Documentation for the The Airborne Phased Array Radar (APAR) Observing Simulation, Processing, and Research Environment (AOSPRE) Version 1.0.2

Brad Klotz
APAR Project Scientist
National Center for Atmospheric Research
Earth Observing Laboratory, Remote Sensing Facility

Created 28 November, 2022
Modified 24 October, 2024

Table of Contents

Version Control and Modification List

Version 1.0.0: Initial creation of this documentation (28 November, 2022)
Version 1.0.1: Updated details of sections that have been modified or removed from the previous version (B. Klotz, October 4, 2024)
Version 1.0.2: Changed all references of AOS to AOSPRE as part of a program approved name change (B. Klotz, October 24, 2024)

Document Description

The purpose of this document is to describe the process for generating Airborne Phased Array Radar (APAR) output from the APAR Observing Simulation, Processing, and Research Environment (AOSPRE). The AOSPRE is an end-to-end simulator that incorporates internal and external tools and procedures to best simulate the expected APAR output for the purpose of testing the uncertainty of the data as well as aspects of the data collection and flight strategies. Provided in this document is a summary of the AOSPRE processing method, instructions on how to compile and run the software, instructions on how to generate flight tracks, and a list of other external software packages that will help with the analysis of the output data.

A. APAR Observing Simulation, Processing, and Research Environment (AOSPRE) Overview and Workflow

1. Initial description and setup

The AOSPRE is a tool that can be used to create simulated APAR output from a certain input dataset. The overall workflow follows that the user has access to numerical model output, typically in Weather Research and Forecast (WRF) format. Tests performed with the AOSPRE to date have focused on idealized significant weather events using the Cloud Model 1 (CM1, Bryan and Morrison 2012) framework that outputs data into WRF format. Once the input dataset is available and ready to use, the user must determine a flight that they wish to operate in the model environment. An automated script is available to generate track leg information (discussed later in this document), but the user can also perform this task independent of the automated script. The next step is to set up the scan file with which the software will generate the simulated data. Several standard scan files are included with the software package, but the user can create their own as they see fit. Figure 1 shows a typical scan coverage for range-height indicator and surveillance scan modes. APAR uses a technique called beam multiplexing representative of the future scan controller. A pair of sequential pulses can deduce reflectivity factor (Z) and Doppler velocity (V_r). A pulse set consisting of 3 or 4 pulses in one beam direction before changing to another direction will be able to deduce polarimetric parameters (e.g., differential reflectivity, Z_{DR}). This process was illustrated for the planned APAR capabilities by Vivekanandan and Loew (2018) who proposed a scanning strategy having 6 beams per acquisition time and their Figure 8 is reproduced below in Figure 2. Each group illustrated in Figure 2 is one acquisition time period containing 6 beams with 20 pulse pairs per beam (20 revisits per acquisition time). Successive beams are at least 4.4° separated in order to suppress second trip echoes by 20 dB.

The AOSPRE is currently uses a radar simulator known as the Cloud-resolving Radar Simulator (CR-SIM, Oue et al. 2020), which was developed by researchers at one of NCAR's partner institutes, SUNY Stony Brook. CR-SIM is also dependent on a configuration file that specifies the radar operating frequency and beamwidth to use. Several other parameters are specified, but should not need changing in most

circumstances. Examples of these files are also included in the software package. Within the code, the user has the option to either run in serial or parallel mode, depending on the capabilities of the computer the user will be using or how the code is compiled. Running in parallel typically reduced the processing time by as much as 90%, but this is dependent on the user's machine. Existing tests for processing speed were performed on a server with 72 processors and 502 GB of memory. Once the flight path, scan information, CR-SIM configuration, and parallel processing option are determined, these can be added to a namelist file (either manually or through the automated script) that will be read by the main AOSPRE processing code. Once these preliminary steps are complete, the AOSPRE code can be started.

