The ASGS Developers Guide

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

1 Introduction

The ADCIRC Surge Guidance System (ASGS) is a portable, transportable, geographically agnostic software system for generating storm surge and wave guidance from ADCIRC + SWAN in real time on high resolution grids.

It derives meteorological forcing either from gridded wind fields (e.g., NCEP's NAM model) or, for tropical cyclones, from a parametric wind / pressure model using storm parameters extracted from the text of the National Hurricane Center's Forecast/Advisories.

For tropical cyclones the system is also configured to run a mini-ensemble of storms comprised of a predefined set of perturbations to the NHC's consensus forecast. The ASGS can be configured to accept river flow forecasts as input as well.

The system runs successfully on a wide range of high performance computing platforms including Ranger at the Texas Advanced Computing Center (Univ Texas), Diamond, Jade and Sapphire at the US Army Corps of Engineers Engineering Research and Development Center, QueenBee and Tezpur at LONI and LSU, and several HPC platforms / clusters at the University of North Carolina at Chapel Hill. The ASGS has been run on several grids covering the western North Atlantic, Caribbean Sea and Gulf of Mexico with high resolution areas in the northern and western Gulf of Mexico, in North Carolina and in Maryland/Virginia.

This document provides (a) a tour of the source, with brief explanations of the function of each major source code file; (b) installation requirements and procedure; (c) details of the vortex meterological model embedded in ADCIRC; (d) the guidelines that ADCIRC Surge Guidance System developers should use when developing new code for ASGS and its related utilities; and (e) a short tutorial for Subversion (svn), the version control software that we use to coordinate development and ease the task of merging in new features.

2 Guided Tour

The following is a list of directories and files that are provided with the ASGS along with a description of their purpose and function.

2.1 base

The root of the ASGS directory structure contains the scripts that provide the main features of the ASGS.

2.1.1 asgs_main.sh

The actual script that is executed to activate the ASGS is called asgs_main.sh. Its main function is to perform a hindcast if needed, then move to a nowcast/forecast loop. It contains code that controls the preparation of new runs, sets up computational jobs, submits them, and monitors them. It also contains a small section that contains conditionally executed system-specific configuration information.

All the code that actually does the specialized work, e.g., downloading meteorological data, constructing control files, generating visualizations of results, etc., is contained in modular external scripts.

The asgs_main.sh script is the most central and most important one in the ASGS.

2.1.2 control_file_gen.pl

The ADCIRC fort.15 and SWAN fort.26 (run control) files are generated by control_file_gen.pl. This script accepts a wide array of input arguments that describe the sort of run the Operator has configured, including the use of tidal forcing, wave coupling, time step size, desired output file format (ascii vs netcdf), and many others.

Other input arguments to this script provide state information, e.g., whether the model is to be hotstarted, the current simulation time, the advisory number from the latest National Hurricane Center Forecast/Advisory (in the case of hurricane-related runs), etc.

When the script runs, it uses templates that have been made for the control files along with the input arguments and a few external programs (for generating tides, etc) and the already-generated meteorological forcing files (if any) and produces control files for ADCIRC or ADCIRC+SWAN along with metadata in the run.properties file.

This is the 2nd most important script in the ASGS (after asgs_main.sh).

2.1.3 get_atcf.pl

The get_atcf.pl script is used by the ASGS to monitor the RSS feed from the National Hurricane Center for new Forecast/Advisories and to download the latest advisory the moment it becomes available. It does this by parsing the xml file posted on the RSS feed, and when the advisory number has been updated by the NHC, get_atcf.pl follows the link to the html file containing the actual advisory information.

After downloading a new Forecast/Advisory from the National Hurricane Center website, the script then downloads the latest hindcast (BEST track) file for the same storm from NHC's anonymous ftp site. Both these files are required for generating an ADCIRC fort.22 (meteorological forcing) file.

Once a forecast is complete, and the next advisory has not yet been posted, the script waits 60 seconds before checking the RSS site again. It will continue checking every 60 seconds until the next advisory is posted.

The script is called get_atcf.pl because the so-called BEST track file format (that describes the files that the script downloads from the anonymous ftp site) was designed by the Automated Tropical Cyclone Forecasting system (ATCF). Please see the section below that discusses the ADCIRC parametric vortex models below for more details on this format, including a link to the online documentation.

2.1.4 nhc_advisory_bot.pl

The nhc_advisory_bot.pl script takes the html file downloaded by get_atcf.pl script and parses out all the information required to write an ATCF formatted forecast file ("OFCL"). It then generates the ATCF formatted file with the forecast parameters; this file could be used directly by one of ADCIRC's internal parametric vortex models, but the ASGS subjects it to further processing as described below.

2.1.5 storm_track_gen.pl

The storm_track_gen.pl script takes the ATCF-formatted file generated by nhc_advisory_bot.pl (representing the forecast), the ATCF-formatted file downloaded by get_atcf.pl (representing the hindcast), and the currrent simulation state (representing the current hotstart time) and constructs an appropriate ADCIRC fort.22 (meteorological forcing) file as output.

