NASA/CR-2010-216290

ADAS Update and Maintainability

Leela R. Watson
ENSCO, Inc., Cocoa Beach, Florida
NASA Applied Meteorology Unit, Kennedy Space Center, Florida

September 2010

NASA/CR-2010-216290

ADAS Update and Maintainability

Leela R. Watson
ENSCO, Inc., Cocoa Beach, Florida
NASA Applied Meteorology Unit, Kennedy Space Center, Florida

National Aeronautics and
Space Administration

Kennedy Space Center
Kennedy Space Center, Florida, 32899-0001

September 2010

Acknowledgements

The author thanks Mr. Peter Blottman of the National Weather Service (NWS) in Melbourne, Florida and Mr. Brian Hoeth of the NWS, Spaceflight Meteorology Group in Houston, Texas for providing input and valuable feedback on the overall direction of this task.

Available from:

NASA Center for AeroSpace Information
7115 Standard Drive
Hanover, MD 21076-1320
(443) 757-5802

This report is also available in electronic form at
http://science.ksc.nasa.gov/amu/

Executive Summary

Since 2000, both the National Weather Service Melbourne (NWS MLB) and the Spaceflight Meteorology Group (SMG) have used a local data integration system (LDIS) as part of their forecast and warning operations. Both groups have benefited from three-dimensional (3-D) analyses that are delivered to forecasters every 15 minutes across the peninsula of Florida. The original LDIS was developed by the Applied Meteorology Unit (AMU) in 1998 and has undergone subsequent improvements. The intent is to generate products that enhance short-range weather forecasts issued in support of NWS MLB and SMG operational requirements within East Central Florida. The current LDIS uses the Advanced Regional Prediction System (ARPS) Data Analysis System (ADAS) package as its core, which integrates a wide variety of national, regional, and local observational data sets. It assimilates all available real-time data within its domain and is run at a finer spatial and temporal resolution than current national- or regional-scale analysis packages. As such, it provides local forecasters with a more comprehensive understanding of evolving fine-scale weather features. However, over the years the LDIS has become problematic to maintain since it depends on AMU-developed shell scripts that were written for an earlier version of the ADAS software. The NWS MLB requested that the LDIS be updated with the latest version of ADAS, new sources of observational data be incorporated into the system, and the AMU-developed shell scripts written to govern the system be upgraded and modified. The AMU met all requests and delivered the new LDIS scripts to NWS MLB and SMG for operational use.

The data that can be ingested by the LDIS includes Level II Weather Surveillance Radar-1988 Doppler (WSR-88D) data from six Florida radars, Geostationary Operational Environmental Satellites (GOES) visible and infrared satellite imagery, Kennedy Space Center (KSC)/Cape Canaveral Air Force Station (CCAFS) wind tower network data, and surface and upper air observations from the National Oceanic and Atmospheric Administration (NOAA) Earth System Research Laboratory/Global Systems Division (GSD)/Meteorological Assimilation Data Ingest System (MADIS). The MADIS data are available in network Common Data Form (netCDF) and must be converted to a format acceptable for use within the ARPS/ADAS system. The AMU modified Fortran code written by GSD that converts all MADIS data into an ADAS-compatible format.

The existing suite of scripts written to govern the LDIS was written in the bash-shell scripting language, however, NWS MLB and SMG agreed that the shell scripts should be rewritten using the Perl programming language since Perl is easier to port to different platforms and is easier to maintain. One of the goals of this task was to keep the new scripts as generic as possible so that users are allowed more flexibility in creating an analysis that fits their needs. The number of user input options was increased to allow more control over the system and so that the scripts and the software could be easily updated in the future. In addition, each script was rewritten so that each program could be run independently from the rest of the model.

The suite of Perl scripts runs a complete model system that includes the pre-processing step, the main model integration, and the post-processing step. The pre-processing step prepares the terrain data sets, surface characteristics data sets, and the objective analysis for model initialization. The scripts allow either the WRF or ARPS model to be used for the main model integration. Finally, a variety of conversion routines are available to post-process the model output so the data can be displayed with various visualization packages.

In addition, the AMU updated the previously developed ADAS GUI. The original GUI was developed by the AMU in 2004 and allows forecasters to quickly and easily interact with ADAS to maintain or improve the integrity of each 15 minute analysis cycle. The AMU rewrote the previous shell script that extracted information about the observations using Perl. The script has been updated to take into account the new MADIS data now assimilated by ADAS. The AMU also created a new map background to replace the existing one using the MapServer software for the ADAS GUI.

Table of Contents

List of Figures

List of Tables

1. Introduction

Since 2000, both the National Weather Service Melbourne (NWS MLB) and the Spaceflight Meteorology Group (SMG) have used a local data integration system (LDIS) as part of their forecast and warning operations. The original LDIS was developed by the Applied Meteorology Unit (AMU) in 1998 (Manobianco and Case 1998) and has undergone subsequent improvements. Each has benefited from three-dimensional (3-D) analyses that are delivered to forecasters every 15 minutes across the peninsula of Florida. The intent is to generate products that enhance short-range weather forecasts issued in support of NWS MLB and SMG operational requirements within East Central Florida. The current LDIS uses the Advanced Regional Prediction System (ARPS) Data Analysis System (ADAS) package as its core, which integrates a wide variety of national, regional, and local observational data sets. It assimilates all available real-time data within its domain and is run at a finer spatial and temporal resolution than current national- or regional-scale analysis packages. As such, it provides local forecasters with a more comprehensive understanding of evolving fine-scale weather features.

Over the years, the LDIS has become problematic to maintain since it depends on AMU-developed shell scripts that were written for an earlier version of the ADAS software. The goals of this task were to update the NWS MLB/SMG LDIS with the latest version of ADAS, incorporate new sources of observational data, and upgrade and modify the AMU-developed shell scripts written to govern the system. In addition, the previously developed ADAS graphical user interface (GUI) was updated. Operationally, these upgrades will result in more accurate depictions of the current local environment to help with short-range weather forecasting applications, while also offering an improved initialization for local versions of the Weather Research and Forecasting (WRF) model used by both groups.

2. Description of Software

2.1 ADAS Analysis Program

The ADAS analysis program (Brewster 1996) was developed by the University of Oklahoma to assimilate a variety of observed data that could then be initialized by the ARPS numerical weather prediction (NWP) model. ADAS interpolates observations onto the ARPS grid by combining the observed field with a background field. The background field is generally supplied by a larger national- or regional-scale model.

ADAS has two main components. The first is a Bratseth objective analysis scheme that evaluates pressure, wind, potential temperature, and specific humidity. In this scheme, an analysis is created from a background or first-guess grid, which is generally supplied by a larger-scale NWP model and the observed data. The analysis is created by modifying the background values of variables at specific grid points based on observational data located within a specified distance from the grid point. Each observation that influences a specific grid point is assigned a weight that is inversely proportional to its distance from the grid point. The sum of the weighted observations is added to the background value at that grid point to generate a new analysis. A new analysis grid is created by repeating this process for all grid points.

The second component of ADAS is a 3-D cloud analysis scheme that is used for the hot-start initialization (Zhang et al. 1998). The ADAS cloud analysis is designed to create consistency with all data and the typical meteorology of clouds by using surface observations of cloud cover and height, satellite data, and radar data to determine the cloud cover, cloud liquid and ice water, cloud type, rain/snow/hail mixing ratios, icing severity, in-cloud vertical velocity, cloud base and top, and cloud ceiling (Case et al. 2002; Zhang et al. 1998; Brewster 2002).

ADAS also includes two automated data quality control (QC) methods. The first pre-analyzes surface data by comparing observations with each other. Data are rejected if they differ significantly from neighboring observations. Observations are also compared to the background field interpolated to the observation site and are rejected if there are significant differences. The second is a manual method of excluding data at specified surface stations. ADAS includes a blacklist file to filter out those stations that regularly have instrument failures or unrepresentative observations.

2.2 ARPS Modeling System

The ARPS modeling system includes data ingest, quality control, and objective analysis packages, as well as radar parameter retrieval and data assimilation procedures, the prediction model, and post-processing packages. The ADAS analyses created within the system can also be used to provide initial conditions for the NWP component of ARPS. The NWP component is a 3-D, non-hydrostatic model created in terrain following coordinates. It uses a split-explicit scheme to integrate the sound-wave containing equations and uses advanced numerical techniques such as variance-conserving fourth order advection and monotonic advection schemes for scalar transport. It also employs physical parameterization schemes for explicit prediction of convective storms.

2.3 WRF Modeling System

The ADAS analyses can also be used to provide initial conditions for the WRF model. The WRF model has two dynamical cores: 1) the Advanced Research WRF (ARW) and 2) the Non-hydrostatic Mesoscale Model (NMM). The ARW core was developed primarily at the National Center for Atmospheric Research while the NMM was developed at the National Centers for Environmental Prediction (NCEP). Both cores are also incorporated into the WRF Environmental Modeling System (EMS) software, which was developed by the NWS Science Operations Officer (SOO) Science and

Training Resource Center (STRC). A benefit of using the WRF EMS is that it incorporates both dynamical cores into a single end-to-end forecasting model (Rozumalski 2006). The software consists of pre-compiled programs that are easy to install and run. The WRF EMS contains the full physics options available for the ARW and NMM cores, however, the physics options for the NMM are more limited than for the ARW. In addition to model core and initialization options, the WRF model can be run with one- or two-way nesting. Nesting allows a high-resolution forecast grid to be run within a coarser domain.

3. Real-time Data Ingest

The LDIS is set up to ingest a variety of observational data (Figure 1). Data that can be ingested into the system include:

- Level II Weather Surveillance Radar-1988 Doppler (WSR-88D) data from six Florida radars,
- Geostationary Operational Environmental Satellites (GOES) visible and infrared satellite imagery,
- Kennedy Space Center (KSC)/Cape Canaveral Air Force Station (CCAFS) wind tower network data, and
- Surface and upper air observations from the National Oceanic and Atmospheric Administration (NOAA) Earth System Research Laboratory/Global Systems Division (GSD)/Meteorological Assimilation Data Ingest System (MADIS; http://madis.noaa.gov).

The Level II WSR-88D data set contains full volume scans of reflectivity at a resolution of 1° by 1 km, radial velocity at 1° by 0.25 km, and spectrum width data at 1° by 0.25 km (Fulton et al. 1998). These data are available every 4 to 6 minutes. The GOES-12 visible imagery is available at a 1 km horizontal resolution every 15 minutes, and the infrared imagery is available at a 4 km horizontal resolution also every 15 minutes. Both visible and infrared imagery provide brightness temperatures to the analysis packages. The KSC/CCAFS wind tower network provides measurements of wind speed and direction, temperature, dewpoint temperature, and pressure and are available within 35 km of KSC/CCAFS.

3.1 MADIS Details

Both surface and upper air observations from the MADIS system are available throughout the US. MADIS data sets report variables such as u- and v-wind components, temperature, dewpoint temperature, pressure, sea surface temperature, relative humidity, precipitation, sensible weather (fog, thunder, hail, etc.), and vertical profiles of wind, among others. MADIS currently supports the following datasets:

- Meteorological surface observations
  - METARs (Automated Surface Observing Systems (ASOS), Automated Weather Observing System (AWOS), and non-automated stations) or Surface Airways Code (SAO),
  - Maritime stations including the Coastal Marine Automated Network (C-MAN), fixed and drifting buoys, and ship reports,
  - Modernized NWS Cooperative Observer stations,
  - UrbaNet mesonet stations, and
  - Various mesonet stations from local, state, and federal agencies and private firms such as the Automatic Position Reporting System (APRSWXNET), Florida Mesonet (FL-Meso), South Florida Water Management District (SFWMD), Remote Automated Weather Stations (RAWS), and many more.
- Radiosondes
- Profiler networks including the 915 and 50 MHz profilers located in and around KSC/CCAFS
- Automated aircraft reports
  - Aircraft Communications Addressing and Reporting System (ACARS),
  - Meteorological Data Collection and Reporting System (MDCRS),
  - Aircraft Meteorological Data Reporting (AMDAR), and
  - Tropospheric Airborne Meteorological Data Reporting (TAMDAR).
- Radiometer
- Satellite wind
- Satellite sounding
- Satellite radiance

- Snow
- Weather In-Situ Deployment Optimization Method (WISDOM) balloon wind

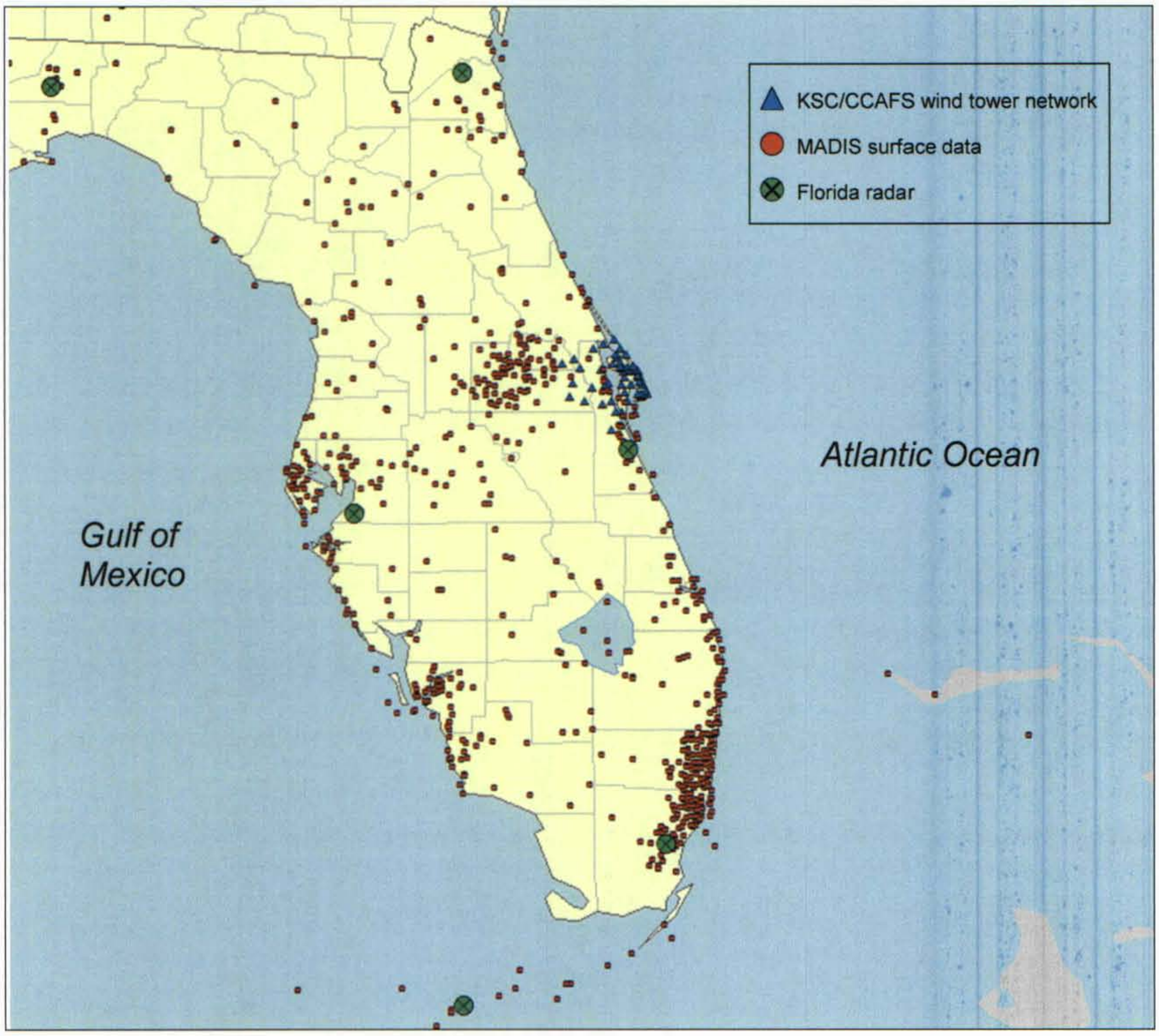


Figure 1. Map of the default Florida domain and distribution of observations across the peninsula used in the LDIS. The six Florida radars are indicated by a green circle with an X, the KSC/CCAFS wind tower locations are indicated by blue triangles, and a sample of the MADIS surface data locations are indicated by red circles.

