The Thesis committee for Xianlong Hou

Certifies that this is the approved version of the following thesis:

Evaluating hydrodynamic uncertainty in oil spill modeling

APPROVED BY
SUPERVISING COMMITTEE:

Supervisor:_____

Ben R. Hodges

_____

Paola Passalacqua

Evaluating hydrodynamic uncertainty in oil spill modeling

by

Xianlong Hou, B.E.

Thesis

Presented to the Faculty of the Graduate School
of the University of Texas at Austin
in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science in Engineering

The University of Texas at Austin
May, 2013

Acknowledgements

I would like to express my very great appreciation to some individuals and institutions for their valuable help and constructive suggestions during this work. First, I would like to offer my special thanks to my research advisor, Dr. Ben R. Hodges, for his very patient guidance when I went into the wrong way and his enthusiastic encouragement when I made progress. My grateful thanks are also extended to the invaluable assistance of TWDB personnel, Dr. Dharhas Pothina, Dale Crockett, and Solomon Negusse who were always ready to help. I would also like to thank Dr. Joseph Zhang from Center for Coastal Resources Management at the Virginia Institute of Marine Science and Dr. Chris Barker at NOAA for their advice and assistance in keeping my research progress on schedule. Finally, I wish to acknowledge the help provided by the PhD student, Itay Rosenzweig, at Stanford University, Dr. Paola Passalacqua at UT-Austin, and all the EWRE faculty and staff of UT-Austin.

This material is based in part upon work supported by the Research and Development program of the Texas General Land Office Oil Spill Prevention and Response Division under Grant No. 10-097-000-3928 and in part by a grant from BP/The Gulf of Mexico Research Initiative through the Gulf Integrated Spill Research (GISR) consortium. Computer support was provided by the UT Center for Research in Water Resources.

Evaluating hydrodynamic uncertainty in oil spill modeling

by

Xianlong Hou, MSE

The University of Texas at Austin, 2013

SUPERVISOR: Ben R. Hodges

A new method is presented to provide automatic sequencing of multiple hydrodynamic models and automated analysis of model forecast uncertainty. A Hydrodynamic and oil spill model Python (HyosPy) wrapper was developed to run the hydrodynamic model, link with the oil spill, and visualize results. The HyosPy wrapper completes the following steps automatically: (1) downloads wind and tide data (nowcast, forecast and historical); (2) converts data to hydrodynamic model input; (3) initializes a sequence of hydrodynamic models starting at pre-defined intervals on a multi-processor workstation. Each model starts from the latest observed data, so that the multiple models provide a range of forecast hydrodynamics with different initial and boundary conditions reflecting different forecast horizons. As a simple testbed for integration strategies and visualization on Google Earth, a Runge-Kutta 4^{th} order (RK4) particle transport tracer routine is developed for oil spill transport. The model forecast uncertainty is estimated by the difference between forecasts in the sequenced model runs and quantified by using statistics measurements. The HyosPy integrated system with wind and tide force is demonstrated by introducing

an imaginary oil spill in Corpus Christi Bay. The results show that challenges in operational oil spill modeling can be met by leveraging existing models and web-visualization methods to provide tools for emergency managers.

Table of Contents

List of Figures

List of Tables

Chapter 1 - Introduction

1.1 Motivation

When an oil spill occurs at night in heavily trafficked shipping lanes, operational models become the key tool for estimating oil spill motion and deciding the initial equipment positioning as part of the emergency response. During an oil spill in coastal waters, the Office of Response and Restoration (OR&R) from the National Oceanic and Atmospheric Administration (NOAA), is responsible for providing scientific support to the U.S. Coast Guard officers who are in charge of emergency response operations (NOAA, 2013). To prepare communities for oil spills, OR&R develops and implements several operational models for response and planning. These include the General NOAA Operational Modeling Environment (GNOME), an oil spill trajectory forecasting model; Automated Data Inquiry for Oil Spills (ADIOS), an oil weathering model; and Environmental Response Management Application (ERMA), a GIS-based model that integrates key response data (NOAA, 2013). Effective oil spill response requires predicting possible oil spill movement. The oil spill transport model (GNOME) relies on forecasts of water surface currents that are typically provided by hydrodynamic models developed and maintained by local agencies. In Texas these are the General Land Office (TGLO) and the Texas Water Development Board (TWDB). The present TGLO/TWDB system requires TWDB/TGLO staff members to manually configure, run, and transfer data between models and visualization software, a system that could be made more efficient with the latest software tools for automatically coupling models.

A problem that has not been addressed in oil spill response is the effect of uncertainty in hydrodynamic modeling. Any prediction based on forecast data and modeling contains uncertainty that depends on the forecast time span and the models involved (Hodges and Hou, 2013). For example, hurricane and typhoon prediction maps typically have cone-shaped regions representing the probably future path for a storm. These cones are updated routinely based on the latest results from multiple models. The GNOME oil spill trajectory model provides methods to estimate uncertainty from forecasts for ocean currents and wind. Unfortunately, the existing hydrodynamic models do not provide any systematic approaches to evaluate the models' contributions to uncertainty.

To improve oil spill emergency response, this research addresses the two motivations above by: (1) developing improved methods for coupling weather/tidal forecasts, hydrodynamic models, oil spill models, and visualization, and (2) providing an approach from creating multiple hydrodynamic model runs as a basis for quantifying the hydrodynamic forecast uncertainty.

1.2 Research Objectives

The objectives of this research are to: (1) improve the model coupling method developed by Rosenzweig and Hodges (2011) into an operational framework suitable for implementation in the TGLO/TWDB Linux-Based computer system with Google Earth visualization (§3.9), and (2) develop an automatic system that provides a sequence of hydrodynamic forecast models running with different wind and tide hind-

casts/nowcasts/forecasts (historical data/current data/future data) with pre-defined intervals (§3.7). The difference between these old forecasts and the newer ones is used to quantify model error which is used to estimate how uncertainty evolves over time for new forecasts (§3.10).

When this advanced automatic system is installed and operational, TGLO/TWDB will have a better understanding of the capabilities and limitations of the combined hydrodynamic/oil spill transport response models and the forecast time horizon over which a prediction is believable.

1.3 Research Overview

This research presents a new model coupling system to automate multiple hydrodynamic model runs, oil spill modeling, visualization in Google Earth, and estimating hydrodynamic forecast uncertainty. This new model coupling system is a set of wrapper codes using the Python scripting computer language. In keeping with recent tradition in naming Python modules, the new model coupling system is known as *HyosPy*, which stands for Hydrodynamic and oil spill Python.

HyosPy is demonstrated with an automated sequencing of hydrodynamic models that enables 12 models (or more depending on the hardware configuration) to run at pre-defined intervals on a single multi-logical-processor workstation. HyosPy automatically downloads wind and tide nowcasts/forecasts from the web and transforms the data into input formats for each hydrodynamic model. Results from the different individual hydrodynamic models are used to drive separate runs of an oil

spill transport model so that multiple predictions of spill transport can be produced and visualized in Google Earth. The ensemble of results from these 12 modes is used to statistically evaluate the forecast uncertainty. Because NOAA is still working on developing a Linux version of the GNOME model that will be compatible with the TGLO/TWDB system, a simple oil spill transport model based on a 4^{th} order Runge-Kutta method (RK4 oil spill transport model) has been developed as a placeholder model that allowed development of the complete integration strategy and visualization. The structure of the HyosPy wrapper is shown in Fig. §1.3.1.

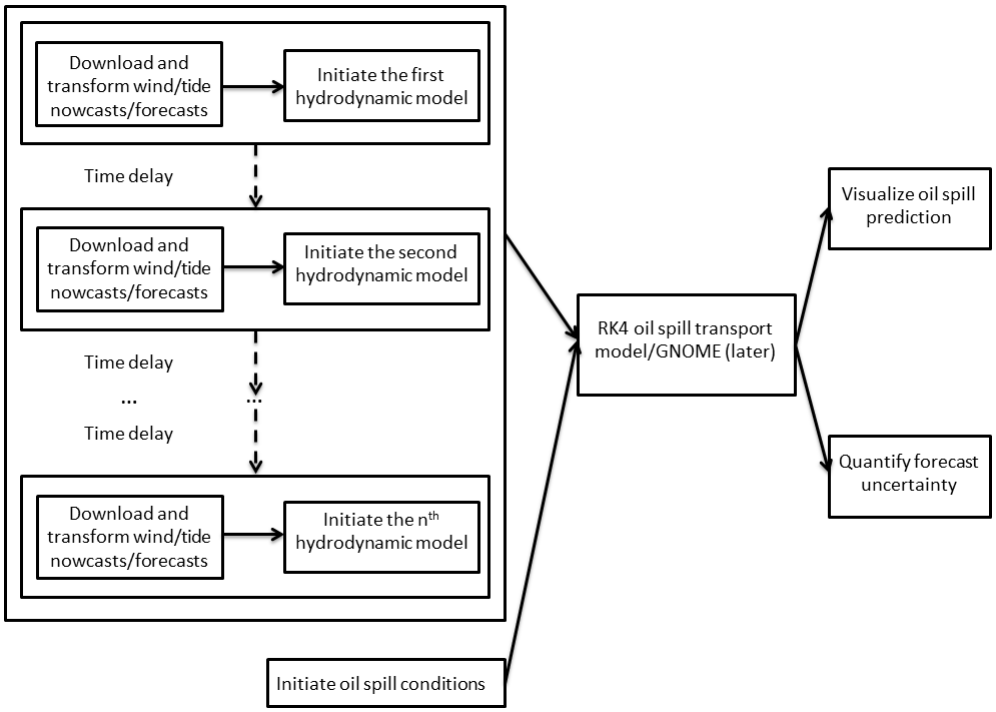

Figure 1.3.1: HyosPy wrapper structure (The big box on left represents a sequence of hydrodynamic models running with wind/tide force and pre-defined intervals marked as Time delay. The results of this model sequence along with the initial oil spill conditions are used to visualize oil spill predictions and quantify forecast uncertainty by using RK4 oil spill transport model.)

Chapter 2 - Background

2.1 Oil Spill Modeling

Advanced numerical oil spill modeling can provide emergency response managers with valuable information for risk assessment and contingency planning. To these ends, the transport and fate of oil spills has been studied using mass balance approaches and trajectory methods (e.g. Mackay et al., 1980; Huang, 1983; Spaulding, 2010; Shen and Yapa, 1988; Shen et al., 1986; Yapa et al., 1994). Some well-established oil spill models have been developed to predict oil transport movement and distribution in water body (Chao et al., 2003), such as GNOME (§2.2), OILMAP (ASA, 1997) and SINTEF (Reed, 2000).

In the last three decades, oil spill movement on water surface has been a significant research focus (Wang et al., 2007), resulting in two-dimensional (2D) oil spill models of advection and spreading (Nagheeby and Kolahdoozan, 2010; Chao et al., 2012, 2001; Inan and Balas, 1980). Advection is a physical process caused by the combined effects of winds, currents and waves. Advection for oil consists of two parts, which are surface oil advection and suspended oil advection in the water just below the surface. The surface oil advection is dominated by the forces of surface current and wind drag on oil, while the advection of suspended oil is the movement of oil droplets entrained in the water column due to water current (Guo et al., 2009). Spreading is generally simulated using the Fay hypothesis (Nagheeby and Kolahdoozan, 2010; Inan and Balas, 1980; Guo et al., 2009). According to Fay hypothesis, spreading, which dominates the surface oil transport at the beginning of the spill,

is caused by the horizontal expansion of the oil slick due to the counterbalance of mechanical forces including gravity, surface tension, inertia and viscosity (ASCE-Task-Committee, 1996; Shen and Yapa, 1988).

After an oil spill event, oil particles can stay in the water column for more than 4 days (Humphrey et al., 1987) degrading the environment. Recent research in three-dimensional (3D) oil spill modeling has included vertical movement and distribution of oil droplets (Chao et al., 2003; Lonin, 1999; Wang et al., 2007). Dispersion, rather than advection and spreading, dominates the oil particle movement in the vertical direction. A main objective of vertical oil dispersion research has been to estimate the rate of oil entrainment in the water column from the surface slick. Generally, vertical particle movement in three-dimensional oil spill modeling is simulated using the random walk technique (Chao et al., 2003; Lonin, 1999; Wang et al., 2007).

The research studies above are some of the foundations for operational oil spill modeling, which focuses on trajectory forecast simulation, probabilistic risk analysis, and information for making real-time responses (William et al., 2013) to minimize impacts and provide a net environmental benefit.

The primary task that should be undertaken when preparing to conduct real-time oil spill response operations is a comprehensive risk assessment and hazard analysis (IPIECA, 2002). Oil spill trajectory models, such as GNOME (§2.2), provide risk assessment, emergency response and contingency planning activities for the surface spills that often result from shipboard accidents and operations, and which comprise the majority of oil spills (Deborah et al., 1999).

