Parametric Hurricane Modeling System

PaHM Manual

September 2022

Panagiotis Velissariou (Panagiotis.Velissariou@noaa.gov)

Saeed Moghimi (Saeed.Moghimi@noaa.gov)

Contents

Abstract

Over the years, various parametric wind models have been developed to estimate the surface winds within a tropical cyclone given the track of the storm. Such models can be very useful on forcing ocean and wave models in storm surge simulations, as they are lightweight and they do not require much time or computational resources to produce the wind fields on the fly for the duration of the storm. The Parametric Modeling System *PaHM* is developed to be used as a general atmospheric modeling system to support coastal applications.

PaHM is an ESMF/NUOPC compatible modeling system that can be used either as a standalone atmospheric model, or as an atmospheric modeling component coupled with ocean and wave models via NOAA's Environmental Modeling System (NEMS), a common modeling coupling framework that implements the National Unified Operational Prediction Capability (NUOPC). The core modeling components of the system are the Holland Model (Holland 1980) and the Generalized Holland Parametric Tropical Cyclone Model (Gao et al. 2015, Gao 2018).

PaHM is developed at the Coastal Marine Modeling Branch under the office of Coastal of Coast Survey at NOAA's National Ocean Service. In a "standalone" configuration it ouputs gridded atmospheric fields to force any ocean/wave model while, in its "coupling" configuration, it feeds the atmospheric fields to the couple ocean/wave model via its own NUOPC Cap. The source code of *PaHM* can be accessed at: https://github.com/noaa-ocs-modeling/PaHM.

Chapter 1

Introduction

Hurricanes are devastating storm events resulting in flooding and destruction in coastal areas due to the severe rise of water level that inundates the vast majority of the low-lying areas. This rise in water level (storm surge) is caused by the surface wind stress and the atmospheric pressure deficit (atmospheric forcing), and the mass and momentum transfer from the wind generated surface waves (wave setup). To mitigate the storm's impacts, local officials need to have "quick" access to "accurate" predictions/forecasts of total water levels and flood inundation in preparation of flood protection and evacuation. Storm surge prediction systems include an atmospheric, a storm surge and a wave model, all coupled together to produce reliable total water level predictions. Such systems require an atmospheric model that generates "accurate" forcing atmospheric fields fast.

In this section, is introduced the parametric tropical cyclone (TC) modeling system *PaHM* developed by the Storm Surge Modeling Team under the Office of Coast Survey (OCS), NOAA. *PaHM* contains various light-weight TC models that require minimal computational resources to produce the wind fields on-demand fast (in the order of minutes) and efficiently. While, such models use limited physics they produce "accurate" wind fields in the vicinity of the storm's path due to the great advancements made in the last decades in the determination of the storm's path and intensity (Vickery et al. 2007). Furthermore, *PaHM* includes build-in automated QA test capabilities, to ensure that the system always delivers the best outcomes under various controlled simulation scenarios.

Chapter 2

Parametric Hurricane Modeling System (*PaHM*)

The Parametric Hurricane Modeling System is not just another parametric atmospheric model but rather an atmospheric modeling system that contains multiple TC models that can be activated during run time to generate the required atmospheric wind fields. Currently, the core parametric models in *PaHM* are the Holland 1980 and the Generalized Asymmetric Vortex Holland models. In development is the process of extendeding the built-in I/O capabilities (CSV, NetCDF and GRIB interfaces) to allow for the digestion and the manipulation of different data formats.

To calculate its wind fields, *PaHM* reads "best track" type of files (e.g., those produced by the National Hurricane Center to generate gridded atmospheric fields (usually, 10-m wind velocities and atmospheric pressures converted to mean sea level). The file formats currently recognized by *PaHM* are: a a/b-deck, b HurDat2, c IBTrACS and d TCVitals. *PaHM* has a built-in CSV I/O interface therefore, it can read and write any of these files in ASCII format.

2.1 Downloading *PaHM*

The source code of *PaHM* is publicly available from the GitHub repository: https://github.↩
com/noaa-ocs-modeling/PaHM (binary distributions of *PaHM* are not currently available).

The online documentation of the modeling system is hosted by GitHub Pages:

- HTML version: https://noaa-ocs-modeling.github.io/PaHM/html/index.html
- PDF version: https://noaa-ocs-modeling.github.io/PaHM/pahm_manual.pdf

PaHM can be downloaded using one of the following methods:

1. Clone the source code from GitHub using the command:

   ```
   git clone https://github.com/noaa-ocs-modeling/PaHM PaHM
   ```

   The source will be downloaded into the target directory PaHM.

2. Download the source archive using the command:

   ```
   wget https://github.com/noaa-ocs-modeling/PaHM/archive/refs/heads/main.zip
   ```

   and extract the sources in the PaHM directory by issuing the following commands:

   ```
   unzip -o main.zip  (the data will be extracted into the PaHM-main directory)
   mv PaHM-main PaHM  (move the extracted files to the PaHM directory)
   ```

   Even if an archive is sufficient, it is advisable to use the distributed version control system Git to follow the *PaHM* development to merge easily to new versions. New Git users are invited to read some of the online guides to get familiar with vanilla Git concepts and commands:

- Basic and advanced guide with the Git Book.
- Reference guides with the Git Reference.
- GitHub reference sheets with the GitHub Reference.
- Manage your GitHub repositories with Git Using Git.

2.1.1 Directory Structure

After downloading *PaHM*, let us look at the physical directories and the source code and configuration files that come with the system. The "*PaHM* ROOT" directory contains the source and all configuration files required to build and run *PaHM* (see Figure [1]). The directories of interest are the **src**, **scripts**, **cmake** and **inputs** directories.

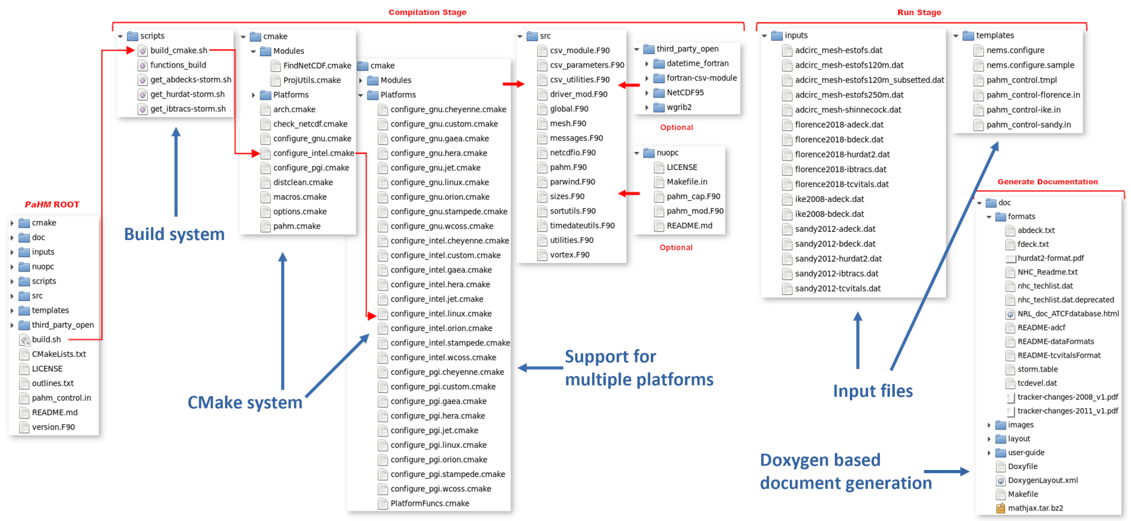


Figure 1: Directory tree of the *PaHM* modeling system.

- **src:** Contains all the Fortran code of *PaHM*.
- **scripts:** *PaHM* uses Bash for its automated build infrastructure and this directory contains all system "build" scripts required to build the model. The top level build.sh script is the script to run to create the *PaHM* executable (extended details are given in section [The Build System]).
- **cmake:** The "build" system of *PaHM* is supported by CMake that manages the build process in a compiler-independent manner. This directory contains the CMake all related configuration files and modules for *PaHM* that are required to properly configure the build process for the host operating system (more details are given in section [CMake Configuration Files]).
- **inputs:** Contains the input data files for *PaHM*, like storm track files and grid/mesh files required to generate the wind data for. As *PaHM* comes with some basic (sample) input files, it is the user's responsibility to populate this directory with additional data files required for the particular simulation(s).
- **templates:** Contains sample control files for *PaHM* required to run the model. The user may copy any of those files into *PaHM's* root folder and modify it for the particular simulation (further details are given in section [[pahmsys_conf]]). In the same directory are included some NEMS configuration files required when *PaHM* is used as a coupled modeling component (see section [[pahmsys_conf_coupled]]).
- **third_party_open:** Contains the sources of third party libraries that the user can use to compile into the *PaHM* executable (if required). The libraries shipped with the *PaHM* distribution are fully open source without any particular licensing restrictions.
- **doc:** This directory contains the documentation sources (markdown .md files) to build *PaHM's* documentation. The documentation is built outside *PaHM's* build

system and it uses the Doxygen documentation creator. It is noted here that all *PaHM* Fortran files have been fully commented using the Doxygen/Markdown directives. More details on how to generate the documentation can be found in the doc/Readme.md file.

2.2 Building *PaHM*

The build system of *PaHM* is designed to compile the model requiring a minimal intervention from the user. *PaHM* supports different compiler families and various Linux/UNIX computing platforms (HPC, clusters, personal desktops, etc).

2.2.1 System Requirements

To compile *PaHM* the following programs and libraries are required:

- Fortran and C compilers: The compilers tested are Intel $\geq$ 18, GCC $\geq$ 4.8 and PGI/NVidia $\geq$ 20.11.
- A recent version of CMake ($\geq$ 3.2).
- Git, the distributed version control system.
- NetCDF-4: the Network Common Data Form (NetCDF) C and Fortran libraries (usually installed in the host OS).
- HDF5: the High-performance software and Data Format (HDF) libraries (usually installed in the host OS).

For a successful compilation and run of *PaHM*, it is required that the model and the Net↩ CDF/HDF5 libraries are built with the same compiler family (not necessarily the same version). This requirement is important as the creation of the NetCDF output files will as this is the case of all modeling systems that use these libraries. Currently, *PaHM* supports the GNU, Intel and Portland Group/NVidia compiler families.

2.2.2 The Build System

To build *PaHM* the user should run the build.sh bash script located (a link to the scripts/build.sh) in the root directory of the downloaded source code (Figure [1]). The following steps will help you to build *PaHM* on your local machine or your cluster of choice.

1. Make sure that CMake and the NetCDF/HDF5 libraries are in the user's PATH environment (in a Cluster/HPC system that uses the environment module system, the user should load the modules for cmake and netcdf, hdf5 before building *PaHM*).

2. Change the working directory to the *PaHM* root:

   ```
   cd /INSTALL_PATH/PaHM
   ```

3. Run the build script as:

   ```
   ./build.sh
   ```

   to get the following screen output:

   ```
   The following variables are defined:
       ACCEPT_ALL         = 0
       PROJECT_SOURCE_DIR = /INSTALL_PATH/PaHM
       PROJECT_BINARY_DIR = /INSTALL_PATH/PaHM/build
       CMAKE_SOURCE_DIR   = /INSTALL_PATH/PaHM
       CMAKE_BINARY_DIR   = /INSTALL_PATH/PaHM/build
       INSTALL_DIR        = /INSTALL_PATH/PaHM
       CMAKE_BUILD_TYPE   = Release
       CMAKE_FLAGS        = -DCMAKE_BUILD_TYPE=Release -DCMAKE_Platform=linux
                            -DCMAKE_VERBOSE_MAKEFILE=TRUE
       PLATFORM           = linux
       COMPILER           = Undefined, Supported values are: [gnu, intel, pgi]
       CC                 = Will use the OS default compiler
       CXX                = Will use the OS default compiler
       FC                 = Will use the OS default compiler
       F90                = Will use the OS default compiler
       VERBOSE            = 0

   Are these values correct? [y/n]:
   ```

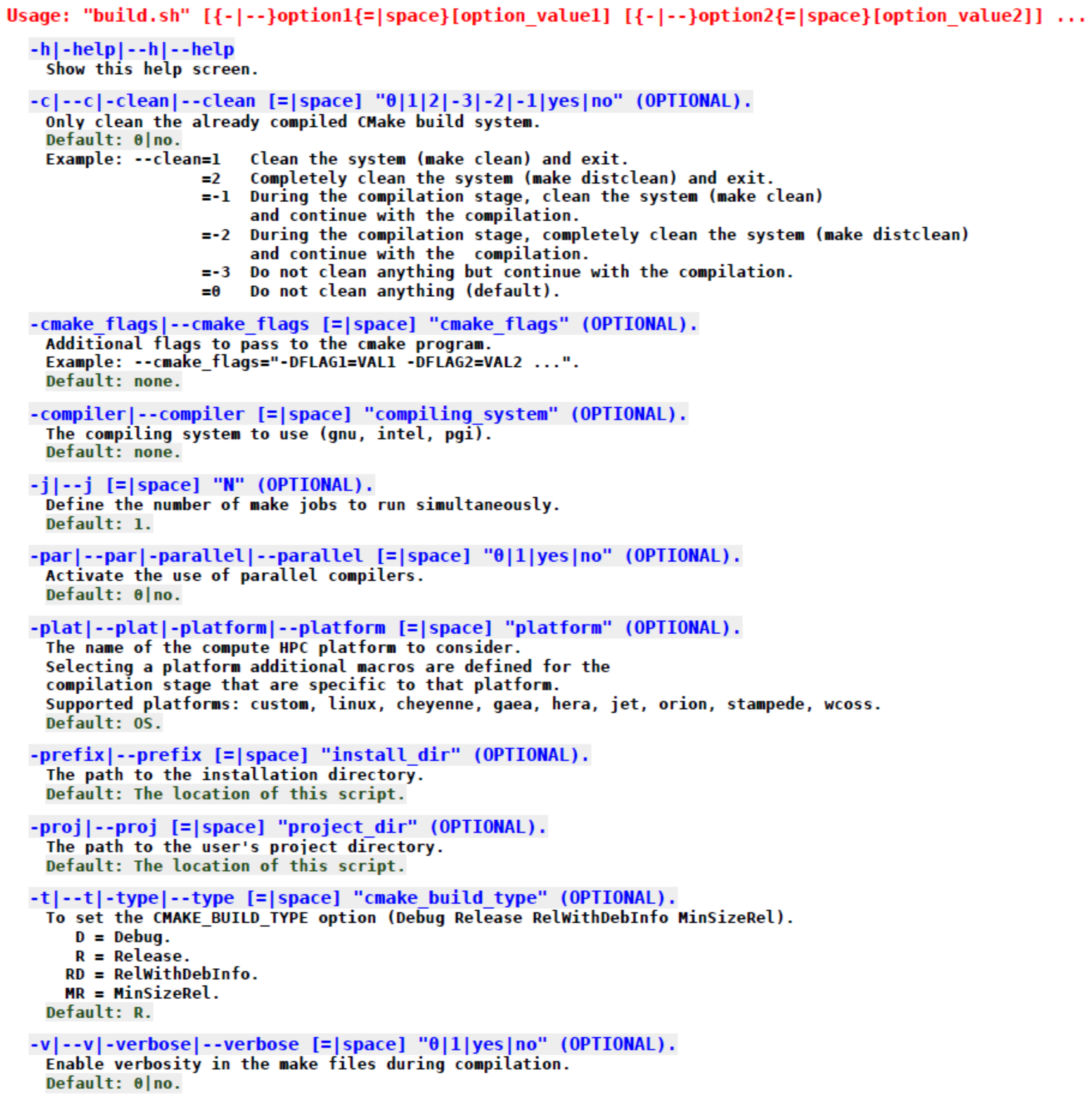
   if the above settings are correct, type "y + <ENTER>" to continue with the compilation. Upon successful compilation the *PaHM* executable will be placed into the "bin" directory at the *PaHM* root.