This script accepts optional arguments that can be used to vary the storm's track (expressed as a percentage of the desired distance between the consensus track and the edge of the cone of uncertainty), the radius to maximum winds (expressed as a percentage of the radius to maximum winds determined from the input), the overland speed of the storm (to speed it up or slow it down), and the maximum wind speed. These variations are configured in the overall ASGS configuration file and can be used to explore "what-if" scenarios during a tropical cyclone event.

2.1.6 get_nam.pl

The get_nam.pl script is used by the ASGS to monitor NCEP's anonymous ftp site for newly released output from their NAM (North American Mesoscale) atmospheric model. The WRF-NMM (Weather Research and Forecasting, Non-hydrostatic Mesoscale Model) is used to produce NAM results.

The script accepts the current time of the ADCIRC simulation as input, and downloads the latest available NAM data, provided that the latest available data is later than the current ADCIRC simulation time. If there are no NAM data available that postdate the current ADCIRC simulation time, the script print and information message and exits. The calling routine (in asgs_main.sh) then retries the download every 60 seconds until new data have been discovered and downloaded.

The get_nam.pl script relies on the "wgrib2" executable to perform basic sanity checks on the grib2 files that it downloads.

2.1.7 NAMtoOWI.pl

The NAM output files in grib2 format that were downloaded by get_nam.pl are converted into a form that ADCIRC (or ADCIRC+SWAN) can use by the NAMtoOWI.pl script. This script uses the wgrib2 utility to extract the sea level u, v, and p from the NAM output files (in grib2 format) and produce ascii data files. The ascii data are then reprojected from the NAM coordinate system (lambert conformal) to the ADCIRC coordinate system (geographic) using the awip_lambert_interp executable. A list of points must be provided where the reprojected and interpolated data should be calculated. Finally, the interpolated data are output in OWI (Ocean Weather Inc) format for use in ADCIRC or ADCIRC+SWAN (NWS=12).

2.1.8 get_flux.pl

If time-varying river flux boundary condition data (ADCIRC's fort.20 file) are available, and the ASGS has been configured to include river flux forcing, the get_flux.pl script retrieves the boundary condition files and formats them for use in ADCIRC. It must reads the ADCIRC mesh file to determine the number of river flux boundary nodes so that it can interpret the data in the incoming flux files, since the file format is not self describing. The script also takes into account the current time in the ADCIRC simulation and the files that are currently available on the remote ftp site when constructing the flux boundary condition file that ADCIRC will actually use.

2.1.9 queue script generators

The ASGS supports many HPC platforms, and it seems that each one is different from the others and has its own idiosyncracies. As a result, there are a number of queue script generation scripts, including erdc.pbs.pl, loadleveler.pl, queenbee.pbs.pl, ranger.serial.pl, ranger.sge.pl, and tezpur.pbs.pl. Although there are different scripts for different platforms, they are broadly similar, in that they all use a template approach . . . a working queue script has "blanks" inserted for parameters that will change for different simulation runs and then a script is written (often cloned from one of the existing ones) that simply fills in the blanks according to the command line arguments.

2.2 config

The "config" subdirectory contains sample configuration files to illustrate the changes to be made for various situations.

2.2.1 config/asgs_config_irene_garnet_ec95d.sh

This is a sample configuration file that sets the ASGS up to run hurricane Irene on the Garnet machine at ERDC using the small ec95d mesh.

2.2.2 config/asgs_config_irene_swan_garnet_r3.sh

This sample configuration file illustrates the settings required for the ASGS to run hurricane Irene with SWAN coupling on the Garnet machine at USACE ERDC using the FEMA R3 mesh for the US mid-Atlantic coast.

2.2.3 config/asgs_config_irene_swan_river_br_ncv6b.sh

During hurricane Irene, this sample configuration file was used in North Carolina to run the ASGS on the Blueridge Dell linux cluster at RENCI with SWAN coupling on the NC v6b mesh.

2.3 doc

This directory contains documentation for the ASGS, as described in the following subsections.

2.3.1 ASGS Operators Guide

ASGSOperatorsGuide.html (and its source document ASGSOperatorsGuide.txt) provide an overall view of the ASGS code, along with a description of how to configure the system to perform the desired type of simulations, and how to run the ASGS.

2.3.2 ASGS Developers Guide

ASGSDevGuide.html (and its source document ASGSDevGuide.txt) provides information to ASGS installers and developers to help with understand the structure and function of each piece in the project, and provides a set of step-by-step installation instructions.

2.3.3 Man Pages

Manual pages have been developed for some of the ASGS component programs, mainly as a way of providing a reference to the meaning of the command line options. Man pages are currently available for the following programs: *adcirc* (which includes *padcirc* and *padcswan*), *adcprep*, *aswip*, *nhc_advisory_bot.pl*, and *storm_track_gen.pl*. The content and number of these manual pages will continue to grow over time.

2.3.4 README Files

Two README Files are available, one for the asymmetric (NWS=9) and one for the configurable asymmetric (NWS=19) vortex meteorological models that are embedded in ADCIRC. These were written as initial documentation to provide the salient information about these models, and will be absorbed into the other types of documentation listed above over time. They will eventually be removed.