2.2 The TGLO/TWDB Oil Spill System

The existing operational hydrodynamic model for TGLO/TWDB oil spill system is the two-dimensional (2D) TxBLEND model, which is used to simulate water velocities with wind and tide forces. Wind data driving the hydrodynamics are obtained from the Eta Model from the National Centers for Environmental Prediction (NCEP). Tide forecasts for TxBLEND use tidal harmonic constituents. Tide hindcasts use field observations from the Texas Coastal Ocean Observation Network (TCOON). The manual transformation and configuration of the TxBLEND output data to GNOME format is used to provide an oil spill trajectory prediction and visualization (Matsumoto, 1993; Rosenzweig and Hodges, 2011).

Hydrodynamic model

Presently, the hydrodynamic model used by TWDB as part of the TGLO oil spill modeling program is TxBLEND, a 2D (depth-averaged) finite-element model which simulates water currents by solving the continuity equation, momentum equations, and the advection-diffusion equation for conservation of salt. TxBLEND follows the generalized wave equation approach pioneered by Lynch and Gray (1979) that was developed into the 2D versions of the ADCIRC model (Blain and Rogers, 1998). TxBLEND has been under continuous development and application by TWDB for more than twenty years (TWDB, 2013). A primary disadvantage of TxBLEND for oil spill modeling is the 2D approach, which cannot capture developing wind-driven currents in the near-surface layer and therefore underestimates the hydrodynamic

contribution to oil spill advection.

To obtain improved estimation of surface currents, TWDB would like to use a newer 3D hydrodynamic model in the operational oil spill system. TWDB has been testing the Semi-Implicit Eulerian-Lagrangian Finite-Element (SELFE) hydrodynamic model for this purpose. SELFE is an open-source hydrodynamic modeling system which is based on unstructured grids and designed for the effective simulation of 3D baroclinic circulation across river-to-ocean scales. To solve the differential equation system, SELFE implements finite-element and finite-volume schemes with Eulerian-Lagrangian algorithm (Zhang and Baptista, 2008). No mode splitting is used in SELFE, thus eliminating the errors associated with the splitting between internal and external modes (Shchepetkin and McWilliams, 2005).

Oil spill trajectory model: GNOME

The GNOME oil spill trajectory model was developed by the OR&R at NOAA. As a nowcast/forecast model in pollution transport analyses, GNOME provides the capability of experimenting with oil spill behavior under different weather conditions (Cheng et al., 2011). GNOME utilizes the NOAA CATS (Current Analysis for Trajectory Simulation) model for dispersion. GNOME uses splots (also called Lagrangian/Eulerian (continuous) elements or LEs), which are collections of point representations that collectively indicate the extent of spilled oil (Beegle-Krause, 2005). Presently, NOAA provides GNOME for Windows and Mac OS X operating systems. Although the GNOME code base is designed to be portable, NOAA is still working on removing the platform-dependent code in a Linux version which is

expected to be released in 2013.

2.3 Python Programming

Python is a powerful dynamic programming language which can be used in a wide variety of application domains. Guido van Rossum created Python in the early1990s. To ensure a clear, easy-to-use language, van Rossum only used ideas that had proven their worth over time in other computer programming languages (Lindstrom, 2005). Although running Python code is usually 10X slower than C language, writing Python code is much easier and faster. In most other programming languages, a significant amount of code is required just to prepare for implementing an algorithm. However, in Python, it is not necessary to define variable types, create iterators or other support objects, break the code into particular files or produce other supporting code, such as the headers used in C++ and Java (Lindstrom, 2005).

2.4 GNOME/hydrodynamic model coupling

Previous work supported by TGLO under this contract (10-097-000-3928) demonstrated that both TxBLEND and SELFE could be coupled with GNOME through a Python wrapper (Rosenzweig and Hodges, 2011). The wrapper was designed to reduce the manual effort associated with transforming the hydrodynamic models output into the correct format for GNOME and pre-processing the model configuration. The wrapper was tested by a series of simulations of SELFE and TxBLEND

coupled with GNOME for a hypothetical spill in Galveston Bay. The prior project demonstrated that (1) a wrapper coupling approach is practical, and (2) Python is an appropriate scripting language for handling the data transformation and model configuration. It was recognized from the outset that the wrapper developed in Rosenzweig and Hodges (2011) could not be directly implemented in TWDB computers with a Linux version of GNOME.

Chapter 3 - Methodology

3.1 Overview

This study builds upon the previous Python wrapper (Rosenzweig and Hodges, 2011) with a completely automated code (HyosPy) for configuring hydrodynamic input files, running multiple hydrodynamic models at pre-defined time intervals using nowcasts/forecasts wind and tide data, computing drifter tracks with an RK4 advection algorithm, visualizing different oil spill predictions in Google Earth, and quantifying the forecast uncertainty of an ensemble of results from multiple models. HyosPy has been developed and tested for the Linux operating system that TWDB uses on computers for hydrodynamic modeling. Until a Linux version of GNOME becomes available, HyosPy cannot be directly tied to GNOME as was done in the previous Python wrapper (Rosenzweig and Hodges, 2011). However, a new RK4 transport algorithm provides a functional placeholder for testing the integration of a complete system. The present research has exclusively used the SELFE hydrodynamic model to be compatible with the future operational needs of TGLO and TWDB.

In this research, the Python programming language (§2.3) is used as a master programming tool to automatically implement each step in Fig. §1.3.1, and integrate them as a whole system. Because each step in Fig. §1.3.1 is programmed by a corresponding Python module that is independent among other modules, the system can be extended to a broader multi-function platform without any re-programming work but just by making some connecting changes in the code. Additionally, Python

can call and edit other programming language module with ease, thus, visualization via Google Earth can be accomplished by calling and editing an existing JavaScript module.

3.2 Study Region

The study region is Corpus Christi Bay (Fig. §3.2.1) which is a shallow embayment in the Texas Coastal Bend region that episodically experiences inverse estuarine conditions (Hodges et al., 2011). The climate is semi-arid and the annual rainfall varies from 25 to 38 inches. Generally, winters are mild with occasional freezes while summers are humid and hot. Tropical storms and hurricanes periodically affect the region (Ward, 1997).

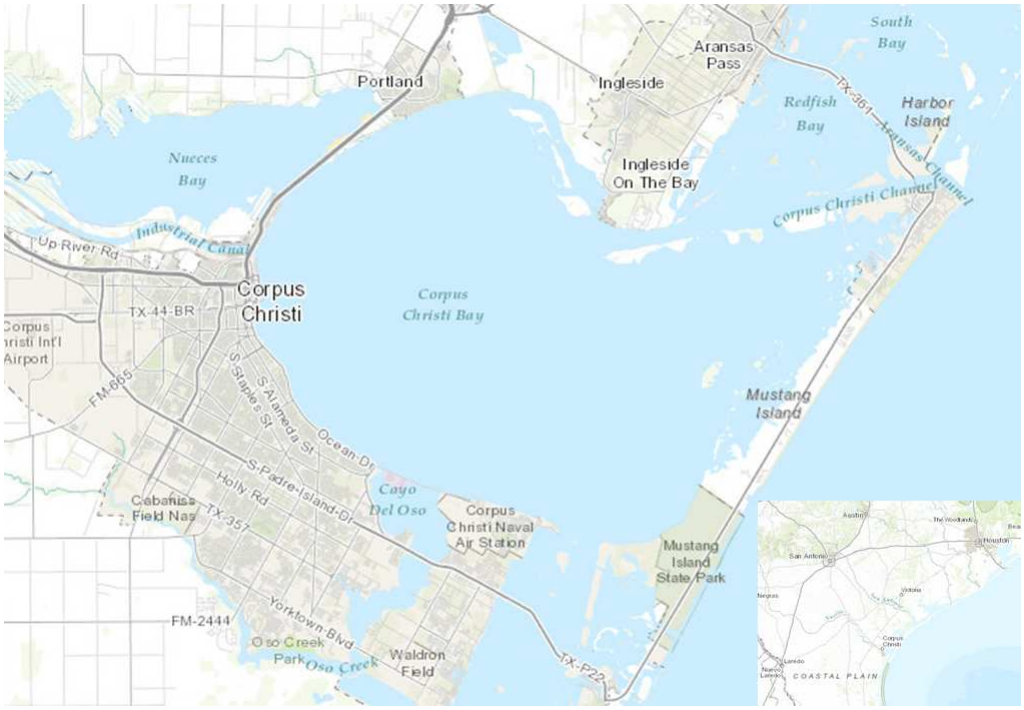

Figure 3.2.1: Corpus Christi Bay

3.3 SELFE Setup

The setup process for MPI SELFE hydrodynamic model implemented with HyosPy (§3.7) requires three steps; the order of the steps is not important since each step is independent with each other. Firstly, The SELFE hydrodynamic model requires four types of mandatory input files containing: (1) model grid with bathymetry elevations, (2) boundary condition and tidal information, (3) parameter input, and (4) interpolation mode. For this study, the SELFE grid and boundary data of Corpus Christi Bay were obtained from TWDB (§3.4). The parameter input file contains all of the necessary model running mechanism information, such as total running period, advection on/off switch, implicitness parameter, and global output options (i.e. parameters to set which output will be recorded). The interpolation mode file specifies

the approach used for vertical interpolation (SELFE-developers, 2013b). Second, to implement SELFE with HyosPy (§3.7) using wind and tide forces, wind and tide hindcasts/nowcasts/forecasts files are also required. Downloading and transforming wind and tide real-time nowcast/forecast data are discussed in §3.5 and §3.6, respectively. In addition, the SELFE source code needs to be compiled before running. In this research, the SELFE source code was compiled with the Intel Fortran compiler, which is recommended by the SELFE developers. When a SELFE simulation is completed, the binary outputs need to be combined with the post-processing tools (SELFE-developers, 2013a) to visualize and integrate with the oil spill models.

3.4 Model grid

The SELFE grid domain used herein extends from Aransas Bay to the north to just below Baffin Bay to the south (Fig. §3.4.1), which includes a small part of the Gulf of Mexico (GoM), Nueces Bay, Corpus Christi Bay. The SELFE domain is generally very shallow, 5m or less (Zhang, 2010), and thus the effects of wind and tide are expected to play a significant role in surface currents which directly affect surface oil spills. Fig. §3.4.2 shows the basic SELFE grid for Corpus Christi Bay.

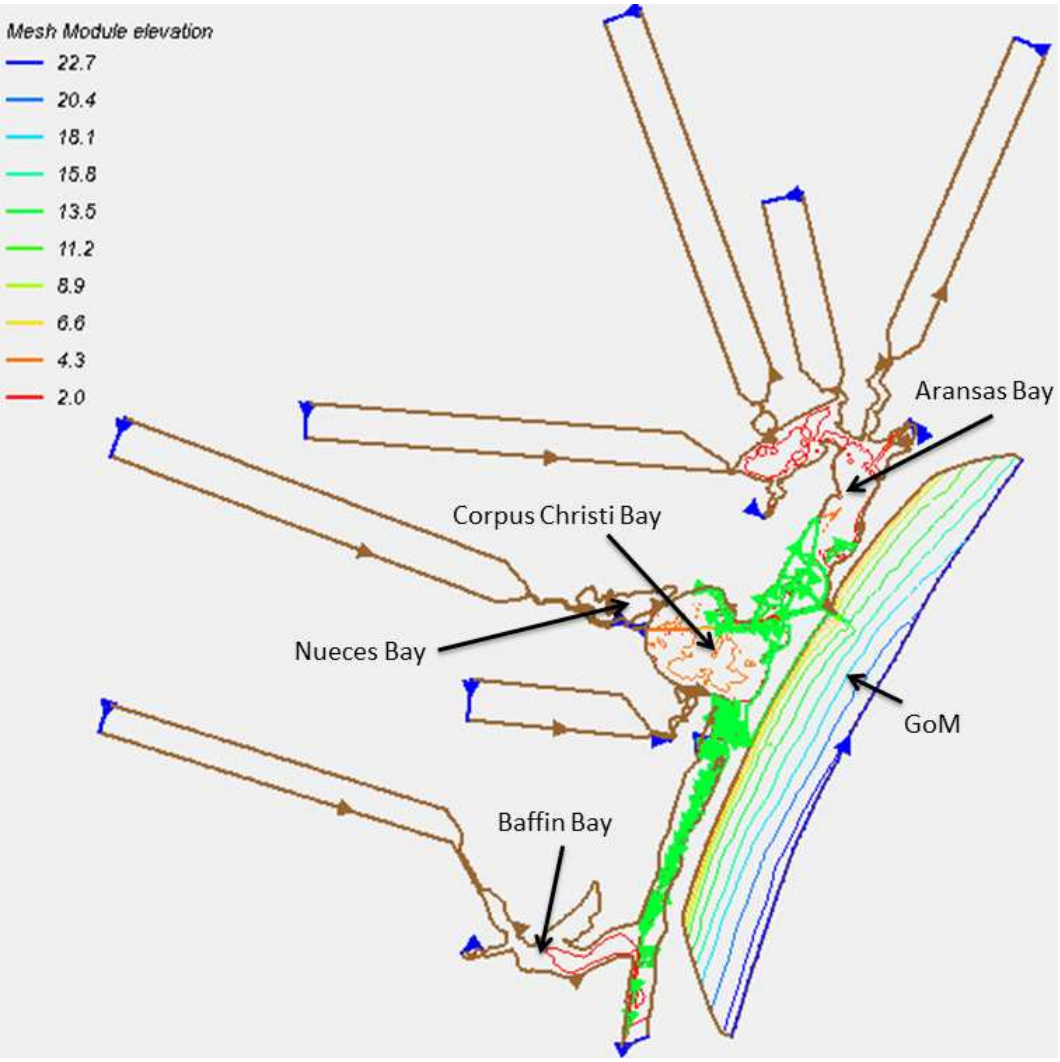

Figure 3.4.1: SELFE grid domain outline. The large rectangular sections are dummy domains for river inflows that are used to reduce boundary effects associated with rapid changes in river inflow rates.

