A Quick Guide to Programs in the ARPS System

DRAFT 0.1

First written: 5/30/2002 by Ming Xue to match ARPS Version 5.0.0IHOP_2
Updated on 6/25/2002

Conventions of notation:

Use CAPITAL letters for the names of PROGRAMS, such as ARPS and ADAS.
Use *italic* for *commands* and *file and directory names*, such as *bin/arps < arps.input*.

Program: **ARPS**

Function: ARPS forward prediction model. Sometimes it also refers to the entire ARPS system, including all supporting packages.

Applications: Starting from an initiation condition, it performs forward time integration of the governing equations of the atmosphere and produces a forecast of the future state of the atmosphere.
The initial condition can be specified by using analytical functions with parameters specified in the input file, by using a single sounding, or by reading in ARPS history or restart format data.

Location of source code *src/arps*
src/arps_mp for MPI job

Compilation and Linking: *makearps arps* for a single-processor executable.

makearps –p arps for a shared-memory multi-processor executable. It requires a Fortran 90 compiler capable of automatically parallelizing the source codes. The compile typically perform a preprocessing step that inserts loop-level parallelization directives (OpenMP is most common) into the source code first.

makearps arps_mpi for a distributed memory multi-processor executable. It requires that MPI is set up properly on the system. makearps makes some assumptions in *makearps* the location of the MPI library. Check the correctness of the assumptions for the particular platform you are using.

makearps takes additional option parameters, with *–p* given above as an example. Enter *makearps –help* for additional information on the options.

Execution: *bin/arps < input/arps.input >! arps.output* for non-MPI runs.

To use multiple shared memory processors, certain environmental parameters usually need to be set first, telling the arps job how many processors to use. On SGI system, for example, it is *setenv MP_SET_NUMTHREADS n,* where *n* is the number of processors to use.

To run the MPI version of ARPS on most systems, enter

mpirun –np n_proc bin/arps_mpi < input/arps.input >! arps.output.

On some other systems, *mpirun* command and system can be different. *prun* is another example.

For the MPI run that initializes from an input data set, program *splitfiles* needs to be run first. After *arps_mpi* is run, *joinfiles* program is run to join together files written out by different processors.

Platform supported:

All common Unix platforms.

External package/library required

MPI library is required for distributed-memory parallel runs.

HDF Version 4.0 library required *when –io hdf* option is invoked for makearps. ARPS will then be able to read and write HDF format history dumps and boundary condition files. Option *–io hdf* is currently the default. If *–io nohdf* is used, the link to HDF library is bypassed, as a result, no HDF format output will be produced if HDF dump option is chosen (in *arps.input*). HDF format is becoming the preferred data format for ARPS because it is portable cross platforms and contains two levels to building compression so that the file size is typical ¼ of the size of the binary format. Please note that HDF 5.0 is not compatible with HDF 4.0 so use HDF 4.0 only. HDF library is freely available from NCSA (http://www.ncas.uiuc.edu).

NetCDF – this I/O format is currently unsupported due to lack of use.

Vis5D – history dumps in Vis5D format can be directly written by ARPS by including *–io v5d* option for makearps. It requires a C compiler.

GrADS – ARPS can directly write output in GrADS format without the need for any external library. No additional makearps option is necessary.

GRIB – GRIB format history dump can be written by ARPS without external library. Currently, ARPS GRIB files are not portable across big endian (SGI, IBM, SUN, HP and Cray vector machines) and little endian machines (Alpha and Intel processor machines).

Program:	**ARPSAGR**
Function:	ARPS with adaptive grid refinement driver for adaptive grid two-way interactive nesting.
Applications:	Run ARPS with two-way interactive nesting.
Location of source code:	*src/arpsagr* and *src/arps*
Compilation and Linking:	*makearps arpsagr*
Execution:	*bin/arpsagr* < *input/arps.input* > *arps.output*
Platform supported:	All common Unix platforms. Distributed-memory parallelization is now supported.
External package/library required	Same as ARPS.

