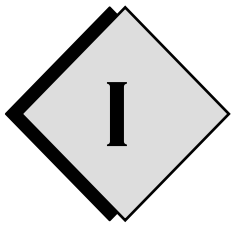
# ARPS2WRF User's Guide

## *CONTENTS*



**Website of this document**:

http://www.caps.ou.edu/ARPS/ARPS5DOC/arps2wrf.pdf



# INTRODUCTION

This guide is written specifically for the programs developed at CAPS recently, which link the ARPS system and the WRF system. It first introduces the two programs ARPS2WRF and WRFSTATIC in section I. Then instructions are presented in section II for helps on setting the control parameters in the NAMELIST file read by these two programs. The steps to run typical real-data cases are outlined in the last section.

ARPS2WRF is a program which ingests data files in ARPS history format and generates WRF input file and lateral boundary file. The generated files are those required to run a WRF real-data simulation. Currently, only the Advanced Research WRF (ARW) core in mass vertical coordinate is supported. The ARPS2WRF program can generate both the WRF initial condition and the lateral boundary conditions from existing ARPS data files, as well the Fortran NAMELIST file for controlling the WRF runs.

ARPS2WRF will accept, as input, ARPS data files from version ARPS5.0.0IHOP_3 and later. These data can be outputs from any programs inside the ARPS package provided that the files are written in the ARPS history format. However, because of a conflict between the HDF 4 library with the netCDF 3.x library, ARPS2WRF cannot read ARPS history files written in the HDF 4 format. At present, the supported ARPS file formats are unformatted binary and netCDF 3.x.

WRFSTATIC is a program to prepare static data for the WRF runs. It calls WRFSI subroutines to process static data sets, such as terrain height, vegetation fraction, soil texture, landuse categories, snow albedo etc. However, the projection grid is built inside the ARPS framework to ensure consistency with the ARPS2WRF processing.

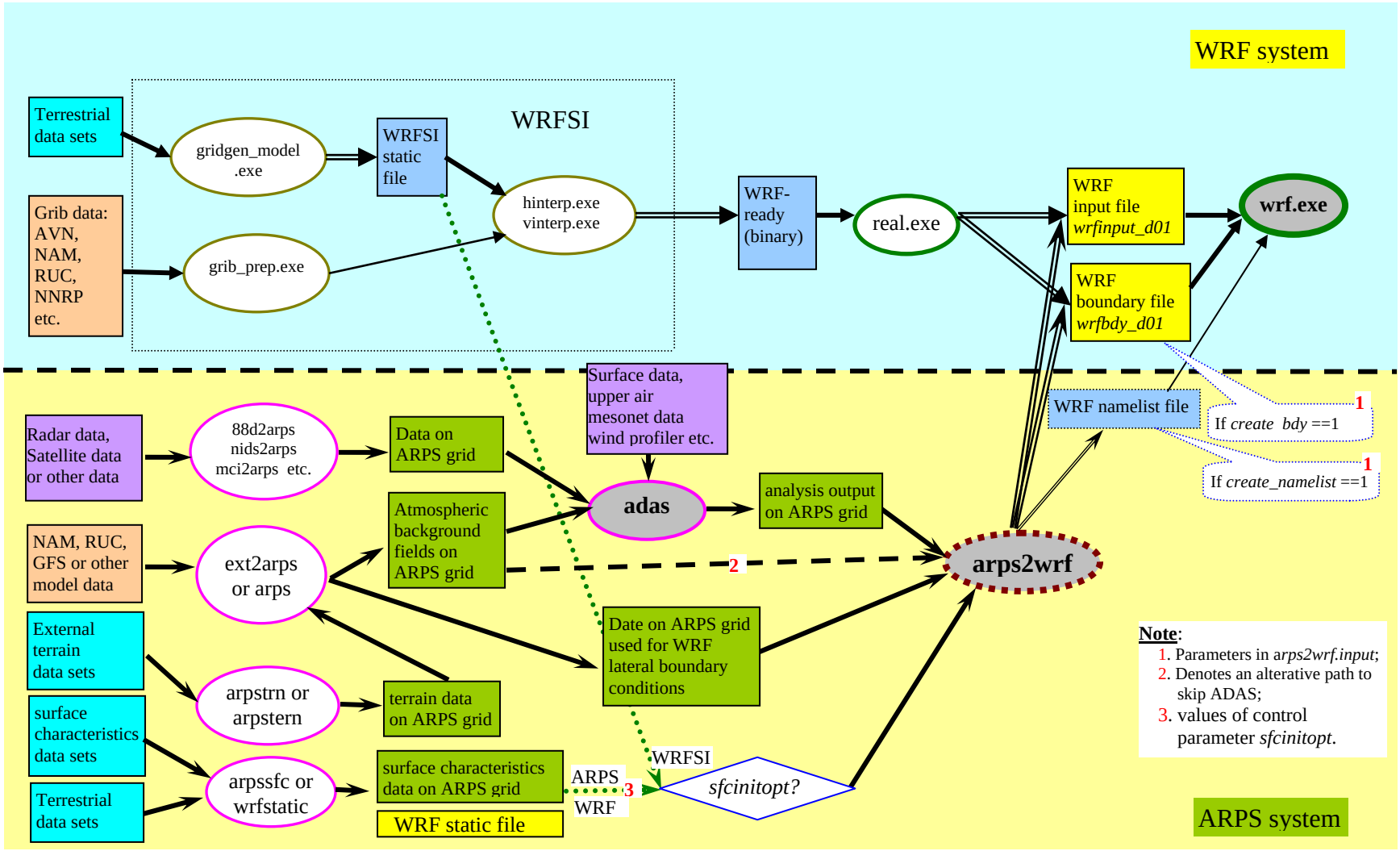



**Figure 1. WRFSI/WRF and ARPS2WRF flow chart**

The ARPS processing (EXT2ARPS, ADAS, WRFSTATIC and ARPSWRF *etc.*) is used to replace both the WRFSI and *real.exe* steps when processing the ARPS model data. The interconnection of these processes is illustrated using a flow chart, as given in **Figure 1**.

## 1.1 History

ARPS2WRF is available as another program within the ARPS package starting with version ARPS5.0.0IHOP_6. ARPS2WRF has gone through several updates since its first release. It has been upgraded recently to work with the latest official WRF release, *i.e.* WRF version 2.1. It also has been rewritten using Message Passing Interface (MPI) to run on multiple processors and it supports several WRF data formats. The supported WRF data formats are netCDF 3.x, PHDF5 and WRF native binary format. A brief history of ARPS2WRF is given in **Table 1**. below.

**Table 1. ARPS2WRF milestones**

| ARPS version | Release date | WRF version | New Capabilities |
|---|---|---|---|
| ARPS5.0.0IHOP_6 | 09/15/2003 | WRFV1.3 | Initial release<br>WRF netCDF file support |
| ARPS5.1.0 | 02/24/2004 | WRFV1.3 | Merged with WDT |
| ARPS5.1.1 | 04/10/2004 | WRFV1.4 | Changed I/O for better performance |
| ARPS5.1.5 | 10/12/2004 | WRF2.0.2 | Added MPI support |
| ARPS5.1.6 | 02/08/2005 | WRF2.0.3 | PHDF5 support for *arps2wrf_mpi*<br>WRF native binary support |
| ARPS5.2.3 | 09/16/2005 | WRFV2.1 | I/O reformatting for efficiency<br>Added support for WRF static file<br>Added selecting of terrain height source |

WRFSTATIC was added since ARPS5.2.3 for user's convenience. ***wrfstatic*** is used to replace both programs *gridgen_model.exe* and *staticpost* in the WRFSI package and it saves us from going through the complex steps to set up and run WRFSI.