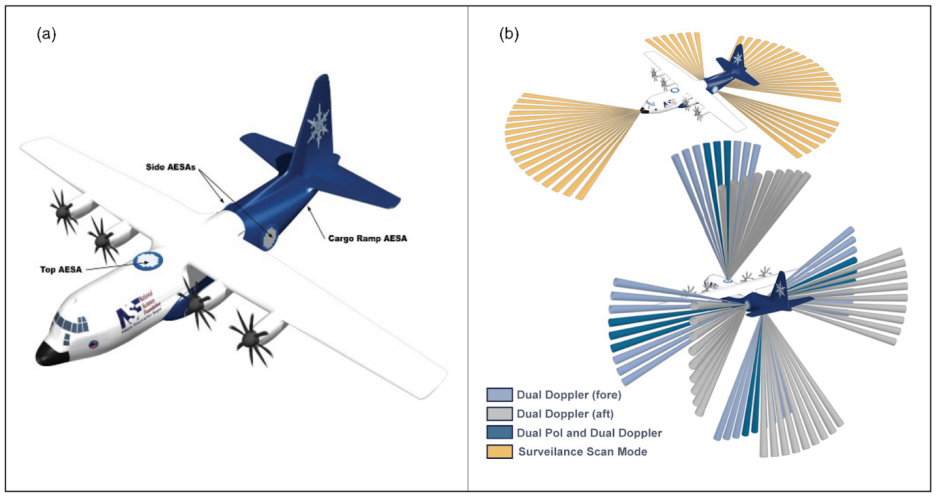

Figure 1. In (a), the location of the four Airborne Electronically Scanned Arrays (AESAs) mounted on the exterior of the NSF/NCAR C-130 is shown. Panel (b) provides the scan schematic for surveillance mode (top) and range height indicator mode (RHI, bottom) with indication for beams that provide the dual-Doppler and dual-polarization observations. Images included here are courtesy of the NCAR Comet office.

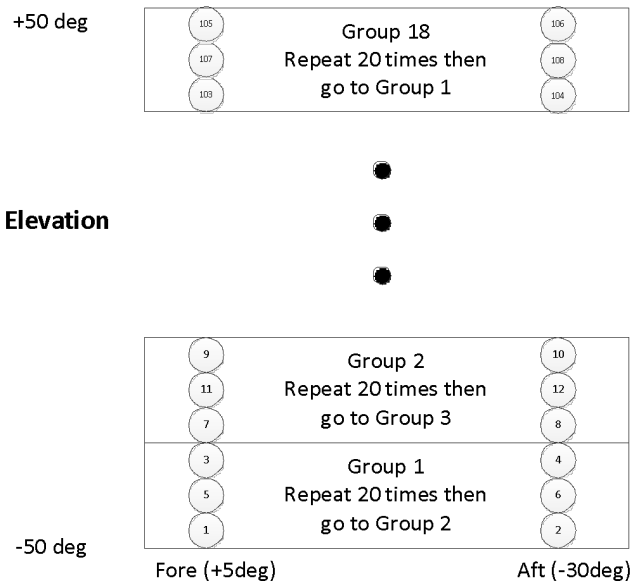

Figure 2: Illustration of multiplex scanning with 6 beams per acquisition time, or group. Reproduced from Vivekanandan et al. (2018), Figure 8.

2. AOSPRE processing

The code runs in several stages, including the initial determination of model output file availability, radar volume allocation based on scan parameters provided, model file opening and reading, looping through the scan beam and gate information to fill the volume for the specified variables, and outputting the data into the common radar output format (CfRadial, Dixon and Lee 2016). This process is completed over the number of model files that exist within the time frame provided for the flight. In most cases, the expectation is for the user to interpolate between model output times, given that the aircraft currently used in the simulation (the NCAR C-130) has an airspeed of 120 m s^{-1} and can produce output about every ~2.0-2.3 s for a traditional fore/aft scan sequence. For example and if processing in serial mode, the code would open and read from two model output files and produce an interpolation factor to apply during calculation of the radar variables. Once the flight is complete, all output files are transferred to the specified location during the initial setup phase. When processing in parallel mode, the code initially assumes to process data in flight segments simultaneously.

3. Post-processing applications

There are several tools that are currently used to analyze the data. The Lidar Radar Open Software Environment (LROSE, Dixon and Javornik 2016) software package includes many tools to examine different aspects of the data, including visualizing the data with HawkEye. Because APAR is a dual-polarization radar, the functions that utilize these output parameters, such as Z_{DR} or K_{DP}, would be available for use. For example, particle identification can be determined using the built-in tool called RadxPid. Another important tool that is included in the LROSE package is the Spline Analysis at Mesoscale Utilizing Radar and Aircraft Instrumentation (SAMURAI, Bell et al. 2012). This software package allows the user to produce a 3-D wind analysis based on the Doppler wind that is observed by the radar. The user is also given freedom to produce their own post-processing software and applications as seen fit for specific needs.

