exponential calculations inside the microphysics. Check your model setup and input control parameters carefully.

Q: The model stopped before the specified stop time was reached. What could be the cause?

A: Most likely your time integration was unstable. ARPS checks the velocity field for stability. If the maximum wind speed exceeds 100 m/s, the model will stop and issue a message. It will also make a history data dump for that time. It is also possible that improper input parameters were specified. Check your output file for the input parameter settings and the model run information.

When the integration is unstable, you should examine the model fields to determine the nature of this instability. Unacceptably large time step sizes, improper mixing coefficients and boundary problems are the most common causes of instability. When the model becomes unstable after only a few large time steps, more probably *dtsml* is too large. Too large computational mixing ceofficients can also cause instability.

Q: I re-ran the same executable file, **arps40**, that worked before and got an I/O error message. What could have happened?

A: Check your disk space to see if the disk is full. ARPS can produce a huge amount of data. When the disk is full, the model will fail. You also need to make sure the you have the permission to write into the output directory, *dirname*) which is specified in *arps40.input*.

Q: I did **makearps arps40**, and everything seemed OK, but the executable code **arps40** does not work even if I repeat the **makearps** command. What is going on?

A: Some systems keep the object code of a source file even if the compilation aborts. In this case, **make** may treat the object code as up to date based on its modification date. The executable file thus produced will be invalid. Check the previous compilation messages to see if any files need to be recompiled. You can recompile the troublesome files or remove all the object codes and re-do **makearps**.

If different computers share the same file system, the object code produced on one system may not be compatible with the one generated on another. In this case, you need to delete all *.o files and re-do **makearps**.

Q: We tried to compile ARPS on the Cray using the command '**make arps40.cray**', but failed. What are we missing?

A: With ARPS 4.0, shell script **makearps** is used to control all the compilations. The proper command is **makearps** + arguments. You can get a list of commands by entering **makearps** without any arguments.

To compile and link ARPS40, do **makearps arps40**. You can omit the machine type parameter.

Q: Is it possible to initialize ARPS with a full 3-D data set yet?

A: You can initialize ARPS using an external 3-D data set with initialization option *initopt* = 3. The format of the initial data set is exactly the same as the history data. A sample program, EXT2ARPS, is provided that interpolates an external data set to the ARPS grid and writes it in the history data format. A user is required to provide his/her own subroutine to read the external data (see rdextfil.f).

In ARPS, the base-state arrays are horizontally homogeneous, but the total time-dependent arrays are fully three dimensional. It is the total fields that you are initializing. Typically the base-state arrays are assigned with some kind of horizontally averaged values and this average should be taken at constant height levels, rather than along the coordinate surfaces.

Q: We had problems getting the Cray version of ARPS to read a terrain data file that we created using our own program. ARPS ends abnormally when trying to read the first record in the file although it opens the file fine. The write statements we used to create the file appear to be identical to those in *arpstern12.f*. What's happening?

A: ARPS uses the IEEE binary format (the one used by most 32 bit Unix workstations) rather than the Cray native binary format for the terrain file (and other binary input and output files). This is achieved by calling two Cray routines

```
CALL asnctl ('NEWLOCAL', 1, ierr)
CALL asnfile(tern_file,'-F f77 -N ieee', ierr)
```

before the open statement for file `tern_file`. The opened file is then assumed to be in the IEEE format. You need to add these two statement before the open statement of your program that creates the terrain data, so that the generated data are written in the IEEE format.

Q: I ran a simulation with ARPS 4.0 on a Cray supercomputer. I wanted to do a 1 hr integration, but the job didn't finish, even after 10 CPU hours. The simulation included ice.

Then, I submitted two short jobs to the Cray using ARPS 4.0. The only difference between the two is that one included ice and the other didn't. The job without ice performed the 600s integration in 1995 CPU seconds. After 1 CPU hour, the job with ice had only integrated out to 287s. So the simulation with ice takes much longer. Does this seem to be right?

A: Our warm rain microphysics subroutine is maximally optimized. We evaluate the power functions in the package using lookup tables, which are much more efficient than direct evaluations.