## 1.2 Manifest

The release of ARPS2WRF inside the ARPS package includes several files, which are distributed into the following subdirectories.

**Source code:**

Source codes for ARPS2WRF are mainly written in Fortran 90 and they are located in a subdirectory relative to the ARPS root directory as *src/arps2wrf/*. The files and their functions are summarized as below.

| | |
|---|---|
| *Makefile* | [Makefile for the **make** utility] |
| *arps2wrf.f90* | [The main program in Fortran 90 free format] |
| *dump_wrf_bdy.f90* | [Upper level subroutine writing WRF lateral boundary file] |
| *dump_wrf_input.f90* | [Upper level subroutines that define and write WRF input file and NAMELIST file] |
| *dump_wrf_static.f90* | [Upper level subroutines that define and write WRF static file] |
| *interplib.f90* | [Interpolation library] |
| *module_wrf_metadata.f90* | [Module that contains WRF metadata and constants] |
| *readnamelit.f90* | [Subroutine to read and process NAMELIST file] |
| *wrf_iolib.f90* | [I/O library reads and writes WRF data files. Subroutines in this file will call each low level I/O subroutine according to WRF I/O format] |
| *wrf_ioncd.f90* | [Low level I/O subroutines for netCDF format] |
| *wrf_ionopdf5.f90* | [Contains dummy subroutines when PHDF5 library is not linked] |
| *wrf_iophdf5.f90* | [Contains wrapper subroutines to call WRF external PHDF5 I/O package] |
| *wrf_mpsubs.f90* | [MPI subroutines for ***arps2wrf_mpi***] |
| *wrf_nompsubs.f90* | [Contains dummy subroutines for non-MPI mode, *e.g*. program ***arps2wrf*** or ***wrfstatic***] |
| *wrfsi_cio.c* | [WRFSI C function to read static data sets, it is a copy from file *src/lib/cio.c* in WRFSI package] |
| *wrfsi_subs.f* | [Copies of several WRFSI subroutines to read and process static data sets] |
| *wrfstatic.f90* | [Main program for ***wrfstatic***] |
| *wrfstaticlib.f90* | [Specific library defined for program ***wrfstatic***] |

*wrf_subs.F90* [General WRF subroutines used by the WRF external I/O packages. It is also the only file that needs to do pre-processing using utility *cpp*.]

**Input file:**

Just as all other ARPS programs, ***arps2wrf*** and ***wrfstatic*** both need to read a NAMELIST input file. A template of such file is proved in directory *input/* as

*arps2wrf.input* [***arps2wrf*** and ***wrfstatic*** shared namelist input file]

**Document:**

A user's guide for ***arps2wrf*** and ***wrfstatic*** (this document) is provided in ARPS documentation directory *docs/* as:

*arps2wrf.pdf* [This document in PDF format.]

**Script:**

*scripts/test_arps2wrf.pl* [Perl script to perform tests using an example case]

To support WRF PHDF5 and native binary data format, this program calls the WRF external I/O package directly. For convenience, those packages from WRFV2.1 are copied into directory *src/external/io_phdf5/* and *src/external/io_int/* respectively. The files inside the ARPS package contain minor changes to work seamlessly within the ARPS system. These two packages are also shared with program ***wrf2arps*** in *src/wrf2arps/*.

## 1.3 Ingested data sources

The files required by program ***arpswrf*** are:

- **Namelist file** (*e.g. arps2wrf.input*);
- **Initial field in ARPS History format**: This can be any files generated using program ADAS, ARPS, ARPSINTRP, EXT2ARPS etc. as long as the data files are in ARPS history data format. They should include one perturbation file (i.e., *runname.FMTxxxxxx*) and one grid and base state file (i.e., *runname.FMTgrdbas*), where "xxxxxxx" represents ARPS model forecast time (in seconds), such as

"000000", "003600", "007200" etc. And "*FMT*" is ARPS data format string, it is "*bin*" for binary format and "*net*" for netCDF format.

- **ARPS History Files at other times:** These files provide lateral boundary conditions for the WRF model. They are only needed when "*create_bdy* = 1" in the namelist file (*arps2wrf.input*). These files may be produced by ARPS or EXT2ARPS (if from another forecast model).

- **Surface Characteristics Data:** Surface characteristics data can be initialized by either the static file created using "*gridgen_model.exe*" from WRFSI package or the surface data created by ***arpssfc*** or ***wrfstatic***. The file generated by program ***wrfstatic*** is recommended. This file should contain surface characteristics for the WRF model including soil types, vegetation types and vegetation fraction etc. If using the static file from WRFSI, the file must be generated on the same grid as that specified in the namelist block *&wrf_grid* (see below). For the case of using output from ARPSSFC, the data must be defined on the same grid as that in both the ARPS history file at initial time and the ARPS history files at other times mentioned above.

***wrfstatic*** reads the same static data sets as WRFSI does and they are available from the WRF ARW User's Page (http://www.mmm.ucar.edu/wrf/users/, "Download" > "WRF V2"). Users should pre-download those data sets into their local working directory before running program **wrfstatic**. The required data sets are summarized as the followings.

- 30" global elevation data
- 30" global USGS landuse data
- 30" global top layer soil type data
- 30" global bottom layer soil type data
- 10' global vegetation fraction data
- 1° global deep soil temperature data
- NCEP 10' monthly albedo
- NCEP 10' maximum snow albedo
- 1° slope data

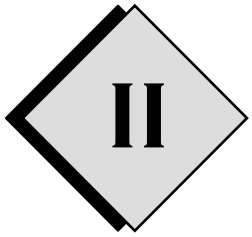
## 1.4 Outputs from the programs

Program ***arps2wrf*** will generate the following files based on the control parameters specified in *arps2wrf.input*.

- *wrfinput_d01* – **WRF input file** (always generated);
- *wrfbdy_d01* – **WRF boundary file** (generated if *create_bdy* = 1);
- *namelist.input* – **WRF namelist file** (generated if *create_namelist* = 1).

Files *wrfinput_d01* and *wrfbdy_d01* can be in netCDF format, WRF native binary format or PHDF5 format according to the output format parameters specified in *arps2wrf.input*.

Program ***wrfstatic*** will generate **WRF static file**, *wrfstatic_d01*. However, only netCDF format is supported for this file at present.




# Runtime/Configuration Reference Guide

ARPS2WRF and WRFSTATIC provide their users with flexible controls over many parameters that can be used to configure the programs for various situations. These parameters can be set at run time without modifying or recompiling the model code. All parameters are specified in an input file, *arps2wrf.input*. The input file is in Fortran NAMELIST format and is read in via standard input (Fortran I/O channel 5 or *). This section provides a reference guide on how to choose appropriate values for these control parameters.

The control parameters are organized into groups that are consistent with the NAMELIST blocks in the input files. The names of these blocks are as followings.