Program:	**SPLITFILES**
Function:	Split all binary format input files used by ARPS into individual patches to be read by individual processors by MPI run of ARPS.
Applications:	Data preparation step for *arps_mpi* job.
Location of source code:	*src/arps_mp.*
Compilation and Linking:	*makearps splitfiles*
Execution:	*bin/splitfiles* < *input/arps.input* > *arps.output*
Platform supported:	All common Unix platforms.
External package/library required	None.

Program:	**JOINFILE, JOINFILES**
Function:	Join binary format output files written by individual processors by MPI run of ARPS into single pieces for the entire model domain.

Applications:	Post-processing step for ARPS MPI job *arps_mpi*.
Location of source code:	*src/arps_mp*
Compilation and Linking:	*makearps joinfiles* *makearps joinfile*
Execution:	*bin/joinfiles < input/arps.input > joinfiles.output* *bin/joinfile < ??*
Platform supported:	All common Unix platforms.
External package/library required	None.

Program:	**SPLITHDF**
Function:	Split all HDF format input files used by ARPS into individual patches to be read by individual processors by MPI run of ARPS.
Applications:	Data preparation step for *arps_mpi* job when input data are in HDF format.
Location of source code:	*src/arps_mp.*
Compilation and Linking:	*makearps splithdf*
Execution:	*bin/splithdf < input/arps.input > splithdf.output*
Platform supported:	All common Unix platforms.
External package/library required	HDF 4.0 library.

Program:	**JOINHDF**
Function:	Join HDF format output files written by individual processors by MPI run of ARPS into single pieces for the entire model domain.
Applications:	Post-processing step for ARPS MPI job *arps_mpi* that writes HDF format output.
Location of source code:	*src/arps_mp*

Compilation and Linking:	*makearps joinhdf*
Execution:	*bin/joinhdf* < *input/arps.input* > *joinhdf.output*
Platform supported:	All common Unix platforms.
External package/library required	HDF 4.0 library.

Program:	**JOINBIN2HDF**
Function:	Join binrary format history output files written by individual processors by MPI run of ARPS into single pieces for the entire model domain and write them in HDF format
Applications:	Post-processing step for ARPS MPI job *arps_mpi* that writes HDF format output.
Location of source code:	*src/arps_mp*
Compilation and Linking:	*makearps joinbin2hdf*
Execution:	*bin/joinbin2hdf* < *input/arps.input* > *joinbin2hdf.output*
Platform supported:	All common Unix platforms.
External package/library required	HDF 4.0 library.

Program:	**ADAS**
Function:	ARPS Data Analysis System – a system that analyzes observational data onto the ARPS grid.
Applications:	The 3D analysis can be used for diagnostic studies, model initialization and for providing model boundary conditions in the case of simulation (not true forecast). The output data is in ARPS history format which can be directly read by ARPS and many other ARPS programs including ARPSPLT.

Location of source code

src/adas

Some include files in *include* directory and some ARPS subroutines in *src/arps* are also used. This is similar for many other programs in ARPS package.

Compilation and Linking:

makearps adas for a single-processor executable.

makearps –p adas for a shared-memory multi-processor executable.

Execution:

bin/adas < input/arps.input > adas.output

ADAS shares an input file with ARPS.

Platform supported:

All common Unix platforms.

External package/library required

Necessary libraries, such as HDF, for data I/O.

Program: **ARPSTRN**

Function:

Prepare a terrain data file on the ARPS grid for use by ARPS and other programs that need to set up the ARPS grid.

Applications:

It reads in one of several terrain data sets and interpolates the data to the ARPS grid. Smoothing is optionally applied to the interpolated terrain field to remove 2 grid-spacing features. The field is written out into a file.

Location of source code

src/arpstrn

Compilation and Linking:

makearps arpstrn Compile without linking to ZXPLOT graphics library. No graphic output will be generated by the program.

makearps –zxncar arpstrn Compile and link with ZXPLOT and NCAR graphics program to produce at the same time a color contour plot of the terrain field in meta file format.

makearps –zxpost arpstrn Compile and link with ZXPLOT graphics program to produce at the same time a color contour plot of the terrain field in Postscript format.