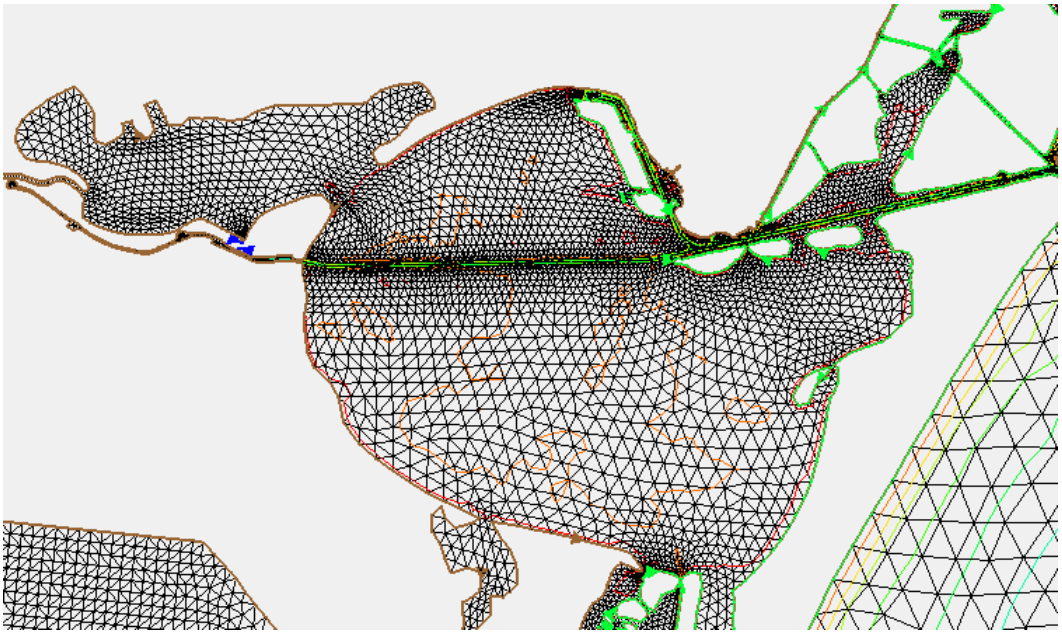

Figure 3.4.2: SELFE grid for Corpus Christi Bay

3.5 Wind nowcasts/forecasts/hindcasts

HyosPy is configured to download and translate wind nowcasts/forecasts/hindcasts data from a server hosted by Texas A&M University (TAMU) (§5.2). This server collects wind data from 341 observation sites along the west part of GoM (Fig. §3.5.1) and provides 3-hourly historical wind data, wind nowcasts, and 4-day wind forecasts for all observation sites in GMT (UTC) local time. The TAMU server updates the wind forecast data for every site every 3 hours. The raw data is downloaded in a tar format which contains individual ASCII text files for each wind site. Each wind data file has its station number along with latitude and longitude location in the header. For this research, wind nowcasts/forecasts from site 51 (i.e. the observation site for Corpus Christi Bay) are used.

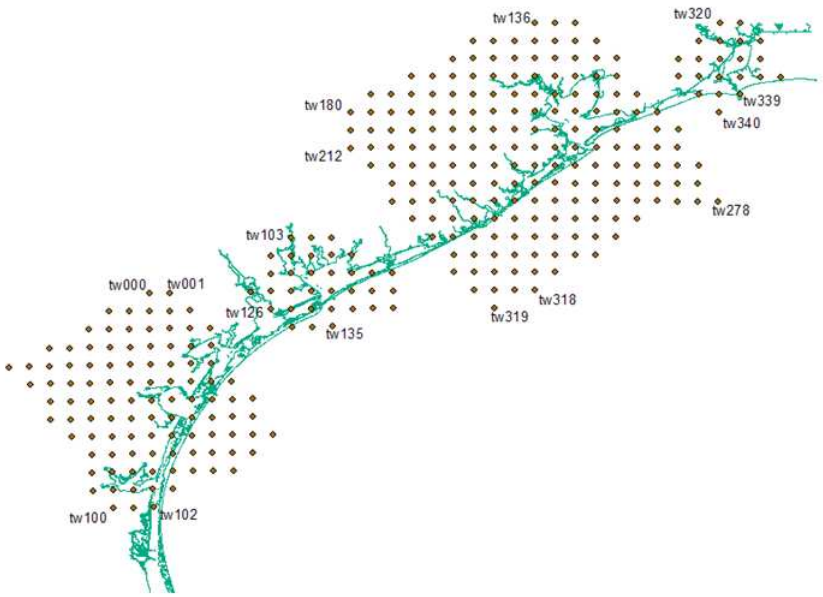

Figure 3.5.1: Distribution of wind sites. Figure provided by D. Robertson, TGLO.

For each wind site, the wind data format from TAMU is shown in Table §3.5.1:

Year	Month	Day	Hour	$U(mi/hr)$	θ
2013	01	01	00	6.4	165.8
2013	01	01	03	17.9	148.4
...	...	...	...	...	...

Table 3.5.1: Wind data format from the TAMU server (Wind direction (θ) is recorded as degrees using the NWS convention where 0 means wind from the north.)

The wind speed and direction data from the TAMU server are transformed into SELFE wind velocity vectors of u and v using (SELFE-developers, 2013b):

$$u = 0.447U \sin\left(\theta \frac{2\pi}{360}\right) \tag{1}$$

$$v = -0.447U \cos\left(\theta \frac{2\pi}{360}\right) \tag{2}$$

where u and v represent the value of wind speed (m/s) in easterly and northerly directions, respectively, such that negative values are westerly and southerly vectors; the constant 0.447 is a unit conversion from mi/hr to m/s.

3.6 Tide nowcasts/forecasts/hindcasts

HyosPy obtains tide nowcasts/forecasts/hindcast from the Texas Coastal Ocean Observation Network (TCOON) hosted by Texas A&M University Corpus Christi. TCOON is a network of scientific data collection platforms that records, maintains, and distributes wind and water data along the Texas Coast. The network currently consists of 37 data collection stations from South Padre Island, Texas to the Texas Louisiana border on the Sabine River (TCOON, 2013). TCOON provides measured tidal elevations for hindcast and nowcast with tidal predctions using harmonic analysis. The tide data can be downloaded with elevation relative to either the station datum or several standard datums (e.g. NAVD88). In this research, the tide data is downloaded with the datum of mean sea level (MSL) to be compatible with SELFE elevation datum of Corpus Christi Bay. Fig. §3.6.1 shows the relationship between the measured and harmonic tides for Bob Hall Pier (sited on the Gulf of Mexico

outside of Corpus Christi Bay) over five days. The data set was downloaded on 4/20/2013.

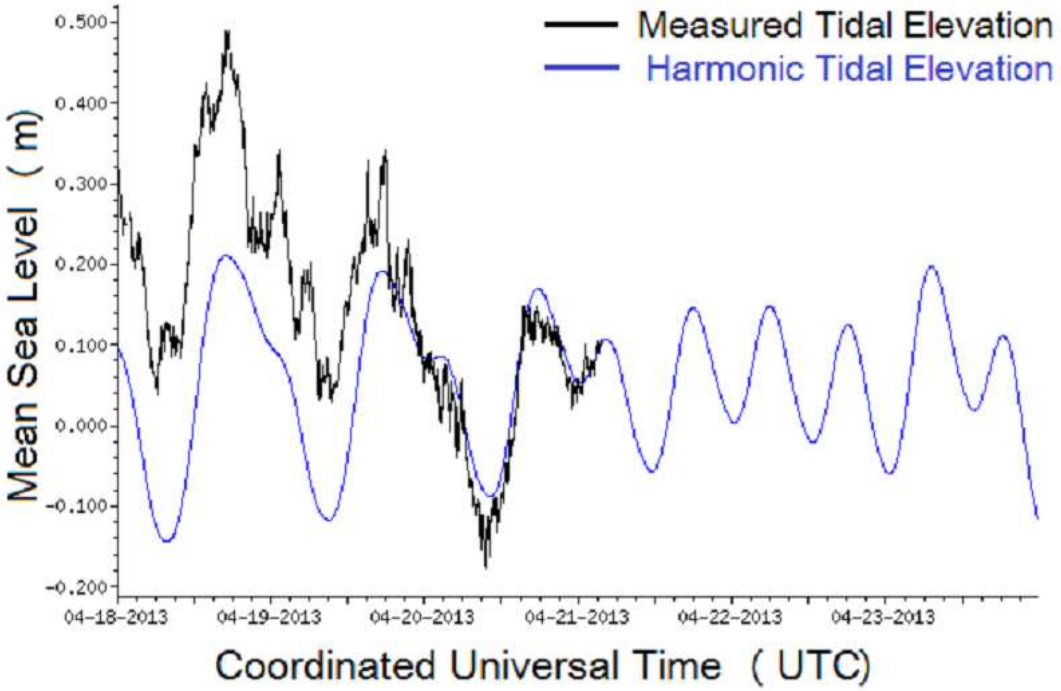

Figure 3.6.1: Typical measured and predicted tidal elevations (TCOON, 2013)

The measured and the predicted tides do not align when transition from past to future, typically because of the effects of wind and circulation in the Gulf of Mexico and storm tides driven by atmospheric pressure effects. An unrealistic sudden jump in the tidal elevation would occur if SELFE were run from a hindcast through to a forecast by simply switching from the measured tide to the predicted harmonic. Thus, to provide a smoother transition of tide series from past to future is required using:

$$Z_P^m = Z_H^m + \frac{\sum_{i=0}^{n}(Z_M^i - Z_H^i)}{n} \qquad : \qquad m = 1, 2, 3... \tag{3}$$

where Z_P^m is the predicted tidal elevation at the m^{th} future time interval, Z_H^m is the harmonic tide at the the m^{th} future time interval, Z_M^i is the measured tidal elevation at the i^{th} past time interval, Z_H^i is the harmonic tidal elevation at the i^{th} past time interval, and n is the number of past time intervals used for the computation. The parameter n (the historical tidal time series) needs to be long enough to remove high-frequency changes from the measured data, but also small enough to guarantee an appropriate jump from the harmonic data. Judging from Fig. §3.6.1, n is set to be 240 (24 hours with data recorded every 6 minutes). Fig. §3.6.2 shows the generated tide prediction (yellow curve) based on the harmonic tidal elevation. The tide hindcast/nowcast/forecast data can be transformed in SELFE input format (SELFE-developers, 2013b) in an ASCII text file using Python (§5.2).

