NOAA Technical Memorandum NOS CS 59

Description of a Python Script to Implement a Probabilistic Surrogate Model for Predicting Tropical Cyclone Driven Flood Depth and Extent

Silver Spring, Maryland
August 2024

noaa **National Oceanic and Atmospheric Administration**

U.S. DEPARTMENT OF COMMERCE
National Ocean Service
Coast Survey Development Laboratory

Office of Coast Survey
National Ocean Service
National Oceanic and Atmospheric Administration
U.S. Department of Commerce

The Office of Coast Survey (OCS) is the Nation's only official chartmaker. As the oldest United States scientific organization, dating from 1807, this office has a long history. Today it promotes safe navigation by managing the National Oceanic and Atmospheric Administration's (NOAA) nautical chart and oceanographic data collection and information programs.

There are four components of OCS:

The Coast Survey Development Laboratory develops new and efficient techniques to accomplish Coast Survey missions and to produce new and improved products and services for the maritime community and other coastal users.

The Marine Chart Division acquires marine navigational data to construct and maintain nautical charts, Coast Pilots, and related marine products for the United States.

The Hydrographic Surveys Division directs programs for ship and shore-based hydrographic survey units and conducts general hydrographic survey operations.

The Navigational Services Division is the focal point for Coast Survey customer service activities, concentrating predominately on charting issues, fast-response hydrographic surveys, and Coast Pilot updates.

NOAA Technical Memorandum NOS CS 59

Description of a Python Script to Implement a Probabilistic Surrogate Model for Predicting Tropical Cyclone Driven Flood Depth and Extent

Fariboz Daneshvar[1, 2], Soroosh Mani[1], William J. Pringle[3], Saeed Moghimi[1], Edward Myers[1]

1. Office of Coast Survey, Coast Survey Development Laboratory, Silver Spring, Maryland
2. Earth Resources Technology (ERT), Laurel, Maryland
3. Environmental Science Division, Argonne National Laboratory, Chicago, Illinois

August 2024

noaa National Oceanic and Atmospheric Administration

NOTICE

Table of Contents

Table of Code Snippets

Table of Figures

Abstract

This document introduces a Python script for generating probabilistic predictions of tropical cyclone (TC) driven coastal flooding. It uses a machine learning (ML) framework to develop a probabilistic surrogate model for maximum water elevation based on an ensemble of water elevations simulated by a hydrodynamic ocean model. It also provides sensitivity analysis, uncertainty quantification, and probabilistic prediction of flood depth and extent based on user-defined criteria. The script employs the Polynomial Chaos ML method to develop the probabilistic surrogate model with several choices of penalized regression and cross-validation options. It has been generalized to work with outputs of different hydrodynamic models (such as the ADvanced CIRCulation (ADCIRC) or Semi-implicit Cross-scale Hydroscience Integrated System Model (SCHISM)), and can be expanded to use other ML methods such as neural network- or decision-tree-based ones. The script was showcased with a step-by-step guide and evaluation of outputs for the case study of Hurricane Florence (2018) with 48 hours lead time. The probabilistic surrogate model provided a reliable prediction of maximum water elevation compared to the outputs of a hydrodynamic ocean model. This user-friendly script can be used for both risk assessment and early warning systems during the forecast period, as well as post storm flood damage assessment and better evaluation of flood insurance costs.

Key Words: coastal flooding, modeling, SCHISM, storm surge, surrogate model, probability, machine learning, uncertainty quantification, Hurricane Florence (2018).

1. Introduction

Tropical Cyclone (TC) driven storm surges pose serious humanitarian and economic threats to coastal communities. TC-driven hurricanes have on average resulted in over $22 billion damage and caused 6,890 fatalities since 1980 (NOAA COAST, 2024). These negative impacts are expected to worsen with climate change trends, and population and economic growth of coastal communities (Marsooli et al., 2019), which signifies the crucial need for fast and reliable prediction of storm surge and coastal flooding.

Probabilistic approaches have become popular for uncertainty quantification and risk assessment of storm surge and coastal flooding especially in a limited-resource environment, to provide early warnings and risk communication (Davis et al., 2010; Kim et al., 2015; Hashemi et al., 2016; Plumlee, 2021; Pringle et al., 2023). Pringle et al. (2023) developed a probabilistic framework based on Artificial Intelligence (AI) technology that provides probabilistic prediction of coastal flooding based on an ensemble of hydrodynamic models' simulations. Pringle et al. (2023) evaluated their methodology for three hurricanes (Irma in 2017, Florence in 2018, and Laura in 2020) and found that it provides reliable prediction of maximum water elevation and flood extent.

The goal of this work is to put together a user-friendly script to implement Pringle et al. (2023) methodology, and generalize implementation of user-defined features for different steps of the process. This python script has been developed based on the EnsemblePerturbation package (EnsemblePerturbation, 2023). The rest of this document will provide an overview of the workflow and features of the script, followed by a step-by-step explanation of the process and expected results for a case study. Outcome of this work will provide a valuable tool for pre- and post-storm probabilistic evaluation of coastal flooding, and can be used for early warning as well as damage assessment.

2. Workflow Overview

In this section, the workflow is described from a high-level perspective (see Figure 1). In order to generate a probabilistic model for maximum water elevation, the workflow relies on water elevations corresponding to an ensemble of storm tracks with trajectory, intensity, and size variables perturbed from the original forecast or hindcast. The workflow takes an ensemble of maximum water elevations, calculated by a hydrodynamic model such as ADvanced CIRCulation (ADCIRC) (Luettich and Westerink, 2004) or Semi-implicit Cross-scale Hydroscience Integrated System Model (SCHISM) (Zhang et al., 2016), as input along with values for the perturbation parameters for each ensemble member track. This ensemble is typically generated by requesting a low-discrepancy sequence (e.g., Korobov) perturbation of the storm parameters (e.g., track pathway, size, and intensity). For testing purposes, the input can be divided into a training and a validation set for developing the probabilistic surrogate model which is a machine learning model which, after training, can be cheaply evaluated at nodes to replace the expensive ocean model in order to cover the full perturbation uncertainty space. To further reduce computational cost, only a subset of the domain near to the landfall area is calculated and used in the subsequent steps of the analysis. Also, the dataset is manipulated and the water elevation is artificially extended over dry nodes next to the flooded area (wet nodes) to distinguish them from dry nodes far from the coastline, and so improve the performance of the probabilistic model. The extended dry node elevation values are negative; the more negative the values are, the farther away from the wet nodes they are, and values closer to zero are closer to wet nodes.

The Karhunen–Loeve (KL) - Polynomial Chaos (PC) framework (Pringle et al., 2023) is used to develop a surrogate model based on the training set, to find the relation between perturbation parameters and the subset of manipulated water elevations. To do so, first, KL decomposition is used to reduce the dimensionality (number of mesh nodes) to O(1-10) modes of variation that explain a user-specified variance within the ensemble. Second, a PC regression model is used to generate a surrogate model that relates the perturbation parameters to the KL reduced modes (KL surrogate). Then the KL-PC surrogate is transformed back into the physical node space to provide a relation between the perturbation and the water elevations. Finally, the developed surrogate model is evaluated against the validation set and interrogated to derive the parameter sensitivities and percentiles of the distribution.

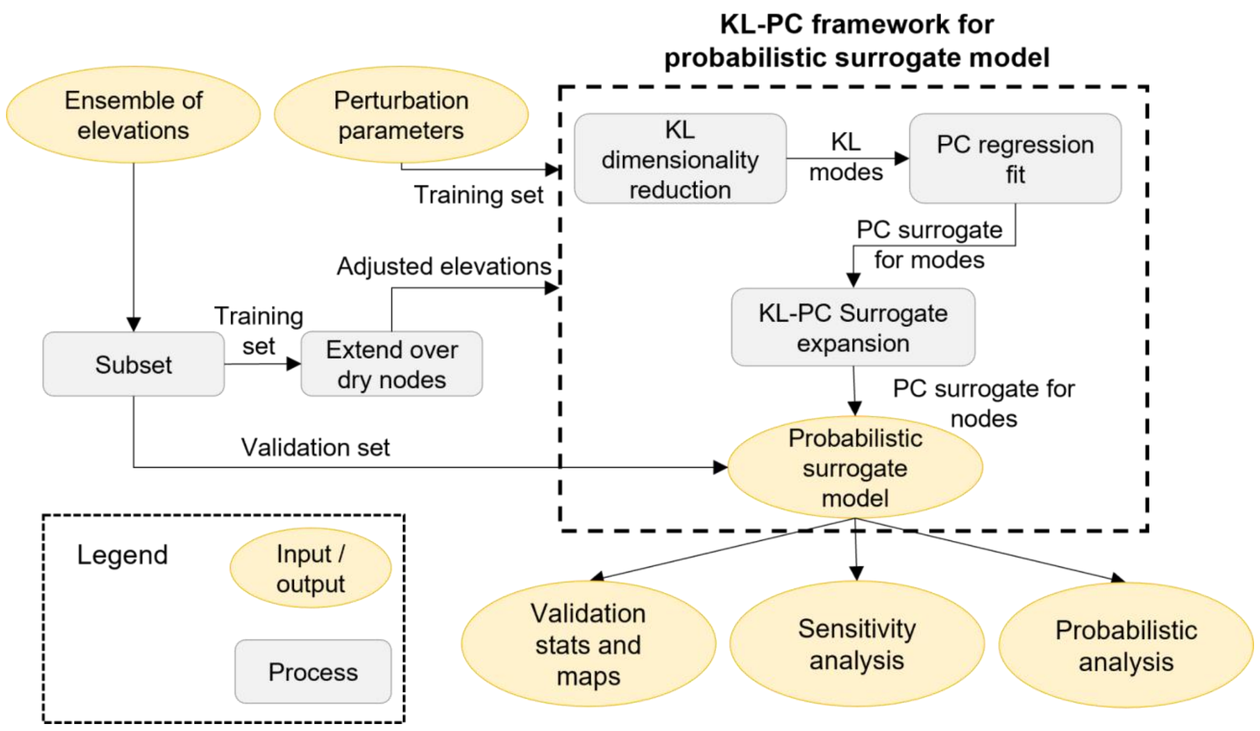


Figure 1. Workflow overview

Features

This probabilistic framework provides the following options and features:

- **Training/validation set division**: user-defined ratio for division of dataset into training and validation sets.
- **Subset the region**: dataset subsetting capability based on user-defined depth and isotach criteria.
- **Eliminate negative predictions**: the option of using the log scale of elevation to avoid negative values for training the probabilistic surrogate model.
- **Distinguishing dry nodes:** dry-node distinction by extrapolation of water elevation over dry nodes based on user-defined hydraulic head loss and inverse distance weight parameters.
- **Accuracy of dimensionality reduction**: user-defined level of explained variance for dimensionality reduction.
- **Multiple choices of PC regression**: ability to use multiple regression fits with a range of cross-validation options.
- **Sensitivity analysis**: global sensitivity analysis of the storm parameters (trajectory, intensity, and size) that contribute to the forecast uncertainty.
- **Probabilistic analysis**: Probabilistic analysis for a range of percentiles and calculation of the probability of exceeding user-defined water levels.