2.4 input

The input subdirectory contains the input files (mesh files, nodal attributes files, etc) and templates for dynamically generated input files as well as dynamically generated queue scripts. It also contains the ptFile_gen.pl script for generating a set of points for reprojecting NAM data into geographic projection.

2.5 logs

This subdirectory is meant for storing the log files that the ASGS generates during execution. The Operator should change to this subdirectory before executing the ASGS.

2.6 output

The output subdirectory contains all the routines that can be used when dealing with output and post processing. The code in this subdirectory can be categorized as follows:

1. Notification via email that new results are available: cera_notify.sh, ncfs_cyclone_notify.sh, ranger_notify.sh, corps_cyclone_notif ncfs_nam_notify.sh, ranger_parttrack_notify.sh, corps_nam_notify.sh, notify.sh.
2. File format transformations and conversions: asgsConvertR3ToNETCDF.pl, asgsConvertToNETCDF.pl, station_transpose.pl.
3. Controlling the generation of line plots, contour plots, images, etc.: ncfs_post.sh, part_track_post.sh, ranger_post.sh, post.sh, corps_post.sh, part_track_post_new.sh, r3_post.sh, asgsCreateKMZs.sh.
4. Posting results files to other locations over the network: corps_post.sh, asgsConvertR3ToNETCDF.pl.
5. Archiving of results: ncfs_archive.sh, ranger_archive.sh.

2.6.1 output/PartTrack

This subdirectory contains infrastructure for performing particle tracking post processing.

2.6.2 output/POSTPROC_KMZGIS

This subdirectory contains code for generating Google Earth (kmz) images and GIS shape files. Much of the underlying technology was developed by RENCI.

2.6.3 output/TRACKING_FILES

This subdirectory contains infrastructure for performing particle tracking post processing.

2.7 PERL

The PERL subdirectory contains the DateCalc perl module that is used for date math during execution, particularly for preparation of control files and meteorological forcing files.

2.8 tides

The tides subdirectory contains code and data for dynamically providing nodal factors and equilibrium arguments for simulations that include tidal forcing. It also has infrastructure for setting tidal boundary conditions.

3 Installation

The list of requirements for deployment of the ASGS at a new high performance computing facility is as follows:

3.1 Hardware Requirements

- ASGS can complete a single tropical cyclone forecast on the North Carolina grid (version 6b, 295328 nodes) in 1 hour 10 minutes on 384 2-year old Nehalem cores. A 5 member ensemble on this machine and using this grid would therefore require 1920 cores.
- The ASGS produced 15 GB of data per advisory with a single ensemble member on the NC v6b grid during Irene. Running one complete storm (estimated at 30 advisories) with 5 ensemble members would therefore require 15*5*30 = 2250 GB = 2.25 TB uncompressed ascii (includes hourly output from all full domain files, also includes all SWAN output). We are presently completing the conversion of ASGS output to NetCDF which will provide significant data storage savings.
- The OCPR Louisiana mesh has 1,088,315 nodes, so the use of this mesh to produce results for Louisiana would require 3.7x more resources than described for the NC v6b mesh, all other things being equal.
- The preferred distribution mechanism for distributing output data files is an opendap server, preferably one that shares a file system with the ASGS. A shared filesystem would allow results to be streamed to the OpenDAP system as they are produced, allowing visualization postprocessing to proceed in parallel with the generation of results.
- Incoming connectivity requirements include http and ftp for downloading data. Outgoing connectivity requirements include mail as well as the ability to expose results to an opendap server.

3.2 Software requirements

- The 2011stable version of the ASGS is compatible with the latest stable release version of ADCIRC v50. The latest trunk version of the ASGS always seems to require the latest trunk version of ADCIRC.
- Running the ASGS requires minimal external libraries and executables, because the System has been designed with portability in mind. These libraries and executables are as follows: a Fortran compiler including mpif90, a C compiler, MPI, NetCDF, GNU make, svn client, bash, perl, tar, gzip, screen, and mail.
- If graphical post processing is required, the list of requirements becomes much longer, including GMT, ImageMagick, ghostscript, gnuplot and RenciGETools and all their dependencies.

3.3 Installation Step-by-Step

1. Determine if your platform already meets the requirements listed in the previous two sections. If not, the unfulfilled requirements must be resolved.
2. If the ASGS is to be installed on an HPC system, determine if the target HPC system is already supported by the ASGS. This determination can be made by checking the *env_dispatch* subroutine in the *asgs_main.sh* source code.

   a. If the target HPC system is not found in the list in *env_dispatch*, then it is not already supported by the ASGS. Adding support for a new HPC system to the ASGS is not that difficult; please see the section entitled Pioneering below for details.
   b. If the ASGS is to be installed on a personal linux machine (i.e., a desktop or laptop without a queueing system where you just run parallel jobs directly via mpiexec), then use *desktop* as the environment.