The build script tries to determine the host OS, either Linux or, MacOSX (in Windows the user should try to compile *PaHM* inside a Windows Subsystem for Linux - WSL, or another Linux setup). By default, the build script will use the native Fortran and C compilers (usually gcc and gfortran) for the compilation stage. The script accepts options to further customize the compilation process and it is suggested that the user should at first run the script as: `./build.sh --help` to see all available options as shown in Figure [2].

The two influential options of the script are `--compiler` and `--platform` where the user can supply what compiler family to use and what platform to compile *PaHM* for (in the cmake folder there are customized platform cmake modules that the script loads per user's request). For example, to compile *PaHM* using the Intel compilers on the Orion HPC cluster of Mississippi State University, the user should run the script as:

```
./build.sh --compiler intel --platform orion
```

Figure [2], shows all currently available options in "build.sh" accompanied by the detailed explanation for each option.



```
Usage: "build.sh" [{-|--}option1{=|space}[option_value1] [{-|--}option2{=|space}[option_value2]] ...
  -h|-help|--h|--help
    Show this help screen.

  -c|--c|-clean|--clean [=|space] "0|1|2|-3|-2|-1|yes|no" (OPTIONAL).
    Only clean the already compiled CMake build system.
    Default: 0|no.
    Example: --clean=1   Clean the system (make clean) and exit.
                    =2   Completely clean the system (make distclean) and exit.
                    =-1  During the compilation stage, clean the system (make clean)
                         and continue with the compilation.
                    =-2  During the compilation stage, completely clean the system (make distclean)
                         and continue with the  compilation.
                    =-3  Do not clean anything but continue with the compilation.
                    =0   Do not clean anything (default).

  -cmake_flags|--cmake_flags [=|space] "cmake_flags" (OPTIONAL).
    Additional flags to pass to the cmake program.
    Example: --cmake_flags="-DFLAG1=VAL1 -DFLAG2=VAL2 ...".
    Default: none.

  -compiler|--compiler [=|space] "compiling_system" (OPTIONAL).
    The compiling system to use (gnu, intel, pgi).
    Default: none.

  -j|--j [=|space] "N" (OPTIONAL).
    Define the number of make jobs to run simultaneously.
    Default: 1.

  -par|--par|-parallel|--parallel [=|space] "0|1|yes|no" (OPTIONAL).
    Activate the use of parallel compilers.
    Default: 0|no.

  -plat|--plat|-platform|--platform [=|space] "platform" (OPTIONAL).
    The name of the compute HPC platform to consider.
    Selecting a platform additional macros are defined for the
    compilation stage that are specific to that platform.
    Supported platforms: custom, linux, cheyenne, gaea, hera, jet, orion, stampede, wcoss.
    Default: OS.

  -prefix|--prefix [=|space] "install_dir" (OPTIONAL).
    The path to the installation directory.
    Default: The location of this script.

  -proj|--proj [=|space] "project_dir" (OPTIONAL).
    The path to the user's project directory.
    Default: The location of this script.

  -t|--t|-type|--type [=|space] "cmake_build_type" (OPTIONAL).
    To set the CMAKE_BUILD_TYPE option (Debug Release RelWithDebInfo MinSizeRel).
       D = Debug.
       R = Release.
      RD = RelWithDebInfo.
      MR = MinSizeRel.
    Default: R.

  -v|--v|-verbose|--verbose [=|space] "0|1|yes|no" (OPTIONAL).
    Enable verbosity in the make files during compilation.
    Default: 0|no.
```

Figure 2: Build script (build.sh) options.

2.2.3 CMake Configuration Files and Modules

The directory "PaHM/cmake" contains the CMake modules and auxiliary files used for the configuration of the CMake based build system used by *PaHM*, so that many setup and com[ilation steps can be automated. The most common steps to build, and install software based on CMake, like *PaHM* are:

1. Create a "build" directory and change into it.
2. Run CMake to configure the build tree (for a minimally CMake configuration run: `cmake ../`).
3. Build the software using the generated Makefile (run: `make`).
4. Install the built files (run: `make install`).

In the parent directory of the "build" folder (PaHM/) it should be located a file called "↩ CMakeLists.txt" that CMake requires for its configuration. This "CMakeLists.txt" in addition of calling CMake modules from the PaHM/cmake folder, it also contains CMake directives that modify the behavior of the configuration process. In step (2) above the user can supply additional flags defined in "CMakeLists.txt" and in PaHM/cmake modules or global CMake flags that might influence the compilation stage as shown in the following example:

```
cmake ../ -DENABLE_DATETIME=TRUE -DF90=ifort
          -DINSTALL_DIR="some installation location"
```

The *PaHM* CMake system will pick-up the values of these variables if set in the user's environment, and continue with the configuration with these variable definitions, for example:

```
In Bash use:

export F90=ifort
export INSTALL_DIR="some installation location"

and run cmake as:

cmake ../ -DENABLE_DATETIME=TRUE
```

The user is encouraged to take a look of the "CMakeLists.txt" and the PaHM/cmake modules to gain an understanding on how the CMake build system in *PaHM* works. By design, the CMake build system in *PaHM* uses the user's environment as well as the command line interchangeably to pick-up the values of any user supplied variables related to *PaHM*. The main CMake module groups in *PaHM* are (a) the System group, (b) the Compiler group and (c) the Platform group (see Figure [1]).

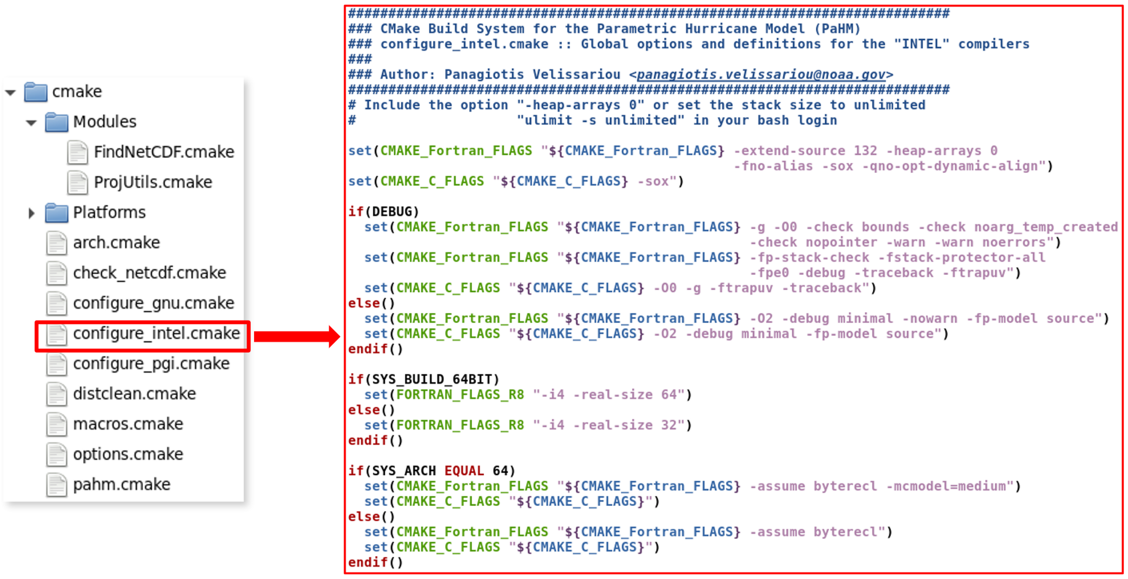
(a) System group : Consists of the core CMake modules required to configure and compile *PaHM*:

arch.cmake :	Architecture specifications
check_netcdf.cmake :	Checks for the availability and usability of the netcdf libraries
distclean.cmake :	Incorporates the "distclean" rule into the cmake generated makefiles

macros.cmake :	Macros to aid the compile/link processes
options.cmake :	Definitions of what libraries and executables to build (supported static and shared libraries for *PaHM*)
pahm.cmake :	Definitions to build *PaHM*, libpahm.a and/or libpahm.so; lists all required sources for each build
Modules :	In this folder are contained the utility and "find" modules for CMake (e.g., FindNetCDF.cmake) required to set depended variable for the configuration and compilation statges

(b) Compiler group : The modules in the compiler group define the compiler flags for the three compiler families supported by *PaHM* (GNU, Intel and PGI). Supplying the compiler option to the "build.sh", as discussed discussed in section [The Build System], the script loads the proper compiler module: "configure_COMPILER.cmake", where: COMPILER = [one of gnu, intel, pgi] In the right panel of Figure [3], are shown the contents of the module file for "intel" that is, the definitions of the flags passed to the compiler during the compilation and link stages. The user may modify the file directly to fit particular project's needs or, to pass the relevant compiler flags via the "build.sh", for example:

```
./build.sh --compiler intel
           --cmake_flags "-DCMAKE_Fortran_FLAGS=\"-g -O2\" -DDEBUG=TRUE"
```



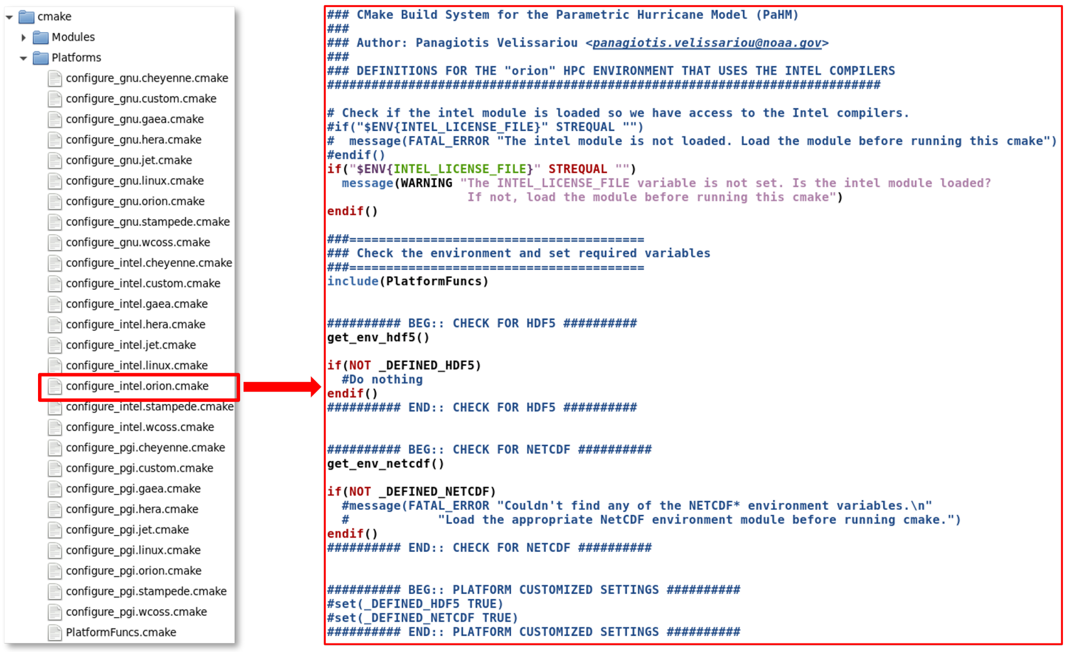
```
############################################################################
### CMake Build System for the Parametric Hurricane Model (PaHM)
### configure_intel.cmake :: Global options and definitions for the "INTEL" compilers
###
### Author: Panagiotis Velissariou <panagiotis.velissariou@noaa.gov>
############################################################################
# Include the option "-heap-arrays 0" or set the stack size to unlimited
#                    "ulimit -s unlimited" in your bash login

set(CMAKE_Fortran_FLAGS "${CMAKE_Fortran_FLAGS} -extend-source 132 -heap-arrays 0
                                                -fno-alias -sox -qno-opt-dynamic-align")
set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS} -sox")

if(DEBUG)
  set(CMAKE_Fortran_FLAGS "${CMAKE_Fortran_FLAGS} -g -O0 -check bounds -check noarg_temp_created
                                                  -check nopointer -warn -warn noerrors")
  set(CMAKE_Fortran_FLAGS "${CMAKE_Fortran_FLAGS} -fp-stack-check -fstack-protector-all
                                                  -fpe0 -debug -traceback -ftrapuv")
  set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS} -O0 -g -ftrapuv -traceback")
else()
  set(CMAKE_Fortran_FLAGS "${CMAKE_Fortran_FLAGS} -O2 -debug minimal -nowarn -fp-model source")
  set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS} -O2 -debug minimal -fp-model source")
endif()

if(SYS_BUILD_64BIT)
  set(FORTRAN_FLAGS_R8 "-i4 -real-size 64")
else()
  set(FORTRAN_FLAGS_R8 "-i4 -real-size 32")
endif()

if(SYS_ARCH EQUAL 64)
  set(CMAKE_Fortran_FLAGS "${CMAKE_Fortran_FLAGS} -assume byterecl -mcmodel=medium")
  set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS}")
else()
  set(CMAKE_Fortran_FLAGS "${CMAKE_Fortran_FLAGS} -assume byterecl")
  set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS}")
endif()
```

Figure 3: CMake modules for the *PaHM* supported compilers.

