UNSTRUCTURED MESH GENERATION AND DYNAMIC LOAD BALANCING FOR COASTAL OCEAN HYDRODYNAMIC SIMULATION

A Dissertation

Submitted to the Graduate School

of the University of Notre Dame

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

by

Keith J. Roberts

Dr. Joannes J. Westerink, Director

Graduate Program in Civil and Environmental Engineering and Earth Sciences

Notre Dame, Indiana

May 2019

UNSTRUCTURED MESH GENERATION AND DYNAMIC LOAD BALANCING FOR COASTAL OCEAN HYDRODYNAMIC SIMULATION

Abstract

by

Keith J. Roberts

This work examines and improves the efficiency of numerical modeling of tides, storm surge and associated flooding on unstructured meshes. Unstructured meshes composed of triangles are frequently used for numerical simulations of the coastal ocean because they can resolve the large gap in horizontal length scales necessary for accurate simulations of total water levels. However, the accuracy and the associated computational expense of the mesh are in direct conflict, which makes the mesh development process challenging.

A comprehensive approach to automatically build sophiscated planar triangulations (meshes) is developed in a toolkit called *OceanMesh2D*. In this software, resolution is controlled via functions of seabed data and shoreline geometry. The most challenging step of simplifying the shoreline boundary in the mesh is made automatic with a sequence of mesh improvement strategies. The main result is that seamless regional and global modeling systems can be built in minutes to hours automatically and approximately reproducibly.

The toolkit is used to investigate the design of unstructured mesh resolution and its impact on the modeling of barotropic tides along the United States coastlines. The key findings indicate that pre-existing mesh designs that use uniform resolution along the shoreline and slowly varying resolution sizes on the continental shelf inefficiently discretize the computational domain. Instead, targeting resolution in narrow geometric features and along large topographic gradients and estuarine channels, while aggressively relaxing resolution elsewhere, leads to an efficient mesh design with an order of magnitude fewer vertices than a reference solution with comparable tidal accuracy ($\pm$3% harmonic elevation amplitudes).

Lastly, coastal ocean models with overland floodplains induce computational workload imbalances because dry elements incur zero computational cost. A capability to evenly distribute computational work dynamically as floodplain regions wet and dry during the passage of a storm is developed for the ADvanced CIRCulation model. The approach is based on partitioning the decomposition of the mesh during run time so that the that the computational load is determined by the degrees of freedom in wetted areas. I demonstrate that the implementation has a low overhead cost and speed-ups of 10-45% can be achieved for real world coastal flooding simulations.

This dissertation is dedicated to Mariana and my beloved family.

CONTENTS

FIGURES vi

TABLES xv

ACKNOWLEDGMENTS xvi

CHAPTER 1: INTRODUCTION 1
- 1.1 Overview 1
- 1.2 Layout of Dissertation 12

CHAPTER 2: AUTOMATED MESH GENERATION FOR COASTAL HYDRODYNAMIC MODELING 15
- 2.1 Overview 15
- 2.2 Introduction 16
- 2.3 Software Architecture Overview 19
- 2.4 Component Design 22
  - 2.4.1 Mesh generation: *meshgen* class 24
    - 2.4.1.1 *DistMesh2D* Algorithm 25
    - 2.4.1.2 Termination criterion 31
    - 2.4.1.3 Mesh improvement strategies during mesh generation 32
    - 2.4.1.4 Mesh improvement strategies after mesh generation . 34
  - 2.4.2 Multiscale meshing capability 38
  - 2.4.3 Geospatial data preprocessing: *geodata* class 41
    - 2.4.3.1 Automatic mesh boundary definition 41
    - 2.4.3.2 Shoreline pre-processing 46
    - 2.4.3.3 Meshing in projected spaces 48
  - 2.4.4 Automatic mesh size function: *edgefx* class 50
    - 2.4.4.1 Distance and feature size 51
    - 2.4.4.2 Wavelength-to-grid size 53
    - 2.4.4.3 Topographic length scale 54
    - 2.4.4.4 Channel thalwegs/polylines 56
    - 2.4.4.5 Finalizing the mesh size function 60
    - 2.4.4.6 Mesh size transitions (gradation) 60
    - 2.4.4.7 Courant-Friedrichs-Lewey (CFL)-limiting 62
  - 2.4.5 Data container: *msh* class 68

2.4.5.1 Automatic merging 70
2.4.5.2 Meshing the floodplain 71
2.5 Performance of library 72
2.6 Discussion and conclusions 74

CHAPTER 3: ON THE AUTOMATIC AND *A PRIORI* DESIGN OF COASTAL OCEAN CIRCULATION MODELS 80
3.1 Overview 80
3.2 Introduction and background 81
3.3 Methods, Data and Tools 86
3.3.1 ECGC Study Domain and Data 86
3.3.1.1 Bathymetric and Shoreline Datasets 88
3.3.1.2 Tide Gauge Data 89
3.3.2 Hydrodynamic Model Configuration 89
3.3.3 Mesh Generation 91
3.3.4 Experimental Design 92
3.4 Results 99
3.4.1 Resolving the shoreline 99
3.4.2 Mesh size gradation 104
3.4.3 Resolution along bathymetric gradients 109
3.4.4 Cross-sectional representation of estuarine channels 114
3.4.5 Summary of experiments 118
3.4.5.1 Predominant variability 118
3.4.5.2 Numerical error versus physical approximation error 119
3.4.5.3 Mesh design combinations 123
3.5 Discussion and Conclusions 128

CHAPTER 4: DYNAMIC LOAD BALANCING FOR PREDICTIONS OF STORM SURGE AND COASTAL FLOODING 137
4.1 Overview 137
4.2 Introduction 138
4.3 Methods 142
4.3.1 ADCIRC Hydrodynamic model 142
4.3.2 Load balancing strategy 143
4.3.3 Mesh partitioning 144
4.3.3.1 Parallel Mesh Partitioning Algorithm 146
4.3.3.2 Elemental Weighting Scheme 149
4.3.3.3 Loop Rearrangement 151
4.3.4 Rebalance strategies 152
4.3.5 Performance metrics 154
4.4 Experimental results 156
4.4.1 Simple Tide on an Idealized Floodplain 157
4.4.1.1 Model and Setup 157
4.4.1.2 Timing Improvements and Scalability 160

4.4.2 Flooding from Hurricane Irene in North Carolina 165
4.4.2.1 Model and Setup 165
4.4.2.2 Simulations 168
4.4.2.3 Comparison with measured data 168
4.4.2.4 Timing Improvements 169
4.5 Discussion and Conclusion 176

CHAPTER 5: CONCLUSIONS AND FUTURE WORKS 183
5.1 Conclusions 183
5.2 Future work 187

APPENDIX A: PARALLEL WEIR PARTITIONING 191

BIBLIOGRAPHY 194

FIGURES

1.1 The western North Atlantic study region that was explored in this work. The continental shelf region is shaded in red and green. The typical track of a tropical cyclone (i.e., hurricane) is indicated by the white arrows and the various regions of interest described throughout the text are labeled (GOM: Gulf Of Mexico, NA: North Atlantic, SA: South Atlantic, MA: Mid-Atlantic). 2

1.2 An overview of the two-dimensional (2D) mesh-based coastal modeling approach. 4

1.3 The triangulation of a typical regional coastal model at various cartographic scales. The green lines in the insets indicate the geometric complexity of the internal type barrier structure that is modeled. . . 6

2.1 The geographical location and triangulation of the three meshes used as examples in this work. The minimum mesh sizes (*h0*) are annotated in black text and the names of the digital elevation models (DEMs) used in the construction of the mesh size functions are annotated in red text on each panel. The colormap indicates topographical data (bathymetric data was removed for production of this figure) in the DEMs, which are freely available through the NOAA Bathymetric data viewer website (`https://coast.noaa.gov/dataviewer`). 21

2.2 Standard workflow in OceanMesh2D. 23

2.3 Panel (a) illustrates an example of a mesh size function, which is used to create the triangulation depicted in panel (b). Panels (c) and panels (d) demonstrate how a coarser minimum element sizes naturally lead to a simplification of the meshing boundary (red lines). 30

2.4 The geometric triangle quality q, Eq. (2.13), as a function of iterations in the mesh generation process for the three mesh examples (Fig. 2.1 and Table 3.1). The dotted and solid lines indicate the progression of quality metrics with the mesh improvement strategies turned off and on, respectively, during mesh generation. At the end of mesh generation using Eq. 2.14 to determine termination, a secondary round of mesh improvement strategies are applied and the resulting quality after this step is indicated by the colored asterisk. In each panel, the dotted vertical black line demarcates when the mesh generation process finished. 33

2.5 Mesh triangulation within the JBAY example before and after different stages of the *Make_Mesh_Boundaries_Traversable* function enabling mesh traversability. (a) After initial mesh generation (before entry to function); (b) After deleting offending exterior elements; (c) After deleting offending interior elements; (d) After exit of function once all offending exterior and interior elements are deleted and traversability is obtained. The thick blue line indicates the mesh boundary at each stage, and red patches indicate the elements that are deleted between stages (sub-plots). 36

2.6 Given a Delaunay triangulation composed of vertices p and vertex-to-element connectivity matrix t, return a p and t that has only two traversal paths along its boundaries in which disjoint portions of the mesh that each make up less than a specified area-fraction μ_{co} of the pre-cleaned, initial mesh area (or less than a specified absolute area in km^2 when $\mu_{co} \geq 1$), have been removed 37

2.7 The mesh improvement strategies that are applied in sequence from left to right after mesh generation, with the red ovals denoting areas of change in the connectivity along with the function's name that performs the operation. The top row indicates various regions in the mesh before the improvement strategy, and the bottom row after improvement. Panels (a) and (d) indicate the deletion of elements that share an edge with only one other element (singly-connected elements); panels (b) and (e) illustrate the reduction of the vertex-to-vertex connectivity to an upper bound of six using the algorithms documented in Massey [102]; and panels (c) and (f) illustrate the single-step implicit smoothing operation [13] that is used to maximize the overall mesh quality. 39

2.8 An example of the multiscale meshing technique applied to a set of domains around the New York/Long Island area. The green boxes are specified by the user. The minimum resolution of the outermost green box in each panel is different: (a) it is 1 km, (b) 500 m, and (c) 35 m. Notice how the regions of overlap gradually transition into each other. 40

2.9 Example of boundary treatment in and around New York, United States; the bounding box of the mesh domain, *bbox*, is indicated by the thick dashed black line, the meshing domain is hatched in blue, and the categorization of land boundary types following the text are indicated according to the colored lines. 44

2.10 (a) The GSHHS fine (i.e., GSHHS_f) shoreline centered around New York, United States; (b) a shoreline extracted from mosaicing NCEI Post-Sandy DEM tiles with the GRASS GIS software. 45

2.11 (a) A high-resolution shoreline dataset that is not a polygon and a bounding box to be meshed dotted black lines. (b) The result of the flood-fill operation that converts the polyline into a closed polygon. In panel (a), the red boundary is the mainland and the green boundaries represent island features that are closed by definition. 47

2.12 Mesh resolution sizes in a global mesh that was created using the *OceanMesh2D* software library in a handful of hours automatically. . 49

2.13 Mesh resolution (defined as the local element circumradius [m]) in the PRVI example (see Table 3.1) around the Puerto Rico and U.S. Virgin Island inset region, with (a) and without (b) the Rossby radius slope filter applied. The 'thermal' color palette from cmocean [147] is used in this figure. 55

2.14 A schematic illustrating the channel mesh size function implementation. The thalweg (deepest part of a channel) is depicted by the maroon line. Mesh size function grid points (defined at the free surface vertical contour) that fall within the channel cones (centered along the thalweg with an assumed bank angle of θ from the vertical) used to estimate the width of the channel are set to follow equation 2.26. . . 58

2.15 Panels (a) and (c) show the bathymetry and mesh connectivity in the GBAY example (Fig. 2.1 and Table 3.1) created without the thalweg mesh size function enabled; panels (b) and (d) are with the thalweg mesh size function enabled. The 'deep' color palette from cmocean [147] is used in palettes (a) and (b). 59

2.16 Depiction of mesh resolution interactions between the grade (α_g), *distance* (DIS), and *feature* (FS) mesh size functions. Panels (a)-(c) depict the resolution with a grade equal to 15% ($\alpha_g = 0.15$), panels (d)-(f) with a grade equal to 35% ($\alpha_g = 0.35$). The first column depicts how mesh resolution is distributed with a *distance* mesh size function and the second and third columns show how the mesh size varies with the *feature* mesh size function with α_R equal to 3 and 6, respectively. In the title of each panel, the number of vertices n in the triangulation is shown. 62

2.17 Panel (a) illustrates the effect of CFL-limiting on the Courant number Cr when constructing the JBAY example (Fig. 2.1 and Table 3.1) and (b) without it. The colored bars indicate the vertices with $Cr > 0.5$ and are shown to assist in the comparison of histograms. 65

2.18 Selected closeup regions in Jamaica Bay, New York (left (a) and (c): West Pond, Queens; right (b) and (d): Old Howard Beach, Queens) of the mesh connectivity built with the JBAY example script (Fig. 2.1 and Table 3.1). Top panels (a) and (b) show the mesh connectivity without invoking the CFL-limiter, and the bottom panels (c) and (d) show it when using the CFL-limiting option with $\Delta t = 2$ s. 66

2.19 Illustration of key *msh* methods: *plot* can be used to visualize mesh resolution (top-left), mesh triangulation and boundary types (top-right), and seabed topography (bottom row); *interp* interpolates seabed topography onto the mesh using cell-averaging or built-in *griddedInterpolant* methods (bottom row); *makens* classifies mesh boundary vertices into land and open ocean types automatically using the native *geodata* class (top-right). 69

2.20 An example of automatically merging a regional higher resolution triangulation into a coarser global shell. 71

2.21 Panel (a) documents the steps involved with the two-step approach to automatically mesh the floodplain. Panel (b) shows the oceanside portion of the doman with the boundary edges and points indicated in red. Panel (c) shows the second step in the floodplain meshing algorithm that incorporates the oceanside's boundary as fixed points and edges to mesh to a higher elevation geometric countour. Panel (d) illustrates the interpolated seabed topogrpahy (using a cell-averaged approach) onto the mesh vertices with (left) and without (right) consideration of lowering the depth along the fixed points to ensure they remain wetted that represent the shoreline geometry. 73

3.1 The study area in which colored zones in the center panel indicate the mesh size upper bounds (h_{max}) in the reference (REF) mesh (minimum mesh size is L_{min} = 50 m). The red and green colored zones together indicate the comparison zones for all the cumulative area fraction error curve calculations. The dashed magenta line indicates the open ocean boundary on which tidal elevations are specified. The top left and right panels indicate TPXO9.1 solutions of the M_2 and K_1 tidal constituent elevation amplitudes (colors) and phase contours in intervals of 30° (M_2) and 15° (K_1). 87

3.2 An illustration of the five mesh size functions that were investigated in and around the Mid-Atlantic Bight region along the Eastern United States coastline. Seabed topography is colored and relevant quantities are noted in the text. 95

3.3 Mesh connectivity near Ossabaw Island, Georgia that illustrates changes to the capturing of narrow channel geometries as minimum mesh resolution is increased from 50 m (left) to 250 m (middle), and when using a shoreline width function that varies minimum mesh resolution between 50 m and 250 m (right) automatically based on shoreline geometric properties. 100

3.4 The shoreline geometry error $\mathcal{A}_{error}$, Equation (3.7), on the left axis for the meshes used in the shoreline approximation experiment along with the total vertex count for each mesh on the right axis. Solid lines represent data for meshes created with uniform shoreline resolution Lx and dashed lines indicate meshes created with the feature size approach FSx. 101

3.5 Panels (a)-(b) depict the relative error in the M_2 harmonic elevation amplitude from the REF solution when the minimum mesh resolution along the shoreline is coarsened from 50 m and 250 m. Panels (c)-(d) depicts the relative error (RE) in the M_2 harmonic elevation amplitude from solutions computed on meshes built with the feature size function. Insets around areas described in more detail are shown. 102

3.6 The cumulative area fraction error (CAFE) from the REF solution in the comparison region for panel (a) the M_2 elevation amplitude and panel (b) the K_1 elevation amplitude. The dotted lines denotes solutions computed on meshes that use the FSx design while the solid-lines denote meshes created with the Lx design. 104

3.7 The seabed approximation error $\mathcal{V}_{error}$ (Equation 3.8) on the left-axis (blue x's) as the mesh size gradation is increased from 15% to 35% in increments of 5% while the shoreline boundary is held fixed (i.e., area of domain is constant). The total vertex count N_{vertex} in each mesh on the right axis (dashed red x's). The REF mesh vertex count is demarcated by a black asterisk in the top left corner of the figure. 107

3.8 Panels (a)-(b) illustrate the RE in the M_2 elevation amplitude from the REF solution as the mesh gradation bound is increased to 35% while in panel (b) it is kept low at 15%. Panels (c) and (d) are the same as panels (a)-(b) but for the K_1 elevation amplitude. The 250-m isobath contour is drawn as a magenta line in each panel for reference. Insets are shown to reflect areas that are described in the text. 108

3.9 The cumulative area fraction error (CAFE) in the comparison zone (c.f., Figure 3.1) in panel (a) for the M_2 elevation amplitude and panel (b) for the K_1 harmonic elevation amplitude using the meshes created for the mesh size gradation experiment. 109

3.10 (a) A cross-shelf transect in the MAB region indicated in blue with the asterisk indicating the start of the transect, the magenta line is the 250-m isobath, and the red line is the shoreline; (b) the mesh resolution along the transect for the Sx, REF, and H50 meshes. Panel (c) illustrates the seabed topography along the transect for each mesh. Panel (d) illustrates the difference in seabed topography from each mesh and the REF mesh along the transect. 111

3.11 Panels (a)-(b) depict the M_2 elevation amplitude RE for solutions computed on the Sx meshes. Panels (c)-(d) depict the RE the K_1 elevation amplitude. 112

3.12 The cumulative area fraction error (CAFE) curves in the comparison zone for the Sx meshes. 113

3.13 Panels (a)-(c) show sections of meshes in the Delaware Bay estuary and their interpolated seabed data to demonstrate the effect of variably resolving channelized seabeds. Panel (d) illustrates the cross-sectional profile of a tidal channel that is annotated as a red line in panel (a). . 115

3.14 Panels (a)-(c) illustrate the error in the M_2 elevation amplitudes for solutions computed on meshes that variably resolve drainage networks that approximate tidal channels. Panels (d)-(f) indicate the mesh and interpolated seabed topography onto the mesh vertices. On panels (d)-(f), the total vertex count is indicated in the visualized portion of the mesh. 117

3.15 The cumulative area fraction error (CAFE) curves in the comparison zone for the Cx meshes. 118

3.16 Standard deviation of the relative error (σ_{RE}) in (a) the M_2 and (b) the K_1 elevation amplitudes for all 15 meshes from the five experiments (Table 3.1). Panels (c) and (d) are the same but for the standard deviation of the dimensional errors (σ_{AE}). Note the differences smaller than the significance threshold defined in this paper *are* shown and that the colorbars are *not* the same between panels (a) and (b). . . . 119

3.17 An estimate of the numerical error calculated via Richardson extrapolation following [128] obtained by refining the L250 mesh using a four-to-one refinement strategy to preserve the approximate problem. 122

3.18 Elemental resolution distribution in the COMBO3 mesh, highlighting how fine resolution is targeted in narrow geometries and along seabed gradients and estaurine channels (see inset in Fancy Bluff Creek). . . 125

3.19 Panels (a)-(c) depict the error in the M_2 elevation amplitude solution that was computed on the COMBOx meshes. Panel (d) illustrates a CAFE plot of the error in the comparison zone for the three COMBOx meshes. 126

3.20 A comparison of the tidal constituent root-mean-square-error (E) for the M_2 tidal elevations at 439 tidal gauge observations (c.f., Section 3.3.1.2) between a solution computed on the REF mesh (x-axis) and the COMBOx meshes (y-axis). The normalized bias (B) and spread (γ^2) error metrics and the total vertex (N) are indicated. Points that fall in the blue shaded region have a smaller value of E than the REF solution. 127

3.21 The complex root-mean square error (E) for the solution computed on, (a) the COMBO2 mesh, and (b) the COMBO3 mesh (includes enhanced resolution along estuarine channels), at 108 tide gauges in the Chesapeake Bay and Delaware Bay estuaries that are illustrated in panel (c). Various error metrics are indicated in the panels (a) and (b). 129

4.1 Panel (a) is an illustration of a high-resolution finite element mesh in which the bathymetry/topography is shaded that is used in the computation of coastal flooding. In panel (b), a histogram illustrates the number of degrees-of-freedom (DoF) per bathymetric depth range below sea level for (a). For the purpose of this figure, the overland category is classified as greater than 0.10 m above sea level, nearshore is less than or equal to 0.10 m and greater than -50 m, and deep is less than or equal to -50 m. 139

4.2 An illustration that depicts some concepts in our load balancing approach. In panel (a), the time evolution of the computational load is shown for a Hurricane-driven coastal flooding event. The shaded grey bars indicate epochs in which a fraction of the total number of DoFs are turned offline. In panels (b) and (c) the computational domain with blue denoting portions of the domain underwater and green overland is shown before the Hurricane (b) and after the Hurricane makes landfall (c). 145

4.3 A triangular finite element. Triangle-based connectivity showing the vertex adjacency in solid black lines and the triangle adjacency otherwise referred to as the dual-graph in dotted-black lines. 146

4.4 The parallel mesh partitioning algorithm. *proc* stands for processor. . 147

4.5 A local communication stencil is indicated in panel (a). The subdivision between each processor's data is demarcated in red. In panel (b) the ghost zones that are composed of duplicated elements and vertices on each processor's boundaries are shaded in a contrasting color. . . 150

4.6 A schematic of loop rearranging applied on a patch of elements. The elemental rearrangement leads to the vertex rearrangement since the three vertices of each element that are online must be iterated over in vertex-based loops. The effect on the elemental arrays and vertex arrays on the ordering through color highlights along with the global ID numbering. The portion of the array that is not looped over in the new reordering is referred to as "clipped" and these data are shaded in light blue. The vertical black bar denotes the end of the array. . . 153

4.7 The rigid lid problem. Panel (a) shows the model's bathymetry on top of the triangular elements. The elevation specified boundary is indicated as a thick dashed black line. The hatched region indicates the areal extent of the inundation. Panel (b) shows a close-up of the wet/dry (W/D) boundary the buffer configurations that were tested. 158

4.8 Panel (a) depicts the vertex distribution on a 24 processor decomposition with BUFDIS*inf* configuration and (b) with the BUFDIS2 configuration. Panel (c) and (d) indicate the extent of subdomains each owned by a processor. 159

4.9 Timing statistics for the rigid lid problem as a function of processor configuration and size (number of elements) of the separation between the online and offline portions of the computational domain (BUFDIS). Panel (a) shows the total simulation time T, panel (b) the time spent performing only the computation T_C, panel (c) shows the time spent rebalancing T_{RB}, and panel (d) shows the speed-up over the static calculation. 161

4.10 Timing statistics for the rigid lid problem as a function of number of vertices (degrees-of-freedom) per processor. Note that for ADCIRC+DLB, the number of vertices per processor is defined as the average number of online (i.e., non-clipped) vertices over all epochs (see Section 4.3.5). The average number of vertices per processor includes halo ghost zone vertices. 163

4.11 Panel (a) shows the simulation average surface-to-area volume SV (Eq. 4.3). Panel (b) shows the simulation average load imbalance R_{IMB} (Eq. 4.2). 164

4.12 The (a) mesh's topography/bathymetry with the region of interest around the Outer Banks of North Carolina indicated by the red dotted box. In panel (a) the solid blue indicates the open ocean boundary where elevation forcing is applied. In panel (b), the simulated maximum free surface elevation above the geoid during the simulation of Hurricane Irene is shown along with the location of eight NOAA gauges (black squares) and 118 USGS high water marks (blue diamonds). The location of the model's extent is indicated by a green line and the track of Hurricane Irene is annotated as a thick red line in all panels. . . . 167

4.13 Time series comparisons of total water levels at the eight National Oceanic Service (NOS) gauges (Figure 4.12). Observations are shown at a solid black line where available. 170

4.14 Agreement in computed high water marks between ADCIRC and ADCIRC+DLB as compared to 74 USGS high water marks (m). 171

4.15 Panel (a) shows the total wall-clock time T simulating Hurricane Irene as a function of number of processors and for the five buffer configurations based on depth. Panel (b) shows T_{RB}, and panel (c) shows the speed-up. 172

4.16 Illustrates the estimated speed up as a function simulation day. Rebalancing events are indicated by the colored markers. The solid black vertical lines indicate the ending up of the ramping period and the time of the hurricane's landfall in North Carolina. 174

4.17 Panel (a) shows the simulation average surface-to-area volume SV (Eq. 4.3). Panel (b) shows the simulation average load imbalance R_{IMB} (Eq. 4.2). 175

4.18 Panels (a) and (b) show the number of DoFs per processor classified as either wet, dry, and offline. The processors that own the maximum number of DoFs are circled in black. Panel (b) uses the BUFDP=0.1 m configuration. 176

4.19 Panel (a) shows the percent anomaly in load from the mean load. Panel (b) shows the region indicated by the black box in panel (a). Panel (c) illustrates the mesh connectivity in the region that contains the largest load imbalance. 177

4.20 Panel (a) shows the speed up associated with the BUFDP=0.10 m simulation when the singly-connected elements are removed from the NC9 mesh (these elements were illustrated in Figure 4.19). Panel (b) illustrates the simulation average load imbalance. 178

A.1 The time required to initially partition a mesh for the number of processors labeled on the x-axis with 14k weirs (a) and 8K weirs (b) using algorithm 2 as compared to the serial algorithm. 193

TABLES

2.1 THREE EXAMPLE MESHES. 20
2.2 WALL-CLOCK TIMES TO BUILD EXAMPLE MESHES. 75

3.1 EXPERIMENTS WITH MESH SIZE FUNCTIONS. 94

ACKNOWLEDGMENTS

A great deal of knowledge and experience has been aquired here at Notre Dame and there are many people that I would like to acknowledge. Above all else, I would like to thank my advisor, Dr. Joannes Westerink, for his guidance, patietence, and support of my work. It has been a rewarding journey.

Many thanks to Diane Westerink for carefully listening and offering her wisdom and insight into all matters of life.

I would like to greatly thank Dr. Casey Dietrich who seeded new ideas and then watched patiently as I both failed and succeed in developing them. You have been a great mentor! I've learned so much from our weekly calls.

I would like to thank Dr. William Pringle for our thoughtful discussions.

I would like to thank Dr. Damrongsak Wiraset for offering his knowledge and perspective on my work.

A special thank you to all my past and present colleagues at the Computational Hydrualics Laboratory especially Dr. Brian Joyce, Marite, and Joaquin. You guys are (were) wonderful colleagues that made me always eager to come into work, share ideas with, and dicuss life.

Thank you to Dr. Andrew Kennedy and Dr. David Richter for serving on my dissertation committee. I hope this work meets your expectations!

I would like to acknowledge my outstanding support system of family members. My mom and dad in particular I cannot thank you both enough.

Lastly, I am incredibly lucky to have a thoughtful and loving partner, Mariana, who has helped me in many not so obvious ways accomplish the work in this document.

CHAPTER 1

INTRODUCTION

1.1 Overview

Coastal flooding associated with tropical cyclones and other storms are a major cause of loss of human life and property in many coastal cities in the United States and around the world. The recent 2017 Atlantic Hurricane season resulted in \$369.89 billion USD in damages along the United States eastern seaboard and Puerto Rico, which was the costliest hurricane season to date for this region[142]. Coastal flooding will only become more frequent in the future. Multiple studies have demonstrated that the steady increase in mean sea levels will lead to a marked increase in the frequency of severe and nuisance flooding events [104, 130, 155]. The global mean sea-level is rising at an estimated rate of approximately 1.3-2.0 mm/year [43, 111]. Satellite altimetry data demonstrate that the global mean sea level rise is accelerating at a rate of 0.084 ± 0.025 mm/y^2 (since 1993) due to human-induced global warming [109]. Given that coastal cities are often densely populated [14] and are vulnerable to coastal flooding from hurricanes due to their low-lying topography, modeling capabilities are essential to both understand and predict coastal flooding in order to mitigate damage and loss of life. These modeling capabilities need to be both accurate and fast in order to successfully aid in disaster preparedness and timely preemptive measures.

Coastal flooding is forced by multiple hydrodynamical processes. The major processes concern the astronomical tides, the mean sea level state, storm surge, wave

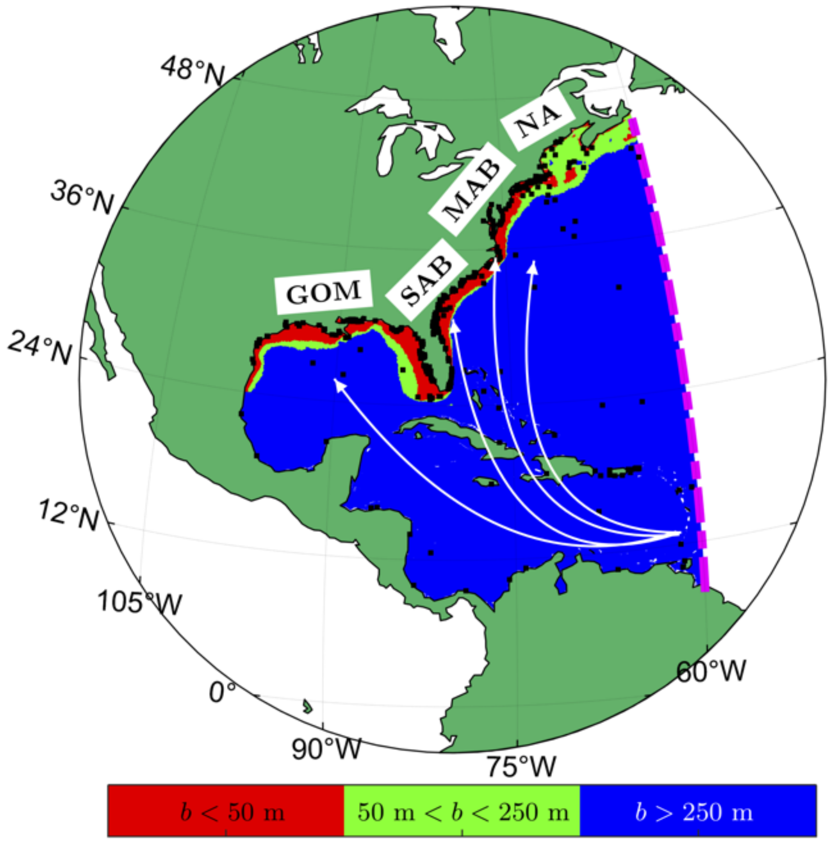


Figure 1.1. The western North Atlantic study region that was explored in this work. The continental shelf region is shaded in red and green. The typical track of a tropical cyclone (i.e., hurricane) is indicated by the white arrows and the various regions of interest described throughout the text are labeled (GOM: Gulf Of Mexico, NA: North Atlantic, SA: South Atlantic, MA: Mid-Atlantic).

setup and run-up, and river discharge. Altogether, these processes can result in an abnormal rise of water levels leading to flooding on occasion. The primary driver of total water level variations are the astronomical tides, which are externally forced by the periodic motion of nearby celestial bodies (i.e., the Moon and the Sun). These nearby celestial bodies exert a gravitational effect on the water column. The largest tidal constituents are governed by the Moon's orbital position (i.e., angle of declination, eccentricity of orbit, etc.) around the Earth and have periods of approximately 12-h [114]. Thus, the tides have wavelengths hundreds of kilometers in horizontal lengthscale and travel at rapid speeds in the ocean. However, nearshore the shallow seabed topography and irregular shoreline geometry largely control the tidal dynamics and form, and these aspects have been extensively studied [91].

In addition to the tides, surface meteorology also explains a sizable component of total water level variance. The meteorologically forced component of water levels this work is concerned with are referred to as storm surges. Storm surge is mathematically defined, under the assumption of tide-surge linearity, as the difference between the observed total water level and the water level that would occur solely from tidal processes relative to a geodetic benchmark (e.g. local mean sea level). Surface meteorology associated with the nearby passage of storms can generate surge events specifically when prolonged surface wind stresses impart momentum into the sea surface pushing a column of water onshore and low pressure anomalies raise the water surface elevation through the inverted barometer effect [114, 127]. Surge events are frequent and intense on wide and shallow continental shelves that intersect with the climatological storm track such as the western Gulf Of Mexico and the Southern Atlantic Bight (Figure1.1; [107, 127]). Occasionally, the episodic rise in water levels associated can lead to catastrophic coastal flooding producing water levels that range from 3-m to 6-m, such as during Hurricane Ike [78], Hurricane Gustav [51], and Hurricane Sandy [40].

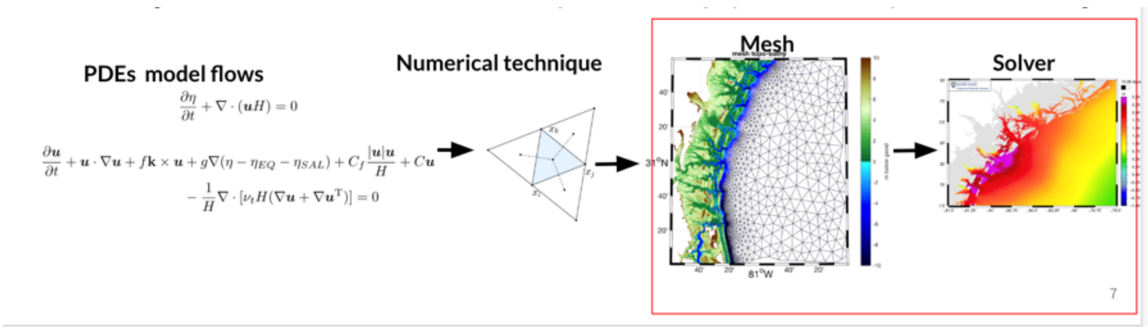


Figure 1.2. An overview of the two-dimensional (2D) mesh-based coastal modeling approach.

Given the substantial damage storm surge events can have on coastal communities, there has been a vast amount of research dedicated to understanding and predicting coastal water levels using computer simulations. One avenue of this research concerns the topic of numerical simulation. Continuous partial differential equations (PDEs) that describe the dynamics of fluid motion are derived from basic physical principles and are discretely approximated on tessellations of space called grids or meshes using numerical methods (e.g., Figure 1.2). Scientists and engineers utilize numerical methods to approximate coastal ocean hydrodynamics by representing the coastal domain using a mesh of polygonal elements. Programs such as the ADvanced CIRCulation Model [96, ADCIRC] are widely used to solve the PDEs that describe the coastal ocean circulations and contain countless options to improve the quality of solutions and incorporate a variety of physics. In the work this dissertation is concerned with, the simulation of coastal hydrodynamic processes involves generating a mesh for a regional or even global ocean domain using geospatial information such as a shoreline boundary and seabed topographic data and using this mesh to calculate a solution with a solver like ADCIRC.

Accurate coastal flooding requires mesh elements of an adequate size that can approximate the shoreline form as it exists in nature. The horizontal length scales of relevant coastal geometry vary by nearly four orders of magnitude (e.g., 10 m to 100 km) and have a fractal geometry [82] making the mesh development process challenging. Narrow waterways and restricted channels on the order of 10-m are encompassed in larger regional/oceanic domains that may span 100's of kilometers (see Figure 1.3). The highly multiscale distribution of mesh resolution is necessary to facilitate larger regional and global ocean model domains, which enable the full extent of surface meteorology and non-local surface meteorological forcing associated with tropical and extratropical cyclones to be resolved without the need for many smaller individual models and nesting paradigms [19, 69]. Mesh resolution on the order of 10-m and below is necessary to represent flood-control structures such as weirs, dikes, coastal groins, and piers, which are all important for the accurate simulation of street-level flooding [141, 148]. Thus, for simulations of coastal ocean hydrodynamics, unstructured meshes are heavily favored over structured grid approaches as they can flexibly vary in size to efficiently conserve computational resources while conforming well to the irregular shoreline geometry.

A number of practical decisions must be made in order to design a coastal ocean model. These decisions concern how the available geospatial datasets that describe the shoreline boundary, namely the estuaries, back-bays, and overland components of the domain are represented in the mesh. The geospatial data used to guide the mesh development process are often disseminated as structured horizontal grids called Digital Elevation Models (DEMs). DEMs may have orders of magnitude disparity in horizontal resolutions and limited/sparse spatial coverage. In the last two decades however, technological improvements to Light Detection and Ranging (LiDAR) technology have created detailed scans of the coastal margins that now exist in the public domain with horizontal resolutions on $\mathcal{O}$(1-m). While the accuracy of the geospatial information is continually improving and datasets have become more accessible to the public, the size and format of the datasets introduce new challenges involved with

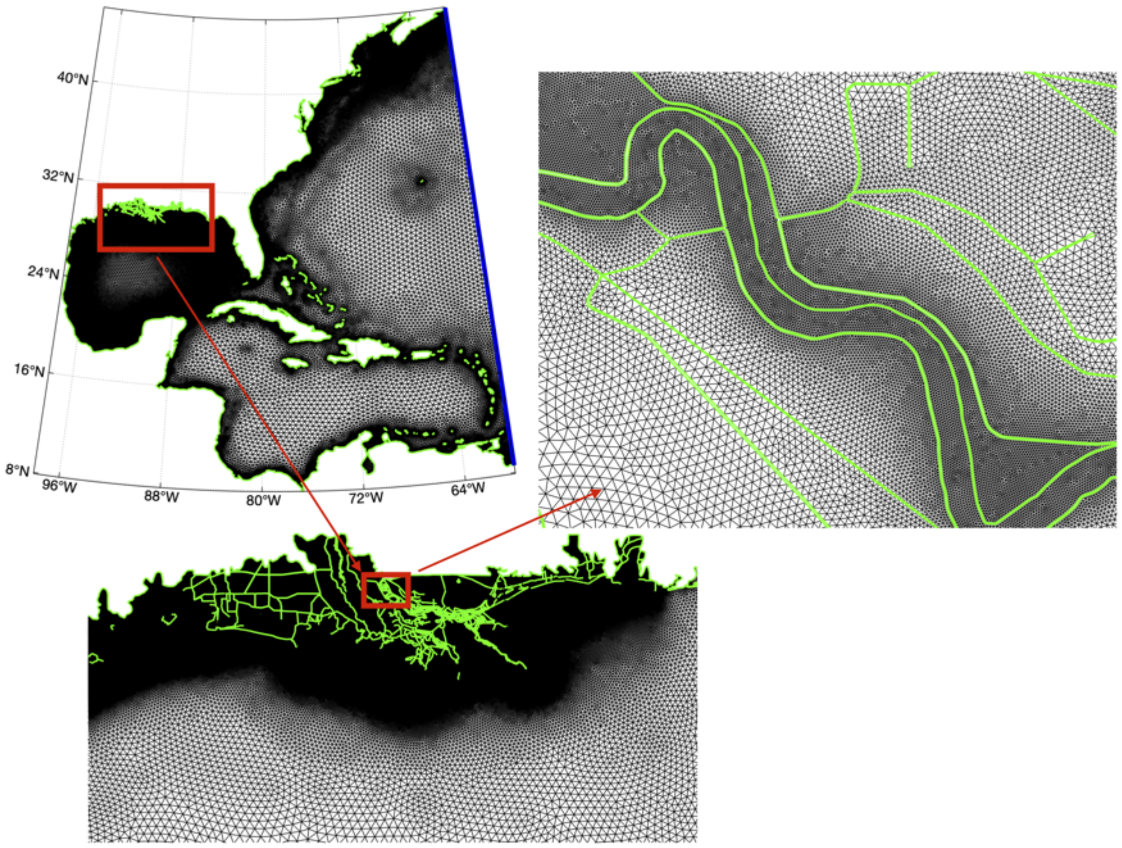


Figure 1.3. The triangulation of a typical regional coastal model at various cartographic scales. The green lines in the insets indicate the geometric complexity of the internal type barrier structure that is modeled.

the mesh development process for coastal modeling. These challenges are related to the simplification of the often excessive details contained therein. In Chapter 2, an approach is devised that to utilize LiDAR datasets for coastal modeling development.

In the current state, scientists and engineers hand-craft meshes by manually drawing polygonal regions that define mesh refinement zones and shoreline boundaries, which hinders scientific reproduciblity. The variations in mesh resolution may be guided with modeling experience [29, 85], considerations of dynamical processes [135, 136], geometrical and topographic-based considerations [41, 57, 129], or some combinations of all of the above.

Mesh developments often take place within a Computer Aided Design Graphical User Interface (CAD-GUI) program, such as the Surface Modeling Systems (SMS) software [162]. Often, resolution in the deep-ocean resolution is prescribed through a tidal wavelength-to-gridscale criteria and finer resolution is often prescribed on the continental shelf break zone [98, 153]. However, the effect resolution heuristics have on the simulation of coastal hydrodynamics has not been systematically explored. Thus, the selection of meshing heuristics vary greatly between study-to-study.

There are some fundamental problems with the pre-existing heuristics (i.e., the so-called "wavelength-to-gridscale heuristic") as they tend to resolve the entire continental shelf zone with fine resolution over-discretizing the coastal domain. The topographic-length scale heuristic to resolve bathymetric gradients with finer resolution also tends to produce exceedingly fine resolution in shallow depths (i.e., depth $\rightarrow 0$) resulting in uniform resolution along shoreline boundaries. Given the range of existing meshing heuristics, the utility and scientific basis of the meshing criteria utilized in the near-shore and shelf areas of coastal modeling domains need to be explored.

Nearshore the issue of mesh development becomes more confused and arbitrary as users prescribe zones of uniformly small sized elements in what are perceived as "critical" regions to resolve shoreline geometry lengthscales. Inevitably this leads to numerical instabilities and a great deal of difficulty in incorporating advective acceleration terms into the calculation because the manual application of resolution cannot readily consider the Courant number. As a result, the user is forced to hand-edit the resolution zones of the triangulation and raise/lower/smooth the bathymetry in an iterative fashion to achieve a more stable and, hopefully, more accurate solution. The manual approach to building coastal mesh designs are slow and arduous taking months to years to develop a stable and accurate triangulation.

Automated approaches are thus essential to further research efforts in coastal modeling applications that use unstructured meshes. These automatic approaches can potentially dramatically reduce model development times and the associated uncertainties with the application of unstructured mesh resolution. Therefore, this work makes the necessary algorithmic developments to support automatic mesh genertion for coastal ocean circulation modeling. Implicitly, the work also demonstrates that automated mesh generation technology has reached a level where users can largely forgo the manual supervision that was once necessary for high-fidelity coastal mesh development. The ease and drafting of new mesh designs enables more general science questions to be asked: how does the simulation of barotropic tides respond to the variably representing shoreline geometry and seabed topography? What are the sources of error in tidal solutions when variably resolving the shoreline and how can they be reduced? While it's well known that poor choices at the initial mesh design stage may compromise the predictive accuracy of simulated water levels due to numerical errors [73], a thorough assessment of commonly employed mesh design practices in a realistic domain and problem configuration has not been explored with a great level of detail or rigor.

High-fidelity meshes of the coastal margins rapidly become prohibitively expensive to compute serially as the study region expands to a regional domain or the size of elements reduces. Therefore, parallel processing techniques and high-performance computers are essential for the development of multi-million degrees-of-freedom (DoF) regional coastal hydrodynamic models. For example, recently a 9M DoF mesh of the Indian and Pacific Ocean has been created requiring $\mathcal{O}(1000 \text{ ranks})$ to process tidal results in wall-clock days[122], and represents one of the largest 2D coastal models developed to date in our community. Parallelism is enabled by decomposing a mesh into subdomains and assigning each subdomain to a processor achieving a state where each processor is responsible for an equal amount of computational work.

To partition the work between processors, unstructured meshes are often represented as an undirected graph whereby the centroids of the elements represent nodes (i.e., dual graph), the connection of the nodes (i.e., edges) of the graph represent communication patterns and data dependencies [77]. The goal then is to partition the graph so that the number of edges cut by the subdomain boundaries is minimized while distributing roughly equal computational work to all the processors. While the partitioning problem is NP-hard [58], graph partitioning algorithms exist [e.g., 83, 117, 133] that utilize efficient heuristics to solve the problem and have enjoyed success in practice.

The performance of parallel simulations is critically dependent on the assignment of computational work to processors in a distributed computing environment, which is referred to as load balancing [46]. In applications with static computational workloads (i.e., when the number of computations does not change in time or space), load balancing is accomplished via a pre-processor call to an application that assigns mesh components to processors before the computation begins. Since the work distribution doesn't change, this step is done once allowing the pre-processor to use paradigms that may be inefficient and depend on file I/O. This type of approach to load balancing is referred to as static load balancing. In contrast, methods such as adapative mesh refinement strategies [64, 92] have computational workloads that change during the course of simulation and often in ways that are not not known ahead of time. These type of applications require that load balancing occur during run-time to maintain an efficient parallel process, hence the usage of the word *dynamic*. However, dynamic load balancing (DLB) techniques present software engineering hurdles because: 1) they require that the pre-processor become a component of the application and execute during runtime rather than before it and 2) sometimes the load must rebalanced frequently during the course of the simulation to maintain a target performance level. As such, components such as file I/O and serial algorithms that were acceptable in static load balancing must be avoided. The success of a dynamic load balancing application is especially sensitive to inefficient algorithms since the speed of the parallel application depends on its slowest component.

In the context of coastal ocean modeling, meshes are often created with large floodplains composed of high resolution elements primarily to research coastal inundation from wind-driven, hurricane-forced storm surges, for real-time predictions of flooding, and the long-term design for coastal flooding mitigation [e.g., 51, 60, 78, 134]. Due to the effort involved in validating the mesh and demonstrating its accuracy and stability in realistic flooding scenarios, typically only one or two meshes are created, built, and then validated against observations for a given coastal ocean domain. Considering the difficulty and work involved with designing high-resolution coastal meshes, scientists and engineers build these models in such a way to account for a variety of categorically severe flooding, which results in a large quantity of DoFs overland in a dry-state. Further, due to the fact that coarser mesh resolution sizes ($>$100-m) produce inaccurate inland flooding and flood swaths can often extend up to 10-m above sea level [84], inevitably this leads to approximately 30-70% of the total mesh remaining in a dry-state for the majority of the simulation.

In ADCIRC and in many other 2D coastal hydrodynamic solvers, overland DoFs that are in a dry-state are masked in the calculation by multiplying by zero as the system of equations are prepared for calculation [47, 103]. Thus, the dry-state component of the mesh are involved with an equal amount of computational work as the wet-state DoFs despite their trivial zero-valued solution. In existing codes such as ADCIRC, the removal of these dry-state DoFs from the problem leads to a time-evolving, dynamic computational workload since the wet- and dry-state DoFs are continuously switching in flooding events. Overall, to develop dynamic load balancing capabilities in ADCIRC with the goal to elminate/reduce dry-state DoFs from the calculation requires a significant amount of modifications to pre-existing static solvers.

In order to facilitate dynamic load balancing in an application such as ADCIRC, the software design must support the ability to relocate data and its associated dependencies during runtime. Since these movement patterns of data are not known before the simulation, fast and efficient unstructured communication algorithms need to be used that can scale with the application [e.g.,Rendezvous algorithm; 120]. Developers also need tools to efficiently manage the location of data dependencies as dynamic load balancing problems constantly change local array sizes due to the movement of data between processors. In a distributed computing environment, determining ownership of data is a non-trivial problem. Altogether, these difficulties force the user to program the support algorithms into their implementation, which can be a time-consuming and unproductive task. For this exact reason, software toolkits like Zoltan [24], Charm++ [81], and PetSC [12] have been developed. Often these packages are combinations of algorithms written in C and C++ and are often employed in parallel unstructured and/or adaptive finite element computations. In this work, I extensively use the Zoltan toolkit [24] to provide essential functionality in the developed application.

As the models of the coastal margins continue to grow in size and complexity, it is vitally important to make efficient usage of the expanding computational resources at our disposal. In order to reduce the cost of modeling the floodplain in regional coastal models, I have developed a new capability in the ADCIRC shallow-water equation solver to Dynamically Load Balance (ADCIRC+DLB). ADCIRC+DLB redistributes the computational workload during run-time to reflect time-evolving flooding in the computational domain. It integrates the once static pre-processor into the ADCIRC solver but with vastly more efficient parallel processing algorithms.

A great deal of work has enabled ADCIRC+DLB to contain the majority of features as the static ADCIRC suite, with the key exception for the SWAN wave-coupling ability [48]. Due to the design of the software, the usage of the DLB capability does not require any additional alteration of pre-existing meshes; however, it is important that the mesh is of sufficient geometric quality as is later pointed out. The dynamic case of load balancing is a general case of the static one, so ADCIRC+DLB can be used to accelerate the pre-processing operations by a large factor over the static pre-processor in cases that models contain many boundary conditions. Overall, the load balancing capability is shown to reduce the cost of a real world calculations of coastal flooding with a reduction in wall-clock time observed that is proportional to the number of DoFs in a dry-state. Perhaps most significant is that ADCIRC+DLB enables a new paradigm of mesh design whereby a nearly uniform resolution can be applied overland to model to represent the floodplain as the associated overland compute cost is significantly reduced.

1.2 Layout of Dissertation

This work is concerned with improving our coastal modeling capabilities by (1) making the mesh development process more reproducible, (2) reducing the time and effort spent in developing high-resolution coastal models, (3) improving the speed of the calculation of the total water levels by dynamically removing the dry-state DoFs from the mesh.

In Chapter 2, I describe the development of an open-source library of codes called *OceanMesh2D* that makes the pre-existing manual approach to coastal model generation no longer the only option. I demonstrate that this approach is capable of integrating a variety of high-resolution geospatial datasets simultaneously with a number of user-defined and varying sizing and geometry constraints by automatically adjusting resolution sizes according to *a priori* functions of shoreline geometry and seabed topography. A focus has been placed on automation and efficiency when developing regional coastal models using the *OceanMesh2D* library. Particularly, the requirement to adapt shoreline boundaries that may contain excessively fine details to the user-requested distribution of vertices before mesh generation is no longer required. The shoreline adaption step now occurs automatically in the mesh generation process, which greatly helps the practical usability of the automated mesh generation approach for solving engineering problems quickly. Chapter 2 in an abbreviated form has been accepted to the journal *Geoscientific Model Development*.

In Chapter 3, I apply the *OceanMesh2D* library to the development of many realistic, high-resolution (minimum resolution 50-m), unstructured triangular meshes that are used for the forward simulation of barotropic tides along the East and Gulf coasts of the United States. These experiments are used to research the utility of the developed automatic mesh generation tools and to understand how to more efficiently develop high-fidelity meshes for the prediction of total water level. The main findings of this study demonstrate that the rate of mesh size expansion along the continental shelf margins can be substantially enlarged but this requires the targeted placement of resolution along bathymetric gradients. The overall result is a lightweight mesh with roughly one order of magnitude fewer vertices than the initial 10.8M vertex reference solution and that has a comparable accuracy in the simulation of tidal harmonics.

Among many other findings, an automatic mesh size function that resolves estuarine channels is investigated. The usage of this new mesh size function is shown to support the larger elemental size expansion rate in the inner shelf and nearshore environment where the other meshing heuristics perform poorly and can easily over-resolve bathymetyric gradients due to noise. The resolution along significant estuarine channels helps to maintain the accuracy of the reference tidal solution in the inner shelf and nearshore zones by accurately representing channel morphology and flux of tidal energy into the nearshore estuaries. Chapter 3 in a modified form has been submitted to the journal *Ocean Modelling*.