3. If NAM data are to by used as input, compile *input/awip_lambert_interp.F* to an executable called *awip_lambert_interp.x* and place the executable in the ASGS base directory. Example instructions for compiling this program are listed in the comments at the top of the source file. This code is used with meteorological input from the NAM (North American Mesoscale) model to convert the data from a Lambert Conformal projection to geographic.
4. If NAM data are to be used as input, download and compile the wgrib2 package from NCEP's Climate Prediction Center (CPC): http://www.cpc.ncep.noaa.gov/products/wesley/wgrib2/ ... instructions for compiling this code are available from that site. The resulting executable should be named *wgrib2* and placed in the ASGS base directory.
5. Compile the program *tides/tide_fac.f* according to the instructions in the source code and name the resulting executable *tide_fac.x*, leaving it in the *tides* subdirectory.
6. Go to the directory that contains the ADCIRC source code that will be used by the ASGS and compile *adcprep*, *padcirc* and *hstime*. If tropical cyclone forcing will be used, also compile *aswip*. If wave coupling will be activated, also compile *padcswan*. Leave the executable files in place.
7. Collect up the ADCIRC input files that would normally be required for the mesh that will be used in real-time operation. At a minimum, this includes a fort.14 (mesh) and fort.15 (control) file. It may also include a fort.13 (nodal attributes), fort.26 (swan control), and swaninit files. Place these files in the *input* subdirectory. The actual file names are arbitrary; the ASGS does not require them to be named fort.14 etc.
8. Make a copy of the fort.15, swaninit, and fort.26 files; convert the copies into template files by removing key parameters and replacing them with a special string that the ASGS will use to fill in the parameters during operation (analogous to filling out a form). The template versions of the control files have the string *.template* appended to their file names by convention, but the names are arbitrary. Have a look at the files *input/FEMA_R3_fort.15.template* and *input/FEMA_R3_fort.26.template* to get an idea of how to turn a control file into a template. Its not complicated, but may be slightly tedious. It only has to be done once.
9. If there was a list of elevation or meteorological recording stations in the fort.15 file, cut and paste the lists of stations into separate files with the same format as required by the fort.15 ... the files *input/FEMA_R3_elev_stations.txt* and *input/FEMA_R3_met_stations.txt* are examples. The station lists should not appear in the fort.15 template file.

Once the above steps are complete, the ASGS should be ready to run and produce results. The installation of programs required for post processing are not included in the above procedure, and will be covered in a later iteration of this document.

Instructions for configuring an instance of the ASGS and starting it up are found in the ASGS Operators Guide (in *doc/ASGSOperatorsGuide.html*).

3.4 Pioneering

The term *Pioneering* is used to refer to the process of installing and running the ASGS on an HPC system where it has never run before. Because the ASGS is not specific to any particular type of HPC system, it is not too difficult to do this, particularly for an ASGS Developer that already has experience with the target HPC system and knows its idiosyncracies.

The differences between HPC platforms that are relevant to ADCIRC and ASGS generally fall into the following categories:

- c, f90, and mpif90 compilers, or different versions of those compilers
- netcdf libraries, versions of those libraries, and procedures for compiling programs that use netcdf
- differences in the type of queueing system that different platforms use for job submission (e.g., ASGS currently supports PBS, SGE, LoadLeveler, LSF and mpiexec)

- setting the PATH, LD_LIBRARY_PATH, and/or loaded "modules" interactively and in compute jobs so that programs can find their libraries when they run
- technical requirements and IT policies for submitting MPI jobs, including
  - differences in the way that the number of cores is actually specified when submitting a job
  - method for handling reservations and special high priority queue submission
  - method of transitioning from a lower priority to a higher priority queue
- technical requirements and IT policies for submitting single processor jobs (i.e., adcprep), including:
  - whether they can be run on a login node or must be processed through a queue
  - how different the submission of single processor jobs is from regular MPI jobs
    - different queue name?
    - different account number?
    - special characters on queue script submission line?
  - whether single processor jobs that require high memory (as adcprep often does on large meshes) have other special requirements for submission
- method of transmitting numerical and/or graphical results to end users
- optional: libraries available for graphical post processing on the target HPC system
- optional: available facilities and methods for archiving data

These differences will be described more fully in each of the sections below.

3.4.1 Compilers

The issue of compilers and compiler versions generally comes up with ADCIRC, rather than the ASGS itself. The first step in moving ASGS to a new system is to put ADCIRC on that system and compile *adcprep*, *adcirc*, *padcirc*, *padcswan*, *aswip* and *hstime*. ADCIRC supports a wide variety of compilers, but different HPC platforms have different compilers, and different compilers balk at different things. In contrast, the Fortran utilities required by the ASGS are generally easy to compile with just about any Fortran compiler.

3.4.2 NetCDF

The NetCDF libraries that are installed on HPC systems varies widely; some systems don't have it at all, while others have it installed in a directory that you have to have in your PATH to get things to compile and run, while others use a "module" system that requires the right modules to be loaded to compile and/or run programs with NetCDF.

If the PATH, LD_LIBRARY_PATH, or module state has to be set in a particular way to get NetCDF programs to run interactively, these settings will probably have to be duplicated in the queue scripts generated by the ASGS, which leads us to the next section.