(c) Platform group : The folder "PaHM/cmake/Platforms" contains the CMake modules for the *PaHM* supported computing platforms (HPC clusters or local desktops). The list of predefined platform modules are shown in Figure [4] (left panel) where in the right panel are shown the contents of the platform file for "orion" using the "intel" compiler. This file is loaded by CMake when the build script is run as:

```
./build.sh --compiler intel --platform orion ... other options
```

When any of these modules is loaded, it sets the environment variables for the loaded libraries (currently NetCDF and HDF5) in an HPC cluster or a local desktop. Computing system configurations change from time to time and the user might need to directly modify the module file to reflect the OS changes or, to use the "custom" platform (e.g., configure↩ _intel.custom.cmake) and incorporate the changes there.


cmake
Modules
Platforms
configure_gnu.cheyenne.cmake
configure_gnu.custom.cmake
configure_gnu.gaea.cmake
configure_gnu.hera.cmake
configure_gnu.jet.cmake
configure_gnu.linux.cmake
configure_gnu.orion.cmake
configure_gnu.stampede.cmake
configure_gnu.wcoss.cmake
configure_intel.cheyenne.cmake
configure_intel.custom.cmake
configure_intel.gaea.cmake
configure_intel.hera.cmake
configure_intel.jet.cmake
configure_intel.linux.cmake
configure_intel.orion.cmake
configure_intel.stampede.cmake
configure_intel.wcoss.cmake
configure_pgi.cheyenne.cmake
configure_pgi.custom.cmake
configure_pgi.gaea.cmake
configure_pgi.hera.cmake
configure_pgi.jet.cmake
configure_pgi.linux.cmake
configure_pgi.orion.cmake
configure_pgi.stampede.cmake
configure_pgi.wcoss.cmake
PlatformFuncs.cmake

```
### CMake Build System for the Parametric Hurricane Model (PaHM)
###
### Author: Panagiotis Velissariou <panagiotis.velissariou@noaa.gov>
###
### DEFINITIONS FOR THE "orion" HPC ENVIRONMENT THAT USES THE INTEL COMPILERS
###########################################################################

# Check if the intel module is loaded so we have access to the Intel compilers.
#if("$ENV{INTEL_LICENSE_FILE}" STREQUAL "")
#  message(FATAL_ERROR "The intel module is not loaded. Load the module before running this cmake")
#endif()
if("$ENV{INTEL_LICENSE_FILE}" STREQUAL "")
  message(WARNING "The INTEL_LICENSE_FILE variable is not set. Is the intel module loaded?
                   If not, load the module before running this cmake")
endif()

###========================================
### Check the environment and set required variables
###========================================
include(PlatformFuncs)


########## BEG:: CHECK FOR HDF5 ##########
get_env_hdf5()

if(NOT _DEFINED_HDF5)
  #Do nothing
endif()
########## END:: CHECK FOR HDF5 ##########


########## BEG:: CHECK FOR NETCDF ##########
get_env_netcdf()

if(NOT _DEFINED_NETCDF)
  #message(FATAL_ERROR "Couldn't find any of the NETCDF* environment variables.\n"
  #             "Load the appropriate NetCDF environment module before running cmake.")
endif()
########## END:: CHECK FOR NETCDF ##########


########## BEG:: PLATFORM CUSTOMIZED SETTINGS ##########
#set(_DEFINED_HDF5 TRUE)
#set(_DEFINED_NETCDF TRUE)
########## END:: PLATFORM CUSTOMIZED SETTINGS ##########
```

Figure 4: CMake modules for the *PaHM* supported computing platforms.

Additional platform modules can be added with ease, by copying one of the existing module files in "PaHM/cmake/Platforms" to say: PaHM/cmake/Platforms/configure_intel.my↩ _own.cmake. This new module file should first be modified to reflect the settings of the host platform and then run the build script as:

```
./build.sh --compiler intel --platform my_own ... other options
```

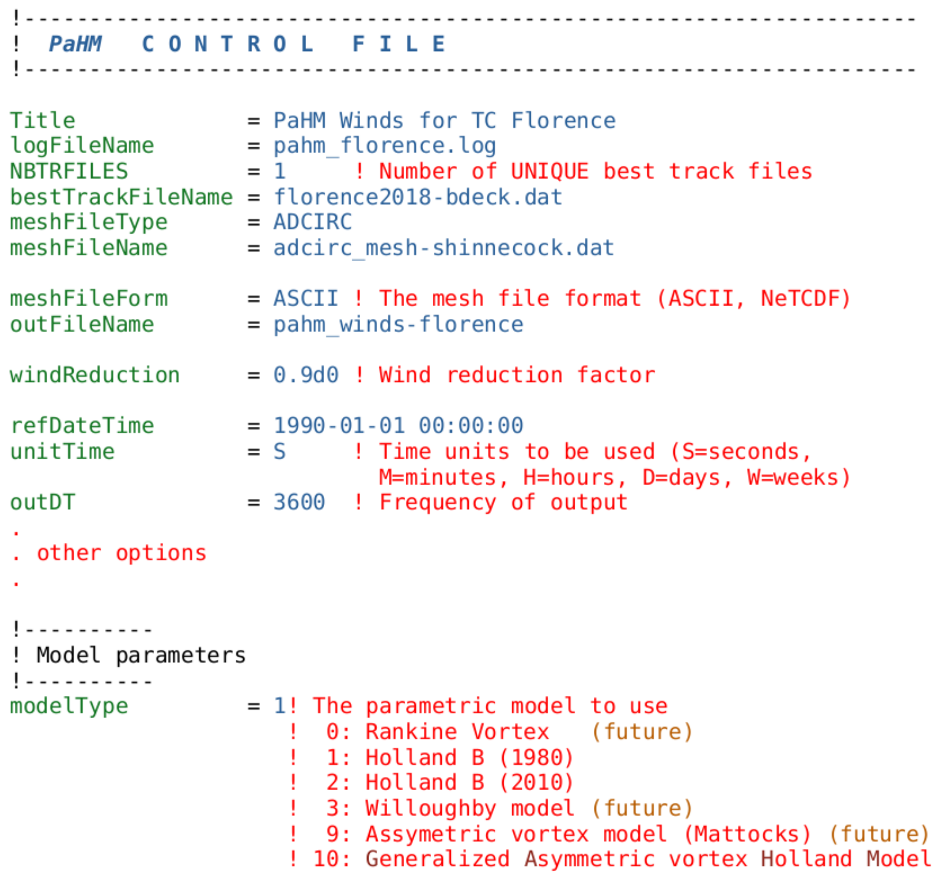
to use the newly developed platform module file.

2.3 Using *PaHM*

Before you can use *PaHM*, you must have a working *PaHM* installation as described in section [The Build System]. Assuming that *PaHM* is installed in the "PaHM/bin" folder (default), the model may be run as: `bin/pahm --help` to print the list of the available command line arguments to the model. By default, running *PaHM* without arguments the model tries to find its default control file (pahm_control.in) located in the current working directory. The user may supply another control file by running *PaHM* as:

```
pahm LOCATION_OF_CONTROL_FILE/custom_control_file.in
```

A typical control file looks like the one shown in Figure [5]. It is the user's responsibility to create the proper control file to be used by *PaHM* for the particular application. The easiest way to create the control file is to copy one from the "PaHM/templates" folder and modify it accordingly.



```
!---------------------------------------------------------------------
!  PaHM   C O N T R O L   F I L E
!---------------------------------------------------------------------

Title             = PaHM Winds for TC Florence
logFileName       = pahm_florence.log
NBTRFILES         = 1     ! Number of UNIQUE best track files
bestTrackFileName = florence2018-bdeck.dat
meshFileType      = ADCIRC
meshFileName      = adcirc_mesh-shinnecock.dat

meshFileForm      = ASCII ! The mesh file format (ASCII, NeTCDF)
outFileName       = pahm_winds-florence

windReduction     = 0.9d0 ! Wind reduction factor

refDateTime       = 1990-01-01 00:00:00
unitTime          = S     ! Time units to be used (S=seconds,
                            M=minutes, H=hours, D=days, W=weeks)
outDT             = 3600  ! Frequency of output
.
. other options
.

!----------
! Model parameters
!----------
modelType         = 1! The parametric model to use
                     !  0: Rankine Vortex   (future)
                     !  1: Holland B (1980)
                     !  2: Holland B (2010)
                     !  3: Willoughby model (future)
                     !  9: Assymetric vortex model (Mattocks) (future)
                     ! 10: Generalized Asymmetric vortex Holland Model
```

Figure 5: Sample *PaHM* input configuration (control) file.

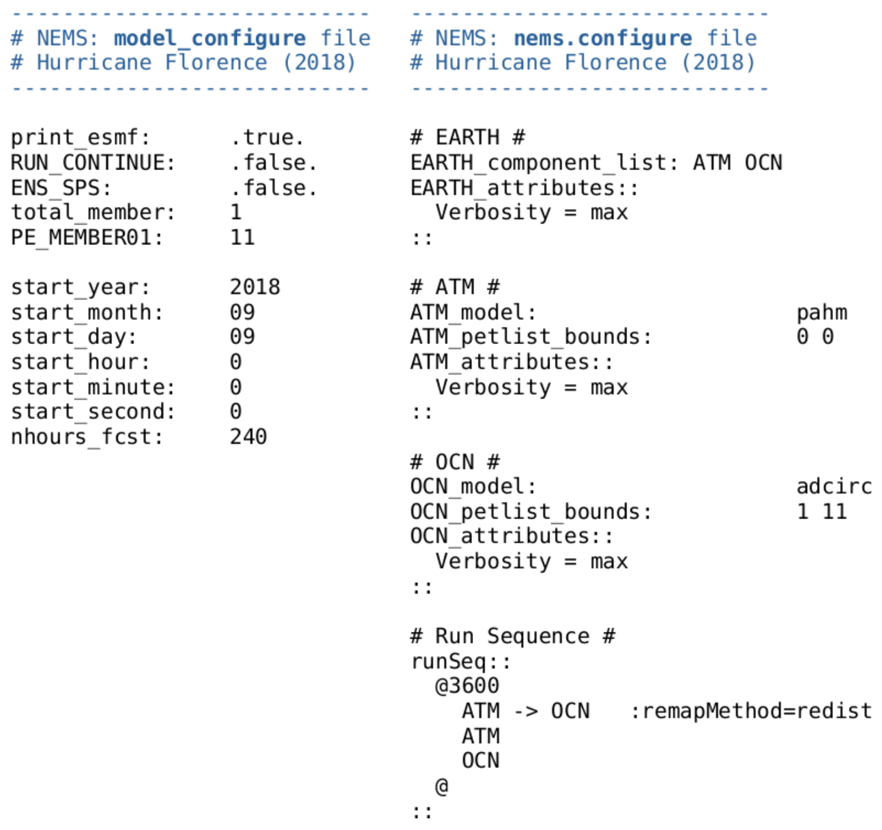
The control file is required either running *PaHM* as a standalone model or as a coupled modeling component.

2.3.1 Coupled Model Configuration

PaHM is coupled with other models via its own NUOPC Cap (section [Coupling Environment]). To build the coupled *PaHM* / ADCIRC "NEMS.x" executable the "build.sh" script of CoastalApp (see CoastalApp's README.md) is run as:

```
./build.sh --compiler intel --platform orion --component "PAHM ADCIRC"
```

"NEMS.x" uses the two configuration files shown in Figure [6] to define the coupled simulation parameters. The "model_configure" file defines the simulation period for the model while, in the file "nems.configure" are defined the coupling parameters of the simulation (e.g., number of cores, the data exchange sequence, rtc).



```
----------------------------
# NEMS: model_configure file
# Hurricane Florence (2018)
----------------------------

print_esmf:      .true.
RUN_CONTINUE:    .false.
ENS_SPS:         .false.
total_member:    1
PE_MEMBER01:     11

start_year:      2018
start_month:     09
start_day:       09
start_hour:      0
start_minute:    0
start_second:    0
nhours_fcst:     240
```

```
----------------------------
# NEMS: nems.configure file
# Hurricane Florence (2018)
----------------------------

# EARTH #
EARTH_component_list: ATM OCN
EARTH_attributes::
  Verbosity = max
::

# ATM #
ATM_model:                    pahm
ATM_petlist_bounds:           0 0
ATM_attributes::
  Verbosity = max
::

# OCN #
OCN_model:                    adcirc
OCN_petlist_bounds:           1 11
OCN_attributes::
  Verbosity = max
::

# Run Sequence #
runSeq::
  @3600
    ATM -> OCN   :remapMethod=redist
    ATM
    OCN
  @
::
```

Figure 6: Sample configuration files of a NEMS/NUOPC coupled *PaHM* / ADCIRC simulation for Hurricane Florence (2018).

When "NEMS.x" is run, it is searching for its two configuration files (should be located in the same directory as the NEMS.x executable) and the configuration and the required input data of its modeling component mentioned in "nems.configure" (right panel of Figure [6]). It is the user's responsibility to create the data for all the coupled modeling components and copy them into the same directory where "NEMS.x" is located. To this end, *PaHM*'s control file (Figure [5]) should also be located in the "NEMS.x" directory.

Chapter 3

Parametric Models in *PaHM*

Numerical modeling of hurricane wind fields has long been used in severe wind impact studies, hurricane-induced storm surge and flooding predictions, and risk assessment studies. Storm surges are primarily induced by the surface wind stresses and secondarily by atmospheric pressure perturbations over shallow water in coastal areas. The accuracy of the storm surge predictions depends on how accurate are the atmospheric predictions that force the ocean and wave models. The use of full physics and high resolution mesoscale/regional atmospheric models might produce more accurate wind predictions but require extensive computing resources and simulation times to produce tropical cyclone (TC) forcing for storm surge forecasting and other hurricane related hazard studies. This is the exact reason that simple parametric TC models are widely used to generate the hurricane wind fields and to provide atmospheric forcing for storm surge and inundation forecasting. The advances made over the past several decades on improved hurricane track and intensity forecasts allow the parametric TC models to produce more accurate forecast in the region of the storm's path. The accuracy though of the parametric TC forecasts depends not only on the track and intensity, but also on the distribution of the wind field.

The development of *PAHM* follows the reasoning outlined above, and the philosophy of producing "accurate" wind fields quickly and effectively. *PAHM* contains various lightweight parametric tropical cyclone (TC) models that require minimal computational resources to produce the wind fields on the fly and fast. Such models use limited physics in producing "accurate" wind fields in the vicinity of the storm's path. The wind fields generated by these models are computed at the gradient level (Figure [7]). The gradient level is roughly $300m \sim 3km$ above the surface of the earth (atop the atmospheric boundary layer, ABL), and is the level most representative of the air flow in the lower atmosphere immediately above the layer affected by surface friction. This level is free of local wind and topographic effects (such as sea breezes, downslope winds etc).

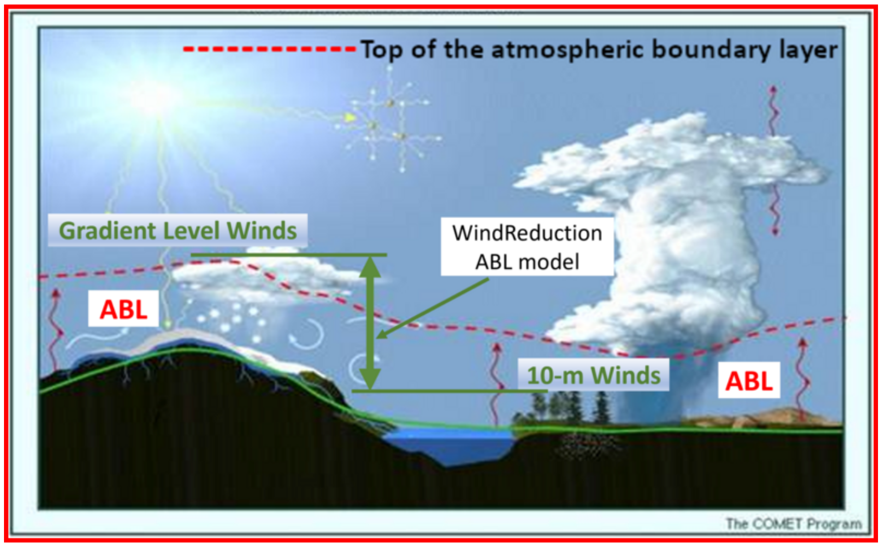


Figure 7: Schematic representation of the atmospheric boundary layer (ABL), its thickness variation over time and the definition of the gradient level. The conversion between the gradient level and the 10-m winds is denoted by the green arrow. This conversion can be achieved using the wind reduction factor (w_{rf}) or, an ABL model formulation.