B. Detailed Processing Steps

This section provides details on the different portions of the AOSPRE workflow and how to run the various scripts. In general, any script that has a .sh tag at the end of the filename will need to run from the terminal command line.

1. How to Download and Install

The code is contained in a Github repository, which can be found here: https://github.com/NCAR/AOSPRE/tree/main. This is currently a private repository, but a tar file version is available to download. Within the repository, navigate to the docs folder, where the user will find instructions on prior downloads and installation, building the software, and initial testing to ensure the software is working. Before you move forward, it is important to understand the prerequisites for installation

2. Software Requirements

In order to run the AOSPRE software, there are certain software requirements and packages you must have access to. Here is the list of what the user should have on their machine along with some optional packages.

i. Linux/Unix Bash terminal environment
ii. Fortran compiler and libraries (gfortran or gcc)
iii. NetCDF libraries
iv. Cloud-resolving Radar Simulator (CR-SIM)
v. NcView application package (optional)
vi. A text editor for modifying files
vii. OpenMPI and OpenMP (optional)
viii. LROSE software (not necessary to run AOSPRE, but extremely useful)

3. Model Output

It is up to the user to either produce their own model simulation output or obtain output from another source. Without it, the AOSPRE cannot run. Some suggestions would be to use a model output resolution no larger than 1 km as anything larger than this would negatively affect the ability to understand the APAR output or use its full capabilities. Make sure your output is easily accessed by the AOSPRE code. Several important considerations for this model output are that it contains certain variables and has utilized appropriate model microphysics schemes. For information on the specifics for CR-SIM, see their documentation here: https://you.stonybrook.edu/radar/research/radar-simulators/. Table 2 in Section C provides the list of model variables necessary for running the AOSPRE code.

4. Namelist file

The namelist file contains all the essential information that the AOSPRE needs to operate. Right now, there are separate versions for RHI and PPI modes that can be used. Within each namelist file is provided several sections of information, which include

i. The 'options' section, which contains the directory and file format of the input model data and information regarding the flightpath, and the flight level coordinate (height or pressure). It also contains information on which beamwidth method to apply.
ii. External aircraft attitude section, which allows the user to input real variations in aircraft yaw, pitch, and roll as it encounters turbulence
iii. A 'scanning' section, which contains information on the CR-SIM config file to use, the scan file to use, as well as important information regarding the scan (beam spacing and multiplexing information)
iv. An 'output' section, which allows the user to specify which variables should be included in the output files

Basic files used: namelist.surveillance and namelist.rhi (text files), namelist.LHS, namelist.RHS, namelist.BOT, namelist.TOP
For the auto scripts, they would be: namelist_template_auto.rhi and namelist_template_auto.surveillance

Commented [BK1]: Not sure the auto scripts are necessary right now.

5. Flightpath determination

There are two methods for determining the flightpath information, manual and through an automated script. The steps for each of these is essentially the same. However, the automated method will place the flight information into the namelist file, but the manual method will not. The flightpath is defined in the model X, Y gridspace, so the determination of the flight path uses the grid indices and model resolution to determine aircraft positions. Using NcView, the user will need to open one or several model output files to visually examine the position of the storm or phenomenon to be observed and base the flightpath on the information obtained. It should be noted that the following scripts run in a Linux

bash script environment (or through Xterm on Mac OS). There is no application for Windows. For the auto script, there is some other information asked regarding the AOSPRE namelist, which will be populated with the information provided in the auto script. Here are the scripts used and steps to follow, assuming the manual method. Please note these scripts are included in the "scripts" directory of the current code repository.