3.4.3 Job Submission

The main difference between different HPC platforms that is relevant to the ASGS is the machine-specific way in which jobs are submitted for execution. There are many subtle differences in the ways that different machines handle job submission that each require different information to be supplied in different ways. Consider a few simple examples:

- Blueridge, a Dell Linux cluster at RENCI, uses PBS.
  - Job submission requires the number of "processors per node" to be specified in the queue script, as well as the total number of nodes that are requested (as opposed to requesting a certain number of processors) as well the name of the queue itself. If the number of CPUs is not evenly divisible by the number of processors per node, the ASGS must round the requested number of nodes up.

  - When running in the dedicated (high priority) queue on blueridge, the queue name must be specified as "armycore", whereas the normal priority queue is specified as "batch".
  - Furthermore, the number of processors per node is different for these two queues (PPN=12 if the queue name is "armycore" and PPN=8 if the queue name is "batch").
  - So, for example, for the ASGS to switch from a lower priority to higher priority queue, it has to be able to dynamically change the queue name, the number of processors per node, and recalculate the number of nodes, although the number of processors has not changed.

- Diamond, a Cray cluster at ERDC also uses PBS and has the following requirements for job submission:
  - the queue name for single processor jobs is different from the queue name for MPI jobs
  - there is a "trick" in the submission process for single processor jobs that need a lot of RAM; it requires the ASGS to actually request 0 CPUs for the serial job
  - there are different queue names for differently sized jobs (small MPI jobs are not allowed in the queue for large jobs, and vice versa) . . . this must be taken into account when testing ASGS on small meshes, then scaling up
  - if a dedicated reservation has been awarded (for high priority runs), the ASGS must start using use special, one-time-only queue names for serial and parallel jobs
  - when running on a dedicated reserved queue, a different account number must be used with the new queue names; this account number cannot be used with the regular queues

The examples above illustrate that (a) different HPC platforms have different requirements for job submission, even if they use the exact same queueing system (PBS for example); and (b) a set of job submission rules and requirements that are simple enough to accommodate when manually running individual jobs on a single machine can require a suprising amount of attention to generalize and automate reliably for the full range of anticipated use cases.

The ASGS deals with all these issues using a template approach. That is, an ASGS Developer starts with some manually developed queue scripts for various types of jobs on the target platform, and abstracts them to a template (or a group of templates in some cases). There are several templates available for currently supported platforms (e.g., *input/erdc.adcprep.template.pbs* and *input/ranger.template.sge*, etc) to use as examples.

The other mechanism that the ASGS uses to manage these requirements is the use of pluggable template filler scripts. The ASGS Developer must write a script or scripts to properly fill in the queue script template(s) described above. There are several existing template filler scripts (e.g., *erdc.pbs.pl*, *loadleveler.pl*, *queenbee.pbs.pl*, *ranger.serial.pl*, *ranger.sge.pl*, and *tezpur.pbs.pl*) for existing platforms that can be used as examples.

Finally, the ASGS Developer must update the *env_batch()* subroutine in *asgs_main.sh* to specify---among other things---the names of the template and template filler that will be used for that platform. When first starting the ASGS, the Operator simply selects the name of the computing environment on the command line, e.g., something like *garnet* or *blueridge* or *desktop*. The ASGS checks to see if the specified machine is supported using the *env_dispatch()* subroutine, which then sets the names of the queue script templates and queue script filler scripts that are appropriate for that platform.

Here is an illustration of this template-based configurability in action:

- In order to run *adcprep*, the ASGS calls the *prepFile()* subroutine, which checks to see if the queueing system was specified as either Sun Grid Engine (SGE, which is used on Ranger at TACC), or Portable Batch System (PBS, which is used on many other machines).
- if the queue system is PBS
  a. the ASGS calls a queue script generation program, whose name is configurable
  b. it feeds the queue script generation program a queue script template, whose name is also configurable
  c. it provides a variety of command line options, such as the number of processors per compute node, the number of compute processors the job should run on, the account number for the job, and the name of the queue
  d. the resulting queue script is then submitted with *qsub*
  e. the ASGS monitors for completion
- if the queue system is SGE

a. the ASGS performs a similar set of actions as for the PBS case above, but with a shorter set of command line options to the queue script generator
b. the ASGS could also perform a "resubmit" step, since SGE on Ranger at TACC had a habit of intermittently rejecting valid compute jobs

- if the queue script is neither PBS nor SGE, the ASGS executes *adcprep* directly, that is, on the login node if running on an HPC machine, or (more likely) on the command line, if running on a desktop or laptop

3.4.4 Results Transmission

The transmission of results to end users is considered part of the post processing in the ASGS. Different ASGS installations will vary widely on the type of post processing that they want to do in-situ, that is, right there in the HPC environment where the ASGS is running and the results are being generated.

This diversity is supported in the ASGS by allowing the ASGS Developer to simply supply the name of an executable (generally a shell script) that the ASGS should run at the end of each forecast. This allows the ASGS Developer to insert any type of post processing at all, and maintains the modular structure of the overall system.

The most basic post processing is simply transmission of numerical results to an external server. In this case, the post processing script would contain an *scp* command to copy the result files. In order to avoid an interactive password prompt, the ASGS Operator's private key should be copied to the proper user account on the receiving server, and key authentication should be enabled for ssh.