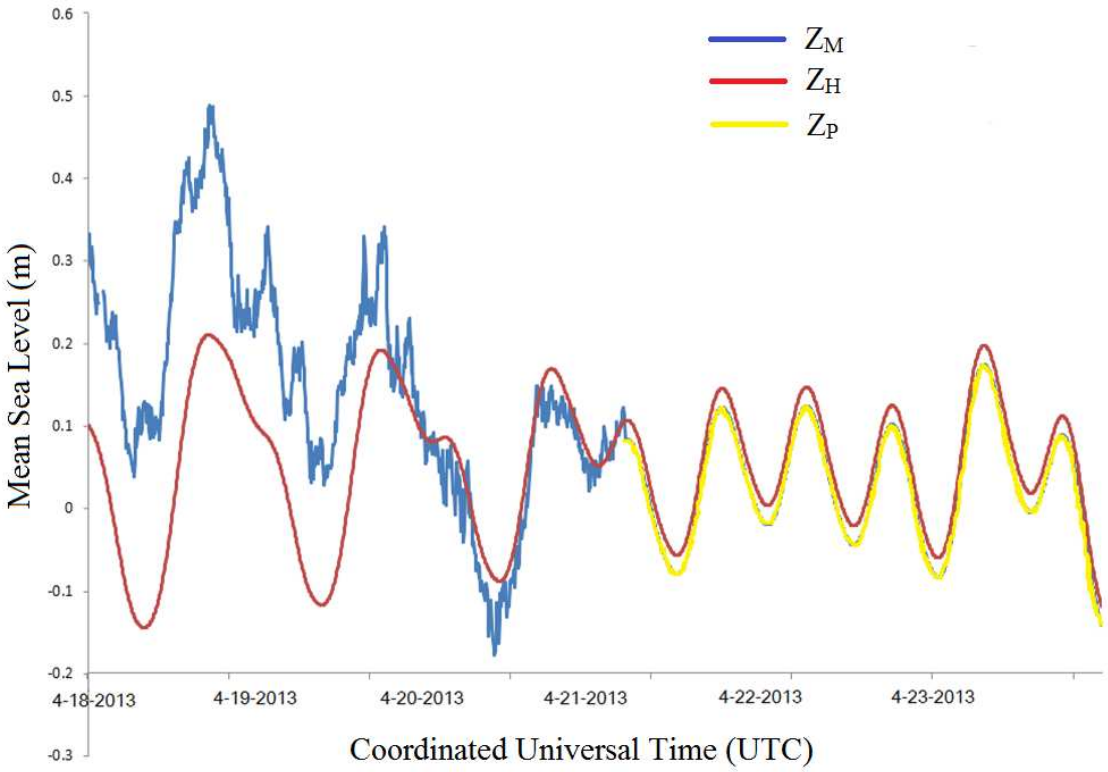

Figure 3.6.2: Tide prediction

3.7 HyosPy automation

HyosPy builds upon and extends the Python wrapper developed in the prior TGLO project (Rosenzweig and Hodges, 2011). When installed with SELFE on a multiple logical-processor workstation, HyosPy can be used to automatically run a sequence of SELFE hydrodynamic models at pre-defined time intervals, with each model using the latest wind and tide nowcasts/forecasts. HyosPy automates the entire pre-processing (i.e. wind and tide hindcasts/forecasts downloading and configuration), the multiple models running with pre-determined intervals, and the post-

processing work (i.e. combining binary outputs for oil spill integration). To be specific, HyosPy does the following steps automatically (Hodges and Hou, 2013):

a. Download wind and tide data (§5.2, §5.2).

b. Convert data to the hydrodynamic model input format (§5.2, §5.2).

c. Initialize the hydrodynamic model and start simulation on a single processor (§5.2).

d. Wait a predetermined time interval and obtain new forecast, nowcast and historical data (§5.2).

e. Start a new simulation on another processor using initial conditions based on the latest forecast/nowcast/hindcast data (§5.2).

f. Repeat steps d and e.

g. Combine the binary outputs for analysis (§3.3).

The approach used for automating the hydrodynamic model sequence is illustrated in Fig. §3.7.1. Before initiating the HyosPy system on a multiple logical-processor workstation, the time interval between simulation starts is defined as ΔT_{start}, and the simulation time duration as T_{total}. Thereafter, up to real-time = 0, HyosPy automatically downloads and transforms the real-time wind and tide nowcasts/forecasts/hindcasts, with which the first SELFE model (the first box with 0 hindcast and T_{total} forecasts in Fig. §3.7.1) runs on the first logical-processor. The process is automated and seamless. At real-time = 0, no hindcast data are used, but T_{total} forecasts are available so that the initial SELFE run is in a completely forecast

mode. After pre-defined interval ΔT_{start}, new wind/tide observations become available. To make use of this data, the HyosPy automatically downloads and transforms another set of real-time wind and tide hindcasts/forecasts (up to real time = ΔT_{start}), and the second SELFE model is started on the second logical-processor with ΔT_{start} hindcasts and (T_{total} - ΔT_{start}) forecasts.

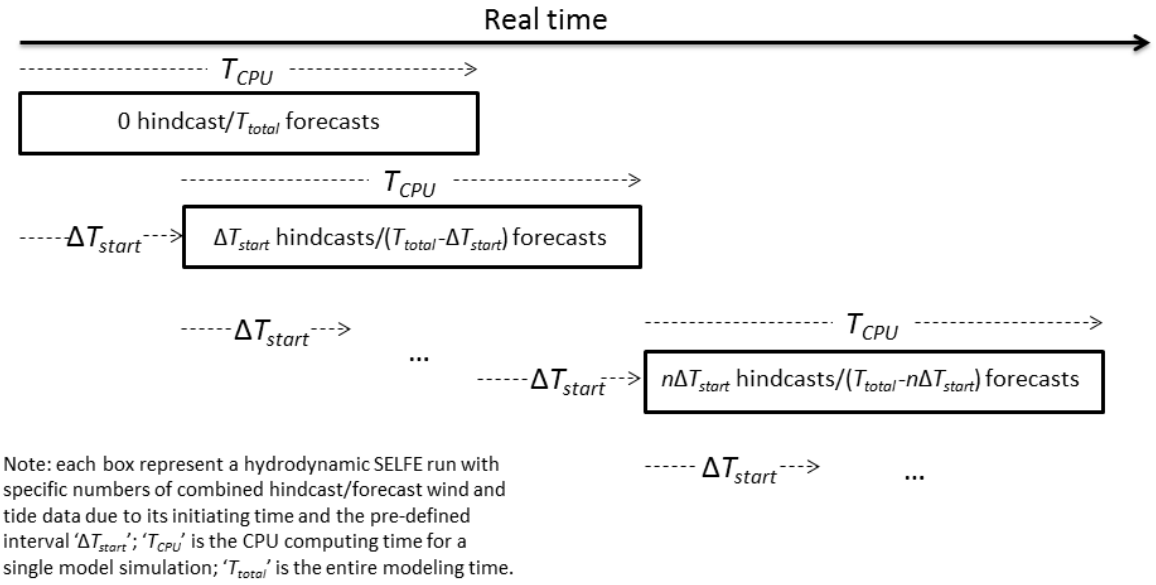

Figure 3.7.1: Hyospy Mechanism

Since the first SELFE model used forecast data from 0 to ΔT_{start}, i.e. the time when the second model begins, the first SELFE model will likely diverge from the second model that uses observed data during the same time interval. As the two models continue forward in time, the difference between them provides insight into the uncertainty associated with the initial forecast data. As new wind/tide observed data becomes available after another pre-defined interval ΔT_{start}, the third SELFE is

started using the last available wind/tide observed data. This sequence is continued until all SELFE are running a different simulation based on different latest available wind and tide hindcast/forecast data. Given that one SELFE requires T_{CPU} time to finish running, at any time the number of models running on the multiple logical-processor workstation depends on the amount of overlap (Fig. §3.7.1).

A limitation of the present approach is that each model starts from the same initial conditions, which means that the third model will complete exactly the same results from 0 to ΔT_{start} as the second model. This is a temporary approach designed for testing the multiple model sequence running with pre-defined time intervals. Future work requires transforming the prior models outputs to create the next model's initial conditions so that redundant computations are eliminated.

3.8 RK4 Surface Tracer Transport Routine Mechanism

Hydrodynamic models are inherently mechanistic, translating tidal and wind forcing into time and space-varying velocities which are recorded as part of the model output. Surface velocities from a hydrodynamic model can be used as input files for oil spill modeling. NOAA is presently rewriting the GNOME oil spill model into a Linux kernel that can be integrated with HyosPy in the future. In the interim, a Runge-Kutta 4^{th} order method particle transport routine (RK4 tracer routine) was developed to use as a simple testbed for integration and visualization strategies (Hodges and Hou, 2013). This transport method can also be used as a basis for drifter modeling, which may be valuable when developed into web-based tool for

the public to use in visualizing flow paths. The RK4 tracer routine was designed to compute the transport of multiple particles spread over a range of starting positions. The mechanism of this tracer routine is illustrated in Fig. §3.8.1.

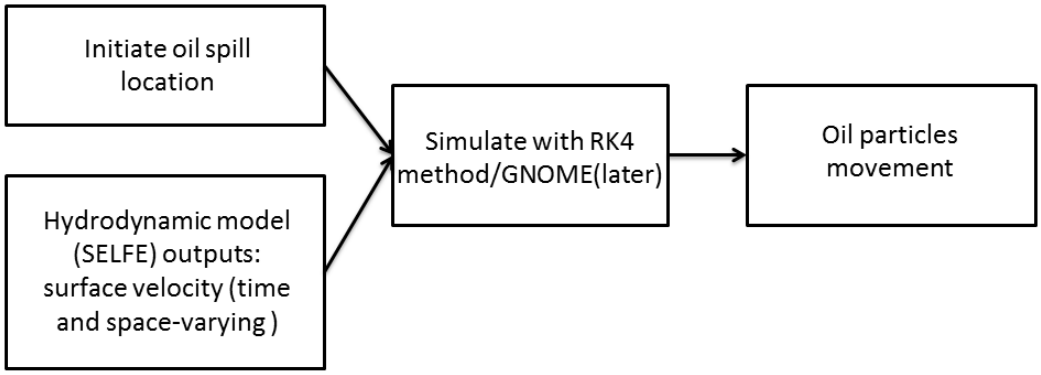

Figure 3.8.1: RK4 Surface Tracer Transport Routine Mechanism

An imaginary oil spill can be introduced at any location represented by a number of particles distributed about a center location. Along with the time and space-varying surface velocity (part of SELFE outputs), the movement of oil particles can be calculated by implementing a Runge-Kutta 4^{th} order method (RK4) (Al-Khafaji and Tooley, 1986). In this RK4 transport method, the movement of oil particles is recorded as a series of Universal Transverse Mercator (UTM) coordinates which uses a two-dimensional Cartesian coordinate system to give locations on the surface of the Earth. The RK4 tracer routine can be integrated with HyosPy (§3.7) at any given time by just reading part of the hydrodynamic model sequence which is still running in progress (§5.2).

The time $i+1$ positions (x, y) of a particle is computed as:

$$x_{i+1} = x_i + \frac{1}{6}\left(k_{1x} + 2k_{2x} + 2k_{3x} + k_{4x}\right) \tag{4}$$

$$y_{i+1} = y_i + \frac{1}{6}\left(k_{1y} + 2k_{2y} + 2k_{3y} + k_{4y}\right) \tag{5}$$

where the k functions are of the form:

$$k_{1x} = \Delta t\, u\left(t_i, x_i, y_i, \right) \tag{6}$$

$$k_{2x} = \Delta t\, u\left(t_i + \frac{\Delta t}{2}, x_i + \frac{k_{1x}}{2}, y_i + \frac{k_{1y}}{2}\right) \tag{7}$$

$$k_{3x} = \Delta t\, u\left(t_i + \frac{\Delta t}{2}, x_i + \frac{k_{2x}}{2}, y_i + \frac{k_{2y}}{2}\right) \tag{8}$$

$$k_{4x} = \Delta t\, u\left(t_i + \Delta t, x_i + k_{3x}, y_i + k_{3y}\right) \tag{9}$$

where Δt is the time interval used for the forward time-marching integration, and $u(t, x, y)$ is the x velocity evaluated at time t and position (x, y) from the hydrodynamic model velocity data. The k_y values are similar, but using the velocity in the y direction, $v(t, x, y)$, instead of $u(t, x, y)$. The Δt used in the RK4 is the same as the model time step in SELFE.

3.9 Visualizing particles on Google Earth

To visualize different particle positions through time on Google Earth, the UTM coordinates calculated by the RK4 tracer routine requires converting to Latitude and Longitude (§5.2) (IBM-developers, 2013) since Google Earth does not provide tools

to read UTM coordinates. Therefore, these converted Latitudes and Longitudes can be copied to a Google Map Marker animation JavaScript module (§5.2) (Google-developers, 2013) using Python. This Google Marker animation JavaScript module is a Google Map API that can mark multiple points with Latitude and Longitude on Google Earth and store as an html file format. Thus, visualizing oil spill particles on Google Earth is available by opening a Google Earth html file with all the particle positions marked. The web-visualization process is shown as Fig. §3.9.1:

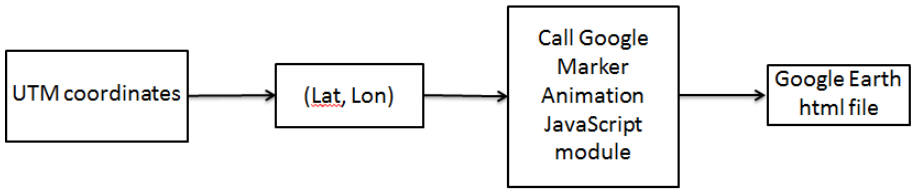

Figure 3.9.1: Web-visualization process

3.10 Evaluating hydrodynamic uncertainty

The combination of HyosPy and the RK4 tracer routine provides multiple forecasts of oil spill motion through time. Each spill forecasts contains different hydrodynamic forecast uncertainty due to the different wind and tide data used as boundary conditions. A simple uncertainty metric is the distance between the predicted particle positions at each time step for the different simulations.