In Chapter 4, I describe the development and performance of ADCIRC+Dynamic Load Balancing (ADCIRC+DLB). This software dynamically removes the floodplain from the calculation in the ADCIRC shallow water equation solver by re-decomposing the mesh between processors. I document the methods and theory of the approach and then demonstrate the capabilities in a realistic forecasting application with a hindcast of Hurricane Irene in North Carolina. The DLB capability leads to speed ups over the static approach by nearly 45% (approximaately 4% less than the theoretical maximum speed up) with a maximum overhead of less than 2.33% of the total static time. The key advantage of this approach is that the majority of model functionality and options are retained, which enables modelers to run existing modeling systems with the technology. The development of this approach will support the next-generation of coastal hydrodynamic models with the ADCIRC solver that will contain the majority of vertices overland. This chapter will be submitted to the journal *Computers and Geosciences*.

CHAPTER 2

AUTOMATED MESH GENERATION FOR COASTAL HYDRODYNAMIC MODELING

2.1 Overview

In this chapter, I document an approach and the related computational mechanics to generate unstrucutured meshes that are used for the simulation of coastal ocean hydrodynamics. This approach is embedded into an open-source library of codes called *OceanMesh2D* that has been released to the community through version control software and is documented in a user guide [129]. It is currently being used for a variety of ongoing research projects with the financial support from National Oceanic and Atmsopheric Administration, Army Corp. of Engineers, and Factory Mutual Global Insurance who are all very interested in more efficiently and objectively developing unstructured meshes for coastal circulation problems. The hope is that new developers will continue the work as the project should be a community based effort. The approach detailed here represents a significant depature from the existing manual practices that have been and are being used to develop coastal ocean modeling systems. The greatest aspects of difference in this approach are: (1) how geospatial datasets are incorporated into the model development procedure, (2) the efficiency and automation of the model development procedure, (3) the resulting model's numerical stability when simulataion is attempted with it, (4) and model reproduciblity. This chapter, in an abbreviated form, has been accepted to the journal *Geoscientific Model Development* [131].

2.2 Introduction

Many phenomena in the coastal ocean, such as tides, tsunamis and storm surges, can be accurately modeled by the shallow water equations. Unstructured meshes are often used for numerical simulations of the coastal ocean because they can resolve the large range of horizontal length scales necessary for accurate hydrodynamic predictions and can conform well to complicated shoreline boundaries. The accuracy and the associated computational expense of the mesh are in direct conflict, which makes the mesh design process challenging. Computational work is governed by the distribution of vertices (mesh resolution) and accuracy is determined, in part, by the representation of relevant geometrical and bathymetric features that may influence the simulation. Due to this balance between accuracy and computational work, the prescription of the mesh resolution often leads to a highly subjective mesh generation process which is often handled through graphical user interface (GUI) based software, e.g., Surface-water Modeling System [1], Blue Kenue [2] and Delft's Flexible Mesh suite [3]. Although GUIs allow the user to carefully edit detailed aspects of the mesh they do not promote automation, objectivity, or reproducibility. To address this issue, the ocean modeling community have developed approaches and tools to support the automated generation of unstructured meshes for coastal circulation problems [8, 17, 31, 41, 64, 65, 73, 89, 126]. Most works have either tried to minimize topo-bathymetric interpolation error on the mesh [e.g., 64] or construct the mesh based on resolving relevant physical processes in the domain and/or preserving the geometry of the shoreline boundary [e.g., 41, 57]. An iterative *a posteriori* method which aims to keep the local truncation error constant throughout the mesh has also

[1]`https://www.aquaveo.com/software/sms-surface-water-modeling-system-introduction`

[2]`https://www.nrc-cnrc.gc.ca/eng/solutions/advisory/blue_kenue_index.html`

[3]`https://www.deltares.nl/en/software/delft3d-flexible-mesh-suite/`

been employed [73].

Modern interpreter-based programming environments such as MATLAB and Python are attractive to many users to develop mesh generators because they include a plethora of built-in or community developed functions, toolboxes, and packages that are freely available. For instance, a simple and easily adaptable mesh generator based on the concept of force equilibrium and written in a few dozen MATLAB lines is *DistMesh2D* [118]. The simplicity of the force-equilibrium algorithm makes it attractive as a general-purpose mesh generator by allowing users and developers to adapt it for various applications [e.g., 55, 94, 110, 149]. However, due to the general nature of *DistMesh2D*, it tends to be computationally inefficient for the large and highly multi-scale geophysical domains that are encountered in coastal ocean hydrodynamic modeling problems. Additionally, there are a number of pre-processing steps that must be performed to prepare the geospatial data for meshing and a number of post-processing steps to make sure the mesh is amenable for simulation. For instance, one must obtain a shoreline boundary that will lead to a mesh that is practical to simulate with. By integrating the tools to pre-process the geospatial data into the mesh generator directly, it reduces the time spent performing these essential tasks and largely automates the mesh development process.

In a related previous work, the Advanced Mesh generator [ADMESH; 41] implemented a *DistMesh2D* based coastal ocean mesh generator in MATLAB. In this work, we build on many of the ideas described in *ADMESH* with the following primary improvements: a) a focus on computational efficiency to enable the software to become practically useful even for large geophysical datasets(e.g., $\sim$1 km resolution global topo-bathy) in the MATLAB scripting language; b) the inclusion of pre- and post-processing workflows; c) a greater variety of mesh size functions and flexibility in their application which offers more control over mesh resolution placement; and critically: d) code written in an open-source environment for the benefit of the community. The codes place emphasis on facilitating automatic mesh design workflows that lead to the creation of meshes and the necessary model inputs for a numerical simulation. These mesh generation workflows (i.e., a user-specified MATLAB control script) are typically represented by a few lines of MATLAB code and take between minutes to an hour to generate relatively large, multiscale, high-fidelity meshes (potentially global-to-channel scale) and their auxiliary components automatically.

The software to build coastal meshes is written in an objected-oriented framework that is divided into a set of standalone classes related to: 1) processing geospatial datasets used in the mesh generation process; 2) computing mesh size functions; 3) performing the mesh generation; 4) storing, visualizing, and post-processing the mesh output. Special attention has been made to ensure that only open-source functions are required to generate a mesh. Further, in its current state the software contains a number of post-processing functions specific to the ADvanced CIRCulation model [ADCIRC; 96], but these can be adapted to other solvers in the future. The rest of this paper is structured as follows: we begin by introducing the framework and organization of the code followed by a detailed description of each of the four standalone classes and ending with a discussion on how the software can be useful for coastal ocean model development.

To demonstrate the overall workflow and the design of the software, three examples located along the East Coast and Gulf Coast of the United States of America are documented (Fig. 2.1). The first example produces a mesh of the Jamaica Bay estuary in New York (JBAY), demonstrating the utility of the software in incorporating high-resolution ($\sim$1/9-arc second or approximately 3-m horizontal resolution) LIght Detection And Ranging (LIDAR) datasets with fine ($\sim$15-m) resolution triangular elements nearshore. The second example meshes the Galveston Bay in Texas (GBAY), demonstrating the utility of a new mesh size function that can be used to target resolution along deep-draft marine navigation and tidal channels. The third example demonstrates how the software can produce truly multiscale unstructured meshes in less than one hour by building a mesh of the Western Atlantic Ocean with focused refinement around Puerto Rico and the U.S. Virgin Islands (PRVI). See Table 3.1 for details of the various options/parameters that were used to generate these example meshes.

Its important to differeniate what constitutes a “high quality” mesh for this problem, as “quality’ is largely application dependent. Note that a “high-quality” mesh and an “high fidelity” (i.e, a model that produces solutions that agree well with observtions) one are not necessarily the same. Often, mesh quality can be viewed as a combination of geometric element measures, application dependencies, and numerics [138]. For 2D shallow water flows, a high quality mesh is often determined by geometric measures (i.e., nearly all equilateral triangles) with a lower bound on the minimum element quality and the majority of vertices having nearly six edges connected to it [9, 30, 138]. When most elements have a high-degree of regularity in the vertex-to-vertex connectivity, it improves the mesh size transitions (gradation) are smoother, the condition number of finite element coefficient matrices is reduced, and the memory footprint of the finite element solver can be reduced [102]. A high degree of regularity in the vertex-to-vertex connectivity also tends to coincide with a mesh with many equilateral or nearly equilateral triangles. In this Chapter, I am focused on generating meshes with a high geometric quality and in Chapter 3, I explore the effects of variably resolving coastal mesh domains with the hope to produce a high fidelity model.

2.3 Software Architecture Overview

The automated generation of geophysical-use unstructured meshes often requires a number of user defined parameters and a variety of geospatial data as inputs. As a result, the mesh is strongly related to the algorithms and data that were used to

TABLE 2.1: THREE EXAMPLE MESHES.

Region	Meshing Parameters								Mesh Quality		Iterations
	$h0$ (m)	h_{max} (m)	α_R	α_{wl}	α_{slp}	α_g	α_{ch}	Δ_t (s)	$\overline{q_E}$	$q_{E_{min}}$	$\ni q_{L3\sigma} > 0.75$
JBAY	15	1,000	3	–	–	0.15	–	2	0.97	0.60	38
GBAY	60	1,000	3	–	–	0.25	0.10	–	0.97	0.53	71
PRVI	10 & 30 & 1,000*	10,000	5	30	15	0.2	–	0**	0.97	0.45	30

*: Different values of $h0$ are used for each separate mesh size function domain as indicated in Fig. 2.1 (PRVI)

**: Setting $\Delta t = 0$ invokes the automatic time step selector option (see section 2.4.4.7)

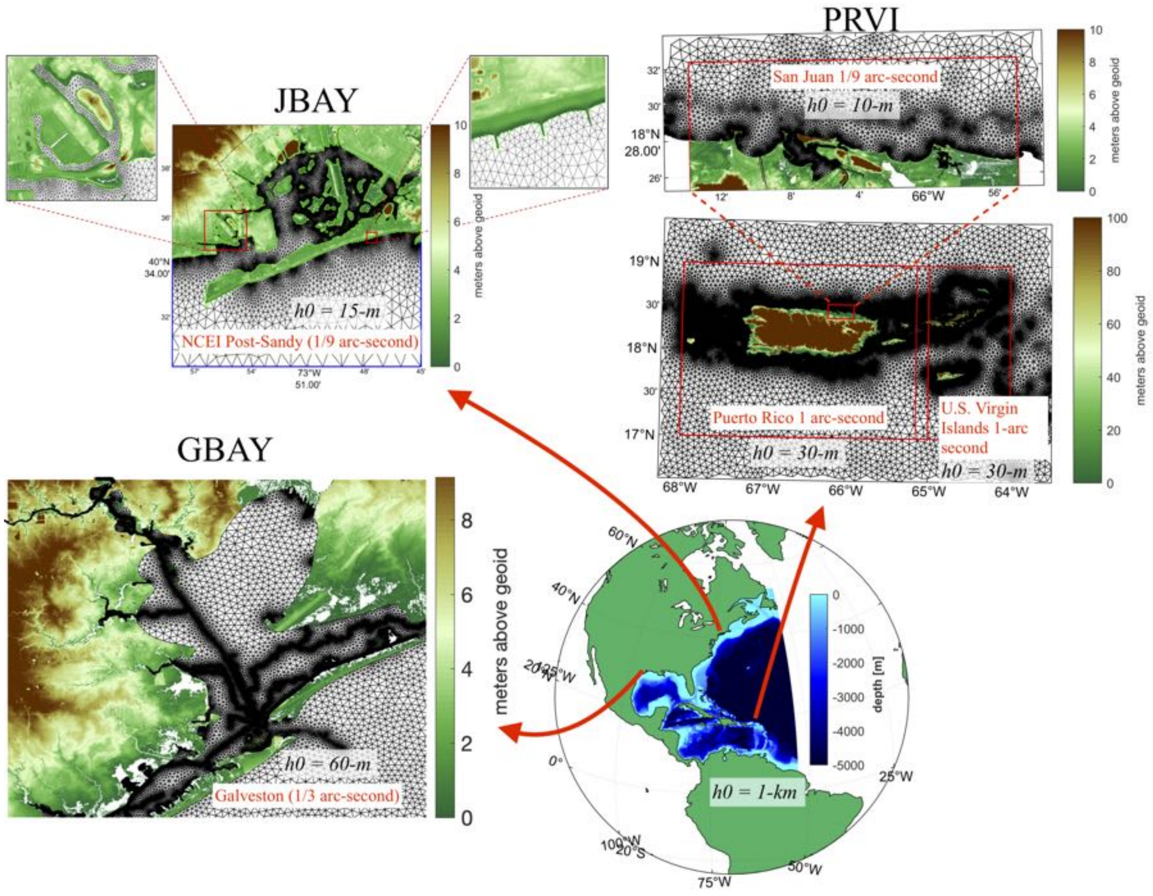


Figure 2.1. The geographical location and triangulation of the three meshes used as examples in this work. The minimum mesh sizes (*h0*) are annotated in black text and the names of the digital elevation models (DEMs) used in the construction of the mesh size functions are annotated in red text on each panel. The colormap indicates topographical data (bathymetric data was removed for production of this figure) in the DEMs, which are freely available through the NOAA Bathymetric data viewer website (`https://coast.noaa.gov/dataviewer`).

create it. These task- and object- specific properties of the mesh generation process provide the motivation behind the development of an objected-oriented programming (OOP) approach. In this software, the use of OOP leads to automation and promotes the usage of efficient workflows.

OceanMesh2D is composed of four classes (*geodata*, *edgefx*, *meshgen*, and *msh*) and a utilities directory containing various standalone functions. The *geodata* class is used as a pre-processor to mesh generation and creates an appropriate meshing boundary from user-supplied geospatial datasets and inputs. The *edgefx* class enables the user to build standardized mesh size functions with a variety of parameters and constraints. The *meshgen* class is associated with mesh generation inheriting various options from the *geodata* and *edgefx* classes. The *msh* class is a data storage class for the mesh and related attributes. These four classes are constructors for creating specific instances of each class otherwise known as objects. All classes are activated through the use of name-value pairs where the "name" represents an option and the "value" is the parameter relevant to that option.

Although each individual class is standalone, there exists a specific workflow that is typically followed to build coastal ocean meshes with the *OceanMesh2D* software (Fig. 2.2). Numerous instances of the *geodata* and *edgefx* classes can be combined to seamlessly mesh high-resolution insets contained within wider coverage geospatial datasets. The ability to incorporate datasets over a wide-range of scales is particularly useful and pragmatic given the finite computational memory and highly-variable horizontal resolution of available high-resolution topo-bathymetric data.

2.4 Component Design

In the following section, each of the four classes that comprise the approach to coastal ocean mesh generation are described.

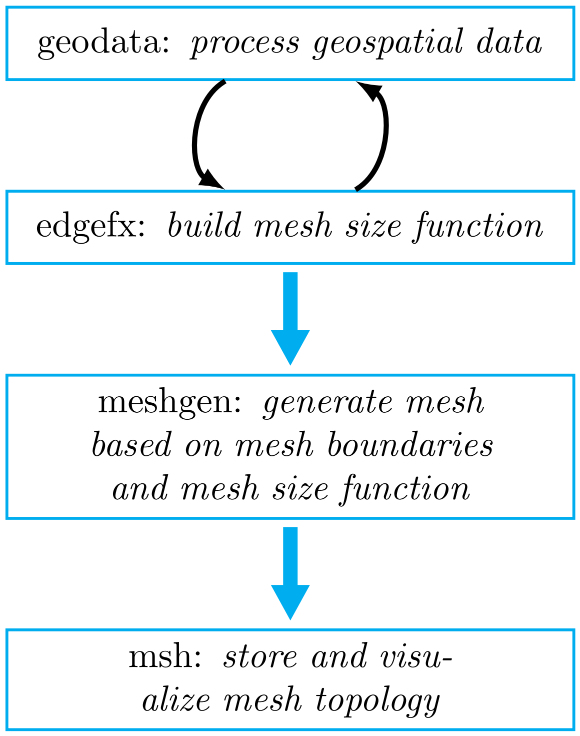

geodata: *process geospatial data*
edgefx: *build mesh size function*
meshgen: *generate mesh based on mesh boundaries and mesh size function*
msh: *store and visualize mesh topology*

Figure 2.2. Standard workflow in OceanMesh2D.

2.4.1 Mesh generation: *meshgen* class

Mesh generation is achieved through the use of the *DistMesh2D* algorithm with a number of modifications to help improve the quality of the final triangulation, the speed of the mesh generation, and the memory footprint of the overall application, which are documented in this section. Without the modifications to the original *DistMesh2D* algorithm, the regional multiscale coastal meshes that I am exploring are not tractable to generate. It's noted that the class-based architecture of *Ocean-Mesh2D* software could additionally support other mesh generation packages besides *DistMesh2D*, such as *JIGSAW-GEO* [57]. In its current state, the class is a wrapper function around the *DistMesh2D* algorithm that automatically uses classes that describe the meshing domain and the mesh size functions.

For coastal mesh generation, a key advantage of using the *DistMesh2D* smoothing-based algorithm over Delaunay refinement and/or Frontal Delaunay mesh generation algorithms is that the boundary is implicitly defined through a signed-distance function. While the boundary of the meshing domain is stored as a set of linear segments, these segments do not represent the boundary of the final mesh as all vertices can move during mesh generation in accordance with the mesh size function. The final mesh boundary that approximates the shoreline is thus dependent on the mesh size function and post-processing strategies that we employ to ensure that there are no self-intersecting boundaries or small disconnected portions of the mesh. Thus, the need for shoreline approximation pre-processing [e.g. 66] to define the mesh boundary as required by Delaunay refinement and/or Frontal Delaunay mesh generation approaches [65] is eliminated. In this section, we document the mesh improvement strategies that occur during the execution of the *DistMesh2D* algorithm that lead to a high-quality approximate representation of the domain and are in congruence with mesh size functions.

2.4.1.1 *DistMesh2D* Algorithm

The *DistMesh2D* algorithm is based on a physical analogy between the edges of a Deluanay triangulation and a two-dimensional (2D) truss structure [118]. In mathematics and computational geometry, a Deluanay triangulation is a type of triangulation in which no vertex lies within the circumcircle of any triangle (so-called "empty circumcircle property") [26]. In the *DistMesh2D* algorithm, equilateral triangles occur when an external force is applied to the edges of the triangles and the vertices of the triangulation are allowed to freely move. In this approach to mesh generation, the external force is varied according to a mesh size function, which enables the generator and thus the user to carefully control the placement of resolution in the type of comptuational domains that are encountered in coastal modeling applications. This algorithm is considered a smoothing-based approach to mesh generation with the general case being the Laplacian operator (i.e., the vertices are moved toward the average of their neighbors). In contrast to smoothing-based mesh generators, restricted Delaunay-refinement schemes [57] and advancing-front type schemes[162] can also be used to generate two-dimensional planar meshes for coastal modeling applications. However, a benefit of *DistMesh2D* is simplicity: the whole algorithm is roughly 30 lines of code and does not require the complex data structures that are necessary to store and edit the triangulation incementally as is the case in other approaches.

Here I briefly describe the mathematical details of the *DistMesh2D* algorithm. I have adopted the mathematical notation that was originally presented in [87]. Let N be the number of vertices in the mesh and $\mathbf{p}$ be the N-by-2 array of vertex locations. The edges of the triangles then relate to bars of a 2D-truss structure and the vertices correspond to joints of the truss. Each bar $e_{ij} = [\mathbf{p}_i, \mathbf{p}_j]$ exerts a varable force $f(\mathbf{p}_i, \mathbf{p}_j)$ that depends of the ratio of its current length $||\mathbf{e}_{ij}||$ to an ideal length $\mathbf{r}_{ij}$ that is described by a mesh size function $h(\mathbf{p})$. A salient aspect of the *DistMesh2D* algorithm is that, the case of a triangulation with the majority (e.g., $> 90\%$) of triangles containing equaliateral angles tends to correspond to:

$$F(\mathbf{p}) = 0 \tag{2.1}$$

where $F(\mathbf{p})$ is a N-by-2 array that describe the x- and y-components of the force exerted by $F(\mathbf{p})$. Equation 2.1 is a stationary solution of the following system of ordinary differential equations given an initial point distribution $\mathbf{p}_0$:

$$\begin{aligned} \mathbf{p}(0) &= \mathbf{p}_0 \\ \frac{d\mathbf{p}}{dt} &= F(\mathbf{p}(t)), t \geq 0 \end{aligned} \tag{2.2}$$

This problem is discretized explicity in time ($t^k = \Delta kt, \mathbf{p}^k = \mathbf{p}(t_k)$) using an Euler forward approach leading to an iterative scheme:

$$\mathbf{p}_i^{k+1} = \mathbf{p}_i^k + \sum_{j \in \mathcal{N}_i} \Delta t f(\mathbf{p}_i^k, \mathbf{p}_j^k) \frac{\mathbf{e}_{ij}^k}{||\mathbf{e}_{ij}^k||}, i = 1, ..., N, \tag{2.3}$$

in which $\mathcal{N}_i$ is the neighborhood of the vertex i or the connected vertices to vertex i and Δt is set to 0.10 seconds. Note that values of Δt greater than 0.10 s exhibit "numerical instabilities" and will not reach the stationary condition (Eq. 2.2). Also note that a variety of alternative timestepping schemes were explored to solve the system (i.e., Runge-Kutta, Backwards Euler) but none outperformed Eq. 2.3 in terms of the time it took to reach an equilibrium state for realistic coastal problems.

The force exerted on each edge of the triangulation in the original algorithm was purely replusive and mimicked the behavior of a linear spring following Hooke's law:

$$f(\mathbf{p}_i^k, \mathbf{p}_j^k) = \begin{cases} r_{ij}^k - ||e_{ij}^k|| & \text{if } ||e_{ij}^k|| < r_{ij}^k, \\ 0 & \text{if } ||e_{ij}^k|| \geq r_{ij}^k \end{cases} \tag{2.4}$$

In the force function evaluation, r_{ij} is computed via the mesh size function h through:

$$r_{ij}^k = \omega s^k h_{mid}^k \tag{2.5}$$

where ω was set to 1.2 following [118] and h_{mid} is the midpoint of e_{ij}. The scale factor s^k was originally the sum of the root-mean square of the measured e_{ij} as compared to the ideal bar lengths r_{ij} (Eq. 2.6) and is designed to inflate the bars so they move move around more quickly in the domain.

$$s_{ij}^k = \frac{\sum_{i,j} ||e_{ij}^k||^2}{\sum_{i,j} (h_{i,j}^k)^2} \tag{2.6}$$

However, to acclerate mesh generation for realistic coastal modeling problems that spanned 2 to 4 orders of mangtitude in horizontal lengthscale, the scale factor as shown in Eq. 2.6 was found to be ineffective and hindered convergence to the stationary condition. Since the distribution of s^k is highly skewed in the case of multiscale coastal ocean mesh generation with locally high resolution elements, the ratio as calculated by Eq. 2.6 would not sufficiently inflate or shrink r_{ij} for the longer/shorter edgelengths in the meshing domain. Instead, a different scale factor was implemented based on the ratio of the median edgelength to the edgelength as described by $h(\mathbf{p})$.

$$s_{ij}^k = \frac{< ||\mathbf{e_{ij}^k}|| >}{< h(\mathbf{p_{ij}^k}) >} \tag{2.7}$$

where $<\dot{>}$ denotes the median over all bars. The ratio of the medians in Eq. 2.7 was found to better approximate the skewed distribution of edgelengths in multiscale coastal meshes than the ratio of the root-mean square edgelengths and more capable of accurately inflating or shrinking the bars.

As was documented in both [87] and [25], the Eq. 2.4 can be improved by utilizing a modified force function that allows for both attraction and replusion between the vertices of the mesh:

$$f(\mathbf{p}_i^k, \mathbf{p}_j^k) = (1 - (\frac{||e_{ij}^k||}{r_{ij}^k})^4) exp(-(\frac{||e_{ij}^k||}{r_{ij}^k})^4) \tag{2.8}$$

The additonal weak-attraction behavior in Equation 2.8 accelerates convergence to achieve an equilibrium state and thus is used in lieu of Equation 2.4.

In this meshing algorithm, the boundary of the meshing domain is implicitly defined through the use of a signed distance function. In the following, I precisely define the notion of a signed distance function. Let Ω be a subset of $\mathbb{R}^2$ and a polygon be a piecewise linear curve composed of a set of vertices $S = (s_0, s_1, ..., s_N)$ listed in consecutive order in which $s_1 = s_N$ otherwise $s_i \neq s_j, \forall i = 1, N$ (i.e., the polygon is not self-intersecting). The mesh domain Ω is then defined as the intersection of a polygon S that represents the shoreline and a polygon represented the bounding box *bbox* or extents of the meshing domain.

$$\Omega = S \cap bbox \tag{2.9}$$

Within Ω, the signed d is defined as a mapping $d_\Omega : \mathbb{R}^2 \to \mathbb{R}$

$$d(\boldsymbol{x})_\Omega = s_\Omega(\boldsymbol{x}) \min_{\boldsymbol{y} \in \partial\Omega} (||\boldsymbol{x} - \boldsymbol{y}||) \tag{2.10}$$

where $|| \cdot ||$ is the standard Euclidean norm in $\mathbb{R}^2$ (i.e., the distance to the boundary of the meshing domain), $\boldsymbol{x}, \boldsymbol{y}$ are points, and the sign function s_Ω is given by:

$$s(\boldsymbol{x})_\Omega := \begin{cases} -1, & \text{if } \boldsymbol{x} \in \Omega. \\ +1, & \text{if } \boldsymbol{x} \in \mathbb{R}^2 \setminus \Omega. \end{cases} \tag{2.11}$$

If $\boldsymbol{x}$ lies within Ω, then $d(\boldsymbol{x})$ will be negative. It follows from the above:

$$\begin{aligned} \Omega &:= \left\{ \boldsymbol{x} \in \mathbb{R}^2 : d(\boldsymbol{x})_\Omega \leq 0 \right. \\ \partial\Omega &:= \left\{ \boldsymbol{x} \in \mathbb{R}^2 : d(\boldsymbol{x})_\Omega = 0 \right. \end{aligned} \tag{2.12}$$

The signed distance function $d(\boldsymbol{x})_\Omega$ serves as a implicit representation of $\partial\Omega$ and is used to form mesh size functions (see section 2.4.4) and during the execution of the meshing algorithm. An example of a signed distance function is illustrated in Figure 2.3. While the boundary of the meshing domain is stored as a set of piecewise linear segments, these segments do not represent the boundary of the final mesh as all vertices can move during generation in accordance with the size function. The final boundary that approximates the shoreline is thus dependent on the mesh size function and post-processing strategies that are employed to ensure that there are no self-intersecting boundaries or small disconnected portions of the mesh exist. Thus, the need for shoreline approximation pre-processing [e.g. 66] to define the mesh boundary as required by Delaunay refinement and/or Frontal Delaunay mesh generation approaches [65] is eliminated. In this section, we document the improvement strategies that occur during the execution of the *DistMesh2D* algorithm which altogether lead to a high-quality approximate representation of the domain and are in congruence with mesh size functions.

The evaluation of the signed distance function for regional coastal domains can quickly become prohibitevly expensive as the nearest distance to the boundary must be computed for every vertex in Ω. To improve the efficiency of the calculation, the signed distance function is computed through a combination of the MATLAB class version [11] of the Approximate Nearest Neighbor (ANN) method [7, 106] (to obtain the absolute distance), and Dr. Darren Engwirda's open source and fast points-in-polygon test (*inpoly.m*) function (to get the sign). The ANN method has high

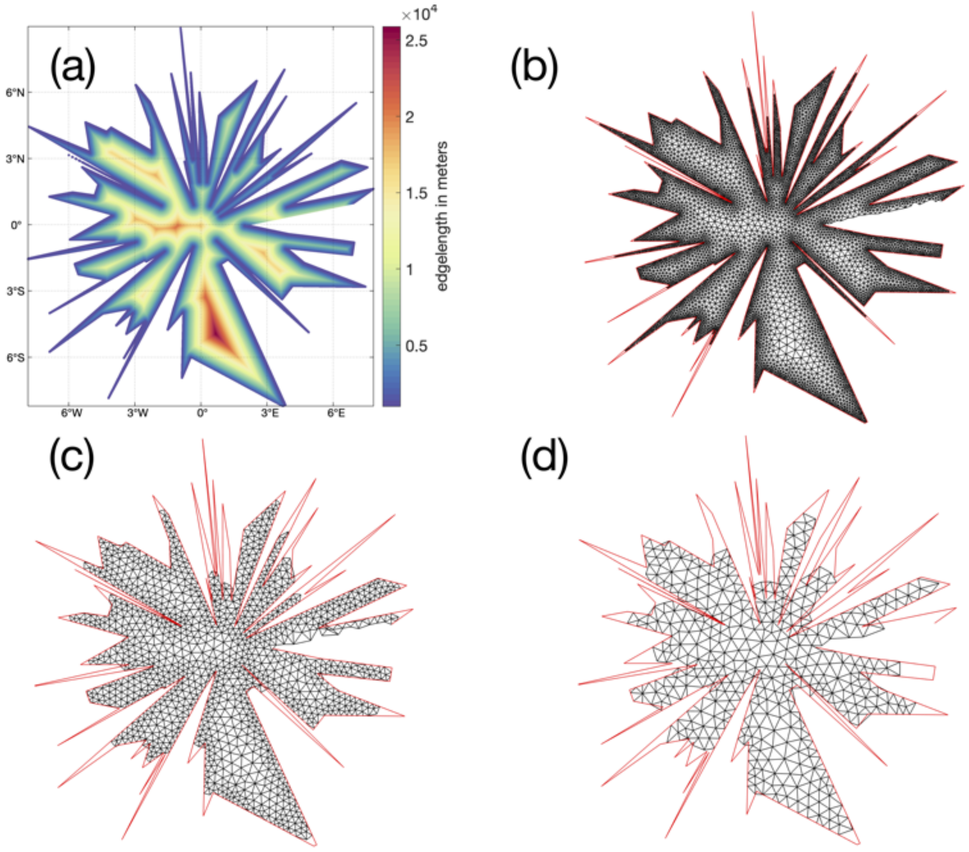


Figure 2.3. Panel (a) illustrates an example of a mesh size function, which is used to create the triangulation depicted in panel (b). Panels (c) and panels (d) demonstrate how a coarser minimum element sizes naturally lead to a simplification of the meshing boundary (red lines).

computational efficiency with a negligible memory footprint in comparison to the *dsegment.m* function in the original *DistMesh2D*, and *inpoly.m* is several hundred-fold quicker ($\mathcal{O}(\log N)$ vs. $\mathcal{O}(n^2)$) than MATLAB's built-in *inpolygon.m*.

2.4.1.2 Termination criterion

Smoothing-based mesh generation approaches, like *DistMesh2D*, have no theoretical guarantees of minimum triangle quality and thus may take a long time to, or may never, reach a desired quality. As a result, heuristics are required to determine when to exit the mesh generation step. In the original *DistMesh2D* algorithm, Persson and Strang [118] proposed a termination criterion based on convergence to a configuration of vertices in which negligible movement of the mesh vertices would occur with additional meshing iterations. In practice, my studies have found that a configuration of vertices with negligible movement is difficult to achieve within hundreds of meshing iterations for realistic coastal ocean mesh domains because of the irregular shoreline boundary and mesh size functions than lead to resolution with orders of magnitude difference in sizes. Thus, an alternative termination criterion is proposed based on geometric element quality.

A geometric measure of triangle equilateral-ness:

$$q_E = 4\sqrt{3}A_E \left(\sum_{i=1}^{3} (\lambda_E^2)_i \right)^{-1} \tag{2.13}$$

where A_E is the area of the triangle and $(\lambda_E)_i$ is the length of the i^{th} edge of the triangle. $q_E = 1$ corresponds to an equilateral triangular element and $q_E = 0$ indicates a triangle that degenerates to a line. A mesh with a sufficiently high minimum bound on q_E is often desired [57, 118, 138]. However, a minimum bound on q_E is a strict measure for large domains with millions of elements and complex shoreline features, and is difficult to achieve within the modified *DistMesh2D* algorithm. Instead, the following termination criterion is used:

$$q_{L3\sigma}(t) = \overline{q}_E(t) - 3\sigma_{q_E}(t) > 0.75 \tag{2.14}$$

where t indicates the meshing iteration, the over-line and σ denote the mean and standard deviation respectively, and $q_{L3\sigma}$ is the "three-sigma lower control limit" element quality, used as a proxy for the minimum element quality. Upon termination through the above criterion, the vast majority (>95%) of triangles are of adequate geometric quality. In the approach used here, a number of mesh cleaning steps are performed *after* this mesh generation termination criterion has been met (Sect. 2.4.1.4) in order to improve a typically small number of the worst quality (Fig. 2.4).

As the release of V2.0 of *OceanMesh2D*, a stricter geometric critera was introducted. This termination critera was based on geometric quality saturation instead of an absolute threshold. Specifically, mesh generation ceases when the following condition is met:

$$|q_{L3\sigma}(t+1) - q_{L3\sigma}((t)| < 0.01 \tag{2.15}$$

where t represents a meshing iteration. In other words, termination of the program occurs when the lower 3^{rd} geometric quality no longer continually improves or begins to degrade more than 1% during the subsequent meshing iteration. In practice Eq. 2.15 tends to produce higher-geometric quality meshes as the $q_{L3\sigma}$ is higher upon exit; however, this requires between 20 to 30 more meshing iterations than Eq. 2.14 to reach the termination state.

2.4.1.3 Mesh improvement strategies during mesh generation

Approximately every ten meshing iterations the $q_{L3\sigma}$ element quality starts to saturate. The termination criteria can be met more quickly by relying on the following

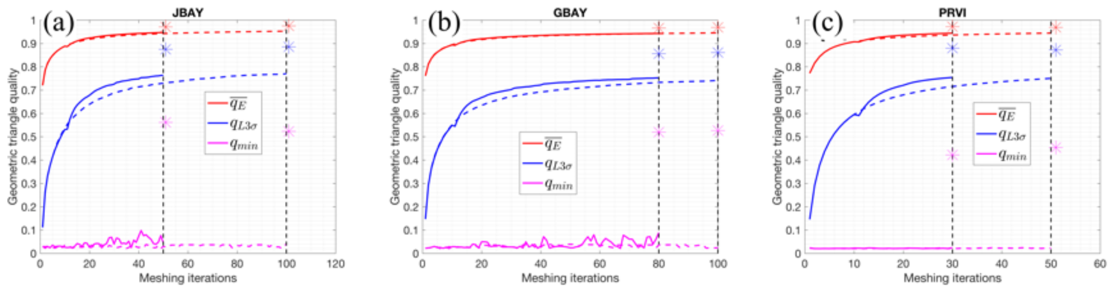


Figure 2.4. The geometric triangle quality q, Eq. (2.13), as a function of iterations in the mesh generation process for the three mesh examples (Fig. 2.1 and Table 3.1). The dotted and solid lines indicate the progression of quality metrics with the mesh improvement strategies turned off and on, respectively, during mesh generation. At the end of mesh generation using Eq. 2.14 to determine termination, a secondary round of mesh improvement strategies are applied and the resulting quality after this step is indicated by the colored asterisk. In each panel, the dotted vertical black line demarcates when the mesh generation process finished.

mesh improvement strategies that are conducted every ten iterations (except item 4 which is executed every meshing iteration):

1. Edges in the mesh that are greater than two times the length as given by the mesh size function (at the midpoint) of the edge are bisected.

2. Edges that are half as short as their intended length are deleted.

3. A vertex not on the mesh boundary that is connected to less than or equal to four vertices is deleted (this is also performed when the termination criterion is satisfied).

4. Triangles with exceedingly thin angles ($< 5°$) and large angles ($> 175°$) are removed every iteration.

Improvement strategies one and two add and delete vertices when they are part of edges that are too long and short, which produces a set of new edges that more closely approximate the mesh size function. Improvement strategy three directly reduces the occurrence of low vertex-to-vertex connectivity (valency of three or four) where a valency of six is ideal [30]. Note that improvement strategy one also helps to reduce high vertex-to-vertex connectivity indirectly by avoiding steep transitions of in the element size where larger valencies greater than six tend to develop. The fourth improvement strategy removes triangles with small and large angles allowing neighboring vertices to form a triangulation that has a better geometric quality.

I demonstrate the benefit from using these mesh improvement strategies through the three example meshes (Fig. 2.1, Table 3.1). The time evolution of the geometric quality demonstrates the benefit directly from these mesh improvement strategies. Figure 2.4 illustrates that in all three examples the mesh improvement strategies lower the number of iterations necessary to achieve the termination criterion. Further, the rate at which $q_{L3\sigma}$ increases is accelerated when mesh improvement strategies are enabled. For the development of large multi-scale meshes, 20-50 iterations can often save between 5-20 minutes for the problems to reach the termination criterion. Based on the termination criterion and the improvements listed here, we generally find that complex coastal ocean meshes are generated in approximately 30-100 iterations. Thus, the maximum allowed number of iterations is commonly set to 100, which typically takes a few minutes to half an hour to compute depending primarily on the geometric complexity of the boundary and the ratio of domain size to minimum element size.

2.4.1.4 Mesh improvement strategies after mesh generation

After mesh generation has terminated, a secondary round of mesh improvement strategies is applied that is focused towards improving the geometrically worst quality triangles that often occur near the boundary of the mesh and can make simulation impossible (e.g., Fig. 2.5(a)). Low quality triangles can occur near the mesh boundary because the geospatial datasets used may contain features that have horizontal lengthscales smaller than the minimum mesh resolution. To handle this issue, a set of algorithms are applied that iteratively address the vertex connectivity problems. The application of the following mesh improvements strategies results in a simplified mesh boundary that conforms to the user-requested minimum element size.

Topological defects in the mesh the can be removed by ensuring that it is valid, defined as having the following properties:

1. The vertices of each triangle are arranged in the counter-clockwise order.

2. Conformity: a triangle is not allowed to have a vertex of another triangle in its interior.

3. Traversability: the number of boundary segments are equal to the number of boundary vertices, which guarantees a unique path along the mesh boundary.

Properties one and two are handled with the *fixmesh.m* function that was provided with the original *DistMesh2D* package. Property three (traversability) is often not satisfied upon termination of the mesh generator because a simplification of the shoreline was not applied. Fragmented patches of triangles may appear near the shoreline boundary destroying traversability (Fig. 2.5).

A function, called *Make_Mesh_Boundaries_traversable*, containing two sub-functions that are recursively called was developed to iteratively remove patches of elements that are either disconnected from the major portion of the mesh or are not disconnected but prevent traversability (Algorithm 2.6). The former set of offending elements are defined as being "exterior" disjoint components of the mesh where, starting from a random seed element in the mesh, the total area of a connected set of elements (i.e., elements that share an edge) is smaller than a user-defined threshold μ_{co}, which is defined in terms of either the total mesh area-fraction or an absolute area cutoff (by default we set $\mu_{co} = 0.25$ which is equivalent to a 25% total mesh area-fraction

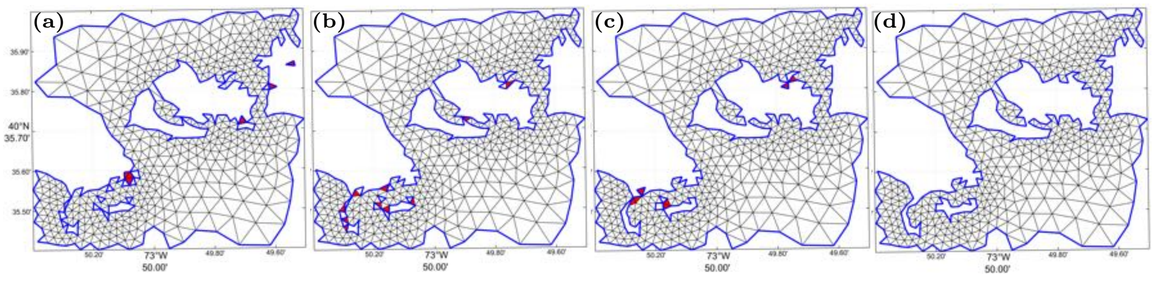
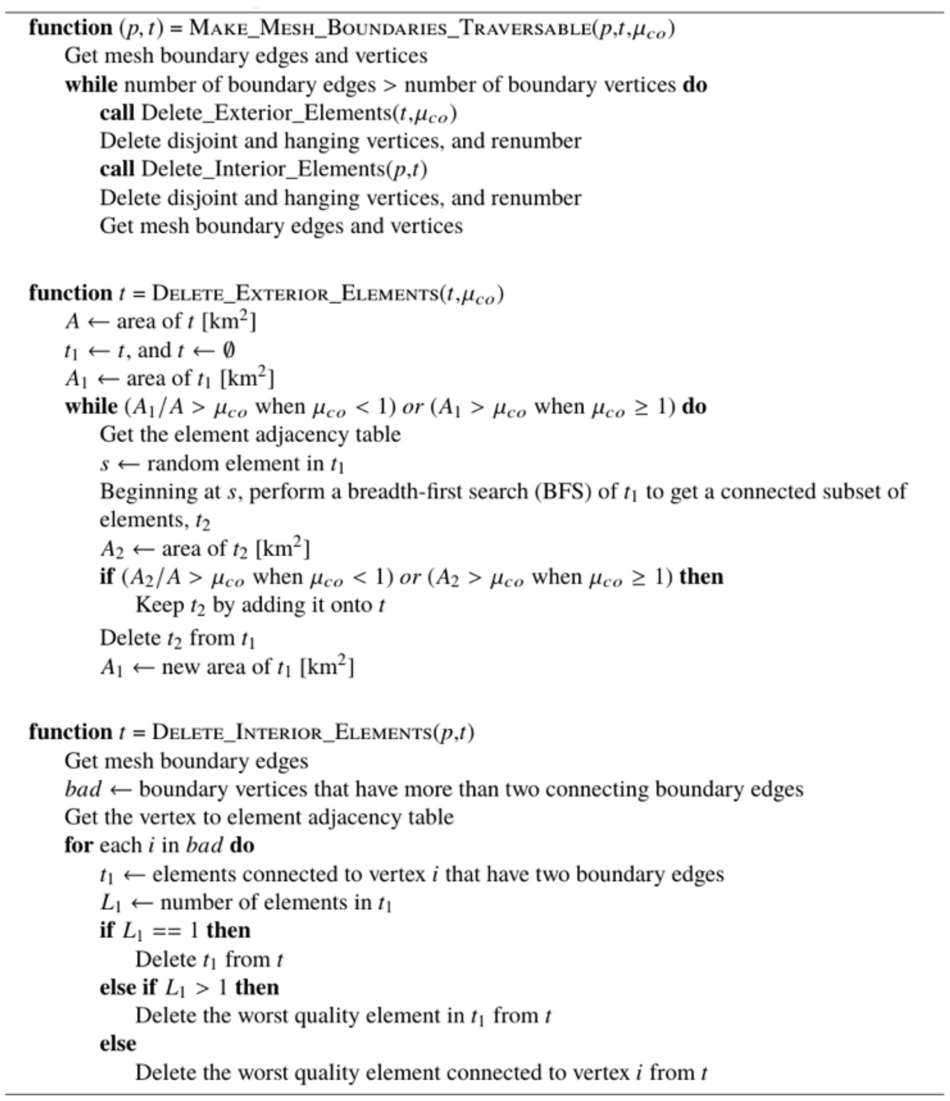


Figure 2.5. Mesh triangulation within the JBAY example before and after different stages of the *Make_Mesh_Boundaries_Traversable* function enabling mesh traversability. (a) After initial mesh generation (before entry to function); (b) After deleting offending exterior elements; (c) After deleting offending interior elements; (d) After exit of function once all offending exterior and interior elements are deleted and traversability is obtained. The thick blue line indicates the mesh boundary at each stage, and red patches indicate the elements that are deleted between stages (sub-plots).

cutoff). These patches are identified and removed by the *Delete_Exterior_Elements* sub-function through the use of a breadth-first search (BFS) (Fig. 2.5(a)-(b)).

The latter set of offending elements are defined as being "interior" elements of the mesh that interfere with the traversability of the mesh boundary path that are not caught by the *Delete_Exterior_Elements* sub-function. The *Delete_Interior_Elements* sub-function deals with identifying and deleting these elements. First, an offending vertex that has more than two connecting boundary edges is identified. One of the elements connected to this vertex is chosen to be deleted based on a hierarchy of, first, triangles that have two boundary edges, and second, triangles with the lowest quality, q_E (Fig. 2.5(b)-(c)). The application of *Delete_Exterior_Elements* followed by the *Delete_Interior_Elements* is conducted iteratively until traversability is achieved (Fig. 2.5(c)-(d)).s

After ensuring traversability, three additional functions, depicted visually in Fig. 2.7, are applied to the mesh in the following order to improve mesh quality:



```
function $(p, t)$ = MAKE_MESH_BOUNDARIES_TRAVERSABLE($p$,$t$,$\mu_{co}$)
    Get mesh boundary edges and vertices
    while number of boundary edges > number of boundary vertices do
        call Delete_Exterior_Elements($t$,$\mu_{co}$)
        Delete disjoint and hanging vertices, and renumber
        call Delete_Interior_Elements($p$,$t$)
        Delete disjoint and hanging vertices, and renumber
        Get mesh boundary edges and vertices

function $t$ = DELETE_EXTERIOR_ELEMENTS($t$,$\mu_{co}$)
    $A \leftarrow$ area of $t$ [km$^2$]
    $t_1 \leftarrow t$, and $t \leftarrow \emptyset$
    $A_1 \leftarrow$ area of $t_1$ [km$^2$]
    while ($A_1/A > \mu_{co}$ when $\mu_{co} < 1$) or ($A_1 > \mu_{co}$ when $\mu_{co} \geq 1$) do
        Get the element adjacency table
        $s \leftarrow$ random element in $t_1$
        Beginning at $s$, perform a breadth-first search (BFS) of $t_1$ to get a connected subset of
        elements, $t_2$
        $A_2 \leftarrow$ area of $t_2$ [km$^2$]
        if ($A_2/A > \mu_{co}$ when $\mu_{co} < 1$) or ($A_2 > \mu_{co}$ when $\mu_{co} \geq 1$) then
            Keep $t_2$ by adding it onto $t$
        Delete $t_2$ from $t_1$
        $A_1 \leftarrow$ new area of $t_1$ [km$^2$]

function $t$ = DELETE_INTERIOR_ELEMENTS($p$,$t$)
    Get mesh boundary edges
    $bad \leftarrow$ boundary vertices that have more than two connecting boundary edges
    Get the vertex to element adjacency table
    for each $i$ in $bad$ do
        $t_1 \leftarrow$ elements connected to vertex $i$ that have two boundary edges
        $L_1 \leftarrow$ number of elements in $t_1$
        if $L_1 == 1$ then
            Delete $t_1$ from $t$
        else if $L_1 > 1$ then
            Delete the worst quality element in $t_1$ from $t$
        else
            Delete the worst quality element connected to vertex $i$ from $t$
```

Figure 2.6. Given a Delaunay triangulation composed of vertices p and vertex-to-element connectivity matrix t, return a p and t that has only two traversal paths along its boundaries in which disjoint portions of the mesh that each make up less than a specified area-fraction μ_{co} of the pre-cleaned, initial mesh area (or less than a specified absolute area in km^2 when $\mu_{co} \geq 1$), have been removed

1. *Fix_single_connec_edge_elements*: elements that share an edge with only one other element (singly connected elements) poorly approximate geospatial datasets and are thus removed from the mesh iteratively (Fig. 2.7(a),(d)).

2. *bound_con_int*: bounds the vertex-to-vertex connectivity (e.g., Fig. 2.7(b),(e)) in the mesh to a user-defined value in order to improve mesh quality and gradation, and also increase solution accuracy and computational speed [102].

3. *direct_smoother_lur*: provides additional improvement to the mesh quality by moving non-boundary vertices based on a single-step implicit operation [13] (Fig. 2.7(c),(f)). The application of this function significantly enhances the statistical distribution of q_E (Fig. 2.4).

2.4.2 Multiscale meshing capability

The *DistMesh2D* algorithm uses memory inefficiently for the development of multiscale regional and global meshes of the coastal ocean because it requires a uniform point spacing to initialize the algorithm. The memory inefficiency becomes especially problematic when employing high-resolution elements locally to fully incorporate the information contained in high-resolution geospatial datasets while using coarser mesh resolution elsewhere. To reduce the memory overhead when constructing regional coastal meshes using the *DistMesh2D* algorithm, the *meshgen* class has been specifically developed to allow the user to pass multiple instances of the boundary description (*geodata*) and mesh size (*edgefx*) classes to the *meshgen* class, an approach that we term 'multiscale meshing'. Instances of these classes are defined within polygonal extents that reflect the available geospatial dataset coverage and can be partially or fully nested any number of times with largely disparate mesh sizes between nests. Examples of the multiscale meshing technique are shown in Figs. 2.1 (PRVI) and 2.8 in which the mesh sizes seamlessly transition between the different DEM extents.

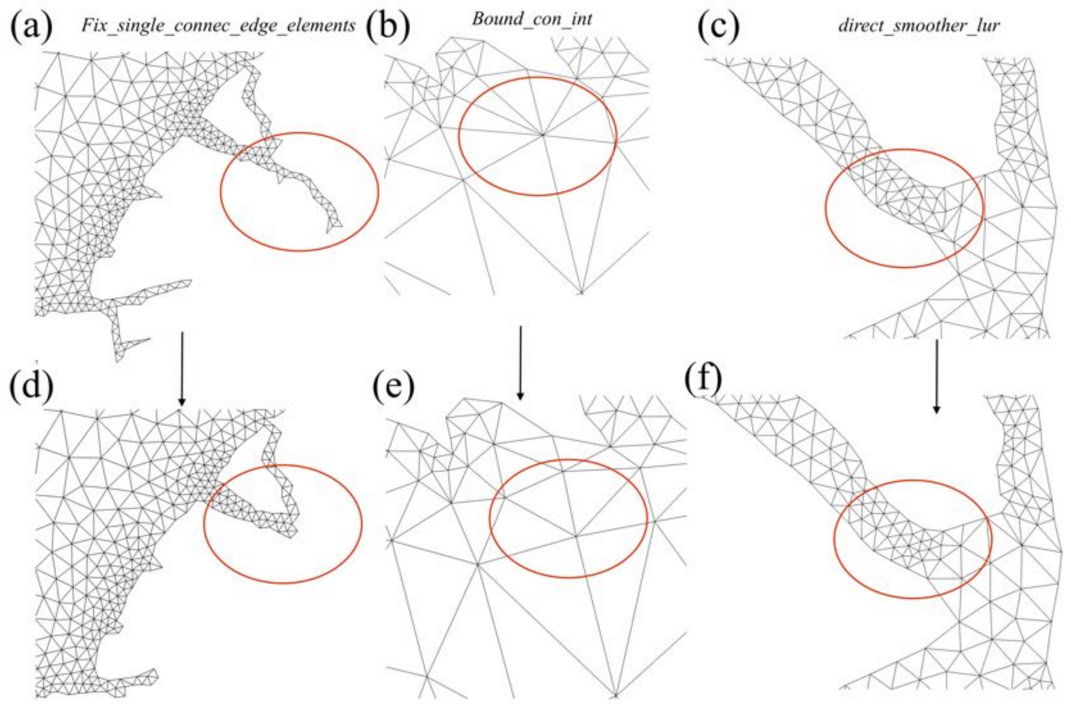


Figure 2.7. The mesh improvement strategies that are applied in sequence from left to right after mesh generation, with the red ovals denoting areas of change in the connectivity along with the function's name that performs the operation. The top row indicates various regions in the mesh before the improvement strategy, and the bottom row after improvement. Panels (a) and (d) indicate the deletion of elements that share an edge with only one other element (singly-connected elements); panels (b) and (e) illustrate the reduction of the vertex-to-vertex connectivity to an upper bound of six using the algorithms documented in Massey [102]; and panels (c) and (f) illustrate the single-step implicit smoothing operation [13] that is used to maximize the overall mesh quality.