Scripts: flight_planner.sh (manual), flight_planner_auto.sh (auto), run_flight_planner.sh (auto)
Config file: flight_config (manual), flight_config_auto (auto)
External tools: ncview (for viewing the model output files)

i. When running the auto version, you will be prompted to enter certain information. Here, the steps are laid out through the manual version.
ii. Using ncview, open one or several model output files to determine the model grids needed for the coverage of the flight, specifying a starting x and y grid location
iii. Open the flight_config file and adjust the variable values as necessary, including the length of the flight in seconds, the resolution of the model output, the starting model x, y, and aircraft altitude
iv. Assign the headings of your flight legs, the duration of each flight leg in seconds, and the altitude of each flight leg in meters. The first entry is a dummy value as indicated by a leg length of 0 seconds.
v. Once the config file is setup, use the terminal window to run the flight_planner.sh script. Output will be displayed in the terminal. The flight grid points can be copied into your namelist file (do not include the dummy points when you copy the information over to the namelist file).

6. APAR Scan file and CR-SIM configuration file

Now that the flightpath has been determined and populated in the namelist file, the user should create a new scan file or use an existing scan file. The CR-SIM config file should also be specified here, where the user should indicate which operating frequency and beamwidth to use. Several CR-SIM config files are provided in the package for use and or modification by the user. For the APAR scan files, they are provided in a two column format text file and contain information about the primary axis of rotation, sweep mode, and the information contained in each column (i.e., rotation and tilt angles). The basic scan files use Z as the primary axis of rotation with beams pointing at 240° and 275° relative to the aircraft fuselage. Tilt angles for the RHI scan are between -53° and 53.5° at intervals of 1.5° for a total of 72 beams per rotation angle. The user, of course, can create their own scans as needed. The surveillance scan mode uses a tilt of 0° with a full azimuthal rotation at intervals of 1.5° for a total of 241 beams. There is a Scanning Table Library in the repository that can be used as a guide for existing scans and how to create a new file.

Files available: CONFIG_C (for CR-SIM), Scanning_Bot_0deg_fore.txt, Scanning_LHS_0deg_fore.txt (LHS = Left Panel),

Scanning_LHS_5deg_fore_30deg_aft.txt, Scanning_LHS_PPI_surveillance.txt, Scanning_RHS_0deg_fore.txt (RHS = Right Panel), Scanning_RHS_PPI_surveillance.txt, Scanning_Top_0deg_fore.txt

7. Running the AOSPRE main script

Once the setup information is determined and populated into the namelist file, it is now time to make sure the AOSPRE code is compiled and ready to run. These steps listed below are in reference to the manual processing mode(s), but notation is also provided for the automated version. It should be noted that the compilation of the code is based on information in a Makefile. The specific directories of the user's machine for requirements such as NetCDF libraries, fortran compiler, OpenMP and OpenMPI need to be changed before compiling. There are also options for using cmake to build to code, and instructions on how to do this are provided in the repository documentation. Here, the instructions offer one path for installation through the Makefile.

The (optional) scripts needed are: RUN_AOSPRE.sh (manual) or RUN_AOSPRE_auto.sh (auto)
The necessary executable is currently listed as a.out (but this can be changed by the user if desired). One thing that might be helpful is for the user to set up a separate testing directory that houses the baseline tests that can then be copied into a more detailed output directory for a specific flight simulation. It is recommended to keep this separate of the code repository test directory. The pre-developed scripts take this method into account.

Here are the steps to follow to run the main AOSPRE script:

i. If following the traditional "make" installation, check the Makefile in your code directory to ensure everything is specific to your machine, then execute 'make all'. This will create the executable file to run the AOSPRE. If installing with the cmake instructions, the commands are slightly different. See the online or repositirydocumentation regarding specific cmake build instructions.
ii. If running without any special configuration, you can execute the following command in the directory where your namelist file is stored. mpirun -np <numprocs> <code_directory_path>/embed-crsim/a.out <namelist_file>; if you do not have mpirun as a command on your machine, use <code_directory_path>/embed-crsim/a.out <namelist file>
iii. If running in manual mode, you can use the RUN_AOSPRE.sh as a guide for running a more organized form of job submission and output. If you do this, make sure your scan mode, namelist file, and output directory are correct. Then enter into the command line in the terminal: ./RUN_AOSPRE.sh from your main AOSPRE directory. You can of course add typical shell commands to the end of this command, such as running it in the background, for example. Make sure that this script is executable as well (chmod 775).

iv. If running in auto mode, you need to make sure you have created a file within the your main AOSPRE directory call “AOSPRE_env.txt”. This file will contain some of directory information needed for the file (see Table 6). Then enter ./RUN_AOSPRE_auto.sh into the command window and follow the prompts. Make sure you have run flight_planner_auto.sh prior to executing this script, otherwise it will not work.