3.2 Modifying MADIS Data for Ingest

All surface and upper air observations except the KSC/CCAFS wind tower data were obtained through MADIS. The data are available in network Common Data Form (netCDF) and had to be converted to a format acceptable for use within the ARPS/ADAS system. GSD provides various programs that convert the available data to American Standard Code for Information Interchange (ASCII) format; however, further modifications were needed in order to use the data within ADAS.

The MADIS Applications Program Interface (MADIS API) is a library of Fortran subroutines that provides access to all of the observations and QC information in the MADIS database. Utility programs included in the API package are used to read station information, observations, and QC information for a single time and to dump them to an output text file. A text parameter file is used to control the operation of each program. The text parameter file includes user-specified options such as the data set desired, the provider, the database being accessed, horizontal domain, QC level, correction handling, and the time window of interest. The AMU inserted extra Fortran code into the suite of GSD programs that convert all surface, rawinsonde, wind profiler, and automated aircraft data to ASCII format to produce ADAS-compatible output files. The original GSD code for the Fortran converter programs was not altered so that it could still be used for its original purpose. The AMU wrote bash shell scripts (*run_acars.sh*, *run_profiler.sh*, *run_raob.sh*, and *run_sfc.sh*) to run the converter programs in real-time. Time stamp and provider name information are defined in each shell script and are incorporated into a template text parameter file (*<datatype>dump.par.template*) using a sequence of substitution statements within the script using the Unix "sed" command (where *<datatype>* stands for the type of data being processed). A new text parameter file called *<datatype>dump.par* is created that contains the updated input parameters and the converter program is run. An ADAS-compatible output data file is then created.

NWS MLB receives MADIS mesonet data separated by data source in comma-separated value (CSV) format. These data sets arrive in a more timely fashion than the netCDF data, therefore, NWS MLB prefers to use these files in the ADAS analyses. The AMU wrote a Perl script, *convertCSV.pl*, to read and rewrite the CSV-formatted mesonet data into a format ingestible by ADAS. The script takes all data within a 15 minute time window and specified horizontal domain and writes multiple files that contain all of the observations categorized by provider name. The output files are named according to the data provider and timestamp. The script also appends all data from all providers into one combined data file. The ADAS program will read the mesonet observations from either the separated files or the appended file. Table 1 lists the mesonet data providers and input and output data file names where *year* is the current year, *mo* is the 2-digit month, *dd* is the 2-digit day, *hh* is the 2-digit hour, *mm* is the 2-digit minute, and *ss* is the 2-digit second.

Table 1. List of mesonet data providers, CSV-formatted input file names for each provider, and output ADAS-compatible output file names for each provider. Alternate input and output file names are given in parentheses.

Data Provider	***Input File Name***	***Output File Name***
APRSWXNET	APRSWXNET.dat.*yearmodd_hhmmss* (APRSWXNET.*yearmodd_hhmmss*)	aprs_*yearmoddhhmm*.lso
AWS Convergence Technologies, Inc. (AWS)	AWS_FL.dat.*yearmodd_hhmmss*	aws_*yearmoddhhmm*.lso
FL-Meso	FAWN.dat.*yearmodd_hhmmss* (FAWN.*yearmodd_hhmmss*)	fawn_*yearmoddhhmm*.lso
RAWS	FIRE.dat.*yearmodd_hhmmss* (RAWS.*yearmodd_hhmmss*)	fire_*yearmoddhhmm*.lso (raws_*yearmoddhhmm*.lso)
Miscellaneous (MISC)	MISC.*yearmodd_hhmmss*	misc_*yearmoddhhmm*.lso
National Estuarine Research Reserve System (NERRS)	nerresmet.*yearmodd_hhmmss*	nerresmet_*yearmoddhhmm*.lso
Non-Fed AWOS (NonFedAWOS)	NonFedAWOS.*yearmodd_hhmmss*	nonfedawos_*yearmoddhhmm*.lso
SFWMD	SFWMD.*yearmodd_hhmmss*	sfwmd_*yearmoddhhmm*.lso
UrbaNet	UrbaNet5.*yearmodd_hhmmss*	urbanet5_*yearmoddhhmm*.lso
WeatherFlow (WxFlow)	wxflow.dat.*yearmodd_hhmmss*	wxflow_*yearmoddhhmm*.lso
Weather for you (WXforYou)	WXforYou.dat.*yearmodd_hhmmss* (WXforYou.*yearmodd_hhmmss*)	wxforyou_*yearmoddhhmm*.lso
All data providers listed above	All input files listed above	mesonet_*yearmodd_hhmmss*.lso

4. Modification of Existing Scripts

The existing suite of scripts written to govern the LDIS was implemented in 1998 and was subsequently improved through the years. The scripts were written in the bash shell scripting language, however, NWS MLB and SMG agreed that the shell scripts should be rewritten using the Perl programming language since it is easier to port to different platforms and is easier to maintain. One of the goals of this task was to keep the new scripts as generic as possible so that the users are allowed more flexibility in creating an analysis that fits their needs. The AMU increased the number of user input options to allow more control over the system and so that the scripts and the software could be easily updated in the future. In addition, each script was rewritten so that each program could be run independently from the rest of the model. Changes made to each individual program are detailed in Sections 4.1 and 4.2.

The suite of Perl scripts runs a complete model system that includes the pre-processing step, the main model integration, and the post-processing step. An overview of the process flow is provided in Figure 2. The pre-processing step prepares the terrain data sets, surface characteristics data sets, and the objective analysis for model initialization. The scripts allow either the WRF or ARPS model to be used for the main model integration. Finally, a variety of conversion routines are available to post-process the model output so the data can be displayed with various visualization packages.

Each component of the LDIS is invoked by the master script *run.adas.wrf.cycle.pl*. The master script steps through all procedures of a single ARPS or WRF cycle. All tunable parameters are entered by the user into a configuration file named *ADAS_input.config* (see Table 4 in Appendix A). The master script opens the configuration file and reads the user input. It determines the model initialization time, latitude and longitude of the user domain, domain grid spacing, and the map projection and writes this information into an intermediate file, *parms.txt*, that each subsequent Perl script can then access. This allows the user to run each component of the LDIS separately, run an archive case easily, and also to avoid recreating the same code in each component of the LDIS. The master script then calls each of the LDIS components to run the complete modeling system. Each component of the system is described below. For ease of reading, the convention used in this document is as follows:

- All directory paths are <u>underlined</u>,
- File names are *italicized*,
- Executable programs or UNIX shell scripts are in **boldtype** font,
- < > indicates a user-specified argument, and
- arps5.x.x references the working version of the ARPS software.

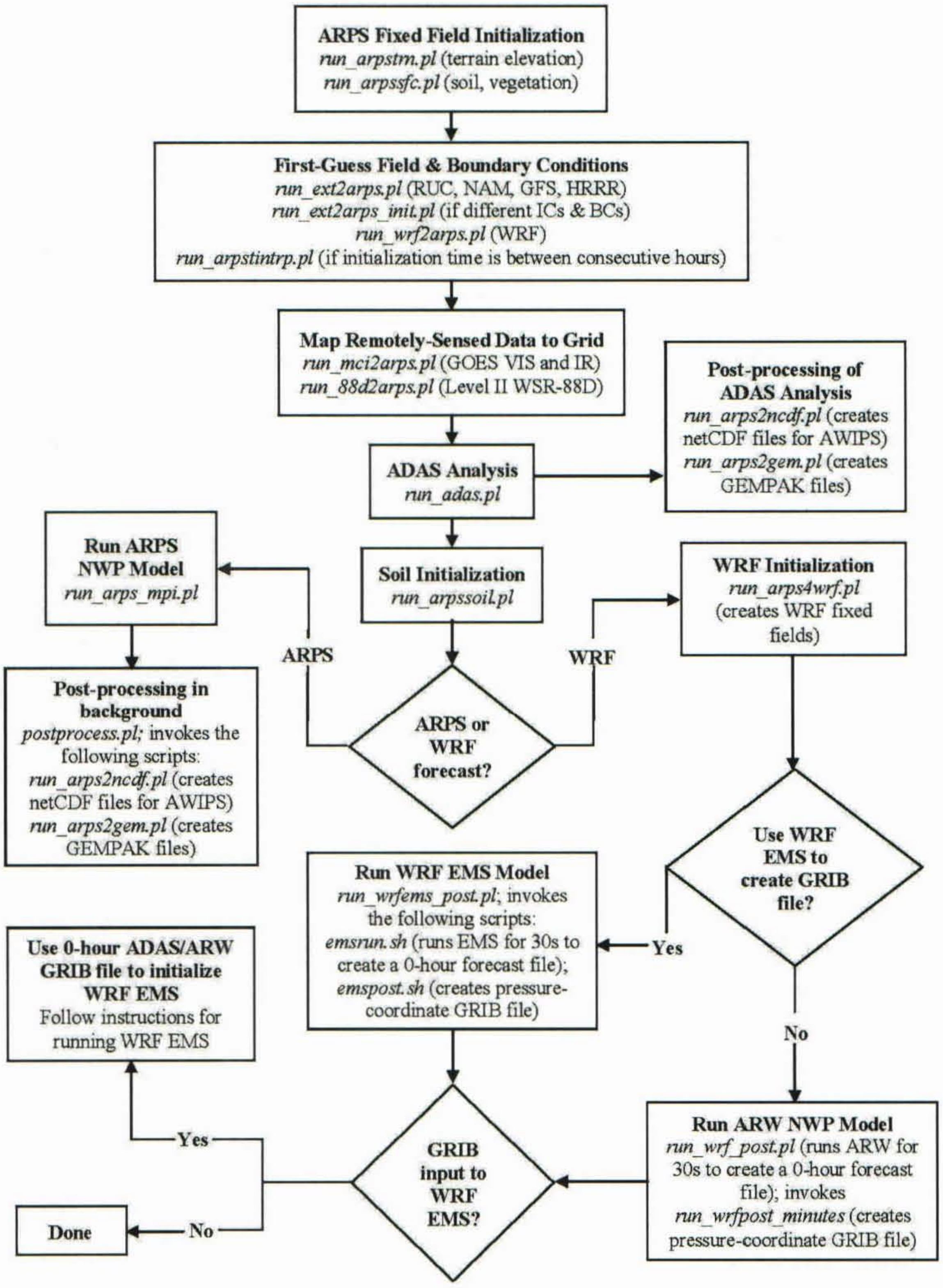

ARPS Fixed Field Initialization
run_arpstrn.pl (terrain elevation)
run_arpssfc.pl (soil, vegetation)
First-Guess Field & Boundary Conditions
run_ext2arps.pl (RUC, NAM, GFS, HRRR)
run_ext2arps_init.pl (if different ICs & BCs)
run_wrf2arps.pl (WRF)
run_arpstintrp.pl (if initialization time is between consecutive hours)
Map Remotely-Sensed Data to Grid
run_mci2arps.pl (GOES VIS and IR)
run_88d2arps.pl (Level II WSR-88D)
Post-processing of ADAS Analysis
run_arps2ncdf.pl (creates netCDF files for AWIPS)
run_arps2gem.pl (creates GEMPAK files)
ADAS Analysis
run_adas.pl
Run ARPS NWP Model
run_arps_mpi.pl
Soil Initialization
run_arpssoil.pl
WRF Initialization
run_arps4wrf.pl (creates WRF fixed fields)
ARPS
WRF
ARPS or WRF forecast?
Post-processing in background
postprocess.pl; invokes the following scripts:
run_arps2ncdf.pl (creates netCDF files for AWIPS)
run_arps2gem.pl (creates GEMPAK files)
Use WRF EMS to create GRIB file?
Run WRF EMS Model
run_wrfems_post.pl; invokes the following scripts:
emsrun.sh (runs EMS for 30s to create a 0-hour forecast file);
emspost.sh (creates pressure-coordinate GRIB file)
Yes
No
Use 0-hour ADAS/ARW GRIB file to initialize WRF EMS
Follow instructions for running WRF EMS
Yes
GRIB input to WRF EMS?
Run ARW NWP Model
run_wrf_post.pl (runs ARW for 30s to create a 0-hour forecast file); invokes
run_wrfpost_minutes (creates pressure-coordinate GRIB file)
Done
No

Figure 2. A flow chart depicting the process flow of the ADAS-ARPS and ADAS-WRF initialization and forecast cycles. Script names are given in *italics* along with brief descriptions of the scripts in parentheses.

4.1 Pre-processing Components

There are seven routines used to process the data for input to the model. They prepare the two-dimensional (2-D) terrain, surface characteristics, background and boundary conditions, radar and satellite data, soil temperature and moisture, and the objective analysis for model initialization.