Execution:

bin/arpstrn < input/arpstrn.input >! arpstrn.output

Currently 5min and 30 second global, and 3-second North American (covers continental US and Alaska**)** data sets are supported. When using 3 and 30-second data sets, the data will be directly downloaded from the USGS ftp server (ftp://edcftp.cr.usgs.gov) and the CAPS ftp server (ftp.caps.ou.edu), respectively. Make sure your computer have FTP access to the internet and network speed is reasonably fast, and you need to set up a *.netrc* file in your home directory for the ftp to work. See *arpstrn.input* for more details.

Platform supported:	All common Unix platforms.
External package/library required	ZXPLOT and NCAR graphics libraries when *–zxncar* option is included for *makearps*.

Program:	**MERGETRN**
Function:	To create a terrain file with gradual transition to another terrain field in a boundary zone. It reads in two ARPS terrain data files, interpolates the first terrain field to the grid of the second field when necessary, and generates a new terrain field that is same as the second one except in the lateral boundary zone of specified width, where the terrain transitions from that of the second file at the interior of the boundary to that of the first one at the domain boundary.
Applications:	Typically used to ensure that the high-resolution terrain generated for nested grid matches that of the coarse grid at the nesting boundary and that the transition from coarse grid to fine grid is gradual. This function can also be realized in EXT2ARPS and ARPSINTRP programs, in which the fine-grid terrain is merged with that of the coarse grid before interpolation.
Location of source code	*src/arpstrn*
Compilation and Linking:	*makearps mergetrn*
Execution:	*bin/mergetrn.input < input/mergetrn.input >! mergetrn.output*
Platform supported:	All common Unix platforms.

External package/library required	ZXPLOT library. Other libraries needed for the choice of data I/O format.

Program:	**EXT2ARPS**
Function:	Read in gridded data from external models and interpolate the fields onto the ARPS grid. Write out the fields on the ARPS grid in a standard ARPS history dump format and/or the external boundary condition format.
Applications:	Used to provide analysis background and/or forecast lateral boundary conditions if the ARPS analysis background is from an external model and/or the ARPS is nested inside an external model. Currently EXT2ARPS handles NCEP ETA, RUC, AVN gridded data in various formats and coordinates. NCAR/NCEP global reanalysis data is also loosely supported.
Location of source code	*src/ext2arps*
Compilation and Linking:	*makearps ext2arps*
Execution:	*bin/ext2arps < input/arps.input > ext2arps.output*
Platform supported:	All common Unix platforms.
External package/library required	Libraries needed by specific choice of history I/O format.

Program:	**ARPSINTRP**
Function:	Read in ARPS gridded data set(s) in history format, and interpolate the fields to another (output) ARPS grid, and write them out in one of the history dump formats for this output grid.
Applications:	Mostly used for generating initial (or background for initial condition analysis) and boundary condition files for one-way nested grid runs inside a coarser ARPS grid. In this case, the new output grid has a higher spatial resolution. The output grid should be no bigger than the input grid.

The input and output grids have to have the same map projections, but do not have to have the same vertical coordinates. The fine grid can use higher-resolution terrain or flat terrain. The latter situation is useful when one wants to examine the fields on the ARPS terrain-following grid using software, such as GrADS and Vis5D, that does not support non-rectangular grid. For such a purpose, it is best to choose the option that directly extends surface values below ground level.

This program can also be used to sub-sample the ARPS output on a coarser resolution and/or smaller grid for easier post-processing, especially when the original grid is very large.

When running ARPSINTRP, an terrain merge option is available, with which the terrain of the output (typically finer resolution) grid is 'merged' with that of the input (typically coarser resolution) grid, in a way similar to what is done in MERGETRN.

Location of source code	*src/arpsintrp*
Compilation and Linking:	*makearps arpsintrp*
Execution:	*bin/arpsintrp < arpsintrp.input >! arpsintrp.output*
Platform supported:	All common Unix platforms.
External package/library required	Libraries needed for reading and/or writing special format history files.