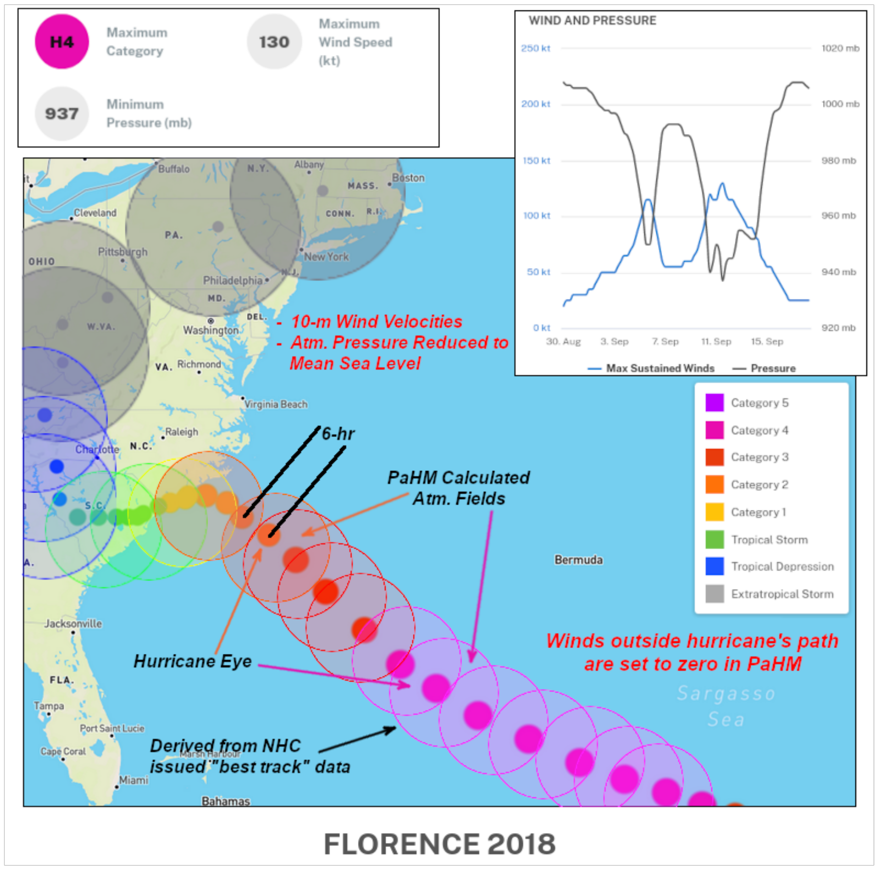


Figure 8: Calculation procedure of *PaHM* wind fields for Hurricane Florence (2018). Source of background images and track data: https://www.coast.noaa.gov/hurricanes/.

The 10-m wind may be estimated by decreasing the gradient level wind speed by approximately 10-20% over the ocean and up to 40% over land (wind reduction factor w_{rf} of 0.9-0.8 to 0.6 respectively). The use of a fixed and empiracally determined w_{rf} to convert the gradient winds to 10-m winds is a limitation of these models as w_{rf} is a function of location, time (ABL thickness is a function of time) and of course the characteristics of the particular TC.

3.1 Holland 1980

The traditional Holland Model (Holland 1980), thereafter HM80, is one of most widely used tropical cyclone (TC) analytical vortex models to generate meteorological forcings near the path of the storm. HM80 solves the reduced physics gradient wind equation 3.1. Gradient winds are those theoretical winds that blow parallel to curved isobar lines or height-contour lines in the absence of turbulent drag (gradient level winds). The gradient level is roughly 1 km above the surface of the earth, and is the level most representative of the air flow in the lower atmosphere immediately above the layer affected by surface friction. This level is free of local wind and topographic effects (such as sea breezes, downslope winds etc) and it is commonly defined atop the atmospheric boundary layer (ABL).

For the gradient winds there is no balance between the Coriolis force and the Pressure gradient force (as this is the case for geostrophic winds), resulting in a non-zero net force F_{net} known as the Centripetal force. This F_{net} is what causes the wind to continually change direction as it goes around a circle (Stull 2017) as shown in Figures [9] and [10].

By describing this change in direction as causing an apparent force (Centrifugal), we can find the equation for the gradient wind. Equation 3.1 defines the steady-state gradient wind and represents the balance of the Pressure Gradient Force (F_{PG}), Centrifugal $(F_{CN} = -F_{net})$ and Coriolis (F_{CF}) forces at the gradient level (Figures [9] and [10]):

$$\underbrace{V_g^2(r)}_{F_{CN}\,\text{term}} + \underbrace{frV_g(r)}_{F_{CF}\,\text{term}} - \underbrace{\frac{r}{\rho_{air}}\frac{\partial P(r)}{\partial r}}_{F_{PG}\,\text{term}} = 0 \tag{3.1}$$

where: $V_g(r)$ is the gradient level wind speed at radius r, $P(r)$ is the pressure at radius r, r is the radial distance, $f = 2\Omega \sin\phi$ is the Coriolis parameter, $\Omega = 7.27221 \cdot 10^{-5} s^{-1}$ is the rotational speed of the earth, ϕ is the latitude in radians and ρ_{air} is the air density (assumed constant: $1.15\,kg/m^3$). The quadratic equation 3.1 has two roots:

$$V_g(r) = \sqrt{\frac{r}{\rho_{air}}\frac{\partial P(r)}{\partial r} + (\frac{rf}{2})^2} - \frac{rf}{2} \tag{3.2}$$

$$V_g(r) = -\sqrt{\frac{r}{\rho_{air}}\frac{\partial P(r)}{\partial r} + (\frac{rf}{2})^2} - \frac{rf}{2} \tag{3.3}$$

where equation 3.2 is the solution for $V_g(r)$ for flow around a cyclone that is, around a low pressure (Figure [9]) while, equation 3.3 represents the solution for $V_g(r)$ for flow around an anticyclone that is, around a high pressure (Figure [10]).

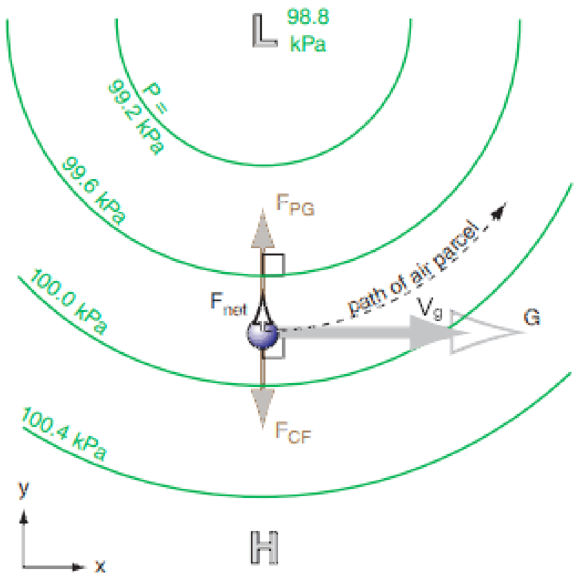


Figure 9: Forces that cause the gradient wind to be faster than geostrophic for an air parcel circling around a low-pressure center (a **cyclone** in N. Hemisphere). The centrifugal force pulls the air parcel inward to force the wind direction to change as needed for the wind to turn along a circular path. Source: Practical Meteorology: An Algebra-based Survey of Atmospheric Science. Roland Stull, The University of British Columbia, Vancouver, Canada.

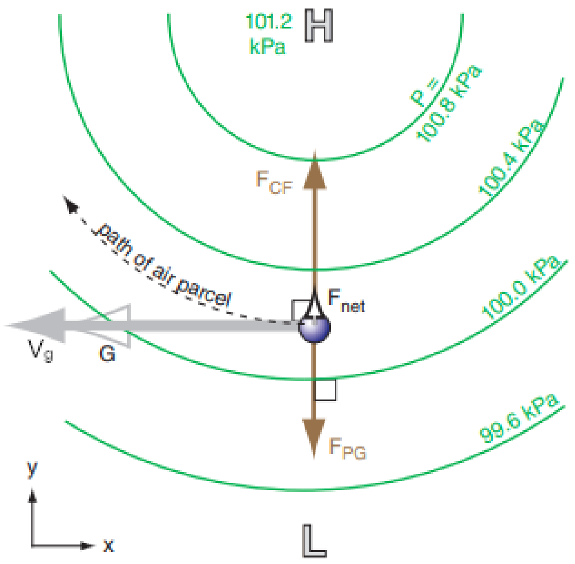


Figure 10: Forces that cause the gradient wind to be faster than geostrophic for an air parcel circling around a high-pressure center (an **anticyclone** in the N. Hemisphere). The centrifugal force pulls the air parcel inward to force the wind direction to change as needed for the wind to turn along a circular path. Source: Practical Meteorology: An Algebra-based Survey of Atmospheric Science. Roland Stull, The University of British Columbia, Vancouver, Canada.

To derive the analytical expression for $V_g(r)$, HM80 assumed a surface pressure profile that is approximated by the following hyperbolic equation:

$$P(r) = P_c + (P_n - P_c)e^{-A/r^B} \tag{3.4}$$

where: A, B are scaling parameters, $P(r)$ is the pressure at radius r, P_n is the ambient pressure (assumed constant: $1013.25\,mbar$) and P_c is the central pressure. Substituting the expression for $P(r)$ into equation 3.2 and, the following expression for the wind speed at the gradient level is obtained:

$$V_g(r) = \sqrt{AB(P_n - P_c)e^{-A/r^B}/\rho_{air}r^B + (\frac{rf}{2})^2} - \frac{rf}{2} \tag{3.5}$$

To determine the scaling parameters A and B, it is assumed that at the region of the radius of maximum winds (RMW) that is, at the region of the sustained maximum wind speeds ($r = R_{max} =$ RMW), the wind speed V_g satisfies the first of equations 3.6. It is also assumed that the Rossby number R_o is very large (the third of equations 3.6), so that the Coriolis forces can be neglected and therefore the air is in cyclostrophic balance (Holland 1980) as described by equation 3.7 for the cyclostrophic wind speed $V_c(r)$.

$$V_g = V_{max}\,; \quad \frac{dV_g}{dr} = 0\,; \quad R_o = \frac{V_{max}}{fR_{max}} \gg 1 \tag{3.6}$$

$$V_c(r) = V_g(r)\Big|_{r \to R_{max}} = \sqrt{AB(P_n - P_c)e^{-A/r^B}/\rho_{air}r^B} \tag{3.7}$$

Applying the second of the conditions in equations 3.6 on equation 3.7, we find that the radius of maximum winds is independent of the relative values of the ambient and the central pressure and it is defined only by the scaling parameters A and B as: $R_{max} = A^{1/B}$, or $A = R_{max}^B$. Substituting the expression for A into equation 3.7 and setting $V_c(R_{max}) = V_{max}$, the expression for the scaling parameter B (widely known as the Holland B parameter) is readily determined:

$$A = R_{max}^B\,; \quad B = \frac{\rho_{air} e V_{max}^2}{P_n - P_c} \tag{3.8}$$

Physically, the Holland parameter B defines the shape of the pressure profile (equation 3.4) while, the parameter A determines its location relative to the origin (Holland 1980). HM80 states that plausible ranges of B would be between 1 and 2.5 to to limit the shape and size of the vortex. Based on equations 3.4, 3.5 and 3.8, and re-organizing the pressure and the gradient wind speed equations, the final equations of the HM80 parametric TC model that *PAHM* solves are summarized as follows:

$$B = \frac{\rho_{air} e V_{max}^2}{P_n - P_c} \tag{3.9}$$

$$P(r) = P_c + (P_n - P_c) \cdot e^{-(R_{max}/r)^B} \tag{3.10}$$

$$V_g(r) = \sqrt{V_{max}^2 \cdot (\frac{R_{max}}{r})^B \cdot e^{1-(R_{max}/r)^B} + (\frac{rf}{2})^2} - \frac{rf}{2} \tag{3.11}$$

As reasoned in Gao 2018, a large R_o (10^3) describes a system in cyclostrophic balance dominated by inertial and centrifugal forces with negligible Coriolis force (e.g., a tornado or the inner core of an intense hurricane), while a small value R_o $(10^{-2} \sim 10^2)$ describes a system in geostrophic balance strongly influenced by the Coriolis force (e.g., the outer region of a TC). Therefore, the cyclostrophic balance assumption made in HM80 is only valid for describing an intense but narrow TC with a large R_o, and not suitable for weak but broad tropical cyclones with small R_o values. Although this is a limitation of the HM80 model, equations 3.9 through 3.11, are widely used in hurricane risk studies and storm surge studies due to their simplicitly and their capability to generate the atmospheric fields quickly and efficiently.

3.2 Generalized Asymmetric Vortex Holland model (*GAHM*)

The Generalized Asymmetric Vortex Holland model (Gao et al. 2015 and Gao 2018) extends HM80 by eliminating the cyclostrophic assumption at the the region of RMW (the third of equations 3.6) to allow the generation of representative wind fields for a wider range of TCs. *GAHM* also introduces a composite wind methodology to fully use all multiple storm isotachs in TC forecast or best track data files to account for asymmetric tropical cyclones such as a land-falling hurricane.

GAHM model solves the gradient wind equation for V_g (equation 3.2) by eliminating the influence of the Rossby number (R_o)) on the gradient wind solution assuming that:

$$V_g = V_{max}\,; \qquad \frac{dV_g}{dr} = 0 \tag{3.12}$$

The pressure profile used is the same as in HM80 (equation 3.4) where the scaling parameter A is slightly re-defined by introducing a new scaling factor (ϕ) : $\phi = A/R_{max}^B$, or $A = \phi R_{max}^B$. Substituting the expression for A into equation 3.5 and using the second of equations 3.12, an adjusted Holland B parameter (B_g), is derived as:

$$B_g = \frac{(V_{max}^2 + fV_{max}R_{max})\rho_{air}e^{\phi}}{\phi(P_n - P_c)} = B\frac{(1+1/R_o)e^{\phi-1}}{\phi}\,; \qquad B = \frac{\rho_{air}eV_{max}^2}{P_n - P_c} \tag{3.13}$$

Substituting the expressions for A and B (replaced by B_g) into equation 3.5 and using the first of equations 3.12, the final expression for the scaling parameter ϕ is derived:

$$\phi = \frac{1 + fV_{max}R_{max}}{B_g(V_{max}^2 + fV_{max}R_{max})} = 1 + \frac{1/R_o}{B_g(1+1/R_o)} \tag{3.14}$$

Based on equations 3.4, 3.5, 3.13, and 3.14 and re-organizing the pressure and the gradient wind speed equations, the final equations of the *GAHM* parametric TC model that *PAHM* solves are summarized as follows:

$$B_g = B\big(1 + \frac{1}{R_o}\big)^{e^{-\phi}/\phi}; \quad \phi = 1 + \frac{1/R_o}{B_g(1+1/R_o)}; \quad B = \frac{\rho_{air} e V_{max}^2}{P_n - P_c}; \quad \text{and} \quad R_o = \frac{V_{max}}{f R_{max}} \tag{3.15}$$

$$P(r) = P_c + (P_n - P_c) \cdot e^{-\phi (R_{max}/r)^{B_g}} \tag{3.16}$$

$$V_g(r) = \sqrt{V_{max}^2 \cdot \big(\frac{R_{max}}{r}\big)^{B_g} \cdot (1 + 1/R_o) \cdot e^{\phi(1-(R_{max}/r)^{B_g})} + \big(\frac{rf}{2}\big)^2} - \frac{rf}{2} \tag{3.17}$$

Given the values for V_{max}, R_{max}, P_n and P_c, the iterative solution of the first two of equations 3.15 produces the final values of B_g and ϕ.