3.4.5 Post Processing

The ASGS already supports a great deal of built-in post processing options that have been required over the years by different sites. These include generation of jpgs of contour plots using FigureGen, generation of GIS shape files of results and generation of Google Earth visualizations of results using RenciGETools, generation of line plots of elevation and wind speed using gnuplot, particle tracking algorithms, conversion of ascii adcirc output to netcdf, and posting of results to external opendap servers. Many of these output types have their own dependencies, which may or may not be satisfied by the target HPC environment. Please see the *output* subdirectory for samples of various post processing scripts that are already available.

3.4.6 Archiving

The final consideration in platform-, machine-, and site-specific customization is the use of data archiving. Some environments have a dedicated archival storage system, while others have no long term storage facilities at all. The ASGS handles site-specific archiving with the same approach as it does with post processing: it allows the ASGS Developer to supply the file name of a script that performs whatever actions will be necessary to archive the results. The ASGS executes the script once all ensemble members in forecast have been completed.

Because of the nature of archiving (data compression, data copying or transmission which may take a long time), the archive script is executed with an ampersand, that is, it is run in the background. This allows the ASGS to get on with looking for the next cycle or advisory while the archive script packages up the previous cycle or advisory.

Please see the *output* subdirectory for samples of various existing archive scripts for hints and ideas about generating new archiving scripts.

4 Meteorological Forcing

The ASGS is capable of using gridded met fields from NCEP's NAM model, or two different types of vortex forcing with ADCIRC's asymmetric wind models (NWS9 and NWS19). An outline of the procedure used by ASGS for obtaining and processing these input data is provided below.

4.1 NAM Forcing

The National Centers for Environmental Prediction (NCEP, a division of the National Oceanic and Atmospheric Administration, NOAA) runs the North American Mesoscale model (NAM) four times per day, producing a nowcast and a 3.5 day forecast of the atmosphere over North America.

The NAM data are produced on a Lambert Conformal grid with 12km spacing and written in a binary file format called grib2. The ASGS uses the following procedure to prepare these data for use in ADCIRC.

1. The *get_nam.pl* program is used to connect to the NCEP ftp site once per minute and check the dates and times of the most recent NAM output files. If it finds files that have dates and times that are more recent than the most recent ADCIRC hotstart file, a new cycle is deemed to have started, and the new files are downloaded in preparation for a nowcast run.
2. Once the data have been downloaded, the ASGS runs the *NAMtoOWI.pl* script to perform the following procedures:
    a. extract the data that are required (u, v, p at 10m or MSL . . . the grib2 files also have a lot of data that is not needed by the ASGS)
    b. call *awip_lambert_interp.x* on each grib2 file to reproject the data from Lambert Conformal to geographic projection and interpolate the data onto a grid that is regular in geographic coordinates (which is required for the OWI file format) using an external list of lat,lon points (e.g., *input/ptFile.txt*)
    c. write the data in OWI format to fort.221 and fort.222 files, and generate a companion fort.22
3. The ASGS then uses *control_file_gen.pl* and the meteorological files that were generated by *NAMtoOWI.pl* to properly formulate the ADCIRC fort.15 control file (and SWAN fort.26 file, if any).
4. Once the nowcast job has been submitted and completed, the ASGS calls *get_nam.pl* again to download the NAM forecast data for the same cycle, using the steps listed above.

The forecast data are not downloaded at the same time as the nowcast data, because NCEP releases the NAM data files as they are generated, which can take more than an hour. As a result, it makes more sense to download and run the nowcast while NCEP continues to post forecast data, then come back when the nowcast run is complete and download the forecast.

The ASGS may be able to finish the nowcast before NCEP finishes posting forecast data, depending on the size of the mesh, number and speed of the processors the ASGS is using, and queue congestion, if any. If that happens, *get_nam.pl* just downloads the forecast data as they become available, and when they have all been downloaded, the ASGS goes on to set up and run the forecast as described above.

4.2 Parametric Vortex Forcing

The ASGS can also be configured to use data from the National Hurricane Center (NHC) to generate parameter files for use in ADCIRC's internal configurable asymmetric vortex model (ADCIRC NWS=19).

4.2.1 Background

The asymmetric vortex wind model input differs from most other types of wind input in ADCIRC in that it consists of storm parameters, rather than the meteorological data itself. ADCIRC then uses these parameters to generate the actual meteorological data internally at each node at every time step during the simulation. The result is that the fort.22 files are very small (e.g. 20kB), in comparison with fort.22 files that contain actual meteorological data.

The format of the fort.22 file for the asymmetric wind model is the ATCF (a.k.a. "BEST track") format. This format was developed by the U.S. Navy, and ATCF stands for Automated Tropical Cyclone Forecast. Historical tracks, real-time hindcast tracks and real-time forecast tracks may be found in this format. The format is documented in detail at the following web site:

http://www.nrlmry.navy.mil/atcf_web/docs/database/new/abrdeck.html

One important thing to remember about this format is that it looks like CSV data, but it is actually fixed column width data. Never change the width of the columns when you edit this data!