- ***message_passing*** [specifies parameters when running on multiple processors]
- ***history_data*** [specifies the ARPS data file for WRF initial condition]
- ***sfcdt*** [specifies the surface data source]
- ***bdyspec*** [specifies the ARPS data files for WRF boundary conditions]
- ***wrf_grid*** [defines both the WRF horizontal and vertical grids]
- ***wrfopts*** [specifies options to run WRF model]
- ***interp_options*** [specifies interpolation options]
- ***output*** [specifies output options]

## 2.1 Parameters Used by Message Passing (MPI) run (&*message_passing*)

Parameters when running on multiple processors
(&**message_passing**)

| Parameter | Definition/Purpose | Options/Suggested Values |
|---|---|---|
| ***nproc_x*** | Processor numbers in *x*-direction | *nproc_x* must be a divisor of (*nx_arps*-3). |
| ***nproc_y*** | Processor numbers in *y*-dirction | *nproc_y* must be a divisor of (*ny_arps*-3). |
| ***readsplit*** | Flag to indicate whether the program needs to split the data files | = 1, Data for all processors is in one joined file;<br>= 0, Data are all split. |
| ***nprocx_in/nprocy_in*** | Processor numbers when the data to be read was written. (Used only when *readsplit* = 0). | They must be multiples of *nproc_x* & *nproc_y* respectively. |

## 2.2 Parameters for WRF input file (&*history_data*)

ARPS data file for WRF initial conditions
(**&history_data**)

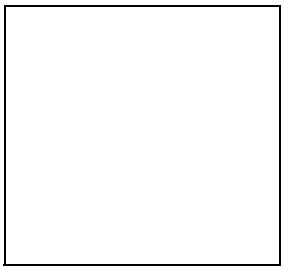
| Parameter | Definition/Purpose | Options/Suggested Values |
|---|---|---|
| ***hinfmt*** | Input data file format | = 1, for binary<br>= 7, for netCDF |
| ***grdbasfn*** | grid and base file name | Directory + filename up to 128 characters |
| ***hisfile(1)*** | File name containing time dependent variables. It is used to generate WRF initial conditions, file *wrfinput_d01*. | Directory + filename up to 128 characters |
| ***hdmpinopt*** | DO NOT change. It is listed here for compatibility with other ARPS programs. | 2 |

## 2.3 Parameters for Surface Characteristics Data Source (&*sfcdt*)

| Surface characteristics data source (**&sfcdt**) | | |
|---|---|---|
| **Parameter** | **Definition/Purpose** | **Options/Suggested Values** |
| ***sfcinitopt*** | Surface characteristics data source | 'WRFSI' = Use WRFSI static file<br>'ARPS' = Use ***arpssfc*** output<br>'WRF' = Use ***wrfstatic*** output |
| ***sfcdtfl*** | Surface data file name | Up to 128 characters, *e.g.* '*wrfstatic_d01*'. |
| ***sfcfmt*** | Surface data format.<br>(Used only when *sfcinitopt* = 'ARPS') | = 1, for binary<br>= 7, for netCDF |
| ***ternopt*** | Option for WRF terrain data | 0 = Use terrain height from the original ARPS grid;<br>1 = Use terrain data from static file (valid only when *sfinitopt* = 'WRF' or 'WRFSI'). |
| ***static_dir*** | Directory name for the downloaded static data sets.<br>(Use only by program ***wrfstatic***) | Character string up to 128 characters, *e.g.* '*/home/xxxx/data/GEOG/*' |
| ***silawt_parm_wrf*** | Control the type of processing of topographic data. See *src/grid/README_toptwvl_silavwt* in WRFSI package. (Used only by program ***wrfstatic***) | Possible values are 0, 1, 2, 3<br>0 is default. |
| ***toptwvl_parm_wrf*** | The wavelength of topographic interpolation in grid-cell size units. (see *silawt_parm_wrf* above, used only by program ***wrfstatic***) | 2.0 (Default) |
| ***start_date*** | Date and time string for WRF static file if *use_arps_grid* = 0. (Used only by program ***wrfstatic***) . | **Example**:<br><br>'*1998-05-25_00:00:00*'. |

## 2.4 Parameters for WRF Boundary Conditions (&*bdyspec*)

| ARPS data files for WRF boundary conditions (**&bdyspec**) | | |
|---|---|---|
| **Parameter** | **Definition/Purpose** | **Options/Suggested Values** |
| ***create_bdy*** | Flag to denote whether WRF boundary file should be generated. | = 0, do not create lateral boundary file, all other parameters in this block will be ignored;<br>= 1, Create lateral boundary files for the WRF model. |
| ***bdyfheader*** | Prefix header string for ARPS history files to be processed for WRF boundary. It corresponds to *runname* preceded by directory name. | **Example:**<br>'./eta25may1998' |
| ***tbgn_bdyin*** | ARPS model time at which input boundary data begins (in seconds). Since data at initial time will be provided by parameter *hisfile(1)* above, *tbgn_bdyin = tintv_bdyin* + data time in file *hisfile(1)*. | **Example:**<br>10800.0 |
| ***tintv_bdyin*** | Time interval in seconds between boundary data dumps | **Example:** 10800.0 |
| ***tend_bdyin*** | Time at which input boundary data ends | **Example:** 21600.0 |
| ***mgrdbas*** | Grid base file option | = 0, share the same grid and base file as ARPS initial fields, *i.e.* "*grdbasfn*" (default);<br>= 1, All of the files share one grid and base file but it is not *grdbasfn*, the file name will be "bdyfheader.*bingrdbas*" or "bdyfheader.*netgrdbas*". |



Those files are usually generated using ARPS;
= 2, Each file at distinct time levels has its own grid and base file. Those files are generated using EXT2ARPS.

**Note:**

The ARPS file names used for WRF lateral boundary conditions will be generated inside the program automatically based on the parameters specified in this block and in namelist block *&history_data* above. These files must be in ARPS history format instead of the ARPS boundary file format and they can be outputs from either program ***ext2arps*** or program ***arps***.

**Example:**

Let, *bdyfheader* = './eta25may1998',

*tbgn_bdyin* = 10800.0,

*tintv_bdyin* = 10800.0,

*tend_bdyin* = 21600.0,

Then, ARPS2WRF will look for the following file in the current directory:

If *mgrdbas* is 2:

./*eta25may1998*.FMT*010800* ./*eta25may1998*.FMT*grdbas.01*

./*eta25may1998*.FMT*021600* ./*eta25may1998*.FMT*grdbas.02*

If *mgrdbas* is 1:

./*eta25may1998*.FMT*010800* ./*eta25may1998*.FMT*grdbas*

./*eta25may1998*.FMT*021600*

If *mgrdbas* is 0 and *grdbasfn* = './*adas25may1998*.FMT*grdbas*':

./*eta25may1998*.FMT*010800* ./*adas25may1998*.FMT*grdbas*

./*eta25may1998*.FMT*021600*

where 'FMT' is a three-letter format string, and it is "bin" for *hinfmt* = 1, "net" for *hinfmt* = 7.

Theoretically, no matter what value of *mgrdbas* is (0, 1 or 2), users will always get the same results because ARPS time dependent files save total variables instead of perturbations from ARPS version 5.0 and later. However, there may be minor difference because of extra computations involved in the ARPS I/O process. *mgrdbas* = 0 is recommended because it read the grid and base file only once.

The relationship of the time sequence in ARPS initial file (*hisfile(1)*) and in ARPS files that provide boundary conditions is illustrated in **Figure 2** below.