4.1.1 Terrain Generation

The program **ARPSTRN** creates the 2-D terrain datasets, using a bi-linear interpolation scheme to analyze data. The first step of the terrain program is to convert the raw terrain ASCII text files into unformatted binary files. A final smoothed terrain field is then created using the unformatted binary files. It currently supports four raw terrain data sets to create the 2-D fields: 1°x1° (~110 km), 5'x5'(~10km), 30"x30" (~1km), and 3"x3" (US coverage only).

The Perl script *run_arpstrn.pl* runs the **ARPSTRN** program and is invoked by the master script *run.adas.wrf.cycle.pl*. The Perl script reads user configured values from the configuration file and domain information from the file *parms.txt*. User-supplied arguments include the terrain run name, the input directory containing the four raw terrain data sets, and the output directory (see Table 4 in Appendix A). The script determines the correct resolution setting of the raw terrain data to use based on the domain information supplied by the user. The arguments read and created by *run_arpstrn.pl* are incorporated into the arps5.x.x/input/*arpstrn.input* file using a series of substitutions from a Perl hash list and the Perl subroutine arps5.x.x/scripts/*mkinput.pm*. A Perl hash lets you create a one-to-one association between a variable, called the key, and a value. This subroutine reads an ARPS namelist file (*arpstrn.input* in this case), substitutes the values of parameters specified in a hash list, and then saves the new file containing the updated input parameters. The new file is named using the terrain run name specified by the user in the configuration file and appended with ".input" (i.e., *<terrain_runname>.input*). The program arps5.x.x/bin/**arpstrn** is then run. The data file (*<terrain_runname>.trndata*), output file (*<terrain_runname>.output*), and input file are all written to the user-specified output directory. The **ARPSTRN** program should be run each time the grid dimensions, location, or resolution of the domain changes. The terrain option can be turned off in the configuration file if these parameters remain the same.

4.1.2 Surface Characteristics

The program **ARPSSFC** prepares the surface characteristics data set for use in ARPS. The program uses soil type, vegetation type, and monthly Normalized Difference Vegetation Index (NDVI) data sets to derive the soil type, vegetation type, leaf area index, and the surface roughness length. The soil type, vegetation types, and NDVI from these files are mapped onto the ARPS grid by using values from the nearest data point. The leaf area index and surface roughness length are derived from vegetation type and NDVI data sets on the ARPS grid and are then smoothed.

The Perl script *run_arpssfc.pl* runs the **ARPSSFC** program and is invoked by the master script *run.adas.wrf.cycle.pl*. The Perl script reads user configured values from the configuration file and domain information from the file *parms.txt*. User-supplied arguments include the surface run name, the input directory containing the raw data sets needed to run the surface program, and the output directory (see Table 4 in Appendix A). The script determines the correct raw surface data sets to use based on the domain resolution specified in the configuration file. The arguments read and created by *run_arpssfc.pl* are incorporated into the arps5.x.x/input/*arps.input* file using a series of substitutions from a hash list and the subroutine arps5.x.x/scripts/*mkinput.pm*. The new file is named using the surface run name specified by the user in the configuration file and appended with ".input" (i.e., *<surface_runname>.input*). The program arps5.x.x/bin/**arpssfc** is then run. The data file (*<surface_runname>.trndata*), output file (*<surface_runname>.output*), and input file are all written to the user-specified output directory. At a minimum, the vegetation, soil, and NDVI data sets need to be generated each month or each time that the grid dimensions, location, or grid spacing changes. The surface option can be turned off in the configuration file if these parameters remain the same.

4.1.3 Background Field and Boundary Conditions

The background field and boundary conditions can be created with a user-supplied 3-D analysis grid. The background or first-guess field for the model initialization/ADAS analysis is created by the program **EXT2ARPS**. The background grids must include specific ARPS variables and arrays of latitude and longitude coordinates and must cover a volume (horizontally and vertically) as large as the user's domain of interest. The program linearly interpolates the user-supplied data to accurately place the data on the ARPS grid and initialize the base state and perturbation fields. A smoother is then applied to the ARPS grid to get rid of any noise resulting from the interpolation. Finally, the wind fields are adjusted to comply with the anelastic mass continuity equation. The **EXT2ARPS** program creates a set of history files that can be used in the ADAS analysis or to initialize the model.

The Perl script *run_ext2arps.pl* runs the **EXT2ARPS** program and is invoked by the master script *run.adas.wrf.cycle.pl*. The Perl script reads user configured values from the configuration file and domain information from the file *parms.txt*. User-supplied arguments include the ext2arps run name, the directory containing the raw model data sets in GRIdded Binary (GRIB) format, the output directory, and raw model information (see Table 4 in Appendix A). The script determines the correct model initialization time, forecast files, and boundary condition information to use based on the input model information and user input. An optional section in the configuration file is included if the user wants to use a different background model to create the boundary and initial conditions. If the optional section is completed, the script is invoked twice – the first creating the boundary and initial conditions using the first listed model and the second creating and replacing the initial conditions using the second model listed. The arguments read and created by *run_ext2arps.pl* are incorporated into the arps5.x.x/input/*arps.input* file using a series of substitutions from a hash list and the subroutine arps5.x.x/scripts/*mkinput.pm*. The new file is named according to the run name specified by the user in the configuration file and is appended with ".input" (i.e., *<ext2arps_runname>.input*). The program arps5.x.x/bin/**ext2arps** is then run.

The first-guess field for the ARPS or WRF model initial conditions or ADAS analysis corresponding to the model initialization or analysis time is contained in the files of the form

<ext2arps_runname>.yearmoddhhmm.bin000000 and *<ext2arps_runname>.yearmoddhhmm.bingrdbas*,

where *year* is the current year, *mo* is the 2-digit month, *dd* is the 2-digit day, *hh* is the 2-digit hour, *mm* is the 2-digit minute, and "*bin*" stands for binary data format. The boundary condition data used to force the ARPS or WRF model at the lateral and top boundaries is contained in files of the form

<ext2arps_runname>.yearmoddhhmm.yearmodd.hhmmss,

where *ss* is the 2-digit second. The background data files, output file (*<ext2arps_runname>.output*), and input file (*<ext2arps_runname>.input*) are all written to the user-specified output directory.

The updated LDIS supports multiple raw model data sets to create the background fields. These include the 13, 20, and 40 km Rapid Update Cycle (RUC) model, the 3 km High-Resolution Rapid Refresh (HRRR), the 12 km North American Mesoscale (NAM) model available in personal tiles or over the contiguous United States (CONUS), and the 40 km CONUS NAM model. Currently, the 13 km RUC is run every hour with forecast output available every hour to forecast hour 18. The 20 and 40 km RUC are run every hour with forecast output available at hours 1, 2, and 3 and then every 3 hours until forecast hour 18. The 3 km HRRR is run every hour with forecast output available every hour to forecast hour 15. The 12 and 40 km NAM are run every 6 hours with output available every 3 hours until forecast hour 84.

In addition to the RUC and NAM models, WRF model forecasts can be used to create the first-guess field for initial conditions or ADAS analysis. Currently, there is no support for converting multiple WRF output files for boundary condition data. The Perl script *run_wrf2arps.pl* runs the **WRF2ARPS** program and is invoked by the master script *run.adas.wrf.cycle.pl*. The Perl script reads user configured values from the configuration file and domain information from the file *parms.txt*. User-supplied arguments include the wrf2arps run name, the directory containing the WRF model output, the output directory, and the WRF forecast output interval in minutes (see Table 4 in Appendix A). The script determines the correct model initialization time and forecast file to use based on the user input. The arguments read and created by *run_wrf2arps.pl* are incorporated into the arps5.x.x/input/*wrf2arps.input* file using a series of substitutions from a hash list and the subroutine arps5.x.x/scripts/*mkinput.pm*. The new file is named *wrf2arps.input*. The program arps5.x.x/bin/**wrf2arps** is then run. The data files, input file, and output file (*wrf2arps.output*) are all written to the user-specified output directory.

If the user wants to create a background field for the ADAS analysis between consecutive hours, the master script will invoke the time interpolation Perl script *run_arpstintrp.pl* that runs the **ARPSTINTRP** program. This program interpolates two ARPS history data files from **EXT2ARPS** on a grid of the same size to a time in between them. This script will not work for WRF model background data since it is assumed that the user will output WRF data at their interval of interest. The Perl script will look for background data files within the last 12 hours in the ext2arps output directory. At a minimum, there must be three output files from **EXT2ARPS** to run the **ARPSTINTRP** program: one base state and grid file of the form *<ext2arps_runname>.yearmoddhhmm.bingrdbas* and two history files of the form *<ext2arps_runname>.yearmoddhhmm.bin000000* and *<ext2arps_runname>.yearmoddhhmm.binffffff*, where *ffffff* stands for a forecast time in seconds. The analysis time must fall in between the two history file times. There are no user inputs for the **ARPSTINTRP** program. All arguments used by the *run_arpstintrp.pl* script are gathered from the **EXT2ARPS** user inputs and are incorporated into the arps5.x.x/input/*arpstintrp.input* file using a series of substitutions from a hash list and the subroutine arps5.x.x/scripts/*mkinput.pm*. The new input file is named *time_intrp.input*. The program arps5.x.x/bin/**arpstintrp** is then run. The input and output (*time_intrp.output*) files are written to the directory specified by 'arpstintrp_dir' in the configuration file, while the data files are written to the user-specified ext2arps output directory.

4.1.4 Radar Data Remapper

The program **88D2ARPS** reads Level II WSR-88D radar data in polar coordinates and remaps the data onto the ARPS grid. The program uses a least-squares fit of the observed data to interpolate and produces one output file for each radar data volume scan. The data returned are reflectivity (dBZ), radial velocity (m/s), spectrum width (m/s), and local Nyquist velocity (m/s) interpolated to the 3-D ARPS grid. This can then be used in the ADAS analysis.

The Perl script *run_88d2arps.pl* runs the **88D2ARPS** program and is invoked by the master script *run.adas.wrf.cycle.pl*. The Perl script reads user configured values from the configuration file and domain information from the file *parms.txt*. User-supplied arguments include the input directory containing the raw radar data and the output directory (see Table 4 in Appendix A). The program **88D2ARPS** requires background data files, terrain data, and surface data to run. The script searches for the required data based on the model initialization time and output directories for the terrain, surface, and background data. The script searches backwards in time for the latest radar file within a 20 minute window. Currently, the script is set to process the six Florida WSR-88D radars. However, this can be changed by altering one line of code in the script (see Appendix A, Section 6 for details). The arguments read and created by *run_88d2arps.pl* are incorporated into the arps5.x.x/input/*arps.input* file using a series of substitutions from a hash list and the subroutine arps5.x.x/scripts/*mkinput.pm*. A new input file is created for each radar processed and is named *remap.input.<radar>.<hres>km* where *radar* is the name of the radar being processed and *hres* is the horizontal resolution of the analysis. The program arps5.x.x/bin/**88d2arps** is then run. The output files have the same naming designation as the input files except for replacing the word '*input*' with '*ouput*' and both are written to the working directory from which *run_88d2arps.pl* is run. The data files are written to the user-specified output directory and are in the form

<*radar*>.*yymodd.hhmm*,

where *radar* is the name of the radar being processed and *yy* is the 2-digit year.

4.1.5 Satellite Data Remapper

The program **MCI2ARPS** remaps Man computer Interactive Data Access System (McIDAS) GOES VARiable format (GVAR) Area files from the satellite-observed pixels to the ARPS grid. The program works with both infrared (IR) and visible (VIS) satellite data and determines what data are in the file without user-specification. The data returned are cloud top temperature from the IR satellite data and albedo from the VIS satellite data. This program must be run to assimilate satellite data in ADAS.

The Perl script *run_mci2arps.pl* runs the **MCI2ARPS** program and is invoked by the master script *run.adas.wrf.cycle.pl*. The Perl script reads user configured values from the configuration file and domain information from the file *parms.txt*. User-supplied arguments include the satellite run name, the input directory containing the raw satellite data, and the output directory (see Table 4 in Appendix A). The script determines the correct satellite data to use based on the model initialization time. The arguments read and created by *run_mci2arps.pl* are incorporated into the arps5.x.x/input/*arps.input* file using a series of substitutions from a hash list and the subroutine arps5.x.x/scripts/*mkinput.pm*. The new file is named according to the satellite run name specified by the user in the configuration file and is appended with ".input" (i.e., <*satellite_runname*>.*input*). The program arps5.x.x/bin/**mci2arps** is run and the output data files are written to the user-specified output directory and are of the form

<*hres*>-*km.yearmoddhhmm.goes12.albedo* and <*hres*>-*km.yearmoddhhmm.goes12.cttemp*

where *hres* is the horizontal resolution of the analysis.

4.1.6 ADAS Analysis

The 3-D analysis for the model initialization is created within the **ADAS** program. ADAS ingests a variety of observational data (radar reflectivity and radial velocity, infrared and visible satellite data, surface observations, etc.) and analyzes these data onto the ARPS grid using the Bratseth objective analysis technique. Details of the analysis program can be found in Section 2.1.

The Perl script *run_adas.pl* runs the **ADAS** program and is invoked by the master script *run.adas.wrf.cycle.pl*. The Perl script reads user configured values from the configuration file and domain information from the file *parms.txt*. User-supplied arguments include the analysis run name, the input directories containing the surface and upper air data sets, and the output directory (see Table 4 in Appendix A). The **ADAS** program requires background data files, terrain data, and surface data to run and can, as an option, assimilate satellite, radar, surface, and upper air data. The script searches for the required and optional data based on model initialization time, output directories for the terrain, surface, background model, satellite, and radar data, and input directories for surface and upper air data. Details regarding how the MADIS surface and upper air data were modified and reformatted into an ADAS-compatible format can be found in Section 3.1. The arguments read and created by *run_arpssfc.pl* are incorporated into the arps5.x.x/input/*arps.input* file using a series of substitutions from a hash list and the subroutine arps5.x.x/scripts/*mkinput.pm*. The new file is named according to the analysis run name specified by the user in the configuration file and is appended with ".input" (i.e., <*adas_runname*>.*input*). The program arps5.x.x/bin/**adas** is then run. The data files are written to the user-specified output directory and are of the form

<*adas_runname*>.*yearmoddhhmm.bin000000* and <*adas_runname*>.*yearmoddhhmm.bingrdbas*.

The ADAS input and output files (<*adas_runname*>.*output*), are written to the user-specified output directory.

Two post-processing scripts can then be run to view the ADAS analyses. The Perl script *run_arps2ncdf.pl* processes the ADAS output and is invoked by the master script *run.adas.wrf.cycle.pl*.