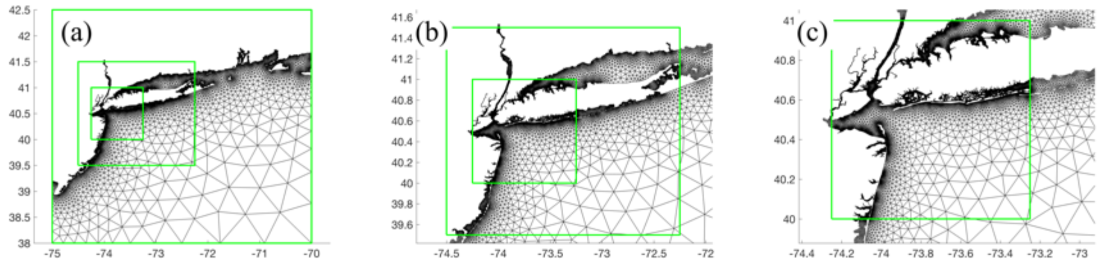


Figure 2.8. An example of the multiscale meshing technique applied to a set of domains around the New York/Long Island area. The green boxes are specified by the user. The minimum resolution of the outermost green box in each panel is different: (a) it is 1 km, (b) 500 m, and (c) 35 m. Notice how the regions of overlap gradually transition into each other.

The mesh size function of an *edgefx* instance is updated in areas of overlap using the mesh size function of comparatively higher resolution. The mesh size function of the coarser *edgefx* instance that was updated is subsequently smoothed using the *limgradStruct.m* function.

Only minor modifications to the *DistMesh2D* algorithm were involved with enabling the multiscale meshing capability. The nested domains are evaluated in a loop inside *DistMesh2D* in a hierarchical order from comparatively coarser to finer resolution minimum mesh sizes. The hierarchical evaluation of the force function (Eq. 2.8) enables vertices of the mesh to move between the nested boxes so long as the outer box fully encloses the inner box. Since the finest local meshing boundary and mesh size function take precedent within each nested box, it enables many variable resolution geospatial datasets to be included into the mesh generation process simultaneously. In order for the multiscale meshing capability to work, it requires smooth mesh size transitions between nests. A routine (*smooth_outer.m*) was developed when multiple *edgefx* classes are present to ensure a smooth resolution transition occurs between nested boxes by using a marching method [4] that has been adapted for structured grids.

The multiscale meshing capability is similar to the multi-grid nesting technique employed by ocean models [e.g., 27, 44, 124] but in the finite element framework without the need for a coupling paradigm. The application of this method allows for the construction of a single seamless unstructured mesh with mesh size transitions that are bounded by the user-defined allowable limit, while the resolution is not significantly altered away from the boundaries of their nests. The multiscale approach is beneficial over traditional structured multi-grid nesting approaches employed by ocean models because it avoids issues associated with interpolation and smoothing at the interfaces between disparate resolution grids that ultimately reduce numerical accuracy.

In practice, this technique has been used to construct high-resolution (≈50-m minmum element size) meshes of the western North Atlantic domain that are simulated with in Chapter 3 using 18 higher resolution domains nested within a coarse outer nest that spans the entirery of the western North Atlantic.

2.4.3 Geospatial data preprocessing: *geodata* class

The *geodata* class is a pre-processor to the mesh generator. It is used to create an appropriate mesh boundary description from user supplied input files. The *geodata* class also stores the region of the digital elevation model (DEM) that overlaps with the desired meshing domain efficiently in memory. This DEM data is used in the construction and computation of a number of mesh size functions (see *edgefx* class) and *msh* methods. The following section describes the methodologies to prepare the mesh boundary description.

2.4.3.1 Automatic mesh boundary definition

Since a coastline is often approximated by a series of piecewise linear segments, the meshing boundary is often unbounded on the ocean-side and may not be a polygon (i.e., first point does not equal the last). Thus, the user often has to turn their segments that represent the coastline into a closed polygon for our meshing algorithm to work properly. To make this process automatic, we enable the user to specify the meshing region as a region bounded by a polygon, *bbox*. Since the mesh domain is then defined as the intersection of the area enclosed by the *bbox* and the area enclosed by the shoreline polygon. The boundary of the meshing domain is implicitly defined through the use of a signed distance function, d, whereby the distance to the nearest coastline point is zero [118]. Note that a negative value of the signed distance function indicates a point within the mesh boundary, and a positive value of the signed distance function indicates a point outside the mesh boundary.

In our methodology, the shoreline polygon is internally partitioned into mainland and island polygons (this categorization is defined below). New vertices are added to the shoreline polygon so that it conforms to the user-requested minimum mesh resolution ($h0$) inside the *bbox*. Vertices are decimated outside *bbox* to save both memory and time during the mesh generation process since the calculation of signed distance function is proportional to the number of shoreline vertices.

1. The segments of shoreline polygon that intersect with *bbox* are read in to memory.

2. The segments of shoreline polygon are classified into three types: mainland, inner, or outer.

   (a) The mainland category contains segments that are not totally enclosed inside the *bbox*.

   (b) The inner (i.e., islands) category contains polygons totally enclosed inside the *bbox*.

   (c) The outer category is the union of the mainland, inner, and *bbox*.

3. New vertices are added on these segments so that no two consecutive vertices along it are further than $\frac{h0}{2}$ apart. This is necessary for accurate re-projection of points that exit the meshing domain during the execution of the *DistMesh2D* algorithm [118].

4. All segments are smoothed using a n-point moving average. Simultaneously, small islands that have an area less than $(p * h0)^2$ are removed where n and p are user-specified integers ($n = 5$, $p = 4$ by default). Likewise small islands that intersect the meshing boundary are removed.

As an example, the following steps are applied to a shoreline extracted from a NCEI Post-Sandy DEM (JBAY in Fig. 2.1) leading to a classification of shoreline points that is crucial for correct automatic meshing of the complicated coastal domain it describes without human intervention (Fig. 2.9).

The capability to use geometric contours extracted directly from geospatial datasets in the mesh generation process without the need for external Geographical Information Systems shoreline simplification algorithms or external shoreline datasets improves workflow efficiency and automation. Further, by using a geometric contour, the resulting shoreline boundary in the mesh is spatially consistent in its location with the topo-bathymetric dataset that is subsequently interpolated onto the mesh vertices. Since many coastal mesh generators rely on the Global Self-consistent Hierarchical High-resolution Shorelines (GSHHS) dataset [150], the automatic geospatial data processing algorithms represent a significant step forward towards more comprehensive coastal modeling efforts. For example, the GSHHS dataset is largely insufficient for meshes with a desired resolution finer than 100-m as it often misses critical connections between water bodies (Fig. 2.10).

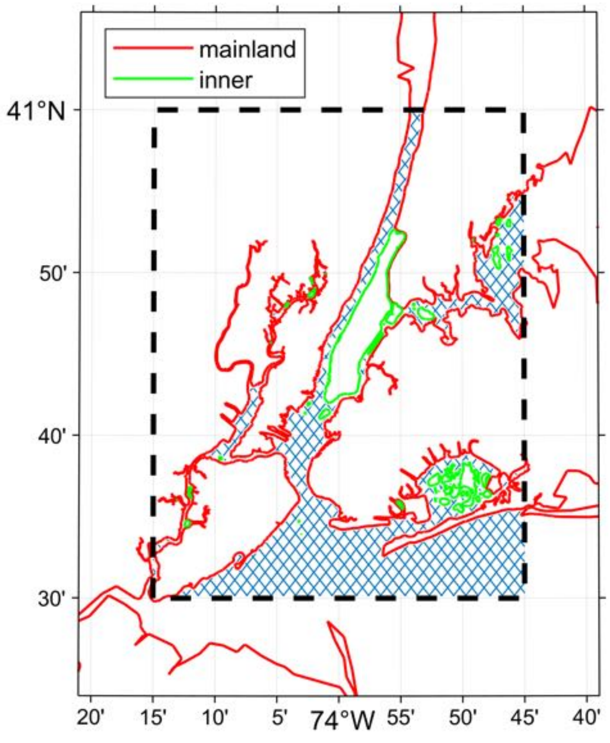


Figure 2.9. Example of boundary treatment in and around New York, United States; the bounding box of the mesh domain, *bbox*, is indicated by the thick dashed black line, the meshing domain is hatched in blue, and the categorization of land boundary types following the text are indicated according to the colored lines.

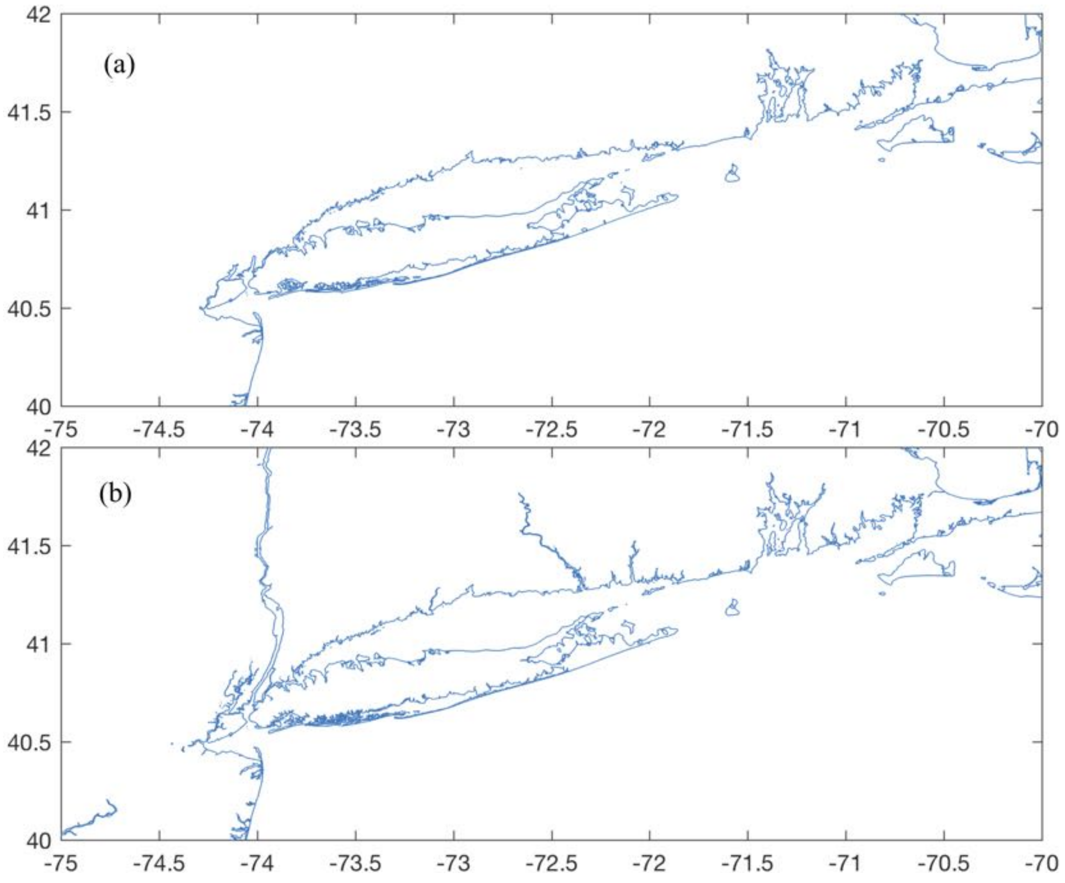


Figure 2.10. (a) The GSHHS fine (i.e., GSHHS_f) shoreline centered around New York, United States; (b) a shoreline extracted from mosaicing NCEI Post-Sandy DEM tiles with the GRASS GIS software.

2.4.3.2 Shoreline pre-processing

The only requirement placed on the polygon S that defines the domain before mesh generation commences is that the vertex spacing on S must be less than the minimum element size $h0$ (Section 2.4.3.1). I have found that ensuring the vertex spacing so that it never exceeds the minimum element size dividied by two performs well. An upper limit to the vertex spacing is necessary for the accurate re-projection of mesh vertices into and out of the meshing domain using the *DistMesh2D* algorithm. To ensure this is the case automatically, a vertex cluster reduction algorithm was programmed into the *geodata* class constructor. In vertex cluster reduction, successive vertices are collapsed into each other to simplify the polygon. Vertex cluster reduction is a brute-force algorithm for polyline simplification.

For this algorithm, a vertex on the polygon is discarded when its distance from a prior initial vertex is less than some minimum tolerance $\delta > 0$. Specifically, after fixing an initial vertex V_0, successive vertexes V_i are tested for closeness and rejected if they are less than δ away from V_0. But, when a vertex is found that is further away than δ, then it is accepted as part of the new simplified polyline, and it also becomes the new initial vertex for further simplification of the polyline. Thus, the resulting edge segments between accepted vertices are larger than the *delta* tolerance.

In the context of the multiscale meshing technique with *DistMesh2D*, the vertex clustering improves the efficiency of the mesh generation by accelerating the calculation of the signed distance function. From Section 2.4.1, the signed distance function relies on a K-d tree to compute the shortest distance from a vertex to the nearest shoreline boundary vertex and has an average computational complexity of $M * \mathcal{O}(log(N))$ where M is the number of vertices contained in the K-d tree and N is the amount of search queries [7, 106]. Thus, the vertex reduction clustering outside of the region of focus becomes highly simplified containing few vertices which altogether reduces the computational expense of the K-d tree calculation by a constant.

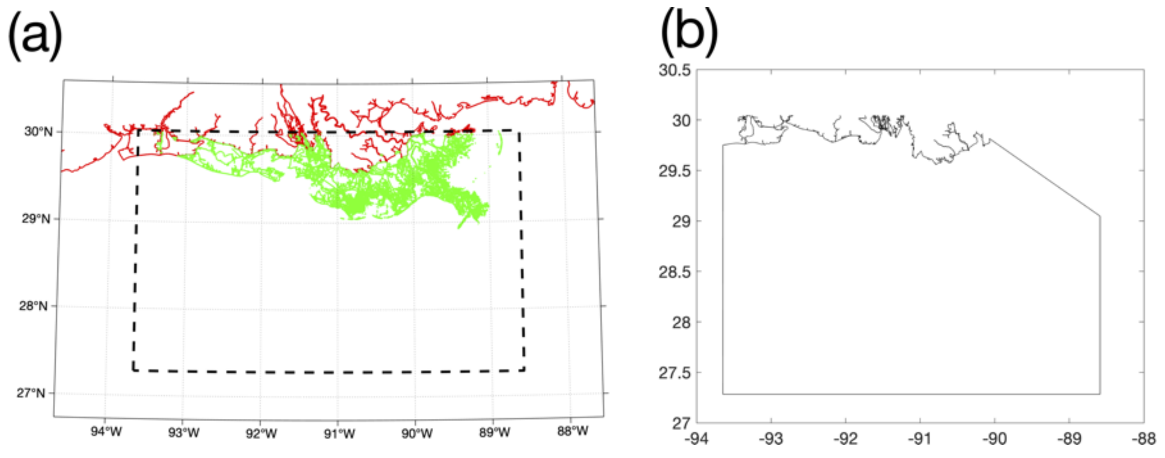


Figure 2.11. (a) A high-resolution shoreline dataset that is not a polygon and a bounding box to be meshed dotted black lines. (b) The result of the flood-fill operation that converts the polyline into a closed polygon. In panel (a), the red boundary is the mainland and the green boundaries represent island features that are closed by definition.

In the case that the shoreline is not a polygon, an ability to turn the shoreline into a polygon is progamed into the *geodata* class as a flood-fill algorithm that I term "shoreline repair". This algorithm is adapted from [57] The user must select a seed location that lies in the intersection of the unclosed shoreline polyline and the bounding box defining the desired meshing domain (Figure 2.11(a)). A constrained Delunay triangulation (CDT) is created of the shoreline and the edges of the bounding box. A CDT differs from a Delaunay triangulation in that a set of edges are fixed in the computational domain and thus do not move and may be locally non-Delaunay. From the seed location, the algorithm traverses the CDT's edges using a breadth-first search and saves all the vistied boundary edges. Once all the visited boundary edges have been stored and there are no more left to visit, the edges are ordered in a walking-order (i.e., counter-clockwise or clockwise) and the a polygon representing the domain is returned to the user (Figure 2.11(b)).

2.4.3.3 Meshing in projected spaces

Mesh resolution sizes need to be accurately controlled according to the mesh size function. In order to accurately enforce these constraints on the curved surface of the Earth, a projection from the spherical geometry of the Earth to a planar one $\mathbb{R}^3 \rightarrow \mathbb{R}^2$ is necessary. In this software, the mesh is generated and output in the World Geographic Coordinate System (WGS84). For the formation of some mesh size functions that rely on bathymetric gradients and distances, we use a simple relationship between WGS84 degrees and planar meters to calculate the underlying grid spacing:

$$\delta^*_{lon} = \delta_{lon}\frac{\pi R_E}{180} cos\phi, \qquad \delta^*_{lat} = \delta_{lat}\frac{\pi R_E}{180} \tag{2.16}$$

where δ_{lon} and δ_{lat} define the DEM resolution in WGS84 degrees between meridians and parallels, respectively, R_E is the mean radius of the Earth ($\approx$ 6378 km), δ^*_{lon} and δ^*_{lat} are the distances between meridians and parallels in meters, and ϕ is the latitude in radians. To enforce mesh resolution constraints, we use the Haversine formula to convert between WGS84 and meters. An assumption is made that the length in geographic degrees forms a horizontal (i.e., latitude parallel) edge starting at the point it is defined at. The distance between the start and end point of this edge are converted to Great Circle distances using the Haversine method and then, later, we invert the Haversine formula and solve for WGS84 degrees by assuming that the distance between latitudes is zero:

$$h_d = 2\arcsin\left(\sec\phi\sin\left(\frac{h^*}{2R_E}\right)\right) \tag{2.17}$$

where h^* is the length of the edge in meters, and h_d is the length of the edge in WGS84 degrees. The assumption that the edgelength extrudes along a latitude parallel is reasonable in practice because the mesh size function constraints matter mostly in

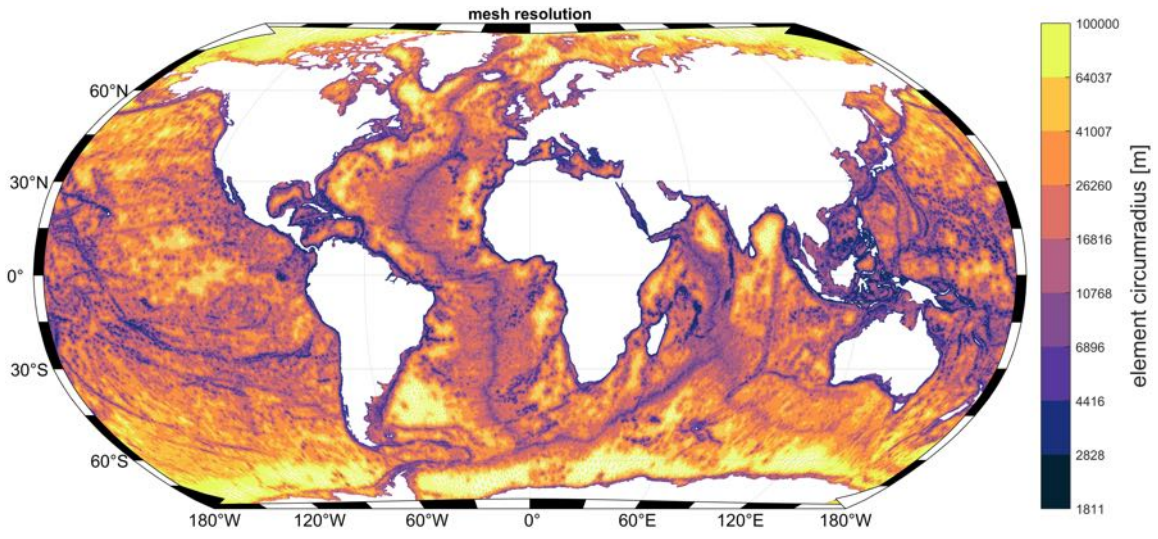


Figure 2.12. Mesh resolution sizes in a global mesh that was created using the *OceanMesh2D* software library in a handful of hours automatically.

areas of relatively high mesh refinement and, in these locations, the variation in ϕ is small.

In V2.0 of the OceanMesh library, the *m_map* library [116] was integrated directly into the software. The *m_map* libary enables meshing in an arbitrary projected space ensuring the user-defined mesh size constraints are accurately enforced on the spherical geoemtry of the Earth. The capability to mesh in projected spaces is enabled by calculating the signed distance function (Eq. 2.10) while projecting the meshing boundaries, and mesh size functions into the desired projected space using *m_map*. The ability build meshes in a projected space (e.g.., stereographic) supports global meshing where the boundaries are wrapped/periodic in space by construction (Figure 2.12).

For example, in Figure 2.12 the mesh boundary when centered on the South Pole becomes Anarctica and thus the $-180°$ meridian is not a discontinuity in the stereographic projected space. When the mesh is projected back into a geographic coordinate system, the elements appear highly anistropic near the Poles but remain isotropic in the projected space. Depending on the application, 18 varous projected spaces can be utilized, including Universal Tranverse Mercator zones for a precise definition of length in small domains.

2.4.4 Automatic mesh size function: *edgefx* class

The careful placement of mesh resolution is critical to create meshes that lead to accurate but efficient simulations. There are a number of heuristics used to design unstructured meshes for shallow water flow applications. A review of some common resolution heuristics utilized in coastal ocean modeling can be found in Greenberg et al. [69]. We have considered a variety of constraints involved in the formation of the mesh size functions by integrating and adapting past work on the topic. The various mesh size functions are detailed in this section.

Mesh resolution is distributed in the domain according to a mesh size function. The mesh size functions are constructed on a *structured* grid that relates every point in the meshing domain to a desired mesh size h, or more precisely, a triangular edgelength (hence the name *edgefx*). There are many techniques to form mesh size functions that vary principally around the methodology to form the background grid on which the mesh sizes are calculated [4]. For example, the simplest approach is to use a Cartesian or structured grid. Defining the mesh size function on a structured grid has advantages over an unstructured one [41, 57] in relation to computational efficiency when storing, querying, interpolating, and performing calculations. Further, bathymetric data is often defined on structured grids (DEMs), and in these cases, computing the mesh size function directly on the same grid can minimize seabed interpolation error for the mesh size function calculation. Given these reasons, I calculate our mesh size functions on Cartesian grids defined in geographical coordinates (i.e., WGS84). A major drawback to this approach is that the entire domain must be uniformly refined which becomes particularly severe for relatively large meshing domains. This impacts the scalability of the subsequent mesh generation process for regional and global coastal mesh generation, but the multiscale mesh capability (Sect. 2.4.2) alleviates this problem by localizing the zones of high resolution.

Each individual mesh size function is based on either shoreline data and/or the bathymetric datasets that were passed to the *edgefx* class constructor. Currently, the software supports a variety of mesh size functions that are used in the ocean modeling community: wavelength-to-grid-size [98, 152], topographic length scale [69, 89], Euclidean distance from the shoreline [118], approximate feature size of the shoreline [4, 87], thalweg/polyline [75], and Courant-Friedrichs-Lewey (CFL)-limiting [17]. Each mesh size function can either be incorporated or omitted based on the user's requirements. The mesh size function is graded using a marching algorithm [4] to ensure that the triangle-to-triangle change in edgelength is bounded below a user-defined percent, α_g.

In Chapter 3, I illustrate the effect these mesh size functions have on simulated results in practice, documenting the considerations of the mesh resolution selection and design on the solution of tides in a relistic and modern problem configuration.

2.4.4.1 Distance and feature size

A high degree of refinement is often necessary near the shoreline boundary to capture its geometric complexity. If mesh resolution is poorly distributed, critical conveyances may be missed leading to larger scale errors in the nearshore circulation [69]. Thus, a mesh size function that is equal to a user-defined minimum mesh size $h0$ along the shoreline boundary, growing as a linear function of the signed distance d from it may be appropriate:

$$h_{dis} = h0 - \alpha_d d \tag{2.18}$$

where α_d is the percent change of mesh size with distance from the shoreline boundary. Eq. (2.18) is what we call the *distance* mesh size function and was originally presented in the *DistMesh2D* algorithm [118].

One major drawback of the *distance* mesh size function is that the minimum mesh size will be placed even along straight stretches of shoreline. If the distance mesh size function generates too many vertices, a *feature* mesh size function that places resolution according to the geometric width of the shoreline should instead be employed [41, 87]. In this function, the feature size (e.g., the width of channels/tributaries, and the radius of curvature of the shoreline) along the coast is estimated by computing distances to the medial axis of the shoreline geometry. Here we have implemented an approximate medial axis method closely following Koko [87]. This involves finding local extrema in the gradient of the d, which in practice, amounts to defining a medial point as a location where $||\nabla d|| < 0.9$ and $d < 0$ [87]. Sometimes due to the configuration of the mesh size function grid juxtaposed on the shoreline geometry, medial points inside small channels may be lost. These medial points can be recovered by classifying mesh size function grid points as medial points if both adjacent neighbors (in the north-south or east-west directions) are outside of the domain (i.e., signed distance is positive) but the mesh size function point under question is within the domain (i.e., signed distance is negative). Once the medial points are computed, the local feature size h_{lfs} is calculated as:

$$h_{lfs} = \frac{2(d_{MA} - d)}{\alpha_R} \tag{2.19}$$

where the numerator is an estimate of the width W of the shoreline, α_R is the user specified number of desired elements per local feature size (commonly $2 \leq \alpha_R \leq 6$), and d_{MA} is the absolute distance to the nearest medial point. Since the medial axis is an approximation, the identification of the full set of medial points depends on the horizontal resolution of the mesh size function. This implies that the *feature* mesh size calculation will work best when computed on a structured grid of resolution similar or finer than the horizontal resolution of the supplied geophysical datasets.

To demonstrate the efficacy of the *feature* mesh size function, we use a 1/9 arc second (∼3-m) topo-bathy DEM. to generate an approximate 10-m minimum element size mesh of Jamaica Bay in New York, United States (JBAY; Fig. 2.1), with $\alpha_R = 3$. Relatively coarse resolution is placed along linear regions of the sand bar, while the dark patches indicate where higher resolution is automatically placed around points of high curvature in the coastline and through channels. For example, two closeups are shown where higher resolution is placed along a narrow constriction and around perpendicular coastal groins along a beach.

A modification to Equation 2.19 can also be utilized by making the denominator α_R a function of the shoreline width, i.e., $R(W)$. In practice, I've found that the following function is suitable for meshing complicated shoreline geometries accurately:

$$\alpha_R(W) = min(ceil(\frac{W}{h0}), maxR) \tag{2.20}$$

where *ceil* denotes the ceiling operator that rounds up to the nearest integer, the minimum element size is represented as $h0$, and $maxR$ is a user-defined maximum bound for the number of elements per shoreline geometric width (W). A reasonable value R is 5 to 7, which enables more elements in wider geometries and fewer in narrower geometries.

2.4.4.2 Wavelength-to-grid size

In shallow water theory, the wave celerity, and hence the wavelength λ, is proportional to the square-root of the depth of the water column. This relationship indicates that more mesh resolution at shallower depths is required to resolve waves that are shorter than those in deep water. With this considered, a mesh size function

h_{wl} that ensures a certain number of elements are present per wavelength (usually of the M_2 dominant semi-diurnal tidal species) can be deduced:

$$h_{wl} = \frac{\lambda_{M2}}{\alpha_{wl}} \tag{2.21}$$

$$h_{wl} = \frac{T_{M2}}{\alpha_{wl}}\sqrt{gb} \tag{2.22}$$

where λ_{M2} and T_{M2} are the wavelength and period (≈12.42 hours) of the M_2 tidal wave, g is the acceleration due to Earth's gravity, b is the bathymetric depth, and α_{wl} is the user specified number of elements chosen to resolve the wavelength. If the M_2 wavelength is sufficiently captured, the diurnal species will also be sufficiently resolved since their wavelengths are approximately twice as large as the M_2. In general, the wavelength parameter α_{wl} is set to a value somewhere between 25 and 100 [90, 152].

2.4.4.3 Topographic length scale

The distance, feature size, and/or wavelength mesh size functions can lead to coarse mesh resolution in deeper waters that under resolve and smooth over the sharp topographic gradients that characterize the continental shelf break. These slope features can be important for coastal ocean models in order to capture dissipative effects driven by the internal tides, transmissional reflection at the shelf break that control the astronomical tides, and trapped shelf waves [79]. The scaling of the slope parameter, commonly called the topographic length scale, is usually represented by the following:

$$h_{slp} = \frac{2\pi}{\alpha_{slp}}\frac{b}{|\nabla b|} \tag{2.23}$$

where $2\pi/\alpha_{slp}$ is the number of elements that resolve the topographic slope, and ∇b is the gradient of the bathymetry evaluated on a structured grid of horizontal

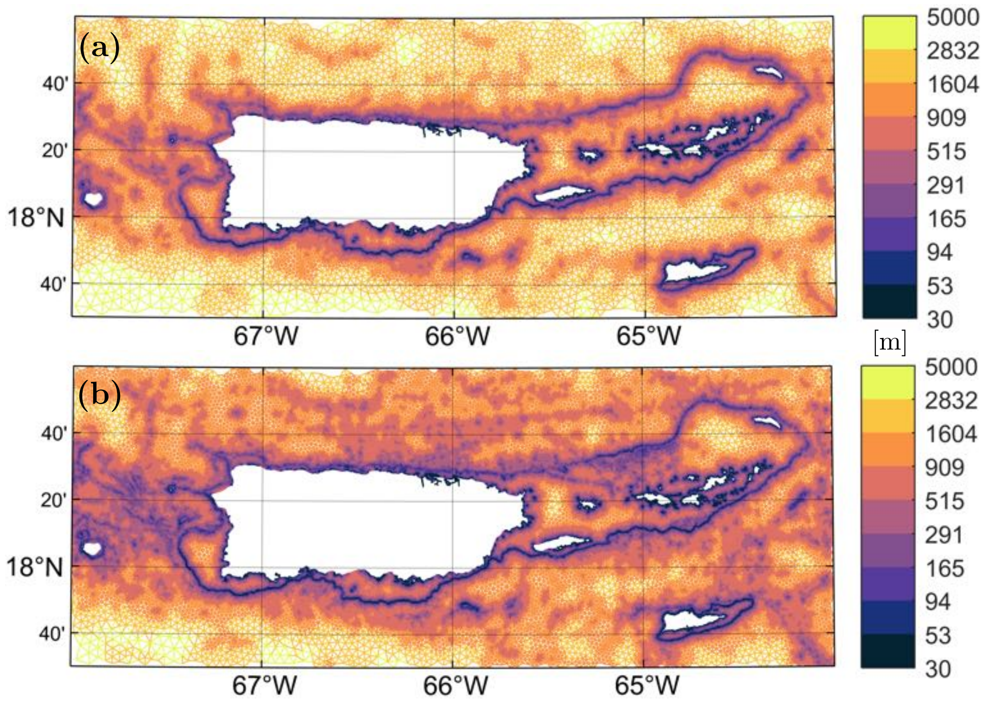


Figure 2.13. Mesh resolution (defined as the local element circumradius [m]) in the PRVI example (see Table 3.1) around the Puerto Rico and U.S. Virgin Island inset region, with (a) and without (b) the Rossby radius slope filter applied. The 'thermal' color palette from cmocean [147] is used in this figure.

resolution $h0$. The 2π factor is a convention introduced by Lyard et al. [99] so that α_{slp} can be set to a value similar in magnitude to α_{wl}, e.g., around 10-30.

Typically the gradient of the bathymetry often contains a high degree of noise, which results in high mesh refinement with the application of h_{slp} despite the fact that small features have marginal effects on shallow water flow, particularly in deep water [91]. We would like to filter bathymetric features that are not relevant to the underlying shallow water processes, like the astronomical tides. Therefore, I proposed to low-pass filter the bathymetry before calculating the gradient.The filter cutoff length is based on an estimate of the *local* Rossby radius of deformation:

$$L_R = \frac{\sqrt{gb}}{f} \tag{2.24}$$

where f is the Coriolis force. By *local* we mean that we discretely bin values of L_R in the meshing domain and apply a low-pass filter to those binned grid points with a cutoff set to L_R at the bin midpoint. For this approach to work correctly, partitioning the meshing domain is critical because meshing domain often spans large regions of latitude with highly varying f. Here, the PRVI example (Fig. 2.1 and Table 3.1) is used to demonstrate the effect of the Rossby radius slope filter (Fig. 2.13). The mesh with the Rossby radius slope filter focuses mesh resolution at large and relatively shallow features such as the continental shelf break avoiding the placement of fine resolution over deep and small scale features that are not comparable to L_R. As a result, the mesh with the filtered seabed has 27% fewer vertices in the illustrated region. It is difficult to estimate the cost-savings attributed to the low-pass filter proposed in this section as it depends largely on the geometry of the domain and bathymetric datasets used.

2.4.4.4 Channel thalwegs/polylines

Closer to the shoreline, the width of the nearshore geometry through which water must flow eventually becomes the dominant length scale instead of L_R. Thus, constraints imposed by continuity normally become more important than dynamic balances in determining spatial scales in estuaries [91]. Following this logic, the representation of the cross-sectional area of the channel that connects the ocean to the estuary is important in order to simulate an accurate exchange of water.

The predominant conveyance through a watercourse is often approximated by a series of neighboring points that connect local minimums in bathymetric depth. These locations are referred to collectively as a thalweg and are represented as polylines in the GIS framework. The level of mesh refinement near and around the thalweg can affect the bathymetric representation in the mesh through aliasing local minimums in bathymetric depth. Often the associated length scale of these features is too small to efficiently resolve through the other mesh size functions. Instead we propose a mesh size function to locally enhance mesh refinement around thalwegs.

Thalwegs can be located by thresholding upslope area [112] in a DEM with GIS software such as GRASS. One difficulty with thresholding upslope area to identify submerged channels is that it may produce spurious non-physical channel networks, especially in areas of flat bathymetry.

This mesh size function treats the thalwegs as a set of connected vertices that form polylines and operates on the polylines that intersect with the meshing domain. The mesh resolution is distributed as follows:

1. A circular region in the mesh size function is formed on each thalweg point with a diameter, dia, equal to:
$$dia = 2b\tan(\theta) \tag{2.25}$$
where θ is the angle of reslope.

2. In each circular region, the mesh size function is assigned resolution by
$$h_{ch} = \frac{b}{\alpha_{ch}} \tag{2.26}$$

This assumes the thalweg has a cross-sectional area that resembles a v-shape with a bank angle of θ (which is set to 60° by default) and that the stencil becomes larger as b increases (Fig. 2.14).

As the water column deepens, the horizontal length scale greatly enlarges, which implies that the dynamical effects from small-scale features like thalwegs should weaken. This dynamic is qualitatively captured through Eq. (2.25) by the enlarge-

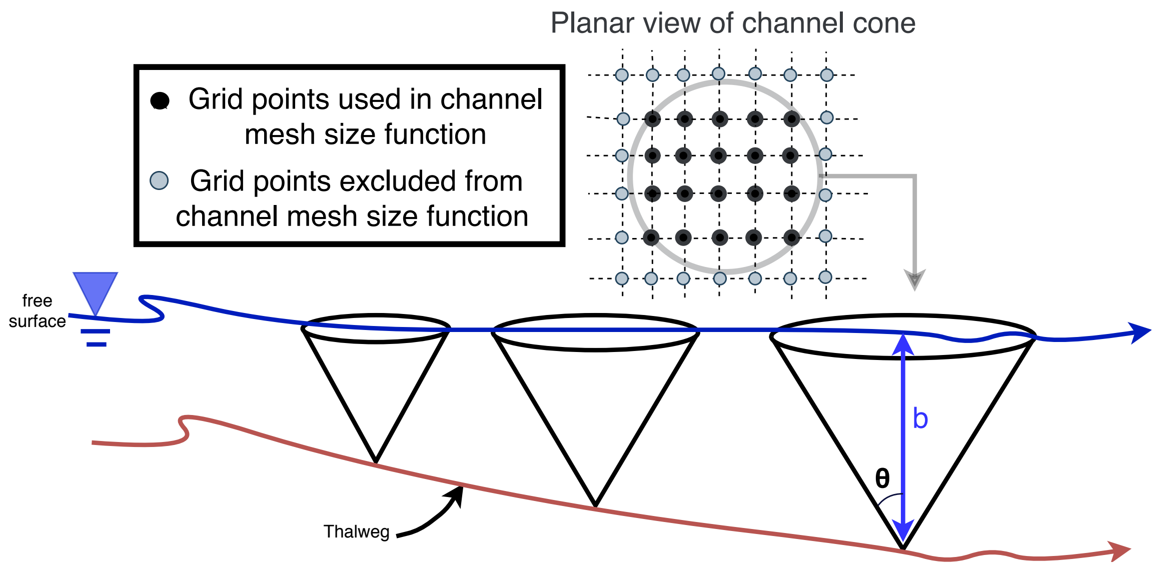

Planar view of channel cone
● Grid points used in channel mesh size function
○ Grid points excluded from channel mesh size function
free surface
b
θ
Thalweg

Figure 2.14. A schematic illustrating the channel mesh size function implementation. The thalweg (deepest part of a channel) is depicted by the maroon line. Mesh size function grid points (defined at the free surface vertical contour) that fall within the channel cones (centered along the thalweg with an assumed bank angle of θ from the vertical) used to estimate the width of the channel are set to follow equation 2.26.

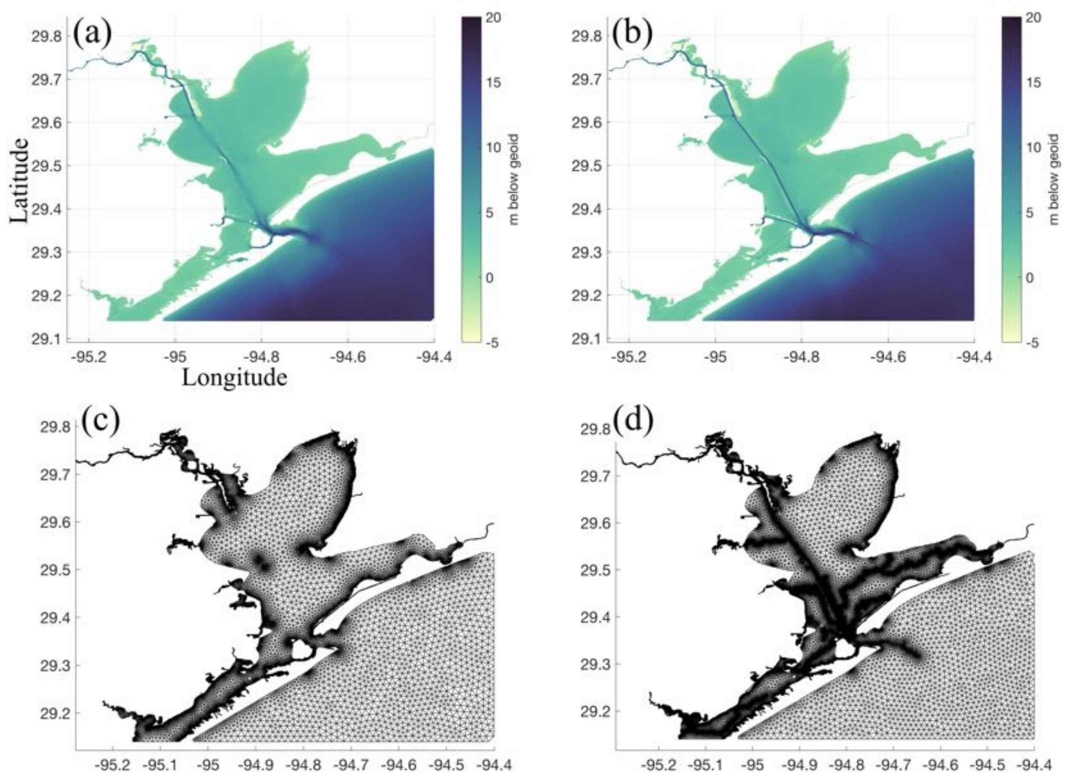


Figure 2.15. Panels (a) and (c) show the bathymetry and mesh connectivity in the GBAY example (Fig. 2.1 and Table 3.1) created without the thalweg mesh size function enabled; panels (b) and (d) are with the thalweg mesh size function enabled. The 'deep' color palette from cmocean [147] is used in palettes (a) and (b).

ment of the thalweg region in the mesh size function as the water depth increases. Additionally, the quotient α_{ch} in Eq. (2.26) alters how the resolution scales with bathymetric depth to further reflect the fact that the horizontal length scale tends to grow as the water becomes substantially deeper, thus reducing the resolution around thalwegs.

As an example of this mesh size function, a mesh is built in and around Galveston Bay Houston (GBAY; Fig. 2.1 and Table 3.1). In this example, we have provided the thalweg points by thresholding the Galveston DEM (Fig. 2.1) with an upslope area of 10,000 cells using GRASS GIS. Visually, the mesh generated using the channel mesh size function clearly captures the bathymetric feature of the Houston Ship Channel to a higher degree of accuracy (Fig. 2.15). Without the use of this mesh size function, the model design would be forced to use an extremely high degree of refinement everywhere to capture the Houston Ship Channel and its adjacent features, or to hand-edit the mesh resolution, which, in both cases, are inefficient methodologies.

2.4.4.5 Finalizing the mesh size function

The final mesh size function, h, is determined by applying the minimum function to the set of individual local mesh size functions, i.e.,

$$h = \min\left[(h_{dis} \text{ or } h_{lfs}), h_{wl}, h_{slp}, h_{ch}\right] \tag{2.27}$$

Note that it is possible to operate on any given subset of mesh size functions. Following this h is further refined based on mesh size transition bounds (Sect.2.4.4.6), Courant-Friedrichs-Lewey limiting (Sect.2.4.4.7), and global user-defined maximum (h_{max}) and minimum ($h0$) mesh size bounds.

2.4.4.6 Mesh size transitions (gradation)

It is necessary to ensure a size smoothness limit α_g such that for any two adjacent vertices $\boldsymbol{x_i}$, $\boldsymbol{x_j}$ connected by an edge, the local increase in mesh size is bounded above such that:

$$h(\boldsymbol{x_j}) \leq h(\boldsymbol{x_i}) + \alpha_g ||\boldsymbol{x_i} - \boldsymbol{x_j}|| \tag{2.28}$$

The mesh size gradation is enforced with the fast marching method that was mentioned in Sect. 2.4.2 and is operated on the mesh size function. A smoothness criteria is essential to produce a mesh that can simulate physical processes with a practical time step as sharp gradients in mesh resolution typically lead to triangles with highly skewed angles that results in low inevitably resulting in poor numerical accuracy [138]. In general, a smoother edgelength function is congruent with a higher overall triangle quality but with more triangles in the mesh. It is important to note that the rate of mesh resolution increase is bounded above by the grade; therefore, if the distance parameter in Eq. (2.18) is set to a value lower than the grade ($\alpha_d < \alpha_g$), then grading should have no effect on the mesh size function. Note that since we are grading the mesh size function and not the mesh, there may be some patches of elements in the triangulation that may contain a gradation larger than the threshold. However, given a sufficient number of meshing iteraations and termination of the mesh generator, the mesh size function will match that of the distribution of vertices and thus approxiately ensure the gradation is bounded above by the size limit α_g.

Here we demonstrate the relative effects of the *distance* and *feature* mesh size functions and their interaction with the grade. To illustrate this mesh size function, we built a mesh over an estuary-like geometry with *distance* ($\alpha_d = 0.15$ and $\alpha_d = 0.35$) and *feature* ($\alpha_R = 3$ and $\alpha_R = 6$) mesh size functions, each using two different gradation bounds ($\alpha_g = 0.15$ and $\alpha_g = 0.35$) (Fig. 2.16). The *distance* mesh size function focuses resolution on the mesh boundary, which is often not necessary to resolve areas that are geometrically simple. Further, the use of a *distance* mesh size function often results in comparatively larger triangles in the center of the geometry; especially with a relatively high grade (i.e., 35%; Fig. 2.16(d)).

In shallow estuaries, this can be undesirable as the bathymetric representation of high conveyance channels in the center of the estuary will be inaccurate, aliasing the transported mass and energy. In contrast, the *feature* mesh size function places a uniform number of triangles across the widest axis of the feature (Fig. 2.16(e)-(f)). It focuses mesh resolution on regions that are narrow and/or where the shoreline has high curvature. The net result is a comparatively smaller number of vertices than the *distance* mesh size function (for $\alpha_R < 20$ in this example). Depending on the selection of α_R in the local feature size equation Eq. (2.19), the size of the

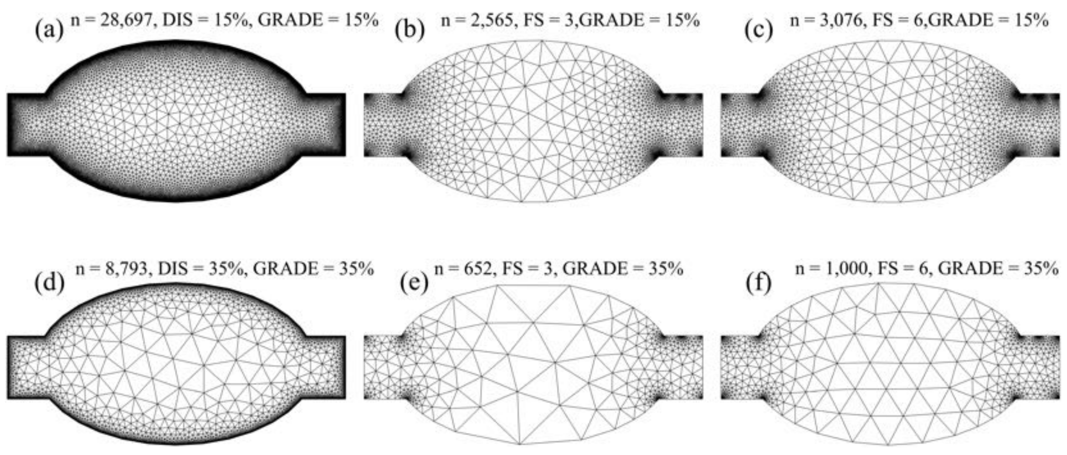

(a) n = 28,697, DIS = 15%, GRADE = 15%
(b) n = 2,565, FS = 3,GRADE = 15%
(c) n = 3,076, FS = 6,GRADE = 15%
(d) n = 8,793, DIS = 35%, GRADE = 35%
(e) n = 652, FS = 3, GRADE = 35%
(f) n = 1,000, FS = 6, GRADE = 35%

Figure 2.16. Depiction of mesh resolution interactions between the grade (α_g), *distance* (DIS), and *feature* (FS) mesh size functions. Panels (a)-(c) depict the resolution with a grade equal to 15% ($\alpha_g = 0.15$), panels (d)-(f) with a grade equal to 35% ($\alpha_g = 0.35$). The first column depicts how mesh resolution is distributed with a *distance* mesh size function and the second and third columns show how the mesh size varies with the *feature* mesh size function with α_R equal to 3 and 6, respectively. In the title of each panel, the number of vertices n in the triangulation is shown.

mesh resolution in the center of the estuary can be bounded even when using a relatively high mesh grade ($\alpha_g > 0.25$). This is advantageous because a higher grade can dramatically lower the overall vertex count. Conversely, a relatively low grade ($\alpha_g < 0.20$) can hinder the *feature* mesh size function from expanding efficiently, and may be somewhat counter-productive to its purpose.

2.4.4.7 Courant-Friedrichs-Lewey (CFL)-limiting

The computational expense of a computational model and associated code is significantly affected by the time step that must be used to ensure stability and accuracy. For models that use explicit time stepping schemes, a necessary condition for numerical stability is determined by the Courant-Friedrichs-Lewey (CFL) condition. Although this is not a sufficient condition, it is a practical way to gauge the success of a new mesh. The CFL condition states that the Courant number (Cr) must be less than or equal to 1. Stricter conditions may be relevant for different numerical schemes and due to nonlinearities in the governing equations [28]. Additionally, the accuracy of a numerical scheme is impacted by the Cr as high values tend to give poorer accuracy even in implicit solvers. For the application of solving the shallow water equations the Cr can be estimated and bounded in the mesh [17]. We define an estimate of Cr at the vertices of the mesh by adding the shallow water wave speed with an estimate of the anticipated flow speed:

$$Cr = \frac{(u + \sqrt{gH})\Delta t}{\Delta X} \tag{2.29}$$

where u is the magnitude of the flow, g is the acceleration due to gravity, H is the total water depth, Δt is the time step, and ΔX is the element size or the shortest connected edge to each vertex. Since the wave speed is a function of depth and ΔX is equivalent to the mesh size, h, the user can estimate the CFL condition *a priori* for a given Δt. Note that to obtain this *a priori* estimate of Cr in Eq. (2.29), we set $H \approx b$, and approximate the flow speed with the long wave linear orbital velocity, i.e., $u \approx \eta\sqrt{g/b}$, where η is the wave amplitude which we set to 1 m by default. Applying these approximations and rearranging gives the following CFL-limiting condition on the mesh size:

$$h \geq \frac{(\eta\sqrt{g/b} + \sqrt{gb})\Delta t}{Cr} \tag{2.30}$$

where b is set to a minimum of 1 m to allow for the CFL condition to be satisfied overland in the case inundation were to occur.

Thus, the user can specify a value of Δt to bound the mesh resolution based on some value of $Cr < 1$. The aim of CFL-limiting is to help facilitate a mesh to be simulated with a certain time step when using explicit time stepping numerical models. However, this often comes with a loss of mesh resolution that may be critical for resolving important conveyances, so the user must consider reasonable values of Δt based on the minimum edgelength. To avoid this choice, we have also implemented an option that automatically selects a suitable Δt that satisfies the condition Eq. (2.30) for the h_{dis} or h_{lfs} (whichever is induced) mesh size functions. The purpose of this is to preserve the nearshore resolution while applying the CFL-limiting to other mesh size functions that may give higher refinement offshore.

To demonstrate the CFL-limiting option, we return to the JBAY example (Fig. 2.1 and Table 3.1), generated using the *feature* mesh size function. In one rendition of the mesh, no CFL-limiting is used (T_{woCFL}), in another rendition, CFL-limiting with $\Delta t = 2$ s (T_{wCFL}) is invoked. In the generation of T_{wCLF}, the mesh size function is conservatively bounded by $Cr = 0.5$ to allow a buffer for the effects of nonlinearities, bathymetric interpolation, and mesh smoothing. After the mesh is generated, the NCEI Post-Sandy DEM is interpolated onto each vertex using a cell-averaging approach (see *interp* method in Sect. 2.4.5, and the resulting CFL is calculated by Eq. (2.29) with $\Delta t = 2$ s. The use of the CFL-limiter acts to shift the distribution of Cr to smaller values (Fig. 2.17). The maximum Cr decreases from 3.64 to 0.96 and the mean Cr shifts from 0.68 to 0.36. In the mesh with the CFL-limiting, there are no vertices that violate the CFL condition as compared to 10,492 in the mesh without it. CFL-limiting thus reduces the number of vertices by locally coarsening h in fact (T_{wCFL} has 45.6% fewer vertices than T_{woCFL}). Again, the user must be careful when selecting Δt as CFL-limiting may lead to choke points in small channels nearshore which are generally the first areas that violate the CFL (Fig. 2.18). Depending on the application this may or may not be tolerable.