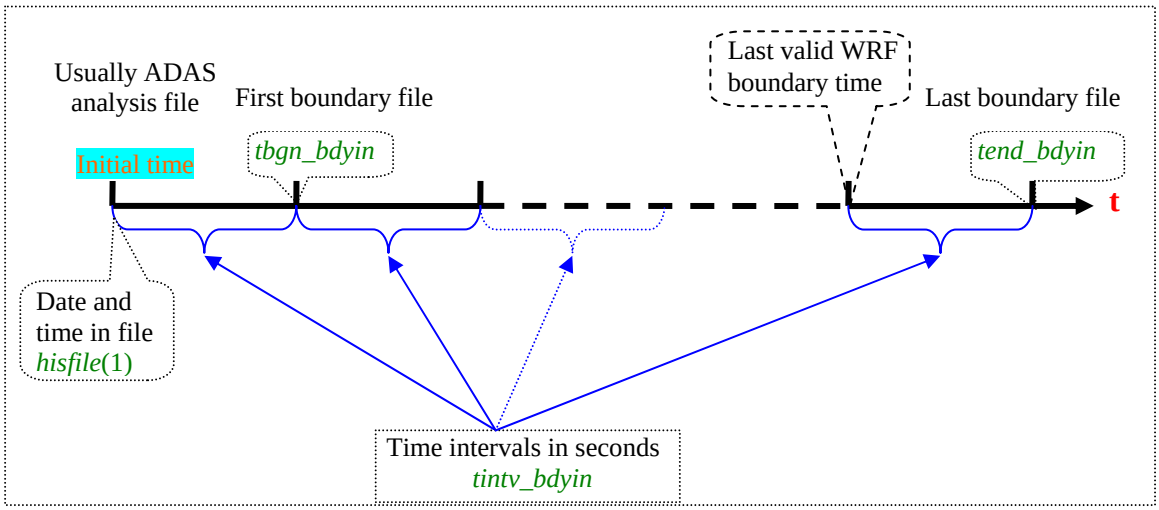



**Figure 2. Time sequence illustration for initial file and the boundary files**

**Note:** The program does not care what the values of "*tbgn_bdyin*" and "*tend_bdyin*" are, *i.e.* when the ARPS forecast starts. We use ARPS forecast time here just because it is the way how ARPS constructing file names. However, the data must be valid at the time showing above, *i.e.* the date and time (*year*, *month*, *day*, *hour*, *minute*, *second* in the file) + the forecast time (*time* in the file) must equal to Initial time + tintv_bdyin, and/or Initial time + n* tintv_bdyin, where *n* is the number of lateral boundary conditions.

## 2.5 Parameters for WRF Grids (&*wrf_grid*)

| WRF horizontal and vertical grid (**&wrf_grid**) | | |
|---|---|---|
| **Parameter** | **Definition/Purpose** | **Options/Suggested Values** |
| ***use_arps_grid*** | Provide a convenient way to use ARPS horizontal grid in the data file. | = 1, Users do not need to specify WRF horizontal grid, the program will build WRF grid based on the grid extracted from |

| | | |
|---|---|---|
| | | ARPS data files;<br>= 0, Users should provide WRF horizontal grid definitions explicitly. |
| | ***nx_wrf*** | Dimension size of WRF grid in *x*-direction (Used only when *use_arps_grid* = 0). | Problem dependent. |
| | ***ny_wrf*** | Dimension size of WRF grid in *y*-direction (Used only when *use_arps_grid* = 0). | Problem dependent |

**Note:** ARPS has one fake point in each side of *x*-direction or *y*-direction. However, WRF model will handle the boundary nudging during runtime. So *nx_wrf* and *ny_wrf* are the actual staggered size of the physical domain and they are 2 points less than *nx_arps* and *ny_arps* respectively if WRF horizontal grid is the same as ARPS horizontal grid.

| | | |
|---|---|---|
| | ***mapproj_wrf*** | Map projection option, as in *arps.input*. | = 1, polar stereographic projection;<br>= 2, Lambert conformal projection;<br>= 3, Mercator projection. |
| | ***sclfct_wrf*** | Map scale factor | Use 1.0 |
| | ***trulat1_wrf*** | 1st true latitude of map projection | Deg. North, negative for South hemisphere |
| | ***trulat2_wrf*** | 2nd true latitude of map projection (used only by Lambert projection) | Deg. N. |
| | ***trulon_wrf*** | True longitude of map projection | Deg. East, Negative for West |
| | ***ctrlat_wrf*** | Latitude of the center of the physical domain | Deg. N. |

| | | |
|---|---|---|
| ***ctrlon_wrf*** | Longitude of the center of the physical domain | Deg. E. |
| ***dx_wrf*** | Grid spacing in *x*-direction (meters) | Problem dependent |
| ***dy_wrf*** | Grid spacing in *y*-direction (meters) | Problem dependent |
| ***ptop*** | Pressure (Pascal) of the model top. | **Default**: 5000<br>It should not be less than the maximum pressure on the ARPS top layer. |
| ***vertgrd_opt*** | Option for schemes to generate WRF vertical grids. | = 0, sigma (WRF Eta) levels will be given explicitly with parameter *zlevels_wrf*;<br>= 1, Calculate linear sigma levels from parameter *nz_wrf*;<br>= 2, Calculate square root sigma levels;<br>= 3, Calculate the top 1/3 of the requested levels in linear and the lower 2/3 of the requested sigma levels in square root. |
| ***nz_wrf*** | Number of WRF vertical levels (used only when *vertgrd_opt > 0*, otherwise, the actual number of levels will be determined by counting the elements in *zlevels_wrf*.) | **Default**: 31 |
| ***pbot*** | Representative surface pressure in Pascal (used only when *vertgrd_opt = 3*). | **Default:** 101300 |
| ***zlevels_wrf*** | List of vertical levels (used only when *vertgrd_opt = 0*), sorted from bottom of the | The first value must be 1.0 (surface) and decrease to 0.0 (corresponds to top of the |

| | |
|---|---|
| | atmosphere to the top for WRF mass coordinate. The levels specified here are "full" levels. All variables except *w* are on the half levels. |

atmosphere).

**Note:**

ARPS2WRF provides fours schemes for WRF vertical grid localization.

**Scheme 0**: Specify the sigma (WRF Eta) levels explicitly and read parameters *vertgrd_opt* and *zlevels_wrf*. The number of vertical levels (*nz_wrf*) will be determined by counting the valid elements in *zlevels_wrf*.

**Scheme 1**: Read parameters *vertgrd_opt* and *nz_wrf* to calculate linear sigma levels.

**Scheme 2**: Read parameters *vertgrd_opt* and *nz_wrf* to calculate square root sigma levels.

**Scheme 3**: Read parameters *vertgrd_opt*, *nz_wrf* and *pbot*. Calculate the top 1/3 of the requested levels in linear and the lower 2/3 of the requested levels in square root in sigma. Because the linear method and the square root method may overlap each other at some levels, the actual number of vertical levels may not be exactly *nz_wrf*, *i.e*. *nz_wrf* may be changed by the code during runtime and a message will be printed for the actual value of *nz_wrf*.