The Perl script reads user-configured values from the configuration file and domain information from the file *parms.txt*. The **ARPS2NCDF** program searches for the ADAS output files based on the model initialization time and ADAS output directory. The script establishes the appropriate time and input parameters for the **ARPS2NCDF** program that converts the analysis data into netCDF files for display in the Advanced Weather Interactive Processing System (AWIPS). The arguments read and created by *run_arps2ncdf.pl* are incorporated into the arps5.x.x/input/*arps2ncdf.input* file using a series of substitutions from a hash list and the subroutine arps5.x.x/scripts/*mkinput.pm*. The new file is named *arps2ncdf.input*. The program arps5.x.x/bin/**arps2ncdf** is then run. The netCDF files are written to the user-specified output directory and are of the form

<adas_runname>.yearmoddhhmm.binffffff.ncd,

where "*ncd*" stands for netCDF format. The forecast variables stored in the netCDF files are available at 19 pressure levels in 50 mb increments from 1000 mb to 100 mb and at the surface. The grids stored in the netCDF files include all primary meteorological variables, including certain derived quantities such as forecast radar reflectivity and echo tops. The **ARPS2NCDF** input and output files (*arps2ncdf.output*) are written to the user-specified ADAS output directory.

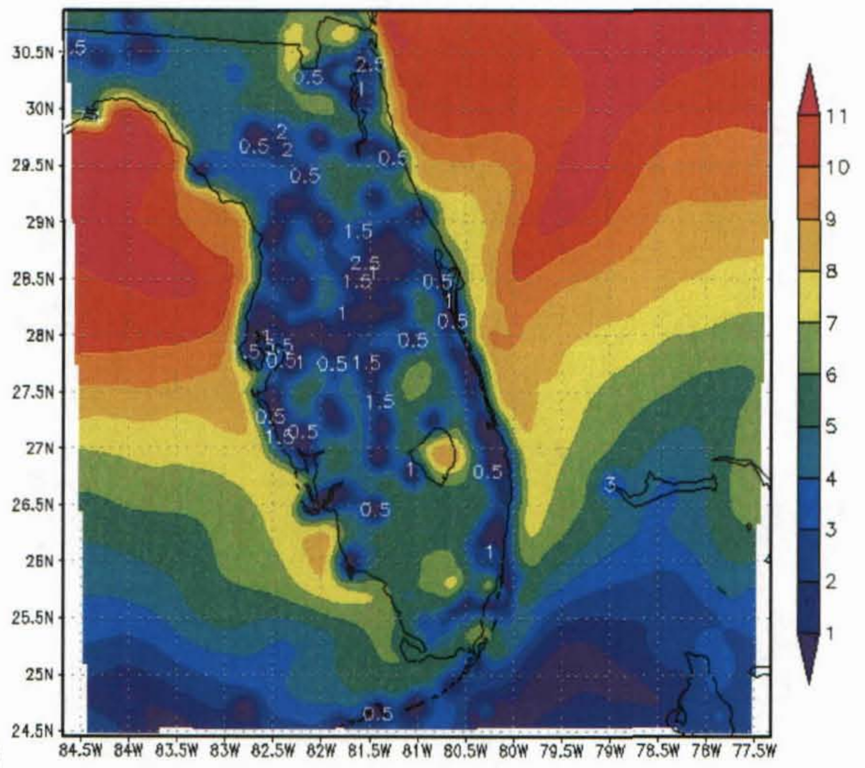
The Perl script *run_arps2gem.pl* processes the ADAS output and is also invoked by the master script *run.adas.wrf.cycle.pl*. The Perl script reads user-configured values from the configuration file and domain information from the file *parms.txt*. The **ARPS2GEM** program searches for the ADAS output files based on the model initialization time and ADAS output directory. The script establishes the appropriate time and input parameters for the **ARPS2GEM** program and converts data into GEMPAK visualization and display files. The arguments read and created by *run_arps2gem.pl* are incorporated into the arps5.x.x/input/*arps2gem.input* file using a series of substitutions from a hash list and the subroutine arps5.x.x/scripts/*mkinput.pm*. The new file is named *arps2gem.input*. The program arps5.x.x/bin/**arps2gem** is then run. The GEMPAK files are written to the user-specified output directory and are of the form

<adas_runname>.yearmoddhhmm.binffffff.gem,

where "*gem*" stands for a GEMPAK grid file. The forecast variables stored in the GEMPAK grid files are available at 19 pressure levels in 50-mb increments from 1000 mb to 100 mb and at the surface. The grids stored in the GEMPAK files include temperature, dew point, height (pressure), u- and v-winds, specific humidity, cloud and precipitation mixing ratios, cloud ceilings and cloud-top heights, and cloud fraction. The **ARPS2GEM** input and output files (*arps2gem.output*) are written to the user-specified ADAS output directory.

Figure 3 shows the influence of assimilating observations in the ADAS analysis on the wind speed in (a) vs. the RUC 13 km background field linearly interpolated to the ARPS grid in (b). The RUC 13 km background field is essentially the same as an ADAS analysis that does not assimilate any observations. The ADAS analysis corrected the background field with numerous negative wind speed adjustments (2 – 3 m s^{-1}) across much of the Florida peninsula. This is an example of how the ADAS analysis adjusts the background field to fit more closely to the observations.

(a)



(b)

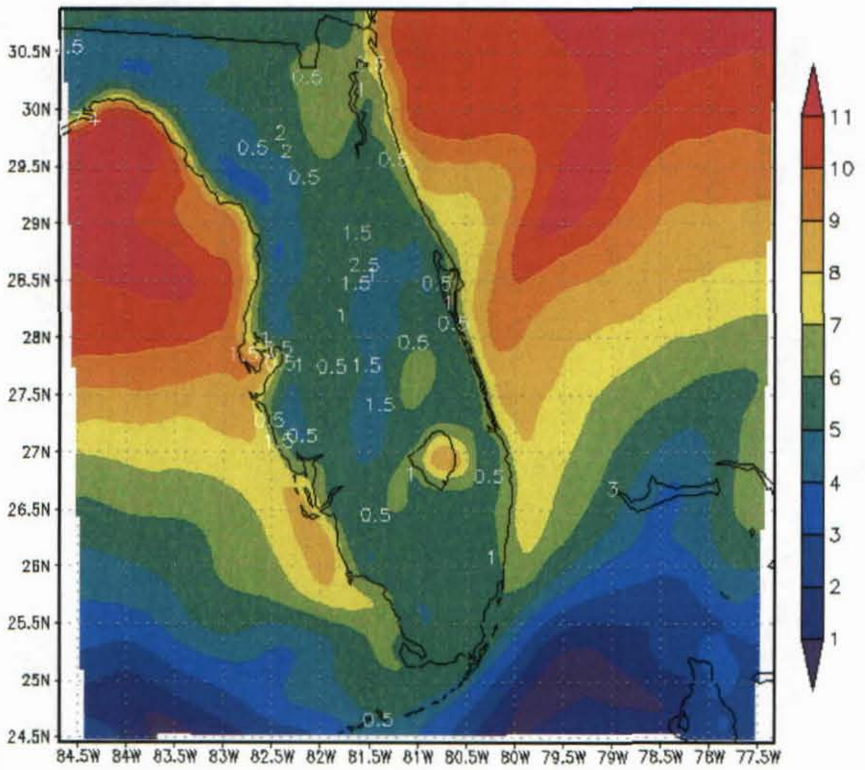


Figure 3. Wind speed (m s^{-1}) for the ADAS analysis (a) with assimilated observations vs. (b) no observations (essentially the RUC 13 km background field) at 0600 UTC 13 August 2010 over the default Florida LDIS domain. Wind speed observations from a subset of the MADIS surface data set are shown in white.

4.1.7 Soil Temperature and Moisture Initialization

The initialization of the soil and moisture variables is created by the **ARPSSOIL** program. Soil temperatures are obtained from a background model forecast and are interpolated to the ARPS grid while the soil moisture variables are initialized by using the Antecedent Precipitation Index (API) scheme available in the **ARPSSOIL** program. The API-initialized soil moisture is based on a first-guess value from a background model and a long-term integration of daily rain gauge measurements. The initial soil moisture is determined by using the rain gauge measurements from each observation to adjust the ratio between the saturation and wilting points for each soil type at each grid point (Case 2006b). The first-guess field is set to 50% of the particular soil type's saturation value at 150 days prior to the current date. It is then adjusted each day based on the rainfall amounts recorded at nearby rain gauges and the evapo-transpiration rate based on the time of year (Case 2006b).

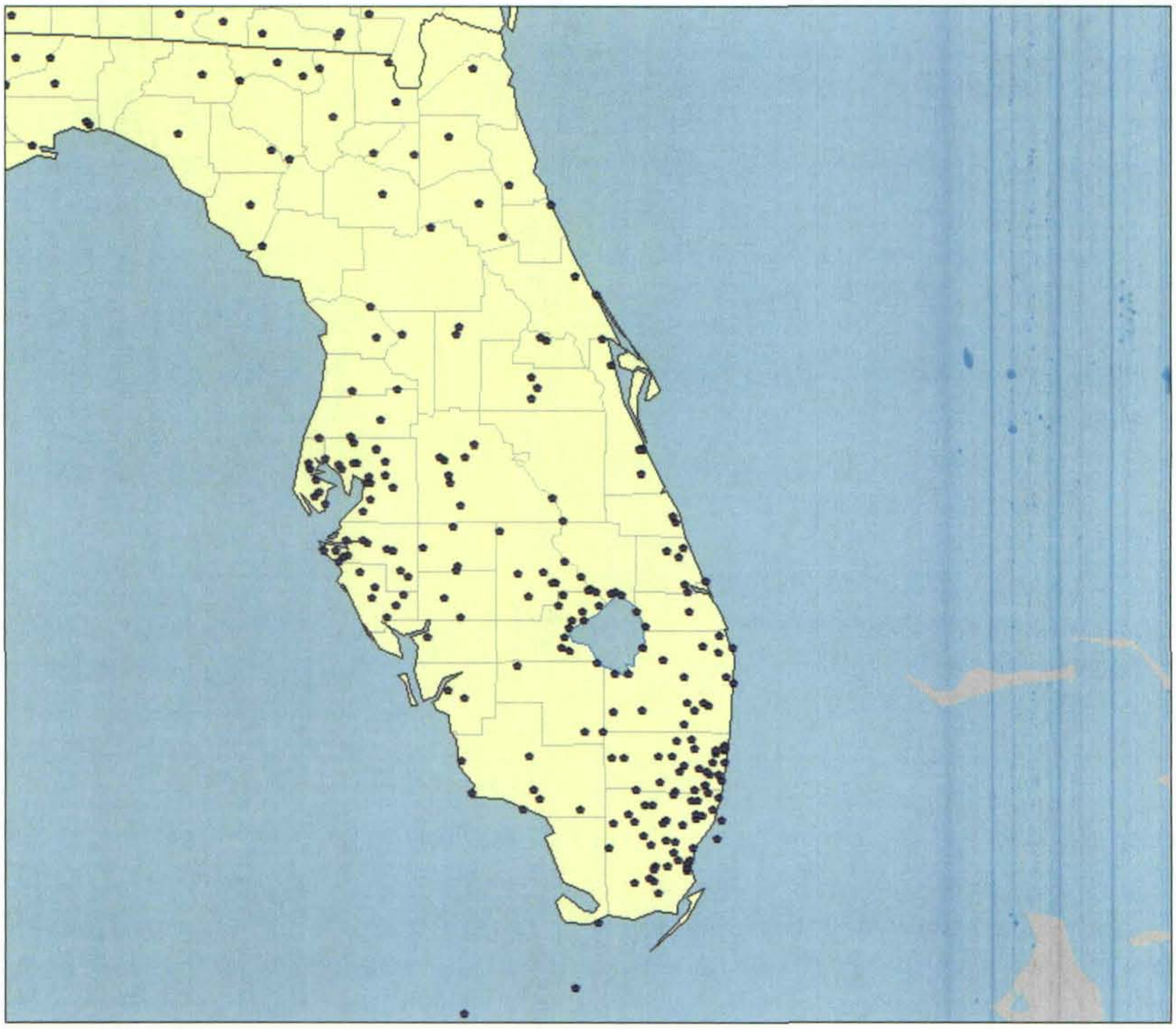


Figure 4. Map of the default Florida LDIS domain and distribution of NCEP daily rain gauge measurements across the peninsula used to initialize soil moisture in the ARPS/ADAS system.

The Perl script *run_arpssoil.pl* runs the **ARPSSOIL** program and is invoked by the master script *run.adas.wrf.cycle.pl*. The Perl script reads user configured values from the configuration file and domain information from the file *parms.txt*. User-supplied arguments include the soil initialization run name, the input directory containing the rain gauge data, and the output directory (see Table 4 in Appendix A). The

ARPSSOIL program requires raw model forecast files to initialize the soil and moisture variables. The same model used to create the first-guess field is used to initialize the soil and moisture variables (RUC or NAM). Two files are generated from the General Meteorological Package (GEMPAK) program **gdlist**, which creates a 2-D listing of near-surface soil temperatures (*sltk.5cm.yearmoddhhmm*) and deep soil temperatures (*sltk.40cm.yearmoddhhmm*). These files are then symbolically linked to generic file names that were hard-coded into the previously AMU-modified source code for **ARPSSOIL**: *sltk.5cm* → *sltk.5cm.yearmoddhhmm* and *sltk.40cm* → *sltk.40cm.yearmoddhhmm*. The soil moisture variables are determined using the API scheme described above. Daily rain gauge measurements are obtained from NCEP. The NCEP rain gauge data files contain ASCII data of station latitude, longitude, 24-hour rainfall total in inches, and station identification. These files are located in a user-specified directory listed in the configuration file and have a naming convention of *usa-dlyprcp-yearmodd*. Figure 4 shows the rain gauge locations across Florida. The arguments read and created by *run_arpssoil.pl* are incorporated into the arps5.x.x/input/*arpssoil.input* file using a series of substitutions using a hash list and the subroutine arps5.x.x/scripts/*mkinput.pm*. The new input file is named *soil.input*. The program arps5.x.x/bin/**arpssoil** is then run. The data files are written to the user-specified output directory and are of the form

<soil_runname>.yearmoddhhmm.bin000000 and *<soil_runname>.yearmoddhhmm.bingrdbas*.

4.2 Model Integration and Post-processing

The model integration portion of the forecast cycle consists of running either the ARPS or WRF model in parallel on all the compute-node processors and post-processing the model output as it becomes available. There are two possible runtime configurations: ADAS initializing a full ARPS prediction or ADAS producing an initial condition for the ARW or WRF EMS software.