8. Verify the output

After the AOSPRE has completed running the prescribed flight, the output will be placed in the directory specified within the RUN_AOSPRE script. Typically, the naming convection for the directory will include information about the weather phenomena, the model resolution, the aircraft altitude, the starting x and y grid points, the frequency band used, and panel name. For example, if the user were to run a simulated flight using a supercell simulation that has 100 m horizontal resolution and 10 second temporal resolution, a flight altitude of 1 km, with C-band configuration, it would be helpful to name the output directory something like the following:

supercell_100m_10s_1.0kmAGL_Cband_x200_y150_LHS

Within this main directory, a directory for rhi or surveillance will be created, depending on the scan type designated by the user. All output files will be copied to one of these directories. With the output in its specific location, it can be used to perform the desired analyses of the user. One step that the user should perform is to verify that the output look reasonable by displaying it with HawkEye (LROSE) or some other radar display software. These tools are beyond the scope of this document.

In addition to the APAR output, a flightpath text file is also provided and copied to the user’s specified directory. It contains the position of the aircraft during the flight as well as the wind components the aircraft experiences during flight.

9. Common Errors or Issues with AOSPRE

AOSPRE has built-in error messages that appear if there are any problems during processing. However, there are several common errors that can be avoided if the user is aware of them. Table 1 lists the associated error or issue and the solution needed to resolve it.

Table 1.

AOSPRE Error	Resolution
Running out of waypoints at the end of a flight	Always add an additional 5 km at the end of your flight path (but not any additional time) to avoid having waypoint errors.

Unable to open WRF model files	Ensure that your model output directory is properly linked and accessed by the code.
Unable to read WRF model files	Make sure that your NetCDF libraries are properly linked through the Makefile or cmake build
Unable to open multiple WRF model files at the same time	Ensure that enough free memory is available. This can be done by writing on the terminal command line: ulimit -s unlimited.
In parallel mode, not enough processors set aside for the time window	Sometimes it is necessary to extend your flight track by 5-10 seconds to ensure that the correct number of processors are used based on the files that are identified in the time period assigned for the flight.
Output files not copied to your directory	This could be caused by multiple issues: Check your directory location to make sure it was created, or it could be related to the waypoint issue noted above.

10. Required Directory Structure

The AOSPRE code and associated scripts are setup using a particular directory structure. The strong suggestion here would be to follow the same structure. If you choose to modify it, then you would need to modify all existing code and scripts that reference these locations.

User directory: "/home/username" or some other suitable directory

--------> Main AOSPRE directory: <User directory>/git/AOSPRE

----------> AOSPRE code directory: <Main AOSPRE directory>/code

----------> Source code: <AOSPRE code directory>/<embed_crsim>

The Main AOSPRE directory should be kept separate of any future tests, especially if the code was cloned with git. If using the RUN_AOSPRE script, it should also be executed from this directory. Although it is not required, the user could make another directory outside of git to store these storm specific directories for ease of use.

11. Summary of Commands

This section contains a summary of the commands to run for both the manual or auto mode. When running the "RUN_AOSPRE.sh" script, there is an option for the user to

set up the mpirun command or not. The current version is not net up to use mpirun by default given the possibility of a user's install not including that capability. See section 7 for details on how to use mpirun, if desired. The auto mode is still listed as an option but is not an active set of scripts in the repository.

Manual mode:

i. Generate flightpath: ./flight_planner.sh
ii. Update scan, namelist, CR-SIM config files
iii. Update variables and paths in RUN_AOSPRE.sh
iv. Run the AOSPRE: ./RUN_AOSPRE.sh

Auto mode:

i. Generate flightpath: ./flight_planner_auto.sh
ii. Double check your scan file and AOSPRE_env.txt
iii. Run the AOSPRE: ./RUN_AOSPRE_auto.sh

Running without mpirun:

If your machine does not have mpirun or other OpenMPI libraries, make sure you adjust the Makefile to include the correct information. The running of the AOSPRE code will follow as above because a check on the existence of mpirun will allow the script to execute using a different command without user intervention.

C. Script-Specific Variable Lists

The following set of tables provides lists of input and output variables for each script along with a description of its value or use and an example of typical use. They are listed in the same order as the scripts provided above.