Although the above example demonstrates that the Cr of all vertices is reduced to under 1 when using the CFL-limiting mesh size function, the maximum Cr is still 0.96 for $\Delta t = 2$ s, which may be too close to the theoretical condition to simulate

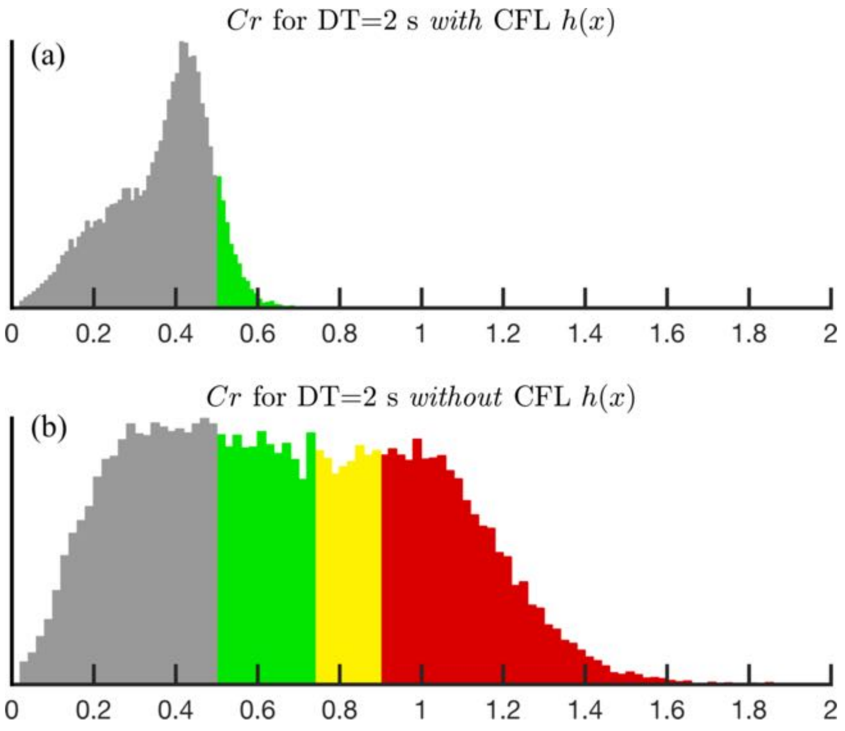


Figure 2.17. Panel (a) illustrates the effect of CFL-limiting on the Courant number Cr when constructing the JBAY example (Fig. 2.1 and Table 3.1) and (b) without it. The colored bars indicate the vertices with $Cr > 0.5$ and are shown to assist in the comparison of histograms.

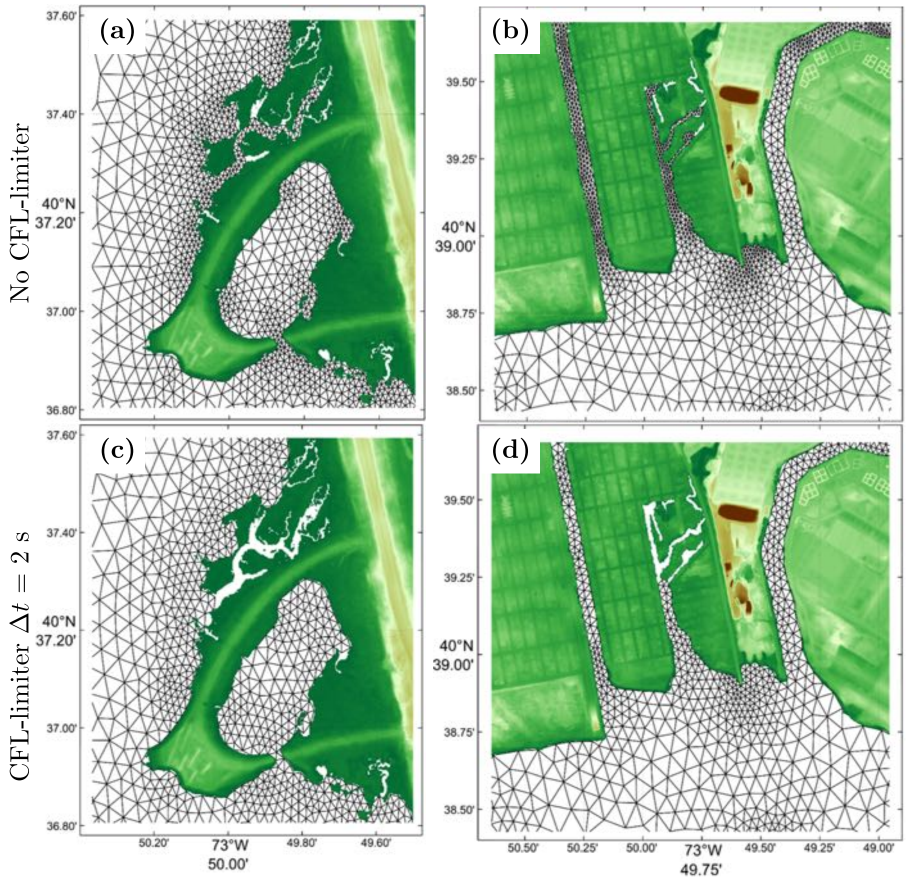


Figure 2.18. Selected closeup regions in Jamaica Bay, New York (left (a) and (c): West Pond, Queens; right (b) and (d): Old Howard Beach, Queens) of the mesh connectivity built with the JBAY example script (Fig. 2.1 and Table 3.1). Top panels (a) and (b) show the mesh connectivity without invoking the CFL-limiter, and the bottom panels (c) and (d) show it when using the CFL-limiting option with $\Delta t = 2$ s.

without instabilities. Based on our experience we need to impose a stricter CFL condition such as $Cr < 0.5$ to ensure numerical stability, accuracy, and to minimize numerical artifacts. To ensure that this more conservative condition is fully satisfied, we propose the *CheckTimestep* post-processing function (Algorithm 1). This function iteratively deletes vertices in the mesh associated with edges that violate the CFL. With each deletion, the vertices on the outer fan containing all the connected elements are smoothed using the Laplacian operator. The algorithm relies on MATLAB's implementation of the Bowyer-Watson incremental Delaunay triangulation to preserve the mesh connectivity outside of the modification patch. For example, in the JBAY example with CFL-limiting, *CheckTimestep* converged after 5 iterations resulting in a mesh with approximately 2,240 less vertices but one that fully satisfied $Cr < 0.5$ everywhere for $\Delta t = 2$ s. In addition to ensuring the CFL condition is fully met, *CheckTimestep* in practice is often used to explore how the mesh would have to be modified in order to achieve a stable simulation for a particular Δt.

```
1: Function $(p,t) = CheckTimestep(p, t, b, \Delta t)$
2: Form nearest neighbor bathymetric interpolant with $p$, $t$, and $b$.
3: Calculate $Cr$ at all vertices using Eq. (2.29) given $b$, $\Delta t$, and the shortest
   connected edge to the vertex.
4: If $Cr \leq 0.5 \; \forall \; p$, then exit.
5: Otherwise, determine point set $p_v$ with $Cr > 0.5$
6: Incrementally delete $p_v$ from $t$ using Bowyer-Watson algorithm.
7: Apply Laplacian operator to the patch around each $p_v$ containing the connected
   triangles.
8: Re-interpolate $b$ onto $p$ using nearest-neighbor interpolant and proceed back to
   3.
```

Algorithm 1: Incrementally adapts a triangulation of points p and connectivity matrix t with bathymetric data b defined at p to a given timestep Δt in seconds through vertex decimation.

2.4.5 Data container: *msh* class

To store the triangulation and related files, the *msh* data storage class contains triangulation-related attributes and support for solver-specific input file enables option-hierarchy, organizes the numerous associated data files in one place, and simplifies the interaction with the underlying data by creating a set of standardized methods. Upon termination of the mesh generator, a *msh* class object containing the triangulation is returned and can be saved efficiently to hard disk as a MATLAB .mat file. While the underlying purpose of the *msh* class is to store the mesh data, the OOP framework enables specific methods to be associated with it. This enables the *msh* class to act as an intermediary between the numerical solver and the user to assist the creation of solver-specific files and perform common data-driven operations on the mesh.

A substantial effort is often required after the triangulation is constructed to enable simulation with a coastal ocean solver such as ADCIRC, FVCOM, SELFE, or SCHISM. For example, the mesh often needs to be visualized and quality checked, boundary conditions must be specified, and seabed topography must be interpolated onto the mesh vertices (Fig. 2.19). Rather than have each user independently write their own methods to accomplish these tasks, we believe it to be more advantageous to place these static or dynamic methods inside the *msh* class that can be edited by everyone using a version control software.

Figure 2.19 illustrates a few of the key methods associated with the *msh* class (see the user guide for a complete list) that I have implemented, such as the visualization of mesh triangulation, resolution, seabed topography, and boundary types. Further, a standardized method for interpolating seabed topography, which employs a generalized cell-averaging approach by default, has been developed. The method can also be used as a wrapper to the built-in MATLAB *griddedInterpolant* function with nearest, linear, and various higher-order interpolation methods. Comparison of

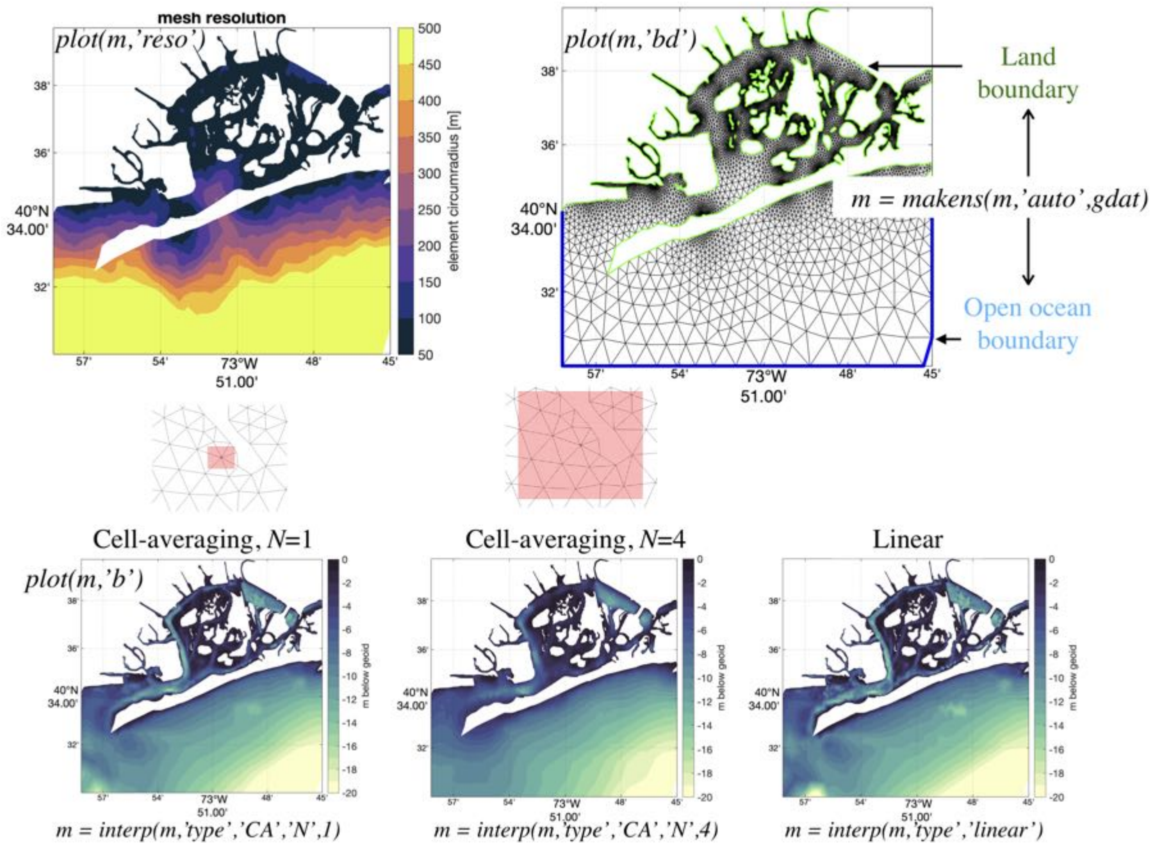


Figure 2.19. Illustration of key *msh* methods: *plot* can be used to visualize mesh resolution (top-left), mesh triangulation and boundary types (top-right), and seabed topography (bottom row); *interp* interpolates seabed topography onto the mesh using cell-averaging or built-in *griddedInterpolant* methods (bottom row); *makens* classifies mesh boundary vertices into land and open ocean types automatically using the native *geodata* class (top-right).

the mesh seabed topography using cell-averaging and linear interpolation methods is shown along the bottom row in Fig. 2.19. Also included is a *msh* method to automatically classify mesh boundary vertices into open ocean, enclosed islands, and mainland types based on the native *geodata* class (top-right in Fig. 2.19).

2.4.5.1 Automatic merging

An approach to merge two triangulations to automatically produce one seamless unstructured mesh was developed and an example is illustrated in Figure 2.20. In this approach, an outer mesh is first developed that encompasses the computational domain of the inset and uses comparatively larger sized elements than the inset. Subsequently, an inset mesh is developed around the region of focus that contains smaller sized elements to more accurately represent the physics of the problem and shoreline and seabed features.

The merging algorithm works by first creating polygonal boundaries of the inset and outer meshes. The polygonal boundaries are used to determine the area of overlap in the outer mesh with the inset by using a polygonal intersection algorithm. The area of overlap is decimated from the outer mesh. After the removal of vertices and elements from the outer mesh, the outer mesh is cleaned with the application of Figure 2.5 to remove disconnected patches of elements that may exist along the boundary. Following this, the inset triangulation is added to the outer triangulation using the incremental Bowyer-Watson algorithm [26] while deleting new elements that form which have low (less than 4) vertex-to-vertex connectivity. After this, a pruning stage occurs which removes triangles with centroids that aren't in the original outer meshes polygonal boundary or aren't in the inset meshes polygonal boundary. Finally, the merged triangulation is cleaned and checked for validility and conformity by applying the same sequence of methods that occur at the end of mesh generation (see Figure 2.7).

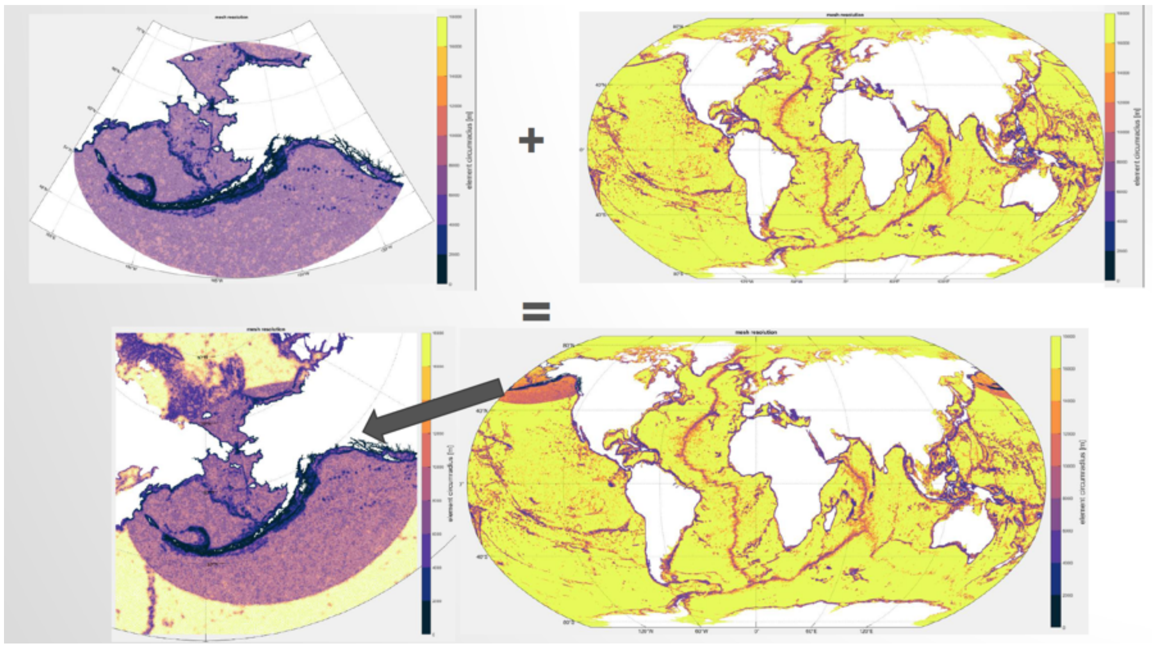


Figure 2.20. An example of automatically merging a regional higher resolution triangulation into a coarser global shell.

The ability to seamlessly merge a higher resolution inset into a outer, coarser mesh that encompasses a larger study region (i.e., the globe) enables new on-demand modeling approaches to be pursued. For instance, a global modeling system can be developed with a minimum mesh resolution of 1 to 4 km and tested for stability and accuracy. As a severe storm develops somewhere in the globe, the region of focus can be narrowed and a smaller high-resolution inset mesh can be quickly developed and automatically merged using the software.

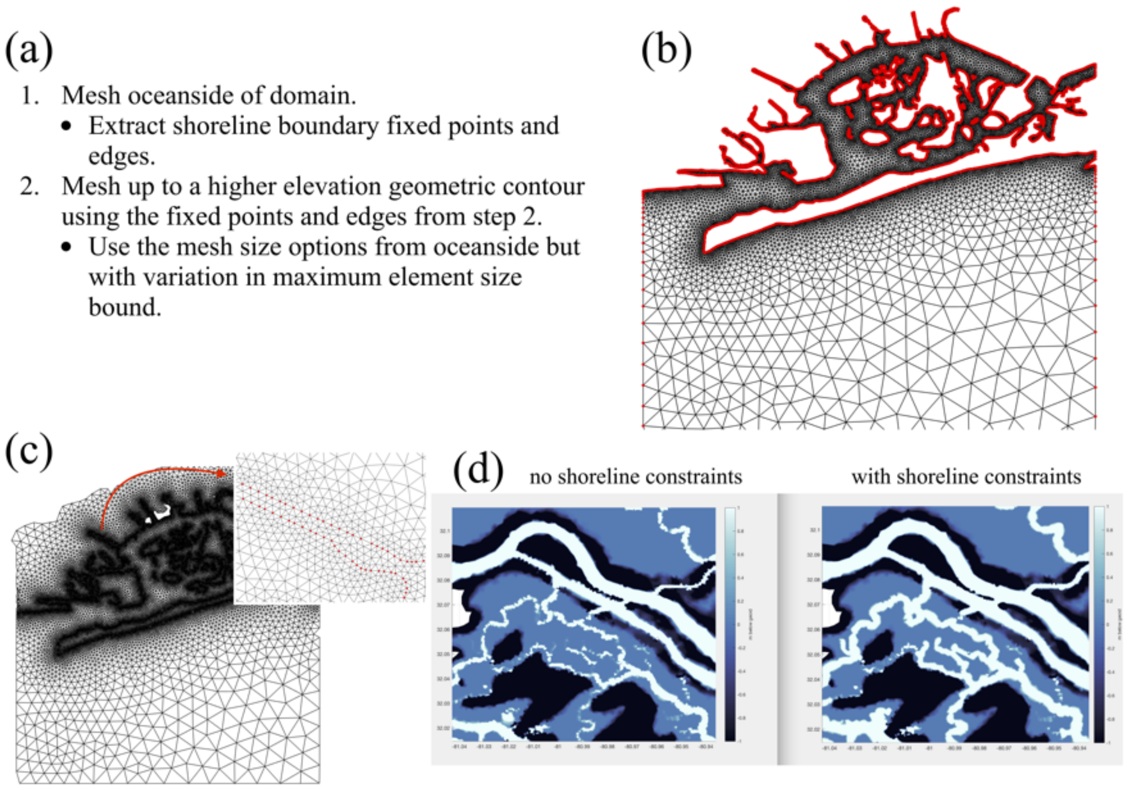
2.4.5.2 Meshing the floodplain

A two-step approach was developed to mesh the floodplain (Figure 2.21). First, a mesh of the oceanside portion of the domain is created. The boundary edges of the oceanside mesh are extracted and stored. A secondary mesh boundary is selected for the same domain but instead extending overland (e.g., 10 or 15 m geometric contour above LMSL). In the subsequent operation, the user passes the oceanside's boundary edges and points to both the *edgefx* and *meshgen* class constructors through a name value pair [see the user guide 129]. The software ensures that either the feature size or distance-based element sizing routines are consistent between the transition zone of the oceanside of the domain onto the floodplain. The points connecting the edges that represent the oceanside's mesh boundary are constrained or "fixed" in the final triangulation that includes both the oceanside and overland component of the domain (Figure 2.21(b)). The edges and points that are fixed ensure two things that are desirable for coastal modeling: i) that the minimum depth on the boundary of the oceanside mesh can be made sufficiently deep to convey flow through complicated and narrow channelized geometries; especially those that are on the order of the minimum mesh sizes, and ii) that logic-based wetting/drying algorithms produce wetting and drying in syncrhonization (i.e, preventing a jagged form of the discrete wet/dry interface to develop during the simulation). Condition ii) is important in channelized sections of the estuaries that have strong along-channel orientated flows because in these cases, if one element becomes dried, the dry element acts as a barrier often leading to both numerical instabilities and incorrect circulations.

This approach to meshing the floodplain also creates a natural division in the model development process enabling the modeler to first assess the numerical stability and the accuracy of the tidal problem before adding on the computational expense of representing the floodplain. At the completion of step 1, the modeler can determine if their mesh size options and shoreline boundary description are capable of meeting their accuracy and performance standards.

2.5 Performance of library

The total and component-based wall-clock times for generating each of the three examples presented in this study is shown in Table 2.2. Overall, the small exam-


(a)

1. Mesh oceanside of domain.
    - Extract shoreline boundary fixed points and edges.
2. Mesh up to a higher elevation geometric contour using the fixed points and edges from step 2.
    - Use the mesh size options from oceanside but with variation in maximum element size bound.

Figure 2.21. Panel (a) documents the steps involved with the two-step approach to automatically mesh the floodplain. Panel (b) shows the oceanside portion of the doman with the boundary edges and points indicated in red. Panel (c) shows the second step in the floodplain meshing algorithm that incorporates the oceanside's boundary as fixed points and edges to mesh to a higher elevation geometric countour. Panel (d) illustrates the interpolated seabed topogrpahy (using a cell-averaged approach) onto the mesh vertices with (left) and without (right) consideration of lowering the depth along the fixed points to ensure they remain wetted that represent the shoreline geometry.

ples (JBAY and GBAY) complete in under 2 minutes, and the large PRVI example takes approximately 45 minutes. Consistently for all examples vertex relocation consumes slightly more time than Delaunay triangulation, and the mesh cleaning (post-processing improvement strategies) accounts for approximately 6% of the total time. The relative balance between mesh generation and pre-processing times depends on the resolution of the shoreline and the size of the meshing domain. For example, in the small domain problems (JBAY and GBAY), the pre-processing time makes up roughly a quarter of the total time. In contrast, in the PRVI example which meshes most of the north western Atlantic ocean using four separate *geodata* and *edgefx* classes, the pre-processing time accounts for 64% of the total time. Therefore, while it is likely possible to speed-up the mesh generation process through e.g., parallel Deluanay triangulation and/or different approaches to the initial point rejection in the *DistMesh2D* algorithm, for large and complex meshes intelligent use of the multiscale meshing approach combined with parallelization of the construction of the individual *edgefx* and *geodata* classes is likely to result in the greatest speedup.

2.6 Discussion and conclusions

A self-contained model development toolkit to automate the generation of two-dimensional (2D) triangular unstructured meshes for coastal ocean models was developed. The overarching goal of the software is to reduce the complexity and hours spent constructing real-world unstructured meshes to the degree that it allows one to more carefully and systemically study the impact on the coastal circulation. This is achieved through a standardized scripted workflow comprising pre- and post- processing steps of geospatial datasets and mesh properties, which are performed by four dedicated classes. Each class was designed to simplify the necessary pre- and post- processing procedures for mesh generation leading to a self-contained model development tool. Whleis focused on producing standardized workflows to automate

TABLE 2.2: WALL-CLOCK TIMES TO BUILD EXAMPLE MESHES.

		Mesh generation time			
Example	**Pre-processing time**	**Vertex relocation**	**Triangulation**	**Cleaning**	**Total Time**
JBAY	13.7 (22.5%)	24.9 (41.0%)	18.3 (30.0%)	3.83 (6.25%)	60.8
GBAY	23.9 (25.1%)	34.1 (35.8%)	31.4 (33.0%)	5.79 (6.09%)	95.2
PRVI	1,770 (64.1%)	498 (18.1%)	316 (11.5%)	174 (6.31%)	2,760

NOTE: Wall-clock time in seconds (% of total) for the steps involved in mesh generation. The pre-processing includes reading and processing the geospatial data (in *geodata*) and forming the mesh size functions (in *edgefx*). The vertex relocation timings include the time elapsed in the initial point rejection, vertex re-projection back into the meshing domain, vertex movement, and mesh improvement strategies during mesh generation (Sect. 2.4.1.3). The cleaning category includes the time spent on mesh improvement strategies after mesh generation (Sect. 2.4.1.4).

and improve the efficiency of coastal mesh generation workflows. While the scripting based approach used to generate meshes promotes automation and approximate reproducibility, the pointers contained within the script do not adequately describe provenance attribution of the geospatial datasets and computing environment used. In the future, employing formal Research Data Management practices in the context of geophysical mesh generation [8, 31] into *OceanMesh2D* would be beneficial to heighten reprodubility.

For coastal mesh generation, a key advantage of using the *DistMesh2D* smoothing-based algorithm over Delaunay refinement and/or Frontal Delaunay mesh generation algorithms is that the boundary is implicitly defined using a signed distance function. The implicit definition of the mesh boundary enables the vertices, including the boundary vertices that represent the highly irregular shoreline boundary, to move during mesh generation step in accordance with the mesh size function. In contrast, Delaunay refinement and advancing front schemes incrementally modify the triangulation starting from a partitioning of the polygonal boundary and then propagate into the interior of the meshing domain. This aspect required by front-based mesh generation schemes requires that the shoreline boundary be simplified in accordance with the mesh size function before mesh generation commences, which can be a challenging step.

A set of common coastal ocean relevant mesh size functions were built into the mesh size function class (*edgefx*) that can handle a variety of user-based constraints and facilitate the approximate reproducibility of mesh vertex locations. The implementation of these mesh size functions were largely borrowed from pre-existing literature with some minor enhancements. We presented a polyline mesh size function to locally enhance resolution around and near marine navigation channels and deep-draft channels (i.e., thalwegs). These features are found by thresholding upslope-area calculated from a digital elevation model (DEM) using GIS software. The polyline mesh size function may have interesting future applications for the development of overland meshes that seamlessly mate with ocean meshes. For example, the user could provide a set of lines that characterize overland ridges so that the polyline mesh size function can be used to locally enhance mesh resolution to better capture the local maximums in the topographic heights. Since the representation of the inter-tidal and floodplain zone in the mesh is critical for coastal flooding applications, ensuring overland features like hills and levees are correctly represented in the mesh is a important feature. In its current state, the toolbox is able to constrain piecewise linear segments that may represent e.g., a series of levees; however, if there is a large degree of disparity between the point spacing on the constraints and the mesh size function, then the resulting mesh will be of poor quality.

To ensure that a mesh is computationally stable with a user-requested time step (relevant when simulating with explicit/semi-implicit numerical models), a CFL-limiting mesh size function similar to Bilgili et al. [17] was introduced. In this approach, we estimate the Courant (Cr) number based on shallow water wave theory and ensure that the final mesh size function satisfies the CFL condition ($Cr < 1$). Although applying CFL-limiting to the mesh size function was shown to help encourage stability by lowering the Cr, the resulting unstructured mesh may not necessarily satisfy the CFL condition due to that fact that bathymetric interpolation from the DEM is not easily constrained. Thus, an iterative algorithm to be applied *after* the mesh was developed (*CheckTimestep*) to locally alter the connectivity by decimating vertices that violate the CFL condition. Depending on the users choice of time step and the various mesh size constraints, the algorithm decimates vertices in certain regions (e.g., small constricted channels) that may or may not be tolerable for the problem at hand. In such regions, anisotropic mesh elements [e.g., 119] that could be generated using mesh size functions which include a directional component may be more beneficial than isotropic equilateral elements. Thus, implementing anisotropic mesh size functions into the software, along with the testing of the resultant meshes in real coastal ocean problems, is an interesting direction for future work.

We emphasized the expensive nature of building large-scale high-fidelity mesh size functions which motivates the use of a multiscale meshing approach. This approach reflects the often sparse spatial coverage and heterogeneous nature of freely available digital elevation data that are often used in the construction of the mesh size functions. Multiscale meshing allows the user to build (extremely) high-resolution local mesh size functions that are embedded in larger scale ocean domains. The end result is a mesh that seamlessly transitions from the high refinement region to coarser elements outside the region of interest. This is practically useful to accurately model coastal flooding in small regions (e.g., a city or a small island – here we show an example of the approach with the mesh refinement region around Puerto Rico and the U.S. Virgin Islands) that may be susceptible to storms and tropical cyclones (TC) passing over it. For large-scale TC-driven storm surge events, it has been shown that a large model domain is essential to capture the pre-event conditions that can alter the modeled severity of the event [18]. In forecasting scenarios, the multiscale meshing approach could be used to mesh around the predicted land-falling region based on the cone of uncertainty of the path of the storm to locally higher resolution. This approach could generate meshes for the prediction of coastal flooding on-the-fly as new forecast data becomes available. Given the local nature of the mesh refinement in this approach, these meshes could be computationally more efficient with smaller minimum element sizes than pre-existing ones, e.g., the U.S. National Ocean Service's (NOS) Hurricane Storm Surge Operational Forecast System (HSSOFS) mesh [146], which cover entire swaths of coastline with medium level resolution.

The objected-oriented structure of the software enables each component to be used in isolation and/or under workflows different to that presented here. For instance, mesh size functions constructed through the *edgefx* class could be used with other mesh generators to distribute vertices. Furthermore, the ability of *OceanMesh2D* to automatically adapt user-supplied shoreline datasets to a mesh size function is a new feature to the authors' knowledge. This ability could be used as a standalone feature to produce polygonal boundaries that approximate the shoreline with a variety of spatial constraints for other mesh generator packages or GIS applications.

Three examples were used for demonstration in this study (Fig. 2.1 and Table 3.1). A further three separate examples are illustrated in the user guide [129]. All six examples are released with this version of the *OceanMesh2D* library. They can be used to become familiar with the software, for testing purposes, and as templates for scripts used to generate the user's custom mesh.

CHAPTER 3

ON THE AUTOMATIC AND *A PRIORI* DESIGN OF COASTAL OCEAN CIRCULATION MODELS

3.1 Overview

This chapter investigates the design of unstructured mesh resolution and its impact on the modeling of barotropic tides along the United States East Coast and Gulf Coast (ECGC). A discrete representation of a computational ocean domain (mesh design) is necessary due to finite computational resources and an incomplete knowledge of the physical system (e.g., seabed topography). The selection of mesh resolution impacts both the numerical truncation error and the approximation of the system's physical domain. To increase confidence in the design of high-resolution coastal ocean meshes and to quantify the efficacy of current mesh design practices, an automated mesh generation approach is applied to objectively control resolution placement based on *a priori* information such as shoreline geometry and seabed topographic features. The simulated harmonic tidal elevations for each mesh design are compared to that of a reference solution, computed on a 11 million vertex mesh of the ECGC region with a minimum shoreline resolution of 50-m. Our key findings indicate that pre-existing mesh designs that use uniform resolution along the shoreline and slowly varying resolution sizes on the continental shelf inefficiently discretize the computational domain. Instead, a targeted approach that places fine resolution in narrow geometric features, along steep, topographic gradients, and along pronounced submerged estuarine channels, while aggressively relaxing resolution elsewhere, leads to an efficient mesh design with an order of magnitude fewer vertices than the reference solution with comparable accuracy (within 3% harmonic elevation amplitudes in 99% of the domain).

3.2 Introduction and background

Two-dimensional (2D) unstructured triangular meshes are widely used to represent the horizontal domain in the simulation of hydrodynamic processes of ocean, shelf and inland coastal water systems. In general, these variable resolution meshes are used to study a broad spectrum of processes in the coastal ocean from wind waves with periods on the order of seconds to large scale shelf and oceanic circulation with timescales on the order of days to months. Most commonly, barotropically-driven long wave processes (tides, surge, and tsunami) with periods on the order of minutes to hours are simulated with these meshes. This includes the modeling of tidal dynamics [23, 36, 122] and the prediction of extreme water levels during high energy events such as tropical and extratropical storms [1, 3, 42, 50, 51, 78, 153, 156, 158, 161]. Critically, unstructured triangular meshes facilitate seamless cross-scale modeling of the complete long wave spectrum [125, 159, 160].

Unstructured meshes are used to capture the detailed hydrodynamic response driven by the governing physical processes and their interactions with the physical system. Historically in fluid mechanics, approaches to mesh design and adaption have often been based on *a posteriori* techniques based on the residual of the flow solution on a per element basis [e.g. 15, 113]. In coastal modeling, an *a posteriori* analysis has been performed using a formal local truncation error analysis (LTEA; [71, 73, 115] with the objective to equalize the truncation error throughout the computational domain. However, a complicating factor when using an *a posteriori* method is that truncation errors that arise from approximating the underlying equations using a numerical technique conflate with approximation errors from representing the system's physical domain. As finer mesh sizes are used to reduce the truncation error, new narrower shoreline details emerge that can alter the system's response. Thus, while the numerical truncation error for a given initial mesh description can be minimized, the system domain error may persist because critical features still do not exist in the boundary description.

The aforementioned considerations motivates us to use a feature-driven *a priori* approach. In fact, for the most part meshes for coastal modeling have been developed using an *a priori* approach adjusting resolution to match both the physical system's length scale and estimated length scales of the dominant physics [e.g., 29, 36, 37, 85, 98, 99]. Feature-driven *a priori* approaches have been proposed to automatically design meshes in this manner [2, 41, 131]. Nevertheless, until now it has been difficult to build a sufficient number of meshes to enable a controlled comparison of the simulated results for realistic coastal ocean hydrodynamic models through the traditional *ad hoc* and tedious [72] development process. However, recent advances in automated unstructured mesh generation technology for the ocean [8, 31, 57, 126, 131] now enable well-defined repeatable workflows for generating detailed multiscale coastal ocean meshes. These approaches alleviate the burden previously associated with the model development steps and ensure that the development process is sufficiently controlled to facilitate inter-comparisons between simulation results from a variety of mesh designs with logical perturbations.

A ubiquitous feature-driven *a priori* meshing criteria for coastal modeling is the wavelength-to-gridscale heuristic that sizes resolution according to an estimate of depth-dependent shallow water wave celerity to maintain constant discretization of the wavelength of the dominant mode [69, 99, 152, 153]. This heuristic produces meshes that contain the finest resolution nearshore, element size transitions that vary smoothly, and nearly constant resolution across the continental shelf. However, the wavelength-to-gridscale heuristic is based on a one-dimensional analysis that assumes no bathymetric gradients and thus cannot capture complexity of seabed features like shelf breaks and isolated banks [69] nor the intricacies of the 2D shoreline. Further, submarine channels that are important to convey flow into the estuarine system can become coarsely discretized with its application. While a long legacy of meshes have been built with this heuristic, the application of resolution using this approach leads to models with many degrees-of-freedom if the parameter dictating the number of nodes per wavelength is set to a large value to compensate for inadequately targeting resolution at the aforementioned features.

Consideration of the topographic-length scale, i.e., applying finer resolution inversely proportional to the seabed depth and directly proportional to seabed topographic gradient has also been widely conducted [37, 57, 99]. This approach refines the resolution in proximity to the shelf break and submarine ridges and banks, which often tend to be co-located with large gradients in the solution [74]. In fact, the LTEA analysis method proposed by Hagen et al. [71, 73] demonstrated that the minimization of truncation error tended to produce a distribution of vertices that resembled the application of the topographic-length scale. Representing steep gradients is also useful to capture submarine ridges and rough topography over which internal tides are generated [63]. This process is often included as a parameterized dissipation process in barotropic tidal models [68, 122, 123]. However, a drawback of the topographic-length scale is that on the inner shelf the topographic gradient to depth ratio can become large due to topographic irregularities which leads to excessively fine resolution as compared to the length scales of the dominant physics.

Unstructured meshes have a powerful capability to efficiently capture the geometrically complex form of the shoreline and of the complex esutaries and the connected dendritic inland channels, but most prior works have not taken full advantage of this capability by applying uniformly fine resolution along shorelines and within inland waterways in regions of interest. For instance, NOMAD (NOAA Operational Model with ADCIRC), a mesh used for real-time predictions of storm surge and tides (e.g., ASGS [59]), uses uniform coastal resolution of approximately 250 m along all the United States East Coast and Gulf Coasts (ECGC) [146]. Other examples of meshes that resolve the shoreline uniformly includes those used in recent long-term regional analyses of storm surge and tides in ECGC [$\sim$5 km; 108] and [$\sim$1 km; 101], and those used for hurricane-induced coastal flooding in the northern Gulf of Mexico [$\sim$100 m; 85]. On one hand, uniform shoreline resolution ensures that the representation of the inlet/backbay system that control coastal inshore hydrodynamics is best represented in the mesh of the specified resolution. On the other hand, the application of nearly uniform resolution nearshore over-resolves many sections of the coastline and inland waters that are straight and geometrically simple leading to a situation where cost constraints then necessitate under-resolving narrow and constricted waterways. Studies in the South Atlantic Bight have demonstrated that the representation of the estuary system as a whole can alter the morphodynamic feedback between the tides and the shoreline form [10, 23]. Thus, beyond applying fine resolution zones nearshore, it is often critical to resolve the intricate dendritic inland waters and to quantify the feedback effects from the integrated system. These irregular shoreline and inland systems are best captured using highly variable mesh resolution.

Another consideration for developing unstructured meshes is the rate of element size transitions between zones of variable resolution otherwise referred to as the gradation [4]. It is known that element size transitions must be smooth and bounded above by a constant to avoid numerical errors and inaccuracies [2, 138]. In fact, the error analysis undertaken by Hagen et al. [71] clearly demonstrates that a gradation above 50% will cause odd order error terms to dominate and subsequently degrade a formally second order numerical method to first order. While a theoretical upper bound value for the gradation is known, the total number of vertices in a coastal ocean discretization can wildly vary depending on the choice of gradation below 50%

(a large gradation will lead to fewer vertices). Thus, the gradation rate needs to be explored to identify a suggested tighter range of values that efficiently discretizes the physical domain while maintaining accuracy in the simulation of the coastal ocean.

A common first step in the production of a coastal hydrodynamic model is to assess the simulated accuracy of astronomical tides [e.g., 122] prior to the simulation of extreme sea levels. At this initial stage of the model development process, the model is calibrated through adjustments to frictional and dissipative parameterizations in order to agree with measured data. However, when the mesh underresolves shoreline and seabed features, the system's response may become distorted leading to an inability to correctly produce solutions across the entire domain and energy spectrum. An example of this would be tuning the model to agree with observations of dominant semi-diurnal elevation tidal constituent regionally but this may not lead to a good agreement globally nor for the other tidal constituents. Instead, by gathering knowledge on how tidal solution depends on mesh resolution in realistic coastal modeling problems, we can enable efficient and uniformly more accurate mesh designs that can then facilitate more dynamically correct calibrations of friction parameterizations.

Our premise is that the circulation and flow of water is largely driven and controlled by the representation of the physical system and the representation of the physical system is integrally related to the mesh sizing functions. Thus, the sizing functions need to be carefully considered for ensuring high fidelity coastal ocean hydrodynamic simulations that have a relatively low associated computational cost. This is particularly relevant for operational/real-time forecast systems in order to be practically computationally feasible. Many of the previously used *a priori* mesh size heuristics (e.g., topographic-length scale, and distance-to-shoreline) have proven useful in practice for producing accurate solutions for tides and storm surges. Thus, we have devised an approach that combines and builds on such mesh size heuristics to variably resolve shoreline geometry, seabed topography, controlling the geometric expansion of element sizes, and capturing submarine channels that convey flow into and out of the estuaries. Our ultimate goal is to represent the physical system and response with the fewest number of degrees of freedom while preserving the accuracy of the solution as compared to measured data. Here, we apply our approach to the widely studied ECGC region and conduct an in-depth analysis of the sensitivity of the barotropic tides to the domain discretization.

This paper addresses the following two questions:

a) How does the simulation of barotropic tides respond to the representation of shoreline geometry and seabed topography in the ECGC region? What are the sources of error and how do these contribute to the measured differences?

b) Can we incorporate our results from a) to make recommendations for a set of mesh size functions that place resolution according to shoreline geometry and seabed topography to efficiently discretize coastal ocean domains that approximately reproduce simulation results from an extremely well-resolved mesh?

3.3 Methods, Data and Tools

3.3.1 ECGC Study Domain and Data

The ECGC study domain for this work (Figure 3.1) contains a single open ocean boundary along the 60°W meridian which is placed here for geometric simplicity and because it lies in the deep ocean where the tides vary gradually and hence suitable for coupling to global tidal model solutions that are highly accurate in the deep ocean [139]. The placement of the open boundary in this way is sufficiently far from the coastal zones to represent tide responses throughout the ECGC domain [152].

The domain is classified into four distinct regions as shown in Figure 3.1 along with co-tidal and co-amplitude lines of the dominant constituents. The tides are predominately semi-diurnal dominated by the M_2 along the Eastern Coast of the

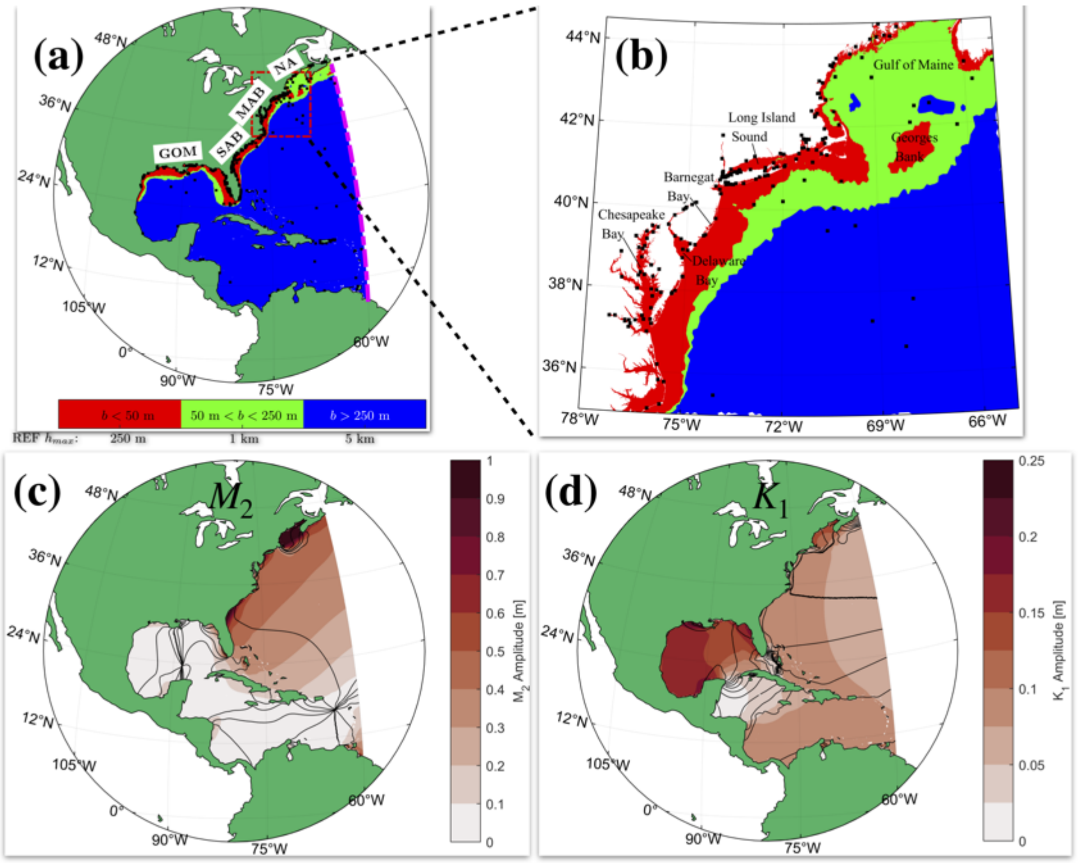


Figure 3.1. The study area in which colored zones in the center panel indicate the mesh size upper bounds (h_{max}) in the reference (REF) mesh (minimum mesh size is $L_{min} = 50$ m). The red and green colored zones together indicate the comparison zones for all the cumulative area fraction error curve calculations. The dashed magenta line indicates the open ocean boundary on which tidal elevations are specified. The top left and right panels indicate TPXO9.1 solutions of the M_2 and K_1 tidal constituent elevation amplitudes (colors) and phase contours in intervals of 30° (M_2) and 15° (K_1).

United States – North Atlantic (NA), Mid-Atlantic Bight (MAB), and South Atlantic Bight (SAB). In the western half of the Gulf of Mexico (GOM) the K_1 and O_1 dominate water level variations, while the eastern side is mixed-diurnal with the M_2, K_1, O_1, and S_2 contributing roughly in equal parts.

3.3.1.1 Bathymetric and Shoreline Datasets

The bathymetric data used for this study are primarily based on SRTM15+ [132] and supplemented in areas of overlap with the Coastal Relief Model [CRM; 5] in addition to local 1/3 and 1/9 arc-sec NCEI topo-bathymetric coastal elevation model datasets where available (`https://www.ngdc.noaa.gov/mgg/coastal/coastal.html`). The entire bathymetric dataset was integrated into a final digital elevation model (DEM) that was re-sampled on a uniform grid spacing of 3 arc-sec ($\sim$100 m), which is equal to the resolution of the CRM. For SRTM15+ and the CRM, the vertical uncertainty in the data is generally larger than the discrepancy between local mean sea level and the NAVD88 vertical reference datums, so no effort was made to rectify the vertical datum for these data. However, all NCEI local and regional datasets were adjusted to local mean sea level using VDatum [154] where the transformation was available. The horizontal datum of the re-sampled DEM is in geographic coordinates or WGS84.

Since the shoreline (where land meets the ocean in the temporal mean sense) as it exists in nature has a fractal geometry and is constantly evolving due to sedimentation and erosional processes, variations in discharge, sea level rise, and anthropomorphic effects, its exact representation may be intractable. For the purposes of this work, we consider a static version of the shoreline as depicted from the relatively recent (5-10 years old) topo-bathymetric data used in this study. A polyline that approximates the local mean sea level shoreline was extracted using the GRASS Geographical Information Systems r.contour module with a cut parameter of 150 [67]. While higher quality shoreline vector datasets exist, a preference was given to the shoreline extracted from the re-sampled DEM that was created for this work given that it would produce mesh boundaries that are aligned with the 0-m contour from the data sources. In other words, this helps to improve the agreement with the location of where the shoreline is when topo-bathymetric data is interpolated onto the mesh vertices. The discrete shoreline extracted from the DEM model can only resolve shoreline length-scales down to its horizontal resolution of 3 arc-sec (approximately 90 m).

3.3.1.2 Tide Gauge Data

Harmonic tidal constituent observations at tide gauges in ECGC (Figure 3.1) are used in this study to the validate the model simulations on selected meshes. The observations are predominantly made up of posted harmonic constituents at 636 National Oceanic and Atmospheric Administration (NOAA) coastal tide gauges (`https://tidesandcurrents.noaa.gov/stations.html?type=Harmonic+Constituents`). An additional 31 observations located on the continental shelf and in deep water [139] are also included (available from [1].

3.3.2 Hydrodynamic Model Configuration

This study uses the ADvanced CIRCulation model (ADCIRC) [96, 153] to perform the hydrodynamic simulations of two-dimensional (2D) barotropic tides. ADCIRC is a continuous-Galerkin finite element model that solves the shallow water equations (SWEs) using the Generalized Wave Continuity Equation [GWCE; 86, 100] on an unstructured triangular mesh [151]. It is numerically a second-order solver that discretizes the domain with linear elements.

[1] `ftp://ftp.legos.obs-mip.fr/pub/FES2012-project/data/gauges/2013-12-16/`

The governing equations are the shallow water equations (SWE) in primitive, non-conservative, and barotropic form:

$$\frac{\partial \eta}{\partial t} + \nabla \cdot (\boldsymbol{u} H) = 0 \tag{3.1}$$

$$\begin{aligned} \frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} + f\mathbf{k} \times \boldsymbol{u} + g\nabla(\eta - \eta_{EQ} - \eta_{SAL}) + C_f \frac{|\boldsymbol{u}|\boldsymbol{u}}{H} + \\ - \frac{1}{H}\nabla \cdot [\nu_t H(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^{\mathrm{T}})] = 0 \end{aligned} \tag{3.2}$$

where η is the surface elevation, $H = h + \eta$ is the total water depth in which h is the still water depth, $\boldsymbol{u}$ is the depth-averaged velocity vector, g is the acceleration due to gravity, $\mathbf{k}$ is the vertical unit vector, and $f = 2\Omega \sin \phi$ is the Coriolis parameter in which Ω is the angular speed of the earth, and ϕ is the latitude. The quantity η_{EQ} is the equilibrium tide, and η_{SAL} is the ocean self-attraction and loading term (SAL). In the dissipation terms, C_f is the coefficient of bottom friction, and ν_t is the horizontal eddy viscosity coefficient.

We perform all simulations with the following setup: the model is forced by astronomical tidal elevation open ocean boundary conditions, astronomical tidal equilibrium potential terms, and astronomical tidal self-attraction and loading (SAL) terms [76].

The time and space advective components were included in all calculations and wetting/drying is enabled although a mminimum depth is enforced on the shoreline of 1 m below sea level to ensure flow through narrow channels on the scale of the minimum resolution. A constant quadratic bottom friction was used with the standard coefficient of 0.0025. Horizontal dissipation was parameterized through a constant lateral eddy viscosity term of 50 $\mathrm{m^2 s^{-1}}$. The GWCE mass matrix is solved using explicit time discretization instead of the consistent semi-implicit method. This choice was not found to affect the simulation results at the 2 second simulation timesteps we are using here with the Courant-limited explicit timestepping scheme. Therefore, the explicit method was preferred due to improved computationally efficiency (approximately twice as fast) [145].

3.3.3 Mesh Generation

The construction of regional coastal ocean meshes for hydrodynamic simulations in models such as ADCIRC is an involved process with many degrees of variation. In order to analyze how mesh resolution may affect numerical simulations, it is vital to have an automated and reproducible workflow to systematically control aspects of the mesh design. By reproducible we mean that given the exact same inputs and options, the vertex locations of a new instance of the mesh will be approximately the same leading to negligible differences between simulation results repeated on various instances of the mesh. The approximate similarity of meshes is evidenced in results throughout the manuscript: nearly similar mesh designs exhibit the smallest relative differences between their solutions.

Some approaches and tools have been developed recently to make these workflows feasible [31, 57, 65, 131]. For this work, all unstructured meshes were developed with the *OceanMesh2D* software [129, 131]. *OceanMesh2D* is a self-contained MATLAB mesh generation toolkit for the development of 2D unstructured triangular meshes. Specifically, we use Version 2.0 of the software which is an extension of V1.0 [131] with support for mesh generation using map projections to ensure that meshes on the sphere conform to Earth's curvature and obey user-defined resolution requests which are specified in meters. Any map projection that is featured in the m_map mapping package [116] can be selected.

A number of ocean meshes are automatically generated in Lambert conformal conic projection space using the multiscale meshing approach [131], whereby multiple boxes are used to cover the region roughly indicated by the green and red colored zones in Figure 3.1(a)-(b). Inside these boxes, the minimum resolution L_{min} is specified to between 50 m and 250 m, depending on the experiment (see Section 3.3.4). A larger box covering the whole study region is used to mesh the rest of the domain with a minimum resolution of 1 km that is placed uniformly along the shoreline. The result is one seamless unstructured mesh, in which the software automatically smooths mesh resolution sizes between regions.

Topo-bathymetric data, available on a structured grid (DEM), is interpolated onto the mesh vertices using the grid-scale averaging approach that is built into mesh generation software [131]. Cell-averaging minimizes sub-grid scale noise arising from resolution disparity between that of the DEM and the mesh. In contrast, using linear interpolation in areas where the DEM has higher resolution than the mesh can lead to substantial changes in the interpolated seabed depth with even small variations in the vertex location. The minimization of sub-grid scale noise in the seabed topography is important in order to study the effect of mesh resolution on the solution.