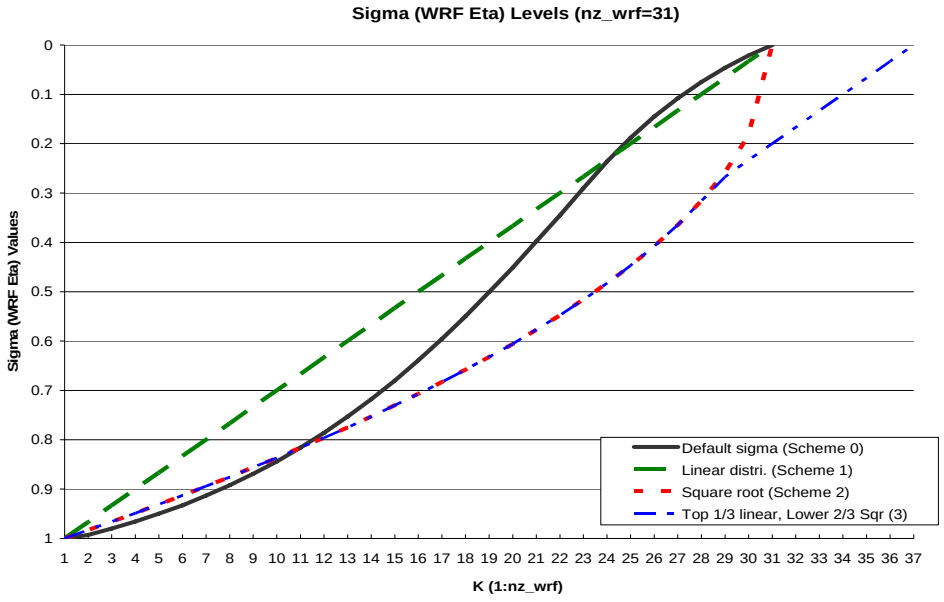



**Figure 3. Sigma (WRF Eta) levels distributions with each scheme**

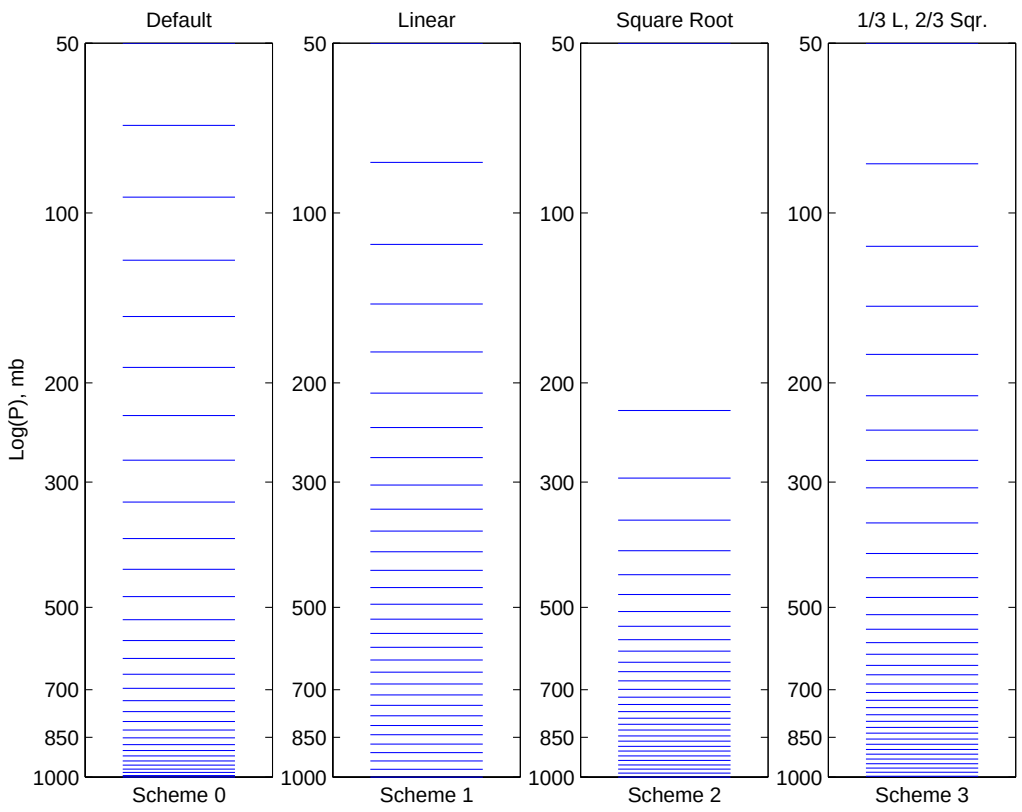



**Figure 4. WRFSI Eta level schemes illustrating in Log pressure coordinate.**

## 2.6 Parameters for WRF Options (&*wrf_opts*)

WRF options
(**&wrf_opts**)

The following variables are required for defining global attributes in the output files for WRF. ARPS2WRF also determines which variables should be included in the WRF initial field based on user's selections below. These options must be the same as WRF namelist file when run WRF later. Please refer to http://www.mmm.ucar.edu/wrf/users/wrfv2/wrf-namelist.html for their meaning and values.

| Parameter | Definition/Purpose | Options/Suggested Values |
|---|---|---|
| ***dyn_opt*** | The dynamical core option (only 2 works for ARPS2WRF at present) | = 2, Eulerian mass-coordinate<br><br>**Note**: removed since arps5.2.3. |
| ***diff_opt*** | Turbulence and mixing option | 0 = no turbulence or explicit spatial numerical filters (*km_opt* IS IGNORED). |

| | | |
|---|---|---|
| | | 1 = evaluates 2nd order diffusion term on coordinate surfaces. uses *kvdif* for vertical diff unless PBL option is used. may be used with *km_opt* = 1 and 4.<br>(= 1, recommended for real-data case when grid distance < 10 km)<br>2 = evaluates mixing terms in physical space (stress form) (x,y,z). turbulence parameterization is chosen by specifying *km_opt*. |
| ***km_opt*** | Eddy coefficient option.<br><br>**Note**:<br>• Option 2 and 3 are not recommended for DX > 2 km;<br>• Option 4 is recommended for real-data case when grid distance < 10 km | =1, constant (uses *khdif*, *kvdif* which are defined below). Different from *diff_opt*=1 because its horizontal diffusion is not along model zeta-surfaces. Same only in the case of no terrain;<br>= 2, 1.5 order TKE closure;<br>= 3, Smagorinsky first order closure;<br>= 4, horizontal Smagorinsky first order closure. |
| ***khdif*** | This parameter specifies the horizontal diffusion constant ($m^2/s$), and it is used with *diff_opt* = 1 option or *km_opt* = 1 option. | **Default**: 0 |
| ***kvdif*** | This parameter specifies the vertical diffusion constant ($m^2/s$), and it is used with *diff_opt* = 1 option or *km_opt* = 1 option | **Default**: 0 |
| ***mp_physics*** | Microphysics option | = 0, no microphysics<br>= 1, Kessler scheme<br>= 2, Lin et al. scheme<br>= 3, WSM 3-class simple ice scheme<br>= 4, WSM 5-class scheme<br>= 5, Ferrier (new Eta) microphysics<br>= 6, WSM 6-class graupel scheme<br>= 8, Thompson 7-class scheme (also predict ice number concentration) |

| | | |
|---|---|---|
| ***ra_lw_physics*** | Longwave radiation option | = 0, no longwave radiation;<br>= 1, rrtm scheme. |
| ***ra_sw_physics*** | Shortwave radiation option | = 0, no shortwave radiation;<br>= 1, Dudhia scheme;<br>= 2, Goddard short wave. |
| ***sf_sfclay_physics*** | Surface-layer option | = 0, no surface-layer<br>= 1, Monin-Obukhov scheme<br>= 2, Monin-Obukhov (Janjic Eta) scheme |
| ***sf_surface_physics*** | land-surface option | = 0, no surface temp prediction<br>= 1, thermal diffusion scheme<br>= 2, Noah land-surface model<br>= 3, RUC land-surface model |
| ***bl_pbl_physics*** | boundary-layer option | = 0, no boundary-layer<br>= 1, YSU scheme<br>= 2, Mellor-Yamada-Janjic (Eta) TKE scheme |
| ***cu_physics*** | cumulus option | = 0, no cumulus<br>= 1, Kain-Fritsch (new Eta) scheme<br>= 2, Betts-Miller-Janjic scheme<br>= 3, Grell-Devenyi ensemble scheme<br>= 99, previous Kain-Fritsch scheme |
| ***base_temp*** | base sea-level temperature (K) | 290. |
| ***dt*** | The time step for advection or large time step (unit in seconds, default is 2 sec) Typically, *dt* can be 5 - 7 times *dx* (in *km*). For example, if the grid size is *10 km*, one may use *dt = 60 sec*. | **Default:** 2 |
| ***spec_bdy_width*** | This parameter specifies the number of rows for specified boundary value nudging. Used in real-data cases. Should be at least the sum of *spec_zone* + *relax_zone* (see WRF namelist instructions). | **Default**: 5 |