Chapter 4

PaHM Features and Capabilities

In this section of the documentation are introduced the features of *PaHM* currently implemented in the system while, in section [Conclusions and Future Considerations] are discussed future features and capabilities of *PaHM*.

4.1 Modeling Multiple Interacting Storms

The presence of multiple TC storms in an oceanic basin is not a rare phenomenon as one could expect but it happens frequently. Some of the most recent examples of such storms are: (a) Hurricane Laura and Tropical Storm Marco (Gulf of Mexico, 2020), (b) Hurricane Irwin and Hurricane Hilary (East Pacific basin, 2017), shown in Figure [11], (c) Hurricane Madeline and Hurricane Lester (Central Pacific basin, 2016), (d) Hurricane Iselle and Hurricane Julio (Eastern Pacific basin, 2014) and (e) Tropical Storm Bonnie and Hurricane Charley (Gulf of Mexico, 2004). During the active TC periods, $\sim 10\% - 57\%$ of TCs form in the presence of at least one other pre-existing TC in the same basin (Schenkel 2017). It is therefore important to account for the effects of multiple storms in the same basin and their interaction.

As shown in Figure [11], both the two storms modulate the wind field therefore, this effect needs to be taken into consideration when generating the wind fields. *PaHM* uses a simple inverse distance weighted interpolation (IDW) to determine the final wind field in the area affected by both storms. The weights in the IDW interpolation are calculated in respect of the location of the "eye" of the storms.

When the cyclones are close to each other, there is the possibility that their centers will circle each other cyclonically about a point between the two systems due to their cyclonic wind circulations. The two vortices will be attracted to each other, and eventually spiral into the center point and merge. When the two vortices are of unequal size, the larger vortex will tend to dominate the interaction, and the smaller vortex will circle around it. This effect is known as the Fujiwhara effect. *PaHM* is not designed to deal with this type of situations and assumes that the wind fields are just modulated without considering the detailed physical interactions between the storms.

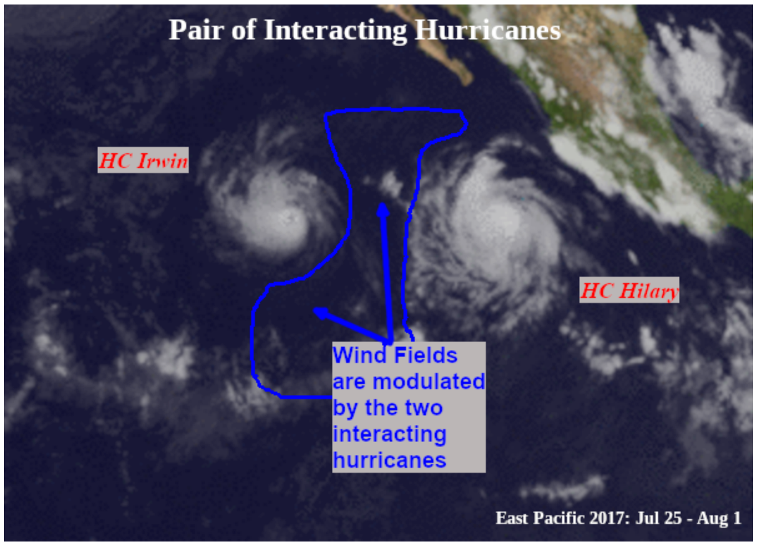
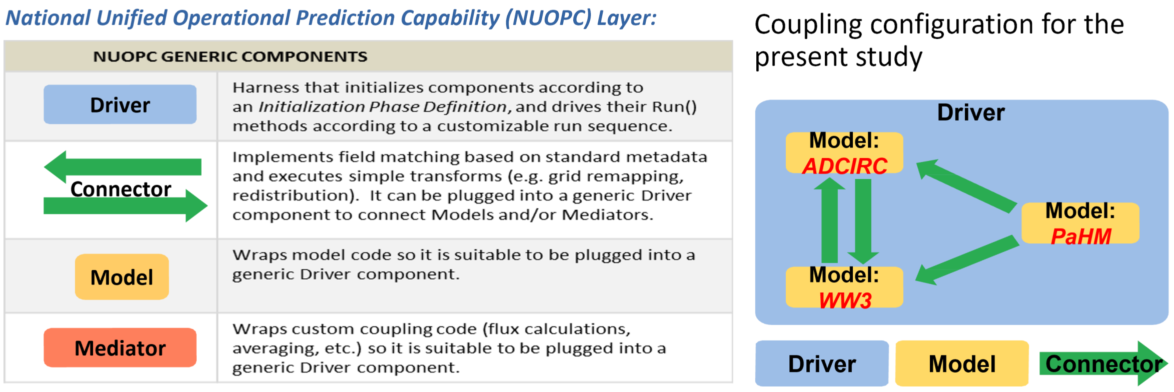

Pair of Interacting Hurricanes
HC Irwin
HC Hilary
Wind Fields are modulated by the two interacting hurricanes
East Pacific 2017: Jul 25 - Aug 1

Figure 11: CO-OPS wind and water level observation locations.

4.2 Coupling Environment

PaHM for its coupling configuration uses the National Unified Operational Prediction Capability (NUOPC) layer framework to exchange the generated wind fields with the other models via its own NUOPC Cap. The NUOPC Cap is a Fortran module that is used to interface to a model in a NUOPC based coupled system that comprises from NUOPC subroutines that are called during the three model phases: **Init Phase** (model initialization), **Run Phase** (model run) and **Finalize Phase** (model finalize part). Each model that uses the NUOPC layer should have these three subroutines available in their codebase.

NUOPC is a software layer built on top of the Earth System Modeling Framework (ESMF). ESMF is a modeling framework that provides data structures, interfaces, and operations for building coupled models from a set of components. NUOPC refines the capabilities of ESMF by providing a more precise definition of what is a model component and how components should interact and share data in a coupled system. Furthermore, the NUOPC software layer is designed to work with High Performance Computing (HPC) models written in Fortran (as is *PaHM*) and are based on a distributed memory model of parallelism (MPI) The NUOPC Cap light-weight software layer on top of the model code, making calls into it and exposing model data structures in a standard way. Figure [12] shows all available generic components defined in the NUOPC layer (left panel) while, on the right panel are shown the components currently available in *PaHM* and the coupling configuration of *PaHM* with ADCIRC and WaveWatch III (not used here).



National Unified Operational Prediction Capability (NUOPC) Layer:

NUOPC GENERIC COMPONENTS	
Driver	Harness that initializes components according to an *Initialization Phase Definition*, and drives their Run() methods according to a customizable run sequence.
Connector	Implements field matching based on standard metadata and executes simple transforms (e.g. grid remapping, redistribution). It can be plugged into a generic Driver component to connect Models and/or Mediators.
Model	Wraps model code so it is suitable to be plugged into a generic Driver component.
Mediator	Wraps custom coupling code (flux calculations, averaging, etc.) so it is suitable to be plugged into a generic Driver component.

Figure 12: NUOPC Cap definition and coupling configuration for the Hurrcane Florence case study.

The *PaHM* NUOPC Cap during its "Init Phase" creates the domain for the data exchange (as an ESMF_Mesh structure), defines and advertises the fields to be exchanged and initializes PaHM by calling its "Init" subroutine (Figure [13]). During the "Run Phase", the NUOPC Cap advances *PaHM's* computations in time by calling its "Run" subroutine (Figure [13]). The defined fields are exported in this stage at the specified coupling intervals (defined in nems.configure file) while, the simulation period of the coupled system is pre-defined in the model_configure file (see section [[intro_conf_coupled]]). During the "Finalize Phase", the NUOPC Cap calls *PaHM's* "Finalize" subroutine (Figure [13]) to free still allocated memory, close any open files and the log system, and finalize the creation of the output NetCDF file that contains all the computed wind data.

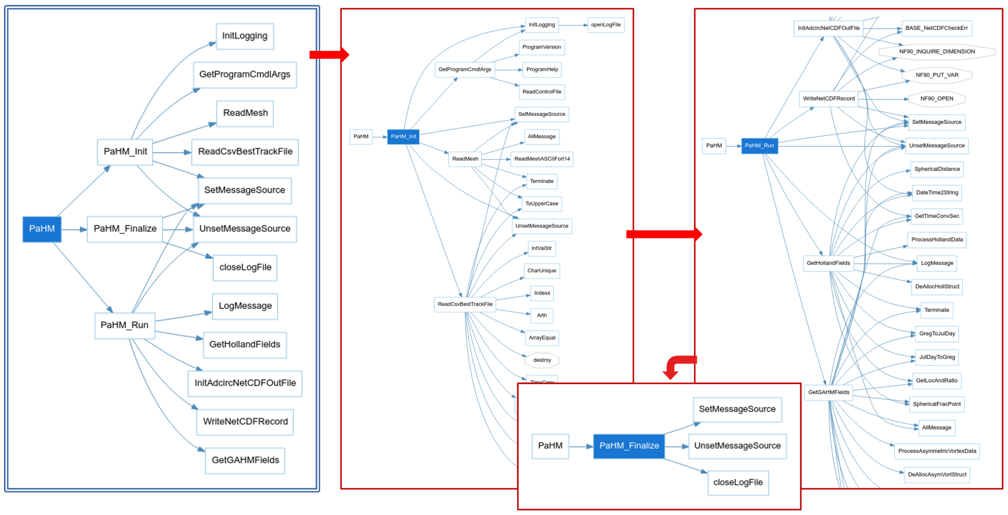


Figure 13: *PaHM* code flowchart showing the three stages of the NUOPC coupling interface.

Chapter 5

Model Evaluation

In this section the developed modeling system and its associated modeling components are evaluated under realistic atmospheric and ocean conditions, varied flow domains and coupling configurations so the performance of PaHM can be analyzed. Although the individual modeling components of PaHM have been evaluated, the overall model performance still needs evaluation and verification. Switching on and off the different modeling components shows the relative performance of each modeling component with respect to each other and to the problem being investigated. PaHM's outputs are: a 10-m wind speed, b wind direction and c atmospheric pressure reduced to mean sea level MSLP.

5.1 Statistical Performance Measures

Only parametric statistical tests are used in the performance evaluation of the developed model that include (*a*) the mean m of the differences between the calculated and the measured or observed data sets, (*b*) the standard deviation SD, (*c*) the root mean square differense RMSE, (*d*) the coefficient of determination (R^2), (*e*) the bias bias, (*f*) the scatter coefficient SI, (*g*) the Willmott (2012) skill WS12 and (*h*) the Nash and Sutcliff (1970) skill NS.

a) Mean: The mean of the differences between the modeled and measured data provides a gross overall measure of the model performance and is calculated as:

$$m = \frac{\sum_{i=1}^{n} \left(M_i - O_i\right)}{n} \tag{5.1}$$

where n is the total number of observation or modeled points, M_i are the modeled and O_i are the observed values of each evaluated variable. The smaller the mean difference the better the agreement between the model and the observed values, with a value of zero denoting absolute agreement.

b) Standard Deviation: The standard deviation SD is a measure of the distance of the difference between the calculated and observed data from the mean difference. Small standard deviations indicate that the differences are closer to the mean. The standard deviation is calculated as:

$$SD = \sqrt{\frac{\sum_{i=1}^{n} \left[\left(M_i - O_i\right) - m\right]^2}{n}} \tag{5.2}$$

c) Root Mean Square Differense: The root mean square difference RMSE is another test of the overall model performance that measures how close the modeled value of a variable is to the observed value. Mathematically, the test is defined as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} \left(M_i - O_i\right)^2}{n}} \tag{5.3}$$

The differences between the modeled and observed data are squared so that more weight is given to larger errors.

d) Coefficient of Determination: The coefficient of determination R^2, where R is the correlation coefficient, indicates the proportion of the variance in the dependent variable that is predicted by linear regression and the independent variable. In general, a high R^2 value indicates that the model is a good fit for the data. An $R^2 = 0.62$, indicates that 62% of the variation in the outcome has been explained. A value of 1 would indicate that the regression line represents all of the data (the best fit) while, a value of 0 shows no association at all. Note that the coefficient of determination shows only the magnitude of the association, not whether that association is statistically significant.

$$R = \frac{\sum_{i=1}^{n} \left(O_i - \overline{O}\right)\left(M_i - \overline{M}\right)}{\sqrt{\sum_{i=1}^{n} \left(O_i - \overline{O}\right)^2 \left(M_i - \overline{M}\right)^2}} \tag{5.4}$$

where: $\overline{M} = (1/n) \sum_{i=1}^{n} M_i$ is the mean of the modeled values and $\overline{O} = (1/n) \sum_{i=1}^{n} O_i$ is the mean of the observation values.

e) Model Bias: Bias is the tendency of a statistical estimator to overestimate or underestimate a parameter. The bias of a statistical estimator is the difference between the expected value of the statistic and the true value of the sample (population) parameter. If the bias is close to zero then the statistical estimator is an unbiased estimator, otherwise it is considered a biased estimator. The statistical estimator used here is the sample or population mean.

$$bias = \overline{O} - \overline{M} \tag{5.5}$$

f) Scatter Index: The scatter index SI is calculated by dividing the RMSE with the mean of the observations and multiplying it by 100 (percent):

$$SI = \frac{RMSE}{\overline{O}} \tag{5.6}$$

where SI is the percentage of RMSE with respect to the mean of the observations that is, the percentage of expected error for the parameter.

g) Willmott Skill Index: The evaluation of model performance, that is the comparison model estimates with observed values, is a fundamental step for model development and use. This validation process includes criteria that rely on mathematical measurements of how well model results simulate the observed values. The parameter WS12, called index of agreement, is a relative average error and bounded measure. The best agreement between model results and observations will yield a skill of one while, a value of ≤ 0 denotes a complete disagreement. This statistic is calculated using the following equations:

$$\left.\begin{array}{l} SM1 = \sum_{i=1}^{n} |O_i - M_i| \;;\; SM2 = \sum_{i=1}^{n} \left|M_i - \overline{O}\right| \left|O_i - \overline{O}\right| \\ WS12 = 1.0 - SM1/(2.0 \cdot SM2) \quad \text{for} \quad SM1 \leq 2.0 \cdot SM2 \\ WS12 = 2.0 \cdot SM2/SM1 - 1.0 \quad \text{for} \quad SM1 > 2.0 \cdot SM2 \end{array}\right\} \tag{5.7}$$

The range of qualification for the Willmott's skill index is given in the following table:

$$\left.\begin{array}{rr} 0.8 < WS12 \leq 1.0 & \text{Excellent} \\ 0.6 < WS12 \leq 0.8 & \text{Good} \\ 0.3 < WS12 \leq 0.6 & \text{Reasonable} \\ 0.0 < WS12 \leq 0.3 & \text{Poor} \\ WS12 \leq 0.0 & \text{Bad} \end{array}\right\} \tag{5.8}$$

h) Nash and Sutcliff Skill Index: The Nash and Sutcliff skill index is similar to the Willmott's skill index and it is calculated as:

$$NS = 1.0 - \frac{\sum_{i=1}^{n} \left(M_i - O_i\right)^2}{\sum_{i=1}^{n} \left(O_i - \overline{M}\right)^2} \tag{5.9}$$

For a perfect model with an estimation error variance equal to zero, the Nash and Sutcliffe index equals 1. Values of the Nash and Sutcliffe index close to 1, suggest a model with more predictive skill. The range of qualification presented in the case of the Willmott's skill index can be used for the Nash and Sutcliff skill index case as well.