3. Step-by-Step Process with a Case Study

This section provides a comprehensive description of the Python script with code snippets, followed by the outputs for a case study of Hurricane Florence (2018), that landed as a category one storm in North Carolina in September 2018 and resulted in 51 fatalities and more than 17 billion dollars damages to the coastal communities of the Carolinas (National Weather Service, 2024).

As noted in section 2, two sets of inputs–ensembles of water elevations and perturbation parameters–are needed. The National Hurricane Center (NHC) forecast errors of four parameters (cross track, along track, maximum sustained wind speed, and radius of maximum speed), and the official advisory storm track of Hurricane Florence with 48 hours lead time (Figure 2) were used to generate perturbed hurricane tracks. Korobov sequence (Korobov, 1959) and random sampling were used to generate two ensembles of 30 and 10 perturbed hurricane tracks for training and validation, respectively. These tracks were used as inputs to the SCHISM model to generate 40 ensembles of maximum elevations for each mesh node in the domain. The SCHISM simulation domain covers the entire East Coast and the Gulf of Mexico with an unstructured mesh ranging from 21 km (in deep ocean) to 4.5 m (in the area of interest) horizontal resolution using 1,291,145 elements and 673,957 nodes (Figure 2). The rest of this section will show how to use perturbation parameters and corresponding SCHISM simulations of the maximum water elevations to generate a probabilistic surrogate model and conduct a probabilistic analysis of the flooding depth and extent for the 48-hr Hurricane Florence forecast.

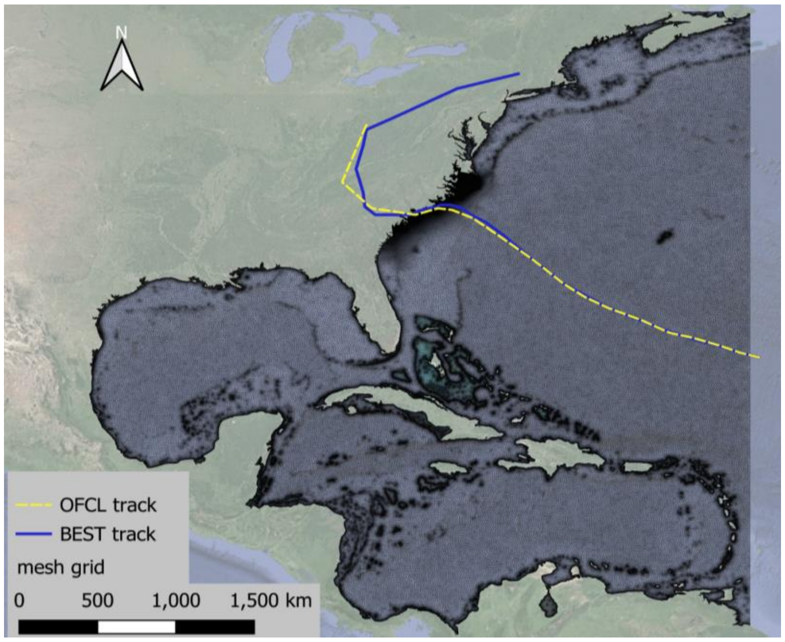


Figure 2. NHC 48-hr official (ofcl) forecast and hindcast best-track for Hurricane Florence (2018), overlaid on the mesh grid for the SCHISM simulation of water elevation. Denser black color indicates where mesh nodes are closer together (higher-resolution), particularly notable around the hurricane landfalling region.

Python Packages and Dependencies

First the script imports required modules and packages (Table 1). This script uses Python Standard Library along with some external packages. Here is the list of packages and modules that should be imported.

Table 1. List of Python dependencies and purposes of use

Purpose	Package/module	class/functions
Dataset manipulation	ensembleperturbation.parsing.adcirc	extrapolate_water_elevation_to_dry_areas
		subset_dataset
Generic I/O	argparse	ArgumentParser
	ensembleperturbation.utilities	get_logger
	Pathlib	Path
KL dimensionality analysis	ensembleperturbation.uncertainty_quantification.karhunen_loeve_expansion	karhunen_loeve_expansion
		karhunen_loeve_prediction
Numerical arrays handling	numpy	-
	xarray	-
Parallelization	dask	-
	pickle	-
Parametrization	chaospy	-
	ensembleperturbation.perturbation.atcf	VortexPerturbedVariable
PC regression	sklearn.linear_model	ElasticNetCV
		LassoCV
		LinearRegression
	sklearn.model_selection	LeaveOneOut
		ShuffleSplit
Plotting (visualization)	matplotlib.pyplot	-
	ensembleperturbation.plotting.perturbation	plot_perturbations
	ensembleperturbation.plotting.surrogate	plot_kl_surrogate_fit
		plot_selected_percentiles
		plot_selected_probability_fields
		plot_selected_validations
		plot_sensitivities
		plot_validations
Post-processing of the surrogate model	ensembleperturbation.uncertainty_quantification.surrogate	probability_field_from_surrogate
		percentiles_from_surrogate
		sensitivities_from_surrogate
		surrogate_from_karhunen_loeve
		surrogate_from_training_set
		validations_from_surrogate

```python
from argparse import ArgumentParser
import chaospy
import dask
from matplotlib import pyplot
import numpy
from pathlib import Path
import pickle
from sklearn.linear_model import LassoCV, ElasticNetCV, LinearRegression
from sklearn.model_selection import ShuffleSplit, LeaveOneOut
import xarray
```

Code Snippet 1. List of built-in Python packages and modules for the analysis.

In addition to that, this script has been developed based on modules from the EnsemblePerturbation package as follows:

```python
from ensembleperturbation.parsing.adcirc import (
    extrapolate_water_elevation_to_dry_areas,
    subset_dataset,
)
from ensembleperturbation.perturbation.atcf import VortexPerturbedVariable
from ensembleperturbation.plotting.perturbation import plot_perturbations
from ensembleperturbation.plotting.surrogate import (
    plot_kl_surrogate_fit,
    plot_selected_percentiles,
    plot_selected_validations,
    plot_sensitivities,
    plot_validations,
    plot_selected_probability_fields,
)
from ensembleperturbation.uncertainty_quantification.karhunen_loeve_expansion import (
    karhunen_loeve_expansion,
    karhunen_loeve_prediction,
)
from ensembleperturbation.uncertainty_quantification.surrogate import (
    percentiles_from_surrogate,
    sensitivities_from_surrogate,
    surrogate_from_karhunen_loeve,
    surrogate_from_training_set,
    validations_from_surrogate,
    probability_field_from_surrogate,
)
from ensembleperturbation.utilities import get_logger
LOGGER = get_logger('KL_PC')
```

Code Snippet 2. List of Python modules exported from the EnsemblePerturbation package.

The next step is setting paths to input files (perturbation parameters, ensemble of perturbed storm tracks, and ensemble of maximum elevations), output directory, and then reading those input files.

```python
# Define paths and read files
output_dir = Path('Path_to_save_outputs')
tracks_dir = Path('Path_to_tracks_dir')
storm_name = tracks_dir / 'original.22'
perturbations = xarray.open_dataset('Path_to_perturbations.nc', chunks='auto')
max_elevations = xarray.open_dataset('Path_to_maxele.63.nc', chunks='auto')
```

Code Snippet 3. Define paths to the inputs/outputs, and reading files.

Training and Validation Sets, and Plot Perturbations

Two sets of perturbed tracks based on Korobov and random sampling were used to generate two sets of ensembles for training and validation, respectively. The following script will read perturbation files and label ensemble members (based on the perturbation methods) for training and validation.

```python
training_set = 'korobov'
validation_set = 'random'
perturbations = perturbations.assign_coords(
    type=(
        'run',
        (
            numpy.where(
                perturbations['run'].str.contains(training_set),
                'training',
                numpy.where(
                    perturbations['run'].str.contains(validation_set),
                    'validation',
                    'none',
                ),
            )
        ),
    )
)

training_perturbations = perturbations.sel(run=perturbations['type'] == 'training')
validation_perturbations = perturbations.sel(run=perturbations['type'] == 'validation')
```

Code Snippet 4. Define training and validation sets of perturbed storm tracks.