Program:	**ARPSTINTRP**
Function:	Read in ARPS history format data at two different times and interpolates them to a time between the two. The output is written out into another history format file. It is assumed that both input data are on the same grid.
Applications:	Mostly used to provide a background field for analysis at times when output used for the analysis background (e.g., ETA model output) is not available.
Location of source code	*src/arpstintrp*
Compilation and Linking:	*makearps arpstintrp*

Execution:	*bin/arpstrinp < input/arpstintrp.input >! arpstintrp.output*
Platform supported:	All common Unix platforms.
External package/library required	Libraries required by the specific history data format.

Program:	**ARPSPLT**
Function:	Vector-based plotting program for processing ARPS history-format data.
Applications:	Generates contour and vector plots of 2D cross sections and vertical profiles. The graphical output is either in NCAR graphics meta file format or Postscript format.
Location of source code	*src/arpsplt*
Compilation and Linking:	*makearps arpspltncar* links with ZXPLOT library and NCAR graphics low-level routines to generate NCAR graphics metafile file. *makearps arpspltpost* links with ZXPLOT library to generate –*p* option can be included for the executable to run on shared-memory multi-processors. This program does not support distributed-memory multi-processing, yet.
Execution:	*bin/arpspltncar < input/arpsplt.input >! arpsplt.output* *bin/arpspltpost < input/arpsplt.input >! arpsplt.output* Necessary –io option should be included when processing special format (e.g., -io hdf) history data.
Platform supported:	All common Unix platforms.
External package/library required	ZXPLOT graphics library (see http://www.caps.ou.edu/ZXPLOT). The object code libraries for common Unix platforms are available at ftp://ftp.caps.ou.edu/pub/zxplot3. NCAR graphics library with *arpspltncar*. NCAR Graphics is freely

available from http://ngwww.ucar.edu/ngdoc/ng/.

Other libraries needed for specific history dump format.

Program:	**PLTGRID**
Function:	Plot a grid map and nested grid boxes given the grid configuration parameters, including the central longitude and latitude, map projection and grid sizes.
Applications:	To help configure the model domains or a quick look at the model grid given the configuration parameters.
Location of source code	*src/arpsplt*
Compilation and Linking:	*makearps –zxncar pltgrid* or *makearps –zxpost pltgrid*
Execution:	*bin/pltgrid < input/pltgrid.input*
Platform supported:	All common Unix platforms.
External package/library required	Need ZXPLOT graphics library with *–zxpost* option and both ZXPLOT and NCAR graphics libraries with *–zxncar* option.

Program:	**ARPSDIFF**
Function:	Reads in two sets of ARPS history format data, calculate the difference fields and write out the difference fields into a file in the ARPS history dump format. The two data sets can be on different grids. The 'verification' grid data are interpolated to the 'forecast' grid first before the differences are calculated and the output will be on the forecast grid. If the two grids are the same, no interpolation will be performed.
Applications:	For comparing two sets of ARPS history format data or for 'verifying' one set of ARPS fields against the other (say analysis). It can be used to find the difference, if any, between the outputs of two ARPS runs.

Location of source code	*src/arpsdiff*
Compilation and Linking:	*makearps arpsdiff*
Execution:	*bin/arpsdiff < input/arpsdiff.input > arpsdiff.output*
Platform supported:	All common Unix platforms.
External package/library required	Libraries required by the specific history data format.

Program:	**ARPSSFC**
Function:	Read in soil type, vegetation type and vegetation fraction data files and construct a set of surface and vegetation characteristics fields for the ARPS grid. ARPSSFC and ARPS soil models supports up to 4 different soil types in each grid cell, each caries its own percentage.
Applications:	Prepare land use/land cover (or surface and vegetation) characteristics data file to be used by ARPS soil model.
Location of source code	*src/arpssfc*
Compilation and Linking:	*makearps arpssfc* *makearps –ncarg arpssfc*
Execution:	*bin/arpssfc < input/arpssfc.input > arpssfc.output*
Platform supported:	All common Unix platforms.
External package/library required	NCAR graphics library when *–ncarg* option is included with *makearps*.

Program:	**ARPSSOIL**
Function:	Read in ARPS initial condition data (in history format) and, for the API case, the precipitation data for a period proceeding the initial time in the API case, and creates and writes out an initial condition file for ARPS soil model.