To calculate the distance, the position of the i^{th} particle at the k^{th} time step for the m^{th} simulation is defined as $(x^k_{i:m}, y^k_{i:m})$. Thus, the distance between a particle in the m^{th} model and its companion in the n^{th} model at the k time step, $L^k_{i:m,n}$, can

be determined by:

$$L^k_{i:m,n} = \sqrt{\left(x^k_{i:m} - x^k_{i:n}\right)^2 + \left(y^k_{i:m} - y^k_{i:n}\right)^2} \tag{10}$$

The uncertainty can be estimated by either the mean distance or root-mean-square distance between corresponding particles compared across every simulation at a single time step based on the assumption that all models are equally likely. Let N_P represent the number of particles representing a spill with N_T as the number of time steps and N_S as the number of simulations that are sequenced by HyosPy. The distance is computed between the i^{th} particle in the m^{th} simulation and all the other $N_s - 1$ simulations for estimating uncertainty. The mean uncertainty at the k time step is an average for all the particles across the distances between each particle and its companions in all the simulations at the same time, which can be written for the k time step as:

$$U^k_M = \frac{1}{2N_P\left(N_S - 1\right)^2} \sum_{i=1}^{N_P} \sum_{m=1}^{N_S} \sum_{n=1}^{N_S} L^k_{i:m,n}\left(1 - \delta_{mn}\right) \quad : \quad k = \{1...N_T\} \tag{11}$$

where δ_{mn} is the Kronecker delta that is equal to unity where $m = n$ and zero where $m \neq n$. Use of δ_{mn} eliminates the zero distance between a particle and itself. The factor 2 in the denominator is necessary because the summation provides a double counting of the distances, i.e. the distance between the i^{th} particle in model $m = 1$ and model $n = 3$ is the same as the distance between that particle for $m = 3$ and

$n = 1$.

A second measure of uncertainty is the root-mean-square distance, which can be written for the k time step as:

$$U_R^k = \left\{ \frac{1}{2N_P \left(N_S - 1\right)^2} \sum_{i=1}^{N_P} \sum_{m=1}^{N_S} \sum_{n=1}^{N_s S} \left(L_{i:m,n}^k\right)^2 \left(1 - \delta_{mn}\right) \right\}^{1/2} \quad : \quad k = \{1...N_T\} \quad (12)$$

The forecast horizon, the time span over which the forecast is believable, can be determined using the measurement of mean distance or RMS distance as a representative of forecast uncertainty to fill in the curve of Fig. §3.10.1 over simulation time.

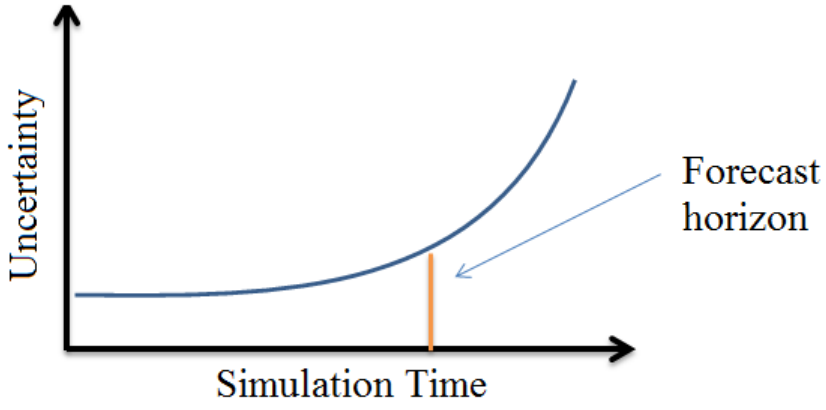

Figure 3.10.1: Forecast horizon

Chapter 4 - Results and Discussion

4.1 HyosPy and RK4 Tracer Simulation

The following example shows a HyosPy simulation of Corpus Christi Bay for 48 hours using wind and tidal data starting at midnight on 2/16/2013. Twelve hydrodynamic SELFE models run at starting intervals of four hours. The first model run uses all forecast data, with subsequent models using available hindcast data (Table §4.1.1). These sequenced simulations are designed only to demonstrate the automated functionality of the HyosPy code in downloading data, running a sequence of hydrodynamic models, tracking particles, and visualizing results. Appropriate spin-up simulations and validation of the model have not been conducted as part of this research.

Model #	Start time	Hindcast	Forecast
1	2/16 0:00	0 hour	48 hours
2	2/16 4:00	4 hours	44 hours
3	2/16 8:00	8 hours	40 hours
...	...	...	...
12	2/17 20:00	44 hours	4 hours

Table 4.1.1: Sequenced model operations

The tidal conditions used for the 1^{st}, 4^{th}, 8^{th} and 12^{th} model are shown in Fig. §4.1.1:

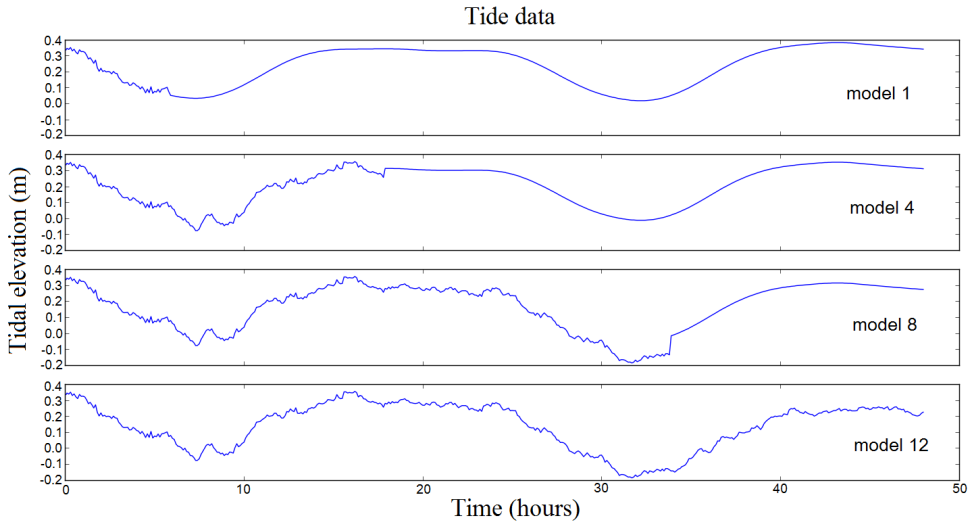

Figure 4.1.1: Tide data used for model 1, 4, 8, and 12

The wind conditions used for the 1^{st}, 4^{th}, 8^{th} and 12^{th} model are shown in Fig. §4.1.2:

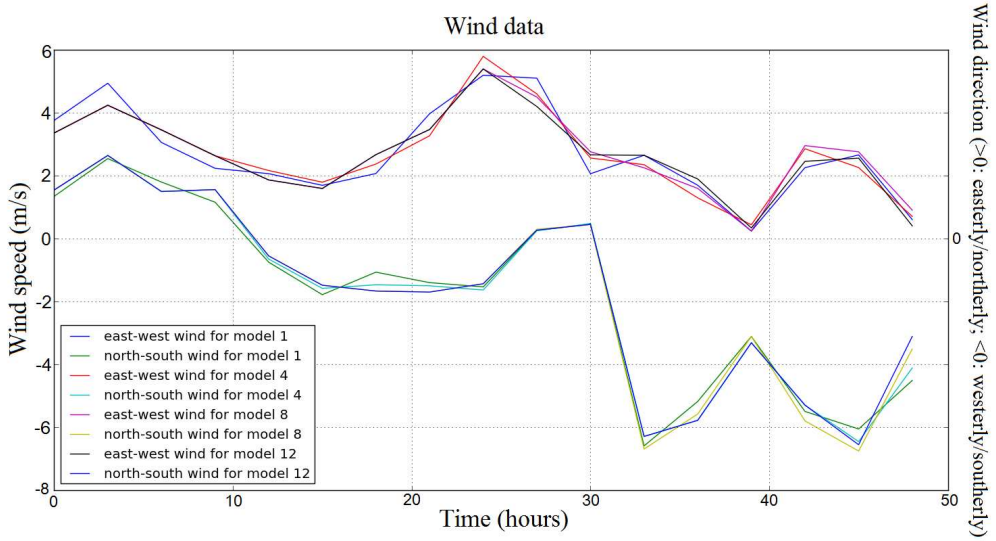

Figure 4.1.2: Wind data used for model 1, 4, 8, and 12

An imaginary oil spill is introduced at 0:00 2/16/2013 represented by 13 particles distributed about a center location (Fig. §4.1.3). More particles can be used in the RK4 method; a reduced number of particles was used in the present simulations to simplify testing of the visualization techniques.

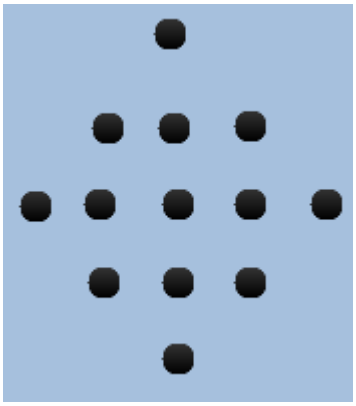

Figure 4.1.3: Initial arrangement of particles

The RK4 method moves the particles at the same 15 minute time step used by SELFE, providing 192 positions over the 48 hours of simulation. By integrating the hydrodynamic modeling results with the RK4 surface tracer transport routine, 12 different oil spill trajectories are obtained. For the present work, the visualization does not distinguish between the different trajectories, but shows all of the particles to provide a visualization of the spread produced by the different models. Fig. §4.1.4 shows all 192 positions of each of the 13 tracking particles over the course of the simulations, while Fig. §4.1.5, Fig. §4.1.6, and Fig. §4.1.7 show enlarged view near the start, in the middle, and near the end of the spill, correspondingly.

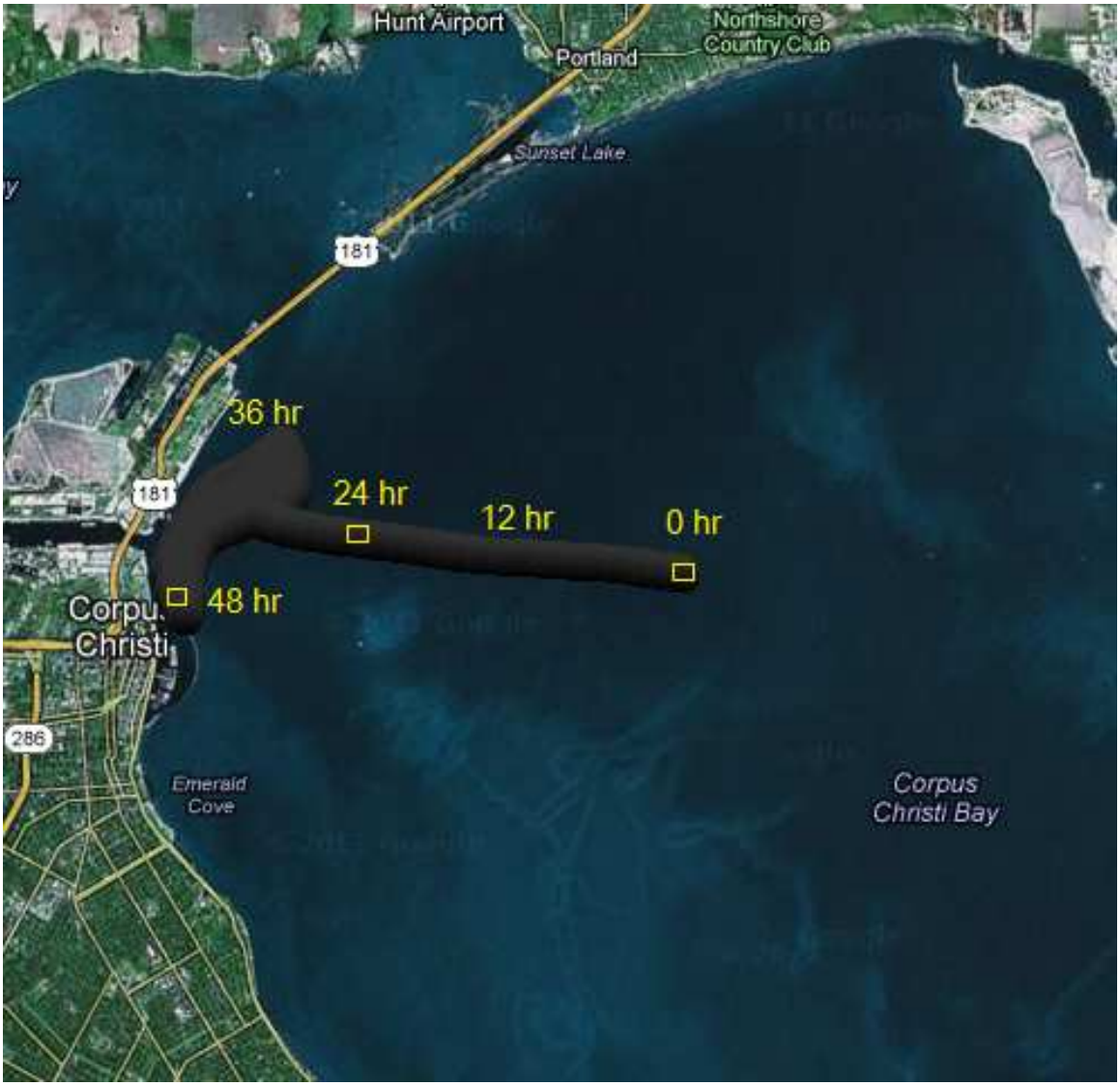

Figure 4.1.4: Oil spill trajectory prediction (48 hours). Results from 12 different RK4 surface tracer simulations plotted using Google Earth