The plot_perturbations() function from the EnsemblePerturbation (code snippet 5) was then used to plot perturbed storm tracks (Figure 3), and distribution of perturbation parameters (Figures 4 to 7). Its inputs include the ensemble of perturbed storm tracks, and two sets of perturbation parameters for training and validation.

```
plot_perturbations(
    training_perturbations=training_perturbations,
    validation_perturbations=validation_perturbations,
    runs=perturbations['run'].values,
    perturbation_types=perturbations['type'].values,
    track_directory=tracks_dir,
    output_directory=output_dir)
```

Code Snippet 5. plot_perturbation() function.

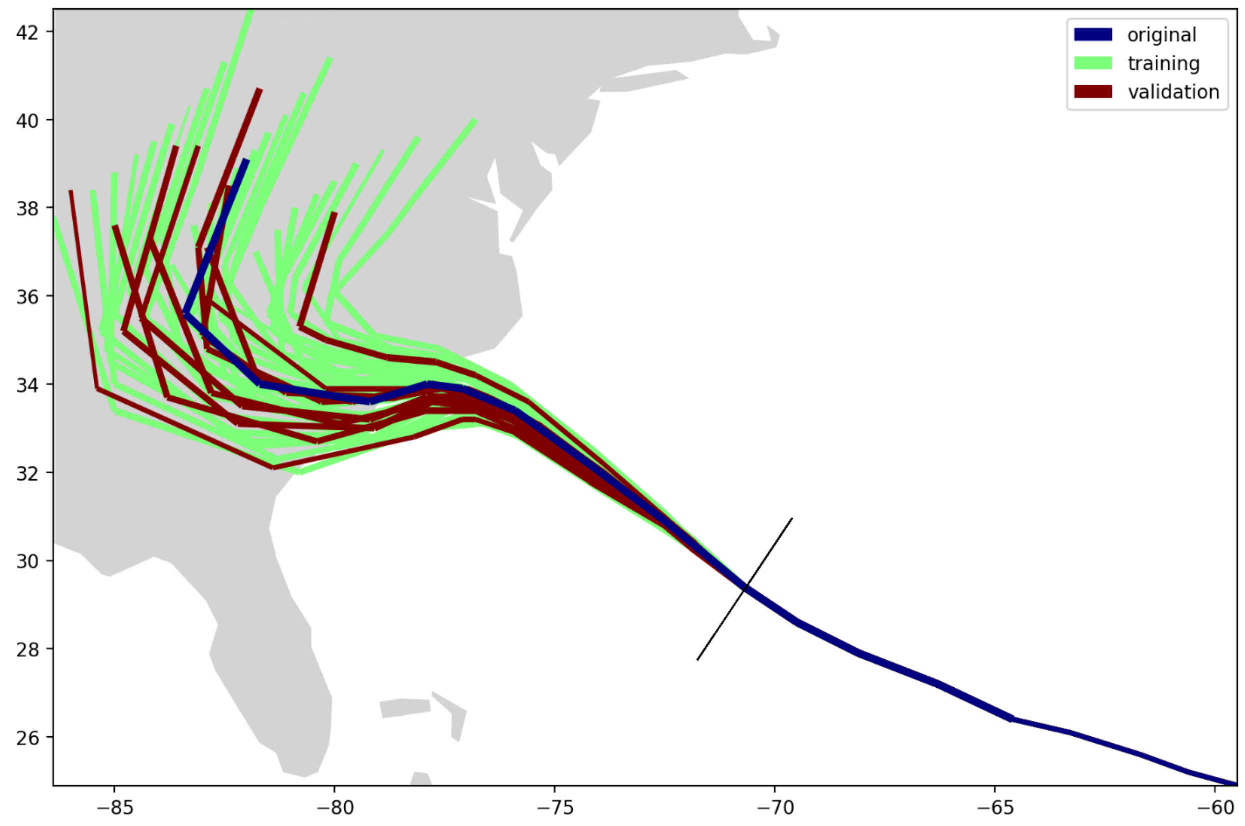


Figure 3. 30-member training and 10-member validation ensemble of perturbed official advisory storm track with 48-hr lead time for Hurricane Florence of 2018. Black line was added manually to roughly differentiate hindcast track (on the right), from perturbed forecast tracks (on the left).

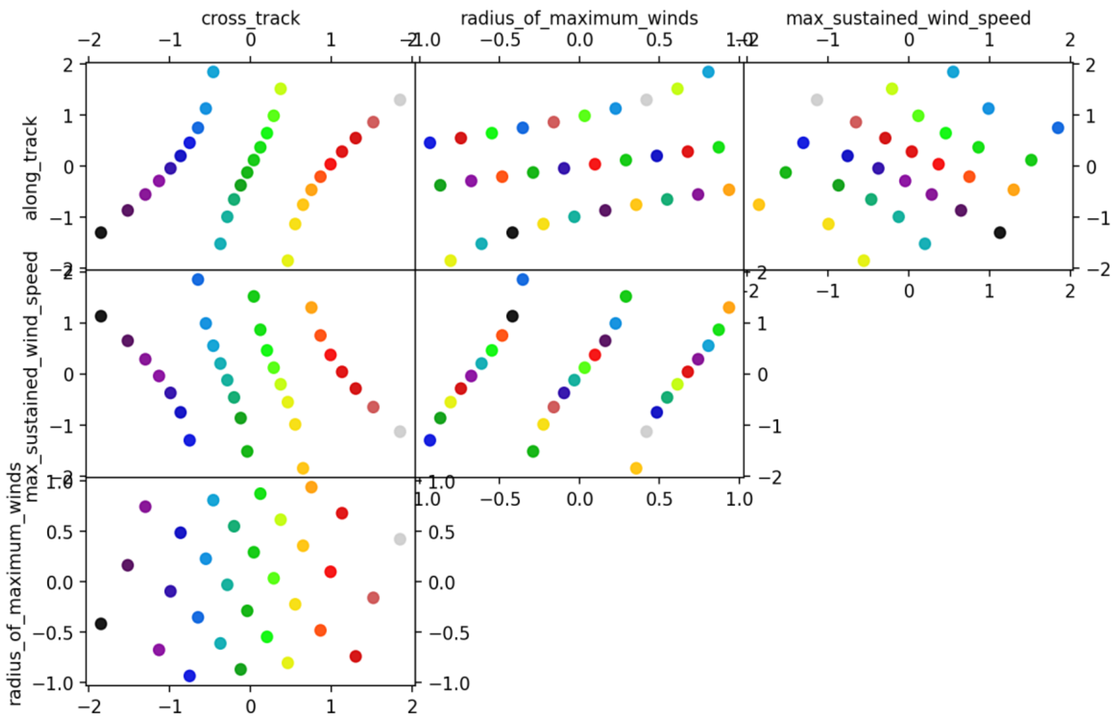


Figure 4. 2-D distribution of four perturbed parameters across 4-D space for 30 training members. The training perturbations were generated using Korobov sequence methods.

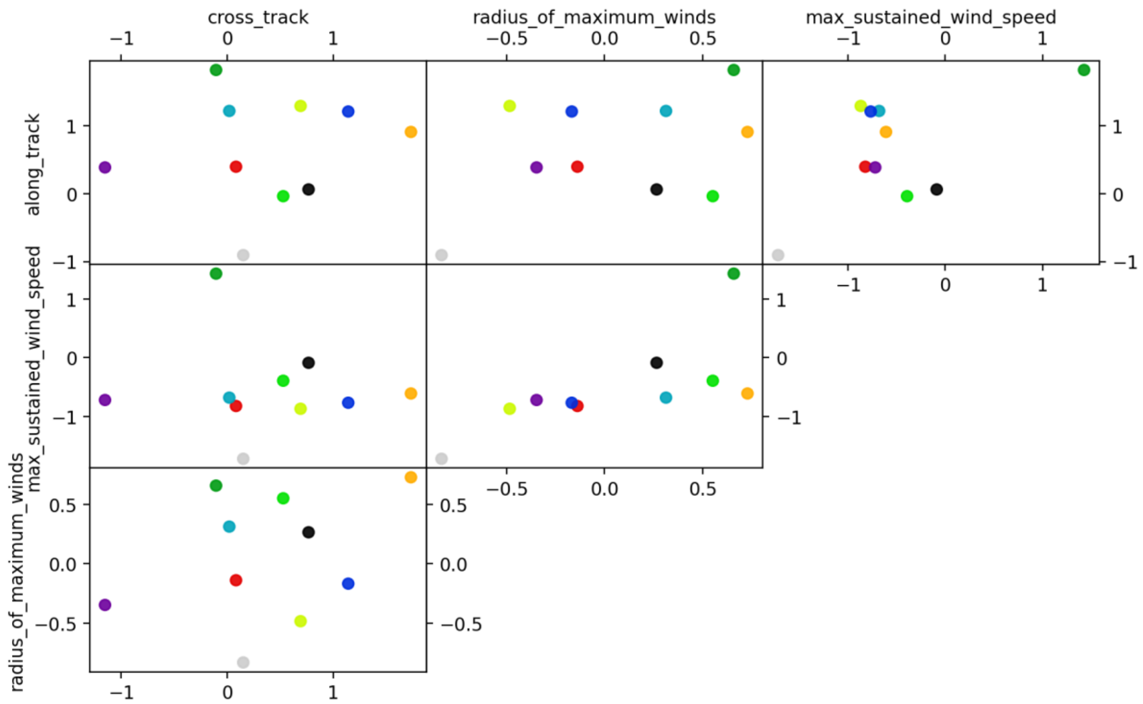


Figure 5. 2-D distribution of four perturbed parameters across 4-D space for 10 validation members. The validation perturbations were generated using random sampling methods.