The ice code has many exponential and power calculations. Our tests showed that the ice subroutines used 8 times as much CPU as the warm rain microphysics (see Chapter 11 on code performance).

There is one big loop that, by itself, cannot be vectorized on the Cray without including a special compilation option. You need to do '**makearps -ice arps40**', so that file *micro_ice3d.f* is compiled with *-aggressive* option. You may need to delete the old *micro_ice3d.o* (see Chapter 3 on **makearps** options) before re-doing **makearps**.

Q: When I tried to compile and link the plotting program by doing **makearps arpspltncar**, I got a message saying 'zxncarf77 command unknown'.

A: The release of ARPS does not come with the ZXPLOT library used by the plotting program ARPSPLT. Install ZXPLOT library first. See the next question.

Q: How do I install ZXPLOT?

A: The compiled object code of ZXPLOT library for IBM RS/6000, Cray Y-MP. Cray YMP-C90, Sun SparcStation, DEC Alpha running Ultrix or OSF1 Mach UNIX and HP-UX are available in *pub/ZXPLOT* of the CAPS anonymous FTP server *tornado.gcn.uoknor.edu.*

You need to transfer the appropriate *tar* file for your system and place the object codes in one of your permanent directories. You need to edit the link scripts, *zxncarf77*, *zxpost0f77* and *zxpostf77*, so that they point to the location where the object codes reside. You should then make these scripts executable and add the directory name that contains these script files into your command search path. Chapter 12 provides more information on ZXPLOT.

It is possible that the object codes provided are not compatible with the system version you are using. In some cases, we have to gain a temporary access to your system to install the package for you.

For availability of ZXPLOT on other systems, contact *arpsuser@tornado.gcn.uoknor.edu*.

Q: I tried to build the plotting program, *arpspltncar*, but got messages about unresolved externals as follows:

Unresolved:
xctrbadv_
xvtrbadv_
xbadval_

fort: Severe: Failed while trying to link.

I was able to build the executable with an earlier version of the plotting program. What's wrong?

A: Those are three new routines added into ZXPLOT in June 1994. They were not called in some earlier versions of ARPSPLT. You need to update your ZXPLOT library.

Q: I tried to generate HDF format history dumps by specifying the format flag *hdmpfmt* as 3, but got a message saying that the program stopped in HDFDUMP; what was I doing wrong?

A: First, to read or write HDF format data, you need to make sure that the HDF library is installed on your system. HDF is a data format developed at NCSA, the National Center for Supercomputing Application at the University of Illinois. The source code is freely available from *ftp.ncsa.uiuc.edu*.

Secondly, you need to include option *-io hdf* for **makearps**, so that the HDF data I/O routines are properly linked.

The same is true for using NetCDF format.

Q: We ran the Del City storm case and compared the results to those discussed in the ARPS 3.0 Users Guide. We used the same grid setup, and we set all of the model parameters as listed in the manual. However, W_{max} of the storm that we simulated was only 29 m/s after 1 hour, whereas W_{max} of the storm discussed in the manual was around 45 m/s. All of the other features of our simulation agree well with what is described.

A: The version that produced the w plot in the 3.0 Guide was in error. The water loading was effectively turned off during that run, therefore *w* was too large. The results with the current version should be correct.

Q: I am having problems with the terrain files. Is the terrain file called *dir30sec.dat*?

A: There are two steps that you have to go through to generate the final terrain file for ARPS40.

First you need run the three *dir*.f* programs to convert the original ASCII terrain files to direct access files, which are machine dependent. You need to do this only once if you keep these files. The input file for these programs is *arpstern.input.*

The direct access files (each set consists of two files, one for data, the other for record description). are what you need to run ARPSTERN (using *arpstern.input*). The final product, *arpstern.dat,* should be used by ARPS and the name of this file is specified in *arps40.input*. After it is created, the name may be changed to better describe its contents; note that the name is also specified in *arps40.input.*

Section 8.2 provides a more detailed description on these preprocessing steps for terrain data.