3.3.4 Experimental Design

In Sections 3.4.1 to 3.4.4, five experiments are explored to examine the effects of targeted placement of mesh resolution at various seabed and shoreline features according to a mesh size function or constraint (Table 3.1). Within each experiment three meshes (categorized as 'fine', 'medium', and 'coarse' resolution) are generated by varying a single mesh size function parameter while holding all other parameters constant. All meshes require a minimum mesh size and an element-to-element mesh size gradation rate (henceforth referred to as gradation), which are set to 50 m and 15%, respectively, unless otherwise stated. The maximum mesh size is set to 10 km for all meshes.

The effect of the mesh size functions on the resulting triangulation's that are used in the various experiments (Table 3.1) are graphically illustrated in Figure 3.2, and described below:

- In the *distance* function (Figure 3.2(a)), mesh resolution is dictated by the minimum mesh size at the shoreline (L_{min}) and the maximum allowable expansion rate (g). The variation of L_{min} forms Experiment 1.

- The *feature size* function (Figure 3.2(b)) places mesh resolution according to the width of the geometric feature. The width is estimated as half the sum of the distance from a point in the computational domain to the shoreline plus the distance from the same point to the nearest medial axis (Figure 3.2(c)). Varying the number of elements per geometric feature width forms Experiment 2.

- The *gradation* function bounds the mesh size transitions on the structured grid that the mesh size function is calculated on, which will determine the gradation (g) on the mesh's triangulation. The variation of this parameter only forms Experiment 3.

- The *slope* function (Figure 3.2(e)) places mesh resolution according to the length of a topographic feature, targeting regions of high topographic gradients such as the continental shelf break and slope. Experiment 4 varies the number of elements per topographic length-scale.

- The *submarine channel* function (Figure 3.2(d)) targets mesh resolution along well-defined submarine channels such as dredged shipping channels or morphodynamic conveyances within estuaries that are identified through an upslope area calculation using a 1000 DEM cell minimum threshold in Geographical Information Systems software. Experiment 5 varies the number of elements per channel width. The channel width is estimated according to the seabed depth near the channel and an assumed slope angle of 30° with the seabed floor.

TABLE 3.1: EXPERIMENTS WITH MESH SIZE FUNCTIONS.

Experiments	Meshes Fine	Meshes Medium	Meshes Coarse	Mesh Size Function [m]	L_{min} [m]	L_{max} [km]	g [%]
1: Minimum Mesh Size	L50	L150	L250	$\mathrm{L}x = x + 0.15d_s$	x	10	15
2: Feature Size	FS8	FS4	FS2	$\mathrm{FS}x = \frac{d_s+d_m}{2x}$	50	10	15
3: Grade	G15	G25	G35	$\mathrm{G}g \Rightarrow 100\left\lvert\frac{L_i-L_j}{X_i-X_j}\right\rvert < g$	50	10	x
4: Slope	S20	S10	S5	$\mathrm{S}x = \frac{2\pi}{x}\frac{b}{\lvert\nabla b\rvert}$	50	10	15
5: Submarine channels	C1.0	C0.5	C0.1	$\mathrm{C}x = \frac{2\pi}{x}\frac{b}{\tan(30°)}$	50	10	35
Reference (REF)	-	-	-	$\mathrm{REF} = 50 + 0.15d_s$	50	$0.25 : b < 50$ m $1 : b < 250$ m $5 : b > 250$ m	15

L_{min}: minimum mesh size
L_{max}: maximum mesh size
L_i: mesh size at i defined by the circumradius of each triangle
X_i: coordinate of i on grid to compute edgelengths
i and j: adjacent elements
g: gradation
d_s: shortest distance to the shoreline
d_m: shortest distance to the medial axis
b: topo-bathymetric depth (positive below sea level)
∇: gradient operator

NOTE: The five experiments explored each containing three meshes in which the variable mesh size function parameter is indicated by x. The properties of the finely-resolved REF mesh used for a baseline comparison is also shown.

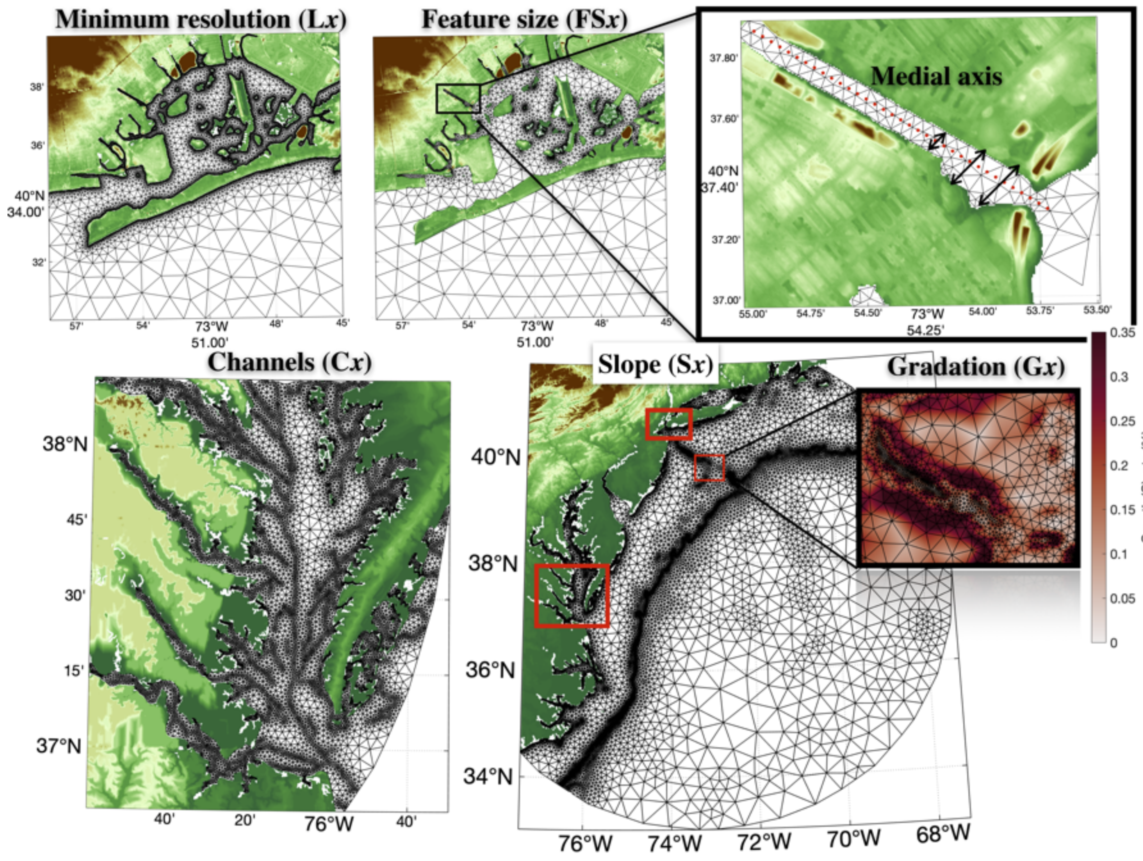


Figure 3.2. An illustration of the five mesh size functions that were investigated in and around the Mid-Atlantic Bight region along the Eastern United States coastline. Seabed topography is colored and relevant quantities are noted in the text.

A highly-refined reference (REF) mesh (Table 3.1) was generated to act as a proxy for the 'true' solution against which our meshes in the experiments are compared. In this mesh, a set of depth-based maximum element size constraints were used and a mesh size gradation of 15%. Specifically, the minimum mesh resolution is 50-m and the maximum resolution was bounded above by 250 m nearshore (depth, $b < 50$ m), 1 km on the continental shelf (50 m $< b <$ 250 m), and 5 km in the deep ocean ($b >$ 250 m). These mesh size constraints are conservative and they represent values that could be accommodated in terms of the total computational cost, Courant-based stability constraint, and the resolution of the geospatial data used ($\sim$100 m). The REF mesh contains N = 10,746,955 vertices and represents a mesh design that we classify as 'overly-discretized' in the sense that as this study will later demonstrate, it is possible to substantially reduce the vertex count while maintaining solution accuracy.

Each mesh was used to perform a 122-day tidal simulation to assess the effects on the astronomical tides due to variations in mesh design. In these simulations, ADCIRC is forced through the tidal equilibrium potential and SAL terms throughout the domain and at the open ocean boundaries with four major semi-diurnal (M_2, N_2, S_2, K_2) and four major diurnal tidal constituents (K_1, O_1, P_1, Q_1). Open boundary elevations are obtained from TPXO9.1 (`http://volkov.oce.orst.edu/tides/global.html`) tidal solutions; SAL terms are obtained from FES2014 tidal loading solutions (`ftp://ftp.legos.obs-mip.fr/pub/FES2012-project/data/LSA/FES2014/`). In the assessment of the results of these simulations, a focus is placed primarily on the variation in the major semi-diurnal tide (M_2) since this is the predominant tidal constituent along the ECGC. The major diurnal tide (K_1) is also included where relevant.

$$RE = \frac{A_{ID} - A_{REF}}{A_{REF}} \times 100[\%] \tag{3.3}$$

where A is the harmonic elevation amplitude of the tidal constituent in the experiment

(ID) and the REF meshes. A focus is placed on the M_2 and K_1 elevation amplitudes as these represent the predominant semi-diurnal and diurnal constituents reproduced in the ECGC domain (Section 3.3.1).

The calculation of the RE is proceeded in this manner to keep data extrapolation to a minimum so that the same shoreline geometric complexity as depicted in each mesh is present in both solutions under comparison. For all differences, statistics are only performed on vertices in which the absolute difference from REF exceeds 1 mm or the RE between solutions is greater than 0.1%. These significance values are considered sufficiently small to ignore for the modeling purposes of barotropic tides along the ECGC, which have magnitudes on the order of centimeters to meters.

The convergence characteristics of the experiments are examined by comparing the cumulative area fraction errors (CAFE) or the RE statistic along the continental shelf margins of the ECGC region ($b < 250$ m) where high mesh resolution zones were deployed (union of the green and red colored zones in Figure 3.1). To be consistent throughout, CAFE curves only consider errors that exceed 1 mm or feature a RE greater than 0.1% and are determined for the RE statistic. On these CAFE plots, the y-axis value of a point falling on these curves indicates the percent area having a difference greater (less) than the positive (negative) value on the x-axis. A solution that has “converged indicates that 99% of the comparison region has a $\pm 5\%$ RE. This definition of convergence may be arbitrary but it represents a statistic that can enable a consistent comparison between solutions.

Last, in Section 3.4.5 we summarize the experiments through the standard deviation of the variation in the RE statistics from the REF mesh. Further, the contribution of numerical error versus error in the physical approximation of the domain is illustrated. Finally, based on the results of the five experiments described above we generate mesh designs that combine mesh size functions/experiments together to achieve an efficient mesh design that can approximately mimic the tidal solution accuracy of the REF mesh.

The following set of statistics are computed to compare the accuracy, in terms of error against tide gauge observations (Section 3.3.1.2), of the simulated tidal solutions between the REF mesh and the combination mesh designs.

$$E = \left(0.5\left[A_o^2 + A_m^2 - 2A_oA_m\cos(\theta_o - \theta_m)\right]\right)^{1/2} \tag{3.4}$$

$$B = \frac{\sum_{t=1}^{T}(E_{ID} - E_{REF})}{\sum_{t=1}^{T} E_{REF}} \tag{3.5}$$

$$\gamma^2 = \frac{var(E_{ID} - E_{REF})}{var(E_{REF})} \tag{3.6}$$

where E is the complex root-mean-square error of a tidal constituent for one cycle and account for the amplitude and phase errors, A and θ are the amplitudes and phase lags of the tidal constituent respectively, the subscripts 'o' and 'm' refer to the observed and modeled values respectively, and T in the sum is the total number of tide gauges. B is the normalized mean bias and γ^2 is the normalized variance (var) of the discrepancies of E between the REF mesh and a particular mesh combination (ID). A positive value of B indicates that the mesh combination has on average greater values of E than REF, while a negative bias indicates the model is outperforming the REF solution. The smaller the value of γ^2, the more similar the mesh's solution is to REF in terms of the distribution of E. Since, any model can be tuned to fit observations, such as by employing variable bottom friction coefficients in regions where errors arise, the main aim here is to minimize γ^2 and B, thereby minimizing the effects of mesh resolution on the solution under the assumption that REF is sufficiently resolved. For reference, the REF solution has a median E for the M_2 of 3.9 cm (computed on all 667 tide gauges, Section 3.3.1.2).

3.4 Results

3.4.1 Resolving the shoreline

The representation of the shoreline determines the simulated accuracy in modeling the physical interaction between forcing agents (e.g., tides, winds, and waves) with shoreline geometrical features (e.g., coves, headlands, back-bays, and lagoons). From a modeling standpoint, the shoreline's representation must be simplified to satisfy computational resources by removing fine-shoreline details from the mesh's boundary description that are smaller than the minimum mesh resolution. However, when the shoreline is simplified, it alters the approximation of the physical domain, and hence possibly the system's tidal response [e.g., 69, 105].

This section uses the results from Experiments 1 (Lx) and 2 (FSx) to explore the effects of varying a specified minimum resolution at the shoreline and of varying shoreline resolution according to a feature size estimation, respectively. A comparative example of the Lx and FSx designs along an estuarine region is illustrated in Figure 3.3. As the minimum mesh resolution is coarsened from 50 m to 250 m, narrow waterways, tributaries, and estuaries that are smaller in horizontal length-scale than the minimum mesh resolution are automatically removed in the mesh generation process [131]. The removal of fine-scale shoreline geometry is considered a shoreline approximation error in the sense that the approximate representation of the shoreline departs from its representation in the original shoreline dataset. In contrast, the feature size approach creates a mesh that represents the physical system accurately by connecting small waterways together in a similar manner to L50, but requiring fewer vertices as resolution can expand in size away from geometric constrictions along the shoreline (Figure 3.3).

The shoreline approximation error is quantified by integrating the area enclosed by the polygonal region that defines the mesh boundary ($\mathcal{S}$ in which the sub-script

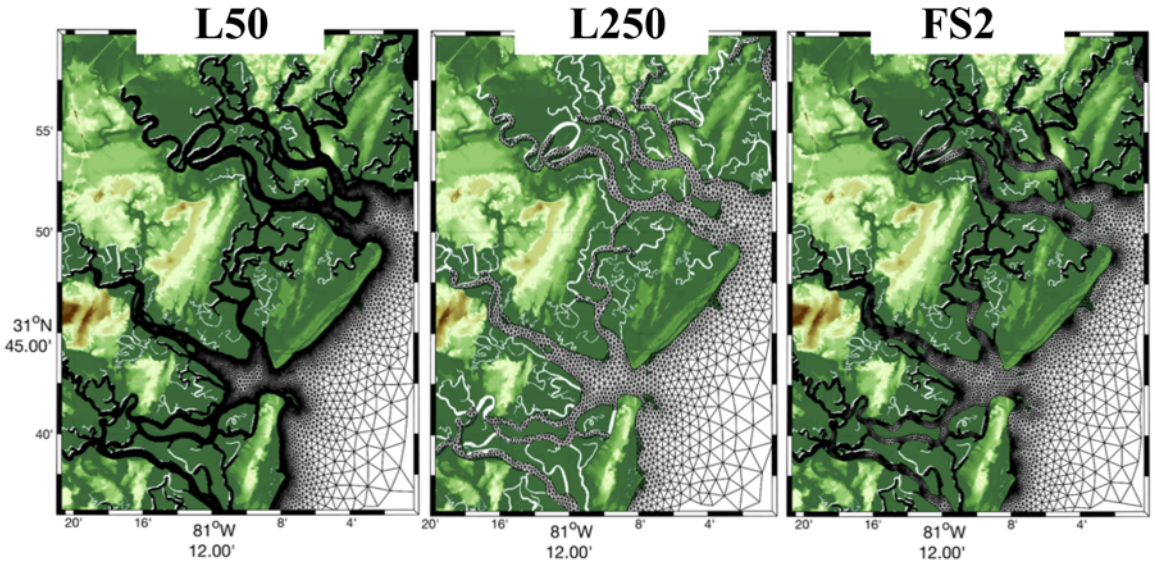


Figure 3.3. Mesh connectivity near Ossabaw Island, Georgia that illustrates changes to the capturing of narrow channel geometries as minimum mesh resolution is increased from 50 m (left) to 250 m (middle), and when using a shoreline width function that varies minimum mesh resolution between 50 m and 250 m (right) automatically based on shoreline geometric properties.

denotes the experiment ID).

$$\mathcal{A}_{error} = |\mathcal{S}_{ID} - \mathcal{S}_{Ref}| \tag{3.7}$$

$\mathcal{A}_{error}$ increases geometrically as the minimum shoreline resolution is coarsened from 50 m to 250 m in the Lx meshes (Figure 3.4). For example, $\mathcal{A}_{error}$ = 2,200 km^2 for L100 increases approximately ten-fold to $\mathcal{A}_{error}$ = 22,000 km^2 for L250, while the total vertex count reduces from 4.9 million to 0.8 million vertices between L250 and L50 mesh designs. In contrast, the FSx experiments exhibits no correspondence between total vertex counts and shoreline approximation error and $\mathcal{A}_{error}$ remains small reaching a maximum of approximately 1,500 km^2. The FSx design distributes 50-m mesh sizes in narrow waterways and along high curvature shoreline sections, while allowing mesh sizes to expand up to 250 m along straighter shoreline segments.

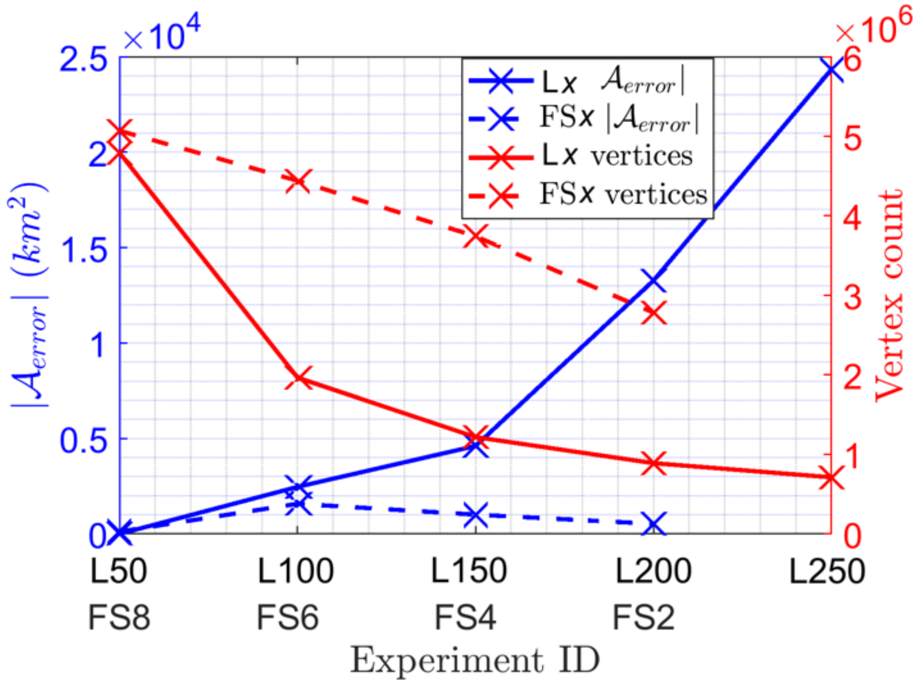


Figure 3.4. The shoreline geometry error $\mathcal{A}_{error}$, Equation (3.7), on the left axis for the meshes used in the shoreline approximation experiment along with the total vertex count for each mesh on the right axis. Solid lines represent data for meshes created with uniform shoreline resolution Lx and dashed lines indicate meshes created with the feature size approach FSx.

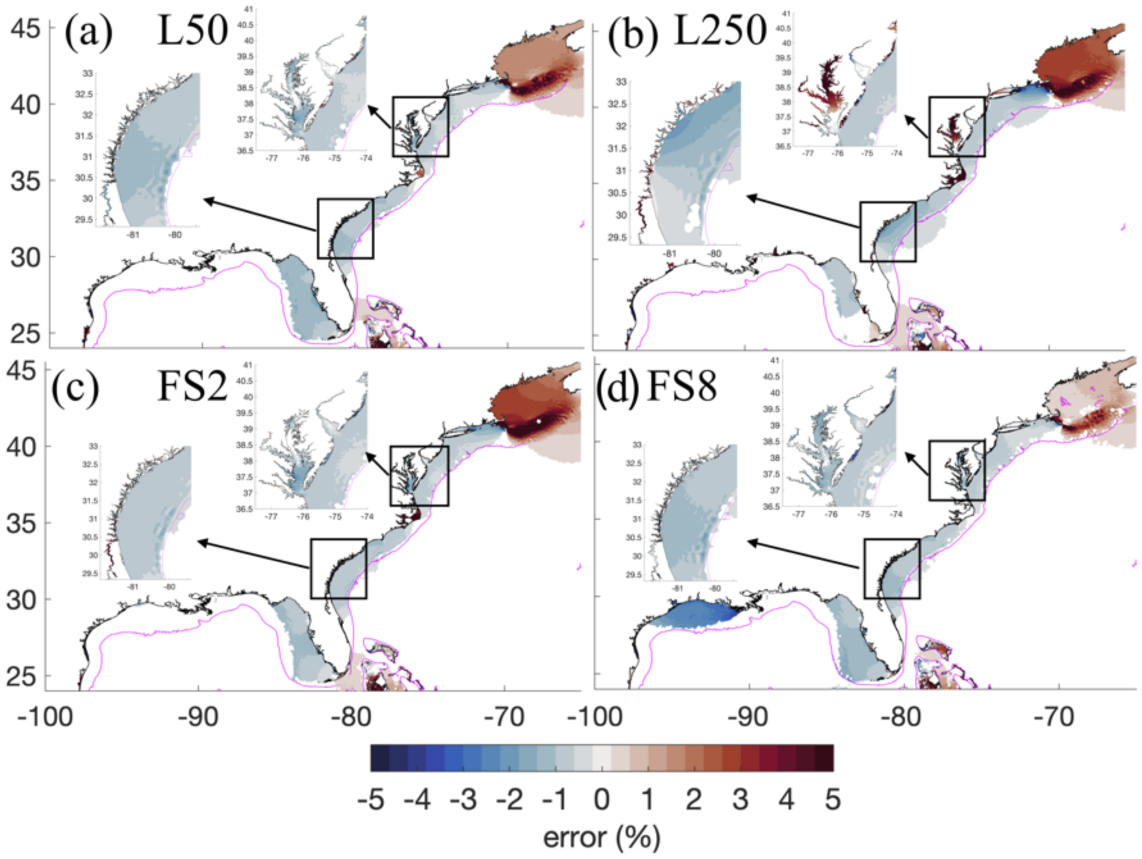


Figure 3.5. Panels (a)-(b) depict the relative error in the M_2 harmonic elevation amplitude from the REF solution when the minimum mesh resolution along the shoreline is coarsened from 50 m and 250 m. Panels (c)-(d) depicts the relative error (RE) in the M_2 harmonic elevation amplitude from solutions computed on meshes built with the feature size function. Insets around areas described in more detail are shown.

The predominate variation in vertex counts in the FSx design is the number of vertices per geometric width of the shoreline, not the minimum element size. Thus, the FS2 design is capable of preserving a similar amount of shoreline geometry as L50 (e.g., Figure 3.3a,c) but with approximately two times fewer vertices.

As is evident in Figure 3.5, the variation in the representation of the shoreline predominately affects the M_2 elevation amplitude in shallow shelf regions ($<$ 250-m depth range). A largely insignificant error ($<$ 1 mm or $\pm 0.1\%$) was observed in the K_1 elevation amplitude (not shown). The relative M_2 errors (RE) among the Lx experiments are greatest for L250 and smallest for L50 (Figure 3.5a-b), demonstrating the improvement of finer resolution. RE are focused in estauries in the SAB and in the MAB around the Chesapeake Bay and the Gulf of Maine where large RE values of 10-15% are found in the L250 mesh (Figure 3.5b). In the MA, SAB, and eastern GOM shelf zones, there is a weak 1-3% deamplification in the M_2 amplitude with the exception of the Chesapeake Bay estuary, which exhibits a pronounced RE of +5-10% as the mesh resolution is coarsened from L50 to L250. In general, the FSx meshes (Figure 3.5c-d) produce similar relative error patterns to the Lx meshes. However, negative RE values are only $<$ 1% in the Chesapeake and SAB for the coarsest Lx design (FS2) compared to RE values in L250 which are approximately $\pm 3\%$ here. Further, FS2 reduces the amplification in the Gulf of Maine by a small amount $\sim$1%. The western GOM shelf region weakly deamplified by 1-3% in the FS8 design, but this was not observed in the other Lx designs.

Although local differences in RE are illustrated in Figure 3.5, the CAFE curves demonstrate remarkable similarity in 99% of the comparison zone between the Lx and FSx solutions (i.e., above the thick 1% cumulative area line) for both the M_2 and K_1 elevation amplitudes (Figure 3.6). The CAFE curves for the M_2 are asymmetrical and indicate more of the domain has a positive error, which is accentuated in the tails below the 1% cumulative area line. While all the solutions in this experiment

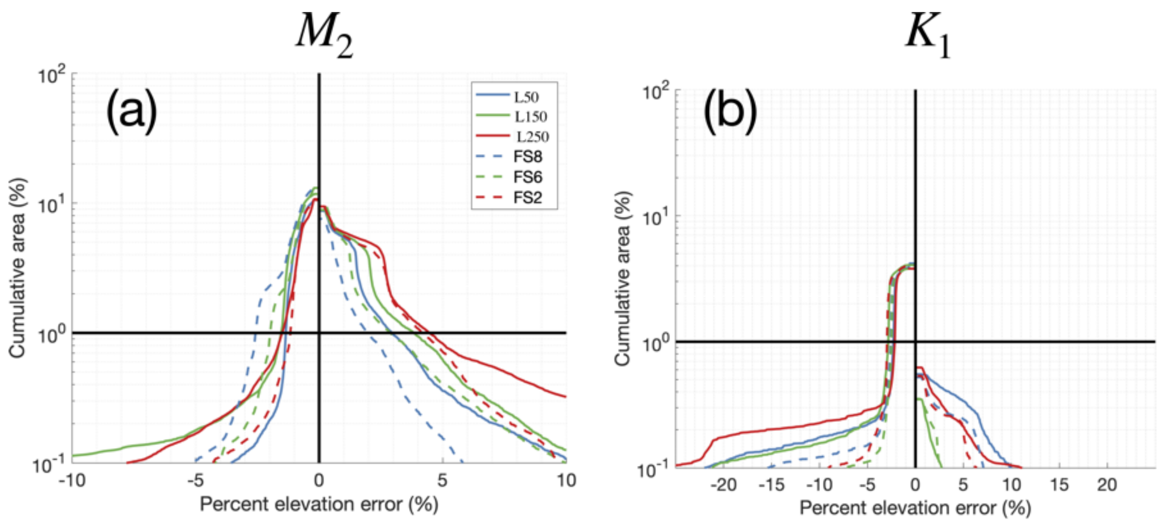


Figure 3.6. The cumulative area fraction error (CAFE) from the REF solution in the comparison region for panel (a) the M_2 elevation amplitude and panel (b) the K_1 elevation amplitude. The dotted lines denotes solutions computed on meshes that use the FSx design while the solid-lines denote meshes created with the Lx design.

have achieved a converged solution, the FS6 and FS8 contain less positive RE than the Lx designs, while the opposite is true for the negative crossing although the difference is marginal (1-2%). s The relatively coarser L250 (+4.0% RE) and FS2 (+3.9% RE) mesh designs exhibited only slightly larger positive errors in the M_2 elevation amplitude as compared to L50 and the FS8 design. These differences are marginal considering the 4 million total vertex count difference between the fine and coarse mesh designs (i.e., L50/FS8 vs. L250). For the K_1, all meshes have converged solutions to our tolerance and respond far less to alterations in mesh design than the M_2.

3.4.2 Mesh size gradation

The concept of grading is a key capability of unstructured mesh finite element or finite volume modeling in which coarse elements in the far-field grade smoothly into the more finely resolved region of interest to efficiently discretize regional and global ocean domains. This gradation rate between zones of variable resolution can greatly influence the number of vertices in the mesh (Figure 3.7). Elemental size grading has been based on bounding an estimate of the Courant number to encourage numerical stability [98]; however, the grade can also be based on geometric criteria by ensuring that neighboring mesh element sizes cannot enlarge too quickly [4], i.e., the gradation is bounded above by a maximum value. It is understood from a general modeling point of view that excessive gradation rates lead to triangles with acute or obtuse angles, which can impact the stability and numerical accuracy of the model [102, 138]. Further, the analysis by Hagen et al. [71] for one dimensional domains demonstrates that a high gradation value ($g \approx 0.5$) leads to the introduction of odd order truncation error term, which lowers the order of the method to first-order accurate.

A higher valued mesh size gradation will degrade the approximate representation of the domain by creating comparatively coarser mesh sizes away from the targeted zones of fine resolution. Note that the mesh generator is bounding the gradation rate above by the user-defined parameter value only on the mesh size function and it is assumed that given the convergence of the mesh generator the gradation rate is similarly bounded in the triangulation (Section 3.3.4). Coarser mesh sizes tend to smooth the interpolation of seabed features onto the mesh vertices and this data interpolation effect can be quantified in the meshes by calculating the overall volume enclosed by the mesh while holding the shoreline boundary fixed (i.e., the surface area of the total mesh is constant). Thus, similar to the shoreline approximation error (Eq. 3.7), the seabed approximation error is calculated as the absolute difference in total volume from the REF mesh:

$$\mathcal{V}_{error} = |TV_{ID} - TV_{REF}| \tag{3.8}$$

where TV is the total mesh volume for the mesh denoted by ID and is calculated as the sum of all the mesh element volumes. An element volume is calculated by multiplying the average depth of the element by its area. Since the REF mesh employs uniform high resolution mesh sizes throughout the nearshore and continental shelf zones (c.f., Figure 3.1), it represents the seabed surface with the smallest approximation error, which implies that the REF mesh encloses the largest total volume in the ECGC domain. Note that the data interpolation approach we are using is a grid-scale average (Section 3.3.3) and is not a globally conservative interpolation scheme. From Figure 3.7, it is apparent that there is a diminishing reduction in the total vertex count of the mesh with increased gradation. For the purposes of this study, we were not able to explore meshes with gradation greater than 35% due in Experiment 3 (Gx) due to the introduction of triangles with very skewed aspect ratios and obtuse and acute angles that created numerical accuracy issues.

The increase in mesh size gradation from 15% to 35% leads to a highly amplified error pattern in the NA region for both M_2 and K_1 constituents as well as along the MAB for M_2 (Figure 3.8). In the NA subdomain (Gulf of Maine), the M_2 RE is increased from 2-5% for G15 to 10-21% for G35 (colors are saturated in Figure 3.8b), in which the maximum RE is focused on the Georges Bank. In the opposite direction, the K_1 RE is nearly uniformly decreased from -3% for G15 to -6% for G35 in the NA subdomain. The M_2 RE in the MAB, SAB, and eastern GOM tends to weakly deamplify by approximately 1% to 5% along the continental shelf zones. In contrast to the shoreline approximation experiment, a relatively large deamplification of the M_2 RE occurs in both the Chesapeake Bay and Delaware Bay as the gradation is enlarged (Figure 3.8a,b). The M_2 RE reaches as high as 15% in this region for the G35 experiment (colors are saturated in Figure 3.8a).

As the mesh size gradation grows, the tidal elevation amplitudes start to diverge substantially from the REF solution (Figure 3.9). In 99% of the comparison zone, the G15 mesh has an M_2 error between -1.3% and +3.0% RE whereas G35 has between

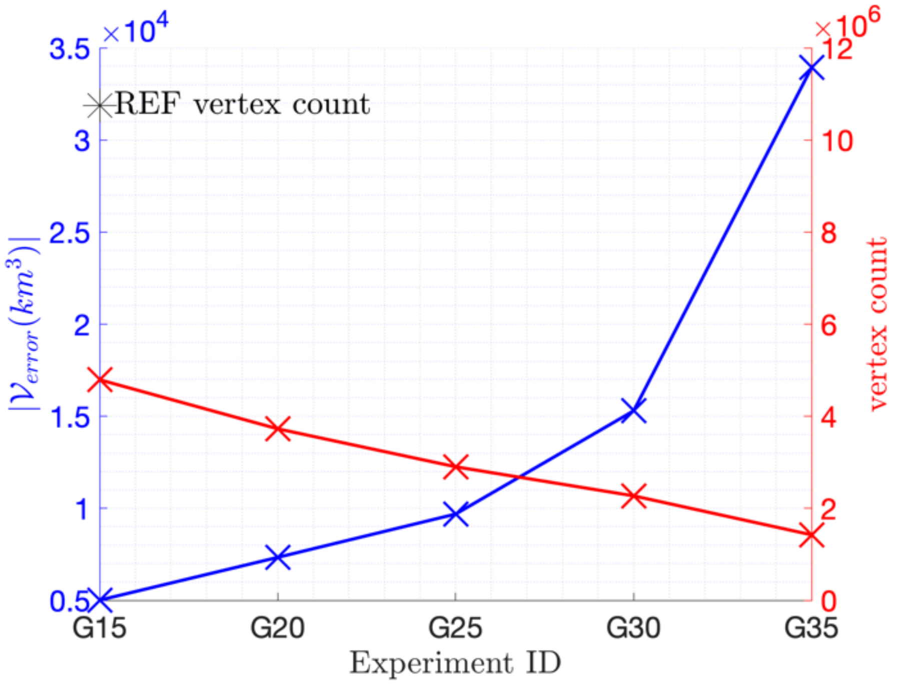


Figure 3.7. The seabed approximation error $\mathcal{V}_{error}$ (Equation 3.8) on the left-axis (blue x's) as the mesh size gradation is increased from 15% to 35% in increments of 5% while the shoreline boundary is held fixed (i.e., area of domain is constant). The total vertex count N_{vertex} in each mesh on the right axis (dashed red x's). The REF mesh vertex count is demarcated by a black asterisk in the top left corner of the figure.

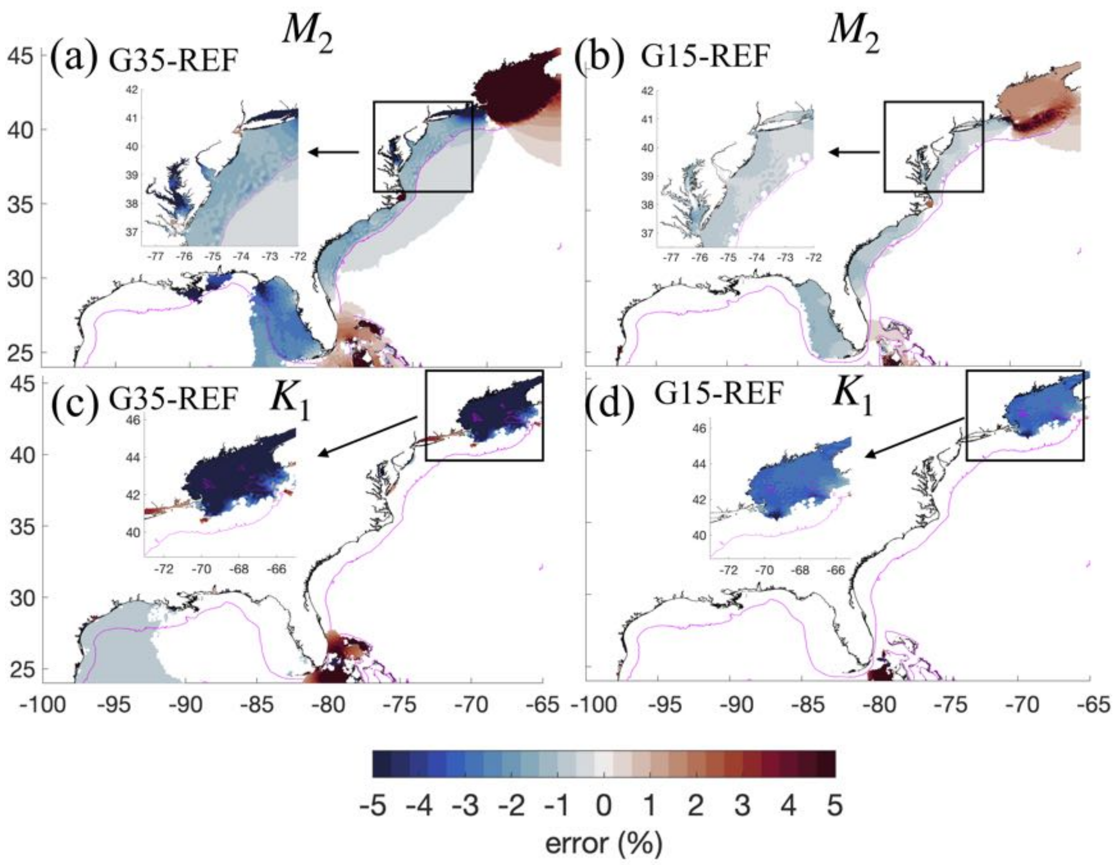


Figure 3.8. Panels (a)-(b) illustrate the RE in the M_2 elevation amplitude from the REF solution as the mesh gradation bound is increased to 35% while in panel (b) it is kept low at 15%. Panels (c) and (d) are the same as panels (a)-(b) but for the K_1 elevation amplitude. The 250-m isobath contour is drawn as a magenta line in each panel for reference. Insets are shown to reflect areas that are described in the text.

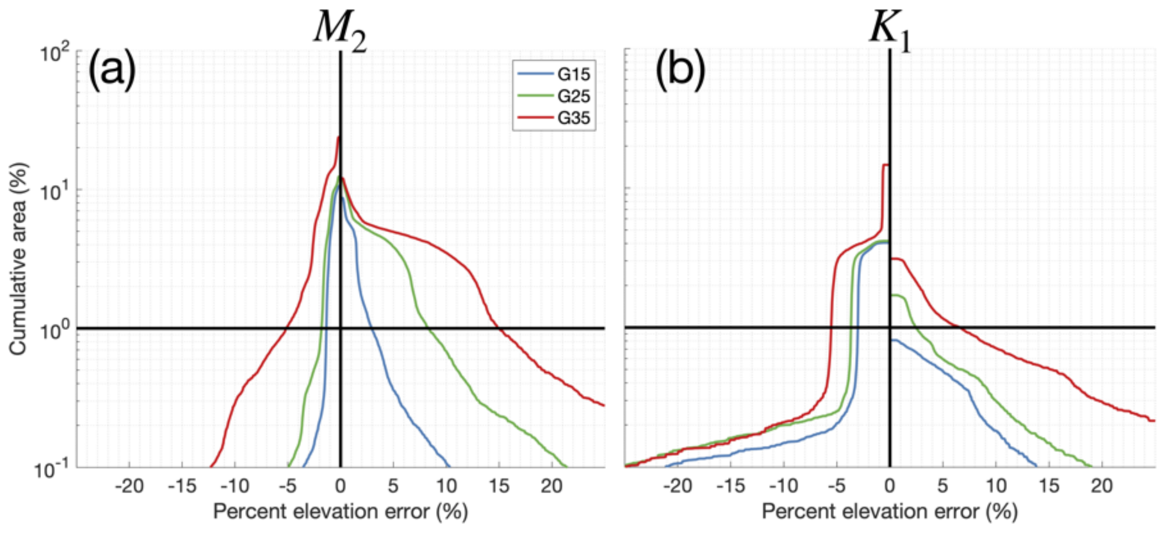


Figure 3.9. The cumulative area fraction error (CAFE) in the comparison zone (c.f., Figure 3.1) in panel (a) for the M_2 elevation amplitude and panel (b) for the K_1 harmonic elevation amplitude using the meshes created for the mesh size gradation experiment.

-5.0% and +15% RE. Furthermore, the G35 mesh design exhibits between -5.5% and +6.5% K_1 RE in the 99% comparison zone, which is compared to -3.0% and +0% K_1 RE for G15. Unlike the shoreline experiment where all meshes converged, only the G15 mesh converges for the M_2 constituent, and the G15 and G25 meshes converges for K_1.

3.4.3 Resolution along bathymetric gradients

The main motivation for increasing the horizontal resolution in the open ocean is to more accurately represent sharp seabed gradients, particularly those that characterize the continental shelf break and slope. The representation of these seabed gradients is captured with the topographic-length-scale Sx (Figure 3.2 and Table 3.1). The topographic-length-scale Sx is considered a useful mesh heuristic [see 69, for a review] to aid in the modeling of shelf break dynamics [74, 79, 98], subtidal dynamics [37, 95], and internal tide generation processes [157] and their effects on barotropic tides [122, 123]. Further, Hagen et al. [72] has demonstrated that an inadequate prescription of resolution along sharp seabed gradients is a source of numerical truncation error for tidal models. However, as $b \rightarrow 0$, the Sx meshing criteria fails as resolution becomes excessively fine in shallow depths both violating the Courant number and introducing unnecessarily fine details into the problem.

The topographic-length-scale Sx parameter must consider the trade-off between the improvement to the solution of barotropic tides and the additional mesh vertex count. Chen et al. [37] suggested resolution sizes between 3.3 to 6 km to capture the shelf break and 2 km to capture the deep slope in the Arctic Ocean. Lyard et al. [99] suggested S15 globally using quadratic finite elements, but noted that this value was restricted in its spatial application due to the excessive computational expense it incurred. In our studies, besides the excessive computational expense incurred by the additional degrees-of-freedom, we have found that using Sx larger than S20 leads to resolution along the shelf-break that can extensively restrict the feasible time step (i.e., time step of 2 s with Courant number bounded to 0.5). Note that the Sx heuristic is only applied where $b > 50$ m to avoid issues in shallow depths, where data many be highly noisy and contain many small-scale features such as channels that we propose an alternative strategy to resolve documented later on.

In Experiment 4 (Sx) the vertex count is increased by 4% to 20% over the L50 mesh, accompanied by improvement to the physical domain approximation, as illustrated along a transect spanning the cross-shelf direction in the MA region (Figure 3.10). Mesh resolution in the vicinity of the shelf break zones is enhanced to approximately 1.2 km and 0.8 km for S5 and S20, respectively. A point worth noting is that seabed features exist on the continental shelf break, such as the drowned Hudson river valley, which will otherwise be completely smoothed over without the Sx heuristic. In comparison, without Sx, resolution is coarser than 8 km (close to the maximum resolution size) in the vicinity of the shelf break (see L50 in Figure 3.10b),

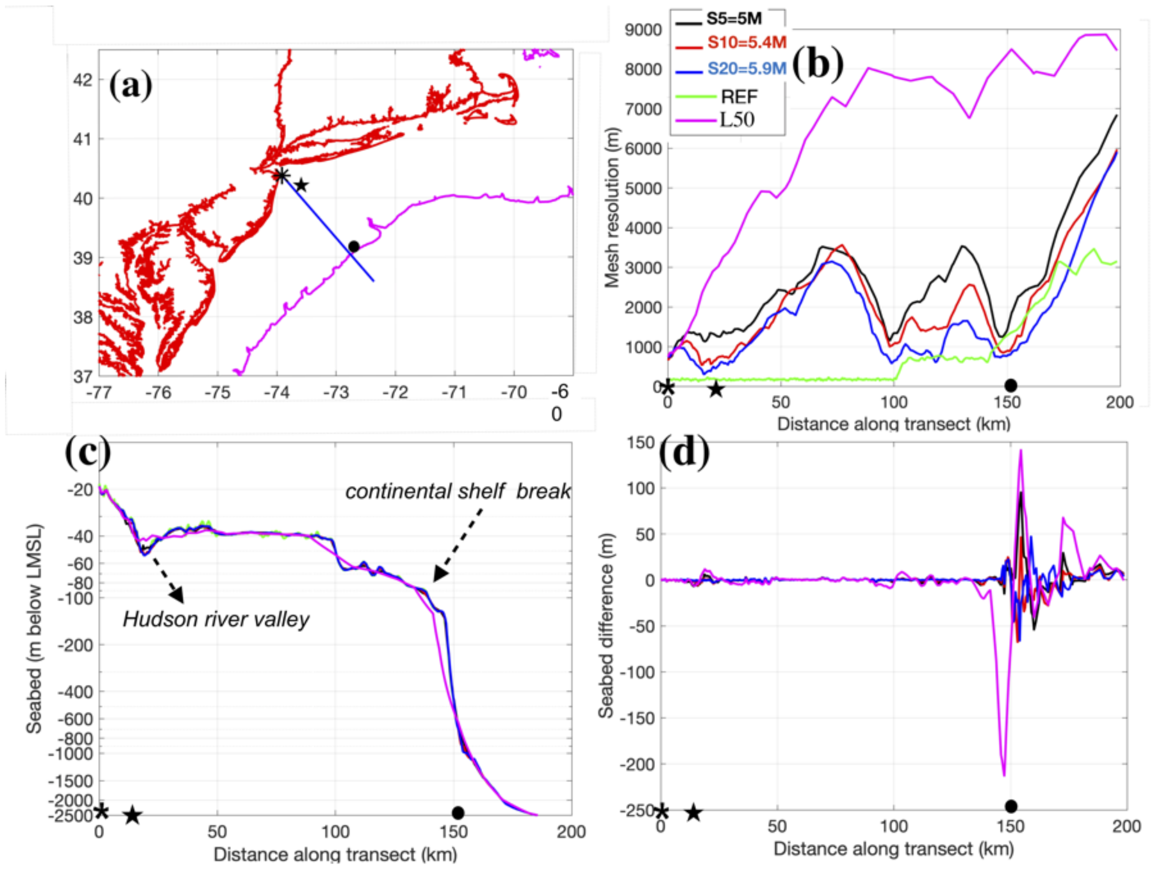


Figure 3.10. (a) A cross-shelf transect in the MAB region indicated in blue with the asterisk indicating the start of the transect, the magenta line is the 250-m isobath, and the red line is the shoreline; (b) the mesh resolution along the transect for the Sx, REF, and H50 meshes. Panel (c) illustrates the seabed topography along the transect for each mesh. Panel (d) illustrates the difference in seabed topography from each mesh and the REF mesh along the transect.

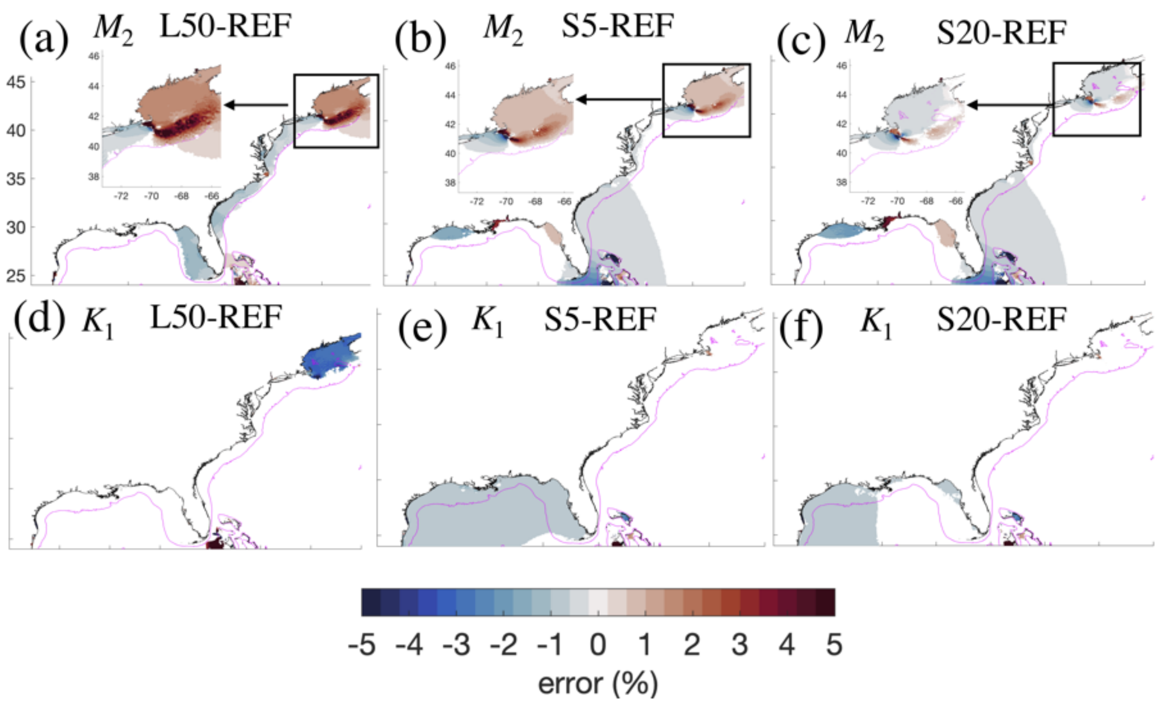


Figure 3.11. Panels (a)-(b) depict the M_2 elevation amplitude RE for solutions computed on the Sx meshes. Panels (c)-(d) depict the RE the K_1 elevation amplitude.

which tends to shift the break zone shoreward and result in a smoother and more gradual representation of the seabed profile along the transect (Figure 3.10c). The Sx heuristic results in a clear improvement in the depiction of the seabed profile. S20 had seabed profile differences of less than 50 m from the REF mesh, whereas the seabed profile difference for L50 is as large as 200 m (Figure 3.10d).

The finer resolution along seabed gradients using Sx leads to a significant overall reduction in the error pattern associated with the M_2 elevation amplitude in the MA and NA subdomains (Figure 3.11a-c), with the M_2 error pattern diminished almost entirely for S20 (Figure 3.11c). Note that although the largest RE is co-located with the phase convergence zone of the M_2 tidal species in the MA and NA domain (where the elevation amplitude is zero), the RE is not confined to solely the amphidromic point and emanates around the entirety of the NA subdomain. Similarly, for the K_1

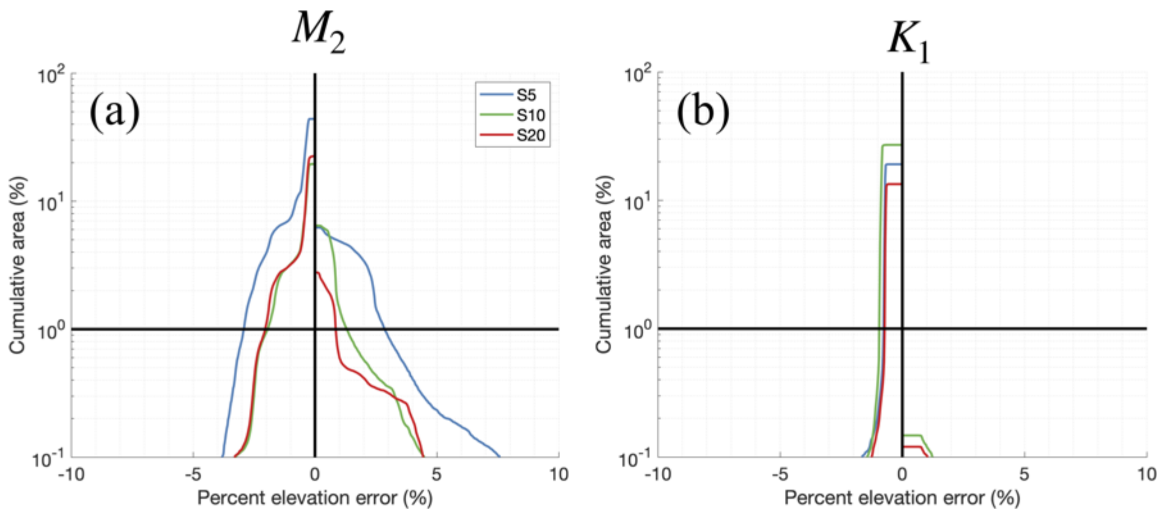


Figure 3.12. The cumulative area fraction error (CAFE) curves in the comparison zone for the Sx meshes.

elevation amplitude, an approximately -4% RE in the NA subdomain for the L50 (S0) mesh, is undetectable for any of the Sx meshes (Figure 3.11d-f). Contrastingly, in the GOM domain the application of Sx tends to introduce differences from the REF mesh rather than reduce them. Upon inspection, the REF is less resolved in parts of the GOM, Bahama Banks, and the Caribbean Sea (c.f., Figure 3.1) in comparison to the Sx meshes here, possibly explaining this result.