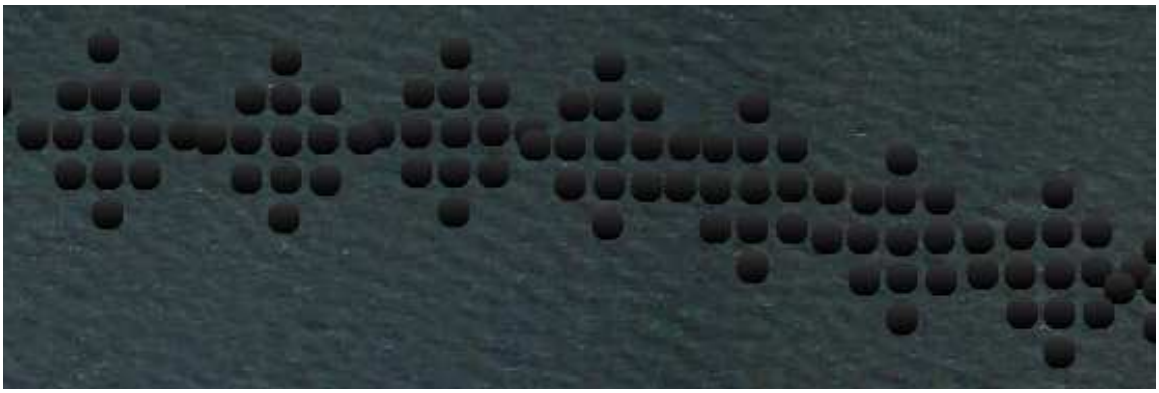

Figure 4.1.5: Particle distribution near the start of the spill

Figure 4.1.6: Particle distribution in the middle of the spill

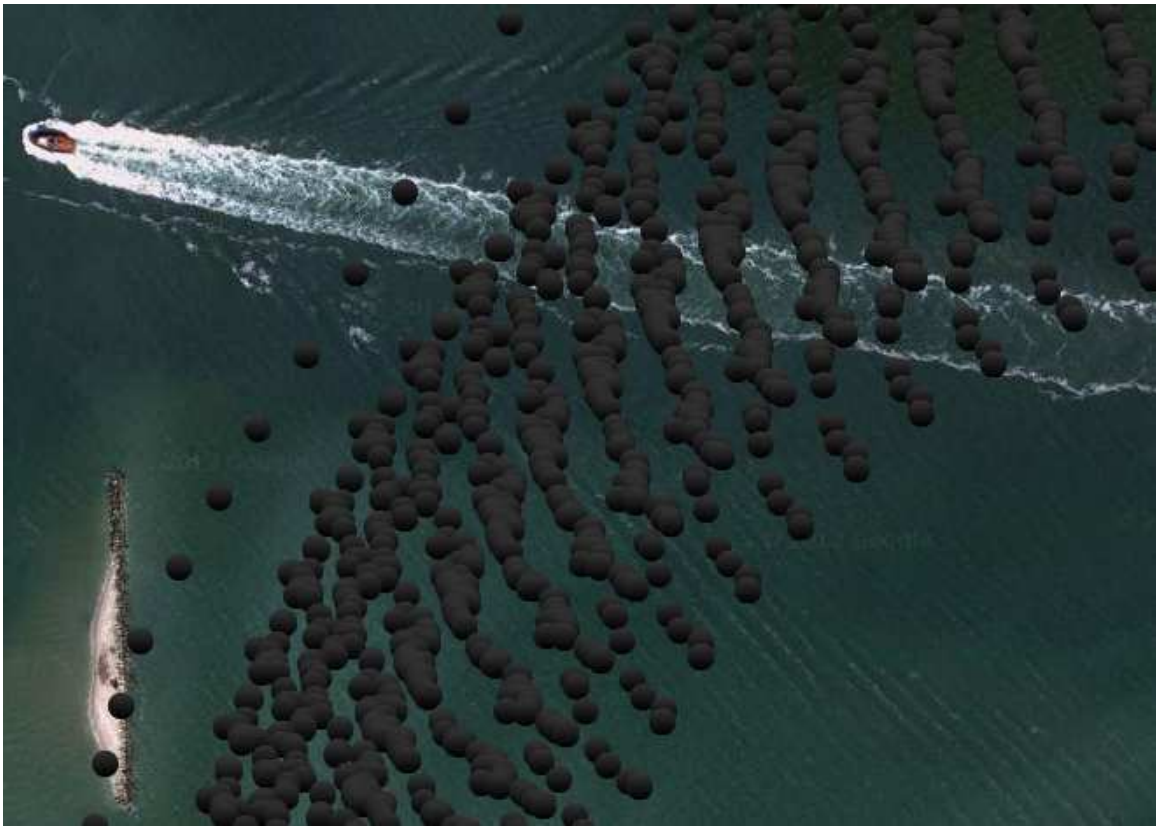

Figure 4.1.7: Particle distribution near the end of the spill

4.2 Hydrodynamic model forecast uncertainty

The HyosPy and RK4 tracer results in Fig. 4.1.4 can be used to compute the hydrodynamic model forecast uncertainty using the approach of §3.10. Fig. §4.2.1 shows the hydrodynamic uncertainty involved with modeling time for the above simulation. The horizontal axis represents the simulation time in hours while the vertical axis is hydrodynamic uncertainty which is quantified using mean distance and RMS distance between the predicted particle positions at each time step for the different simulations.

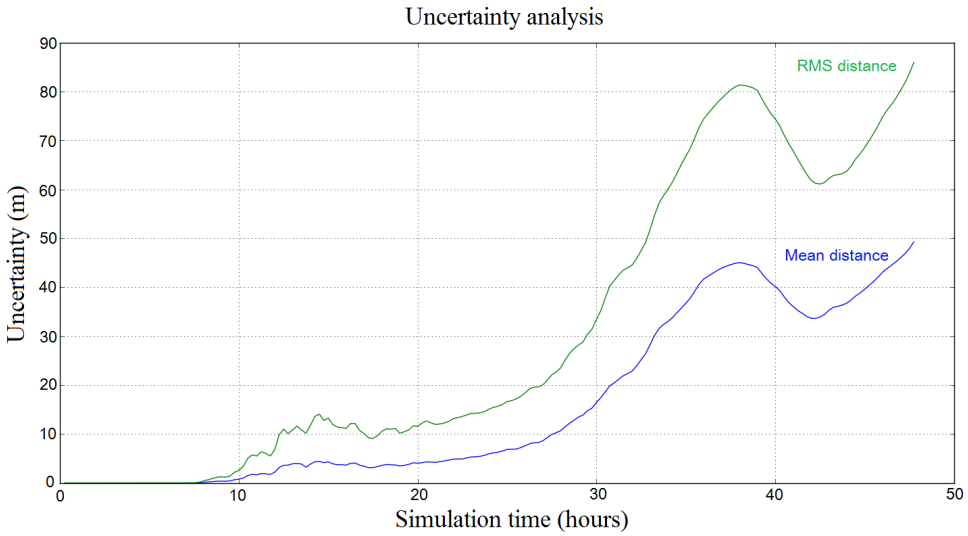

Figure 4.2.1: Hydrodynamic forecast uncertainty

4.3 Discussion

Because the present modeling exercise was to test model functionality, the SELFE model was started with zero velocities and a flat free surface; i.e. no "spin-up" time was allowed. The resulting particle transport (Fig. 4.1.4) is dominated by currents along the ship channel driven by the initial adjustment of the basin driven, principally by tidal flow. It does not appear that significant wind-driven effects were developed; however, this remains an area where more detailed study of SELFE is necessary.

The hydrodynamic uncertainty curve (Fig. §4.2.1) shows an increase in the first 12 hours, followed by a small decrease, then slowly increasing for 12-24 hours, followed by a rapid increase to 37 hours, followed by another decrease until 42 hours.

This behavior is consistent with slowing and reversal of the tidal current, which is expected to be approximately 90 degrees out of phase with the tidal elevation. From Fig. 3.6.1 it is expected that the initial current will begin slowing at 8 hours and be fully reversed near 14 hours, which leads to particles reversing their path and decreasing the dispersion distance (which is the uncertainty measurement). The dramatic increase of uncertainty beyond 24 hours occurs when the particles leave the ship channel and enter the near-shore current (see Fig. 4.1.4), which is coincidentally near the time of the next current reversal. The increase in dispersion of the particles entering the near shore current dominates the effects of current reversal at this time. However, once the particles are well established in the nearshore current, then the current reversal beginning around 36 hours leads to decreasing uncertainty as the particles are again forced back upon their previous tracks. Generally, the forecast horizon might be around 30 hours for this simulation compared with Fig. §3.10.1.

Fig. §4.1.5 shows that there are only 13 particles distributed in the original shape at the first few time steps, which indicates that these twelve different oil spill trajectories are overlapped with each other and there is little uncertainty. However, the distribution of the particles becomes disordered in Fig. §4.1.6 and Fig. §4.1.7, which means the twelve oil spill trajectories are no longer overlapped and there is increasing uncertainty involved with simulation time. Because the hydrodynamic prediction at any time depends on the behavior at previous times (including error), there is a forward integration of uncertainty involved with the hydrodynamic forecast simulation.

Chapter 5 - Conclusions and Future Work

5.1 Conclusions

This study develops a system, HyosPy wrapper & RK4 surface tracer transport routine, for automated sequencing of multiple hydrodynamic (currently SELFE) model runs and oil spill transport visualization on Google Earth. By setting up a sequence of hydrodynamic models running with pre-defined intervals, the HyosPy wrapper can provide an ensemble of model predictions based on wind and tide forcing for creating an estimate of the hydrodynamic forecast uncertainty. As a testbed for sequenced models, HyosPy does not presently include the initial conditions for SELFE (i.e. the newer model initiates just with latest wind and tide data but not with the output conditions from the prior model). The hydrodynamic model forecast uncertainty is estimated using simple statistical measurements and the forecast horizon can be determined through the forecast uncertainty curve. The simulation result shows that hydrodynamic model uncertainties will increase slowly at the beginning of the prediction period and will increase sharply after some point, which is defined as forecast horizon. Therefore, this advanced forecast modeling system will provide quantitative insight into how far in the future the hydrodynamic model forecast results can be trusted, which will provide operational managers with better information to effectively position emergency response equipment.

5.2 Future Work

In the prior project, an automated linkage between TWDB hydrodynamic models and GNOME oil spill trajectory model was developed under Mac OS X operating system (Rosenzweig and Hodges, 2011). In this project extension, the development platform has moved to Linux to suit for TWDB policy and system implementation convenience. NOAA is still working on the port of GNOME to Linux, so a simple RK4 surface tracer transport routine was developed in this research to simulate the surface oil spill trajectory. Future work should include integrating GNOME with the HyosPy when the Linux version of GNOME is available.

Since the current HyosPy only initiates multiple hydrodynamic SELFE model running with wind and tide data (i.e. not with initial SELFE condition from the prior model outputs) it is not presently suitable for operational implementation: sequenced models will consume too much computational time repeating the same computations. Work to be completed in the summer of 2013 will include translating the prior SELFE output into the initial condition of the next SELFE model.

In this research, the tide forecast is generated using the harmonic tidal elevation with a matching condition that adjusts the forecast for a smooth transition. This approach becomes questionable over longer time scales. This method could be improved in the future by using predictions from Gulf of Mexico circulation models rather than harmonic tides.