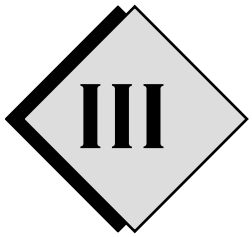
| | | |
|---|---|---|
| ***nprocx_wrf/ nprocy_wrf*** | Used for MPI WRF run when RSL_LITE library is linked. The values are passed to WRF NAMELIST file without changes. (Optional parameters) | **Default:** -1 = Determined by the WRF system automatically. |

## 2.7 Interpolation Parameters (&*interp_options*)

| Interpolation options (**&interp_options**) | | |
|---|---|---|
| **Parameter** | **Definition/Purpose** | **Options/Suggested Values** |
| ***iorder*** | Order of horizontal polynomial interpolation | = 1, bi-linear interpolation;<br>= 2, bi-quadratic interpolation. |
| ***korder*** | Order of vertical polynomial interpolation | = 1, bi-linear interpolation;<br>= 2, bi-quadratic interpolation. |

## 2.8 Output Parameters (&*output*)

| Output options (**&output**) | | |
|---|---|---|
| **Parameter** | **Definition/Purpose** | **Options/Suggested Values** |
| ***dirname*** | Directory name into which output files will be written | Up to 80 characters. |
| ***readyfl*** | A flag to dump a file when all of the output files are ready. Users can check this file to make sure the output files are available for use. This option assists with automatic job control. | = 0, Do not dump ready file;<br>= 1, Dump a ready file. |
| ***io_form*** | Flag for WRF output file format, | = 1, WRF internal binary format<br>= 7, NetCDF format,<br>= 5, PHDF5 format, for MPI mode only |
| ***create_namelist*** | Integer flag indicates whether to write a namelist file for WRF run. | = 1, write a namelist file;<br>= 0, do not write namelist file. |
| ***wrfnamelist*** | WRF namelist file name if *create_namelist* = 1 | Up to 132 characters.<br>'*namelist.input*' |




# Steps to Run the Programs

This section outlines the steps required to compile and run programs ***arps2wrf*** and ***wrfstatic***. It also illustrates these steps by setting up and running a real data case on May 25 1998. The data for this case is a set of GRIB files, which were defined over NMA (ETA) grid #212. Users can obtain the data for this case from ARPS download site (http://www.caps.ou.edu/ARPS/downloadarps.html).

## 3.1 Download and install the ARPS system

ARPS2WRF and WRFSTATIC are components of the ARPS system. The whole ARPS package is accessible via direct download on-line. Go to *http://www.caps.ou.edu/ARPS/downloadarps.html* and follow the detailed instructions as provided.

Having retrieved the compressed ARPS package *arps5.2.x.tar.gz* (where '*x*' defines a specific sub-version of the code), place it in an appropriate directory on your local system. To decompress and expand the code, type the following:

| | |
|---|---|
| **gunzip** *arps5.2.x.tar.gz* | [the original file was compressed using the GNU Free Software Foundation **gzip**, and is expanded using **gunzip**] |
| **tar** –xvf *arps5.2.x.tar* | [split the tar file into separate files that are placed into a subdirectory called *arps5.2.x* ] |
| **cd** *arps5.2.x* | [change directory to *arps5.2.x*] |
| **ls** | [to view the contents of the main ARPS directory] |

You will see several files and subdirectories inside the ARPS root directory. The following files and subdirectories will be our concern (for descriptions of other files and subdirectories, please refer to Chapter 3 of the ARPS User's Guide).

| | |
|---|---|
| *makearps* | [an UNIX C-shell script for compiling ***arps2wrf***, ***wrfstatic*** and other ARPS utility programs on common Unix platforms, the specific type of which is automatically detected. ] |
| *HISTORY* | [a history/log of modifications to the ARPS system of various versions. Always read this for latest updates. ] |
| *Makefile* | [a **make** description file used by ***makearps***.] |
| *src/* | [directory contains the source code by subdirectories for each of the program that are included in the full ARPS package. Sources for program ***arps2wrf*** and ***wrfstatic*** are located in *src/arps2wrf/*. ] |
| *input/* | [directory contains the NAMELIST input files that provide the control parameters of all ARPS associated utility programs. The input file for program ***arps2wrf*** and ***wrfstatic*** is *input/arps2wrf.input*. ] |
| *include/* | [directory contains include files used by various programs.] |
| *scripts/* | [directory contains several script files (mainly in Perl) that allow for automated runs of ARPS for select test cases.] |
| *bin/* | [directory will be created upon a successful compilation of any ARPS programs including ***arps2wrf*** and ***wrfstatic***, in which the executable is placed.] |
| *lib/* | [directory will be created upon a successful compilation of some ARPS programs in which the shared libraries are placed. The shared libraries used by program ***arps2wrf*** and ***wrfstatic*** are *libadas.a*, *libarps.a, libwrfio_int.a* or *libwrfio_phdf5.a*.] |

## 3.2 Required/Optional Libraries

The only external library that is always required for program ***arps2wrf*** and ***wrfstatic*** is the netCDF package from Unidata. You should check to ensure that the variables *INC_NET* and *LIB_NET* in *makearps* are pointing to the right netCDF paths on your system prior to build any programs.

**Hint**:

1. NetCDF version 3.6.0_p1 or later is required for large netCDF file support (> 2Gb), which is using 64bit file offset. You should uncomment several lines in subroutine *open_ncd_for_write* of file *src/arps2wrf/wrf_ioncd.f90* to support large files.

2. On a Linux system, if one wants to compile the programs using PGI (Intel or G95) compiler, please make sure the netCDF library is also installed using the same compiler.

There are optional external libraries that may be used by programs ***arps2wrf***: MPI implementation library and PHDF5 library. Neither of them is required for standard builds of the program. You need a version of MPI implementation only if you are going to be running distributed memory ***arps2wrf*** job. It is the users' responsibility to ensure that script *makearps* is making the right link with MPI library on their systems. Script *makearps* has been written to work for most common Unix systems and on most well-known machines, such as those machines at NCSA or PSC, but it is not guaranteed to work on users' specific system. The variables to be checked are *mpi_inc* and *mpi_lib* in *makearps.*

The PHDF5 (Parallel HDF5) library is required only if you are going to generate WRF files in PHDF5 format. Please note that PHDF5 library is a different library than HDF5 library. You can only generate PHDF5 file in MPI mode and MPI I/O support is required for the MPI implementation.

## 3.3 Prepare ARPS Data

The ARPS data sources are provided by program ***ext2arps*** and ***adas***. Program ***ext2arps*** is an ARPS preprocessing procedure that interpolates data from a user-supplied 3-D analysis grid to ARPS grid. A detailed description for this utility is found in Chapter 8 of the ARPS User's Guide. Program ***adas*** is the analysis program of the ARPS Data Assimilation System, which interpolates observations onto the ARPS grid, combining the observed information with a background field. The observed data sources can be rawinsondes and wind profilers, surface observations, raw radar data, radar-retrieved data or satellite observations. Document for ADAS is available from ARPS website as a supplement to the ARPS User's Guide (see, http://www.caps.ou.edu/ARPS/arpsdoc.html).

The steps required to make a complete real case run are listed as follows:

- Steps to prepare the terrain data on ARPS grid, assuming the terrain data base has been properly set up (Unix *vi* editor is used for illustrative purpose, those in [ ] are comments. This step is necessary if external terrain option is chosen for ***ext2arps***.)