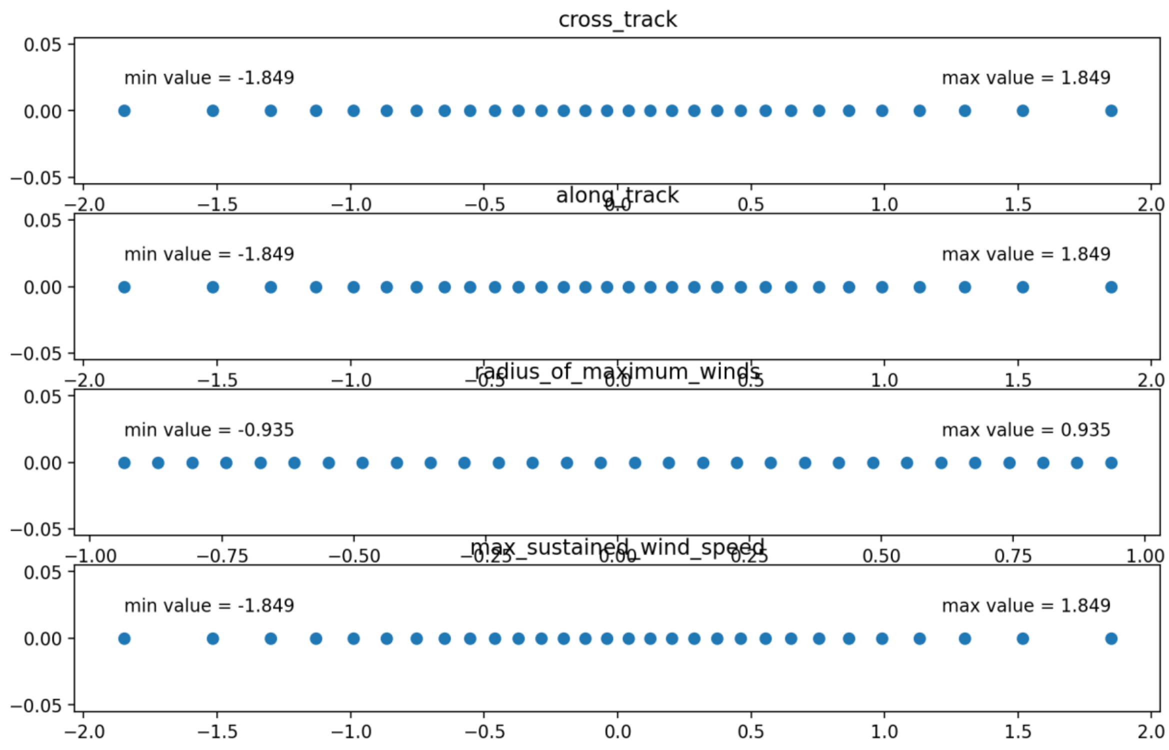


Figure 6. 1-D distribution of four perturbed parameters for 30 training members. The training perturbations were generated using Korobov sequence methods.

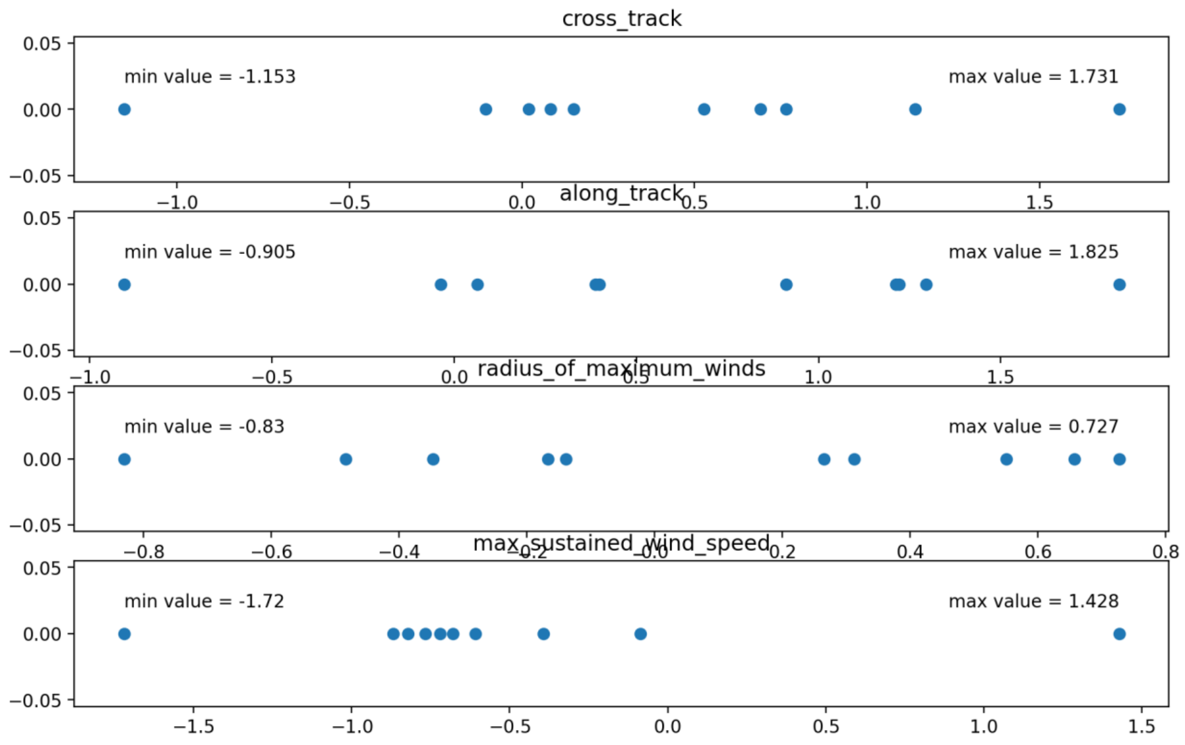


Figure 7. 1-D distribution of four perturbed parameters for 10 validation members. The validation perturbations were generated using random sampling methods.

Subset Dataset and Extrapolation Over Dry Nodes

The outputs of the SCHISM simulations have water elevations at 673,957 computational nodes for the entire simulation domain (Figure 2). In order to reduce computational cost, only the area of interest (i.e., the area around the hurricane Florence landfall), is used for further analysis. Because surge is typically only significant when winds are strong over shallow depths, the subsetting of model outputs (i.e., maximum elevation, called zeta_max) is based on user-defined criteria of the hurricane isotach and maximum ocean depth. Typically, the 34-kt isotach and 25-m isobath is used as shown below for inputs to the subset_dataset() function from the EnsemblePerturbation package below:

```python
# subsetting parameters
variable_name = 'zeta_max'
isotach = 34
depth_bounds = 25.0

# sample based on the subset
LOGGER.info('subsetting nodes')
subset_filename=output_dir/'subset.nc'
subset = subset_dataset(
    ds=max_elevations,
    variable=variable_name,
    maximum_depth=depth_bounds,
    wind_swath=[storm_name, isotach],
    output_filename=subset_filename,
)
```

Code Snippet 6. Subsetting model outputs based on user-defined criteria for the domain.

The output of the script shown in code snippet 6 is a subset of the dataset down to 328,715 nodes, which is 48.8% of the original mesh. The next step involves loading the subsetted dataset (code snippet 7), and dividing it into two sets for training and validation (code snippet 8).

```python
# load saved subset
LOGGER.info(f'loading subset from "{subset_filename}"')
subset = xarray.open_dataset(subset_filename)
if 'element' in subset:
    elements = subset['element']
subset = subset[variable_name]
```

Code Snippet 7. Load subsetted dataset.

```
# divide subset into training/validation runs
with dask.config.set(**{'array.slicing.split_large_chunks': True}):
    training_set = subset.sel(run=training_perturbations['run'])
    validation_set = subset.sel(run=validation_perturbations['run'])

LOGGER.info(f'total {training_set.shape} training samples')
LOGGER.info(f'total {validation_set.shape} validation samples')
```

Code Snippet 8. Divide the subsetted dataset into training and validation sets.

In order to improve the probabilistic prediction of inundation, as recommended by Pringle et al. (2023), the training dataset is adjusted by artificially extrapolating water elevation over dry neighboring mesh points. This manipulation of the dataset will take into account nearshore dry nodes for model development and differentiate them from dry nodes far from the flood region that might never get flooded. For this purpose, inverse distance weighting (IDW) extrapolation with a hydraulic head loss factor (with user-defined parameters suggested by Pringle et al. (2023) is used as follows:

```
# make an adjusted training set for dry areas..
mann_coef = 0.025 # manning's n coefficient
k_neighbors = 1
idw_order = 1
u_ref=0.4 # flow velocity
d_ref=1.0 # flow depth

training_set_adjusted = extrapolate_water_elevation_to_dry_areas(
    da=training_set,
    k_neighbors=k_neighbors,
    idw_order=idw_order,
    compute_headloss=True,
    mann_coef=mann_coef,
    u_ref=u_ref,
    d_ref=d_ref,
)

training_set_adjusted += training_set_adjusted['depth']
```

Code Snippet 9. Manipulate training set.

Dimensional Reduction

The following script (shown in the code snippet 10) uses the Karhunen-Loeve (KL) expansion, based on Principal Component Analysis (PCA) (Abdi and Williams, 2010), to reduce the dimensions of the subsetted (328,715) nodes with user-defined level of variance (i.e., 0.99).

```
# Evaluating the Karhunen-Loeve expansion
variance_explained = 0.99
nens, ngrid = training_set.shape

LOGGER.info(f'Evaluating Karhunen-Loeve expansion from {ngrid} grid nodes and {nens} ensemble members')
kl_expansion = karhunen_loeve_expansion(
    training_set_adjusted.values,
    neig=variance_explained,
    method='PCA',
    output_directory=output_dir,
)
LOGGER.info(f'found {kl_expansion["neig"]} Karhunen-Loeve modes')
LOGGER.info(f'Karhunen-Loeve expansion: {list(kl_expansion)}')
```

Code Snippet 10. Evaluation of KL expansion.