All the above tests give information on the size, but not of the nature of the error, which make them adequate measures for a preliminary model evaluation. However, a deeper analysis might require specific tests that can reveal the nature of the errors and help with future model improvements.

5.2 Hurricane Florence (2018) Case Study

PaHM was thoroughly evaluated through standalone and coupled model configurations for Hurricane Florence (2018) on the Eastern Coast of the United States. In the coupled configuration, *PaHM* was used within the CoastalApp modeling framework to supply its forcing wind fields to the ocean model ADCIRC and the wave model WaveWatch III. The standard statistical measures described in section [Statistical Performance Measures] were used to (a) validate the performance of the standalone and coupled system and (b) to identify spatial and temporal system limitations.

The simulation period for hurricane Florence was from 2018-09-07 00 UTC to 2018-09-19 00 UTC (based on the best track file for Florence from the NHC). Simulations were performed using the high resolution Surge and Tide Operational Forecast System (STOFS) mesh (Figure [14]) with a fine resolution of about $120m$ near the coastal areas. The wind fields were exchanged with the ocean and the wave model via *PaHM*'s NUOPC Cap. For evaluation purposes, three series of simulations were performed, namely (a) ADCIRC tide only simulations to verify that the model performs as expected, (b) 1-way coupled PaHM + ADCIRC simulations using wind reduction factors of 0.65, 0.70, 0.75, 0.80, 0.85, 0.90 and 0.95 to determine the optimal wind reduction factor for the particular storm and the particular geographical region and (c) 1-way coupled PaHM + ADCIRC simulation using the results from (a) and (b).

The STOFS triangular mesh contains 9,997,402 elements and 5,131,901 nodes with 35 boundary segments defined. The topo-bathymetry data are composed from a variety of data sources and they are referenced to the MSL vertical datum. The resulting ADCIRC mesh/topo-bathymetry file size is about $800MB$. The boundary conditions for all the simulations are tidal.

All model results in both standalone and coupled configurations were compared against NOAA's CO-OPS wind and water level observations at selected locations as shown in Figure [15].

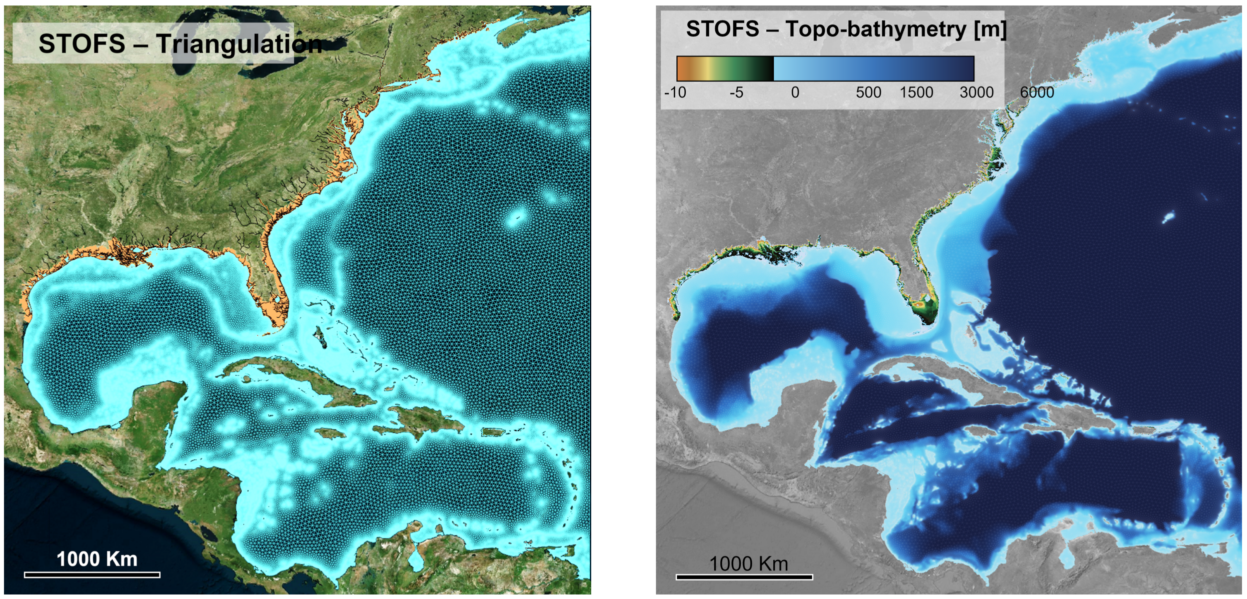
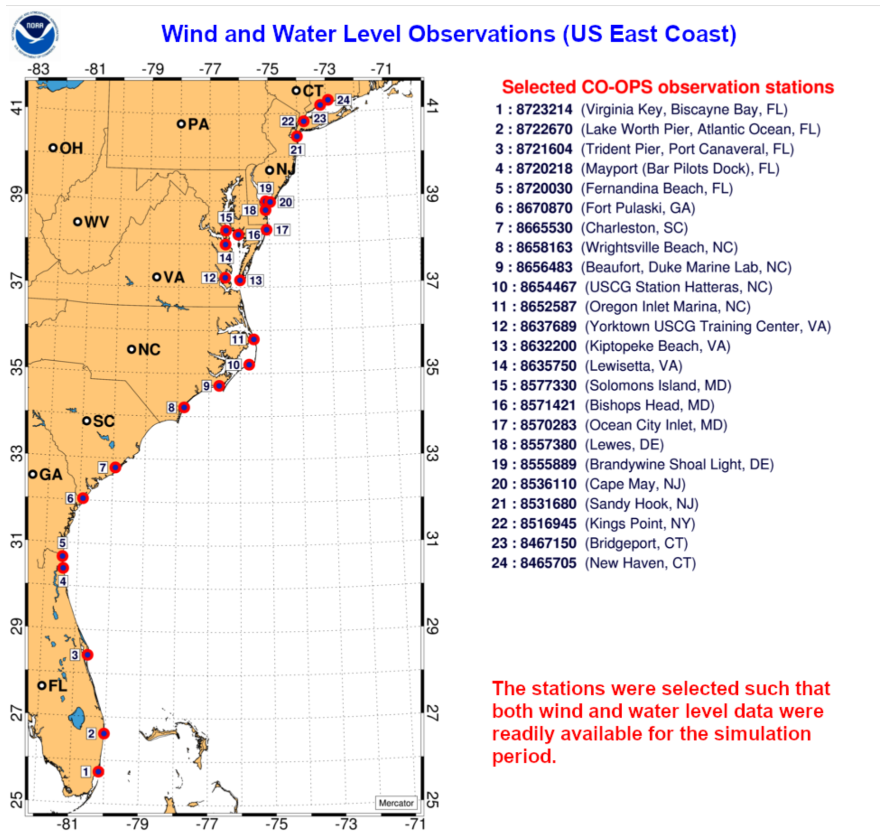


Figure 14: STOFS computational Domain. The left panel shows the triangular mesh with coarse resolution of about $3\,km$ and a fine resolution near all coastal areas of about $120\,m$. The right panel shows the topo-bathymetry used in the simulations.



Selected CO-OPS observation stations

1 : 8723214	(Virginia Key, Biscayne Bay, FL)
2 : 8722670	(Lake Worth Pier, Atlantic Ocean, FL)
3 : 8721604	(Trident Pier, Port Canaveral, FL)
4 : 8720218	(Mayport (Bar Pilots Dock), FL)
5 : 8720030	(Fernandina Beach, FL)
6 : 8670870	(Fort Pulaski, GA)
7 : 8665530	(Charleston, SC)
8 : 8658163	(Wrightsville Beach, NC)
9 : 8656483	(Beaufort, Duke Marine Lab, NC)
10 : 8654467	(USCG Station Hatteras, NC)
11 : 8652587	(Oregon Inlet Marina, NC)
12 : 8637689	(Yorktown USCG Training Center, VA)
13 : 8632200	(Kiptopeke Beach, VA)
14 : 8635750	(Lewisetta, VA)
15 : 8577330	(Solomons Island, MD)
16 : 8571421	(Bishops Head, MD)
17 : 8570283	(Ocean City Inlet, MD)
18 : 8557380	(Lewes, DE)
19 : 8555889	(Brandywine Shoal Light, DE)
20 : 8536110	(Cape May, NJ)
21 : 8531680	(Sandy Hook, NJ)
22 : 8516945	(Kings Point, NY)
23 : 8467150	(Bridgeport, CT)
24 : 8465705	(New Haven, CT)

The stations were selected such that both wind and water level data were readily available for the simulation period.

Figure 15: CO-OPS wind and water level observation locations.

5.2.1 Standalone Model Evaluation

The analysis in the following sections focuses in two important aspects in parametric TC modeling: (a) the determination of a proper wind reduction factor (w_{rf}) and b) the choice of the choice of the parametric model to use for the particular simulation.

As outlined in section [Holland 1980], w_{rf} is an empirical gradient to 10-m surface boundary layer wind reduction factor used to convert the gradient winds to 10-m winds. The reported values for w_{rf} in the literature ([Vickery et al. 2007], [Powell et al. 2003], [Powell et al. 2005], [Batts et al. 1980], [Georgiou 1985] and many others), have a wide range based on the the spatial surface roughness geographic location, the time of the storm event and the characteristics of the TC. The wind reduction factors over the ocean used in the past (and in many occassions still used today) vary from as low as low of about 0.65 to as high as 0.95. Batts et al. 1980 used a value of 0.865, and Georgiou 1985, used a value of 0.825 near the eyewall, reducing to 0.75 away from the eyewall.

The developed *PaHM* system and its associated parametric modeling components need to be applied and tested to determine what parametric TC formulation is suitable for this evaluation study. As indicated in sections Holland 1980 and Generalized Asymmetric Vortex Holland, the TC models solve the same wind gradient equation 3.1 3.1 using different assumptions and possibly slightly different parameterizations of the independent or random variables of the equation. Depending on the TC characteristics and the geographical location of the storm event, some TC models perform better than others. Vickery et al. 2007, presents an excellent review of the TC models pointing out the limitations and advantages of each TC modeling approach. The argument that none of the TC models "fits" all the TC modeling senarios, requires an analysis for the determination of the "optimal" TC model to be used in each case. Using data from past TC events (possibly grouped seasonally) for a specific region, the modeler and/or forecaster could obtain a very good guess of the "optimal" TC model for the region.

Wind Reduction Factor Considerations

To determine the appropriate value for w_{rf} to be used for the *PaHM* generated wind fields for Hurricane Florence (2018) a series of simulations were performed using different values for w_{rf}. The values of wind reduction factor used were: 0.65, 0.70, 0.75, 0.80, 0.85, 0.90 and 0.95. The *PaHM* generated 10-m wind speeds were compared against wind observations at all the selected CO-OPS locations (Figure [15]). Figure [16] shows the comparison at three radomnly selected locations for w_{rf} values of 0.65 (left panel) and 0.9 (right panel) clearly indicating that the value of 0.9 for w_{rf} improves all the statistical measures significantly.

The $w_{rf} = 0.65$ case produces the worst statistics in all observation stations while, the case The $w_{rf} = 0.90$ seems to produce the best results in all locations as shown in Figure [17], indicating that for Hurricane Florence, the optimal value of the wind reduction factor is 0.9. The googness of the results using the other values of w_{rf} (not presented here) falls between the ones for cases 0.65 and 0.90. Figure [17] shows the profile of the three statistical measures (R^2, RMSE and Willmott SKILL) for all sorted CO-OPS stations shown in Figure [15].

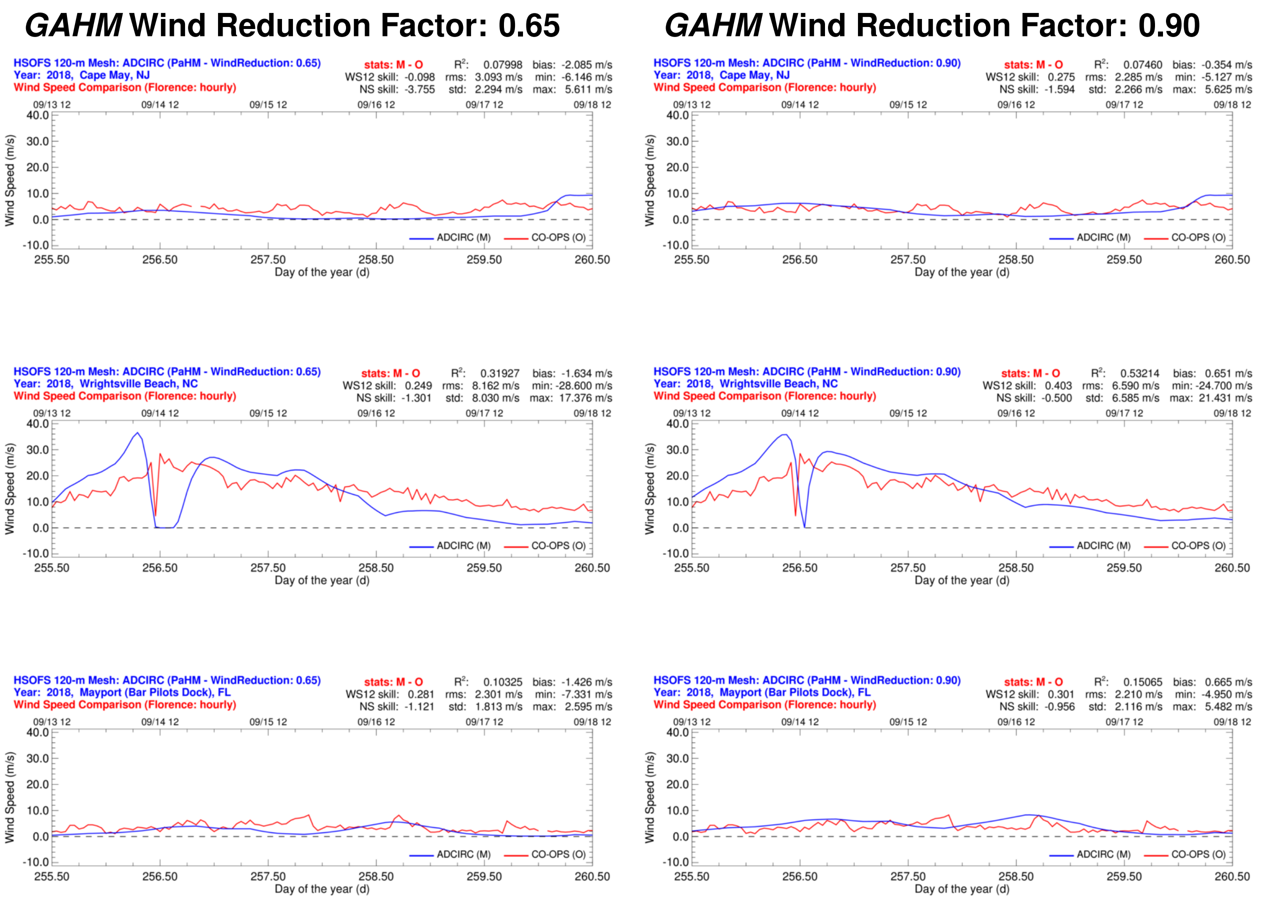


Figure 16: Effect of the wind reduction factor w_{rf} on *PaHM* predicted wind fields using the asymmetric vortex model *GAHM*.