  **vi** *arpstrn.input* [Set parameters for model grid configuration and terrain data analysis]
  **makearps** *arpstrn* [Compile and link program ***arpstrn***]
  **bin/arpstrn** < *arpstrn.input* > *arpstrn.output*
  **[**Execute program ***arpstrn***]

- Steps to produce background data set for ADAS and the data files used to generate the lateral boundaries of WRF. The data are interpolated from data sets from another model. In this example, it is a data set from ETA grid #212 GRIB files.

  **vi** *arps.input* [Set control parameters, including the input data file names for EXT2ARPS]
  **makearps** *ext2arps* [Compile and link program ***ext2arps***]

  **bin/ext2arps** < *arps.input* > *ext2arps.output*
  **[**Execute program ***ext2arps***]

- Steps to produce a three-dimensional initialization data set using ***adas***. ADAS uses the first file generated by EXT2ARPS as a background field.

**vi** *arps.input* [Set control parameters, including background file name and observation data sources, for data analysis]

**makearps** *adas* [Compile and link program ***adas***]

**bin/adas** < *arps.input* > *adas.output*
**[**Execute program ***adas***]

For users' convenience, several Perl scripts files are provided in subdirectory *scripts/* that allow for automated run of these steps for a real case with data on May 25, 1998 (downloadable from http://www.caps.ou.edu/ARPS/downloadarps.html). These scripts are *test_arpstrn.pl*, *test_ext2arps.pl*, and *test_adas.pl*. Instructions are also available in an ASCII file, see *scripts/README_plscripts*.

## 3.4 Prepare Surface Characteristic Data

There are totally three ways to prepare surface characteristic data set for ARPS2WRF. The first one is the traditional ARPS program ***arpssfc***. It is easy to be used. However, since it was specifically written for the ARPS system, the user domain is defined on an ARPS grid and several terrestrial datasets required for WRF run are not provided, such as maximum snow albedo, monthly albedo, slope data etc. Detailed definition and usage of program ***arpssfc*** is found in Chapter 8 of the ARPS User's Guide.

The second method is to generate WRFSI static file, *static.wrfsi.d01* through program *gridgen_model*. The disadvantage is that users have to download and install the whole WRFSI package even only one utility is needed. Furthermore, the steps for compiling and running WRFSI programs are different with the procedure to run ARPS programs, and it is not convenient for automatic workflow. Another drawback is that WRFSI programs only can be compiled using PGI compiler on Linux platform at present.

The third method to provide ARPS2WRF the surface characteristic file is through the enclosed program ***wrfstatic***. WRFSTATIC combines the common features for all ARPS utilities and the power of WRFSI for processing terrestrial data sets. It is the recommended method. Program ***wrfstatic*** reads the same NAMELIST input file as

***arps2wrf***. However, it is unnecessary to set all of the parameters for ***wrfstatic***. Table 3.1 describes those required parameters. Detailed definitions for these parameters are found in Section II.

Table 3.1 Control Parameters for Static Data Preprocessor - WRFSTATIC

| **Parameter** | **Definition** | **Options/Defaults** | **Conditions** |
|---|---|---|---|
| *hinfmt* | ARPS file format | 1 = Unformatted binary<br>7 = NetCDF | Required only if *use_arps_grid* = 1 |
| *grdbasfn* | Grid and base file name | '*adas25may1998.netgrdbas*' | |
| *hisfile(1)* | File name for initial ARPS data set | '*adas25may1998.net000000*' | |
| *sfcdtfl* | Static data file name to be written. | '*wrfstatic_d01*' | Always required. |
| *static_dir* | Directory name in which the subdirectories for external terrestrial data sets are located. | *e.g.* '*/home/xxx/data/GEOG/*' | |
| *silawt_parm_wrf* | Control parameters for topographical interpolations. | 0.0 | Always required. See file *README_toptwvl_silavwt* in WRFSI package. |
| *toptwvl_parm_wrf* | | 2.0 | |
| *start_date* | Date and time string to do interpolations for vegetation fraction or albedo. | *e.g.* *'1998-05-25_00:00:00'* | Required only if *use_arps_grid* = 0. |
| *use_arps_grid* | Denotes how to define the WRF horizontal grids. | 0 = Defines using parameters below.<br>1 = Defines automatically based on ARPS grid in file *hisfile(1)*. | Always required |
| *nx_wrf*<br>*ny_wrf*<br>*mapproj_wrf*<br>*sclfct_wrf*<br>*trulat1_wrf*<br>*trulat2_wrf*<br>*trulon_wrf*<br>*ctrlat_wrf*<br>*ctrlon_wrf*<br>*dx_wrf*<br>*dy_wrf* | Definitions of the WRF horizontal grid. See Section II for descriptions of these parameters. | Problem dependent. | Required only if *use_arps_grid* = 0 |
| *dirname* | Directory name into which output file is written. | './' | Always required |
| *readyfl* | Flag for ready file. | 0/1 | |

To compile and run program ***wrfstatic***, enter the following:

**makearps** *wrfstatic* [Compile and link the program]
**vi** *input/arps2wrf.input* [Editor control parameters]
***bin/wrfstatic*** < *arps2wrf.input* > *wrfstatic.output*
[Execute the program]

The general ARPS make shell script, **makearps** creates the executable file ***wrfstatc*** and puts it into subdirectory *bin/*. After the program is executed successfully, the output is a netCDF static file, *wrfstatic_d01*, which is conforming to WRF I/O API and it can be ingested by ARPS2WRF or WRF directly.

## 3.5 Compile and Run ARPS2WRF

The compilation and linking of ARPS2WRF, as 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, the root make description file is named *Makefile* in the ARPS root directory. Command **makearps** for ARPS2WRF accepts the same options as those described in Chapter 3 of the ARPS User's Guide.

To compile ***arps2wrf***, type:

**makearps** [option] *arps2wrf*

and the executable ***arps2wrf*** will be created and put into subdirectory *bin/*. To compile MPI mode of the executable, type

**makearps** [option] *arps2wrf_mpi*

and the parallel version of the program will be generated as ***arps2wrf_mpi*** and put into the directory *bin/*. The users should make sure that the MPI implementation was installed properly on their local system and the paths are all set correctly in the script *makearps* before invoking it.

If you want to clean all object files and executables associated with ARPS2WRF, type "**makearps** *clean.arps2wrf*". The command "**makearps** *clean*", however, will remove all built ARPS programs and object files including the objects for ARPS2WRF.

ARPS also provides a convenient way to tar up all ARPS2WRF related files including sources for those shared libraries. Type "**makearps** *arps2wrf.tar*", a tar file named *arps2wrf.tar* is generated in the ARPS root directory, which is an independent ARPS2WRF package containing the program ***arps2wrf*** and ***wrfstatic***.

After successful compilation and generating of the required data sources, it is time to run the program. The first step is to set the right control parameters as described in section II by editing file *arps2wrf.input*. Then type:

**bin/arps2wrf** < *arps2wrf.input* > *arps2wrf.output*

The standard output file *arps2wrf.output* contains detail messages about the runtime behavior of the program. You should first check this file for useful information when encountering any problems.

For distributed memory parallel system, some form of **mpirun** command is required. For example, on a Linux cluster, the command to run MPI jobs is **mpirun** or **mpiexec**:

**mpirun –**np 4 ***arps2wrf_mpi*** < *arps2wrf.input* > *arps2wrf.output*

On IBM, the command is

**poe *arps2wrf_mpi*** –procs 4 -stdinmode 0 < *arps2wrf.input* > *arps2wrf.output*

The programs will generate the following model outputs after a successful execution depending on specified control parameters.

- *wrfinput_d01* – WRF initialization file;
- *wrfbdy_d01* – WRF lateral boundary file;
- *namelist.input* – WRF namelist file.