It will reduce the dimension of the dataset and return KL expansion parameters such as eigenvalues shown in Figure 8.

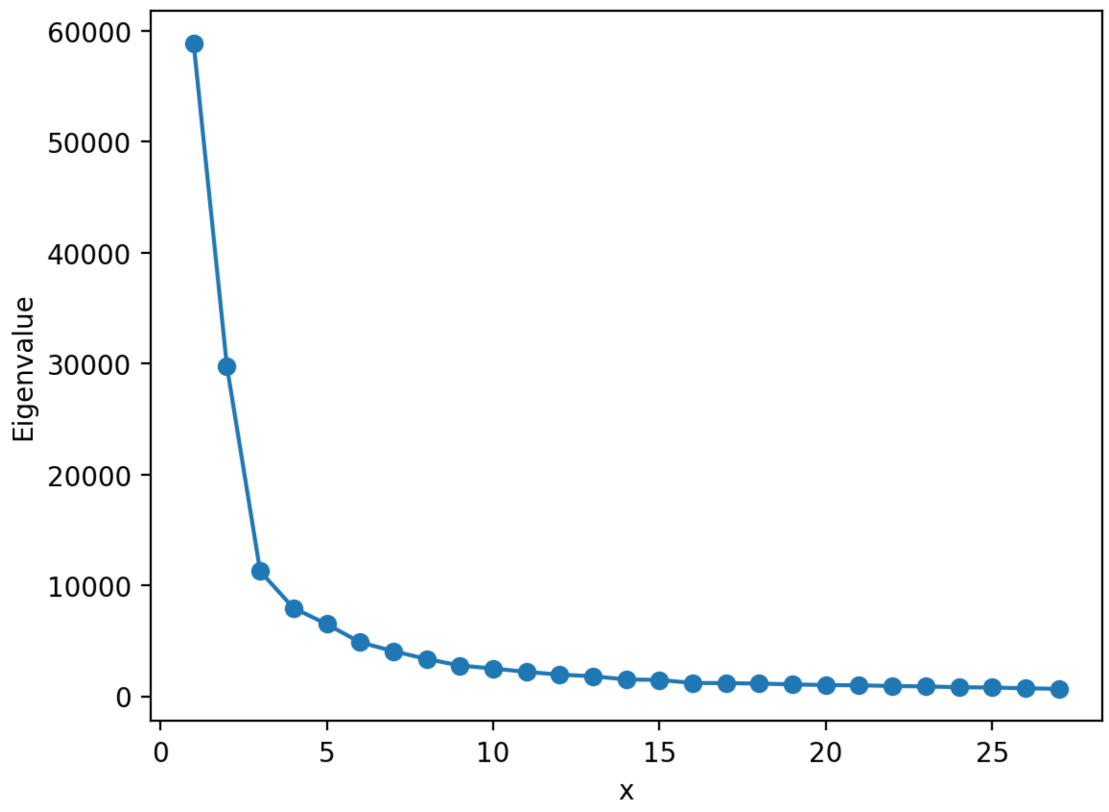


Figure 8. Plot of 27 eigenvalues of the subsetted dataset.

The next step is the evaluation of the KL prediction (code snippet 11) to reconstruct water elevation for each ensemble member. Figure 9 shows a plot of reconstructed elevations versus actual values for each ensemble run of (30) training sets, and Figure 10 shows the spatial plot of actual and reconstructed adjusted water elevations (with depths) for a randomly selected ensemble member (No. 15). Note that plotted depths include negative values due to the artificial extrapolation. Negative values are kept for training the surrogate model but are omitted at the time of validation or probabilistic prediction.

```
# plot prediction versus actual simulated
kl_predicted = karhunen_loeve_prediction(
    kl_dict=kl_expansion,
    actual_values=training_set_adjusted,
    ensembles_to_plot=[0, int(nens / 2), nens - 1],
    element_table=elements,
    plot_directory=output_dir,
)
```

Code Snippet 11. Evaluation of KL prediction.

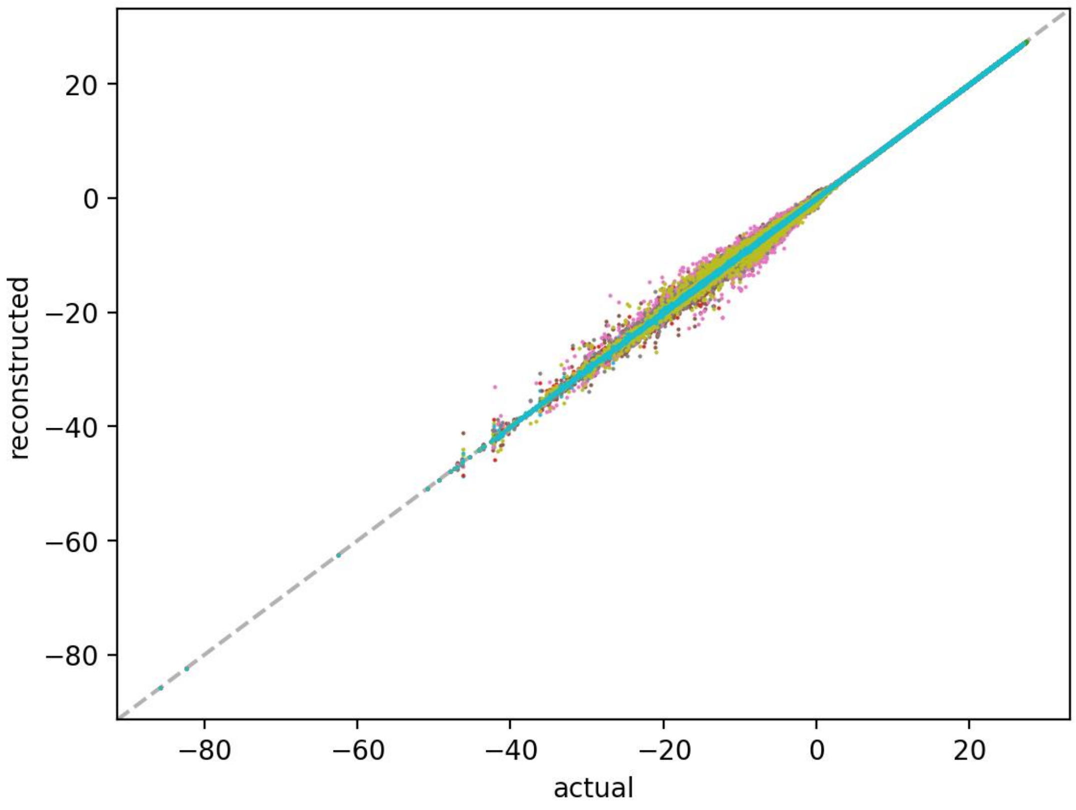


Figure 9. KL fit of adjusted water elevation (in meters) at each subsetted node among all training ensemble members.

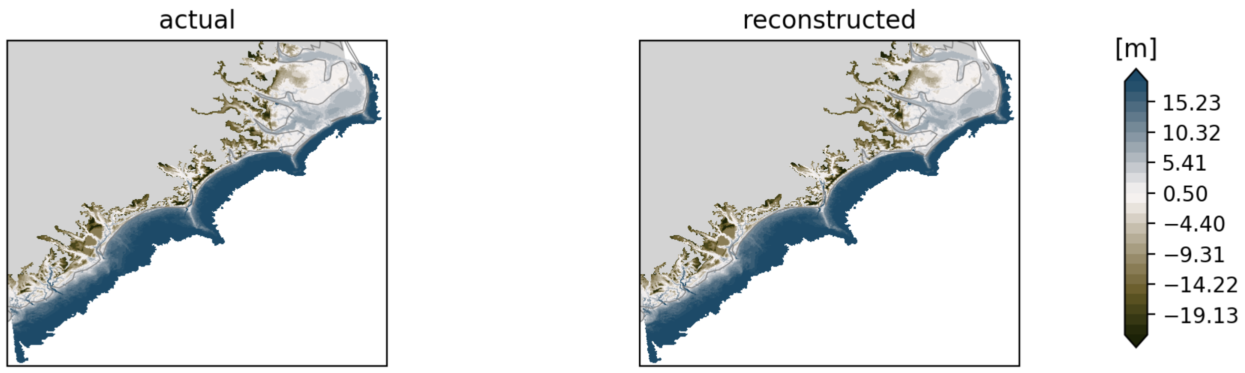
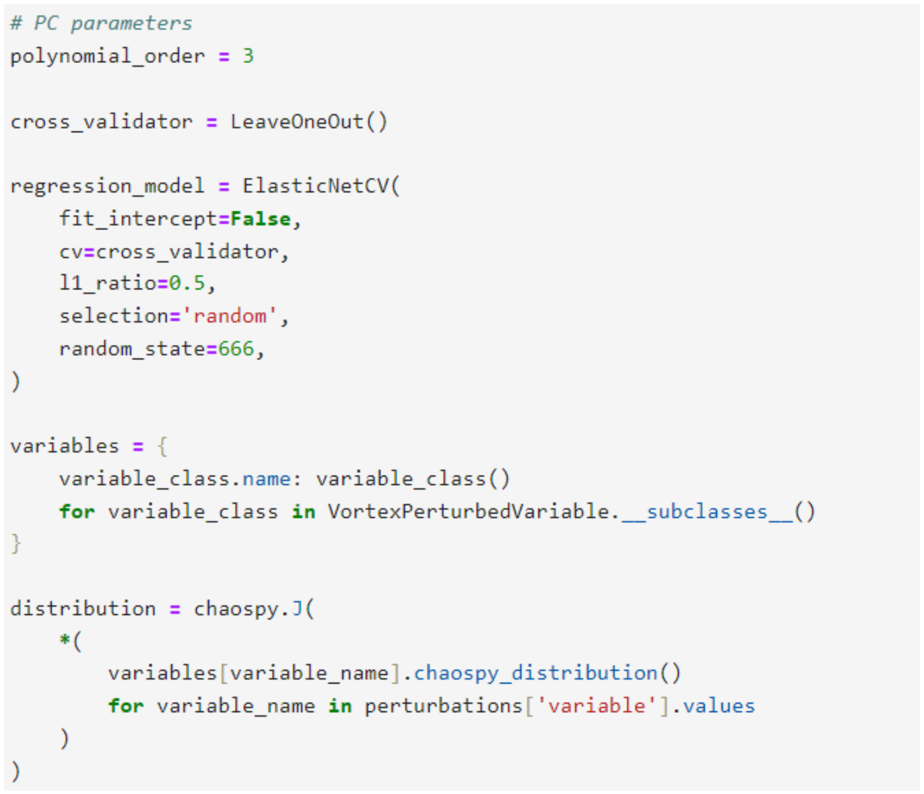


Figure 10. Comparison of the actual and KL reconstruction of simulated water depth (with extrapolation adjustment) (in meters) for ensemble No. 15.