It is assumed by the asymmetric wind code within ADCIRC that the first entry in the fort.22 file corresponds to the cold start time (ADCIRC time=0) if the simulation is cold started, or to the hotstart time if the simulation is hotstarted. Therefore, the forecast period (column #6) needs to be edited by the workflow scripts to reflect the time of the forecast/nowcast for each track location (each line) in hours from the start of the simulation (0, 6, 12, 18, etc). The original data in that column depends on what type of best track format data is being used. The original data might have 0 or other numbers in that column.

The behavior of the configurable asymmetric meteorological model (NWS=19) is similar to its ancestor, the original asymmetric meteorological model (NWS=9). However, the configurable asymmetric vortex wind model was developed for several reasons: (1) to allow more visibility into the parameters, such as Rmax, that control a storm's size and shape and are calculated within the wind model code; (2) to allow the user to control parameters such as Rmax so that they may be adjusted by the user; and (3) to allow the user to deterministically compensate for input data that are missing or nonexistent (such as wind radii in various quadrants for a particular isotach).

The mechanism for achieving the goals described above is a preprocessing program called the asymmetric wind input preprocessor, or *aswip*, that takes the ATCF formatted input data that would normally be used for the NWS8 or NWS9 and adds columns to it that describe the following things:

1. The Rmax from the hindcast (i.e., BEST lines) has been persisted from the Rmax column (described as MRD in the ATCF documentation) from the value in the forecast (i.e., OFCL lines).
2. The storm direction DIR and speed SPEED in the ATCF file have been replaced with the calculated direction and speed to be used by ADCIRC. The values are provided in the same format as the ATCF file to provide compatibility between methods. The speed in given in knots and the direction is given in compass coordinates, with zero degrees indicating North and values increasing clockwise.
3. In the 2nd column after the storm name, the cycle number is provided. A *cycle* is an entry or set of entries in the file that all have the same storm time or forecast period. For cycles that have more than one isotach, this value will be repeated for each isotach (starting from 1 for the first cycle in the file).
4. The 3rd column after the storm name contains the number of isotachs that are reported for that particular cycle. This value is also repeated on each line for each isotach that is reported per cycle. For example, if the cycle has a 34kt and a 50kt isotach entry then this column will contain a *2* for both entries in that cycle.
5. The following 4 columns contain the flags that tell the ADCIRC NWS 19 code whether or not to use a particular wind radius from the isotach under consideration. There is a flag for each quadrant. A 0 indicates that the wind radius for that isotach and quadrant will not be used. A 1 indicates that the wind radius for that isotach and quadrant will be used. For example: if only the 34kt isotach is provided, then then all four wind radii must be used, and the columns will all be set to 1.
6. In the next 4 columns, the calculated Rmax for each quadrant is listed in the following order: NE SE SW NW.
7. The next column contains the overall Holland B value.

Another example: if 3 isotachs are provided then the columns may look like the following:

```
34 ... 3 0 0 0 0 ...
50 ... 3 0 0 1 1 ...
64 ... 3 1 1 0 0 ...
```

this indicates -

- use NO radii from the 34 kt isotach
- use the 3 & 4 quadrant radii from the 50 kt isotach
- use the 1 & 2 quadrant radii from the 64 kt isotach

Users could potentially modify these flags in the input file to manually select which radii to use for each cycle.

Finally, one valuable aspect of this file format is that it can be used by NWS8, NWS9, or NWS19, since the original data have not been modified. The extra columns are used as input by NWS19, and as a result, they provide both metadata and control over these parameters.

4.2.2 Procedure

The steps that the ASGS uses to prepare the Forecast/Advisory data for use in ADCIRC are described below.

1. The ASGS uses the *get_atcf.pl* script to check the NHC RSS feed (which is just an xml text file they keep on their web site). The RSS feed contains the current advisory number and a hyperlink to the html-formatted file containing the forecast/advisory text.
2. The *get_atcf.pl* starts by downloading the latest ATCF formatted hindcast file from the NHC anonymous ftp site. The current name of the storm is parsed out of the hindcast file for use in *get_atcf.pl* during parsing of the RSS feed; e.g., the storm name can change from TD TWO to TD BERTHA from one advisory to the next. The hindcast data are also used later when ASGS calls *storm_track_gen.pl*.
3. The *get_atcf.pl* script then downloads the NHC forecast/advisory as follows
    a. if the ASGS has just started, it follows the hyperlink in the RSS feed and downloads the current forecast/advisory
    b. if the ASGS has already run through a complete advisory cycle, it polls the NHC RSS feed once per minute . . . when it detects that the advisory number has been updated in that file, it follows the hyperlink in the RSS xml file and downloads the current forecast/advisory, returning the new advisory number to the ASGS
4. The ASGS calls the *nhc_advisory_bot.pl* script to parse the forecast/advisory text out of the html and parse the relevant parameters out of the forecast advisory text to produce an ATCF-formatted forecast file.
5. After *get_atcf.pl* and *nhc_advisory_bot.pl* have run, the ASGS has an ATCF-formatted hindcast file (containing the past and present states of the tropical cyclone) and an ATCF-formatted forecast file containing the present and predicted future states of the tropical cyclone.
6. The ASGS then calls *storm_track_gen.pl*, providing it with the current hotstart time in ADCIRC; the *storm_track_gen.pl* script melds the hindcast and forecast data together such that it
    a. starts at the current ADCIRC hotstart time
    b. ends at the end of the nowcast period, which is the same as the end of the hindcast data, if the run is a nowcast run
    c. ends at the end of the forecast period, if the run is a forecast run
    d. has the specifed track perturbations applied, if any perturbations were specified, and if the run is a forecast run
7. The *storm_track_gen.pl* script writes out the melded data to an ATCF-formatted fort.22 file.
8. The ASGS then calls *aswip* to process the fort.22 file produced by *storm_track_gen.pl*; *aswip* calculates the Rmax in each quadrant and writes out an NWS_19_fort.22 for use in ADCIRC.