Applications:	Prepare an initial condition file for the ARPS soil model, when such initial conditions are based on offsets from surface atmospheric conditions and the soil moisture content can be derived using API (antecedent precipitation index) method. This program may not be needed when the soil model is initialized by other means, such as interpolating from the soil model state of another model. The soil model variables can be carried in the history file.
Location of source code	*src/arpssoil*
Compilation and Linking:	*makearps arpssoil*
Execution:	*bin/arpssoil < input/arpssoil.input >! arpssoil.input*
Platform supported:	All common Unix platforms.
External package/library required	Libraries required by your specific choice of history data format.

Program:	**ARPSCVT**
Function:	Convert ARPS history dumps among ARPS supported formats.
Applications:	This program can be used to convert the history dump from ARPS into other formats for various purposes. E.g., it can convert a set of history dumps into one Vis5D file for visualization, or Grads format for display. While ARPS can output data in Vis5D or GrADS format directly but it is not recommended because most other programs expect data at individual times.
Location of source code	*src/arpscvt*
Compilation and Linking:	*makearps –io io_options arpscvt* *io_options* can be *hdf* and *v5d*.
Execution:	*bin/arpscvt < arpscvt.input >! arpscvt.output*
Platform supported:	All common Unix platforms.

External package/library required	Libraries required by your specific choice of history data format. Need to include *–io v5d* option for *makearps* in order to write Vis5D format data.
Program:	**ARPSEXTSND**
Function:	Extract columns (profiles) from ARPS history dumps, and write out these profiles in a text format.
Applications:	The program can be used to extract profiles for plotting skew-T diagrams using program SKEWT.
Location of source code	*src/arpsextsnd*
Compilation and Linking:	*makearps arpsextsnd*
Execution:	
Platform supported:	All common Unix platforms.
External package/library required	Libraries required by your specific choice of history data format.
Program:	**ARPSENSIC, ARPSENSBC**
Function:	Generate perturbed initial/boundary conditions for ARPS ensemble forecast, using the SLAF (Scaled Lagged Average Forecast) or the BGM (Breeding Fast Growing Mode) method. It reads in three sets of data files in ARPS history format and derive perturbations from the first two and add/subtract the perturbation to the third to generate perturbed initial conditions/boundary conditions.
Applications:	For creating initial and boundary conditions for ARPS ensemble members.
Location of source code	*src/arpsens*
Compilation and Linking:	*makearps arpsensic* *makearps arpsensbc*
Execution:	*bin/arpsensinc < input/arpsensic.input > arpsensbc.output* *bin/arpsensnc < input/arpsensbc.input > arpsensbc.output*

Platform supported:	All common Unix platforms.
External package/library required	Libraries needed by specific choice of history I/O format.

Program:	**ARPSENSCV**
Function:	Reads in a series of ARPS history dumps and generate ensemble forecast products, and write the 2D fields out for plotting.
Applications:	Used to process ensemble forecast output.
Location of source code	*src/arpsens*
Compilation and Linking:	*makearps arpsenscv*
Execution:	*bin/arpsenscv* < *input/arpsencv.input* > *arpsenscv.output*
Platform supported:	All common Unix platforms.
External package/library required	Libraries needed by specific choice of history I/O format.

Program:	**ARPSVERIF**
Function:	Verifies ARPS forecast against point observations and gridded fields.
Applications:	Used for objective forecast verification.
Location of source code	*src/arpsverif*
Compilation and Linking:	
Execution:	

Platform supported:	All common Unix platforms.
External package/library required	

Program:	**F77TOF90**
Function:	Convert Fortran 77 code into the Fortran 90 free form and stylize the code to conform to the ARPS Fortran 90 coding standard.
Applications:	Used to convert F77 ARPS 4.5.x into F90 ARPS 5.0.
Location of source code	*src/f77tof90*
Compilation and Linking:	*f90 –o bin/f77tof90 src/f77tof90*
Execution:	/bin/ls *.f > f77_filelist ; bin/f77tof90 < f77_filelist
Platform supported:	All common Unix platforms.
External package/library required	None

Program:	**88D2ARPS**
Function:	Read in WSR-88D Level-II radar data and remap (interpolate) to the ARPS grid, and write out data in a format usable by ADAS
Applications:	Used to prepare Level-II radar data for ADAS.
Location of source code	*src/88d2arps*
Compilation and Linking:	*makearps 88d2arps_a2*
Execution:	*bin/88d2arps –options*
Platform supported:	Tested on Sun, SGI and Linux.