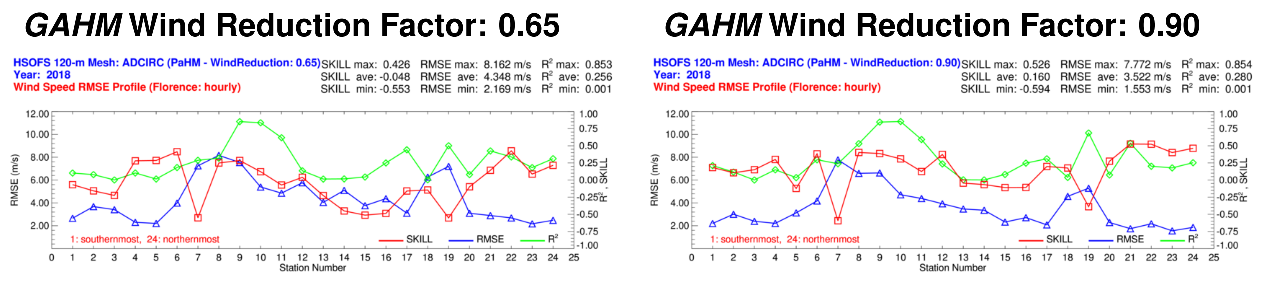


Figure 17: Spatial distribution of the wind reduction factor w_{rf} statistics along the Eastern U.S coastline. The *PaHM* wind fields were produced using the asymmetric vortex model *GAHM*.

Similar conclusions were obtained using the HM80 parametric TC model in *PaHM*, indicating that $w_{rf} = 0.9$ is the optimal value for the wind reduction factor for the Hurricane Florence case, and this is the value used in all subsequent analyses.

Parametric Model Type Considerations

This section is dedicated to the determination of the best TC model to be used for the current simulations. Figure [18] shows the results at selected CO-OPS locations using the *Holland 1980* (left panel) and *GAHM* (right panel) TC parametric models. The three CO-OPS locations were chosen as the nearest ones to the path of Hurricane Florence. As mentioned in section [Parametric Models in PaHM], TC parametric models produce their best results in the region of the tropical storm's path. It is noted here that for most of the w_{rf} cases *GAHM* produces the best results. There are a few cases where *Holland 1980* produced better results and others where the results were inconclusive.

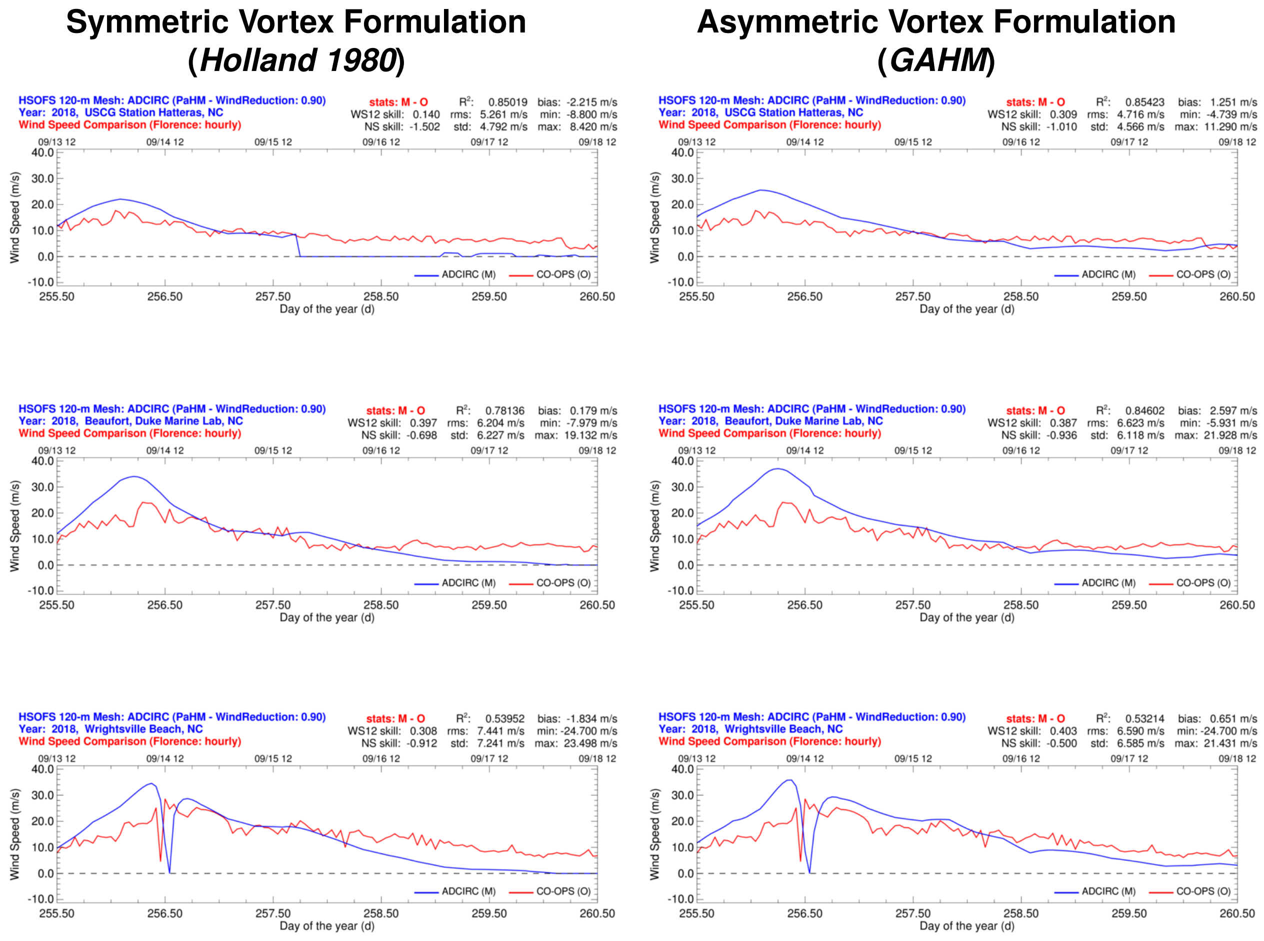


Figure 18: Comparison between Symmetric and Asymmetric Vortex Formulations (*Holland 1980* and *GAHM* respectively) of the *PaHM* predicted wind fields.

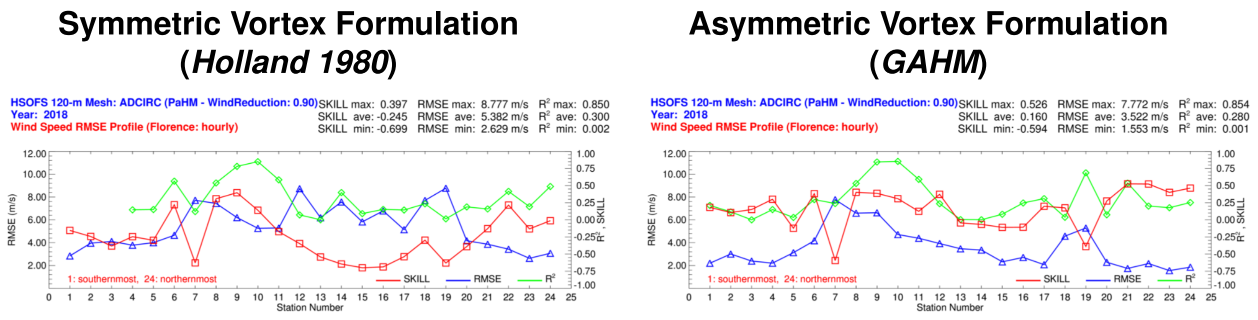


Figure 19: Spatial distribution of the model statistics along the Eastern U.S coast. The *PaHM* wind fields were produced using the traditional Holland 1980 model (left panel) and generalized asymmetric holland model *GAHM* (right panel).

As shown in Figure [19], *GAHM* produces results with much improved statistical measures (RMSE and Willmott SKILL) for all sorted CO-OPS stations indicating that *GAHM* is the best TC model to be used for the Hurricane Florence case study.

5.2.2 Coupled Model Evaluation

The coupled modeling approach to address the impacts of TC events such as hurricanes on coastal areas. *PAHM* is coupled (one way) with ADCIRC within CoastalApp using the NUOPC capability in both models. The NUOPC Caps share the *PaHM* data with ADCIRC on the common STOFS computational mesh shown in Figure [14]. *PaHM* is configured to use *GAHM* as its parametric model with $w_{rf} = 0.9$. The ADCIRC produced water levels were compared with the CO-OPS observations as shown in Figure [20]. The landfall of Hurricane Florence occured Shouthern of Wrightsville Beach, NC (CO-OPS station 8) during the morning of September 14, 2018. It is clear from Figures [18] and [20] that the best results for both the wind speed and the water levels are obtained near the landfall region and around the landfall time.

Moving away from the landfall region, the *PaHM* winds are gradually reduced to zero thus, in those areas the ADCIRC generates tidal type water elevations (zero atmospheric forcing). Apparently the *PaHM* winds need to to be coupled with external wind products to produce proper atmospheric forcings across the whole computational domain. Figure [22] shows the storm surge innundation at the impacted areas by Hurricane Florence.

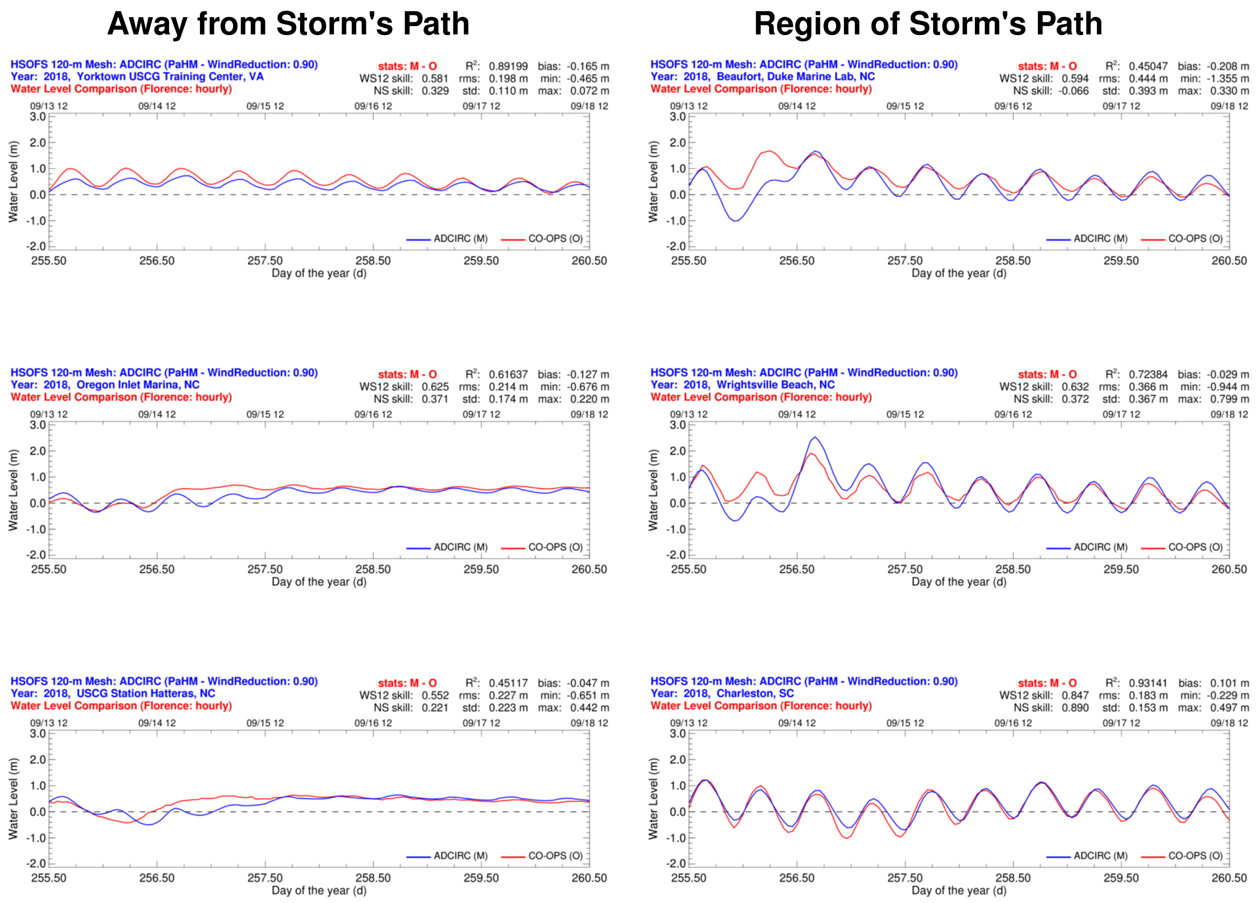


Figure 20: Comparison of ADCIRC generated water levels with CO-OPS observations.

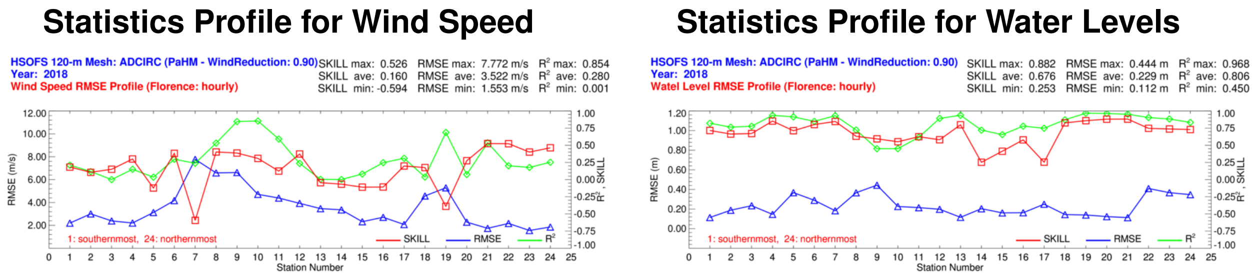


Figure 21: Spatial distribution of the statistics for the *PaHM* winds and the ADCIRC generated water levels along the Eastern U.S coast.