Surrogate Model Expansion

A third-order Polynomial Chaos (PC) based on elastic net penalized regression with leave-one-out cross-validation (from scikit-learn package) is used to develop the surrogate model. These parameters are user-defined and can be changed to other potential values such as second-order PC (=2), lasso regression (=LassoCV), and shuffle-split cross-validation (=ShuffleSplit). Code snippets 12 and 13 show the parameter setup for the PC model, and its implementation for surrogate model development, respectively.



```
# PC parameters
polynomial_order = 3

cross_validator = LeaveOneOut()

regression_model = ElasticNetCV(
    fit_intercept=False,
    cv=cross_validator,
    l1_ratio=0.5,
    selection='random',
    random_state=666,
)

variables = {
    variable_class.name: variable_class()
    for variable_class in VortexPerturbedVariable.__subclasses__()
}

distribution = chaospy.J(
    *(
        variables[variable_name].chaospy_distribution()
        for variable_name in perturbations['variable'].values
    )
)
```

Code Snippet 12. PC parameters setup.

```python
# evaluate the surrogate for each KL sample
kl_training_set = xarray.DataArray(data=kl_expansion['samples'], dims=['run', 'mode'])
kl_surrogate_filename = output_dir / 'kl_surrogate.npy'

kl_surrogate_model = surrogate_from_training_set(
    training_set=kl_training_set,
    training_perturbations=training_perturbations,
    distribution=distribution,
    filename=kl_surrogate_filename,
    use_quadrature=False,
    polynomial_order=polynomial_order,
    regression_model=regression_model,
)
```

Code Snippet 13. Generating a surrogate model from the training set.

In order to evaluate the KL-PC surrogate model, the fit of polynomial surrogate to (27) KL eigenmodes for (30) training sets are evaluated as follows:

```python
# plot kl surrogate model versus training set
kl_fit = validations_from_surrogate(
    surrogate_model=kl_surrogate_model,
    training_set=kl_training_set,
    training_perturbations=training_perturbations,
    filename=output_dir / 'kl_surrogate_fit.nc',
)

plot_kl_surrogate_fit(
    kl_fit=kl_fit,
    output_filename=output_dir / 'kl_surrogate_fit.png'
)
```

Code Snippet 14. Validation of surrogate fit for training set.

The output of code snippet 14 is shown in Figure 11. Note that the correlation of some of the higher modes is zero since the regression penalizes inconsistent fits across the cross-validation set and chooses to return no prediction for that mode instead, resulting in a more parsimonious surrogate model.

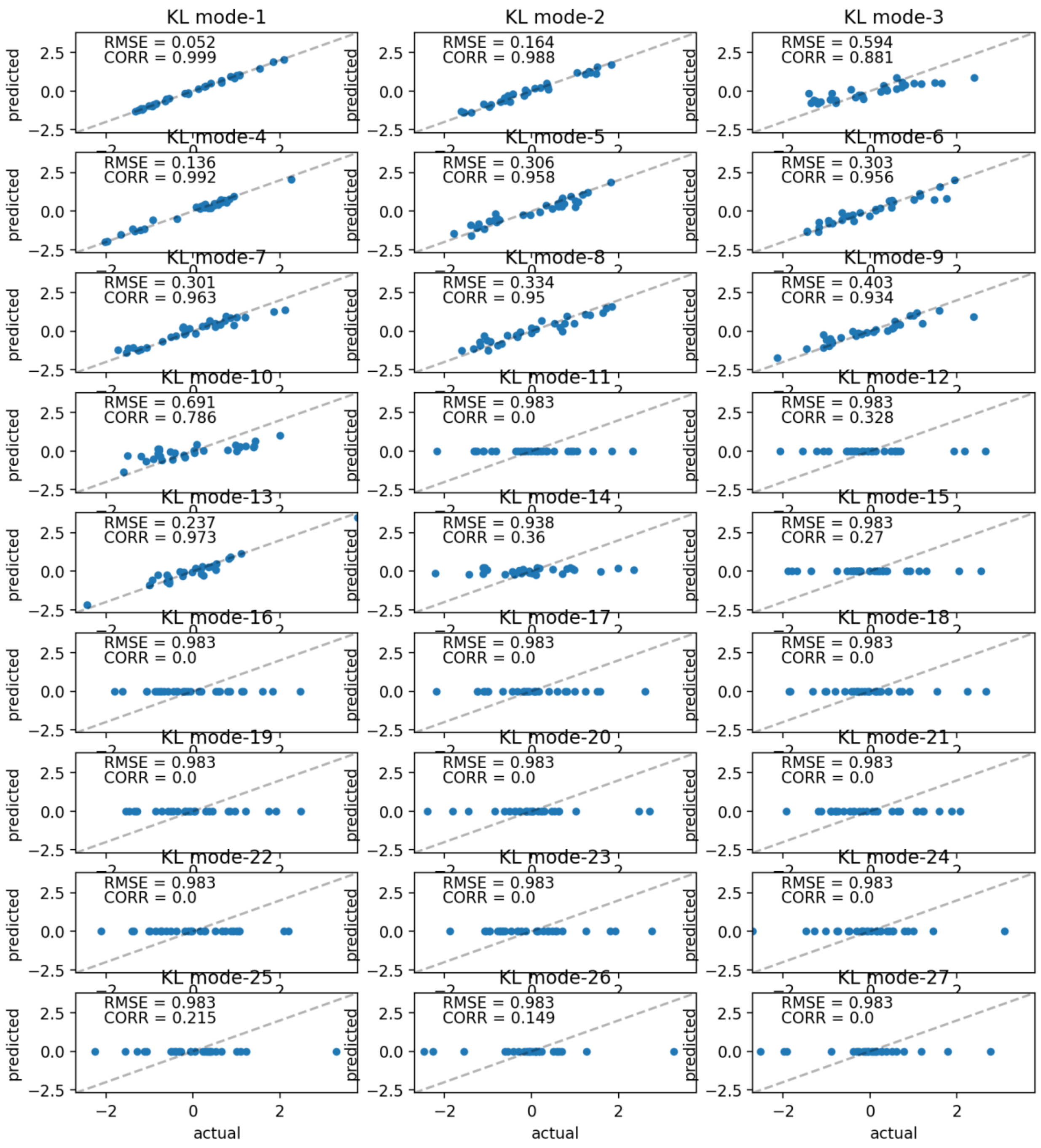


Figure 11. Comparison of surrogate for the KL samples.

Once the performance of the surrogate model for (27) KL eigenmodes has been evaluated, the final step for KL-PC surrogate model expansion involves transforming the surrogate from (27) KL eigenmodes to all 328,715 subsetted nodes, as shown in code snippet 15.

```python
# convert the KL surrogate model to the overall surrogate at each node
surrogate_model = surrogate_from_karhunen_loeve(
    mean_vector=kl_expansion['mean_vector'],
    eigenvalues=kl_expansion['eigenvalues'],
    modes=kl_expansion['modes'],
    kl_surrogate_model=kl_surrogate_model,
    filename=output_dir / 'surrogate.npy',
)
```

Code Snippet 15. KL-PC expansion to subsetted nodes.

Surrogate Model Evaluation

The surrogate model for all (328,715) subsetted nodes is validated against training and validation sets and used for sensitivity analysis and probabilistic predictions.

Validation

The validations_from_surrogate() function from the EnsemblePerturbation package (shown in code snippet 16) calculates statistics of the surrogate model performance for both training and validation sets. The plot_validations() function plots them as shown in Figure 12.

```python
# validation plots
min_depth = 0.8 * max_elevations.h0  # the minimum allowable depth
node_validation = validations_from_surrogate(
    surrogate_model=surrogate_model,
    training_set=training_set,
    training_perturbations=training_perturbations,
    validation_set=validation_set,
    validation_perturbations=validation_perturbations,
    convert_from_log_scale=False,
    convert_from_depths=True,
    minimum_allowable_value=min_depth,
    element_table=elements,
    filename=output_dir / 'validation.nc',
)

plot_validations(
    validation=node_validation,
    output_directory=output_dir,
)
```

Code Snippet 16. Validation of surrogate with ensemble runs.