After the NWS_19_fort.22 file has been produced, the ASGS feeds it to *control_file_gen.pl*, which generates a fort.15 file that will cover the time period specified by the parameters listed in the NWS_19_fort.22 file.

5 Development Strategy

The ASGS development strategy consists of the maintenance of two separate versions at any given time: the stable version and the development version. The stable version only receives bug fixes, while the development version receives bug fixes as well as new features.

At some point, the development version becomes stable. At that time, a stable branch is created from it, and then new features are added to the previosly stable code, creating a new development version.

As of this writing (7 May 2012) the latest stable version is on the branch named 2011stable. The ASGS trunk is where new features are developed.

6 Subversion

Subversion is a version control system. That is, it provides a means for many software developers to collaborate on a single software project.

A subversion repository is a storehouse of code for a particular project. Subversion repositories consist of a trunk, which is the mainline code that everyone shares, and any number of branches that are created by and for individual users.

A branch may be created by a developer that has a lot of changes to make over a longer period of time. This lone developer can work on a branch without affecting the mainline code. Once the changes in the branch are satisfactory, the branch can be merged back to the trunk, making the changes available to all.

A further advantage of Subversion is that it automates the process of merging new features into ASGS. For example, in the past, without Subversion, developers modified their local copies of ASGS and wanted to merge the modifications into the mainline code. This required hours or days of painstakingly examining each change they had made and cutting and pasting the changes one by one into the up-to-date version of ASGS. With Subversion, the software examines the changes that were made locally and makes corresponding changes to the mainline version in an automated way.

6.1 Policies

Working with subversion requires greater coordination among ASGS developers. For example, if two developers change the same line of code in two different branches in two different ways, this will create a conflict that will have to be resolved manually.

Furthermore, if one developer makes changes to trunk that prevent the system from running, it will be difficult for other developers to continue working if they update to the latest code. As a result, ASGS development with Subversion will adhere to the following policies:

- Communicate. Jason Fleming is responsible for making sure ASGS development is smooth, pain-free and productive---when in doubt, email him (jason.fleming@seahorsecoastal.com).
- Make a branch to develop new features, rather than making changes to trunk.
- Don't commit changes to trunk that might cause issues that prevent the system from running, at least on your platform.

6.2 Instructions

Comprehensive documentation for Subversion is available: http://svnbook.red-bean.com/.

The ASGS code is hosted at the following Subversion repository location:

```
http://www.seahorsecoastal.com/svn/asgs
```

To check out code from the trunk, please type

```
svn checkout http://www.seahorsecoastal.com/svn/asgs/trunk
```

Here are a few examples of things that we commonly do with svn.

6.2.1 Compare Revisions

In order to see the changes between revisions of the repository, type (for example)

```
svn diff -r 15:16 control_file_gen.pl
```

6.2.2 Package Up Code From Repository

In order to create a tar file that contains just the ASGS code and excludes the .svn directories, the first step is to export the code to a new subdirectory. For example:

```
svn export ./trunk ./asgs_today
```

Would copy the local files under version control from the trunk subdir to the asgs_today subdir, excluding the .svn files. Particular revisions of the svn repository can also be extracted, see the svn documentation for details on how to do this.

6.2.3 Pull Old Version from Repository

One advantage of using svn is that it is easy to go back to an older version of the code, if necessary. The first thing to do in this case is usually to look at the svn log to see what changes were made, so that you can select the right version of the repository to extract. Go to the subdirectory of the code tree that you want an older version of and issue the command

```
svn log
```

Look at the log entries, and note the revision that you'd like to extract. Let's say you want to retrieve asgs/trunk at revision 65 of the repository (for example). Issue the following command:

```
svn checkout -r 65 http://www.seahorsecoastal.com/svn/asgs/trunk
```

6.2.4 Make a Branch

To make a branch off trunk to add a new feature and call it the myfeature branch:

```
svn copy http://www.seahorsecoastal.com/svn/asgs/trunk http://www.seahorsecoastal.com/svn/
```

Then to start using the newly created branch:

```
svn checkout http://www.seahorsecoastal.com/svn/asgs/branches/myfeature
```

A This Document

This document was prepared from the text file ASGSDevGuide.txt using software called asciidoc (http://www.methods.co.nz/-asciidoc/). The document can be formatted as an html page with the command

```
asciidoc --backend html5 -a toc2 ASGSDevGuide.txt
```

or formatted as a pdf with the command

```
a2x --format=pdf -a toc2 ASGSDevGuide.txt
```