External package/library required	Requires a2io and tpio libraries which are currently not freely distributed.

Program:	**NIDS2ARPS**
Function:	Reads WSR-88D Level-III (NIDS) data and remap them to the ARPS grid for ingest into ADAS.
Applications:	Used to prepare Level-III (NIDS) radar data for ADAS.
Location of source code:	*src/88d2arps*
Compilation and Linking:	*makearps nids2arps*
Execution:	*nids2arps -options*
Platform supported:	All common Unix platforms.
External package/library required	None.

Program:	**ARPSREAD**
Function:	A template program for reading ARPS history format data.
Applications:	An example for reading ARPS history format data.
Location of source code:	*src/arps*
Compilation and Linking:	*makearps arpsread*
Execution:	*bin/arpsread < input/arpsread.input > arpsread.output*
Platform supported:	All common Unix platforms.
External package/library required	Libraries needed by specific choice of history I/O format.

Program:	**SKEWT**
Function:	
Applications:	
Location of source code:	*src/skewt*
Compilation and Linking:	
Execution:	
Platform supported:	Unix platforms that supports command line argument input in Fortran.
External package/library required	

Program:	**ARPSASSIM**
Function:	ARPS Themodynamic retrieval package.
Applications:	For retrieval temperature and pressure fields when wind fields and their time tendencies are known. The latter are usually obtained from single or dual Doppler radar retrieval packages.
Location of source code:	*src/arpsassim*
Compilation and Linking:	*makearps arpsassim*
Execution:	
Platform supported:	All common Unix platforms.
External package/library required	Data I/O libraries based on options chosen.

Program:	**A3DVAR**

Function:	Perform objective analysis using three-dimensional variational analysis scheme (under development).
Applications:	To obtain a best estimate of the current state of the atmospheric to serve as the initial condition for forward prediction.
Location of source code:	*src/a3dvar*
Compilation and Linking:	*makearps a3dvar*
Execution:	*bin/a3dvar < input/a3dvar.input > a3dvar.output*
Platform supported:	All common Unix platforms. Distributed-memory parallelization is currently not yet supported.
External package/library required	Necessary I/O libraries.

Program:	**ARPSADJ**
Function:	
Applications:	
Location of source code	
Compilation and Linking:	
Execution:	
Platform supported:	All common Unix platforms.
External package/library required	

Program:	**MCI2ARPS**
Function:	Reads GOES satellite data in GEMPAK format, remap the data to the ARPS grid to be ingested by ADAS.
Applications:	Prepare data for use by cloud analysis procedure in ADAS.

Location of source code:	*src/mci2arps*
Compilation and Linking:	*makearps mci2arps*
Execution:	*mci2arps ????*
Platform supported:	All common Unix platforms. ???
External package/library required	

Program:	**TESTMKARPS**
Function:	Test the compilation and link of most ARPS programs using *makearps*.
Applications:	To make sure that programs in the ARPS package compile on various platforms.
Location of source code:	*scripts*
Compilation and Linking:	Direct execution of the scripts.
Execution:	*scripts/testmkarps[–opts_makearps "makearps_options"]* The options used by *makearps* can be included as "*makearps_options*".
Platform supported:	All common Unix platforms.
External package/library required	Depending on the program being tests.

Program:	**Standard test scripts in *Perl***
Function:	Run the standard ARPS test suite
Applications:	Test variously ARPS programs using automated procedures
Location of source code:	*scripts*

Compilation and Linking:	Direct execution of the scripts.
Execution:	*scripts/testarps.pl -all* Run all tests. Individual tests can be run with appropriate options. See inside the script.
Platform supported:	All common Unix platforms.
External package/library required	Depending on the tests

Program:	
Function:	
Applications:	
Location of source code:	
Compilation and Linking:	
Execution:	
Platform supported:	All common Unix platforms.
External package/library required	