1. Model output required variables

Table 2.

Variable Name	Description
XTIME	Time since start of simulation
RDX	Inverse x grid length
RDY	Inverse y grid length
XLAT	Latitude (North is positive)
XLONG	Longitude (East is positive)
U	x-wind component
V	y-wind component
W	z-wind component
T	Perturbation potential temperature

P	Perturbation pressure
PH	Perturbation geopotential
PB	Base state pressure
PHB	Base state geopotential
QVAPOR	Water vapor mixing ratio
QCLOUD	Cloud water mixing ratio
QRAIN	Rain water mixing ratio
QICE	Ice mixing ratio
QSNOW	Snow water mixing ratio
QGRAUP	Graupel water mixing ratio
QNICE	Ice number concentration
QNSNOW	Snow number concentration
QNGRAUPEL	Graupel number concentration
QNRAIN	Rain number concentration

2. Namelist Variables

Table 3.

Variable Name	Description	Example
&options section		
wrf_glob_pattern	Directory and file naming convention of the model output	<user_dir>/CM1_OUT_SUPERCELL_100m/wrfout_000????s.nc"
output_filename_format_string	This is the naming format of your output files	'("RHI.",A,"_to_",A,".nc")'
leg_initial_time	Start time of your flight in seconds (based on your model output times)	leg_initial_time = 4490
leg_time_seconds	This is the length of your flight in seconds	leg_time_seconds = 600
time_evolution	This variable is a logical that determines if there should be interpolation between model output times. Setting the variable to .TRUE. would mean there will be interpolation between times.	time_evolution = .TRUE.
flight_waypoints_x	These are the model x-grid locations of your flight as you determine from the flightpath script.	flight_waypoints_x = 215, 2500
flight_waypoints_y	These are the model y-grid locations of your flight as you determine from the flightpath script.	flight_waypoints_y = 375, 375
flight_waypoints_vert	These are the aircraft altitude locations of your flight as you determine from the flightpath script.	flight_waypoints_vert = 1000, 1000

flight_level_coordinate	This is the coordinate system for your vertical dimension, and it can be set to Z for height or P for pressure. The pressure option has not been fully tested yet.	flight_level_coordinate = “Z”
air_speed	This is the air speed of your aircraft. For APAR, we are using the typical speed of the C-130	air_speed = 120.
herky_jerky	This variable is a logical which tells the AOSPRE whether to introduce simulated variational motion. The default should be .FALSE.	herky_jerky = .FALSE.
bwtype	This defines the method for how to interpret the beamwidth. The idealized case does not incorporate beamwidth (option 0), and the constant or variable beamwidths use the beamwidth to define which model indices are used in the interpolation (constant = 1, variable = 2)	bwtype = 0
ref_angle	Associated with bwtype = 1 or 2, it will be used to orient the frame to the correct panel (Left=270, Right=90, Top = 0, Bottom = 180	ref_angle = 270
&attitude_external_source		
use_external_attitudes	This variable is a logical which tells the AOSPRE whether to use the file containing actual flight motion parameters to simulate more realistic aircraft motion within a storm. The default should be .FALSE.	use_external_attitudes = .FALSE.
attitude_file	This is the name of the file to read the aircraft motion information	attitude_file = "attitude.nc"
attitude_orientation_rotate_degrees	This variable assigns the amount of rotation needed for the values listed in the file to make it align with the simulated aircraft orientation.	attitude_orientation_rotate_degrees = 182.0
&scanning		
CRSIM_Config	This is the name of the CRSIM configuration file you wish to use.	CRSIM_Config = "CONFIG_C"
scanning_table	This is the file containing your scan information	scanning_table = "Scanning_Table_LHS_Multiplexing_RHI_Zprime_ordered"