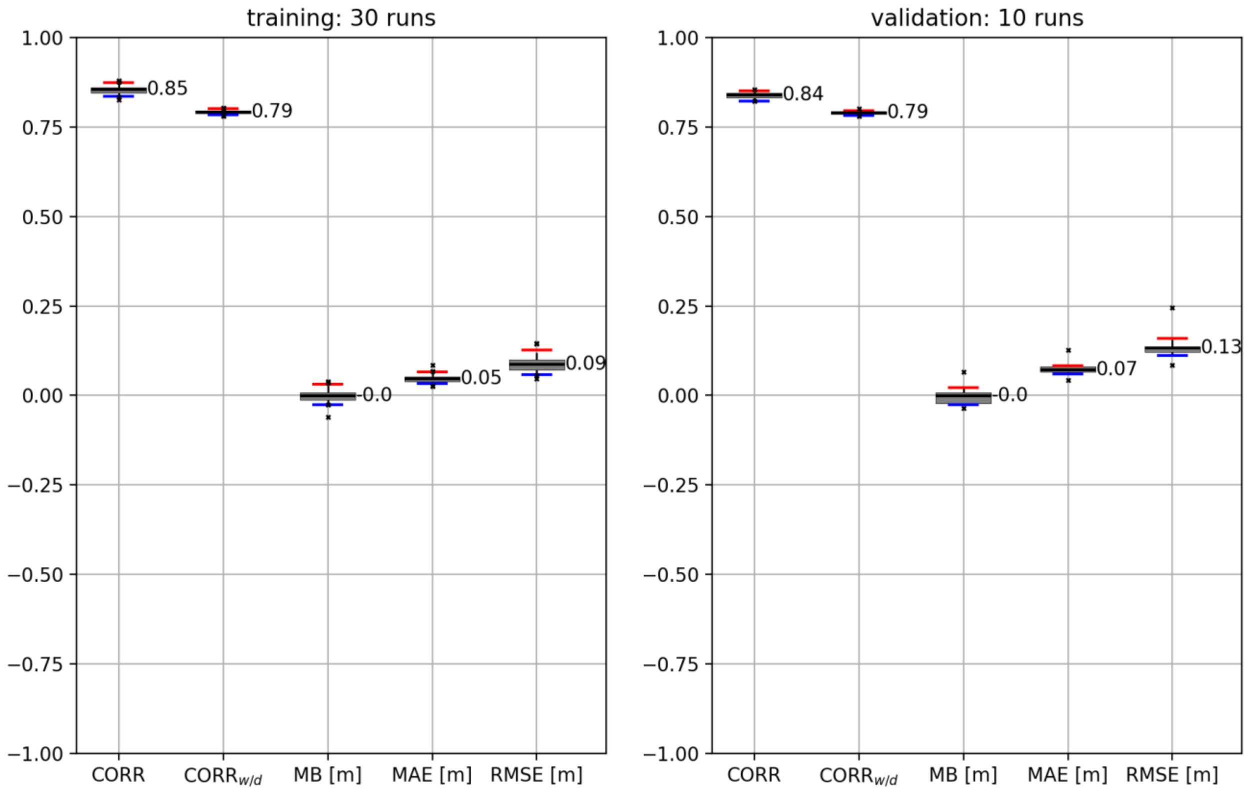


Figure 12. Error statistics of the surrogate model compared to the training (left), and validation (right) ensemble members. (CORR: correlation coefficient, CORRw/d: wet/dry correlation coefficient, MB: mean bias, MAE: mean absolute error, RMSE: root mean square error).

In addition to error statistics, another way to validate the surrogate model is the comparison of predicted water elevation from the surrogate model with water elevation from selected validation runs (code snippet 17) as shown in Figure 13.

```
plot_selected_validations(
    validation=node_validation,
    run_list=validation_set['run'][
        numpy.linspace(0, validation_set.shape[0], 6, endpoint=False).astype(int)
    ].values,
    output_directory=output_dir,
)
```

Code Snippet 17. Plot water elevation from surrogate and selected validation runs.

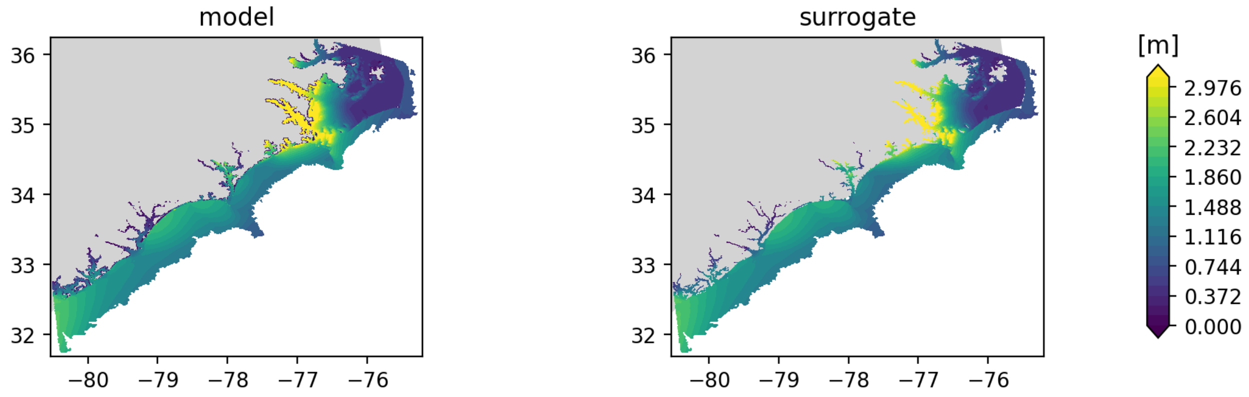


Figure 13. Validation of surrogate model for the random ensemble run No. 2. (in meters)

Sensitivity Analysis

The sensitivities_from_surrogate() function from the EnsemblePerturbation package calculates main and total sobol sensitivities (Sochala et al., 2020) of water elevation to the four perturbed parameters across the (328,715) subsetted nodes. Code snippet 18 shows how to extract sensitivities from surrogate and make sensitivity plots (shown in Figure 14). For the case of study of hurricane Florence (2018), these results showed that variation of hurricane cross track had the highest impact on maximum water elevation, followed by hurricane along track, while the radius of maximum winds had the lowest impact on maximum water elevation (Figure 14).

```
# extract sensitivities from surrogate and make sensitivity plots
sensitivities = sensitivities_from_surrogate(
    surrogate_model=surrogate_model,
    distribution=distribution,
    variables=perturbations['variable'],
    nodes=subset,
    element_table=elements,
    filename=output_dir / 'sensitivities.nc',
)

plot_sensitivities(
    sensitivities=sensitivities,
    storm=storm_name,
    output_filename=output_dir / 'sensitivities.png',
)
```

Code Snippet 18. Extract sensitivities from surrogate and plot them.

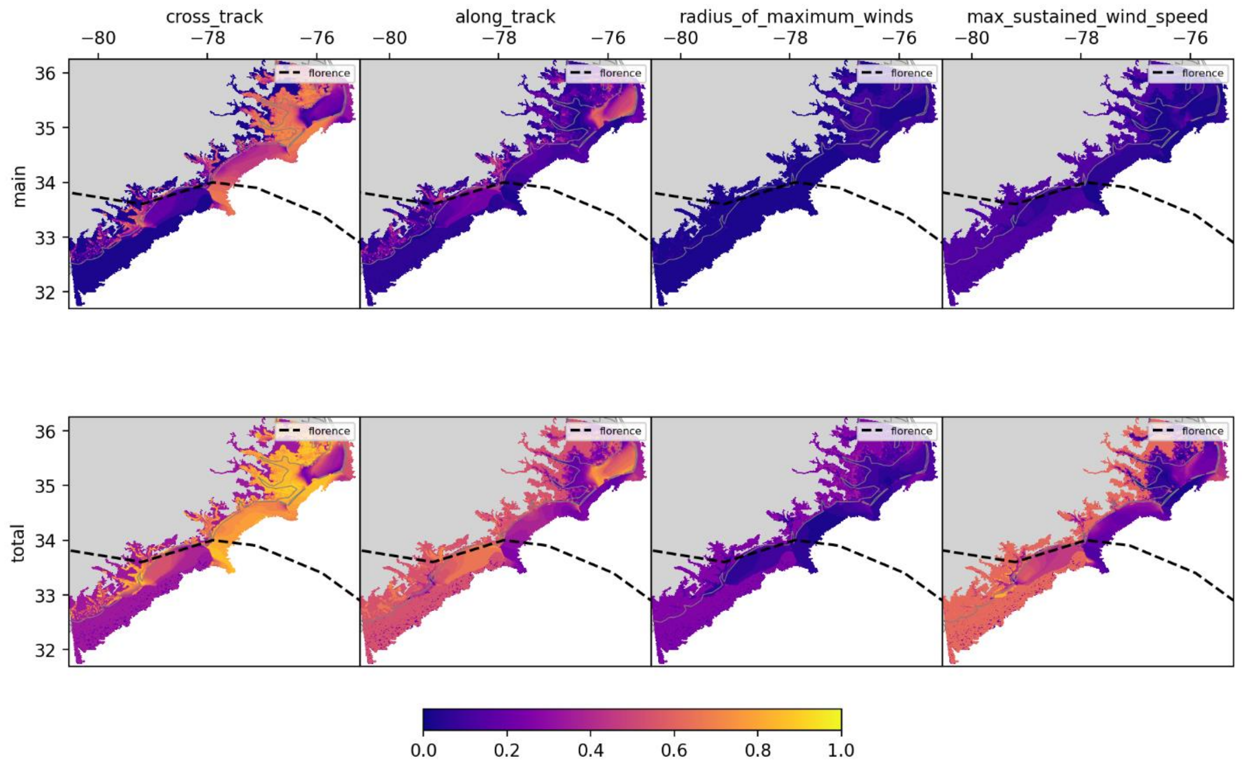


Figure 14. Sobol sensitivities of four variables along 328,715 nodes. Main sensitivity (top) quantifies the effect of each of four variables independently, while total sensitivity (bottom) measures the contribution of a variable based on its dependency to/interactions with the other variables. The colorbar (unitless) ranges from zero (no sensitivity) to one (the highest sensitivity).