Q: We obtained the latest version of ARPS. Does this version have NESTING capability? If so, what information (input parameters) do we specifiy in the input file in order to do a nested simulation. If not, is it possible to send us a version that includes grid nesting?

A: Our nesting capability is built on top of an adaptive grid refinement interface. The interface wraps around the ARPS grid solver and controls the time integration of the entire system.

The interface worked with Version 3.3, but has yet to be updated to work with version 4.0. With the version you have, it's not possible to turn on grid nesting. We will try to update the interface as soon as we can.

Q: What type of graphics package is available to us for animation of model results? We currently are using ZXPLOT or GRADS plotting packages, but unfortunately cannot animate the 2d and 3d model results. Do you use a special package? If so, how can we get it?

A: The latest version of NCAR Graphics metafile displayer **idt** comes with a animation function, that can be used to build an animated sequence of 2-D images. GraDS also has 2-D animation capabilities. If you are only interested in 2-D color raster images, ARPS can produce HDF image file sequences to be played back by NCSA Ximage.

At CAPS, we have a commercial software Savi3D for 3-D visualization and animation. Other visualization packages inlcuding DataExplorer and AVS (Advanced Visualization System) are also available locally.

ARPS was interfaced with a free software Vis5D by the Lawrence Livermore group with little effort. Vis5D is available from anonymous FTP site *iris.ssec.wisc.edu.* If you are interested in 3-D visualization and/or animation, we suggest you consider using Vis5D.

Q: We tried to plot ARPS history data generated on the Cray on our DEC Alpha workstations using ARPSPLT. The program failed when trying to read the data. We guessed that the binary data from Cray are not compatible with DEC Alpha. If that's true, what is the best solution?

A: ARPS generates history dumps in IEEE instead of the Cray native binary format. The IEEE data can be read on the IBM RS/6000, Sun and IRIS but not on a DEC Alpha (as far as we know).

A suggested solution is to use HDF or NetCDF format for the history dumps. To do this, you need to have HDF or NetCDF library installed on both of your machines. The source code of both libraries can be obtained freely, from addresses given in Section 10.1.

Q: I'm running the model in 2-D mode, and am using *nx* = 1003 and *nz* = 45 (along with *ny* = 4). This requires about 13 MW of memory on the Cray, which seems excessive. Some of it is probably due to the way that the y-direction is handled, and some may be due to the fact that it appears that unused arrays (such as those for surface physics) are fully dimensioned even if that package is not used. I suggest putting in a parameter option at compile time that allows the user to completely deactivate modules (terrain, surface physics) that are not needed, so that the storage associated with their arrays can collapse down to a minimum size. (Also, collapse ice storage arrays when ice microphysics is not used). What do you think?

A: The memory usage is about right. ARPS has 66 3-D arrays, which use $66 \times 1003 \times 45 \times 4 \approx 12$ MW in your case. The 2-D surface arrays account for a small amount of memory. *ny* = 4 is the main reason for the "excessive" usage, and this is a penalty we chose to pay in return for maintaining a single 3-D version of the code. You can easily save the storage used by the ice variables by adding an equivalence (*qi, qs, qh*) statement in the declaration portion of the ARPS40 main program. If you are not using *tke*, you can add *tke* to the equivalence list too. As a result, *qi, qs, qh* and *tke* would occupy the same memory space.

In ARPS, 3-D arrays used by the externally-forced boundary condition option can be collapsed to $1 \times 1 \times 1$ arrays. The dimensions are set in *exbc.inc*. ARPS defines 15 3-D work arrays that are reused frequently throughout the code. We tried to strike a balance among the memory and CPU usage and the code readability. We also assume that memory on today's computers is relatively cheap.

Your suggestion on using options at compile to turn off certain packages completely is a very good one. But this typically involves using code preprocessors (conditional compilation is not generally available with Fortran) that will increase the procedure complexity and might affect the code readability. At this time, we chose to handle everything with **make** and the script **makearps**.

Further answers to FAQ will be maintained in a file available by anon ftp. This will be periodically updated and will include errata for the printed User's Guide.