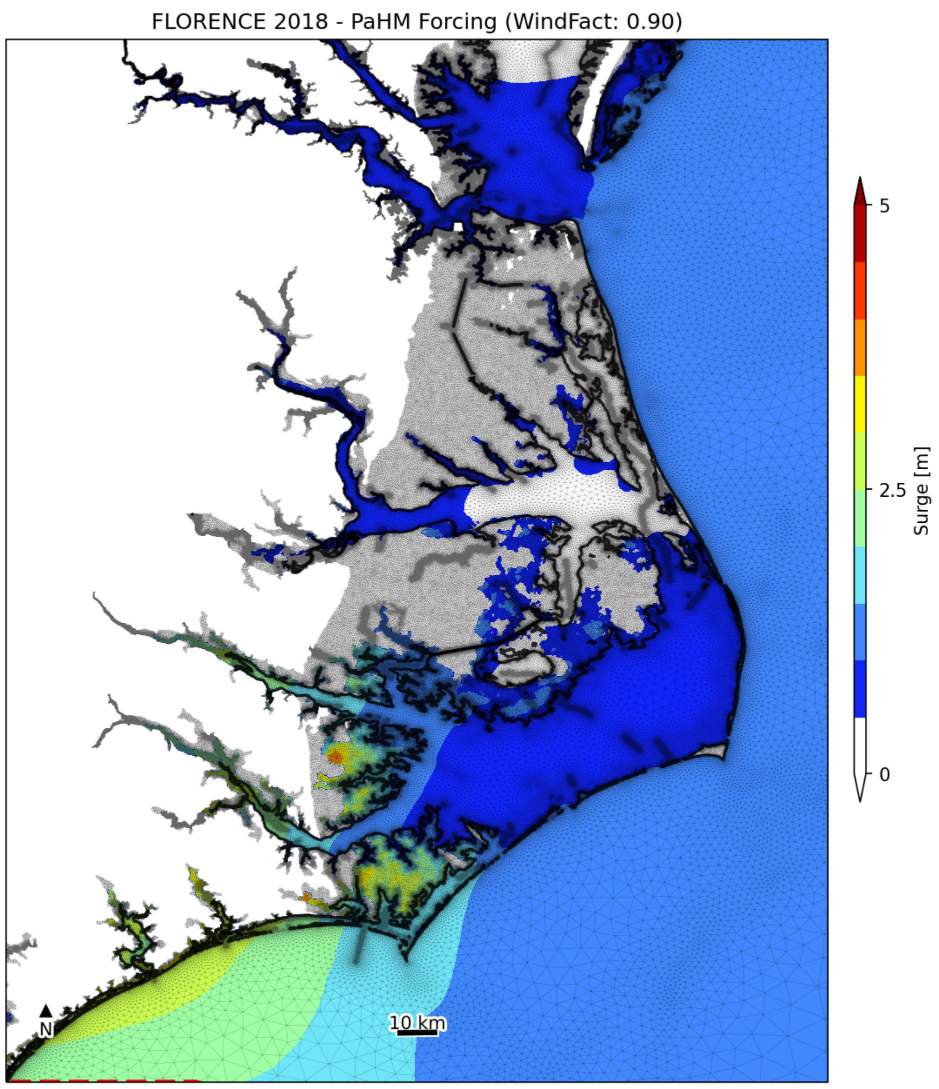


Figure 22: Distribution of the coupled *PaHM*/ADCIRC produced maximum water levels at the landfall area of Hurricane Florence.

Chapter 6

Conclusions and Future Considerations

Given the limitations of the TC parametic models discussed in sections [Holland 1980] and [Generalized Asymmetric Vortex Holland], *PaHM* performs as expected and with the anticipated accuracy near the region of the path of the tropical storm. *PaHM's* capability to run coupled with ocean and wave models makes the system a strong candidate for on-demand, hurricane forced storm surge and inundation simulations. The simulation results for Hurricane Florence were promising on predicting both the total water levels and the flood inundation exposing though the limitations of the TC parametric models as previously discussed.

As *PaHM* continues its development path, it is anticipated that improved physics will be incorporated into its parametric modeling components and new capabilities will be included in the overall system. Next are outlined some development tasks (in priority order) for improving *PaHM* and establish it as a true community atmospheric modeling system.

(1) Implementation of an Atmospheric Boundary Layer (ABL)

As mentioned earlier, the wind reduction factor w_{rf} is an empirically estimated variable with spatial and temporal dependencies. The constant value used in the TC models produce winds that are either over-estimating or under-estimating the observed winds. There were attempts in the past to use spacially varied values for w_{rf} (Vickery et al. 2007) that showed an improvement in some cases on the predicted wind fields. Lin and Chavas 2012, analyzed the uncertainties associated with w_{rf} and reported that both wind and surge estimates change greatly with w_{rf}, because it directly affects the magnitude of the surface wind associated with the storm, which dominates the wind field for extreme events. They found that every incremental increase or decrease of the value of w_{rf} by 0.05 from 0.85 increases or decreases the surge estimates on average by about 6%-7%. They also reported that the largest mean variation of the wind estimates is 26%-27% and the largest mean variation of the surge estimates was 29%-36% due to the deviations of the estimates using $w_{rf} = 0.9$ from those using $w_{rf} = 0.7$. Powell et al. 2005 state that the value of w_{rf} is uncertain and it may vary with the wind speed, and uncertainties in wind and surge estimates may be induced when using a w_{rf}. Clearly, based on the above reasoning, w_{rf} needs to be replaced in the TC parametric model calculations by a physics based atmospheric boundary layer formulation to improve the wind predictions.

(2) Implementation of extended NetCDF and GRIB capabilities

The GRIB 1/2 and NetCDF interfaces already built into *PaHM* need to be extended to allow the I/O operations in these formats to manipulate external atmospheric data as required by item (3) below. Furthermore, depending on how the TC track data were generated, the storm track data may come in different formats like NetCDF and/or GRIB.

(3) Implementation of a blending algorithm to couple background winds with *PaHM*

PaHM, as all TC parametric models, generates its fields at the gradient level such that the wind speeds outside the last closed isobar are set to zero while, the atmospheric pressure is set equal to some background pressure value (e.g., $1013.25 mb$) as shown in Figure [23]. In coastal and storm surge applications it is important to have a full forcing atmospheric field across the computational domain so "far-field" physical properties are accounted for.

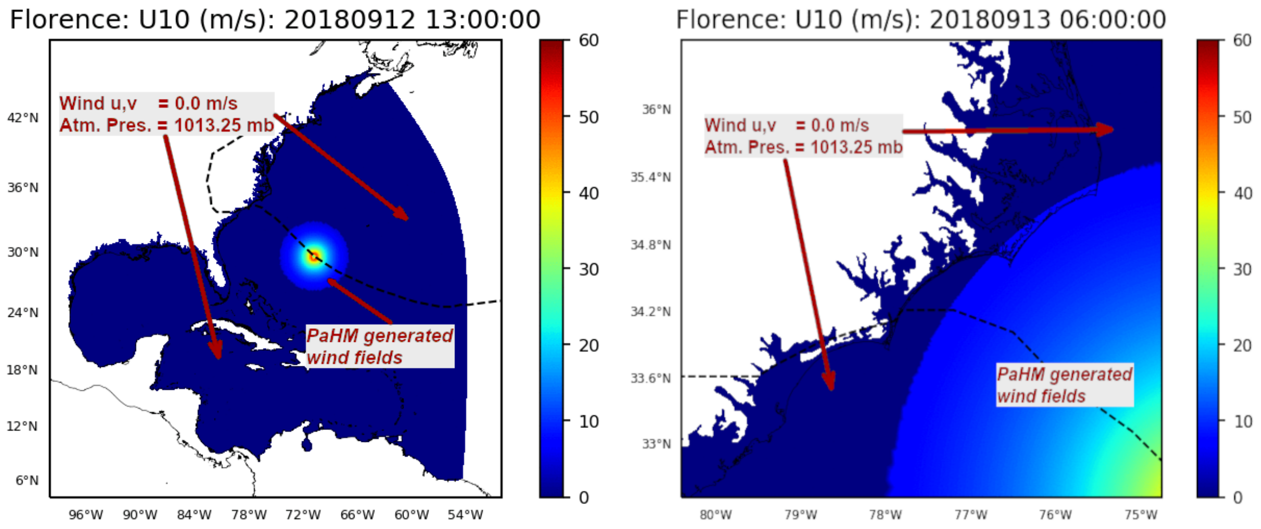

Florence: U10 (m/s): 20180912 13:00:00
Wind u,v = 0.0 m/s
Atm. Pres. = 1013.25 mb
PaHM generated wind fields
Florence: U10 (m/s): 20180913 06:00:00
Wind u,v = 0.0 m/s
Atm. Pres. = 1013.25 mb
PaHM generated wind fields

Figure 2: *PaHM* 10-m winds across the computational domain (left panel) and near the landfall region (right panel).

The background winds could come from atmospheric databases, from global weather models like *GFS*, *CFS*, etc. or from regional atmospheric models.

(4) Implementation of additional TC parametric models

PaHM code is designed to easily allow the incorporation of additional tropical cyclone parametric models. It is a good approach to have additional models like Willoughby and Rahn 2004 as each model might perform better than others under certain TC conditions. Furthermore, it will be a useful capability to allow the use of additional parametric TC models in the asymmetric vortex formulation (e.g., Rankin Vortex, Willoughby model).

(5) Implementation of additional grid/mesh definitions

Currently *PaHM* only supports ADCIRC type mesh definitions to generate data for. The mesh/grid interface needs to be extended to include definitions for other main-stream models (e.g., FVCOM, ROMS, HYCOM, etc.) and most importanly, to allow for user supplied grid/mesh definitions (through input definition file(s)).

(6) Implementation of the capability to include atmospheric fluxes

Incorporate the capability to include atmospheric fluxes and/or additional atmospheric fields for the calculation of atmospheric fluxes required to force hydrologic models like the National Water Model (NWM) or other hydrologic models. This will also allow for a consistent exchange of atmospheric fluxes between the atmosphere and the ocean, important for air-sea interactions in coastal applications.

References

[1] M. E. Batts, M. R. Cordes, L. R. Russell, J. R. Shaver, and E. urricane ind Speeds in the United States. National Bureau of Standards, Report Number BSS-124, U.S. Department of Commerce, 1980.

[2] Jie Gao, Rick Luettich, and Jason Fleming. Generalization of the Holland Parametric Tropical Cyclone Model for Forecast Applications. 14th International Workshop on Wave Hindcasting and Forecasting / 5th Coastal Hazards Symposium / 2nd International Storm Surge Symposium, 2015.

[3] Jie Gao. On the Surface Wind Stress for Storm Surge Modeling. PhD thesis, The University of North Carolina, Chapel Hill, NC, 2018.

[4] P. N. Georgiou. Design Windspeeds in Tropical Cyclone-Prone Regions. Ph.D. Thesis, Faculty of Engineering Science, University of Western Ontario, London, Ontario, Canada, 1985.

[5] Greg J. Holland. An Analytic Model of the Wind and Pressure Profiles in Hurricanes. Monthly Weather Review, 108:1212-1218, 1980.

[6] Greg J. Holland, James I. Belanger, and Angela Fritz. A Revised Model for Radial Profiles of Hurricane Winds. Monthly Weather Review, 138:4393-4401, 2010.

[7] Ning Lin and Daniel Chavas. On hurricane parametric wind and applications in storm surge modeling. Journal of Geophysical Research, 117:D09120, doi:10.1029/2011↩ JD017126, 2012.

[8] S. Moghimi, A. Van der Westhuysen, A. Abdolali, E. Myers, S. Vinogradov, Z. Ma, F. Liu and A. Mehra. Development of an ESMF Based Flexible Coupling Application of ADCIRC and WAVEWATCH III for High Fidelity Coastal Inundation Studies. Journal of Marine Science and Engineering, 8(5), 2020.

[9] J. E. Nash and J. V. Sutcliffe. River Flow Forecasting Through ConceptualModels Part I - A Discussion of Principles. Journal of Hydrology, 10:282-290, 1970.

[10] P. E. O'Connell, J. E. Nash, and J. P. Farrell. River Flow Forecasting Through ConceptualModels Part II - The Brosna Catchment at Ferbane. Journal of Hydrology, 10:317-329, 1970.

[11] M. D. Powell, S. H. Houston, L. R. Amat and N. Morisseau-Leroy. The HRD real-time

hurricane wind analysis system. J. Wind Eng. Ind. Aerodyn., 77/78:53-64, 1998.

[12] M. D. Powell, P. J. Vickery and T. A. Reinhold. Reduced drag coefficient for high wind speeds in tropical cyclones. Nature, 422(6929):279-283, 2003.

[13] M. D. Powell, G. Soukup, S. Cocke, S. Gulati, N. Morisseau-Leroy, S. Hamid, N. Dorst, and L. Axe. State of Florida hurricane loss projection model: Atmospheric science component. J. Wind Eng. Ind. Aerodyn., 93:651-674, 2005.

[14] Benjamin A. Schenkel. Are Multiple Tropical Cyclone Events Similar among Basins? Journal of Climate, 30(15):5805-5813, doi: 10.1175/JCLI-D-17-0088.1, 2017.

[15] Roland Stull. Practical Meteorology: An Algebra-based Survey of Atmospheric Science. University of British Columbia, Vancouver, Canada, pages 940, ISBN: 978-0-88865-283-6, 2017.

[16] Peter J. Vickery, Forrest J. Masters, Mark D. Powell, and Dhiraj Wadhera. Hurricane Hazard Modeling: The Past, Present and Future. Keynote lecture at the ICWE12 Cairns, Australia 2007, pages 29, 2007.

[17] Cort J. Willmott. On the Validation of Models. Physical Geography, 2:184-194, 1981.

[18] Cort J. Willmott. Some Comments on the Evaluation of Model Performance. Bulletin of the American Meteorological Society, 63:1309-1313, 1982.

[19] Cort J. Willmott, Scott M. Robesonb and Kenji Matsuura. Short Communication. A Refined Index of Model Performance. Bulletin of the American Meteorological Society, 32:2088-2094, 2012.

[20] H. E. Willoughby and M. E. Rahn. Parametric Representation of the Primary Hurricane Vortex. Part I: Observations and Evaluation of the Holland (1980) Model. Monthly Weather Review, 132:3033-3048, 2004.

[21] H. E. Willoughby, R. W. R. Darling, and M. E. Rahn. Parametric Representation of the Primary Hurricane Vortex. Part II: A New Family of Sectionally Continuous Profiles. Monthly Weather Review, 134:1102-1120, 2006.