Probabilistic Predictions

The EnsemblePerturbation package provides two types of probabilistic analysis of the surrogate model: 1) quantiles/percent exceedances (code snippet 19), and 2) probabilities of exceeding a given elevation (code snippet 20). Figures 15 and 16 show sample outputs for the 50th percentile and probability of exceeding 3ft (~1m), respectively.

```
# make_percentile_plot
percentiles = [10, 30, 50, 70, 90]
node_percentiles = percentiles_from_surrogate(
    surrogate_model=surrogate_model,
    distribution=distribution,
    training_set=validation_set,
    percentiles=percentiles,
    convert_from_log_scale=False,
    convert_from_depths=True,
    minimum_allowable_value=min_depth,
    element_table=elements,
    filename=output_dir / 'percentiles.nc',
)

plot_selected_percentiles(
    node_percentiles=node_percentiles,
    perc_list=percentiles,
    output_directory=output_dir,
)
```

Code Snippet 19. Calculate and plot five percentiles from the surrogate and ensemble model.

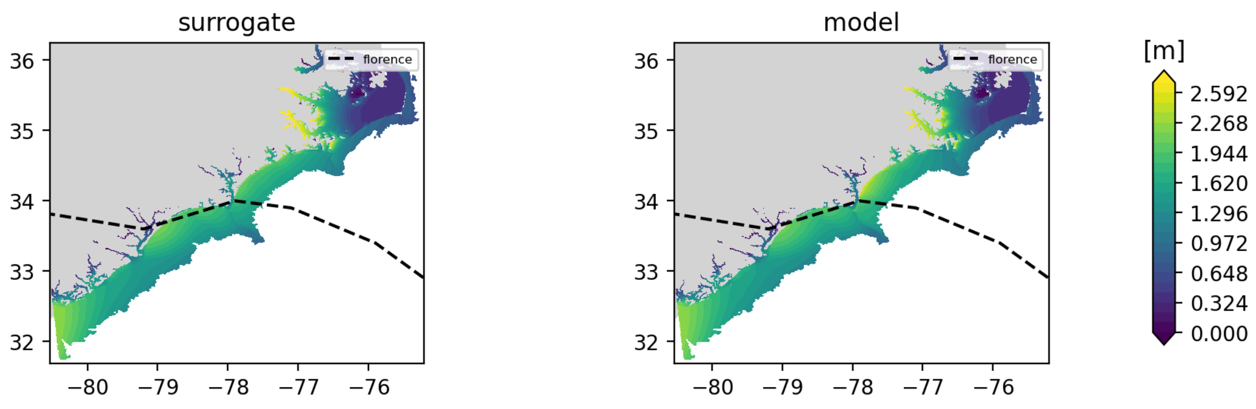


Figure 15. Comparison of 50th percentile of water elevation (in meters) from the surrogate (left) and validation set from the ensemble of elevations from hydrodynamic model (right).

```
# make plots for probability of exceeding a given elev.
level_ft = numpy.arange(1, 21)
level_m = (level_ft * 0.3048).round(decimals=4)

node_prob_field = probability_field_from_surrogate(
    levels=level_m,
    surrogate_model=surrogate_model,
    distribution=distribution,
    training_set=validation_set,
    minimum_allowable_value=min_depth,
    convert_from_log_scale=False,
    convert_from_depths=True,
    element_table=elements,
    filename=output_dir / 'probabilities.nc',
)

plot_selected_probability_fields(
    node_prob_field=node_prob_field,
    level_list=level_m,
    output_directory=output_dir,
    label_unit_convert_factor=1 / 0.3048,
    label_unit_name='ft',
)
```

Code Snippet 20. Calculate and plot probability of exceeding user defined levels from the surrogate and ensemble model.

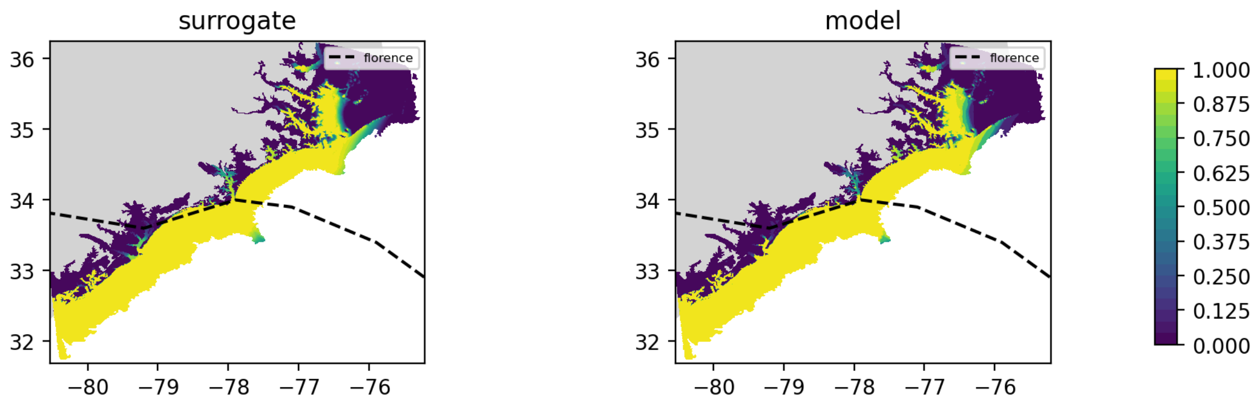


Figure 16. Probability of water level exceeding 3ft (~1m) from the surrogate (left) and validation set from the ensemble of elevations from the hydrodynamic model (right).

4. Summary

This document provides an overview of a Python script for probabilistic prediction of TC-driven coastal flooding. It showed how to use an ensemble of perturbed storm tracks and corresponding simulated water elevations, to develop a surrogate model for probabilistic analysis of coastal flooding. A KL-PC framework was used to generate the surrogate model for a training set and validated against an independent validation dataset. Lastly, it was shown how the developed probabilistic surrogate model can be used for global sensitivity analysis of water elevation to the perturbed variables. This framework provides probabilistic prediction of water elevation based on user-defined quantiles, and also the probability of water level exceeding given elevations across the domain. All functions used in this script are publicly available through the EnsemblePerturbation Python package on GitHub (EnsemblePerturbation, 2023).

Acknowledgement

The authors would like to thank Zachary Burnett for his significant role in development of EnsemblePerturbation and Dr. Khachik Sargsyan for his contribution to development of the KL-PC framework.

The results presented here are supported by funds from Disaster Relief Supplemental Appropriations Act, 2022 and from H.R.3684 Infrastructure Investment and Jobs Act (IIJA) projects among others.

References

Abdi, H. and Williams, L.J., 2010. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, *2*(4), pp.433-459.

Davis, J.R., Paramygin, V.A., Forrest, D. and Sheng, Y.P., 2010. Toward the probabilistic simulation of storm surge and inundation in a limited-resource environment. *Monthly Weather Review*, *138*(7), pp.2953-2974.

EnsemblePerturbation, 2023. URL: https://ensembleperturbation.readthedocs.io/en/latest/

Hashemi, M.R., Spaulding, M.L., Shaw, A., Farhadi, H. and Lewis, M., 2016. An efficient artificial intelligence model for prediction of tropical storm surge. *Natural Hazards*, *82*, pp.471-491.

Kim, S.W., Melby, J.A., Nadal-Caraballo, N.C. and Ratcliff, J., 2015. A time-dependent surrogate model for storm surge prediction based on an artificial neural network using high-fidelity synthetic hurricane modeling. *Natural Hazards*, *76*, pp.565-585.

Korobov, N. M., 1959: Approximate evaluation of repeated integrals. In *Dokl. Akad. Nauk SSSR* 124(6), pp. 1207–1210.

Luettich, R. A., and J. J. Westerink, 2004: Formulation and numerical implementation of the 2D/3D ADCIRC finite element model version 44.XX. *University of North Carolina at Chapel Hill & University of Notre Dame Tech. Rep*., 74 pp., https://adcirc.org/wp-content/uploads/sites/2255/2018/11/adcirc_theory_2004_12_08.pdf.

Marsooli, R., Lin, N., Emanuel, K. and Feng, K., 2019. Climate change exacerbates hurricane flood hazards along US Atlantic and Gulf Coasts in spatially varying patterns. *Nature communications*, *10*(1), p.3785.

National Weather Service, 2024. Hurricane Florence: September 14, 2018, URL: https://www.weather.gov/ilm/HurricaneFlorence (accessed on July 23, 2024)

NOAA COAST, 2024. Hurricane Costs, Office for Coastal Management, URL: https://coast.noaa.gov/states/fast-facts/hurricane-costs.html (accessed on May 9, 2024).

Plumlee, M., T. G. Asher, W. Chang, and M. V. Bilskie, 2021: High-fidelity hurricane surge forecasting using emulation and sequential experiments. *The Annals of Applied Statistics,* 15, pp. 460–480, https://doi.org/10.1214/20-AOAS1398.

Pringle, W.J., Burnett, Z., Sargsyan, K., Moghimi, S. and Myers, E., 2023. Efficient Probabilistic Prediction and Uncertainty Quantification of Tropical Cyclone–Driven Storm Tides and Inundation. *Artificial Intelligence for the Earth Systems*, *2*(2), p.e220040.

Scikit-learn, URL: https://scikit-learn.org/stable/index.html

Sochala, P., C. Chen, C. Dawson, and M. Iskandarani, 2020: A polynomial chaos framework for probabilistic predictions of storm surge events. *Computational Geoscience* 24, pp. 109–128, https://doi.org/10.1007/s10596-019-09898-5.

Zhang, Y., Ye, F., Stanev, E.V. and Grashorn, S., 2016. Seamless cross-scale modeling with SCHISM, Ocean Modelling, 102, 64-81.