These are all the input data required by the WRF model to start a real-data simulation.

A Perl script in subdirectory *scripts/*, *test_arps2wrf.pl* contains automatic steps to compile and run **arps2wrf** for a real-data test case on May 25, 1998. Users are welcome to give it a try first to get a feeling about how the program is working.

## 3.6 The Parallel version of ARPS2WRF

One important advantage of the **ext2arps/arps2wrf** sequence over WRFSI is that both **ext2arps** and **arps2wrf** have already been parallelized and they can handle high resolution cases over large model grid.

The program **arps2wrf** is parallelized by inserting MPI calls at appropriate locations in the source codes. To minimize the message passing between each processes, it is required that the WRF grid and the ARPS grid should define over the same physical domain. It is possible because both WRF horizontal grid and ARPS horizontal grid are defined over Arakawa-C grid. However, the WRF horizontal grid is not aligned with the ARPS horizontal grid. This is because the ARPS model defines one more fake point outside the physical boundaries to facilitate the implementation of boundary conditions. The WRF model, however, handles the boundary condition inside the program and it does not write these boundary zones explicitly. Figure 5 illustrates the relationship between the ARPS horizontal grid and the WRF horizontal grid in *x*-direction as well the physical domain in concern. The relationship in *y*-direction is the same.

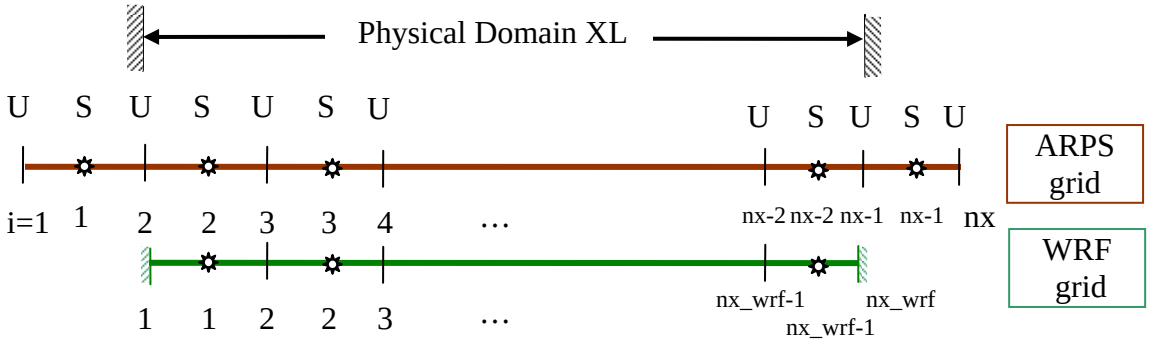



**Figure 5.** Illustrations of the grid structure of the ARPS grid and the WRF grid in *x*-direction, where S indicates the scalar points and U the u-velocity points.

From Figure 5, the physical dimension of the model domain in *x*-direction is $XL = (nx-3)\,dx$ for the ARPS grid, and it is $XL = (nx\_wrf-1)\,dx\_wrf$ for the WRF grid.

If the control parameter *use_arps_grid* is set to be 1, then the program will configure the WRF grids automatically based on the relationship outlined above. Since the horizontal grid points are coincided with each other, the program does not have to do horizontal interpolation from the ARPS grid to the WRF grid. The parallelization principle of ARPS2WRF is just based on this feature. In order for simple decomposition of the grid points over each process, the program further requires that the number of processes in each direction is a divisor of the number of the grid points in that direction, *i.e.* (*nx*-3) or (*ny*-3) should be dividable by *nproc_x* or *nproc_y* respectively.

For very large problem, it is sometimes infeasible to write all variables from the whole model domain into one file either because the file size may be too large to be handled by the file system or because it may take too much time to write such a large file. The ARPS system has a wonderful feature to avoid the problem by writing several small files for one large domain, *i.e.* the split files. For this purpose, one MPI process will write one split file which contains the data over its own patch on the physical domain. It is also possible that the number of patches for the ARPS data may be different from the number of processes (*nproc_x*×*nproc_y*) to be running for **arps2wrf_mpi**. So additional control

parameters are added in the *&message_passing* NAMELIST block and they are *readsplit*, *nxproc_in* and *nyproc_in*.

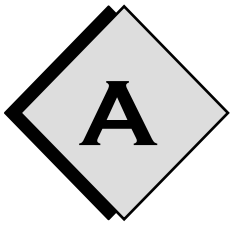
When the input data are in several patches, *readsplit* should 0. Otherwise, data of the physical domain is in one joined file and the parameter is set to be 1, which means that the program should read the data file and then split it over all ARPS2WRF processors. If *readsplit* = 0, the program distinguishes two further situations. The first case is that the number of data patches is the same as the number of processors for **arps2wrf_mpi**, *i.e. nxproc_in = nproc_x*, and *nyproc_in = nproc_y*. Otherwise, the program requires that *nxproc_in* (*nyproc_in*) must be larger or equal to *nproc_x* (*nproc_y*), and *nxproc_in* (*nyproc_in*) is exactly dividable by *nproc_x* (*nproc_y*). So that the program only needs to join several small data patches to get the data over its local MPI patch.

## 3.7 Known Problems

First, users should distinguish the WRF input file and WRF history files. Although both kinds of files have the same structure, they have minor difference in the data set contents. What ARPS2WRF generates is WRF input file, not WRF history files. The accompanied program **wrf2arps**, however, reads WRF history files instead of WRF input file. WRF2ARPS does a similar job as EXT2ARPS, but WRF2ARPS processes only WRF data in either netCDF format, or PHDF5 format or WRF native binary format instead of data from any other models. Although it is not the design purpose of WRF2ARPS, it is shown by users that WRF2ARPS stills can read WRF data generated by ARPS2WRF. The results are acceptable if users' concern is the atmospheric fields only, such as U, V, PT, P, QV, *etc*. instead of the surface characteristics.

Furthermore, Program **arps2wrf** and **wrf2arps** use totally different interpolation schemes and they are not invertible to each other. The interpolation scheme for ARPS2WRF is much similar to that used in WRFSI and REAL.EXE. WRF2ARPS, however, does interpolation following the framework established in EXT2ARPS. Due to

there are two separate polynomial interpolations involved, diverged results from the origin may be found if you run these two programs in series.




# Outlined Steps

This appendix lists the steps from an initial data interpolator EXT2ARPS to data assmilating with the observed information using ADAS, to the applying of ARPS2WRF for starting a WRF real-data simulation. The reference materials listed below are all available on either the ARPS web or the WRF web and users are encouraged to refer to them for complete instructions.

- ADAS 5 User's Guide
- ARPS 5 User's Guide
- WRF ARW version 2 User's Guide

## **Steps outlined**:

1. Run ARPSTRN (or ARPSTERN), EXT2ARPS, 88D2ARPS, NIDS2ARPS, MCI2ARPS and ADAS *etc*. to prepare data on ARPS grid for WRF.
2. Copy ADAS outputs (data analysis file as well as the grid and base file) to the ARPS2WRF working directory.
3. Copy ARPS history files created from either EXT2ARPS or ARPS to the ARPS2WRF working directory to be used as WRF lateral boundary conditions.
4. Copy the ARPS2WRF namelist input file to the working directory from *./input/arps2wrf.input*.
5. Edit the namelist file following the instructions in that file and section II in this document.
6. Run either ARPSSFC or WRFSI or WRFSTATIC to generate the static surface characteristic data set.
7. Compile and run ARPS2WRF.
8. Copy the generated file, "*wrfinput_d01*", "*wrfbdy_d01*", and "*namelist.input*" to WRF working directory.
9. Run the WRF model.