The CAFE curves for M_2 and K_2 (Figure 3.12) clearly illustrate that increased resolution along seabed gradient leads to a converged solution in 99% of the domain for S5, S10 and S20 according to our tolerance level of $\pm 5\%$. The S5 mesh has the largest M_2 error of $\pm 2.9\%$ RE in 99% of the comparison zone, which predominantly corresponding to the errors in the MA and NA domains. As evident from Figure 3.12, the K_1 was far less responsive to the choice of Sx mesh design than M_2, with differences on the order of $\pm 1\%$.

3.4.4 Cross-sectional representation of estuarine channels

Estuarine hydrodynamics are controlled by the depth and form, together referred to as the morphology, of the estaurine seabed [54, 61, 114, 121]. Thus, when designing a model to simulate coastal hydrodynamics, it is important to apply sufficient resolution to approximate the nearshore seabed topography. In particular, coarse mesh resolution in the presence of fine and narrow channelized bed forms will alias the channel's cross-sectional profile (Figure 3.13a,b) and lead to the inaccurate computation of transports and fluxes [69, 105]. In the boarder context of mesh generation techniques for coastal ocean modeling, mesh design heuristics that target resolution inversely proportional to seabed's depth [e.g., 153] will also tend to coarsen the resolution in the center of the estuary in the deepest component of the tidal channel. Thus, the pre-existing techniques used to build coastal models are inefficient at adequately resolving long and narrow channelized bed forms that are critical to conveying water into and throughout inlets.

An automatic mesh size function Cx that localizes finer mesh resolution in close proximity to the thalwegs of important estuarine channel morphology was developed as part of the *OceanMesh2D* meshing software suite [131]. An example of the *estuarine channel* mesh size function Cx is illustrated in Figure 3.13(c) for the Delaware Bay estuary located in the MA region. With 44% less vertices than REF in this subset of the ECGC, the C0.5 mesh represents the cross-sectional area of the deepest thalweg in the estuary with the same accuracy. In comparison, the L50 mesh is only 8 m deep at the thalweg compared to almost 14 m in the REF and C0.5 meshes. Notice that other less pronounced thalwegs are not captured by C0.5 due to the application of finer resolution.

The effects of the *estuarine channel* mesh size function have been investigated in Experiment 5 (Cx) using a mesh size gradation of 35% (G35). A high gradation motivates the resolution targeting approach because mesh element sizes are relaxed

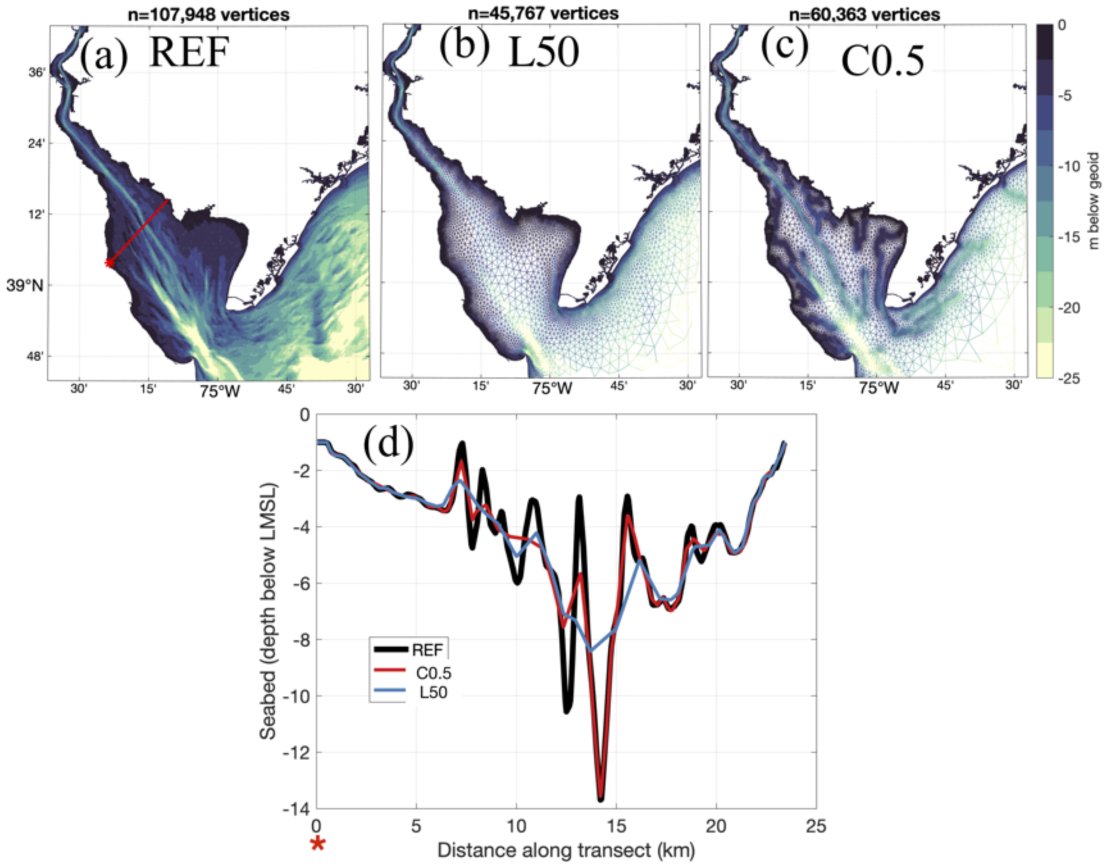


Figure 3.13. Panels (a)-(c) show sections of meshes in the Delaware Bay estuary and their interpolated seabed data to demonstrate the effect of variably resolving channelized seabeds. Panel (d) illustrates the cross-sectional profile of a tidal channel that is annotated as a red line in panel (a).

quickly away from the channel thalwegs where finer resolution is applied, thus obtaining an efficient mesh. The mesh vertex count in the finest Cx mesh (C1.0) is increased by more than two-fold from the G35 mesh to approximately 3.1 million vertices (Figure 3.14d-f), still approximately 60% of the G15 mesh vertex count.

The refinement of the estuarine channel network primarily impacts the M_2 elevation amplitude solution locally in the estuarine regions of the MA and NA subdomains (Figure 3.14a-c). A consistent reduction in M_2 RE from the high mesh size gradation solution (G35) is observed, particularly the 5-10% RE under-prediction error in large estuaries such as the Chesapeake Bay, Delaware Bay, and Long Island Sound. The remaining under-prediction error in these large estuaries is under 1-2% RE for the C1.0 mesh. Some smaller-scale estuarine systems also exhibit reduction to the RE. For example, the large negative error for G35 (<-5% RE) in Barnegat Bay (c.f., Figure 3.1) off the coast of New Jersey is reduced to the point that the error changes sign for C1.0 (+1-2% RE) (Figure 3.14a-c).

Similarly, the CAFE curves also demonstrate a consistent reduction in M_2 and K_1 RE in the comparison zone for the Cx meshes and a substantial reduction of RE as compared to the solution computed on G35 (Figure 3.15). While none of the meshes have converged with the application of resolution along estuarine channels, the sequence exhibits convergence. Despite the approximately 0.7 million vertex count difference between the C0.5 (2.4 million vertices) and C1.0 (3.1 million vertices) meshes, their associated solutions perform similarly. In 99% of the comparison zone, the C1.0 mesh M_2 error ranges between -1.6% and +5.5% RE, and -2.8% to +0% RE for the K_1 producing non-converged solutions for the M_2 but converged solutions for the K_1. Nevertheless, the narrowing of the error range in 99% of the comparison zone for the Cx meshes over that of the G35 mesh (-5.0% to +15% for M_2) even though the same 35% gradation is employed is substantial.

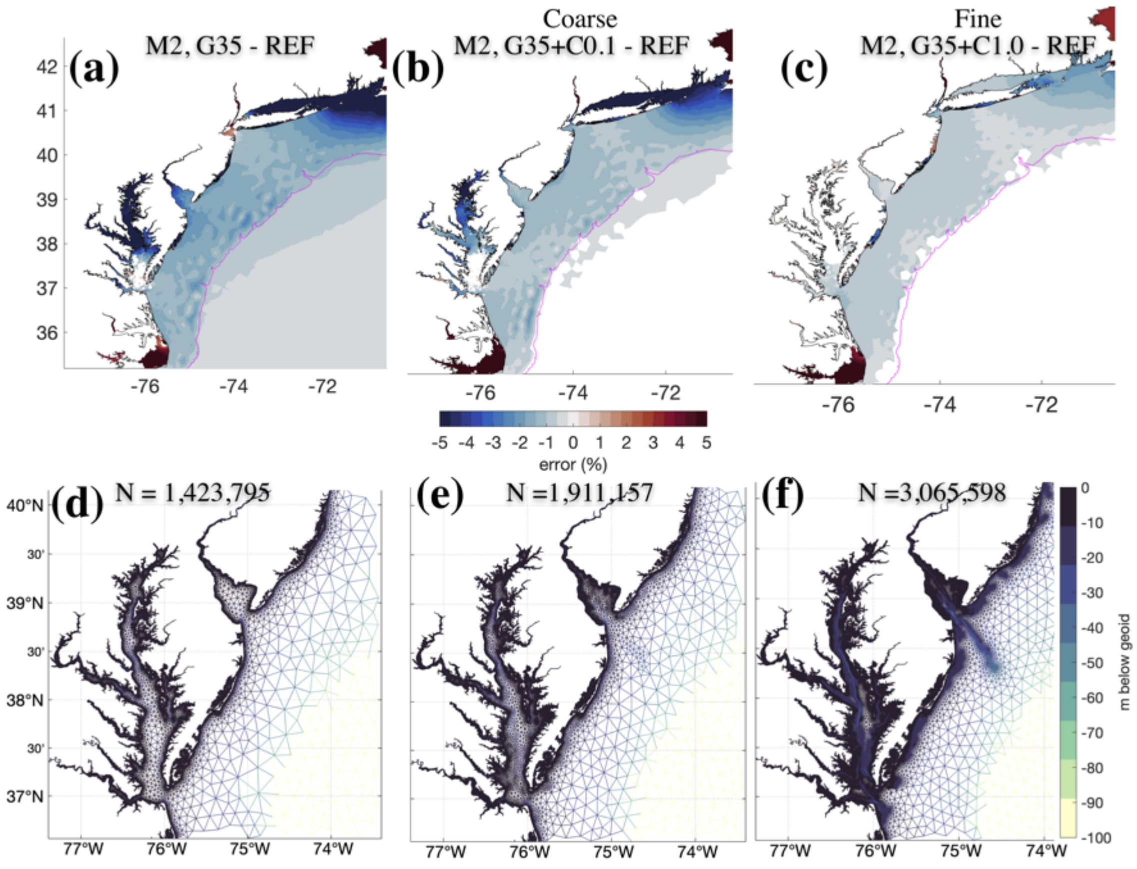


Figure 3.14. Panels (a)-(c) illustrate the error in the M_2 elevation amplitudes for solutions computed on meshes that variably resolve drainage networks that approximate tidal channels. Panels (d)-(f) indicate the mesh and interpolated seabed topography onto the mesh vertices. On panels (d)-(f), the total vertex count is indicated in the visualized portion of the mesh.

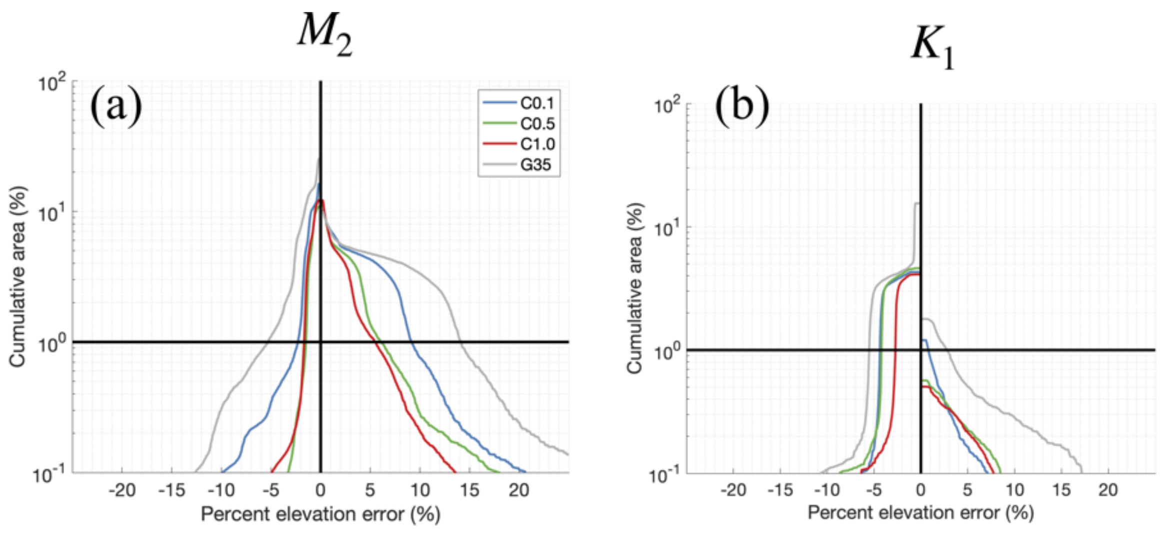


Figure 3.15. The cumulative area fraction error (CAFE) curves in the comparison zone for the Cx meshes.

3.4.5 Summary of experiments

3.4.5.1 Predominant variability

A summary of the variation in amplitude errors throughout the ECGC region in response to changes in mesh resolution from all 15 meshes over the five experiments (Table 3.1) is summarized by taking the standard deviation (σ) of RE and the dimensional error, AE $= A_{ID} - A_{REF}$ (Figure 3.16). The greatest changes in the M_2 elevation amplitudes are collocated with M_2 phase convergence zones and amphidromic points (c.f., Figure 3.1), and in some large and small estuaries such as the Chesapeake Bay and the Deleware Bay. In the Gulf of Maine, NA which is a resonant basin with a large tidal range (2-10 m), σ_{RE} is 1-4% and σAE is well above 2.5 cm for M_2. The K_1 differences in the Gulf of Maine are also larger than most other regions. In the GOM which has a small semidiurnal tidal range, σ_{RE} is large in the central region around the convergence zone for M_2 but this only corresponds to less than around 2 mm of dimensional variability (σ_{AE} is very small). In general, the K_1 is noticeably less responsive to changes in mesh resolution with σ_{RE} barely

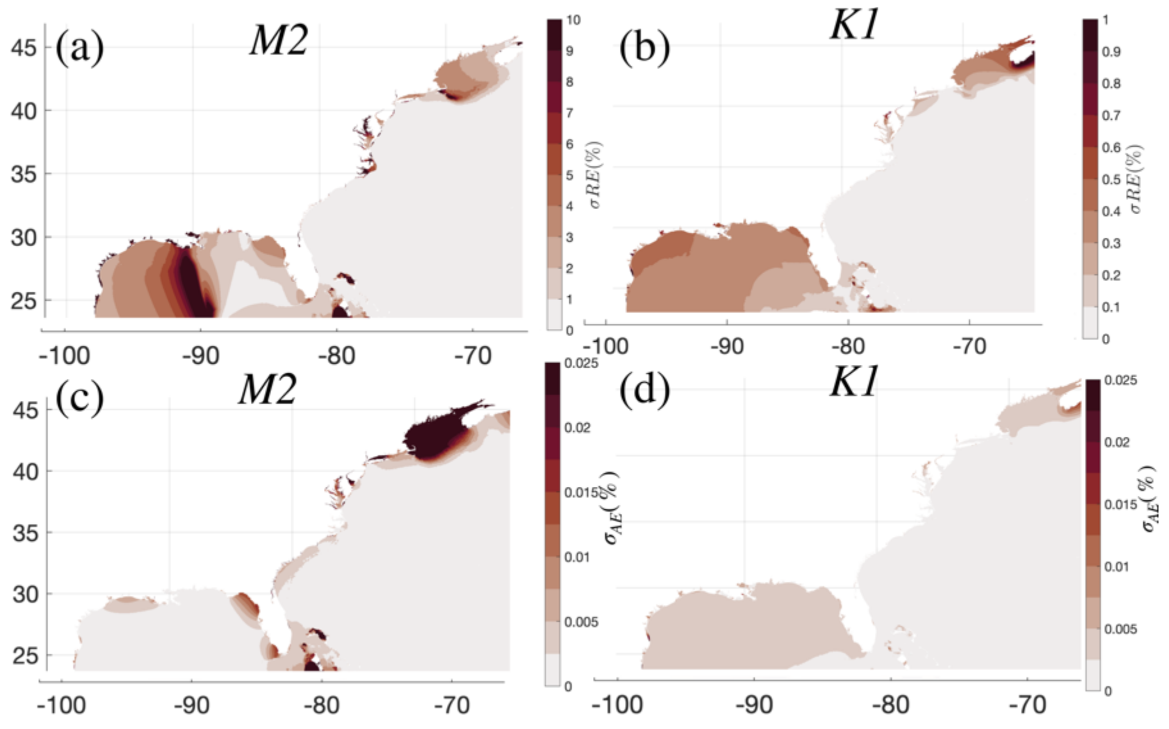


Figure 3.16. Standard deviation of the relative error (σ_{RE}) in (a) the M_2 and (b) the K_1 elevation amplitudes for all 15 meshes from the five experiments (Table 3.1). Panels (c) and (d) are the same but for the standard deviation of the dimensional errors (σ_{AE}). Note the differences smaller than the significance threshold defined in this paper *are* shown and that the colorbars are *not* the same between panels (a) and (b).

exceeding 1%. The K_1 exhibits the greatest variation in the NA subdomain (Gulf of Maine), in large estuaries, and throughout most of the GOM. The relatively small response in the K_1 is to be expected given that it is less energetic and has a longer wavelength than the M_2, and it does not typically exhibit resonance on wide shelves [39].

3.4.5.2 Numerical error versus physical approximation error

An outstanding issue with the results is that the numerical and physical approximation component of error are intertwined both contributing to the RE observed in the experiments. As the approximation of the bathymetry and shoreline boundary becomes more accurate with the application of finer resolution, the study of convergence in the tidal response becomes challenging as new bathymetric and shoreline features emerge. From a model design point of view, the isolation of the numerical component of the tidal error can provide clarity into how to improve the physical approximation component of error.

To isolate the numerical error in the tidal harmonics studied here, changes in the physical domain approximation was held constant by refining the relatively lightweight L250 mesh so that all triangular edges, except for those within 1° of the open ocean boundary, were bisected about their midpoints producing four new triangles for every pre-existing one following a shape-preserving scheme [56]. The bathymetry from the L250 mesh was linearly interpolated onto this new refined mesh (L250R1) ensuring that the approximation of the seabed topography are identical between the two meshes. Further, the bisection of the elements preserves the representation of the shoreline geometrical features between meshes. The numerical error was then estimated with Richardson extrapolation technique [19, 128]. In order to use this approach to estimate numerical truncation error, it was first verified that the leading order error terms indeed controlled the numerical convergence (i.e., asymptotic regime), spatial errors were found to be orders of magnitude greater than the time discretization errors, and the ADCIRC solver in the current configuration is a second order accurate method in space and time [96].

The Grid Convergence Index (GCI) following [128] is calculated to estimate numerical error with the following formulas:

$$
\begin{aligned}
GCI[\text{coarse mesh}] &= \frac{|\epsilon| r^n}{(r^n - 1)} \\
GCI[\text{fine mesh}] &= \frac{|\epsilon|}{(r^n - 1)} \\
n &= \text{spatial order of ADCIRC} = 2 \\
\epsilon &= 100 \times \frac{|\tilde{f}_{L250} - \tilde{f}_{L250R1}|}{\tilde{f}_{REF}} \\
r &= \frac{X_{L250}}{X_{L250R1}} = 2 = \text{refinement factor}
\end{aligned}
\tag{3.9}
$$

where $\tilde{f}_{L250}$ and $\tilde{f}_{L250R1}$ are the solutions computed on the original and refined meshes and $\tilde{f}_{REF}$ is the solution computed on the reference mesh. X_{L250} and X_{L250R1} denote the spatially varying mesh sizes throughout the computational domain.

The numerical error normalized by the reference solution for the L250 and L250R1 M_2 amplitude elevation is presented in Figure 3.17c,d and compared against the total error that was calculated from the REF solution (Figure 3.17a,b). There is a similarity in the numerical and total error estimates particularly in the NA subdomain where the magnitude of both errors are 3-5% for the L250 mesh, and 1-2% for the L250R1 mesh. However, the estimate of the greatest magnitude numerical error is co-located with the periphery of the Georges Bank near sharp seabed topographic gradients, while the total error is spread across the entire Georges Bank. In general, a weaker reduction in the total error is observed compared to the numerical error. In particular, the total error is not reduced over the Georges Bank or along most of the SAB and MAB coastline (Figure 3.17a-b). However, the numerical error is reduced almost everywhere. For instance, the refinement of L250 to L250R1 reduces the numerical error estimate in the Chesapeake Bay estuary in the MAB region markedly. However, the error from the REF solution does not diminish in the MAB region (particularly the Cheaspeake Bay), which suggests these regions are more responsive to changes in

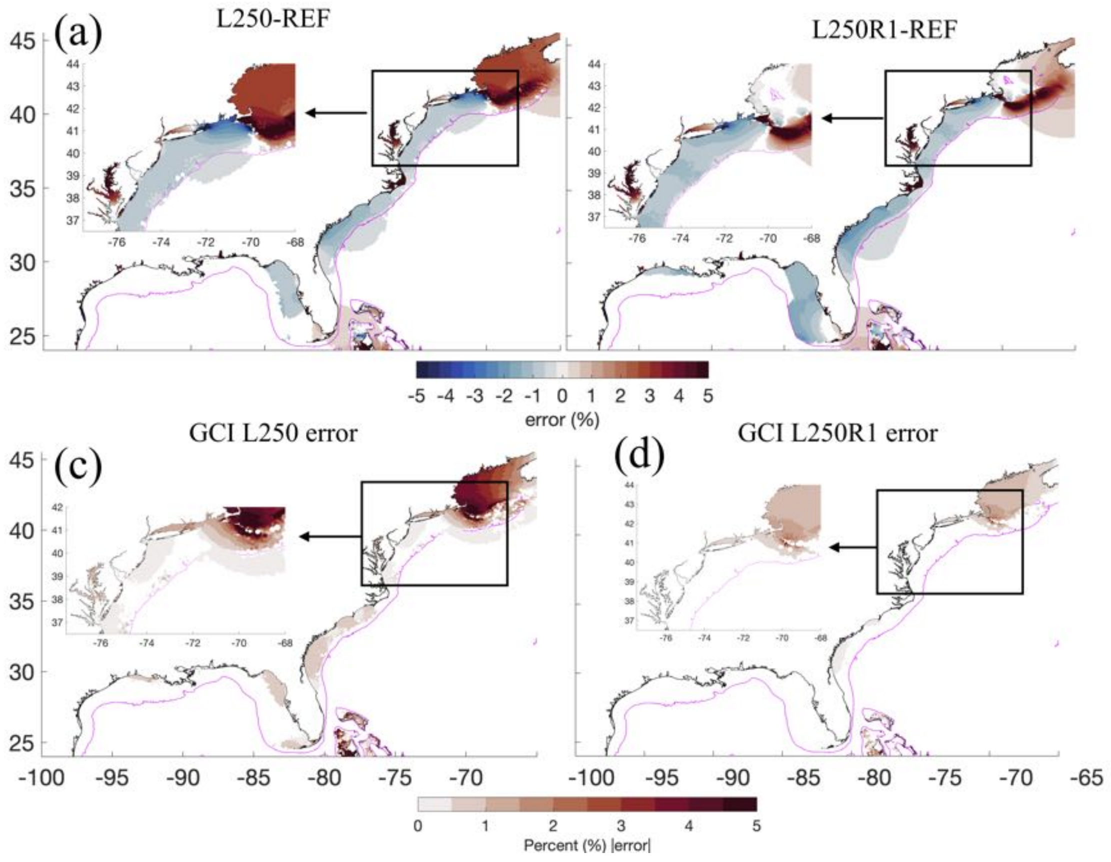


Figure 3.17. An estimate of the numerical error calculated via Richardson extrapolation following [128] obtained by refining the L250 mesh using a four-to-one refinement strategy to preserve the approximate problem.

the physical domain approximation (Figure 3.17). Overall, even though the numerical error has become insignificant (1-2% in magnitude) and converged as the mesh has been refined, relatively large physical domain approximation errors still remain in the Cheaspeake Bay, the Long Island sound, and the Georges Bank (≈1-5%). Thus, a method that will reduce the numerical error through an iterative refinement strategy like LTEA may be incapable of improving the accuracy of the solution as compared to observations even if it minimizes the numerical truncation error.

3.4.5.3 Mesh design combinations

The previously described mesh size functions (Table 3.1) can be used in combination by taking the minimum of each individual function for each point in a regional or global domain [41, 131]. Certain combinations of mesh size functions can be regarded as more or less efficient at sufficiently approximating the physical domain. For instance, if the user were to rely on a low mesh size gradation (e.g., 10-15%), the *estuarine channel* mesh size function becomes far less useful because elements in proximity to the channel are already close to the resolution at the shoreline.

Based on our resolution targeting approach, a sequence of mesh designs with different combinations of mesh size functions, all with a high gradation (35%), were built with the goal of maintaining the accuracy of tidal solution while significantly reducing the vertex count as compared to the REF mesh:

COMBO1: L50+G35+S20 $\rightarrow$ employs 50-m resolution everywhere along the shoreline (L50), a steep mesh size gradation of 35% (G35), and enhanced resolution on seabed gradients (S20). A total of 2.3 million vertices.

COMBO2: FS2+G35+S20 $\rightarrow$ uses *feature size* function to vary mesh resolution between 50 m and 250 m along the shoreline while maintaining a minimum of two elements across the width of the shoreline (FS2), a steep mesh size gradation of 35% (G35), and enhanced resolution on seabed gradients (S20). A total of 1.1 million vertices.

COMBO3: FS2+G35+S20+C0.5 $\rightarrow$ uses *feature size* function to vary mesh resolution between 50 m and 250 m along the shoreline while maintaining a minimum of two elements across the width of the shoreline (FS2), a steep mesh size gradation of 35% (G35), enhanced resolution on seabed gradients (S20), and enhanced resolution along estuarine channel features. A total of 1.3 million vertices.

The idea behind the choice of mesh combinations (COMBOx) is to proceed from a more simple design and move towards a more complex design to test the additive effects, i.e., start with uniform shoreline resolution (COMBO1); use variable shoreline resolution (COMBO2); add additional resolution along estuarine channels (COMBO3). COMBO1 begins with a high gradation rate and a large slope function parameter because of the resolution targeting philosophy that we believe, and which the experimental results support, lead to more efficient mesh designs. Figure 3.18 highlighting this targeting approach by illustrating the resolution distribution for the COMBO3 mesh.

Similar to the error patterns in Experiment 4 using 15% gradation (c.f., Figure 3.11), the RE in M_2 for all COMBOx meshes is reduced significantly from the G35 mesh, primarily in the NA and MA subdomains (Figure 3.19a-c). Conspicuous positive values of RE near the Georges Bank in proximity to the M_2's amphidromic point persists, but this is reduced from 10-21% for the G35 mesh to under 5% for all COMBOx meshes. The improvement to M_2 RE for the COMBOx meshes is also reflected in their CAFE curves (Figure 3.19d), which perform similarly to the S20 mesh in 99% of the comparison zone for the negative crossing (-1% to -2% RE), but contain slightly larger RE for the positive crossing (+3% to +4% RE). Overall, the RE is substantially reduced from the +16% RE positive crossing for the G35 mesh. Furthermore, the resulting pattern of errors against observations (Figure 3.20) for

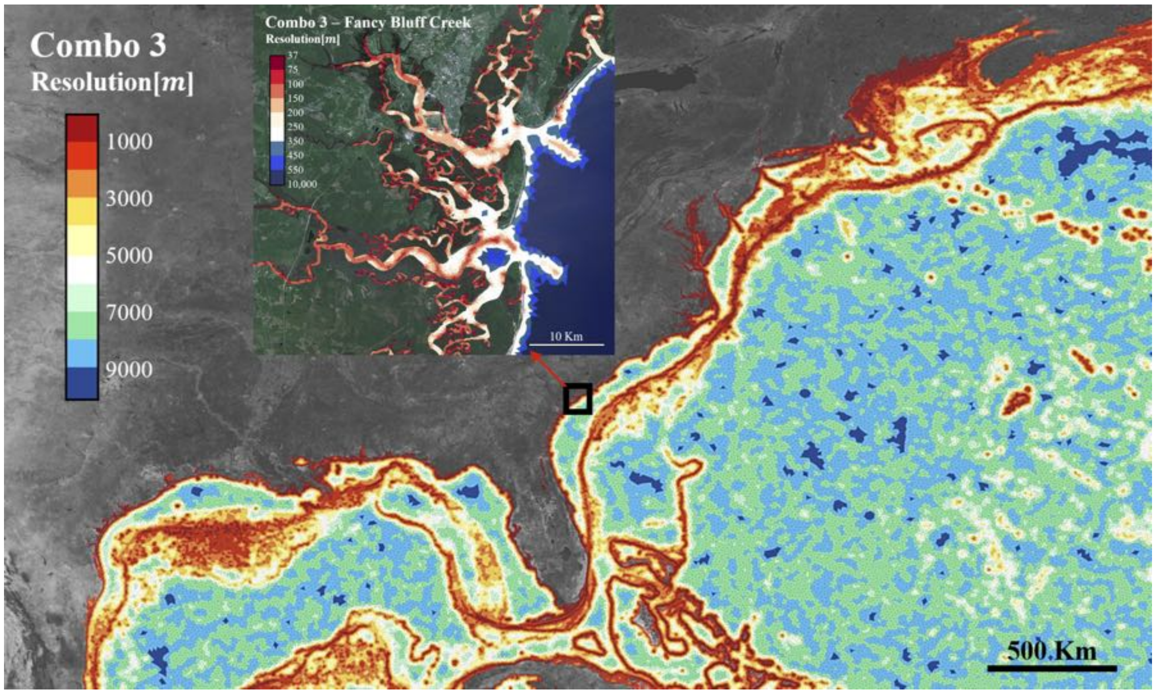


Figure 3.18. Elemental resolution distribution in the COMBO3 mesh, highlighting how fine resolution is targeted in narrow geometries and along seabed gradients and estaurine channels (see inset in Fancy Bluff Creek).

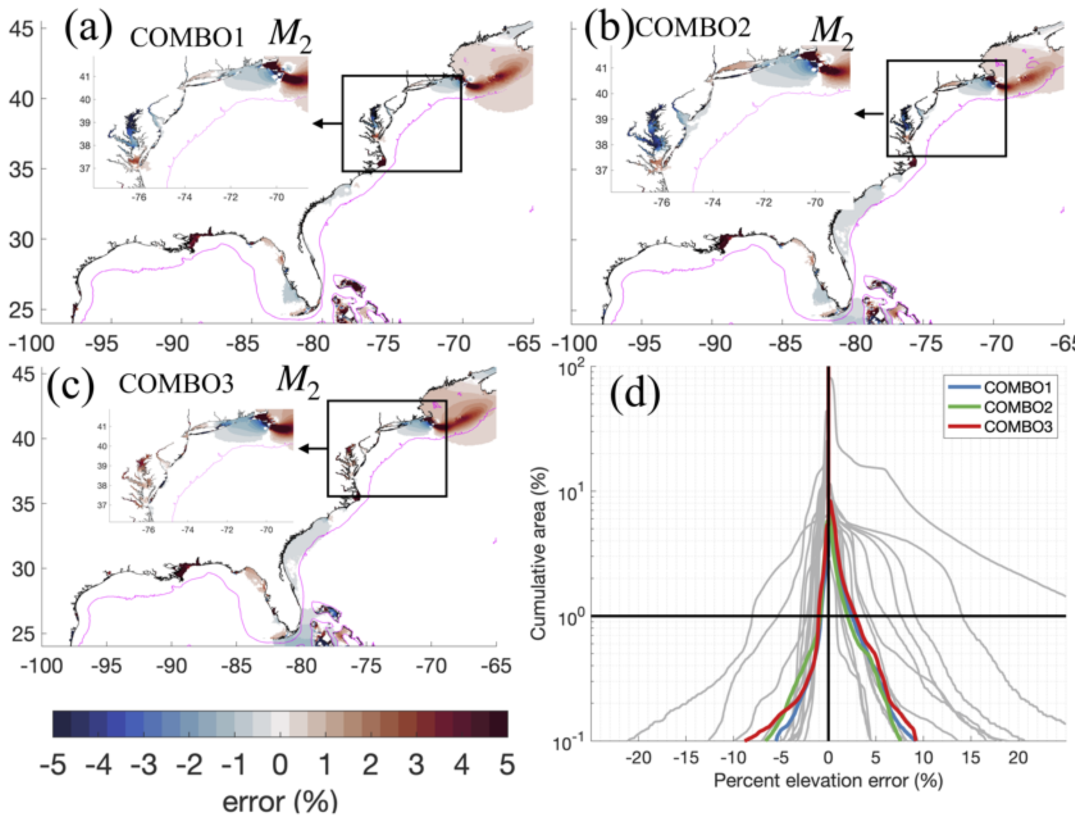


Figure 3.19. Panels (a)-(c) depict the error in the M_2 elevation amplitude solution that was computed on the COMBOx meshes. Panel (d) illustrates a CAFE plot of the error in the comparison zone for the three COMBOx meshes.

the COMBOx meshes approaches that of the REF mesh ($B = 0.01$ to 0.04, $\gamma^2 = 0.03$ to 0.05). In comparison, the positive bias and spread of the errors is significantly greater for the G35 mesh ($B = 0.08$, $\gamma^2 = 0.33$).

The effect on M_2 RE when moving from a uniform shoreline resolution (COMBO1) to variable shoreline resolution (COMBO2) based on the feature size approach in the combination meshes is small (Figure 3.19a-b). Differences less than 1% are noticeable in the Long Island Sound, Delaware estuary, and around the Georges Bank and Gulf of Maine. Furthermore, the resulting pattern of errors against observations from

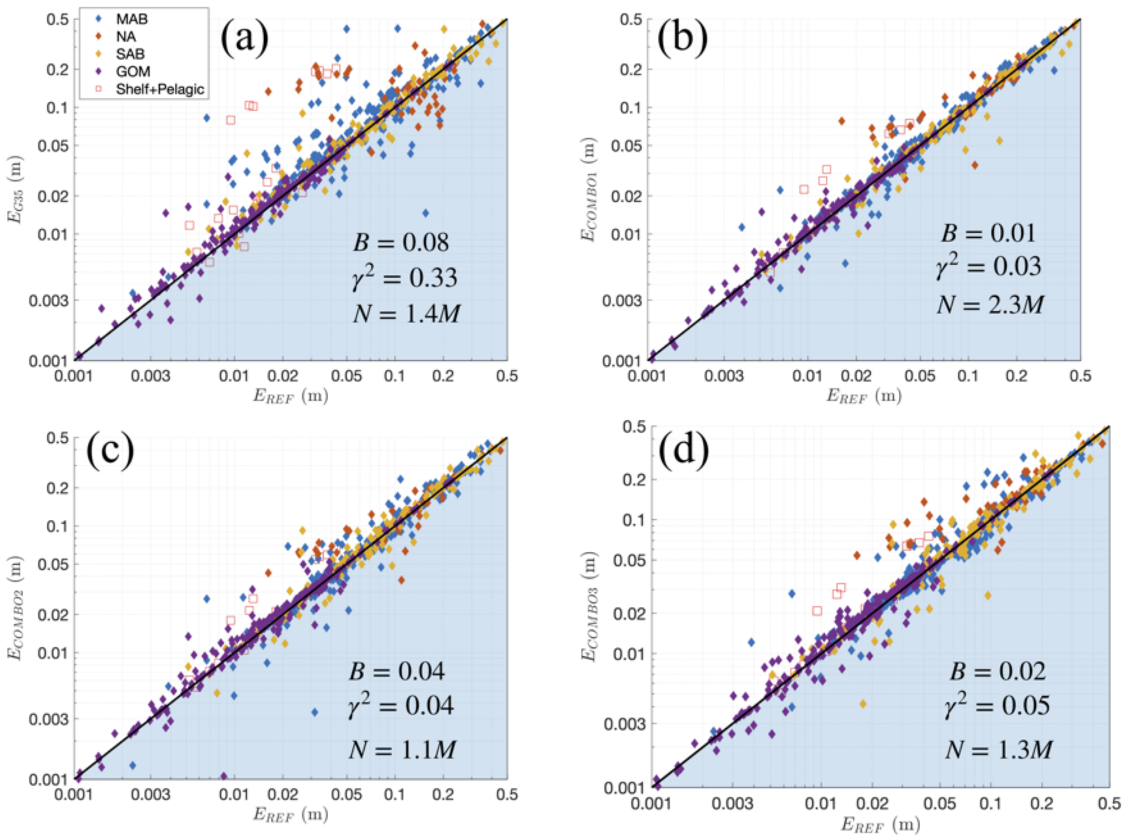


Figure 3.20. A comparison of the tidal constituent root-mean-square-error (E) for the M_2 tidal elevations at 439 tidal gauge observations (c.f., Section 3.3.1.2) between a solution computed on the REF mesh (x-axis) and the COMBOx meshes (y-axis). The normalized bias (B) and spread (γ^2) error metrics and the total vertex (N) are indicated. Points that fall in the blue shaded region have a smaller value of E than the REF solution.

REF is similar between COMBO1 and COMBO2, although the bias has increased to from 0.01 to 0.04 (Figure 3.20b-c). Considering that the usage of the FSx shoreline resolution in COMBO2 leads to 53% fewer vertices than in COMBO1, a small increase to the bias and variance is expected.

The effect on M_2 RE when additional resolution is placed along important estuarine channels (COMBO3 versus COMBO2) can be important in localized regions. The overall picture, as illustrated through the CAFE curves (Figure 3.19d) and the domain-wide tide gauge error pattern (Figure 3.20), is relatively unaffected, as evidence by the relatively small change in measured statistics. Predominately, the region of positive RE over the Georges Bank and the Gulf of Maine is increased by approximately 1% when moving to the COMBO2 and COMBO3 meshes. However, RE is noticeably reduced in the Delaware Bay, Chesapeake Bay, and Long Island Sound to under +1% RE in most areas (Figure 3.19b-c). Focusing only on the tide gauges ($n = 108$) contained inside the MAB estuaries (Figure. 3.21), the effect of targeting finer resolution along the channels is further highlighted. The normalized bias is reduced from a positive bias in COMBO2 (B = 0.03) to a negative bias for COMBO3 (B = -0.02) inside both estuaries, indicating that COMBO3 is slightly more accurate than the REF mesh here. The normalized spread of the errors γ^2 also reduced but only marginally.

3.5 Discussion and Conclusions

A series of controlled unstructured mesh resolution experiments were conducted over a large area of ocean in high-resolution ($\approx$50 m at the coast) and with a realistic shoreline boundary has been achieved through an automatic mesh generation approach facilitated by the *OceanMesh2D* software [131]. The sensitivity of the barotropic tidal response to unstructured mesh resolution was investigated by controlling the distribution of mesh sizes according to functions of *a priori* seabed and

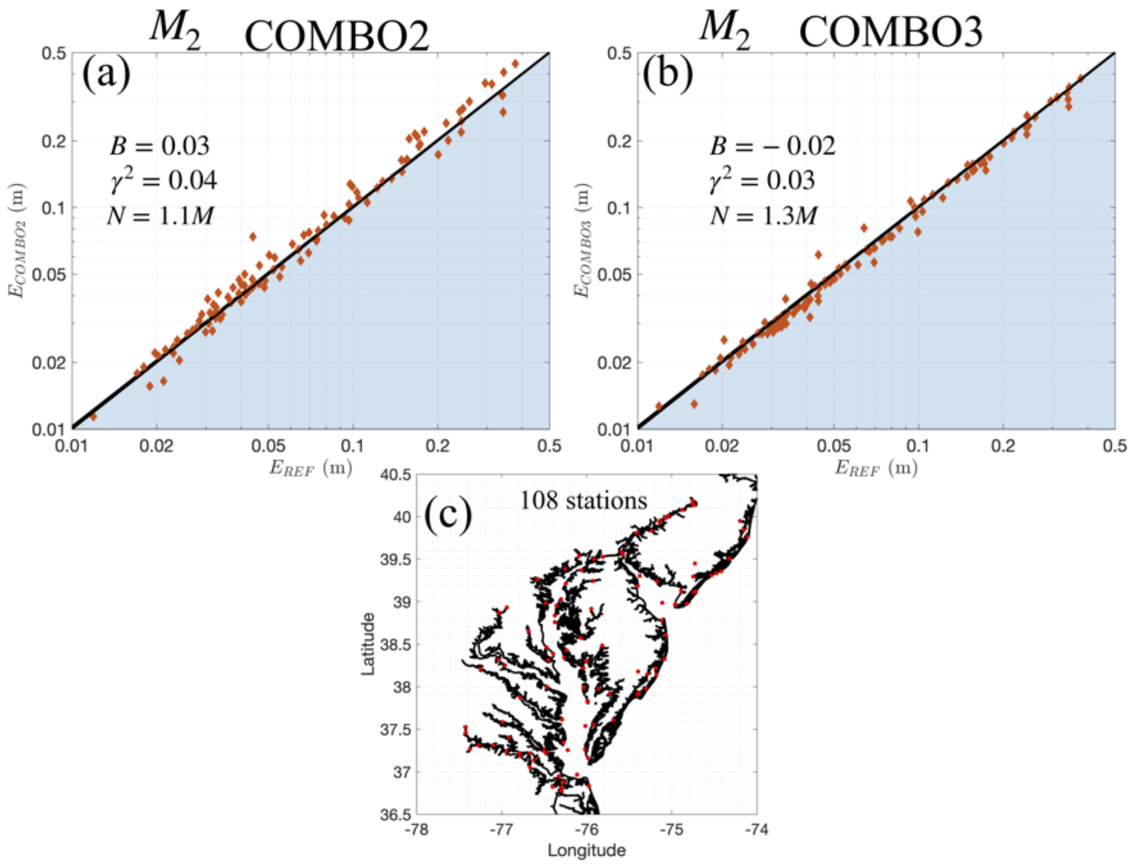


Figure 3.21. The complex root-mean square error (E) for the solution computed on, (a) the COMBO2 mesh, and (b) the COMBO3 mesh (includes enhanced resolution along estuarine channels), at 108 tide gauges in the Chesapeake Bay and Delaware Bay estuaries that are illustrated in panel (c). Various error metrics are indicated in the panels (a) and (b).

shoreline geometry information. It is noteworthy to mention that the whole process was scripted and thus automatic using the mesh generator suite. All meshes were numerically stable with a time step of 2 s without requiring post-processing hand-edits (vertex re-location, element re-shaping, or bathymetric smoothing), or *ad hoc* limiters (e.g., `https://wiki.adcirc.org/wiki/Fort.13_file#Elemental_Slope_Limiter`) and dissipation attributes.

The results from Experiments 1 through 5, and our examination of the effect of numerical error versus physical domain approximation error allow us to attempt an answer to the question we posed in Section 4.2: "a) How does the simulation of barotropic tides respond to the representation of shoreline geometry and seabed topography in the ECGC region? What are the sources of error and how do these contribute to the measured differences?".

In coastal ocean modeling applications, the shoreline resolution determines the predominate computational expense of the model. We explored ways to quantify the effect of simplifying the shoreline's representation in the mesh in Experiment 1) by coarsening the resolution from 50 m to 250 m and (Experiment 2) automatically varying resolution according to the width of shoreline features (*feature size* function). Coarsening the minimum resolution (Lx meshes) noticeably decreased the total area of the mesh by decimating fine scale shoreline features like embayments, headlands, and coves leading to a reduction in the total number of vertices up to a factor of five. However, the associated variation in the tidal elevation amplitudes over most of the domain was comparatively small, the relative errors against the REF solution in 99% of the domain did not vary by more than 2%, although noticeable differences are noticeable in the tail of the CAFE plots corresponding to highly localized regions. Experiment 2 demonstrated that the feature size approach FSx preserved the area enclosed by the shoreline of the mesh using the 50-m uniform shoreline resolution (see L50) while requiring approximately half the number of vertices. Further, the relative errors from the REF solution for FS2 showed a significant improvement over L250 in the tail, comparable to L50.

An important point is that the constraints from the sizing functions interact. For example, the increase in feature size parameter from 2 to 8 improves the representation of nearshore seabed topography (c.f. Figure 3.7) by using finer resolution across the width of the shoreline feature, but the higher feature size parameter does not improve the ability to resolve the complexity of the shoreline (c.f. Figure 3.4). Thus, our recommendation is that meshes intended for high-resolution tidal modeling to be constructed with a *feature size* approach (also see Conroy et al. [41]) with maximally two or three vertices across the shoreline's width instead of applying a minimum resolution uniformly along the shoreline [29, 85]. Note that in the feature size approach, a consideration should be taken to make sure that the element sizes along the shoreline cannot become too coarse. In this work, we applied a five-to-one ratio upper bound so that the element sizes do not exceed 250 m given that the length scales of the physical processes are still controlled by the proximity to fine scale shoreline geometry here, and coarse element sizes nearshore may not be conducive to accurately model other coastal processes that were not considered in this study such as wave setup induced through wave breaking [80].

Experiment 3 demonstrated how increasing the gradation rate can negatively impact the approximation of seabed topography in the mesh and the simulated accuracy of tidal solutions were highly degraded. The mesh with the highest gradation (G35) was the worst performing mesh in terms of the M_2 and K_1 relative error values out of all 15 meshes in the five experiments. The effect of increasing the gradation is likely to have increased the numerical error ([71]) in addition to the physical domain approximation error making the determination of the root cause of the poor performance challenging. However, experiment 4 clearly demonstrated that placing resolution along seabed gradients ($\sim$1 km along the continental shelf break and slope) improved the accuracy of tidal solutions, which is in agreement with prior works [37, 98, 153]. At the same time, increasing the gradation rate coarsened the representation of the continental shelf break as resolution sizes would grow faster from the shoreline. Thus, it is likely that our application of resolution along seabed gradients reduces the numerical error as large gradients in the solution are co-located with steep seabed topographic gradients [e.g., 72, 74]. Our recommendation is the use of a high value for the slope mesh size function (S10-S20) in combination with a high gradation rate (G35) to offset the increased gradation's rate negative impacts on both error sources, while largely reducing the total number of vertices in the mesh.

Experiment 5 demonstrated that the approximation of the seabed topography across some estuaries in the Mid-Atlantic region (Cheaspeake Bay and Delaware Bay) could be improved by using the *estuarine channel* mesh size function to place targeted high-resolution zones along submarine channels inside and leading into estuaries. In estuaries that are characterized by well-defined submarine channels that occupy non-trivial portions of the width of the estuary, it is important to ensure that adequate resolution is placed along these channels so that the total cross-sectional area and local ocean depth minima are preserved. Indeed, the application of progressively placing finer mesh resolution along the estuarine channel network (extracted using an upslope area computation on the DEM) was shown to reduce tidal error metrics as compared to both the reference solution and measured data. We remark that other mesh size heuristics, such as the slope mesh size function and using finer resolution along the shoreline with a low gradation rate can implicitly, but inefficiently, capture these submarine channel features. Thus, the application of the *estuarine channel* mesh size function motivates the usage of a higher mesh size gradation so as to focus resolution only on the submarine channels allowing us to efficiently discretize the estuarine environment.

Considering the variations in the total vertex count and the tidal solution errors throughout Experiments 1 to 5, an attempt was made at answering “b) recommending a set of mesh size functions that place resolution according to shoreline geometry and seabed topography to efficiently discretize coastal ocean domains that approximately reproduce simulation results from an extremely well-resolved mesh” (from Section 4.2). We tested the performance of mesh design strategies that involved using a steep mesh size gradation rate (G35) in combination with the targeted mesh sizing functions along the shoreline (FSx), sharp topographic gradients (Sx), and estuarine channel systems (Cx). Three combination meshes (COMBOx) that ranged from 1.1 million to 2.3 million vertices were generated. Overall, all COMBOx meshes performed similarly to the REF mesh both directly and as compared to measured tide gauge data. The additive effects of multiple mesh size functions reduced the error metrics largely, especially in the comparison to the G35 solution, which had a noticeably degraded solution without the usage of other sizing functions (in particular the *slope* function) used in the COMBOx sequence.

Echoing our findings from Experiment 1, the COMBO2 mesh utilized a small value of the *feature size* function parameter (FS2) and had approximately half the vertex count of COMBO1 (uniform shoreline resolution) with little increase in relative error, thus the FS2 is considered an efficient mesh design choice. However, estuarine channels are more likely to poorly represented with the high gradation (G35) and FSx design combination as mesh sizes will become coarser in certain regions depending on the cuspate shape of the shoreline. Our conclusion is the 15% increase in the total vertex count associated with the addition of the C0.5 component of COMBO3 to better capture estuarine channels, can be considered a good investment particularly since the solution in nearshore estuaries of high importance is improved; even to a point beyond the performance of the REF mesh (e.g., Figure 3.21). Our results imply that the 250-m bounded blanket resolution applied across the large estuaries in reference solution is coarser and less effective than the targeted resolution that follows the channelized seabed in the C0.5 mesh size function. In fact, a key drawback of mesh designs that apply uniformly fine zones of resolution throughout regions of similar ocean depths (the wavelength-to-gridscale heuristic [e.g., 152] is that there is less flexibility to more finely capture targeted seabed features and shoreline constrictions due to the baseline expense of the model. In many regions, the application of targeted refinement can produce more finely resolved solutions in localized areas of importance with far fewer vertices.

Through the combination of the constraints imposed by a set of mesh size functions (COMBOx meshes), the vertex count was reduced by nearly an order of magntiude from the reference mesh and had a converged solution with tidal error metrics in 99% of the East and Gulf Coast waters ranging from -2% to +1%. For instance COMBO3 (1.3 million vertices) had eight times fewer vertices as reference (10.8 million vertices). These results suggest that pre-existing operational models may be largely inefficient, over-discretizing in some areas and under-discretizing in others as pre-existing models use nearly uniform resolution nearshore following the wavelength-to-gridscale sizing heuristic. For example, the Hurricane Surge Operational Forecasting system (HSOFS) mesh [146] used in real-time predictions employs a minimum shoreline resolution of 250 m and contains 0.75 million underwater vertices, which is similar in number to our L250 mesh. In contrast, the COMBO3 mesh, which spans the same ECGC study region, utilizes up to five times finer resolution nearshore (50 m compared to 250 m) and up to ten times finer resolution along the continental slope (1 km compared to 10 km), with only 1.6 times the total number of underwater vertices than HSOFS.

We highlight that an important first step in the coastal model development procedure is to construct a mesh that minimizes the physical domain approximation error before model tuning occurs vis-a-vis varying bottom friction, other dissipative coefficients, viscous models, and manually altering ocean depths and shoreline form.

As was evident in this paper, by improving the accuracy of the approximate problem (i.e., the representation of the shoreline and seabed topography as per the available geospatial data used), the tidal solutions exhibited convergence towards a reference solution. The primary variation in the M_2 (c.f., Figure 3.16) tended to coincide with zones of the ECGC in which the bottom friction coefficient are typically modified [143]. For instance, since the Chesapeake Bay has a muddy seabed floor, the friction coefficient, C_f is often set low a value ($C_f \approx 0.001$) and this is found to improve comparisons with tidal harmonics [61]. However, our results indicate that the the M_2 tide in the Chesapeake estuary is largely sensitive to mesh design with changes on the order of 10% between the mesh design variations explored here (c.f., Figure 3.16). It is thus likely that the bottom friction application procedure may be tuned incorrectly depending on the local mesh design; for instance, depending on the complexity of the estuarine network. In this estuary, the representation of the axis-aligned center channel was largely related to an underprediction error in the M_2 tidal constituent.