The RK4 method in this study uses the same time step as the SELFE model. However, the RK4 has a strict stability limit for the Courant-Friedrichs-Lewy con-

dition (CFL condition) which is a necessary condition for convergence while solving certain partial differential equations. But SELFE can be stable with large CFL conditions in the velocity field. Thus, it is possible that the time step for SELFE may be too large for the RK4 method. Future work requires conducting more analysis to the CFL conditions for RK4 and taking a smaller time step with the RK4 and interpolating in both time and space between SELFE velocity fields.

In addition, the forecast uncertainty analysis in this study is a preliminary demonstration of a simple approach. Better methods to evaluate the hydrodynamic forecast uncertainty should be developed by conducting in-depth analysis of different kinds of statistics to calculate the forecast uncertainty.

Appendix

This appendix shows all the modules and structures of hyospy, the RK4 tracer routine, and the forecast uncertainty tool. The comments explaining of each module is presented with a # in front of the code.

MODULE 1: Wind Forecasts Download and Conversion

```
# wind_th.py
# This module would automatically download the wind forecasts,
# transform the data format,and save the wind.th to SELFE directory.

import urllib
import string
import math
import os
import tarfile
from contextlib import closing

# Download wind forecast data and save it at "./twdb_1c.tar".
url = 'http://seawater.tamu.edu/tglopu/twdb_lc.tar'
path = './twdb_1c.tar'
data = urllib.urlopen(url).read()
f = file(path,'wb')
f.write(data)
f.close()

# Unzip the tar file to './wind' when the download is over.
path = './twdb_1c.tar'
if os.path.isfile('./twdb_1c.tar'):
   os.mkdir('./wind')
   with closing(tarfile.open('./twdb_1c.tar','r')) as t:
        t.extractall('./wind')

# Create wind.th
string1='*'
a=open('./wind/twdb051.wndq', 'r').readlines()
open('./wind.txt','w').write('')
for x in a:
```

```
    if string1 in x:
         continue
    open('./wind.txt','a').write(x)

string2='days'
d=open('./wind.txt','r').readlines()
open('./wind1.th','w').write('')
for y in d:
    if string2 in y:
        continue
    open('./wind1.th','a').write(y)

file = open('./wind1.th','r')
row=[]
for s in file.readlines():
    column=[]
    line=s.split()
    for field in line: column.append(field)
    row.append(column)
file.close

for m in range(len(row)):
     row[m][0:4]=[]

for n in range(len(row)):
    aa=string.atof(row[n][0]);bb=string.atof(row[n][1])
    row[n][0]=0.447*aa*math.sin(bb*math.pi/180)
    row[n][1]=-0.447*aa*math.cos(bb*math.pi/180)

# Save wind.th to the SELFE directory
f=open('./SELFE/wind.th','w')
for i in row:
    k='   '.join([str(j) for j in i])
    f.write(k+'\n')
f.close
```

MODULE 2: Tide Forecasts Download and Conversion

```
# wind_th.py
# This module would automatically download the wind forecasts,
# transform the data format,and save the wind.th to SELFE directory.

import urllib
import string

# Download pwl and harmwl data from TCOON
# Change date part of url to get different tide forecast
url = 'http://lighthouse.tamucc.edu/pd?stnlist=014&serlist\
=pwl%2Charmwl&when=03.03.2013-03.11.2013&whentz=\
UTC0&-action=c&unit=metric&elev=stnd'

path = './elevation.txt'
data=urllib.urlopen(url).read()
# print data
f = file(path,'wb')
f.write(data)
f.close

# Create elev.th
string1='#'
a=open('./elevation.txt', 'r').readlines()
open('./elev.txt','w').write('')
for x in a:
    if string1 in x:
         continue
    open('./elev.txt','a').write(x)

string2='NA'
d=open('./elev.txt','r').readlines()
open('./elev1.th','w').write('')
for y in d:
    if string2 in y:
        continue
    open('./elev1.th','a').write(y)

file = open('./elev1.th','r')
row=[]
for s in file.readlines():
    column=[]
    line=s.split()
    for field in line: column.append(field)
```

```
    row.append(column)
file.close

row.pop()
for m in range(len(row)):
     del row[m][0]

sum=0
for n in range(len(row)):
    aa=string.atof(row[n][0]);bb=string.atof(row[n][1])
    diff=aa-bb
    sum=sum+diff
x=sum/len(row)

file1 = open('./elev.txt','r')
row1=[]

for q in file1.readlines():
    column1=[]
    line1=q.split()
    for field1 in line1: column1.append(field1)
    row1.append(column1)

file1.close
row1.pop()

for aa in range(len(row1)):
    del row1[aa][0];del row1[aa][1]

y=row1.index(['NA'])

file2 = open('./elev.txt','r')
row2=[]

for q in file2.readlines():
    column2=[]
    line2=q.split()
    for field2 in line2: column2.append(field2)
    row2.append(column2)

file2.close
row2.pop()

for i in range(y,len(row2)):
    row2[i][1]=x+string.atof(row2[i][2])
```

```
for k in range(y):
    row2[k][1]=string.atof(row2[k][1])

for m in range(len(row2)):
    row2[m][0]=360*m
    del row2[m][2]

for t in range(len(row2)-1):
    row2.insert(2*t+1,[(2*t+1)*180,\
(row2[2*t][1]+row2[2*t+1][1])/2])

del row2[0]

# Save elev.th to SELFE directory
h=open('./SELFE/elev.th','w')
for l in row2:
    g='   '.join([str(j) for j in l])
    h.write(g+'\n')
h.close
```

MODULE 3: Hyospy Wrapper

```
# hyospy_wrapper.py
# This module runs multiple hydrodynamic models(SELFE)
# with pre-defined intervals

import subprocess
import time

def runSELFE(direc):
     subprocess.Popen('./SELFE.exe',cwd=direc,shell=False)

# Put all SELFE folder directions in selfedir.txt
for line in file('./selfedir.txt'):
     thisdirec = line.strip('\n')
     runSELFE(thisdirec)
     time.sleep(7200)       # Set pre-defined intervals
```

MODULE 4: Rk4 Surface Tracer Transport Routine

```
# RK4.py
# Compute the transport of multiple particles in multiple
# hydrodynamic models(SELFE) with RK4 method
# Mark the multiple oil spill trajectory predictions
# on Google Earth

import string
import heapq
import pyselfe_v1
import math

# Get all of the coordinate info and save them in
# the list of location

file = open('./drag.gr3','r')
location=[]
for s in file.readlines():
    column=[]
    line=s.split()
    for field in line: column.append(field)
    location.append(column)
file.close

del location[0:2];del location[23286:]

for i in range(len(location)):
    del location[i][0]
for j in range(len(location)):
    del location[j][2]

# Velocity reader: read velocity at point(x,y) at time t
def readVel(t,x,y):

    # Find the n nearest nodes to (x,y)
    distance=[]
    for m in range(len(location)):
        aa=string.atof(location[m][0])-x
        bb=string.atof(location[m][1])-y
        distance.append(aa*aa+bb*bb)

   # Find 3 smalleset number and save them at list 'nearest'
    nearest=heapq.nsmallest (3, distance)
```

```
    n1=distance.index(nearest[0]); n2=distance.index(nearest[1])
    n3=distance.index(nearest[2])

    x01=string.atof(location[n1][0]);y01=string.atof(location[n1][1])
    x02=string.atof(location[n2][0]);y02=string.atof(location[n2][1])
    x03=string.atof(location[n3][0]);y03=string.atof(location[n3][1])

    # Save the velocity in x direction at time step n in ui;
    #  y direction at time step n in wi, i=1,2,3
    if isinstance(t,int):
          u01=mdata[t,n1,5,0];w01=mdata[t,n1,5,1]
          u02=mdata[t,n2,5,0];w02=mdata[t,n2,5,1]
          u03=mdata[t,n3,5,0];w03=mdata[t,n3,5,1]

    else :
          u01=mdata[int(t),n1,5,0]/2+mdata[int(t)+1,n1,5,0]/2
          w01=mdata[int(t),n1,5,1]/2+mdata[int(t)+1,n1,5,1]/2
          u02=mdata[int(t),n2,5,0]/2+mdata[int(t)+1,n2,5,0]/2
          w02=mdata[int(t),n2,5,1]/2+mdata[int(t)+1,n2,5,1]/2
          u03=mdata[int(t),n3,5,0]/2+mdata[int(t)+1,n3,5,0]/2
          w03=mdata[int(t),n3,5,1]/2+mdata[int(t)+1,n3,5,1]/2

    # Save the distance between (x,y) and
    # the three nearest points to Li,i=1,2,3
    L1=math.sqrt((x01-x)*(x01-x)+(y01-y)*(y01-y))
    L2=math.sqrt((x02-x)*(x02-x)+(y02-y)*(y02-y))
    L3=math.sqrt((x03-x)*(x03-x)+(y03-y)*(y03-y))
    Lsum=L1+L2+L3

    # Interpolate to find velocity component of (x,y) at time step n
    # and save them as u,w
    u=L1/Lsum*u01+L2/Lsum*u02+L3/Lsum*u03
    w=L1/Lsum*w01+L2/Lsum*w02+L3/Lsum*w03
    return t,u,w

# Set how many hydrodynamic models(SELFE) need to run
# Define selfe_data_dirn as the direction of each model's outputs
selfe_data_dir1 = './SELFE_outputs_1'
selfe_data_dir2 = './SELFE_outputs_2'

dirlist=[]
# Add selfe_data_dirn into a list
dirlist.append(selfe_data_dir1);dirlist.append(selfe_data_dir2)

traceroutput=[] ; gradientoutput=[] ; dt=900 ; h=900
```

```
# Main loop(loop with different models)
for nmodels in range(2):

    selfe = pyselfe_v1.Dataset(dirlist[nmodels]+'1_hvel.64')

    # Argument 'nfiles' represents the total running time
    [t,t_iter,eta,dp,mdata]=\
    selfe.read_time_series('hvel.64',nfiles=48,\
datadir=dirlist[nmodels])

    length=len(mdata[:,7,5,0])

    for a in range(3):

        x=[None]*(length+1) ; y=[None]*(length+1)
        u=[None]*length ; w=[None]*length
        tracer=[] ; element=[] ; velocity=[] ; uv=[] ; gradient=[]
        xy=[] ; vel0=[] ; vel1=[] ; vel2=[] ; vel3=[]

    # Set N particles at different starting positions
        if a ==0:
            x[0]=691731.019;y[0]=3059752.263
        elif a ==1:
            x[0]=676785.5671;y[0]=3071586.4213
        else :
            x[0]=678077;y[0]=3066553

        for n in range(length-1):

            vel0=list(readVel(n,x[n],y[n]))
            u[n]=vel0[1];w[n]=vel0[2]

            dx=dt*u[n] ; dy=dt*w[n]
            uv=[u[n],w[n],dx,dy]
            velocity.append(uv)

            # Transport the oil spill particles by using
            # RK4 algorithm

            k1x=h*u[n] ; k1y=h*w[n]
            vel1=list(readVel(n+0.5,x[n]+k1x/2,y[n]+k1y/2))
            k2x=h*vel1[1] ; k2y=h*vel1[2]
            vel2=list(readVel(n+0.5,x[n]+k2x/2,y[n]+k2y/2))
            k3x=h*vel2[1] ; k3y=h*vel2[2]
```

```
            vel3=list(readVel(n+1,x[n]+k3x,y[n]+k3y))
            k4x=h*vel3[1] ; k4y=h*vel3[2]

            x[n+1]=x[n]+(k1x+2*k2x+2*k3x+k4x)/6
            y[n+1]=y[n]+(k1y+2*k2y+2*k3y+k4y)/6

            element=[(n+1)*900,x[n],y[n]]
            tracer.append(element)

        traceroutput.append(tracer)

# Mark oil spill trajectories on Google Earth
lonlat=[]

# nmodelpoint is the number of "# of model * # of point"
for nmodelpoint in range(6):
    for nposition in range(len(traceroutput[nmodelpoint])):
        aaa=utmToLatLng(14,traceroutput[nmodelpoint][nposition][1],\
traceroutput[nmodelpoint][nposition][2],northernHemisphere=True)
        lonlat.append([aaa[0],aaa[1]])

coordinates=str(lonlat)

file1 = open('./javascript.txt','r')  # Open Google Marker JavaScript
row=[]
for s in file1.readlines():
    row.append(s)
# Copy all of the coordinates in the javascript
row[11]='    '+'var'+' '+'locations'+'='+coordinates

# Generate the final html file
h=open('./tracers_googlemap_markers.html','w')
for l in row:
    g=''.join([str(j) for j in l])
    h.write(g+'\n')
h.close
```