4.2.1 ARPS Model

The complete ARPS modeling system consists of data ingest, quality control, and objective analysis packages, as well as radar parameter retrieval and data assimilation procedures, the prediction model, and post-processing packages. Details about the NWP component of the ARPS modeling system can be found in Section 2.2.

The Perl script *run_arps_mpi.pl* runs the ARPS model and is invoked by the master script *run.adas.wrf.cycle.pl*. The Perl script reads user configured values from the configuration file (see Table 4 in Appendix A) and domain information from the file *parms.txt*. If the ARPS model is being run, the 3-D data file generated by the **ARPSSOIL** program is used to initialize the model. The ARPS model also requires boundary condition files, terrain data, and surface data to run. The ARPS model is set up to run in parallel on a Linux cluster. The arguments read and created by *run_arps_mpi.pl* are incorporated into the arps5.x.x/input/*arps.input* file using a series of substitutions from a hash list and the subroutine arps5.x.x/scripts/*mkinput.pm*. The new file is named *arps_mpi.input*. The program arps5.x.x/bin/**arps** is then run. While the model is running, the model status is set to "RUNNING" and is stored in the file *status.txt*. The ARPS data files are written to the user-specified output directory and are of the form

yearmoddhhmm.binffffff.

The ARPS input and output files (*arps_mpi.output*) are written to the user-specified output directory. Once the ARPS model integration is complete, "RUNNING" is replaced with "COMPLETE" in the *status.txt* file.

4.2.2 ARPS Model Post-processing

The Perl script *postprocess.pl* processes the ARPS model output and is invoked by the script *run_arps_mpi.pl*. The Perl script reads user configured values from the configuration file and domain information from the file *parms.txt*. The post-processing script runs in the background and continually monitors the user-specified ARPS model output directory for new forecast data files while the *status.txt* file indicates that the model is running. Once new data are identified, the post-processing script launches into the routines **ARPS2NCDF** and **ARPS2GEM** (described below). Once the file *status.txt* indicates that the model run is complete, the post-processing script exits.

The Perl script *run_arps2ncdf.pl* processes the ARPS model output and is invoked by the script *postprocess.pl*. The Perl script reads user-configured values from the configuration file and domain information from the file *parms.txt*. The **ARPS2NCDF** program searches for the required ARPS model forecast files based on the model initialization time and availability of forecast data. The script establishes the appropriate times and input parameters for the **ARPS2NCDF** program that converts the forecast data into netCDF files for display in AWIPS. The arguments read and created by *run_arps2ncdf.pl* are incorporated into the arps5.x.x/input/*arps2ncdf.input* file using a series of substitutions from a hash list and the subroutine arps5.x.x/scripts/*mkinput.pm*. The new file is named *arps2ncdf.input*. The program arps5.x.x/bin/**arps2ncdf** is then run. The netCDF files are written to the user-specified output directory and are of the form

yearmoddhhmm.binffffff.ncd,

where "*ncd*" stands for netCDF format. The forecast variables stored in the netCDF files are available at 19 pressure levels in 50 mb increments from 1000 mb to 100 mb and at the surface. The grids stored in the netCDF files include all primary meteorological variables, including certain derived quantities such as forecast radar reflectivity and echo tops. The **ARPS2NCDF** input and output files (*arps2ncdf.output*) are written to the user-specified ARPS model output directory. Both files are over-written each time a forecast file is processed.

The Perl script *run_arps2gem.pl* processes the ARPS model output and is also invoked by the script *postprocess.pl*. The Perl script reads user-configured values from the configuration file and domain information from the file *parms.txt*. The **ARPS2GEM** program searches for the required ARPS model forecast files based on the model initialization time and availability of forecast data. The script establishes the appropriate times and input parameters for the **ARPS2GEM** program and converts data into GEMPAK visualization and display files. The arguments read and created by *run_arps2gem.pl* are incorporated into the arps5.x.x/input/*arps2gem.input* file using a series of substitutions from a hash list and the subroutine arps5.x.x/scripts/*mkinput.pm*. The new file is named *arps2gem.input*. The program arps5.x.x/bin/**arps2gem** is then run. The GEMPAK files are written to the user-specified output directory and are of the form

yearmoddhhmm.binffffff.gem,

where "*gem*" stands for a GEMPAK grid file. The forecast variables stored in the GEMPAK grid files are available at 19 pressure levels in 50-mb increments from 1000 mb to 100 mb and at the surface. The grids stored in the GEMPAK files include temperature, dew point, height (pressure), u- and v-winds, specific humidity, cloud and precipitation mixing ratios, cloud ceilings and cloud-top heights, and cloud fraction. The **ARPS2GEM** input and output files (*arps2gem.output*) are written to the user-specified ARPS model output directory. Both files are over-written each time a forecast file is processed.

4.2.3 WRF Model

The ADAS program can produce an initial condition file to initialize the ARW model (used within the WRF EMS software or on its own) for the main model integration. The NMM dynamical core of the WRF model can be run, however, a hot-start initialization is not possible since the cloud and precipitation microphysics derived by the ADAS cloud analysis scheme cannot currently be initialized into the NMM core due to limitations in the WRF code. Details about the WRF modeling system are in Section 2.3.

4.2.3.1 Preparing ARPS Data for WRF

The program **ARPS4WRF** converts the ARPS initialization data to the grid format of either the ARW or NMM core of the WRF system. It works in conjunction with the WRF Preprocessing System (WPS) program **geogrid.exe** and the **real.exe** program from the WRF software package and replaces the WPS program **metgrid.exe**. The program **geogrid.exe** creates the static fields and grid domains for the WRF model while **metgrid.exe** horizontally interpolates the data. The **real.exe** program uses the output from **metgrid.exe** or **ARPS4WRF** to create the model initial and boundary condition files for the WRF model integration. The **ARPS4WRF** program requires that the data file produced by **geogrid.exe** already exists, however, a new file only needs to be created when the grid dimensions, location, or resolution of the domain changes.

The Perl script *run_arps4wrf.pl* runs the **ARPS4WRF** program and is invoked by the master script *run.adas.wrf.cycle.pl*. The Perl script reads user configured values from the configuration file and domain information from the file *parms.txt*. User-supplied arguments include the length of the WRF forecast, specification of dynamical core, maximum number of domains to process, output directory, model initialization hour/minute, WRF model and post-processing software directory, and the post-processing working directory (see Table 4 in Appendix A). The 3-D data file generated by the **ARPSSOIL** program is used to create the initial condition file and, if needed, background output data files from the **EXT2ARPS** program are used to create the boundary condition files. The arguments read and created by *run_arps4wrf.pl* are incorporated into the arps5.x.x/input/*arps4wrf.input* file using a series of substitutions from a hash list and the subroutine arps5.x.x/scripts/*mkinput.pm*. The new file is named *arps4wrf.input*. The program arps5.x.x/bin/**arps4wrf** is then run. The WRF initialization files are written to the user-specified output directory and are of the form

met_em.dxx.year-mo-dd_hh:mm:ss.nc,

where *xx* is the domain of interest and "*nc*" stands for netCDF format. The **ARPS4WRF** input and output files (*arps4wrf.output*) are written to the user-specified output directory.

4.2.3.2 Creating the GRIB File Using WRF and WRF Post-processor Software

The output data files from **ARPS4WRF** can then theoretically be used to directly initialize the WRF model. However, the current ARPS/ADAS system does not include enough soil temperature/moisture variables in its output to properly initialize the WRF model. Therefore, the WRF model must be run for 30-seconds to write an initial condition file that is post-processed to create a pressure coordinate GRIB file that can then be ingested by WRF as the ADAS initial condition. There are two ways to create the GRIB file: 1) using the WRF and WRF post-processor software or 2) using the WRF EMS software.

To use the WRF and WRF post-processor software, the master script *run.adas.wrf.cycle.pl* can be set to automatically call the Perl script *run_wrf_post.pl*. The Perl script reads user-supplied values from the configuration file and domain information from the file *parms.txt*. User-supplied arguments include specification of dynamical core, output directory, boundary condition update interval, number of horizontal and vertical grid points in the domain, the horizontal grid-spacing, WRF model and post-processing software directory, and the post-processing working directory (see Table 4 in Appendix A). The script will find the appropriate initialization files from the **ARPS4WRF** program based on the model initialization time. The arguments read and created by *run_wrf_post.pl* are incorporated into the file *namelist.input* using a series of substitutions from a hash list and the subroutine arps5.x.x/scripts/*mkinput.pm*. The new file is copied to the WRF model directory. The script then runs the WRF **real.exe** program and the **wrf.exe** program for 30 seconds and then kills the process. The WRF forecast file is written to the WRF model directory.

Next, the WRF post-processor is run to create the GRIB file that is then used in WRF EMS as the ADAS initial condition. The script *run_wrf_post.pl* invokes the *run_wrfpost_minutes* script located in the <topdir>/WPPV3/scripts directory (*topdir* is a user-specified argument in the configuration file). This script is a modified version of the script provided by NCEP as part of the official NMM release. The *run_wrfpost_minutes* script runs the **wrfpost.exe** conversion program located in the

<topdir>/WPPV3/exec directory and converts the WRF forecast data into a pressure coordinate GRIB file. The GRIB file is copied to the user-specified **ARPS4WRF** output directory and is of the form

yymoddhhmm_arw_dxx.grb1f000000.

4.2.3.3 Creating the GRIB file using WRF EMS Software

To use the WRF EMS software, a new WRF domain must be created using the WRF EMS domain wizard that matches the size and location of the ADAS analysis domain (see the WRF EMS User's Guide for instructions on using the domain wizard). A new WRF domain must only be created if the size or location of the ADAS domain changes. The master script *run.adas.wrf.cycle.pl* can be set to automatically call the Perl script *run_wrfems_post.pl*. The Perl script reads user-supplied values from the configuration file and domain information from the file *parms.txt*. User-supplied arguments include the output directory and the WRF EMS software directory (see Table 4 in Appendix A). The script will find the appropriate initialization files from the **ARPS4WRF** program based on the model initialization time. The script invokes a bash shell script, *emsrun.sh*, that runs the **real.exe** program and the **wrf.exe** program within the WRF EMS software for 30 seconds and then kills the process. Another bash shell, *emspost.sh*, is then invoked that post-processes the WRF forecast data to create the pressure coordinate GRIB file that is used as the ADAS initial condition. The GRIB file is copied to the user-specified **ARPS4WRF** output directory and is of the form

yymoddhhmm_arw_dxx.grb1f000000.

To ingest the pressure coordinate GRIB files into WRF, the AMU previously created two "Vtable" files specific to the variables found in the ADAS data set, *Vtable.ADASARW* and *Vtable.ADASNMM*. The Vtable defines the characteristics for each variable in the external grid data set that is being interpolated to the WRF grid. In addition, the AMU created two configuration files (*adasarw_gribinfo.conf* and *adasnmm_gribinfo.conf*) for the ADAS initialization files in which the file-naming convention, directory location, and time attributes were defined.

5. Modifications to the Official ARPS Code

The AMU made modifications to the official ARPS code so that the ARPS/ADAS system could be customized to the needs of NWS MLB and SMG. Modifications were kept to a minimum so that updating to the most recent version of the ARPS code could be done with minimal effort. Table 2 lists the altered source code program names and directories in which they reside, and changes that were made. All changes made to the Fortran source code were documented in each individual program.

Table 2. List of modified Fortran programs in the official ARPS source code, the directory in which they reside, and the changes made to the code.

Program Name	***Changes Made***
/arps5.x.x/src/external/g2lib/*Makefile*	To use GRIB2 model background data: • variable FDEFS was uncommented • set environment variables JASPER and PNGLIBPATH before compiling ext2arps (see Appendix A, Table 3)
/arps5.x.x/src/adas/*prepua.f90*	• added code to incorporate MADIS profiler data • added code to incorporate satellite sounding data • added code to incorporate KSC/CCAFS wind tower network data
/arps5.x.x/src/adas/*prepradar.F*	• minor editing to code
/arps5.x.x/src/adas/*rdprofiles.f90*	• incorporated previously written subroutine RDTOWER to read KSC/CCAFS wind tower network data • other small variable changes
/arps5.x.x/src/adas/*cldinsert.f90*	• incorporated previously modified code – See Case (2006b)
/arps5.x.x/input/*arps.input*	• increased number of possible radars files • increased number of possible single level data files and source files • increased number of possible external data files
/arps5.x.x/include/*adas.inc*	• increased max number of vertical levels in sounding (nz_ua = 200) • increased number of sources of single level data (nsrc_sng = 9)
/arps5.x.x/src/wrf2arps/*wrf2arps.f90*	• commented out 'if' statement to get a base-state and grid file at every forecast time

6. Modification of ADAS GUI

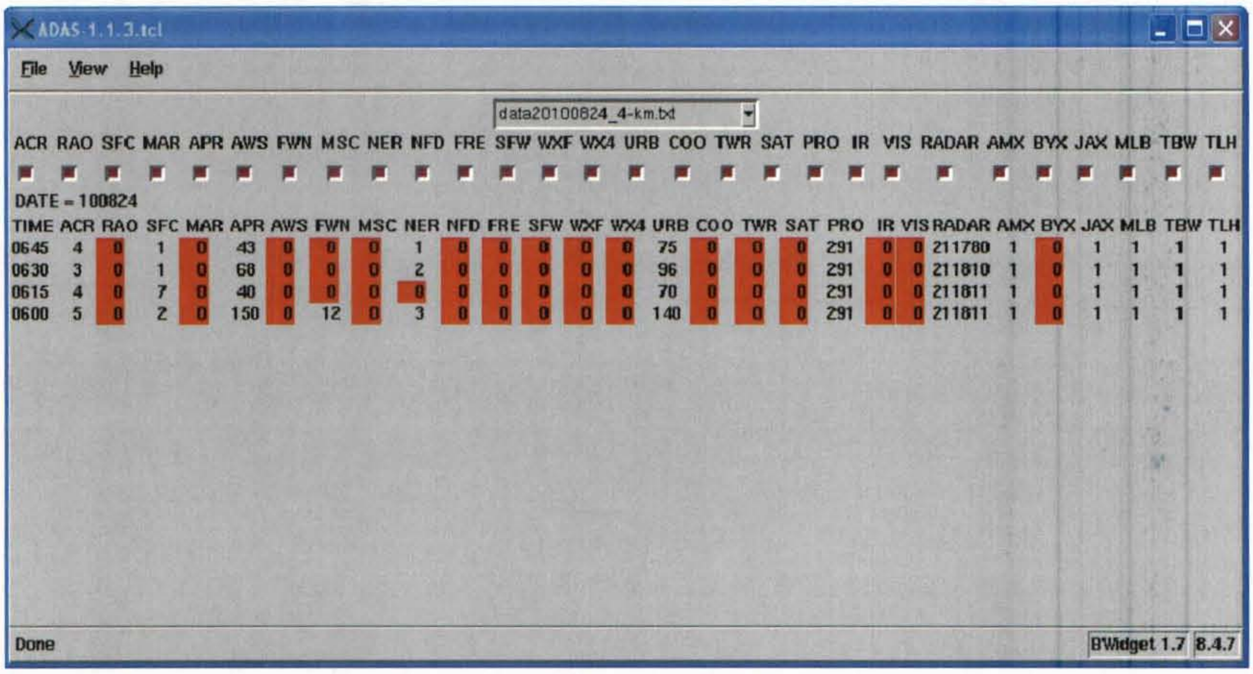
One of the goals of this task was to update the previously developed ADAS GUI. The original GUI was developed by the AMU in 2004 and allows forecasters to quickly and easily interact with ADAS to maintain or improve the integrity of each 15 minute analysis cycle. The intent was to allow forecasters to monitor and manage the observational data streams ingested by ADAS without having prior ADAS expertise. The GUI was created using the Tool Command Language and its associated toolkit (Tcl/Tk). Please refer to Case and Keen (2006) for instructions on how to run the GUI.