Hagen et al. [72, 73] explored the distribution of mesh resolution for 1D models of the continental shelf and 2D tidal models through the LTEA method with the goal to reduce the overall vertex count of a tidal model using *a posteriori* truncation error indicators. A limitation of the LTEA approach is that it does not consider the contributing effect on the physical domain approximation error that we analyzed in this study, and relies on an existing mesh to begin with. Our feature-driven *a priori* approach allows for meshes to be generated without the need to generate a finely resolved mesh and iteratively reduce the vertex count through simulations of the tidal solution. Moreover the feature-driven approach is more generalizable to nontidal geophysical flows (e.g., tsunamis, surge, sub-tidal dynamics and wind waves). Nevertheless, an interesting future direction of this work would be to take a mesh that produces a low approximate error (e.g., COMBO3) and apply the LTEA method to further reduce the vertex count and/or reduce the numerical truncation error. This could be particularly useful to better understand the source of the error around the Georges Bank and the Gulf of Maine, NA near the M_2 amphidromic point, which conspicuously appears in nearly every mesh design.

CHAPTER 4

DYNAMIC LOAD BALANCING FOR PREDICTIONS OF STORM SURGE AND COASTAL FLOODING

4.1 Overview

This chapter improves upon the efficiency of simulating coastal flooding on unstructured meshes using a dynamic load balancing technique. Computational models can accurately predict the flooding of coastal regions due to tropical cyclones and other storms, but they have the potential for workload imbalances. Large floodplains are allowed to wet and dry during the storm, and thus the workload for wet regions may not be distributed evenly over the computational resources. In existing models for real-time prediction of and long-term design for coastal flooding mitigation, their parallel implementations are based on a static paradigm in which a computational mesh is partitioned into sub-domains at the beginning of the simulation. We demonstrate that the static paradigm can be sub-optimal in detailed coastal flooding calculations that involve large, normally-dry, low-lying areas that could be inundated during a storm event. In this case, many processors are not fully utilized from a computational perspective as many degrees-of-freedom remain in a dry-state with a zero-valued solution. We integrate a capability to dynamically rebalance computational work into a state-of-the-art, shallow-water circulation model to reduce the parallel execution times associated with calculations of extensive flooding driven by tidal and meteorological forcings. Our development is based on dynamically partitioning the decomposition of mesh during run time so that the that the computational load is determined by the degrees of freedom in wetted areas. The implementation has a low overhead cost, and we demonstrate the flooding simulations can achieve speed-ups 8-45% over the static case. This chapter will be submitted in a modified form to the journal *Computers and Geosciences*.

4.2 Introduction

In horizontal two-dimensional (2D), unstructured-mesh, finite-element modeling of wind-and tidally-driven coastal circulations, a portion of the computational domain is included above the local mean sea level (LMSL) state to simulate coastal flooding. The computations use an unstructured mesh composed of triangular elements with highly-variable sizes to represent the large horizontal scale separation (i.e., $\mathcal{O}$(10 m)-$\mathcal{O}$(100 km)) in processes that occur in coastal flooding situations. Predictions with these unstructured meshes are used to provide crucial information for coastal hazard assessment and design [CSTORM Project; 38] and emergency management operations [21, 22, 53, 140].

A widely-used model for hurricane-related, coastal flooding predictions is the ADvanced CIRCulation hydrodynamic model [ADCIRC; 96], which utilizes the finite-element method to solve the shallow-water equations on an unstructured mesh with triangular elements. The coastal ocean is discretized with millions of elements with varying resolution to represent the development and propagation of waves, tides, and surge from the open ocean to the nearshore and over complex coastal topography (Figure 4.1). In the region of focus, the mesh often contains an extensive floodplain extending to an elevation of 10-m to 15-m above mean sea level. On the floodplain, high-resolution elements range in size from 10-m to 250-m and enable an accurate representation of the fine horizontal length scales and complex land cover variability that control coastal inundation patterns [e.g., 50, 60, 78, 134]. Thus, many degrees-of-freedom (DoFs) in the modeling system are located overland. In the Hur-

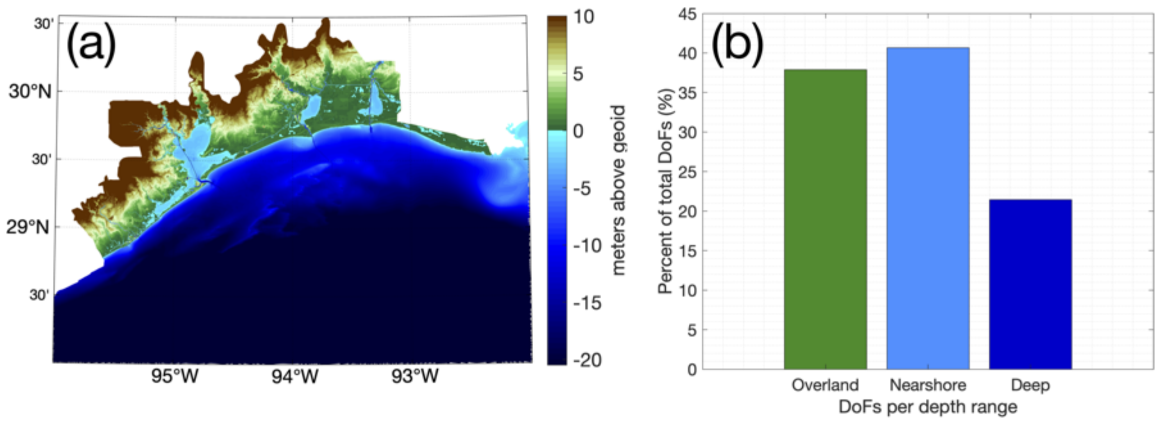


Figure 4.1. Panel (a) is an illustration of a high-resolution finite element mesh in which the bathymetry/topography is shaded that is used in the computation of coastal flooding. In panel (b), a histogram illustrates the number of degrees-of-freedom (DoF) per bathymetric depth range below sea level for (a). For the purpose of this figure, the overland category is classified as greater than 0.10 m above sea level, nearshore is less than or equal to 0.10 m and greater than -50 m, and deep is less than or equal to -50 m.

ricane Surge Operational Forecast System (HSOFS) mesh [146] used for real-time predictions (`https://coastalrisk.live/`; [59]), approximately 55% percent of the vertices are above the mean sea level state at initialization of the simulation. The transient and episodic nature of hurricane flooding further imply that, although the overland regions are included due to uncertainty in the magnitude and location of potential storms, the majority of overland DoFs will never be flooded during any given simulation.

The computationally expensive aspect of modeling coastal flows over land have motivated the development of many numerical schemes and approaches [6, 16, 33, 35, 93, 144]. One approach is to start the calculation with an initially coarse mesh and then adaptively refine the mesh as the event approaches and propagates throughout the domain. Adaptive mesh refinement methods perform well for the simulation of transoceanic tsunamis and the associated coastal flooding, in which the background state is quiescent before the event occurs [16, 93]. However, for coastal circulation models that are concerned with total water level predictions from hurricanes and other storms, the background tidal signal continuously interacts with the irregular geometry that characterizes the shoreline boundary and complicates mesh refinement strategies. For example, the representation of a narrow waterway that conveys tidal flow will be lost at coarse levels of refinement distorting the nearshore circulation patterns.

Multi-scale approaches [e.g., 144] also have been shown to reduce wall-clock times for detailed coastal flooding simulations. However, these approaches require an extensive development of meshes, each using various levels of horizontal resolutions at different cartographic scales and then coupling paradigms to merge the solutions. Methods such as sub-grid scale modeling techniques [e.g., 32–34] are also capable of cost-effectively resolving the floodplain by modeling wetting/drying through a porosity function. However, sub-grid scale methods have not been applied yet to the widely-used unstructured finite-element and -volume software suites that can already seamlessly produce coastal ocean circulation predictions across scales [e.g., 35, 62, 96, 160]

Considering the challenges and recognizing the previous efforts in unstructured mesh development for coastal flooding studies during the past two decades, an approach is designed to reduce the computational cost of modeling the floodplain with high-resolution, unstructured elements using the ADCIRC solver. Our approach does not require any alteration to pre-existing meshes or the numerical schemes used in the solver and keeps static the connectivity of the mesh, which altogether minimizes its invasiveness. The qualities of our approach ensure that pre-existing solutions can still be reproduced, but with a gain in computational efficiency and with a minimal effort by the user. The approach dynamically redistributes the computational workload between processors in a distributed computing environment using message passing to reflect the time-varying movement of the wet/dry boundary during a coastal flood. For the approach to result in a net speedup of the simulation, (1) the re-balancing of work must be fast relative to the work savings; and (2) there must exist a sufficient number of dry-state DoFs to generate a work savings. The former consideration creates a significant software engineering problem to implement capabilities in pre-existing static solvers, such as ADCIRC, because their timely execution is vital to the success of the approach.

To facilitate dynamic load balancing in an application like the ADCIRC solver, the software design must support the ability to efficiently relocate data and their associated dependencies during runtime. Software toolkits such as Zoltan [24], Charm++ [81], and PetSC [12] have been developed to be integrated into solvers with this purpose in mind. Often these packages are combinations of algorithms written in C and C++ and are employed in parallel unstructured and/or adaptive finite element computations. In this study, we rely on the Zoltan toolkit to provide essential functionality to the application.

The rest of this article is organized as follows: first, we describe our methodology to reduce the computational cost of the floodplain by integrating the Zoltan toolkit within ADCIRC and developing an algorithm to reduce the cost of the floodplain. Then we describe the effect of the load balancing application on the computational performance and behavior of ADCIRC when using an idealized case study. Finally, we apply the approach to a models with realistic topography and setup. The paper concludes with a discussion on the key findings.

4.3 Methods

4.3.1 ADCIRC Hydrodynamic model

The ADCIRC model solves the non-linear Shallow Water Equations (SWE) using the Generalized Wave Continuity Equation (GWCE; [86, 100]) in parallel using a Single Program Multiple Data (SPMD) approach with the message passing interface (MPI). The SWE are discretized by using a continuous Galerkin (CG) finite element (FEM) scheme in space and a finite difference scheme in time, and the method is formally second-order accurate [96]. For each simulation timestep, the GWCE is first solved to calculate the water-surface elevation (WSE) by either an implicit Jacobi-Conjugate gradient (JCG) method [70]) or an explicit mass-lumping method [145]. The WSE are then used to compute the elemental wetting/drying through the fulfillment of a set of logical conditions [97]. Finally, the depth-averaged momentum equations are solved semi-explicitly (with the exception of the Coriolis term) using a time-centered finite difference scheme.

ADCIRC represents the wetting and drying process on an elemental basis, in which elements must either be fully wet or fully dry [97]. A thin film of water of thickness *Hmin* is applied initially on all dried elements, and a set of criteria must be satisfied for an element that was previously dried to become wet. The criteria require that (i) the local WSE must exceed the WSE of the highest elevated vertex in the dry element by a minimum height of Hmin; and (ii) the WSE in the adjacent wet elements must be high enough to create a WSE gradient sufficient to overcome the resistance from bottom friction. For an element to remain classified as wetted, the local WSE must be a minimum height of *Hmin* above the WSE of all vertices in the element.

The wetting/drying process creates a moving free surface boundary in the computational domain that can move at most one element per timestep, otherwise the scheme will go numerically unstable [49]. The wet/dry boundary (W/D) can be located through the vertex-to-element connectivity by comparing the number of wet elements to which a vertex is connected, to the total number of elements to which the vertex is connected. If the number of connected wet elements is less than the total number of elements, then the vertex belongs to the wet/dry boundary. It should be noted that the algorithm used in ADCIRC performs well for advancing the wet/dry boundary landward; however, it often exhibits difficulty in the drying condition when the wet/dry front recedes, as small patches of elements that persistently remain wetted are created [103].

4.3.2 Load balancing strategy

Our strategy to reduce the cost of representing the floodplain with high resolution elements involves trading a computational workload balance for a memory imbalance. In this strategy, the majority of the dry-state DoFs become located on a subset of processors, which indirectly reduces the size of subdomains that own the majority of wet-state DoFs. When the dry-state DoFs are located on a subset of the processors, their computational work can be reduced in calculations through a rearrangement of data in memory that will be explained in the subsequent section.

The load balancing strategy divides the computational domain into an offline and online component in which the offline component is located overland and the online component represents the underwater portion of the domain (Figure 4.2(a)). At the start of the simulation, the wet/dry boundary is located near the local mean sea level state, which enables the majority of dry-state DoFs to be located in the offline portion and a work savings to be obtained through our approach (Figure 4.2(a)). At some point in the calculation, the computational workload may need to be rebalanced as the movement of the wet/dry boundary begins to reach the boundary of the offline-state. When the problem rebalances, the simulation becomes divided in time into a computational and rebalancing phase. The rebalancing phase involves computing a new data distribution between processors, migrating the data to reflect the new distribution, and restarting the calculation phase. The combination of a rebalancing phase with the subsequent calculation phase is termed an epoch. In the current implementation, the calculation and rebalancing phases do not overlap in simulation time.

The increase in computational load each epoch occurs in a stepwise fashion as the storm nears its landfall or reaches peak intensity. The storm-induced flooding increases the frequency of the rebalance events and at the same time shortens the duration of each epoch (Figure 4.2(a)). The transition from one epoch to the next is accomplished through a partitioning phase. In the case that all DoFs becoming wetted, the load and thus the speed of the parallel application become approximately equivalent to that of static ADCIRC. After the storm makes landfall, the wet/dry boundary recedes back to the mean sea level state, and the number of dry-state DoFs increases enabling the load balancing strategy to reduce the total number of DoFs and acclerate the calculation.

4.3.3 Mesh partitioning

For the assignment of mesh data (elements and vertices) to processors, the elements of the unstructured mesh are represented as an undirected graph. The triangular element's centroid is a node of the dual graph and it is assigned a weight proportional to its computational expense (Figure 4.3).

To partition the graph of the unstructured meshes used in the calculation of coastal flooding, the ParMETIS V4.0.3 program is used with the implementation of the K-way Multi-level graph decomposition algorithm [83]. ParMETIS's Part K-way algorithm is called via Zoltan's interface [24]. In our implementation, elements and their associated vertices belong to a set of k-partitions that are termed subdomains in

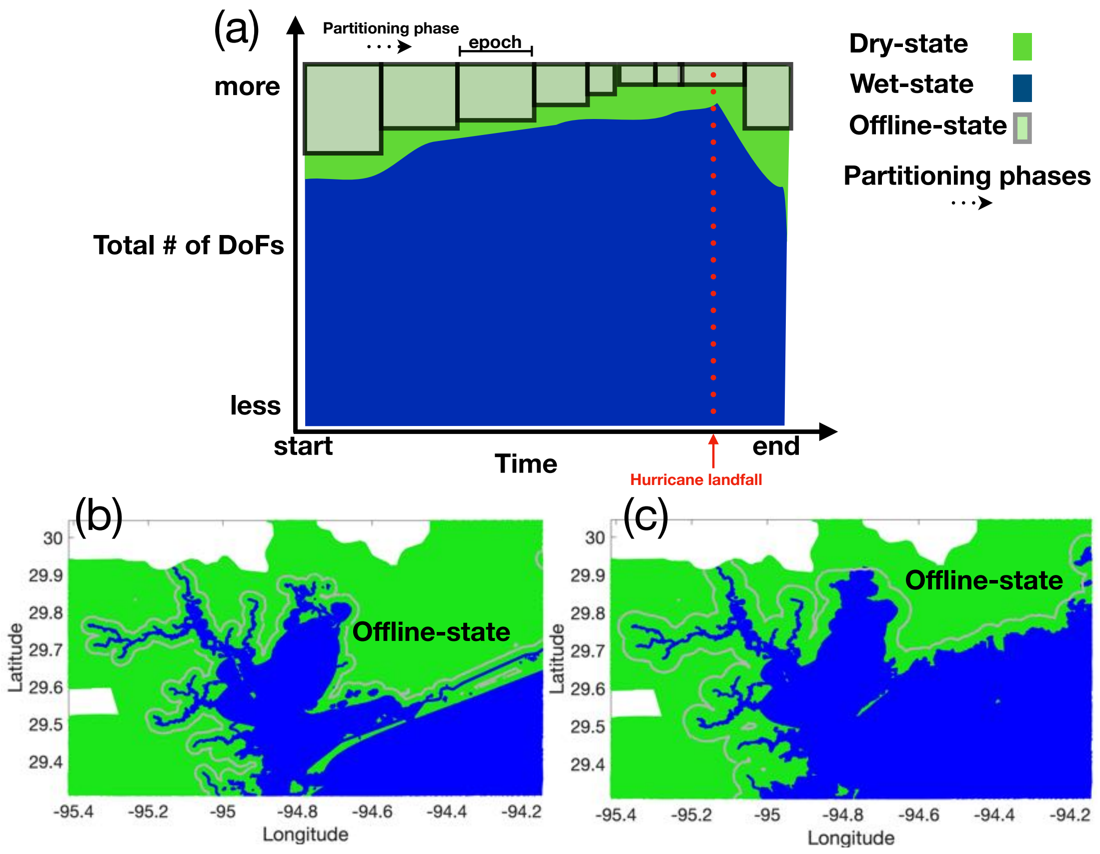


Figure 4.2. An illustration that depicts some concepts in our load balancing approach. In panel (a), the time evolution of the computational load is shown for a Hurricane-driven coastal flooding event. The shaded grey bars indicate epochs in which a fraction of the total number of DoFs are turned offline. In panels (b) and (c) the computational domain with blue denoting portions of the domain underwater and green overland is shown before the Hurricane (b) and after the Hurricane makes landfall (c).

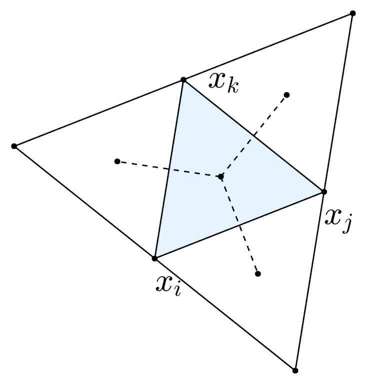


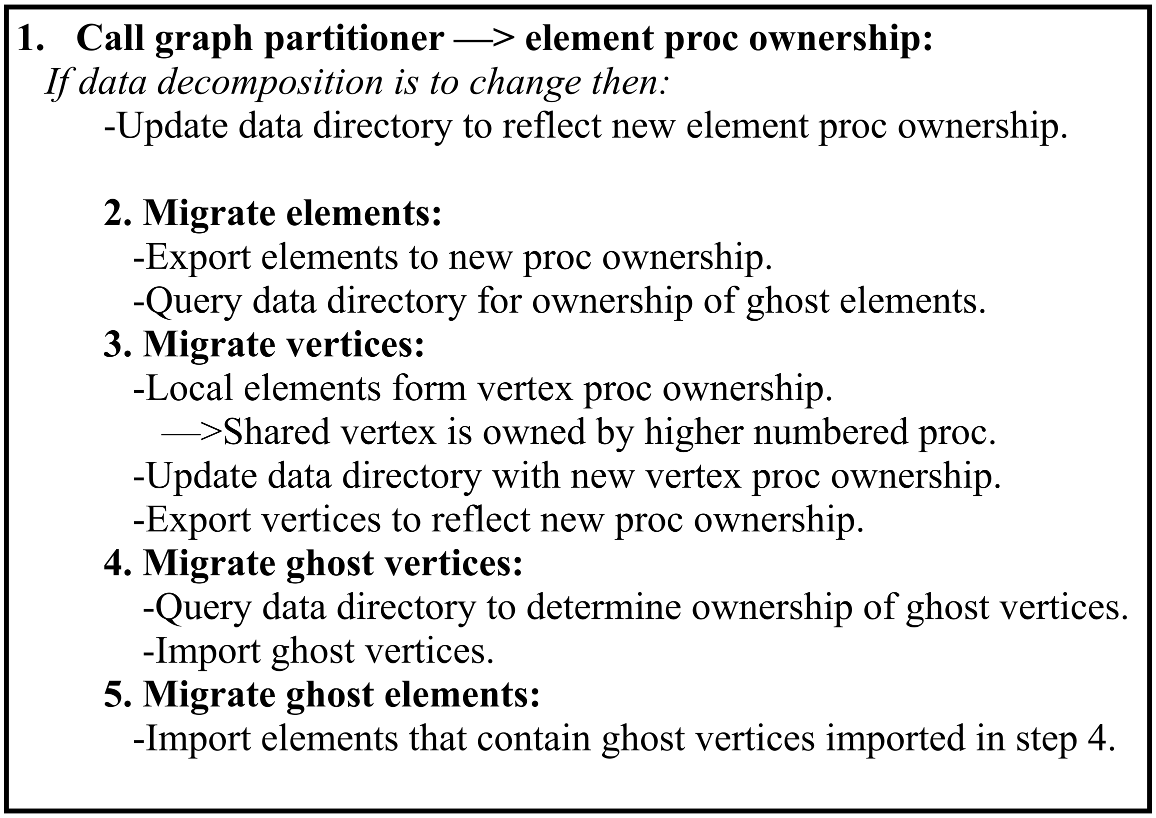
Figure 4.3. A triangular finite element. Triangle-based connectivity showing the vertex adjacency in solid black lines and the triangle adjacency otherwise referred to as the dual-graph in dotted-black lines.

which each k^{th} subdomain is owned by one processor. A vertex or element is a termed a border if one of its adjacent neighbors is on another subdomain. A border element or vertex is further classified as a ghost if its owned by more than one processor. If a subdomain contains at least one element or vertex that is also owned by anther processor, then those two subdomains are neighbors to each other. If a subdomain contains vertices that cannot be all be visited by traversing either the vertex-to-vertex or element-to-element connectivity breadth or depth first search manner from a given starting vertex or element located in the subdomain, the subdomains are termed discontiguous.

4.3.3.1 Parallel Mesh Partitioning Algorithm

An overview of the parallel mesh decomposition steps are outlined in Fig. 4.4. Each processor participates in the partitioning algorithm and operates on a component of the global mesh.

The algorithm begins by calling ParMETIS to divide the weighted graph of the mesh. Lists are generated that indicate which data (vertices and elements) need to migrated between processors to create the updated data decomposition. ParMETIS's



```
1.  Call graph partitioner —> element proc ownership:
  If data decomposition is to change then:
      -Update data directory to reflect new element proc ownership.

      2. Migrate elements:
        -Export elements to new proc ownership.
        -Query data directory for ownership of ghost elements.
      3. Migrate vertices:
        -Local elements form vertex proc ownership.
          —>Shared vertex is owned by higher numbered proc.
        -Update data directory with new vertex proc ownership.
        -Export vertices to reflect new proc ownership.
      4. Migrate ghost vertices:
        -Query data directory to determine ownership of ghost vertices.
        -Import ghost vertices.
      5. Migrate ghost elements:
        -Import elements that contain ghost vertices imported in step 4.
```

Figure 4.4. The parallel mesh partitioning algorithm. *proc* stands for processor.

scratch-remap scheme is utilized to decompose the mesh by partitioning the graph to balance the load but also to minimize migration overhead by remapping the data ownership to reflect the existing data owernship [88]. The scratch-remap type scheme to partition the graph is in contrast to diffusive-based approaches that diffuse overweight processors to their neighboring subddomains [e.g., 133]. For our problem's configuration in which a large amount of imbalance occurs in localized overland zones, we find the scratch-remap scheme can produce lower edge cuts and better minimize the load imbalance as compared to the multilevel diffusion type approach. However, the differences in timing between either diffiusion or scratch-remap schemes are marginal in comparison with the overhead spent performing the logic and data migration to achieve the updated decomposition. In other words, the majority of time spent rebalancing is not spent calling ParMETIS's graph decomposition algorithm and thus a preference is given to producing subdomains with low edge cuts which tend to occur moreso with the scratch-remap approach.

The elements are first migrated between processors so that each element of the mesh is owned by only a single processor. The migration of elements implicitly forms the vertex ownership among processors since the three vertices of each element must also be local to the processor in order to form a conforming mesh necessary for CG FEM. A single layer of shared vertices and elements on the border of the subdomains must also be created to enforce the necessary continuity requirements required by the CG FEM. At the end of step 2, only the higher-numbered neighboring partition owns the vertex on the border of a sudomain. In step 3, these shared vertices on the border of the subdomain are localized on the lower-numbered neighboring processor. Finally, the connected elements to any localized vertices in step 3 are imported to finish the formation of the halo ghost zones creating a single layer of halo ghost vertices and elements that border each subdomain (Figure 4.5(b)).

In order to facilitate the steps that involve data migration between processors, Zoltan's unstructured communication library is used [45]. A sequence of call-back functions that (1) determine the sizes of data to be migrated between processors, (2) pack the data-to-be-migrated, and (3) unpack the received data on their new processors are registered and execute in a procedural manner with each migration step (Fig. 4.4;[24]). A benefit of Zoltan's unstructured communication software is that it enables a local inter-processor communication pattern whereby each processor shares its information only with its neighboring subdomains (Figure 4.5(a)). The local communication pattern is efficient since messages are only sent to the neighboring processors, not every processor in the communicator, which would be the case with an all-to-all reduce operation. The local communication pattern is used in the partitioning phase to exchange the location of the moving wet/dry boundary, to decompose boundary conditions, and to form halo ghost zones.

Distributed data directory's are used to query the processor ownership of vertex and elemental data after each migration step has occurred. A distributed directory operates like two hash tables that are distributed across a number of processors [45]. The first hash table determines the processor that owns the data and the second hash table determines the location of data on the processor that owns the data.

The performance of the parallel partitioning algorithm is overall represented in the timing statistics that are later described. The general performance and time complexity of the parallel partitioning algorithm is challenging to assess given the large variety of potential problem configurations in the ADCIRC model (i.e. varying amounts of boundary conditions).

4.3.3.2 Elemental Weighting Scheme

A weighting scheme is applied to the elements of the mesh (dual graph) to reflect the disparity between the associated computational work with the dry-state and wet-state DoFs. To enable more control over what component of the domain is included or

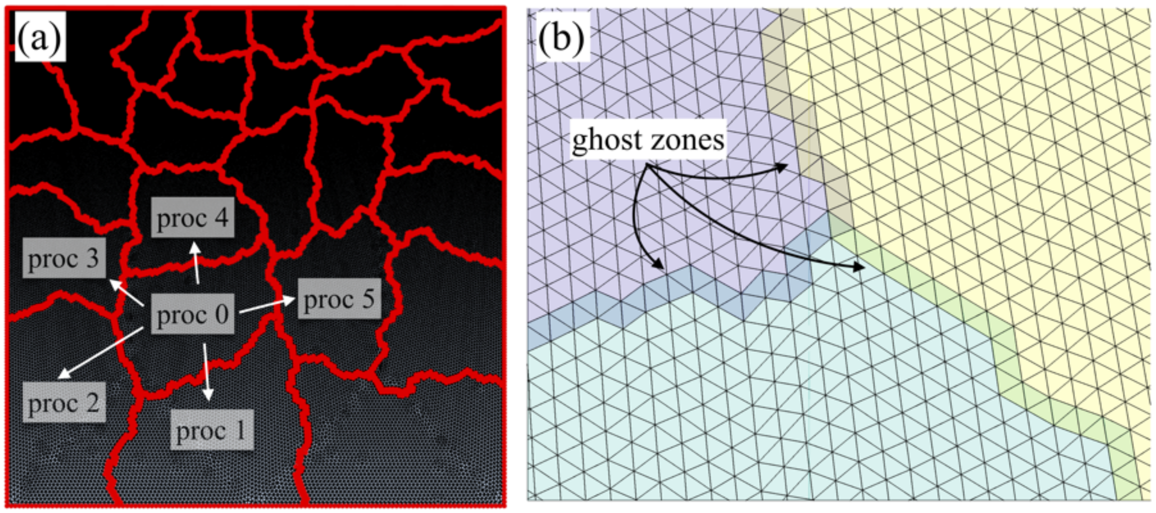


Figure 4.5. A local communication stencil is indicated in panel (a). The subdivision between each processor's data is demarcated in red. In panel (b) the ghost zones that are composed of duplicated elements and vertices on each processor's boundaries are shaded in a contrasting color.

excluded in the calculation, elements and vertices are given an offline or online status. The online and offline status is based on a set of user-defined criteria. Elements that have an offline status must be in a dry-state and are weighted with a value of 0, whereas online elements are weighted with a value of 1. To reflect the reduction in interprocessor communication associated with offline elements, dual graph edges that connect two offline elements (i.e., elements share an edge) are weighted with the value 1, whereas dual graph edges that connect either two online elements or one online element to an offline element are weighted with a relatively larger value (1,000 in our implementation). The large disparity in edge weights between the online and offline edges represents the near zero communication overhead incurred between dry-state and offline-state DoFs.

4.3.3.3 Loop Rearrangement

Data in memory is rearranged according to the cost of the dry-state DoFs (Figure 4.6). Specifically, the memory location of elemental and vertex data is rearranged based on the online or offline status of the data. Online elements are placed contiguously at the beginning of the array and offline elements are then located contiguously after the last online element. In this memory arrangement, the number of loop iterations can be trivially reduced by the number of offline elements on each processor because the online-state and offline-state data is contiguous in memory.

In order to produce a conforming finite element mesh when the elemental loops are reduced in length, an online element's vertices must be located in the online-state portion of the array. The conforming property of the mesh is ensured by iterating through the online-state element-to-vertex connectivity table and placing the online-state vertices at the start of the array while making sure that the vertices are represented no more than once in memory. The process is repeated for the offline-state portion of the element-to-vertex connectivity array but placing these offline-state vertices after the last online-state vertex.

Before the calculation begins, the elemental and vertex loop extents are set to the memory location of the online element and online vertex whose adjacent memory location is classified as offline-state. Note that mesh data located in the offline-state portion of the array may represent dry-state boundary conditions or halo ghost structures that are used for parallelism, however,

To trigger a rebalance event during a coastal flood, the set of vertices that are shared between the boundary of the online-state and offline-state portion of the mesh (termed checkpoint vertices and illustrated in Figure 4.2) are checked for wetting every simulation timestep. If any checkpoint vertex is wetted on any processor, the simulation must temporarily stop and rebalance the data by executing the parallel partitioning algorithm (c.f., Section 4.3.3.1) and rearrange the vertex and elemental loops.

4.3.4 Rebalance strategies

The frequency of rebalancing events can influence the overall timing statistics of the simulation. In our application, the approach may fail to produce speed ups if the simulation requires a rebalance event too frequently because the time required to complete one rebalance can become comparable to the time spent in the calculation phase. Thus, a buffer zone of elements and vertices between the online and offline portions of the computational domain must be created to ensure the time spent rebalancing is significantly less than the time spent computing. While the formation of the buffer zone will inherently lead to a suboptimal speed up by leaving a number of dry-state DoFs in an online-state, we find that it enables the approach to be feasible for realistic total water level and flooding problems that typically contain a periodic flooding/drying signal associated with the tides and would otherwise demand nearly continuous rebalancing.

In the context of the modeling coastal flooding, the criteria for configuring the width of the buffer zone is based on the location and movement of the wet/dry boundary. The key assumption in the formation of the buffer zone is that the solution behaves in a physically realistic manner and does not exhibit numerical artifacts or instabilities. In the current implementation, the definition of the criteria to determine online and offline state vertices are globally defined (i.e. among all processors). The following three rebalance strategies are pursued:

1. Distance: If the vertices' nearest distance from the wet/dry boundary exceeds a user-defined threshold *BUFDIS* in geographical degrees, the vertex is set to an offline-state. If the three vertices of the element all exceed *BUFDIS*, then the element is set to an offline-state. The key assumption here is that locations in close proximity to the shoreline are most likely to be flooded.

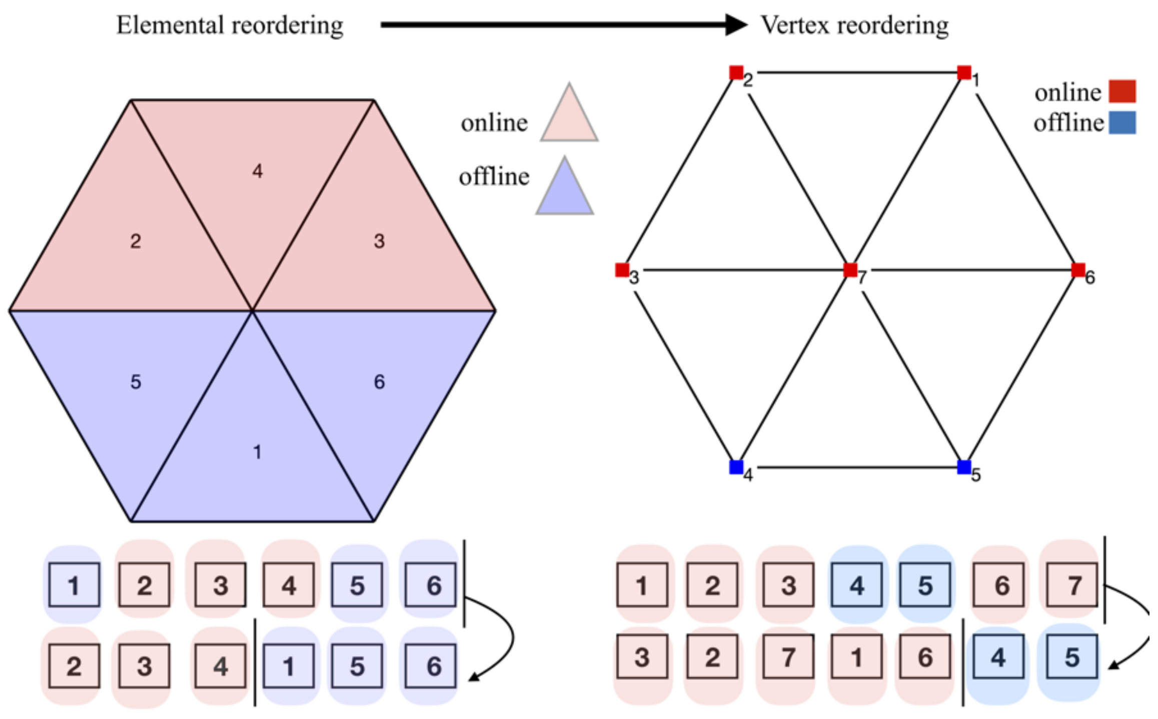


Figure 4.6. A schematic of loop rearranging applied on a patch of elements. The elemental rearrangement leads to the vertex rearrangement since the three vertices of each element that are online must be iterated over in vertex-based loops. The effect on the elemental arrays and vertex arrays on the ordering through color highlights along with the global ID numbering. The portion of the array that is not looped over in the new reordering is referred to as “clipped” and these data are shaded in light blue. The vertical black bar denotes the end of the array.

2. Topographical: The vertices' elevation must be located above a minimum elevation greater than a user-defined value $BUFDP$ in meters above the model's geoid. A key assumption here is that areas that are low-lying are most likely to be flooded. In practice, the selection of the $BUFDP$ value is motivated by the local inter-tidal range.

3. Solution-driven: The value of $BUFDP$ is elevated periodically to prevent frequent rebalancing events. In our experiments, if a rebalance event occurs less than four simulation hours since the previous one, the elevation criteria is raised by 10 cm. The period of simulation time to wait to elevate the buffer is selected by considering the overhead from rebalancing (1-3 seconds per event), time scales of hurricane-driven coastal flooding events (6-18 hours), and the computational performance of ADCIRC [145].

4.3.5 Performance metrics

A set of statistics are calculated to assess the performance of ADCIRC+DLB. The total wall-clock time T of the simulation is divided into the sum of the time spent computing T_C and the time spent rebalancing T_{RB}:

$$T = T_C + T_{RB} \tag{4.1}$$

where T_C and T_{RB} are summed over all the epochs. T_{RB} represents the time spent performing the parallel partitioning algorithm (Section 4.3.3.1).

The performance of a parallel application is dependent on the load imbalance as it determines the speed of the slowest process. The load imbalance can be strictly defined as the ratio of the maximum load L as compared to the average load calculated over all processors:

$$R_{IMB} = 100 \times (\frac{L_{max}}{L_{average}} - 1) \tag{4.2}$$

where L is calculated as the number of online-state vertices per processor. Note here I have decided to measure the number of vertices per processor to represent the computational load despite that I partition the elements of the mesh. Since the calculation inside ADCIRC depends on both elements and vertices, ideally both the number of elements and vertices per processor should be well-balanced for an efficient calculation. However, a focus is placed on the distribution of the vertices as this is not directly constrained by the graph partitioner when it partitions the mesh. For efficient parallel computing, all processors should maintain an equal load ideally having an $R_{IMB} = 0\%$ to ensure that the time spent idle is at a minimum. With the usage of ParMETIS, the desired imbalance factor option for partitioning the mesh is set to $R_{IMB} \leq 1.0\%$.

Another consideration for efficient parallel processing is to minimize the inter-processor communication volume (i.e, total number of messages by their size). The inter-processor communication volume is approximated through the surface-area-to-volume (SV) ratio of the subdomain, which approximates the relative amount of communication cost as compared to the computing cost:

$$SV = 100 \times \frac{\text{number of halo ghost vertices}}{\text{total number of vertices}} \tag{4.3}$$

The number of halo ghost vertices in Eq. 4.3 is influenced by the shape of the subdomain and whether the subdomains are connected in the computational domain (contiguity). As the number of vertices per processor is reduced, SV will increase in an analogous manner to the increase in the surface-area of water droplets as the volume of the droplet is reduced.

To measure the time savings attributed to DLB, the speed-up factor over the static version of the code is calculated as:

$$\text{speed up} = 100 \times \frac{(T_{DLB} - T_{Static})}{T_{Static}} \tag{4.4}$$

In which T_{DLB} is the total wall-clock time for a given simulation computed with the DLB enabled and T_{STATIC} is the total time to compute the simulation-only component of the run while ignoring the time spent pre-processing files using the static version of ADCIRC. An estimate of the speed up factor is calculated by dividing the maximum number of resident vertices on all processors by the number of total vertices in the problem divided by the total number of processors utilized.

4.4 Experimental results

All calculations were run on the Aegaeon computer cluster located at the University of Notre Dame's Center For Research Computing (`https://crc.nd.edu/`). The Aegaeon cluster contains 83 compute nodes, and the machine in total has 1,992 processors. Each compute node contains 72 GB of random access memory that is shared among each node's 24 processors. The nodes are connected via a high-speed 56 GB Infiniband network that facilitates message passing between processors.

Both ADCIRC v53 and ADCIRC v53 + Dynamic Load Balancing (ADCIRC+DLB) source codes are compiled identically using Intel's ifort 17.1 compiler with the -O2 optimization strategy and the MVAPICH implementation of message-passing. In all simulations, file input/output was minimized by disabling the output of global output files and file logging. ADCIRC requires that the input files for parallel simulation be pre-processed using the program ADCPREP. In contrast, ADCIRC+DLB processes the input files in parallel and in memory during the start-up period of the calculation. All timing results were repeated three times and the average of them is documented.

4.4.1 Simple Tide on an Idealized Floodplain

The DLB approach is evaluated first in an idealized problem with a simple channel and floodplain. This problem was selected because it allows for a simple tidal signal to inundate and recede on a gradual linear sloping beach, with a narrowing channel to provide variations in both horizontal directions.

4.4.1.1 Model and Setup

An idealized channel and floodplain (Fig. 4.7) are represented with an unstructured mesh with 64,415 vertices and 127,784 elements. The topography/bathymetry is characterized by an axially-symmetric channel that has a parabolic cross-sectional profile. The model contains a large floodplain and it has an initial distribution of vertices with approximately 66% dry and 34% wet. Bathymetry varies linearly and gradually with a constant slope of ≈ 0.002 from -8 m below LMSL to 2 m above the model's geoid on the floodplain. The resolution also varies from a maximum size of 75-m near the open boundary to 15 to 35 m near the shoreline and overland.

A periodic boundary condition is applied with an amplitude of 1 m and a period of 12.42 hr (semi-diurnal frequency). This wave extends from the bottom of the computational domain causing an episodic inundation of 13,240 vertices or 20.55% of the total vertices (Fig. 4.7). The application of the elevation specified boundary in this way creates a wet/dry boundary that rises in a "rigid" fashion that mimics the rise of a storm surge in a small enclosed estuary. The simulation uses a 1-second timestep and solves the GWCE mass matrix with an explicit mass-lumping approach.

To assess the performance of using ADCIRC with dynamic load balancing (ADCIRC+DLB) capability, the model is run on seven processor configurations (3, 6, 12, 24, 48, 96 and 192 processors) that span a wide range of number of DoFs-to-processor that are utilized in coastal modeling applications with ADCIRC (i.e., 300k to 500 DoFs per processor). Timing statistics are compared to the static version (V53) of

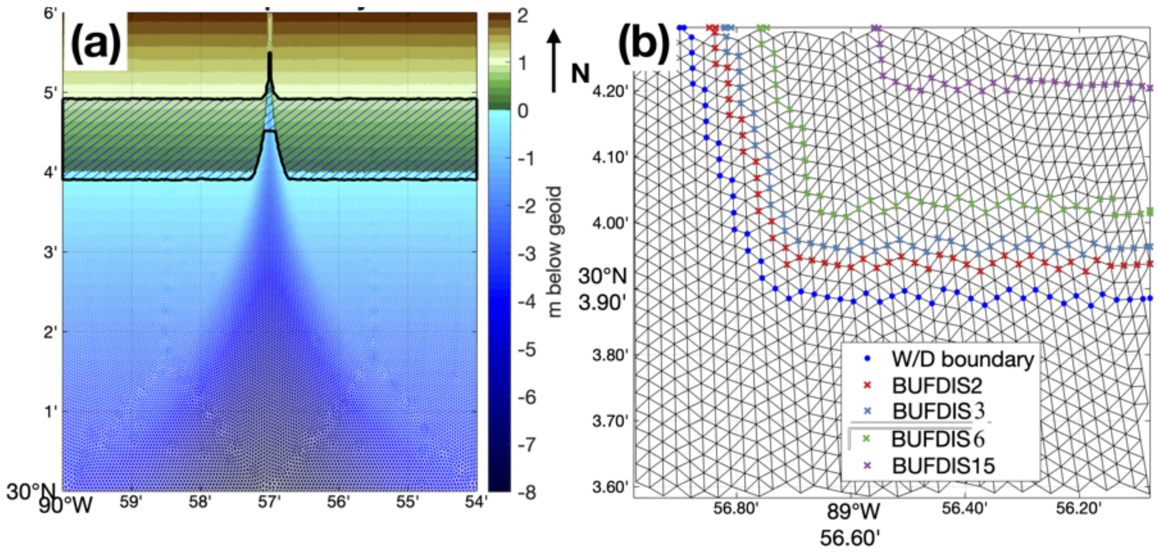


Figure 4.7. The rigid lid problem. Panel (a) shows the model's bathymetry on top of the triangular elements. The elevation specified boundary is indicated as a thick dashed black line. The hatched region indicates the areal extent of the inundation. Panel (b) shows a close-up of the wet/dry (W/D) boundary the buffer configurations that were tested.

ADCIRC. Each processor configuration is run with a sequence of five progressively larger buffer configurations that vary the separation zone between online and offline data between 2-3 elements ($\approx$ 50-m) BUFDIS2, 3-4 elements ($\approx$ 100-m) BUFDIS3, 6-7 elements ($\approx$ 200-m) BUFDIS6, 15-16 elements BUFDIS15($\approx$500-m) and an infinite size BUFDIS∞ (Figure 4.7). The range in BUFDIS represent a range of practical selections given the size of the inundation extent ($\approx$1.2 km) and the maximum element sizes overland ($\approx 15-35$-m) (Figure 4.7(b)). BUFDIS2 ($\approx$50-m) implies at least one or two elements separate the online portion of the mesh from the offline zone, and this represents a lower bound selection for BUFDIS. In contrast, a selection of BUFDIS∞ implies that no elements and vertices are turned offline and the computation never rebalances, thus this represents an upper-bound selection for BUFDIS.

The application of the weighting scheme (Section 4.3.3.2) with the BUFDIS2 configuration greatly enlarges the number of vertices in a number of partitions located

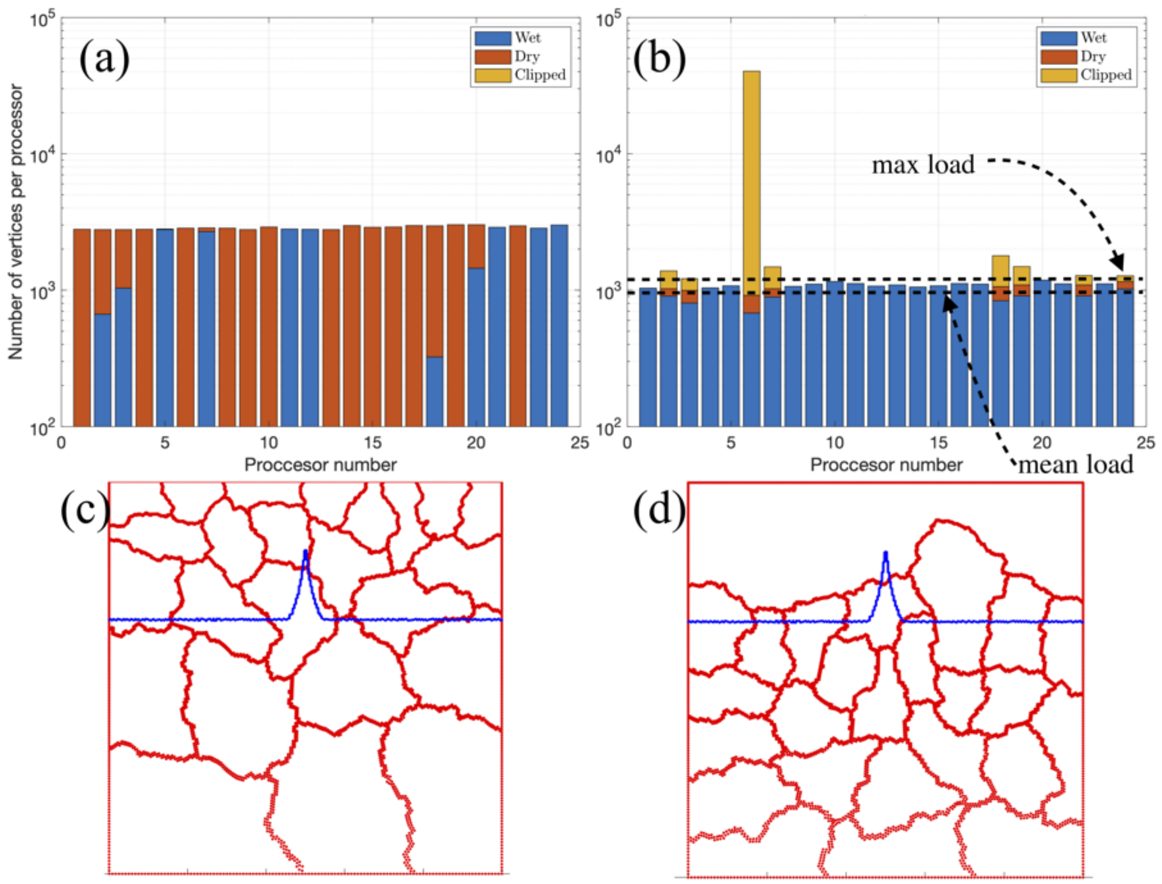


Figure 4.8. Panel (a) depicts the vertex distribution on a 24 processor decomposition with BUFDIS*inf* configuration and (b) with the BUFDIS2 configuration. Panel (c) and (d) indicate the extent of subdomains each owned by a processor.

in overland regions as compared to the case when everything is weighted equally (Figure 4.8(c)-(d)). Consequently, the partitions that contain many wet-state DoFs substantially shrink in number of DoFs and, in general, the number of dry-state DoFs is greatly reduced per processor (Figure 4.8(a)-(b)). The distribution of online-state data is balanced between processors (i.e., computational load) even though the number of total number of vertices (i.e., online + offline state) per processor is largely disparate (memory load) (Figure 4.8(b)).

4.4.1.2 Timing Improvements and Scalability

For processor configurations from 3 to 48, ADCIRC+DLB reduced the total wall-clock time by to 45.5% 10.0%, respectively, relative to the static version of ADCIRC (Figure 4.9). BUFDIS6 featured the greatest speed-up with the exception of 96 and 192 processor configurations in which a larger BUFDIS15 produced the least slow down (Figure 4.9(a),(d)). For buffer configurations greater than 48 processors, the total wall-clock time did not exhibit a reduction at a linear rate with a proportional amount of computational resources as static ADCRIC did (Figure 4.9(b)).

As the buffer is reduced in width, there were a greater number of rebalance events (Figure 4.9(c)). When the time spent rebalancing is removed from the total simulation time (Figure 4.9(b)), BUFDIS2 has the quickest execution time and progressively wider buffer configurations are slower with the exception of the 192 processor configuration demonstrating the work savings is proportional to the number of dry-state DoFs set to an offline-state (Figure 4.9(c)). While a thinner buffer width reduces the T_C, an inflection point develops in T at a point in which the time spent rebalancing exceeds the savings from the approach (Figure 4.9(a)). As more processors were used, T_C was reduced and becomes closer in time in comparison with T_{RB}, making it more challenging to achieve a speed up.

An theoretical maximum speed up in which all dry-state DoFs have zero cost could only be achieved with a perfectly balanced computational load and zero time spent rebalancing. In practice, a net zero time spent rebalancing is un-achievable and a load imbalance is expected, thus a reduction in the speed up factor from its optimal value is expected. The theoretical maximum speed-up for this problem ignoring all dry-state DoFs at every timestep is 59.0%, which is roughly 15.0% greater than the largest measured speed-up of 45.5%. When the speed-up factor is computed with the T_{RB} removed from the total time T, the speed up factor ranges between 47.8% to 11.9% using 3 to 48 processor configurations, respectively. Because the rebalance

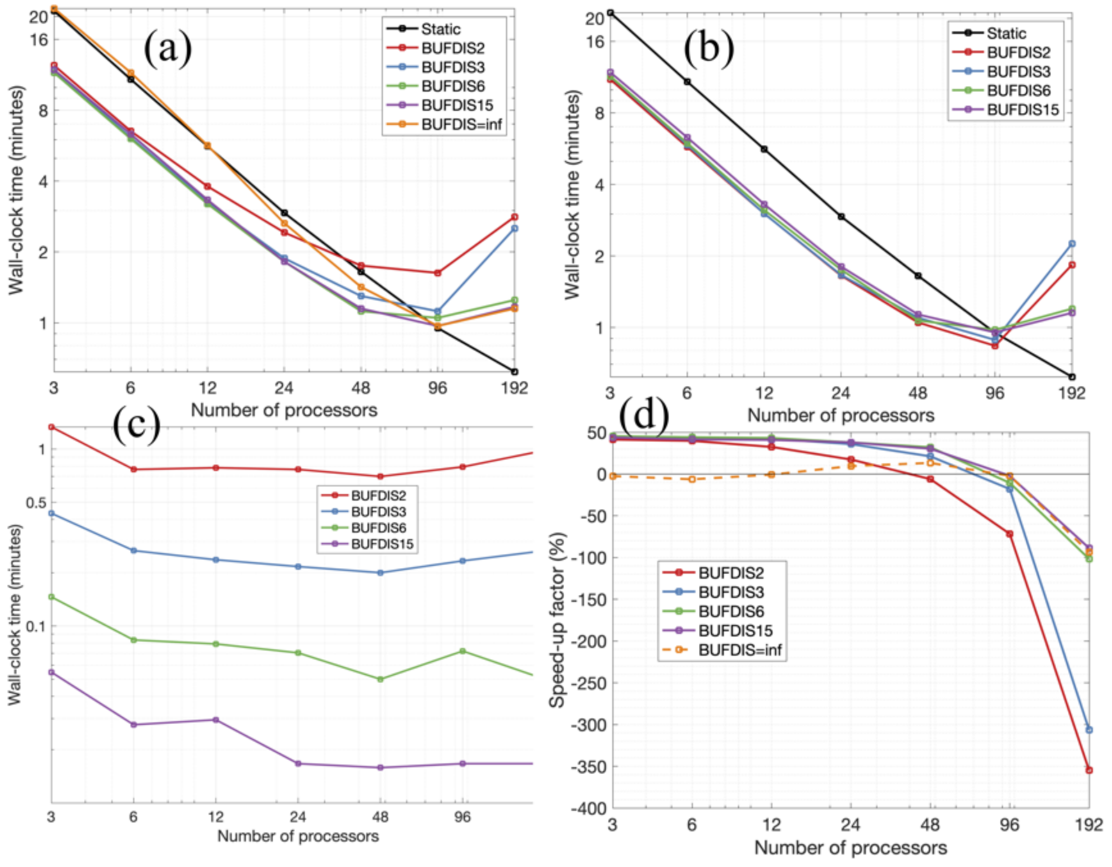


Figure 4.9. Timing statistics for the rigid lid problem as a function of processor configuration and size (number of elements) of the separation between the online and offline portions of the computational domain (BUFDIS). Panel (a) shows the total simulation time T, panel (b) the time spent performing only the computation T_C, panel (c) shows the time spent rebalancing T_{RB}, and panel (d) shows the speed-up over the static calculation.

strategy is based only on the landward movement of the wet/dry boundary, the ebb of the wet/dry boundary is not taken into consideration to trigger rebalance events. If the ebb of the wet/dry front is not considering in the calculation of the optimal speed up, then the optimal speed-up is 50.1%, which is closer to peak measured speed ups of 45.5% (and 47.8% that neglected T_{RB}).