pulse_repetition_frequency	This is the PRF for the radar	pulse_repetition_frequency = 2500
pulses_per_pulse_set	Using the multiplexing technique, this assigns the number of pulses to use	pulses_per_pulse_set = 2
revisits_per_acquisition_time	This variable sets the number of revisits within an acquisition time	revisits_per_acquisition_time = 20
beams_per_acquisition_time	This is the number of beams needed to obtain the relevant Doppler and dual polarization informatiom	beams_per_acquisition_time = 6
skip_seconds_between_scans	This can be changed to allow a certain amount of time between scans. For surveillance mode, this is set to 28.32.	skip_seconds_between_scans = 0.0
meters_between_gates	This is the along beam resolution. For APAR, it is 150 m.	meters_between_gates = 150.
meters_to_center_of_first_gate	This value tells the code to not assign any values within the radar and halfway between the center of the first gate. The typical value should be 150 m.	meters_to_center_of_first_gate = 150.
max_range_in_meters	This is the maximum range for the radar, which for APAR, is expected to be 75 km.	max_range_in_meters = 75000.
&config_output		
This section allows the user to assign True or False to variables being included in the output files. If it is left empty, then all variables will be assigned to the file. If the user decides to explicitly state which ones to include it would be written as: CONFIG_ZHH%OUTPUT = T.		

3. Flightpath Variables

These are the variables that are either populated manually or through the automated flightpath script. When running the automated flightpath script, you will be prompted to enter some of this information in the command line.

Table 4.

Variable Name	Description	Example
TOTAL_TIME	The length of the flight in seconds	TOTAL_TIME = 600
MODEL_DX	This is the horizontal spacing of the model output in meters	MODEL_DX = 100
START_X	This is the starting x-grid location of the aircraft position	START_X = 215
START_Y	This is the starting y-grid location of the aircraft position	START_Y = 375
START_ALT	This is the starting aircraft altitude in meters	START_ALT = 1000
AC_HEAD	This contains the vector of aircraft headings for each leg; the first entry	AC_HEAD = 90 90 45 0

	should be considered a dummy value and is set to the same value as the first leg.	
LEG_TIME	The length of time for each leg in seconds; the dummy time should be set to 0.	LEG_TIME = 0 240 120 240
AC_ALT	The altitude of the aircraft in vector form allows the user to change the altitude in flight if necessary; the dummy value should be set to START_ALT	AC_ALT = 1000 1000 1000 1000
AC_SPEED	This is the aircraft air speed in meters per second.	AC_SPEED = 120

4. Scan File Variables

The listed variables are to be populated by the user. They provide information related to the scan that will be used during the AOSPRE processing.

Table 5.

Variable Name	Description	Example
PRIMARY_AXIS	This is the primary axis of rotation for the scan	PRIMARY_AXIS = Z
SWEEP_MODE	This will specify whether you are doing and RHI or PPI scan; if it is a PPI scan, the user would enter “sector”	SWEEP_MODE = RHI
PARAMETERS	These are the terms that define the other axes of the 3-D scan space. See Lee et al. 1994 discuss these different 3-D rotational spaces.	PARAMETERS = ROT, TILT
<sweep>	This marks the beginning of a sweep	---
</sweep>	This marks the end of a sweep	---

5. CR-SIM Config File Variables

The listed variables are provided in the CONFIG files needed for applying the CR-SIM portion of the code. Rather than a variable name, these are listed by their description with associated value. The ones that the user should likely change are included here. If there is interest in changing other variables, consult the CR-SIM documentation.

Table 6.

Variable Name and Description	Example
#Specify radar frequency (3.0d0, 5.5d0, 9.5d0, 35.d0, 94.d0)	5.5d0
#Turn off the polarimetric variables: yes == 1, any other number is no	2

#Specify the radar beamwidth (one-way angular resolution) in degrees	2.0d0
#Specify the radar range resolution dr in meters	150.d0
#Specify value of coefficient ZMIN in relation to dBZ_min(dBZ)=ZMIN(dBZ)+20 log10(range in km)	-31.d0

6. Variables for AOSPRE run script

If you are running the AOSPRE with the auto script, this information will be prompted by the script. If you are running the manual mode, then these descriptions will be helpful as well.

Table 7.

Variable Name	Description	Example
base_dir	This is the Main AOSPRE directory listed above in the directory structure section.	See directory structure section
operating_dir	This is the directory where your namelist file is stored (referred to as Storm directory above)	See Storm directory structure listed above
scan_mode	This is the type of scan being performed, rhi or surveillance	“rhi”
namelist_file	This is the explicit name of the namelist file you want to use for the flight	namelist.rhi
experiment_dir	This would be the optional directory store output in if the user desires; It can be left as a blank string, if desired	“individual_tests”
test_dir	This is the directory name that the output would actually get stored in as noted in subsection 6 above.	See example in subsection 6
Auto-specific needs		
In AOSPRE_env.txt	This text file asks you to enter the location of your code directory, the full storm directory where your namelist is stored, and an R or S to designate the scan type	---

7. Download Contents

Table 8 provides a detailed listing of the files included in the AOSPRE repository download. For reference, the main Fortran script to run the AOSPRE is extract_apar_wrf.F.