Information about the data ingested by ADAS is passed to the GUI from the output of the ADAS analysis. The AMU rewrote the previous shell script that extracted information about the observations using Perl. The new script, *run_datatype_realtime.pl*, extracts information from the ADAS output file about the numbers of each observation type that are analyzed by ADAS. It is invoked by the master script *run.adas.wrf.cycle.pl*. The Perl script reads user configured values from the configuration file and domain information from the file *parms.txt*. The script searches for the number of surface, upper air, satellite, and radar observations assimilated by the ADAS program. The script has been updated to take into account the new MADIS data now assimilated by ADAS. These values are written out to a data file in the user-specified ADAS output directory and is of the form

datayearmodd_<hres>-km.txt

where *hres* is the horizontal grid-spacing of the domain. New data are appended to the output data file each time the ADAS program is run. The file is purged at the end of every day. The ADAS GUI accesses the output data file and displays the observation information on its Data Availability Display (Figure 5).

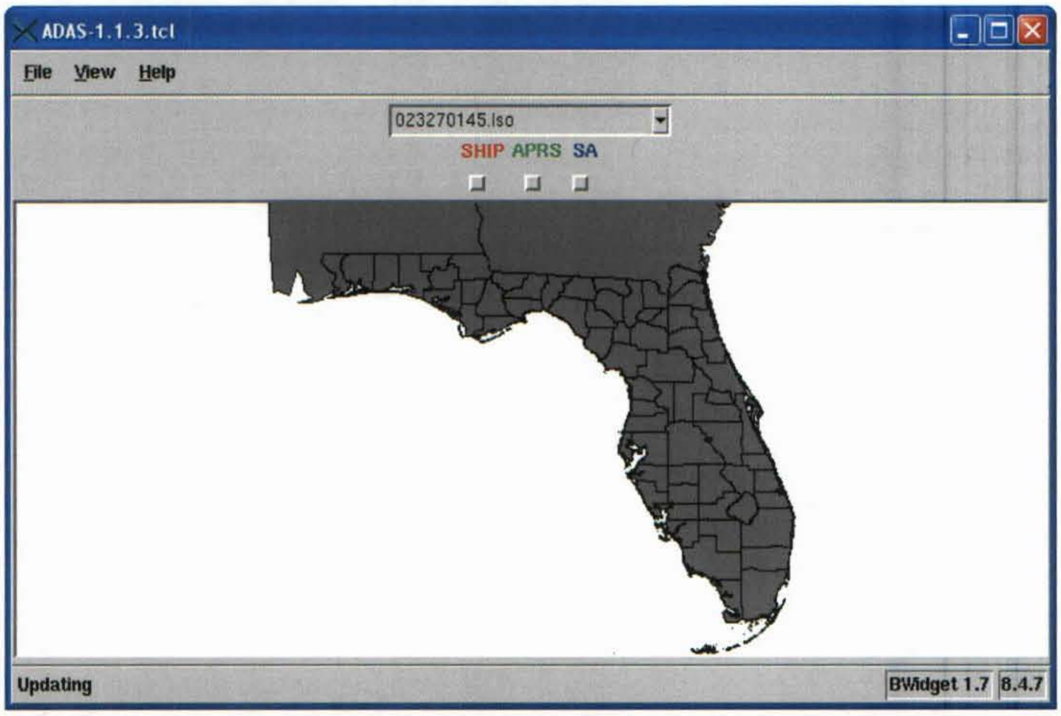
The AMU also created a new map background to replace the existing one using the MapServer software for the ADAS GUI. The new map includes the state boundaries, Florida county boundaries, and significant lakes and rivers in Florida. Figure 6 shows the existing and the new map backgrounds for Florida and the surrounding areas.


ADAS-1.1.3.tcl
File View Help
data20100824_4-km.txt
ACR RAO SFC MAR APR AWS FWN MSC NER NFD FRE SFW WXF WX4 URB COO TWR SAT PRO IR VIS RADAR AMX BYX JAX MLB TBW TLH
Done
BWidget 1.7
8.4.7

DATE = 100824

TIME	ACR	RAO	SFC	MAR	APR	AWS	FWN	MSC	NER	NFD	FRE	SFW	WXF	WX4	URB	COO	TWR	SAT	PRO	IR	VIS	RADAR	AMX	BYX	JAX	MLB	TBW	TLH
0645	4	0	1	0	43	0	0	0	1	0	0	0	0	0	75	0	0	0	291	0	0	211780	1	0	1	1	1	1
0630	3	0	1	0	68	0	0	0	2	0	0	0	0	0	96	0	0	0	291	0	0	211810	1	0	1	1	1	1
0615	4	0	7	0	40	0	0	0	0	0	0	0	0	0	70	0	0	0	291	0	0	211811	1	0	1	1	1	1
0600	5	0	2	0	150	0	12	0	3	0	0	0	0	0	140	0	0	0	291	0	0	211811	1	0	1	1	1	1

Figure 5. Data availability table displayed at GUI start-up. Red shading indicates data sources that do not have any observations being ingested by ADAS.

(a) 

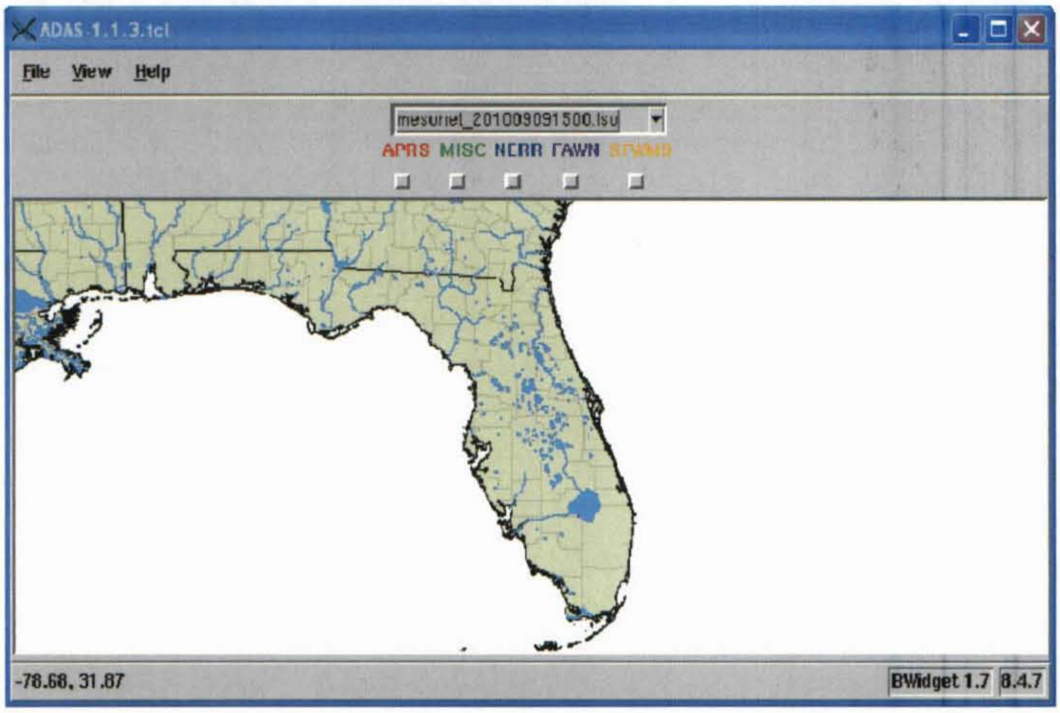
(b) 

Figure 6. (a) Existing ADAS GUI map background and (b) new map background of Florida and the surrounding areas.

7. Optimization of Error Statistics

The ADAS Bratseth method requires estimates of observation errors, which are specified in the observation error file. These error files are found in the directory /arps5.x.x/data/adas. Each data source has its own error characteristics that have to be specified. In some cases, it is useful to break up the observations into separate source names to account for special error characteristics within a network. The specified errors include possible instrument error, round-off imprecision in reporting and archiving, and representativeness error in the observation. The Bratseth technique relies on the relative size of the error among the different types and the background model error. Larger (smaller) error assigned to a data source results in a smaller (larger) influence of that data on the resulting nearby grid points.

The AMU examined a set of control error parameters assigned to the observations and then varied them to determine how much the background field is modified by the observations with a different set of error parameters. This was done to find out if the ADAS analyses could be improved by optimizing the error parameters for the MADIS mesonet data. The first experiment involved reducing the observational error variances to half of their control values while the second doubled the error variances. The results from both experiments revealed that varying the error parameters for each data source has little impact on the ADAS analyses. The primary change that takes place is an increase (decrease) in magnitude of some of the observational "bulls eyes" due to the greater (lesser) weight assigned to the observations. Case (2006) found that varying the "xyrange" parameter in the ADAS namelist had a larger effect on correcting the ADAS analyses over much of the domain. In addition, multiple data sets available from MADIS do not have enough hourly observations available to fully optimize the error parameters. Therefore, the values of error that were used in the previous version of the LDIS scripts were unaltered and new error parameter files created for the MADIS data use the same error statistics.

8. Summary and Conclusions

The NWS MLB and SMG have used an LDIS as part of their forecast and warning operations since 2000. The original LDIS was developed by the AMU and has undergone subsequent improvements. The 3-D analyses are delivered to forecasters every 15 minutes across the peninsula of Florida with the intent to generate products that enhance short-range weather forecasts issued in support of NWS MLB and SMG operational requirements within East-Central Florida. The current LDIS uses the ARPS/ADAS package as its core, which integrates a wide variety of national, regional, and local observational data sets. It assimilates all available real-time data within its domain and is run at a finer spatial and temporal resolution than current national- or regional-scale analysis packages. As such, it provides local forecasters with a more comprehensive understanding of evolving fine-scale weather features.

The AMU met all requirements including updating the NWS MLB/SMG LDIS with the latest version of ADAS, incorporating new sources of observational data, and rewriting and modifying the AMU-developed shell scripts written to govern the system in the Perl scripting language. In addition, the previously developed ADAS GUI was updated. The AMU installed and configured the new LDIS scripts on the NWS MLB and SMG Linux systems for operational use. In addition, the AMU provided training on how to set up and run the scripts in an operational setting. These upgrades will result in more accurate depictions of the current local environment to help with short-range weather forecasting applications, while also offering an improved initialization for local versions of the WRF model.

Appendix A

LDIS User's Guide

This User's Guide provides a detailed description of the steps needed to successfully install and download the LDIS scripts and supporting external libraries and packages. In addition, this guide provides a description of the components of the user configurable file, *ADAS_input.config*, in which all tunable parameters used to run the LDIS reside. It also describes various options for running the LDIS scripts. The parameter name, the description of the parameter, and the default settings are listed for each LDIS program. The following information can also be found in the *README.config* file included in the LDIS configuration code package.

1. External Libraries and Packages

A series of external libraries and packages need to be downloaded and installed before the LDIS can be run. The external packages are required by the ARPS software to successfully compile each of the system components. Section 1h or 1i is optional. The user can download and install either the WRF modeling software and the NCEP WRF post-processing software *or* the WRF EMS modeling software.

a. Download JPEG image compression software (must use version 6: jpegsrc.v6b) written by the Independent JPEG Group. Version 6 of the JPEG software can be found at: http://dir.filewatcher.com/d/GNU/Other/jpegsrc.v6b.tar.gz.613261.html.Install using the following instructions. This software is required to build HDF (see bullet 3).
    1. ./configure –prefix=<*path_to_jpeg_software*>/jpeg-6b
    2. make
    3. make test
    4. make install

b. Download zlib software (current version: zlib-1.2.5) from the official zlib website (www.zlib.net/). Install using the following instructions. This software is required to build HDF (see bullet 3).
    1. ./configure
    2. make test
    3. make install prefix=<*path_to_zlib_software*>/zlib-1.2.5

c. Download HDF software (current version: hdf-4.2.5) from The HDF Group website (www.hdfgroup.org). Install using the following instructions. This software is needed to compile ARPS.
    1. on the command line enter: export LDFLAGS="-L/usr/lib -lm"
    2. ./configure –with-zlib=<*path_to_zlib_software*>/zlib-1.2.5 –with-jpeg=<*path_to_jpeg_software*>/jpeg-6b –prefix=<*path_to_hdf_software*>/hdf-4.2.5
    3. gmake >& gmake.out
    4. gmake check >& check.out
    5. gmake install
    6. Add entry for HDFPATH in *.bashrc* file (see Table 3).

d. Download netCDF software (current version: netcdf-4.1.1) from the Unidata website (www.unidata.ucar.edu/software/netcdf). Install using the following instructions. This software is needed for some ARPS programs.
    1. ./configure –prefix=<*path_to_netcdf_software*>/netcdf-3.6.1
    2. make check
    3. make install
    4. Add entry for NETCDF in *.bashrc* file (see Table 3).

e. Download JasPer software (current version: jasper-1.900.1) from the JasPer Project Home Page (www.ece.uvic.ca/~mdadams/jasper). Install using the following instructions. This software is needed to use GRIB2 background data in ext2arps.
   1. ./configure –prefix=<*path_to_jasper_software*>/jasper-1.900.1
   2. make
   3. make install
   4. Add entry for JASPER in *.bashrc* file (see Table 3).

f. Download the PNG reference library software (current version: libpng-1.4.3) from the Portable Network Graphics website (www.libpng.org/pub/png/libpng.html). Install using the following instructions. This software is needed to use GRIB2 background data in ext2arps.
   1. ./configure –prefix=<*path_to_libpng_software*>/libpng-1.4.3
   2. make check
   3. make install
   4. Add entry for PNGLIBPATH in *.bashrc* file (see Table 3).

g. Download GEMPAK binaries (current version: GEMPAK 5.11.4) from the Unidata website (www.unidata.ucar.edu/software/gempak). Install using the following instructions. This software is needed for the ARPS soil and post-processing programs.
   1. Edit *Gemenviron.profile*.
   2. Source *Gemenviron.profile* in *.bashrc* (see Table 3).

h. (optional – complete this section or 1i) Download WRF modeling software (current version: WRF Version 3.2) from the WRF Model User's Page (www.mmm.ucar.edu/wrf/users). Install using the following instructions.
   1. ./configure
   2. Add –DRUC_CLOUD \ to ARCHFLAGS in *configure.wrf*. This must be done so the model microphysics are not zeroed out for the hot-start initialization.
   3. compile em_real

   Download NCEP WRF post-processor software (current version: WPPV3.2) from the WRF-NMM User's Page (www.dtcenter.org/wrf-nmm/users). Install using the following instructions.
   4. Add the *run_wrfpost_minutes.template* file (in WPPV3 directory of tar file) to WPPV3/scripts directory.
   5. Add the *wrf_cntrl.parm* to the parm directory of your working directory (see README.config).

i. (optional – complete this section or 1h) Download the WRF EMS modeling software (current version: Version 3.1) from the WRF EMS website (http://strc.comet.ucar.edu/wrf/index.htm). Install following the directions on the webpage.