MODULE 5: UTM coordinates conversion

```
# This module convert (x,y) coordinates to (lat,lon)

def utmToLatLng(zone, easting, northing, northernHemisphere=True):
    if not northernHemisphere:
        northing = 10000000 - northing

    a = 6378137
    e = 0.081819191
    e1sq = 0.006739497
    k0 = 0.9996

    arc = northing / k0
    mu=arc/(a*(1-math.pow(e,2)/4.0-\
*math.pow(e,4)/64.0-5*math.pow(e,6)/256.0))

    ei=(1- math.pow((1-e*e),(1/2.0)))/(1+math.pow((1-e*e),(1/2.0)))

    ca = 3 * ei / 2 - 27 * math.pow(ei, 3) / 32.0

    cb = 21 * math.pow(ei, 2) / 16 - 55 * math.pow(ei, 4) / 32
    cc = 151 * math.pow(ei, 3) / 96
    cd = 1097 * math.pow(ei, 4) / 512
    phi1=mu+ca*math.sin(2*mu)+cb*math.sin(4*mu)+\
cc*math.sin(6*mu)+cd*math.sin(8*mu)

    n0=a/math.pow((1-math.pow((e*math.sin(phi1)),2)),(1/2.0))

    r0=a*(1-e*e)/math.pow((1-math.pow((e*math.sin(phi1)),2)),(3/2.0))
    fact1 = n0 * math.tan(phi1) / r0

    _a1 = 500000 - easting
    dd0 = _a1 / (n0 * k0)
    fact2 = dd0 * dd0 / 2

    t0 = math.pow(math.tan(phi1), 2)
    Q0 = e1sq * math.pow(math.cos(phi1), 2)
    fact3=(5+3*t0+10*Q0-4*Q0*Q0-9*e1sq)*math.pow(dd0,4)/24

    fact4=(61+90*t0+298*Q0+45*t0*t0-252*e1sq-3*Q0*Q0)*\
math.pow(dd0,6)/720

    lof1=a1 / (n0 * k0)
```

```
    lof2=(1 + 2 * t0 + Q0) * math.pow(dd0, 3) / 6.0
    lof3=(5-2*Q0+28*t0-3*math.pow(Q0,2)+\
    8*e1sq+24*math.pow(t0,2))*math.pow(dd0,5)/120
    _a2=(lof1 - lof2 + lof3) / math.cos(phi1)
    _a3=_a2 * 180 / math.pi

    latitude =180*(phi1-fact1*(fact2+fact3+fact4))/ math.pi

    if not northernHemisphere:
        latitude = -latitude

    longitude = ((zone > 0) and (6 * zone - 183.0) or 3.0) - _a3

    return [latitude,longitude]
```

MODULE 6: Google Map Marker animation JavaScript

```
#Google_marker.html

<html>
<head>
  <meta http-equiv="content-type" content="text/html;charset=UTF-8"/>
  <title>Google Maps Multiple Markers</title>
  <script src="http://maps.google.com/maps/api/js?sensor=false"
          type="text/javascript"></script>
</head>
<body>
  <div id="map" style="width: 1300px; height: 800px;"></div>

  <script type="text/javascript">
    var locations = [];

    var map =
new google.maps.Map(document.getElementById('map'), {
      zoom: 10,
      center: new google.maps.LatLng(27.8, -97.35),
      mapTypeId: google.maps.MapTypeId.ROADMAP
    });

    var infowindow = new google.maps.InfoWindow();

    var marker, i;
```

```
    for (i = 0; i < locations.length; i++) {
      marker = new google.maps.Marker({
        position:
new google.maps.LatLng(locations[i][0], locations[i][1]),
        map: map
      });

google.maps.event.addListener(marker, 'click', (function(marker, i)
{
        return function() {
          infowindow.setContent(locations[i][0]);
          infowindow.open(map, marker);
        }
      })(marker, i));
    }
  </script>
</body>
</html>
```

MODULE 7: Forecast Uncertainty Routine

```
import string
import heapq
import pyselfe_v1
import math
from pylab import *
import numpy

# forecast.py
# Estimate and quantify hydrodynamic forecast uncertainty

...# Use RK4 tracer routine to calculate all of the coordiantes

um=[] ; ur=[] ; np=13 ; ns=12 ; nt=len(multiplemodel[0][0])
# np is the total particle number; ns is the total model number
# nt is the total time steps

for k in range(nt):   # k is time step

    sum0=0 ; sum1=0

    for i in range(np):        # i is the particle number

        for m in range(ns):     # m is the model number
```

```
            for n in range(ns):   # n is the model number

                sum0=sum0+math.sqrt(math.pow(\
(multiplemodel[m][i][k][1]-multiplemodel[n][i][k][1]),2)\
+math.pow((multiplemodel[m][i][k][2]-multiplemodel[n][i][k][2]),2))


                sum1=sum1+math.pow(math.sqrt(math.pow(\
(multiplemodel[m][i][k][1]-multiplemodel[n][i][k][1]),2)\
+math.pow((multiplemodel[m][i][k][2]-multiplemodel[n][i][k][2]),2)),2)


    um.append([(k+1)*900,sum0/2/np/math.pow((ns-1),2)])
    ur.append([(k+1)*900,math.sqrt(sum1/2/np/math.pow((ns-1),2))])

uum=numpy.asarray(um)
uur=numpy.asarray(ur)

xlabel('Simulation time (hours)')
ylabel('Uncertainty')
title('Uncertainty analysis')
grid(True)

plot(uum[:,0]/3600,uum[:,1],'-')
plot(uur[:,0]/3600,uur[:,1],'-')

show()
```

References

Al-Khafaji, A. W. and J. R. Tooley (1986). *Numerical Methods in Engineering Practice*. CBS College Publishing.

ASA (1997). OILMAP for Windows technical manual. Technical report, Applied Science Associates, Narragansett, R.I.

ASCE-Task-Committee (1996). State-of-the-art review of modeling transport and fate of oil spills. Technical report, ASCE task committee on modeling of oil spills of the water resources engineering division.

Beegle-Krause, C. J. (2005). General NOAA oil modeling environment (GNOME): a new spill trajectory model. In *International Oil Spill Conference*.

Blain, C. A. and W. E. Rogers (1998). Coastal Tide Prediction Using the ADCIRC-2DDI Hydrodynamic Finite Element Model: Model Validation and Sensitivity Analyses in the Southern North Sea/English Channel. Technical report, Naval Research Laboratory, Slidell, LA, USA.

Chao, X. B., N. J. Shankar, and H. F. Cheong (2001). Two- and three- dimensional oil spill model for coastal waters. *Ocean Engineering 28*, 1557–1573.

Chao, X. B., N. J. Shankar, and S. Wang (2003). Development and application of oil spill model for Singapore coastal waters. *Journal of Hydraulic Engineering 129*, 495–503.

Chao, Y. S., T. K. Kim, W. Jeong, and T. Ha (2012). Numerical Simulation of Oil Spill in Ocean. *Journal of Applied Mathematics 2012*.

Cheng, Y. C., X. F. Li, Q. Xu, O. G. Pineda, O. B. Andersen, and W. G. Pichel (2011). SAR observation and model tracking of an oil spill event in coastal waters. *Marine Pollution Bulletin 62*, 350–363.

Deborah, P. F. M., A. J. Mark, and C. Louis (1999). Oil spill modeling for contingency planning and impact assessment and example application for Florida Power & Light. Technical report, Applied Science Associates, Narragansett, Rhode Island.

Google-developers (2013). Marker animations. `https://developers.google.com/maps/documentation/javascript/overlays#MarkerAnimations`.

Guo, W. J., M. X. Me, and Y. J. Gui (2009). Modeling oil spill trajectory in coastal waters based on fractional brownian motion. *Marine Pollution Bulletin 58*, 1339–1346.

Hodges, B., J. Furnans, and P. Kulis (2011). Thin-Layer Gravity Current with Implications for Desalination Brine Disposal. *Journal of Hydraulic Engineering 137*, 356–371.

Hodges, B. R. and X. Hou (2013, January). Uncertainty estimation in operational oil spill modeling for bays and estuaries. In *Poster presented at the Gulf of Mexico Oil Spill & Ecosystem Science Conference*, New Orleans, Louisiana, USA.

Huang, J. C. (1983). A review of the state-of-the-art of oil spill fate/behavior models. In *Proceeding of the 1983 Oil Spill Conference*, Washington,DC, pp. 313–322.

Humphrey, B., D. R. Green, B. R. Fowler, D. Hope, and P. D. Boehm (1987). The Fate of Oil in the Water Column Following Experimental Oil Spills in the Arctic Marine Nearshore. *Arctic Institute of North America 40*, 124–132.

IBM-developers (2013). Coordinate conversions made easy. `http://www.ibm.com/developerworks/java/library/j-coordconvert/index.html`.

Inan, A. and L. Balas (1980). Numerical Modelling of Oil Spill. In *Proceedings of the National Conference on Control of Hazardous Material Spills*, Louisville, KY, USA, pp. 364–368.

IPIECA (2002). Oil Spill Responder Safety Guide. Technical report, International Petroleum Industry Environmental Conservation Association, London, UK.

Lindstrom, G. (2005). Programming with Python. *IT Professional 7*, 10–16.

Lonin, S. A. (1999). Lagrangian model for oil spill diffusion at sea. *Spill & Technology Bulletin 5*, 331–336.

Lynch, D. R. and W. G. Gray (1979). A Wave Equation Model for Finite Element Tidal Computations. *Computers and Fluids 7*, 207–228.

Mackay, D., S. Paterson, and S. Nadeau (1980). Calculation of the evaporation rate of volatile liquids. In *Proceedings of the National Conference on Control of Hazardous Material Spills*, Louisville, KY, USA, pp. 364–368.

Matsumoto, J. (1993). User's Manual for the Texas Water Development Board's Hydrodynamic and Salinity Model, TxBLEND. Technical report, Texas Water Development Board, Austin, Texas, USA.

Nagheeby, M. and M. Kolahdoozan (2010). Numerical modeling of two-phase fluid flow and oil slick transport in esturine water. *International Journal of Environmental Science and Technoloogy 7*, 771–784.

NOAA (2013). OR&R. `http://response.restoration.noaa.gov/oil-and-chemical-spills/oil-spills`.

Reed, M. (2000). A multicomponent 3D oil spill contingency and reponse model. In *23rd Arctic and Marine Oilspill Programme (AMOP) Technical Seminar*, Ottawa, Canada, pp. 663–680.

Rosenzweig, I. and B. R. Hodges (2011). A Python Wrapper for Coupling Hydrodynamic and Oil Spill Models. Technical report, Center for Research in Water Resources, The University of Texas at Austin, Austin, Texas, USA.

SELFE-developers (2013a). Post-processing tools. `http://www.stccmop.org/CORIE/modeling/selfe/utilities.html`.

SELFE-developers (2013b). SELFE v3.1d User Manual. `http://www.stccmop.org/CORIE/modeling/selfe/input_mpi.html`.

Shchepetkin, A. F. and J. C. McWilliams (2005). The regional oceanic modeling system (ROMS): a split-explicit, free-surface, topography-following-coordinate, oceanic model. *Ocean Modeling 9*, 347–404.

Shen, H. T. and P. D. Yapa (1988). Oill slick transport in rivers. *Journal of Hydraulic Engineering 114*, 529–543.

Shen, H. T., P. D. Yapa, and M. E. Petroski (1986). Simulation of oil slick transport in Great Lakes connecting channels. Technical report, Department of Civil and Environmental Engineering, Clarkson University, Potsdam, NY.

Spaulding, M. L. (2010). Hydraulic & Hydrology, Energy & Environmental Eng. Series. In *Proceedings of the* 5^{th} *IASME/WSEAS Int. Conf.*, pp. 62–67.

TCOON (2013). Texas Coastal Ocean Observation Network. `http://lighthouse.tamucc.edu/TCOON/HomePage`.

TWDB (2013). TxBLEND Hydrodynamic Model for Oil Spill. `http://www.twdb.state.tx.us/surfacewater/bays/oil_spill/index.asp`.

Wang, S. D., Y. M. Shen, Y. K. Guo, and J. Tang (2007). Three-dimensional numerical simulation for transport of oil spills in seas. *Ocean Engineering 35*, 503–510.

Ward, G. H. (1997). *Processes and Trends of Circulation Within the Corpus Christi Bay National Estuary Program Study Area*. Texas Natural Resource Conservation Commission.

William, B. S., E. A. David, B. Rakesh, and Z. Christopher (2013). Development of a Global Oil Spill Modeling System. *Earth Science Research 2*.

Yapa, P. D., H. T. Shen, and K. Angammana (1994). Modeling oil spills in a river-lake system. *Journal of Marine System 4*, 453–471.

Zhang, Y. J. (2010, May). Technical support Inter-model comparison for Corpus Christi Bay testbed. Technical report, NSF Center for Coastal Margin Observation & Prediction,OGI School of Science & Engineering,Oregon Health & Science University (OHSU).

Zhang, Y. J. and A. M. Baptista (2008). SELFE: A semi-implicit Eulerian-Lagrangian finite-element model for cross-scale ocean circulation. *Ocean Modeling 21*, 71–96.