Table 8.

Directory Name	Contents
AOSPRE (main directory)	
	code/
	docs/
	scripts/
	test/
	cmake/
	Scanning_Table_Library/
	BUILD_CMake.md
	CMakeLists.txt
	README.md
(main directory)/code	
	embed-crsim
	CMakeList.txt
(main directory)/scripts	
	RUN_AOSPRE.sh
	cfrad_name.sh
	derecho_qsub_example.sh
	flight_planner.sh
	flight_planner_auto.sh
	flight_config
	flight_config_auto
	run_flight_planner.sh
(main directory)/docs	
	AOSPRE_Documentation.docx
	AOSPRE_Documentation.pdf
	README_AOSPRE.txt
	Inclusion of .md files that create online documentation
(main directory)/test	
	CONFIG_crsim
	namelist.LHS, namelist.RHS, namelist.TOP, namelist.BOT
	scanning_lhs.txt, scanning_rhs.txt, scanning_top.txt, scanning_bot.txt
	wrfout_test_4650.nc, wrfout_test_4660.nc
(main directory)/cmake	
	FindNetCDF.cmake
(main directory)/Scanning_Table_Library	
	Scanning_Bot_0deg_fore.txt
	Scanning_LHS_0deg_fore.txt
	Scanning_LHS_5deg_fore_30deg_aft.txt
	Scanning_LHS_PPI_surveillance.txt
	Scanning_RHS_0deg_fore.txt

	Scanning_RHS_PPI_surveillance.txt
	Scanning_Top_0deg_fore.txt
(main directory)/code/embed-crsim	
	a.out
	crsim_luts_mod.F
	crsim_mod.F
	extract_apar_cm1.F
	kwm_date_utilities.F
	module_access_radsim.F
	module_aircraft.F
	module_configuration.F
	module_external_attitude.F
	module_geometry.F
	module_llxy.F
	module_scanning.F
	extract_apar_wrf.F
	module_access_wrf.F
	module_cfradial_output.F
	module_crsim_wrapper.F
	phys_param_mod.F
	ReadConfParameters.F
	wrf_var_mod.F
	Makefile
	CMakeLists.txt
	CONFIG

D. References

Bell, M. M., Montgomery, M. T., and Emanuel, K. A., 2012: Air-sea enthalpy and momentum exchange at major hurricane wind speeds observed during CBLAST. *J. Atmos. Sci.*, **69**, 3197-3122.

Bryan, G. H., and Morrison, H., 2012: Sensitivity of a simulated squall line to horizontal resolution and parameterization of microphysics. *Mon. Wea. Rev.,* **140**, 202-225.

Dixon, M., & Javornik, B., 2016: Lidar Radar Open Software Environment (LROSE) Core Software. UCAR/NCAR - Earth Observing Laboratory. doi: 10.5065/60HZ-RY38.

Dixon, M. J., and Lee, W.-C., 2016: NCAR/UNIDATA CfRadial data file format: Proposed CF-compliant netCDF format for moments data for RADAR and LIDAR in radial coordinates, doi: 10.5065/k0md-1642.

Lee, W.-C., Dodge, P., Marks, F. D., and Hidebrand, P. H., 1994: Mapping of Airborne Doppler radar data, *J. Atmospheric and Oceanic Technology*, **11**, 572-578.

Oue, M., Tatarevic, A., Kollias, P., Wang, D., Kwangmin, Y., and Vogelmann, A. M., 2020: The Cloud-resolving model Radar SIMulator (CR-SIM) Version 3.3:

description and applications of a virtual observatory, *Geosci. Model Dev.*, **13**, 1975-1998, doi: 10.5194/gmd-13-1975-2020.

Vivekanandan, J. and Loew, E., 2018: Airborne polarimetric Doppler weather radar: trade-offs between various engineering specifications, *Geosci. Instrum. Method. Data Syst.*, **7**, 21-37, doi:10.5194/gi-7-21-2018.