2. External Data

The ARPS terrain and surface programs require external data sets to create the 2-D terrain and surface characteristics data sets used in the ARPS/ADAS system. Both programs convert the external data in ASCII format into unformatted binary direct-access files which are then used by the ARSTRN and ARPSSFC programs. The ARPS soil program requires NCEP rain gauge data to initialize the soil moisture variable using the API scheme available in the ARPSSOIL code.

a. Download and untar *arpstopo30.data.tar.gz* from the ARPS website (www.caps.ou.edu/ARPS/arpsdown.html). This data is needed for the terrain program.

b. Download and untar *arpssfc.data.tar.gz* from the ARPS website (www.caps.ou.edu/ARPS/arpsdown.html). This data is needed for the surface program.

c. Set up an anonymous ftp of the NCEP rain gauge data from ftp.emc.ncep.noaa.gov, under directory /mmb/gcp/precip/katz.
    1. Add file *usstns.list* included with the LDIS scripts to the /soil/gauge directory.

3. Comprehensive Perl Archive Network (CPAN) modules

CPAN is a large collection of Perl software and documentation called modules. Perl uses these external libraries of code to allow a single file to contain common routines used by several programs. Several Perl modules need to be installed to successfully run the LDIS scripts. Use the following instructions to configure the environment and install the modules (the instructions can also be found here: http://linuxgazette.net/139/okopnik.html).

a. Create a directory where the Perl modules will be located and append /lib to its name (ex., /home/user/modules/lib).

b. Enter the following in the *.bashrc* file (see Table 3). This is needed to use the mkinput.pm script available in the ARPS software and to use the Perl CPAN modules (see Section 3).
PERL5LIB=/<*path_to_CPANmodules*>/lib; export PERL5LIB
MANPATH=$MANPATH:/< *path_to_CPANmodules*>/man; export MANPATH.

c. Create the following directories (where 'modules' is the Perl module top directory specified in bullet a):
    1. mkdir –p ~/modules/lib
    2. mkdir –p ~/modules/man/man{1,3}

d. Configure the CPAN module.
    1. On the command line enter: perl –MCPAN –we 'shell'
    2. When the script asks for any extra arguments for *Makefile.PL*, supply it with the following list:
    LIB=~/modules/lib INSTALLSITEMAN1DIR=~/modules/man/man1
    INSTALLSITEMAN3DIR=~/modules/man/man3
    3. Set UNINST=0 when that question comes up during installation.
    4. If the CPAN shell was previously configured, it will need to be modified. Start the shell as above and issue the following commands at the 'cpan>' prompt:
    o conf makepl_arg "LIB=~/modules/lib
    INSTALLSITEMAN1DIR=~/modules/man/man1
    INSTALLSITEMAN3DIR=~/modules/man/man3" o conf make_install_arg UNINST=0
    o conf commit
    5. Install the Date::Pcalc and File::Util modules:
    perl –MCPAN –we 'install "Date::Pcalc"'
    perl –MCPAN –we 'install "File::Util"

4. Setting Up the MADIS Data Feed

The MADIS API is a library of Fortran subroutines that provides access to all of the observations and QC information in the MADIS database. Utility programs included in the API package are used to read station information, observations, and QC information for a single time and to dump them to an output text file. A subset of these files were altered and used to create ADAS-compatible files. The following instructions detail the setup and usage of this software.

a. Untar the madis-3.7.tar file included with the LDIS scripts in the /MADIS_software directory.

b. Replace src/*acarsdump.f*, *mapdump.f*, *npndump.f*, *raobdump.f*, *satsnddump.f*, and *sfcdump.f* with the new files located in the /MADIS_software directory.

c. Install the MADIS API software using step 3 from the instructions found here: http://madis.noaa.gov/doc/INSTALL.unix.

d. Add entry for MADIS_STATIC in *.bashrc* (see Table 3 – also see step 5 in above link - bullet c).

e. Add entry for MADIS_DATA in *.bashrc* (see Table 3 – also see step 5 in above link - bullet c).

f. Change path names in lines 6, 7, 9, 40 in *run_sfc.sh*, lines 6, 7, 9, 19 in *run_profiler.sh*, lines 7, 8, 10, 45, 76 in *run_raob.sh*, and lines 6, 7, 9, 32 in *run_acars.sh*.

g. Set up shell scripts to run in the cron – for example:

```
13,28,43,58  * * * * /home/user/MADIS_software/run_sfc.sh
10           * * * * /home/user/MADIS_software/run_raob.sh
12,27,42,57  * * * * /home/user/MADIS_software/run_acars.sh
42           * * * * /home/user/MADIS_software/run_profiler.sh
```

ADAS-compatible files are created in the /MADIS_software directory. Surface data is split into provider designations: METAR, mesonet, COOP, UrbaNet, and MARITIME.

Alternatively, mesonet CSV-formatted files split by data provider can be used in place of the mesonet output file created by the MADIS API program. In this case, the Perl script, *convertCSV.pl*, should be set to run in the cron at a user-specified interval. This script reads and rewrites the CSV-formatted mesonet data into a format ingestible by ADAS. The script takes all data within a 15 minute time window and specified horizontal domain and writes multiple files that contain all of the observations split by provider name. The output files are named according to the data provider and timestamp. The script also appends all data from all providers into one data file. The ADAS program will read the mesonet observations from either the separated files or the one appended file. The following instructions detail user-input needed to run the *convertCSV.pl* program.

h. Set MESODATA environment variable in *.bashrc* file (see Table 3). This defines the directory location of the MADIS CSV-formatted mesonet data.

i. Define the west, east, north, and south latitude/longitude boundaries of the domain of interest ($wlon, $elon, $nlat, and $slat variables in *convertCSV.pl*).

5. Installing the ARPS software

The compilation of all ARPS programs are orchestrated by a UNIX shell script, **makearps**. The script invokes a make command based on parameters supplied to it. Use the following instructions to compile all ARPS programs.

a. Replace /arps5.x.x/input/*arps.input* with the new file in the /altered_ARPS_code directory included with the LDIS scripts.

b. Replace /arps5.x.x/src/adas/*cldinsert.f90*, *prepua.f90*, *rdprofiles.f90*, and *prepradar.F* with the new files in the /altered_ARPS_code directory included with the LDIS scripts.

c. Replace /arps5.x.x/src/wrf2arps/*wrf2arps.f90* with the new file in the /altered_ARPS_code directory included with the LDIS scripts.

d. Replace /arps5.x.x/include/*adas.inc* with the new file in the /altered_ARPS_code directory included with the LDIS scripts.

e. Uncomment FDEFS variable in /arps5.x.x/src/external/g2lib/*Makefile*. This must be done in order to use GRIB2 background model data in EXT2ARPS.

f. Enter the command: makearps <options> all (where 'options' are the compiler options).

g. Compile 88D2ARPS, MCI2ARPS, ARPS4WRF, WRF2ARPS, ARPSTINTRP, ARPS2GEM, and ARPS2NCDF separately by replacing 'all' in the above command with the appropriate program name.

h. Add entry for ARPS in *.bashrc* file (See Table 3). This is needed to run the LDIS scripts.

Table 3. An example *.bashrc* file listing all required pathnames to successfully run the LDIS scripts and supporting software. Directory pathnames are user-specific.

```
# .bashrc

PATH=.:\
/bin:\
/usr/lib:\
/usr/local/mpich-1.2.7p1-pgi/bin:\
/usr/local/pgi/linux86/6.1/bin:\
/usr/local/pgi/linux86/6.1/lib:$PATH

export PATH

export LM_LICENSE_FILE=/usr/local/pgi/license.dat
export HDFPATH=/home/user/hdf-4.2.5
export NETCDF=/home/user/netcdf-4.1.1
export JASPER=/home/user/jasper-1.900.1
export PNGLIBPATH=/home/user/libpng-1.4.3/lib
export LD_LIBRARY_PATH=/home/user/libpng-1.4.3/lib:$LD_LIBRARY_PATH
export MADIS_STATIC=/home/user/ADAStask/MADIS_software/static
export MADIS_DATA=/usr/local/ldm/export
export MESODATA=/home/user/CSVMadisData
export ARPS=/home/user/ADAStask
. /home/user/GEMPAK5.11.1/Gemenviron.profile

PERL5LIB=/home/user/arps5.2.12/scripts:\
/home/user/cpan_modules/lib; export PERL5LIB
MANPATH=$MANPATH:~/home/user/cpan_modules/man; export MANPATH
```

6. Setting Up and Using the LDIS scripts

Each component of the LDIS is invoked by the master script *run.adas.wrf.cycle.pl*. The master script steps through all procedures of a single ARPS or WRF cycle. All tunable parameters are entered by the user into a configuration file named *ADAS_input.config*. Table 4 lists the different parameters in the configuration file, their default setting, and a description of the parameter for each of the LDIS components. The master script calls each of the LDIS components to run the complete modeling system. Details about each LDIS component can be found in Section 4 of this report. The following steps also need to be followed to successfully run the LDIS scripts.

a. Copy all error files from directory /error_files included with the LDIS scripts to /arps5.x.x/data/adas.

b. A data file containing the surface characteristics must be in the /wrfinit directory for ARPS4WRF to run. The file is generated with **geogrid.exe** in the WPS package or by using the command 'ems_prep –nometgrid –nodegrib' within the WRF EMS software.

c. Create directories /postprd and /parm in the ARPS4WRF working directory ('workdir' entry under ARPS4WRF parameters in configuration file – see Table 4). This step is only needed if using the WRF software and NCEP WRF post-processing software to create a 0-hr GRIB forecast file. If using the WRF EMS software, this step is not needed.

d. To change default FL radars in the 88D2ARPS program: replace KAMX, KBYX, KJAX, KMLB, KTBW, and KTLH on line 103 of *run_88d2arps.pl* with desired radar sites.

e. Set up LDIS scripts to run in the cron – for example:

```
13,28,43,58 * * * * /home/user/do_perl /home/user/ADAS/run.adas.wrf.cycle.pl >&
/home/user/ADAS/run-`date +\%Y-\%m-\%d-\%H:\%M`.out
```

Table 4. A description of the tunable parameters in the configuration file, *ADAS_input.config*. The configuration file is used by each component of the LDIS Perl scripts to run the complete modeling system.

Parameter	***Default Setting***	***Description***
# General Parameters - User chosen directory pathnames for each step in the ARPS/ADAS process.		
ARPS_dir	/home/user/arps1	Directory where the ARPS software resides.
Main_dir	/home/user/ADAS	Main directory where all ARPS subdirectories (listed below) are located.
terrain_dir	/home/user/ADAS/terrain	ARPSTRN working directory.
surface_dir	/home/user/ADAS/surface	ARPSSFC working directory.
ext2arps_dir	/home/user/ADAS/ext2arps	EXT2ARPS working directory.
wrf2arps_dir	/home/user/ADAS/wrf2arps	WRF2ARPS working directory.
arpstintrp_dir	/home/user/ADAS/arpstintrp	ARPSTINTRP working directory.
mci2arps_dir	/home/user/ADAS/mci2arps	MCI2ARPS working directory.
radar_dir	/home/user/ADAS/88d2arps	88D2ARPS working directory.
adas_dir	/home/user/ADAS/adas	ADAS working directory.
soil_dir	/home/user/ADAS/soil	ARPSSOIL working directory.
wrfinit_dir	/home/user/ADAS/wrfinit	ARPS4WRF working directory.
arpsmpi_dir	/home/user/ADAS/arps_mpi	ARPS MPI working directory.
# General Parameters - Which ARPS/ADAS steps are we doing?		
do_terrain	y	Run the ARPSTRN program.
do_arpssfc	y	Run the ARPSSFC program.
do_ext2arps	y	Run the EXT2ARPS program. Set this to y if using RUC or NAM as background model.
do_arpstintrp	y	Run the ARPSTINTRP program. Set this to y if background data is needed between model forecast output times (for example, if running 15 minute ADAS analysis). NOTE: This can be set to 'y' regardless of whether it is needed or not. The scripts will determine whether the program should be run.
do_wrf2arps	n	Run the WRF2ARPS program. Set this to y if using WRF forecast output for the background model.
do_mci2arps	y	Run the MCI2ARPS program.
do_88d2arps	y	Run the 88D2ARPS program.
do_adas	y	Run the ADAS program.
do_arpssoil	n	Run the ARPSSOIL program.
do_arpsmpi	n	Run the ARPS MPI program.
do_wrfinit	n	Run the ARPSSFC program.

Table 4. A description of the tunable parameters in the configuration file, *ADAS_input.config*. The configuration file is used by each component of the LDIS Perl scripts to run the complete modeling system.

Parameter	***Default Setting***	***Description***
# General Parameters - ARPS model information used in all programs.		
version	5.2.12	Current working version of ARPS software.
dx	4000	Grid spacing in x-direction (in meters).
dy	4000	Grid spacing in y-direction (in meters).
dz	450	Average spacing of vertical levels (in meters).
dzmin	20	Minimum spacing of vertical levels near surface (in meters).
nx	177	Number of grid points in the west-east (x) dimension.
ny	179	Number of grid points in the north-south (y) dimension.
nz	40	Number of grid points in the vertical (z) dimension.
nprocx	2	Number of parallel processors in the x dimension.
nprocy	2	Number of parallel processors in the y dimension.
tstop	0	Length of ARPS/WRF forecast in seconds. = 0 if running ADAS analysis AT TOP OF THE HOUR only or if running ARPS4WRF to produce only the initial condition file $\geq$ 3600 if running ADAS analysis at < 1 hour interval or running ARPS/WRF model
center_lat	27.70	Center latitude of the ARPS grid.
center_lon	-81.00	Center longitude of the ARPS grid.
# terrain (ARPSTRN) parameters		
terrain_runname	FL_terrain_4km	Run name used to identify this job.
dir_trndata	/home/user/DATA	Directory where raw terrain data resides.
outdir_trn	home/user/terrain	User chosen directory into which ARPSTRN output files are written.
# surface (ARPSSFC) parameters		
surface_runname	FL_surface_4km	Run name used to identify this job.
dir_sfcdata	/home/user/DATA	Directory where raw surface data resides.
outdir_sfc	/home/user/surface	User chosen directory into which ARPSSFC output files are written.

Table 4. A description of the tunable parameters in the configuration file, *ADAS_input.config*. The configuration file is used by each component of the LDIS Perl scripts to run the complete modeling system.

Parameter	***Default Setting***	***Description***
# background model (EXT2ARPS) parameters - These are parameters for boundary and/or initial conditions. These parameters MUST be set to run EXT2ARPS.		
ext2arps_runname	FL_ext2arps	Run name used to identify this job. The output files will have a timestamp appended to the ext2arps run name.
raw_data_dir	/home/user/DATA/RUC	Directory where background model data resides.
outdir_mod	/home/user/ext2arps	User chosen directory where EXT2ARPS output files are written.
model_type	ruc	Background model for initial and/or boundary conditions. Options are: 'ruc' or 'nam'. If using WRF for background model, use wrf2arps option below. For HRRR data use 'ruc'.
vert_coord	hybrid	ONLY FOR RUC DATA. Specify vertical coordinate of background model data. Options are: 'pressure' or 'hybrid'.
data_type	full	ONLY FOR NAM DATA. Specify whether using NAM tile data or full NAM CONUS data (OR a subset of it). Options are: 'tiles' or 'full'.
model_res	13	Specify the horizontal grid spacing of the background model data.
model_prefix	ruc13	Prefix string of background model file name (everything before the first .).
bcupdate	3600	Specify the forecast output interval (in seconds). = 3600 if using hourly data = 10800 if using 3 hourly data
extdfmt	1	Specify whether the ARPS or NCEP naming convention is being used for the background model data. Currently there is no NCEP option for RUC data. Choose 0 for NCEP, 1 for ARPS.

Table 4. A description of the tunable parameters in the configuration file, *ADAS_input.config*. The configuration file is used by each component of the LDIS Perl scripts to run the complete modeling system.

Parameter	***Default Setting***	***Description***
These are parameters for initial conditions only (if different model is used for the initial conditions). If the same model is used for both initial and boundary conditions, simply set ic_model_type to the same value as model_type (above) and forget the rest. Otherwise fill in the parameters following the same directions from above.		
ic_raw_data_dir		
ic_model_type	ruc	
ic_vert_coord		
ic_data_type		
ic_model_res		
ic_model_prefix		
ic_bcupdate		
ic_extdfmt		
# background WRF model (WRF2ARPS) parameters		
forecast_int	15	WRF forecast output interval in minutes.
wrf_data_dir	/wrfems/runs/ADASARW/wrfprd	WRF forecast output data directory.
wrf2arps_runname	FL_wrf2arps	Run name used to identify this job. The output files will have a timestamp appended to the wrf2arps run name.
outdir_wrf2arps	/home/user/ADAS/wrf2arps	User chosen directory into which WRF2ARPS output files are written.
# satellite (MCI2ARPS) parameters		
mci2arps_runname	FL_satellite	Run name used to identify this job.
sat_data_dir	/usr/local/ldm/export/SAT	Directory where raw satellite data resides. Satellite files must use be named with either the prefix 'KSC_IR' or 'KSC_VIS' or the prefix 'goes'.
outdir_sat	/home/user/ADAS/mci2arps	User chosen directory into which MCI2ARPS output files are written.
# radar (88D2ARPS) parameters		
rad_data_dir	/usr/local/ldm/export/nexrad	Directory where raw radar data resides. Must have individual radar site directories, i.e. separate directories for KMLB, KJAX, etc.
outdir_rad	/home/user/ADAS/88d2arps/radar	User chosen directory into which 88D2ARPS output files are written. NOTE: This directory gets cleared out at the beginning of the script.

Table 4. A description of the tunable parameters in the configuration file, *ADAS_input.config*. The configuration file is used by each component of the LDIS Perl scripts to run the complete modeling system.

Parameter	***Default Setting***	***Description***
# ADAS parameters		
adas_runname	FL_adas	Run name used to identify this job.
sfc_data_dir	/home/user/ADAS/MADIS	Directory where raw surface data resides.
ua_data_dir	/home/user/ADAS/MADIS	Directory where raw upper air data resides.
outdir_adas	/home/user/ADAS/adas	User chosen directory into which ADAS output files are written.
# soil (ARPSSOIL) parameters		
soil_runname	FL_soil	Run name used to identify this job.
precip_data_dir	/home/user/ADAS/soil/gauge	Directory where NCEP daily rain gauge data resides.
outdir_soil	/home/user/ADAS/soil	User chosen directory into which ARPSSOIL output files are written.
# WRF initialization (ARPS4WRF) parameters - (includes NCEP post processing - WPPV3 - parameters)		
wrf_core	ARW	WRF core that will be used. Options are: 'ARW' or 'NMM'.
max_dom	1	Number of domains to be processed. Currently only support 1 domain.
outdir_arps4wrf	/home/user/ADAS/wrfinit	User chosen directory into which ARPS4WRF and post-processed files are written.
init_hrs	00,03,06,09,12,15,18,21	Initialization hours for ARPS4WRF to be run.
init_min	00,30	Initialization minutes for ARPS4WRF to be run.
topdir	/home/user/WRF	Directory where the WRF model and/or WPPV3 are located. If both are not located under the same main directory, create a symbolic link to a common main directory. (WPPV3 directory is not needed is using WRF EMS software.)
workdir	/home/user/WRF/POST/FL	Post processing working directory. Directories <u>postprd</u>, <u>wrfprd</u>, and <u>parm</u> should be located here. The <u>parm</u> directory should include the file *wrf_cntrl.parm*. (Not needed if using WRF EMS software.)

Table 4. A description of the tunable parameters in the configuration file, *ADAS_input.config*. The configuration file is used by each component of the LDIS Perl scripts to run the complete modeling system.

Parameter	***Default Setting***	***Description***
# ARPS MPI parameters - These parameters are set based on a 4km grid & based on suggestions in the previously written shell scripts.		
outdir_arps	/home/user/ADAS/arps_mpi	User chosen directory into which ARPS MPI output files are written.
output_flag	1	History data dump option (see hdmpopt in *arps.input*).
restart_flag	0	Flag for restart. 0 = not a restart, 1 = restart.
init_option	3	Model initialization option (see initopt in *arps.input*).
restart_file	199707261200.rst001800	Name of restart file if init_option = 2 or 4 (see rstinf in arps.input)
nudge_flag	0	Analysis increment updating nudging option (see nudgopt in *arps.input*).
nudge_start	0	Time (in seconds) of beginning of IAU window (see ndstart in *arps.input*).
nudge_stop	0	Time (in seconds) of end of IAU window (see ndstart in *arps.input*).
incr_file	dummy	File containing analysis increments (see incrfnam in *arps.input*).
dump_flag	1	History data dump format (see hdmpfmt in *arps.input*).
tintvebd	10800	Time interval (in seconds) at which boundary data files will be searched.
mphy_opt	2	Microphysics option (see mphyopt in *arps.input*).
conv_opt	0	Option for convective cumulus parameterization (see cnvctopt in *arps.input*).
kf_opt	0.3	Factor for Kain-Fritsch scheme (see kffbfct in *arps.input*).
rad_opt	0	Option to choose the longwave schemes (see rlwopt in *arps.input*).
dtrad_opt	600	Time interval (in seconds) to update the radiation forcing (see dtrad in *arps.input*).
out_freq	900	Time interval (in seconds) between history data dumps (see thisdmp in *arps.input*).
restart_freq	0.0	Time interval between restart data dumps (see trstout in *arps.input*).

Table 4. A description of the tunable parameters in the configuration file, *ADAS_input.config*. The configuration file is used by each component of the LDIS Perl scripts to run the complete modeling system.

Parameter	***Default Setting***	***Description***
# ARPS2GEM & ARPS2NCDF parameters - Post processing for ARPS MPI.		
outdir_gem	/home/user/ADAS/arps/gempak	User chosen directory into which ARPS2GEM output files are written.
outdir_ncdf	/home/user/ADAS/arps/netcdf	User chosen directory into which ARPS2NCDF output files are written.

References

Brewster, K., 1996: Implementation of a Bratseth analysis scheme including Doppler radar. Preprints, *15th Conf. on Weather Analysis and Forecasting,* Norfolk, VA, Amer. Meteor. Soc., 92–95.

Brewster, K., 2002: Recent advances in the diabatic initialization of a non-hydrostatic numerical model. *Preprints, 15th Conf on Numerical Weather Prediction and 21st Conf on Severe Local Storms*, San Antonio, TX, Amer. Meteor. Soc., J6.3.

Case, J. L., 2006a: ADAS/ARPS Modifications for Improvement of Forecast Operations Task Summary. Applied Meteorology Unit, Kennedy Space Center, FL, 17 pp. [Available from ENSCO, Inc., 1980 N. Atlantic Ave., Suite 830, Cocoa Beach, FL, 32931]

Case, J. L., 2006b: Reference guide for the operational WRF/ARPS at the National Weather Service in Melbourne, FL and Spaceflight Meteorology Group. Applied Meteorology Unit, Kennedy Space Center, FL, 18 pp. [Available from ENSCO, Inc., 1980 N. Atlantic Ave., Suite 830, Cocoa Beach, FL, 32931]

Case, J. and J. Keen, 2006: User Control Interface for ADAS Data Ingest: Task Summary and User's Guide. Applied Meteorology Unit, Kennedy Space Center, FL, 20 pp. [Available from ENSCO, Inc., 1980 N. Atlantic Ave., Suite 830, Cocoa Beach, FL, 32931]

Case, J. L., J. Manobianco, T. D. Oram, T. Garner, P. F. Blottman, and S. M. Spratt, 2002: Local data integration over east-central Florida using the ARPS data analysis system. *Wea. Forecasting*, **17**, 3-26.

Fulton, R. A., J. P. Breidenbach, D. Seo, D. A. Miller, T. O'Bannon, 1998: The WSR-88D rainfall algorithm. *Wea. Forecasting*, **13**, 377-395.

Manobianco, J. and J. Case, 1998: Final report on prototype local data integration system and central Florida data deficiency. NASA Contractor Report CR-1998-208540, Kennedy Space Center, FL, 57 pp. [Available from ENSCO, Inc., 1980 N. Atlantic Ave., Suite 830, Cocoa Beach, FL, 32931 and http://science.ksc.nasa.gov/amu/final-mm.html]

Rozulmalski, R., 2006: WRF Environmental Modeling System user's guide. NOAA/NWS SOO Science and Training Resource Coordinator Forecast Decision Training Branch, 89 pp. [Available from COMET/UCAR, P.O. Box 3000, Boulder, CO, 80307-3000].

Zhang, J., F. H. Carr, and K. Brewster, 1998: ADAS cloud analysis. Preprints, *12th Conf. on Numerical Weather Prediction,* Phoenix, AZ, Amer. Meteor. Soc., 185–188.

List of Acronyms

ACARS	Aircraft Communications Addressing and Reporting System
ADAS	ARPS Data Analysis System
AMDAR	Aircraft Meteorological Data Reporting
AMU	Applied Meteorology Unit
API	Antecedent Precipitation Index
ARPS	Advanced Regional Prediction System
APRSWXNET	Automatic Position Reporting System
ARW	Advanced Research WRF
ASCII	American Standard Code for Information Interchange
ASOS	Automated Surface Observing Systems
AWIPS	Advanced Weather Interactive Processing System
AWOS	Automated Weather Observing System
CCAFS	Cape Canaveral Air Force Station
C-MAN	Coastal Marine Automated Network
CONUS	Contiguous United States
CSV	Comma-Separated Value
EMS	Environmental Modeling System
FL-Meso	Florida Mesonet
GEMPAK	General Meteorological Package
GOES	Geostationary Operational Environmental Satellites
GRIB	GRIdded Binary
GSD	Global Systems Division
GUI	Graphical User Interface
HRRR	High-Resolution Rapid Refresh
IR	Infrared
KSC	Kennedy Space Center
LDIS	Local Data Integration System
MADIS	Meteorological Assimilation Data Ingest System
MADIS API	MADIS Applications Program Interface
McIDAS	Man computer Interactive Data Access System
MDCRS	Meteorological Data Collection and Reporting System
NAM	North American Mesoscale
NASA	National Aeronautics and Space Administration
NCEP	National Centers for Environmental Prediction
NDVI	Normalized Difference Vegetation Index
netCDF	network Common Data Form
NMM	Non-hydrostatic Mesoscale Model
NOAA	National Oceanic and Atmospheric Administration
NWS	National Weather Service
NWS MLB	NWS in Melbourne, FL
NWP	Numerical Weather Prediction
QC	Quality Control
RAWS	Remote Automated Weather Stations
RUC	Rapid Update Cycle
SAO	Surface Airways Code
SFWMD	South Florida Water Management District
SMG	Spaceflight Meteorology Group
SOO	Science Operations Officer
TAMDAR	Tropospheric Airborne Meteorological Data Reporting
Tcl/Tk	Tool Command Language/toolkit
VIS	Visible
WPS	WRF Preprocessing System
WRF	Weather Research and Forecasting
WSR-88D	Weather Surveillance Radar-1988 Doppler