Both ADCIRC and ADCIRC+DLB demonstrate similar scalability for the rigid lid problem up to 192 processors when the number of DoFs for each processor configuration are plotted against the number of online-state DoFs (Figure 4.10). Note that the scalability of the default application is achieved when the loop-clipping approach is deactivated (BUFDIS*inf*). The predominate differences in the scaling curves between static and ADCIRC+DLB are a shift slope and a y-offset in the scaling curves. The difference in the slope of eac scaling curve becomes steeper for thinner BUFDIS as the simulation's average number of DoFs per processor no longer approximates the highly variable problem size due to more substanial loop-clipping. Further, the consistently larger y-offsets observed in the scaling curves with thinner BUFDIS are the result of a greater amount of time spent rebalancing.

The simulation average surface-area-to-volume ratio SV and load imbalance R_{imb} are both depicted in Figure 4.11 for all buffer and processor configurations. Thinner buffer configurations reduce the problem size by a significant amount (40-60%) that the problem becomes over subscribed to computational resources given the scalability properties of ADCIRC [145]. This is evident by the large increase in both the SV as compared to the same statistics for the static decomposition using the same number of processors. For example, the maximum value of $SV = 29\%$ occurs with the thinnest BUFDIS2 configuration using the greatest number of processors (192 processes) and the SV reduces with both wider buffer configurations and fewer processors.

Further, the thinnest buffers (BUFDIS2 and BUFDIS3) produce the maximum values of R_{IMB}=44-46% indicating large load imbalances (Figure 4.9. The 192 pro-

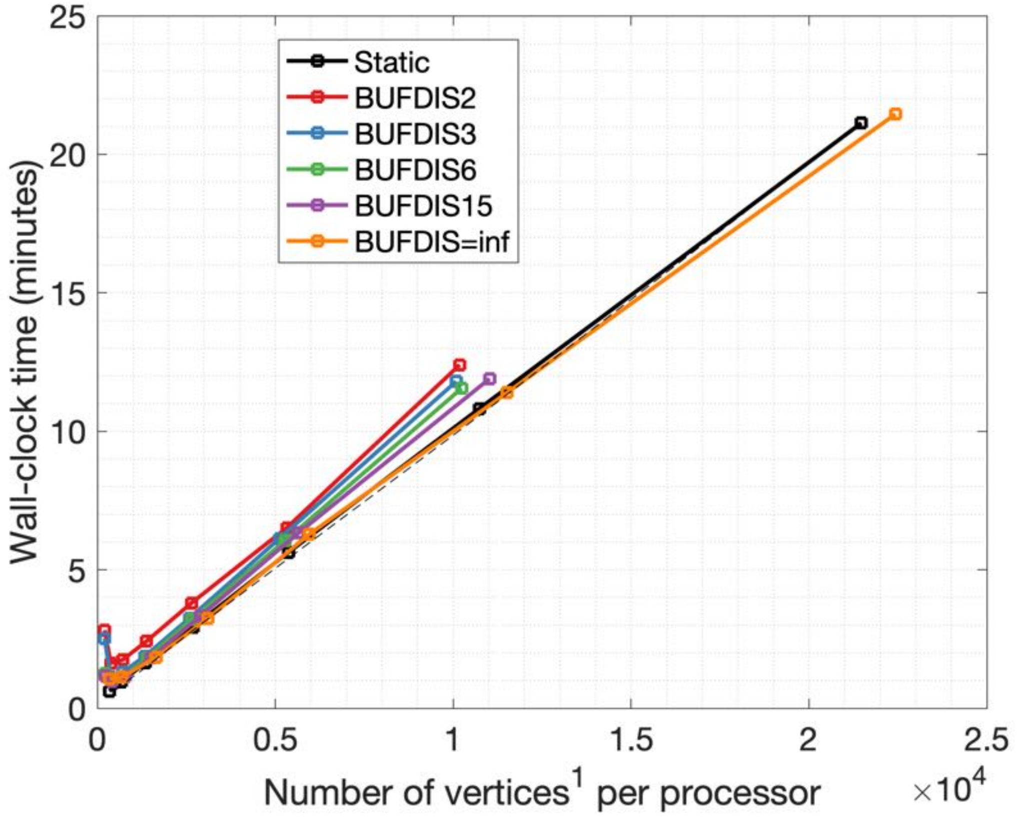


Figure 4.10. Timing statistics for the rigid lid problem as a function of number of vertices (degrees-of-freedom) per processor. Note that for ADCIRC+DLB, the number of vertices per processor is defined as the average number of online (i.e., non-clipped) vertices over all epochs (see Section 4.3.5). The average number of vertices per processor includes halo ghost zone vertices.

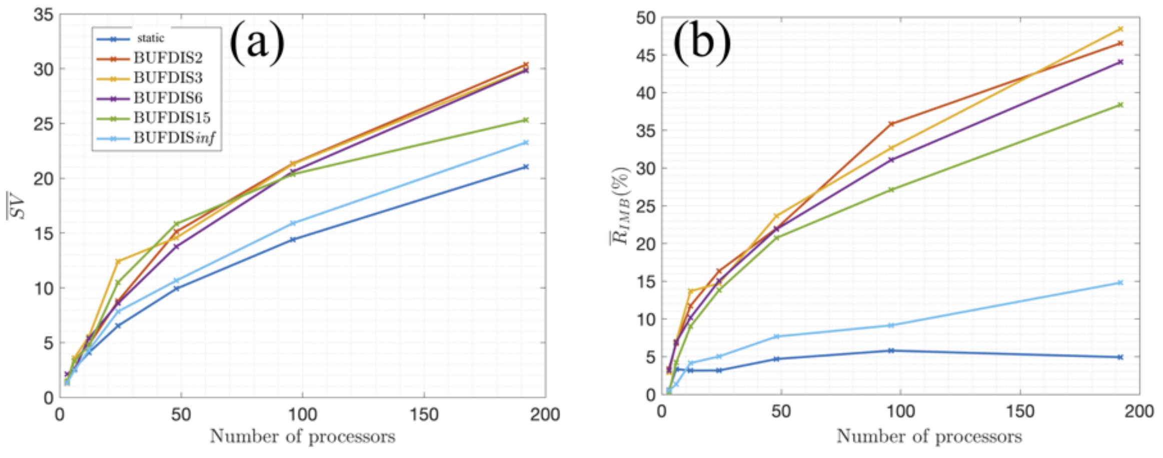


Figure 4.11. Panel (a) shows the simulation average surface-to-area volume SV (Eq. 4.3). Panel (b) shows the simulation average load imbalance R_{IMB} (Eq. 4.2).

cessor BUFDIS2 configuration is decomposing an effective problem size of 23k DoFs (when the offline-state vertices are subtracted out), which leads to approximately 100 DoFs on average per processor. Approximately 100 DoFs per processor is not expected to scale with the ADCIRC solver, with the expected linear scaling range ceasing around 1,000 DoFs-per-processor for the model setup used here [145]. The R_{IMB} and SV are only slightly larger by 2-4% over the values measured for ADCIRC when the BUFDIS is set to *infinity*. In BUFDIS*inf*, the problem size is approximately the same as ADCIRC's and default performance in these statistics is measured, which agrees with the timing results (Figure 4.10). In summary, thinner buffer widths with many processors can result in large load imbalances and great communication volumes; however, these aspects are primarily the result of over subscribing to computational resources as the application of loop-clipping reduces the size of the computational problem. The default scability of the application can be restored by increasing the width of the buffer or using fewer processors.

4.4.2 Flooding from Hurricane Irene in North Carolina

The performance of ADCIRC+DLB is further assessed in a hurricane-driven coastal flooding simulation of Hurricane Irene (2011) using a validated mesh of the mid-Atlantic United States region. The problem was selected for the following three reasons: 1) the mesh is built using measured geospatial datasets (e.g., shoreline boundaries, and seabed topography) creating a more realistic problem, pre-existing coastal ocean modeling applications have validated this model during Hurricane Isabel (2003) [22], Hurricane Irene [53], and the model is used for real-time predictions for the North Carolina Forecasting System [21], and 3) the mesh contains a large quantity of dry-state DoFs.

Hurricane Irene impacted the mid-Atlantic region as a weak hurricane with 10-minute sustained winds of approximately 78 mph on August, 27-28 2011 [137]. The hurricane generated water levels of approximately 3-m above local mean sea level on the westward side of the cyclone's track near the Tar/Pamlico Sound and Neuse River basins in North Carolina [Figure 4.12; 53]. Elsewhere in the Pamilco Sound, peak water levels of 1-m to 2-m above local mean sea level were observed.

4.4.2.1 Model and Setup

The mesh used for this experiment is referred to as North Carolina V9 (NC9) and contains 608,114 vertices and 1,200,767 elements with the finest resolution located nearshore of approximately 30-m and expanding to approximately 500-m in size over the floodplain. At initialization, the vertices are distributed as approximately 56% dry. The inter-tidal zone produces water levels that maximally range between $\approx$ 0.40 to 0.60 m above sea level and approximately 5 to 6% of the floodplain vertices wet with solely the tides (Figure 4.12). The simulation in this configuration has a maximum theoretical speed up of 49% given the flooding from wind and tides. The extent of the floodplain for the NC9 mesh is substantial and was designed based on a historical review of flooding in the region by the developers of the model [21]. In areas nearby large rivers such as the Tar and Neuse River, the inland extent of the model extends up to the 8-m elevation contour above mean sea level. Outside of the riverine areas, the model domain extends up to the 15-m elevation contour above sea level.

The run setup used in this section resembles the typical operational simulation [59]: a duration of approximately one week (8 simulation days), explicit numerical scheme, 0.5 second timestep, winds and tides, and point-based outputs of free surface elevation. Wind and pressure data from Ocean Weather Incorporated (OWI) are used to force the model between the dates of August 21, 2011 12:00:00 UTC to August 29, 2008 00:00:00 UTC at a time increment of 15-minutes. The OWI wind and pressure products were available on two structured grids: an outer nest covering the western North Atlantic with a horizontal resolution of approximately 10 km and an inner nest with a horizontal resolution of approximately 1 km that covered the North and South Carolina region in greater detail.

Temporally consistent tidal processes are included in the simulation by specifying elevation boundary conditions on an open ocean segment using the TPXO9.1 atlas [1] for four major semi-diurnal (M_2, N_2, S_2, K_2) and four major diurnal tidal constituents (K_1, O_1, P_1, Q_1). Nodal factors and equilibrium arguments are computed and applied for the simulation start time. Forcings are ramped from a quiet resting state using a hyperbolic tangent function over the first two days of the simulation to avoid exciting spurious modes in the study region. Total water level heights are recorded every 6 simulation minutes from simulation days 2 to 8 at 74 rapidly deployed gauges by the United States Geological Survey Service (USGS) and an additional 8 National Oceanic Service (NOS) gauges depicted in Figure 4.12.

[1] `http://volkov.oce.orst.edu/tides/global.html`

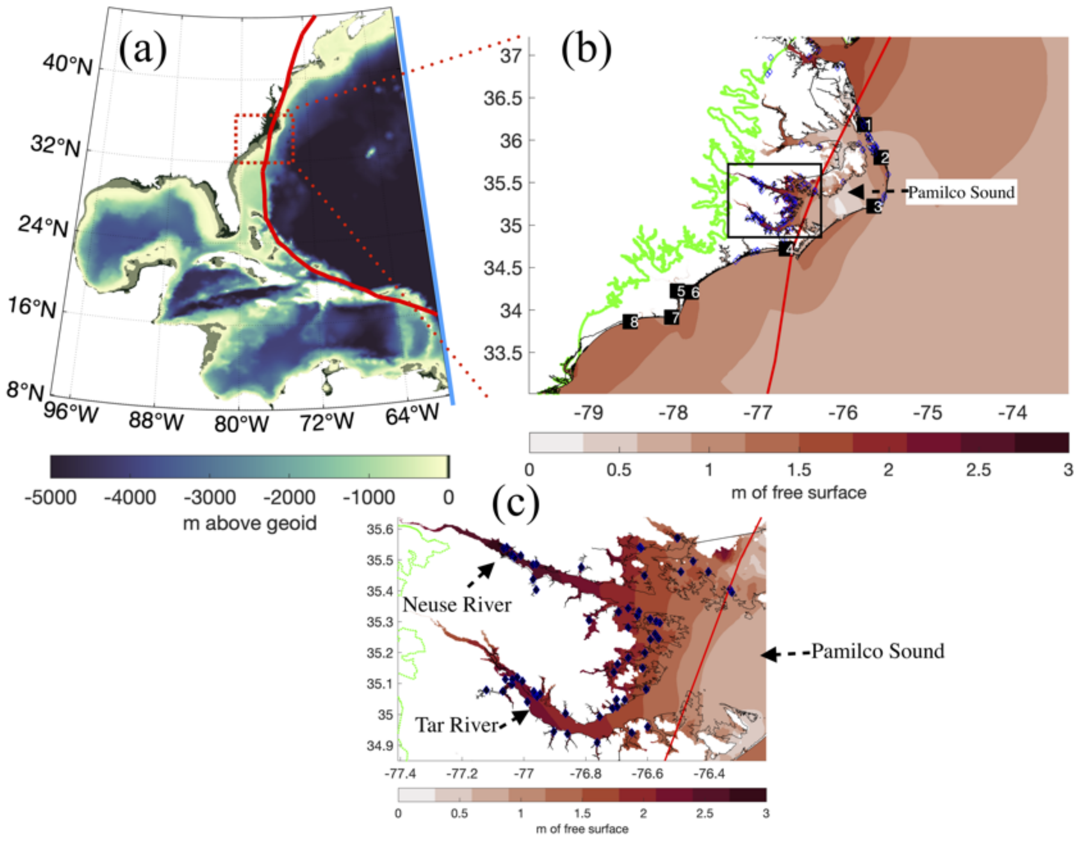


Figure 4.12. The (a) mesh's topography/bathymetry with the region of interest around the Outer Banks of North Carolina indicated by the red dotted box. In panel (a) the solid blue indicates the open ocean boundary where elevation forcing is applied. In panel (b), the simulated maximum free surface elevation above the geoid during the simulation of Hurricane Irene is shown along with the location of eight NOAA gauges (black squares) and 118 USGS high water marks (blue diamonds). The location of the model's extent is indicated by a green line and the track of Hurricane Irene is annotated as a thick red line in all panels.

4.4.2.2 Simulations

The model setup is executed on five processor configurations (60, 120, 240, 360, 480 processors) that lead to approximately 10,000 to 1,000 DoFs per processor with static ADCIRC, respectively. In contrast to configuring the buffer zone based on distance from the wet/dry boundary (Section 4.4.1), buffer configurations are determined in this section by a combination of elevation criteria (BUFDP) and the frequecy of rebalance events. For this problem, the distance-based buffer configurations did not perform well given the highly variable NC9 mesh resolution overland, which complicated the definition of buffer width, and these results were not shown.

Simulation 1 used a BUFDP at the start of the simulation which was located in the elevation range periodically flooded by the tides (e.g., 10 cm above local mean sea level), while simulations 2 and 3 were elevated above the inter-tidal range (e.g., greater than 50 cm above local mean sea level). If a rebalance event occurred less than 4 simulation hours since the last event, the program would elevate BUFDP by 10 cm above its last value to relax the frequency of rebalancing. The selection of these solution-driven buffer adaption parameters were chosen to prevent excessive rebalancing and $\pm 5\%$ perturbations to the parameters did not change the overall timing statistics significantly. The selection of the buffer parameters represent reasonable initial selections given the physical lengthscales involved in the problem and computational domain.

4.4.2.3 Comparison with measured data

Given the modifications to the program to support DLB, it is important to demonstrate first that the ADCIRC+DLB can reproduce simulation results that match with the default ADCIRC solver. Both ADCIRC and ADCIRC+DLB produced identical time series results with differences on the order of millimeters at both the eight NOS gauges (Figure 4.13) and 74 USGS gauges (Figure 4.14). Difference on the order of millimeters is to be expected given the parallel implementation of the logic-based wetting/drying algorithm [49]. Compared to observations of measured water levels, the timing and peak of simulated water level responses at the eight NOAA Oceanic Service (NOS) gauges in the region were accurately captured with the exception of Stations 1 and Station 4, which both underpredicted the water level response (Figure 4.12). Agreement between simulated results and high water mark observations at 74 USGS sensor in the Neuse, Tar, and Pamilco river basins and between ADCIRC and ADCIRC+DLB does not vary by more than 0.5 mm. Together the agreement with measured data and the static ADCIRC solution demonstrate that ADCIRC+DLB is both capturing an accurate system response to within 0.001% of the existing static ADCIRC solver (Figure 4.14).

4.4.2.4 Timing Improvements

The total time spent performing the simulation is shown in Figure 4.15(a). For the three elevation-based buffer configurations, ADCIRC+DLB completed the simulation in less wall-clock time with speed ups ranging between 7% to 21%. However, for all buffer configurations using 480 processors, the application was slowed down by 6% to 15% (Figure 4.15(c)). The largest speed ups occurred when the initial BUFDP was set to the lowest elevation above sea level (0.10 m) and the least speed ups occurred with the highest elevation buffer (1.0 m), which demonstrate that the speed up is proportional to the number of dry-state DoFs set to an offline-state.

The cumulative time spent rebalancing T_{RB} was relatively small occupying between 0.001% and 2.33% of the total static simulation time (0.33 to 0.70 wall-clock minutes) (Figure 4.15(b)). The T_{RB} was at a minimum when using the 120 processors configuration and became marginally greater by 10 to 30 seconds when using both more and less processors, but this variation in T_{RB} is small in comparison with the overall simulation time (30 to 250 wall-clock minutes). For lower elevation crite-

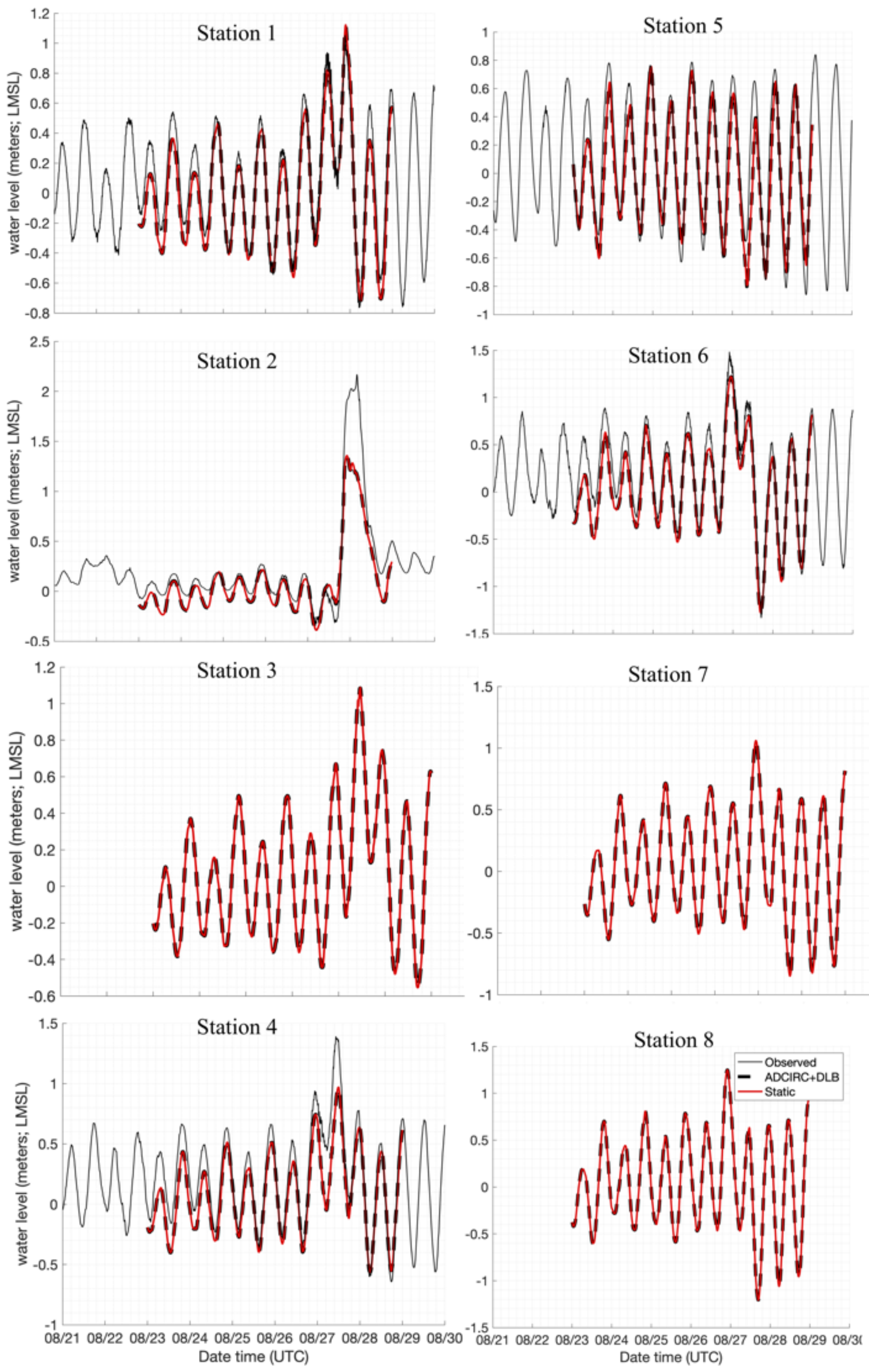


Figure 4.13. Time series comparisons of total water levels at the eight National Oceanic Service (NOS) gauges (Figure 4.12). Observations are shown at a solid black line where available.

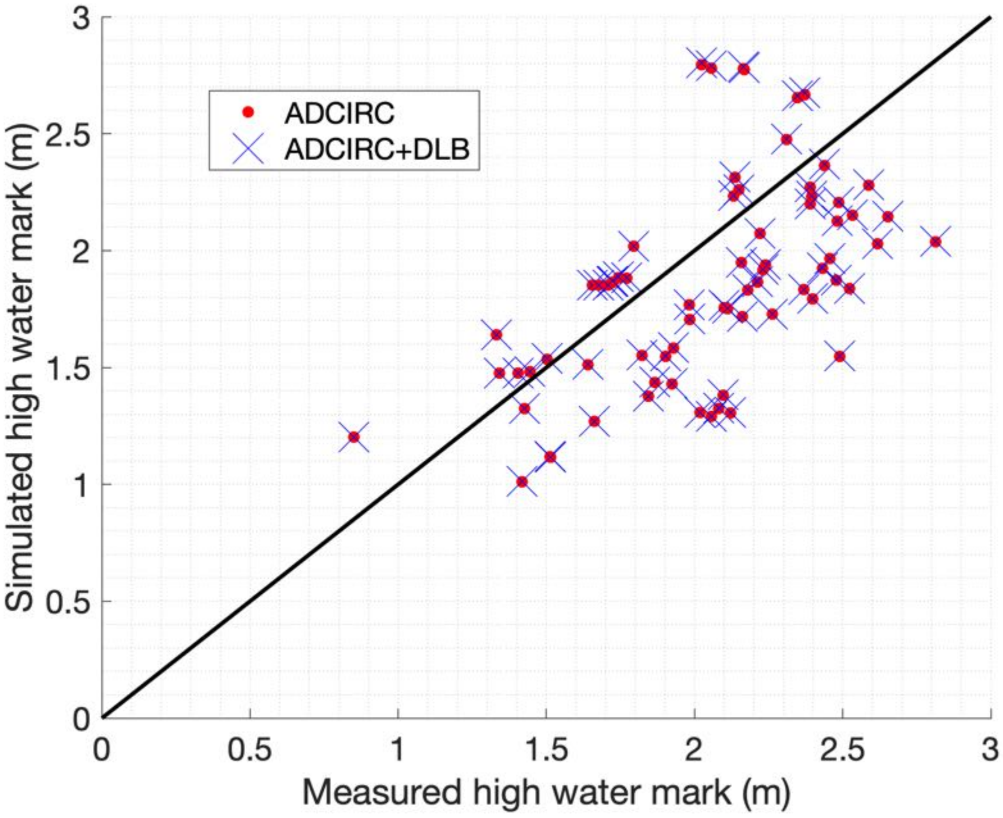


Figure 4.14. Agreement in computed high water marks between ADCIRC and ADCIRC+DLB as compared to 74 USGS high water marks (m).

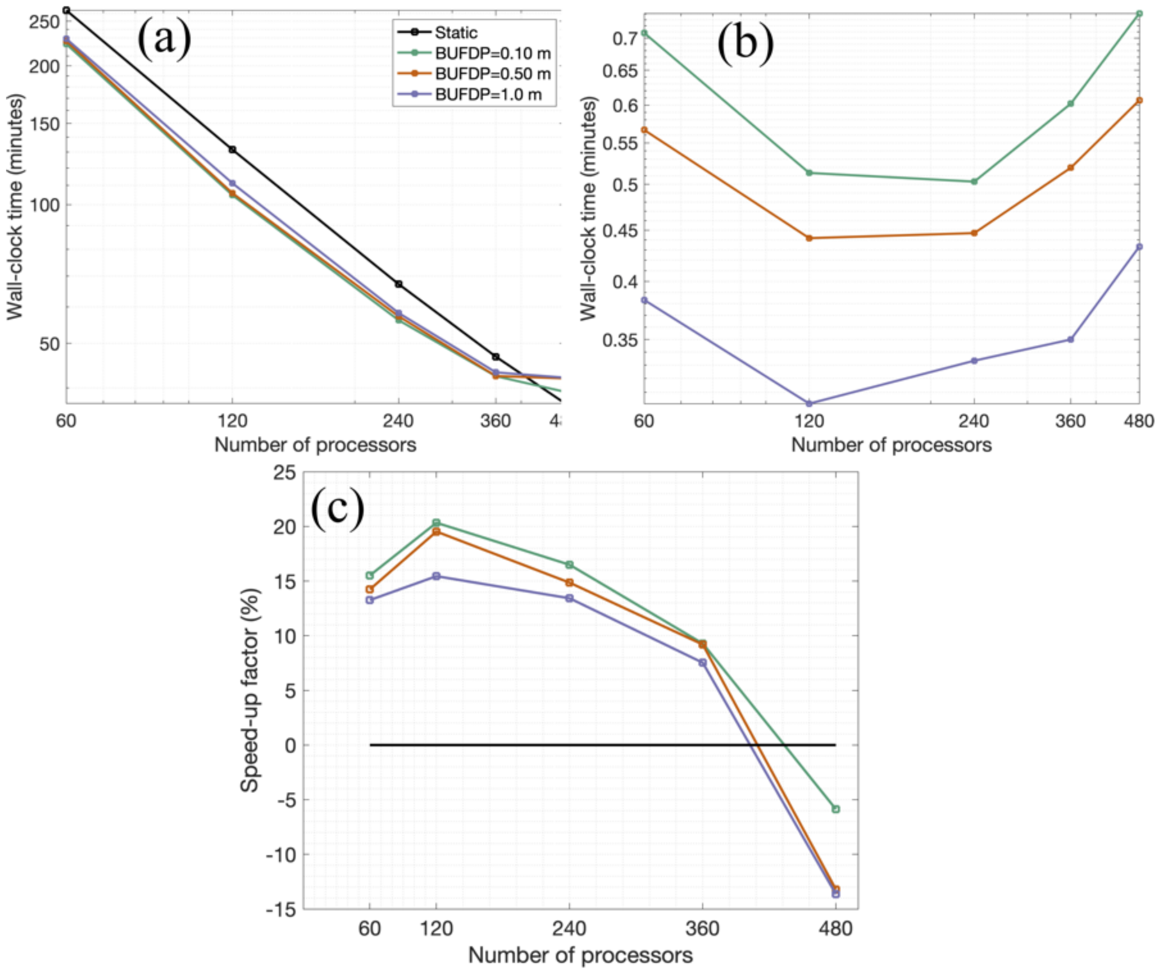


Figure 4.15. Panel (a) shows the total wall-clock time T simulating Hurricane Irene as a function of number of processors and for the five buffer configurations based on depth. Panel (b) shows T_{RB}, and panel (c) shows the speed-up.

ria (e.g., BUFDP=0.10 m), the problem rebalanced a total of 29 times, whereas for higher elevation buffer criteria (e.g., BUFDP=1.0 m) the problem rebalanced a total of 16 times (Figure 4.16). The majority of rebalance events occurred around the time of Hurricane Irene's landfall in the study region in conjunction with the period of associated coastal flooding.

The time in each simulation when rebalance events occurred is indicated in Figure 4.16. Events that occurred in quick succession (i.e., less than four hours apart) led to reductions in the estimated speed up because the program would raise the buffer elevation. By the end of the simulation's ramp period (2 days), all three buffer configurations produced similar estimated speed ups that ranged between 29% and 33%. Since BUFDP=1.0 was located above the local inter-tidal range at the start of that simulation, the first rebalance event did not occur until the time of Hurricane Irene's landfall (around simulation day 5.5). Likewise, the BUFDP=0.5 m configuration produced the fist rebalance event sooner than BUFDP=1.0 but later than BUFDP=0.1 m–at approximately simulation day 3. Overall, the lower elevation buffer configuration was not able to produce a significant differences in estimated speed ups as the tides inevitably controlled the pre-event elevation of the buffer's elevation, and this aspect is reflected in the previously described timing results (Figure 4.15).

The overall trends of the simulation average SV and load imbalance R_{IMB} are in agreement with the results also documented for the simple tide problem (Figure 4.17). However, slight reductions of 1-2% were measured in the SV as compared to static ADCIRC, but these differences in SV are expected given the usage of ParMETIS in lieu of METIS. The computational performance for this problem is sampled further away from the scaling limit of ADCIRC naturally leading to lower SV. More noticeably, buffer configurations that set more of the floodplain to an offline state led to significantly greater simulation average load imbalances $\overline{R}_{IMB}$ that ranged between approximately 30% to 80% (Figure 4.17(b)). Even when the loop-clipping approach

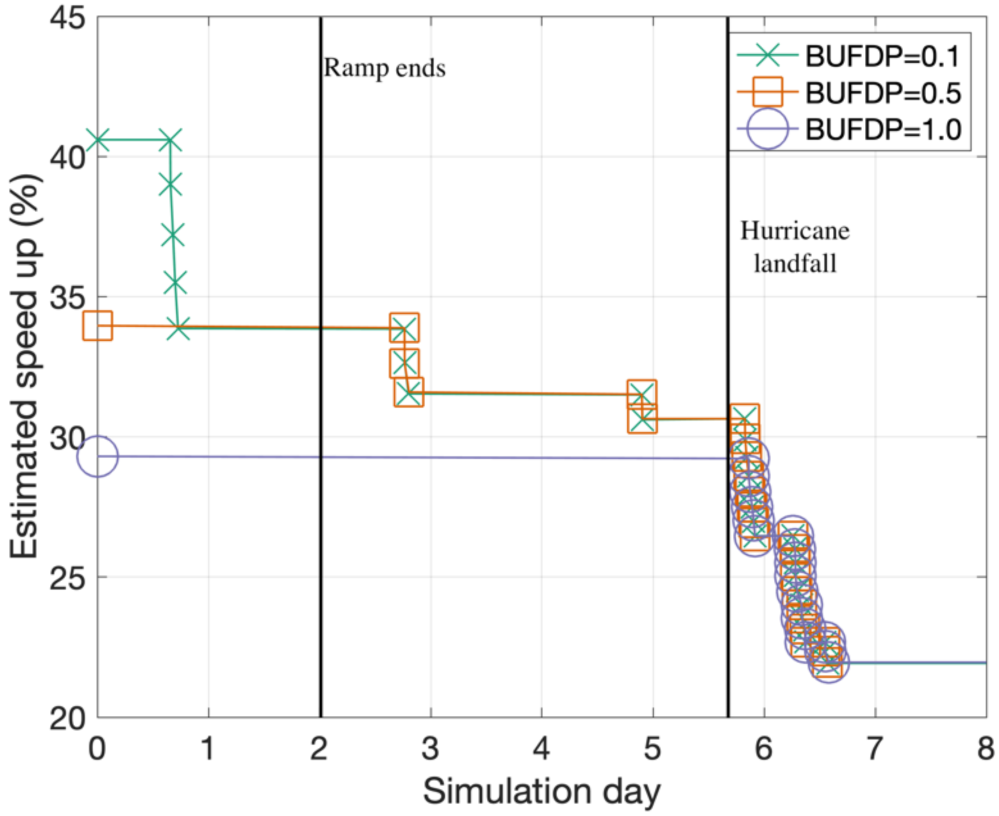


Figure 4.16. Illustrates the estimated speed up as a function simulation day. Rebalancing events are indicated by the colored markers. The solid black vertical lines indicate the ending up of the ramping period and the time of the hurricane's landfall in North Carolina.

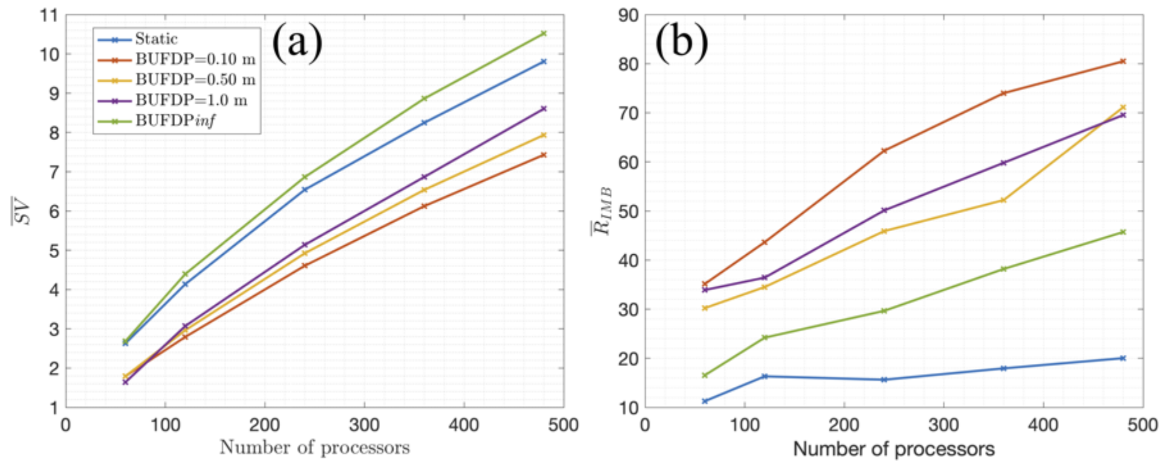


Figure 4.17. Panel (a) shows the simulation average surface-to-area volume SV (Eq. 4.3). Panel (b) shows the simulation average load imbalance R_{IMB} (Eq. 4.2).

was not utilized, a nearly two-fold increase in $\overline{R}_{IMB}$ was measured over the static approach.

In the configurations that produced large load imbalances, the distribution of DoFs per processor was inspected in greater detail. In all the configurations tested, between two to four processors contained more than 50% of the average number of online-state DoFs, while the majority of processors remained well-balanced with $\overline{R}_{IMB}$ never greater than 5% (Figure 4.18). However, since slowest process in parallel will be dictated by the process that owns the maximum number of DoFs, this explains in part the diminishing speed ups as more processors were used (c.f., Figure 4.15) and the 24.0% reduction from the theoretical maximum speed up.

Spatially, the greatest computational loads occur along the Southern Atlantic Bight in the computational domain. The anomalous loads tend to be co-located with patches of elements that are connected to only a single-neighboring element (singly-connected elements) (Figure 4.19). It is straightforward to see that elements that are connected to only a single-neighboring element contain more vertices than ele-

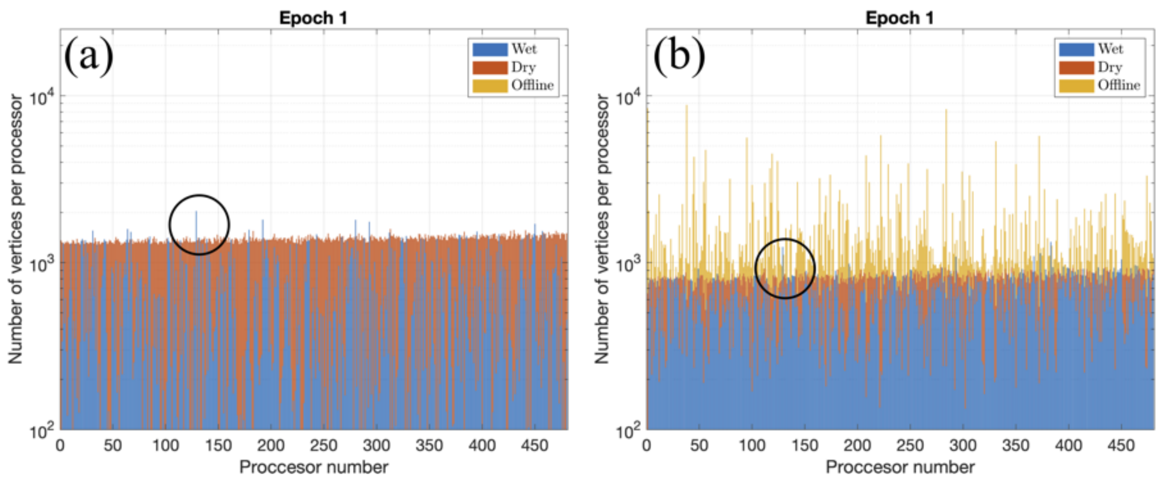


Figure 4.18. Panels (a) and (b) show the number of DoFs per processor classified as either wet, dry, and offline. The processors that own the maximum number of DoFs are circled in black. Panel (b) uses the BUFDP=0.1 m configuration.

ments. Since ADCIRC+DLB partitions and balances the elements of the mesh, the vertex graph can become largely imbalanced given the presence of singly-connected elemental connectivity. When the Southern Atlantic portion of the mesh is replaced with elements that contain at least 2 or 3 connected elements, the load imbalance largely diminishes by a factor of two and the speed ups increase two-fold to maximally 44.6%, which is approximately 5% less than the theoretical maximum speed up (Figure 4.20).

4.5 Discussion and Conclusion

The aim of this work was to reduce the cost of modeling wind-driven coastal flooding on unstructured triangular meshes using the ADvanced CIRCulation (ADICRC) Shallow-Water program. Regional coastal ADCIRC meshes often contain relatively large amounts of dry-state degrees-of-freedom (DoF) that are used to represent finely-detailed aspects of the coastal floodplains but this can make the models' parallel

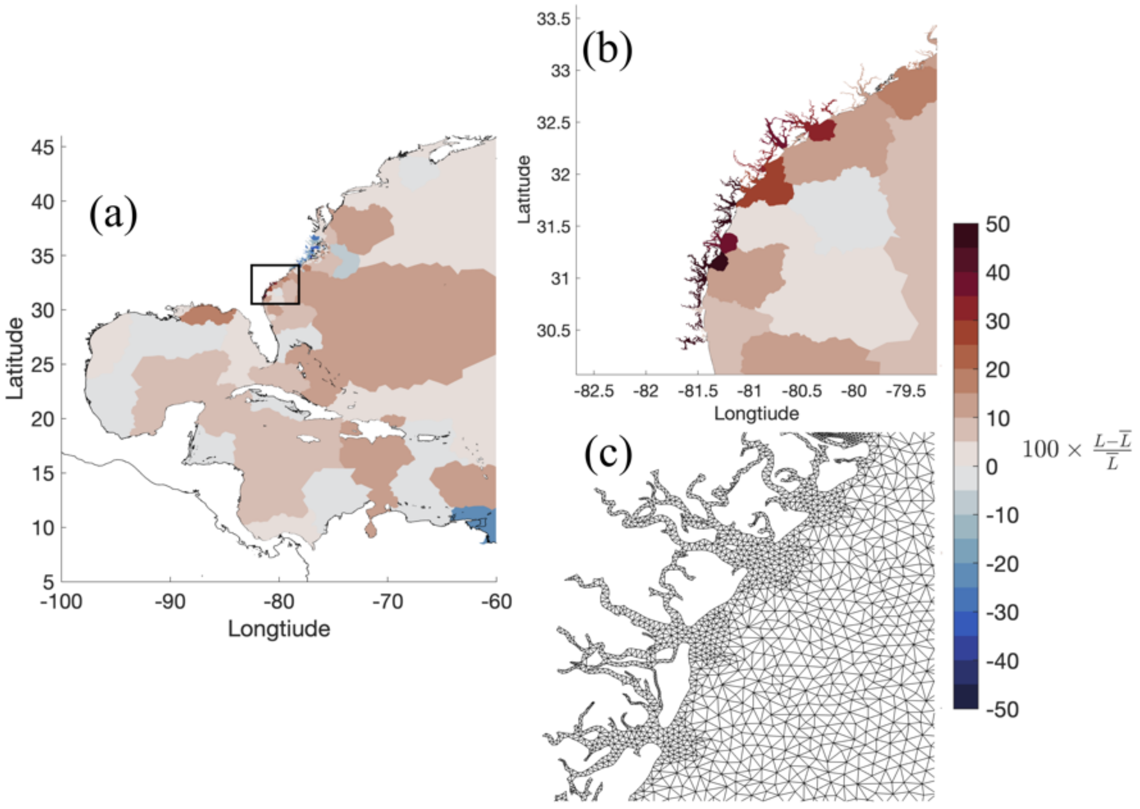


Figure 4.19. Panel (a) shows the percent anomaly in load from the mean load. Panel (b) shows the region indicated by the black box in panel (a). Panel (c) illustrates the mesh connectivity in the region that contains the largest load imbalance.

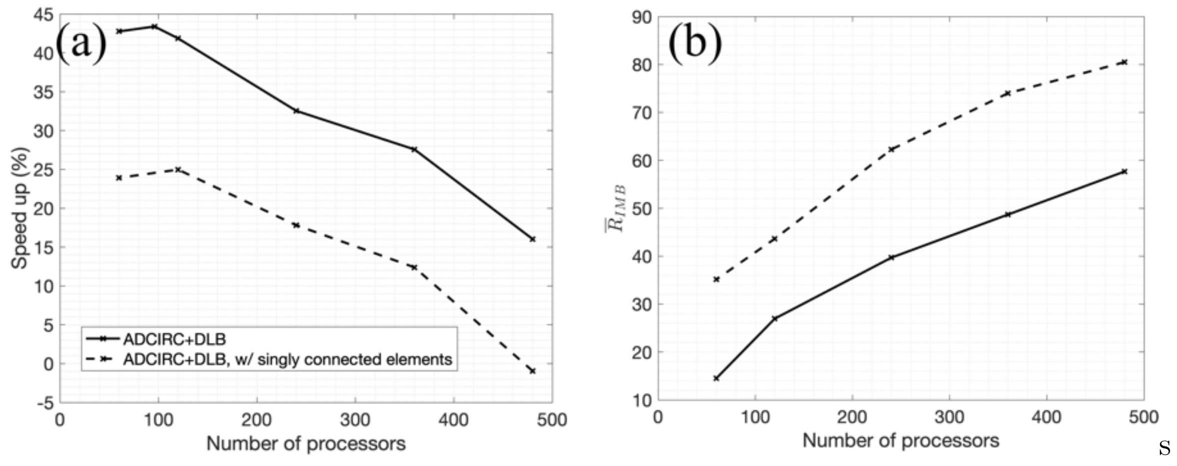


Figure 4.20. Panel (a) shows the speed up associated with the BUFDP=0.10 m simulation when the singly-connected elements are removed from the NC9 mesh (these elements were illustrated in Figure 4.19). Panel (b) illustrates the simulation average load imbalance.

execution susceptible to large work imbalances and thus inefficient.

Our solution to the load balancing problem for modeling coastal floodplains involves re-decomposing the unstructured mesh in parallel based on the location of the moving free-surface boundary. Overall, significant reductions in the total wall-clock time of coastal flooding simulations could be achieved through the integration of a dynamic load balancing (DLB) capability in the parallel implementation of ADCIRC (ADCIRC+DLB). To create the ADCIRC+DLB implementation, well-developed third-party libraries such as the Zoltan toolkit [24] and ParMETIS [83] were used to provide essential functionality. A simple elemental weighting scheme was developed along with a parallel mesh partitioning algorithm to redistribute the mesh during runtime.

For the idealized simulation and the more realistic simulation of coastal flooding driven by Hurricane Irene (2011), ADCIRC+DLB demonstrated speed ups ranging between 10-45% over static ADCIRC in the total wall-clock time. More importantly, these speed ups were proportional to the number of dry-state DoFs set to an offline-state. In the simple problem with a tide-like signal, the measured speed ups were within 3% of the theoretical maximum speed up. In the more realistic example with coastal flooding driven by Hurricane Irene (2011), peak speed up of 25% were measured that were largely 24% under the theoretical maximum speed up of 49%.

However, this large reduction in speed up from the theoretical maximum was shown to be related to the mesh connectivity. Two connected triangular elements that are connected to only each other contain more vertices than elements. Thus, when the mesh elements were decomposed between processors, large vertex-based load imbalances developed and these imbalances largely slowed down the parallel calculation. When the patch of singly-connected elements was removed from the mesh, speed ups improved (to maximally 45%) and load imbalances were reduced by nearly a factor of two. Recent advances in mesh generation techniques for unstructured coastal modeling now can produce meshes that do not contain singly-connected elements; in fact this aspect of the mesh design was directly considerd in Roberts et al. [131] and described in Chapter 2. Thus, to take full advantage of the DLB capabilities, a mesh of the coastal domain should have a bounded valency preferably with each vertex having exactly six connected neighbors. In this ideal mesh connectivity configuration, the number of neighboring elements and vertices are both six and thus both the vertex and elemental decomposition can be simultaneously load balanced optimizing performance of ADCIRC+DLB.

The performance of ADCIRC+DLB as compared to static ADCIRC was studied given a range of computational resources (3 to 480 processors) in both the idealized problem and the realistic case. Overall, ADCIRC+DLB exhibited similar scaling behavior with a linear reduction in simulation time with a proportional number of computational resources. However, the reduction in the problem size associated with DLB, given the same amount of computational resources, can lead to calculation slow-downs in some of the high-processor configurations that were tested (specifically in the idealized problem). The surface-area-to-volume ratio of the subdomain associated with reducing the problem size will grow more quickly given the reduced problem state and cause the scaling limit of ADCIRC to be reached with using fewer processors.

The success of ADCIRC+DLB to produce speed ups depended also in part on the selection of criteria that determined when to rebalance the mesh data. Rebalance criteria that was based on the topographic elevation of mesh data was found to be more robust than using a minimum-distance criteria from the wet/dry boundary. Distance-based criteria became mesh dependent as realistic problems often utilize highly variable element sizes. The topographic rebalance criteria also considered the rising rate of the water levels, which automatically lifted it out of the inter-tidal zone when tidal flooding was occurring. For example, if coastal flooding was occurring rapidly creating subsequent rebalance events less than four hours apart, then the rebalance criteria was raised by 10 cm. Overall, the rebalance criteria based on both elevation and the rising rate of water levels was successful in both producing speedups and producing a trivial time overhead. In the realistic example studied, ADCIRC+DLB produced between 20-80 rebalance events cumulatively representing 0.03 to 2.33% of the total static simulation. Thus, our recommendation for other problems is to remove as much of the coastal floodplain using a thin buffer (BUFDP=0.10 m) and let the program automatically lift it as coastal flooding occurs throughout the domain to maximize the pre-event computational savings.

Some aspects of future work concern improving the rebalance criteria to increase the amount of floodplain that can be dynamically removed from the calculation. Currently, the rebalance criteria is defined globally over the entire computational domain and can make the algorithm ineffective for modeling total water levels in regional domains that may contain substantial spatial variations in the inter-tidal range. For example, large tidal amplitudes away from the primary flooding region may result in many dry-state DoFs that remain in the calculation despite that coastal flooding may be far away from this region. Thus, future work with ADCIRC+DLB should improve the definition of the rebalance criteria based on the local water surface conditions only. The local rebalance criteria could increase the measured speed ups for models with more regional floodplain coverage by enabling more dry-state DoFs to be dynamically removed from the calculation.

Future modeling systems may contain even more than 50% of their total DoFS overland. When ADCIRC+DLB is used with these modeling systems, it may create prohibitive memory imbalances between processors. Further, the distributed data directories [120] used to locate off-processor data may become inefficient given the potentially extremely large memory imbalances and thus slow down rebalancing. A potential solution to the memory imbalance issue is to use a subset of the total number of processors for the rebalancing operation; analogous to the dedicated writer processors that were described in [145]. In the subset of processors involved with only rebalancing, the ownership of data can remain equally distributed ensuring the memory load is well-balanced and thus the search-speed of the distributed data directories can be much more efficient.

Overall, this technology represents a new direction to reduce the cost of high-fidelity unstructured, mesh-based, modeling approaches that require extensive floodplains for flooding simulations. Our approach does not sacrifice the well-established accuracy and the majority of functionality available in the ADCIRC shallow-water solver. However, future work is required to enable more of the functionality in the static ADCIRC suite. Specifically, the inclusion of the tightly coupled SWAN model [52] should be developed. The developed technology can be used for new modeling systems to be developed that can better resolve the floodplain's physical system with less worry for the associated computational cost. For example, building the next-generation comprehensive modeling system of the East and Gulf Coasts of the United States with nearly uniform 100-m resolution overland would imply that the vast majority (>95%) of vertices' would remain in a dry-state for a given event. In this case, the size of the problem and the number of dry-state DoFs would make it a suitable candidate to take advantage of ADCIRC+DLB to speed up the calculation.

CHAPTER 5

CONCLUSIONS AND FUTURE WORKS

5.1 Conclusions

This thesis examined and developed methods to improve the efficiency of numerical simulations of tides, storm surges, and the related flooding on unstructured meshes of the coastal ocean. I developed and implemented methods to automatically and efficiently construct high-geometric quality triangulations that are intended for the numerical simulation of the Shallow Water Equations (SWEs) using the Finite Element Method (FEM). These mesh generation procedures were integrated into *OceanMesh2D*, which is a freely available and well-documented open-source software. The software represents a self-contained meshing solution (i.e., it does not require external software besides MATLAB) together dealing with the geospatial data of the shoreline and seabed topograhy to create meshes of the coastal ocean. The technology also develops the scripting procedures that are necessary to automatically and efficiently build high-fidelity modeling systems of the coastal ocean anywhere in the world.

A general conclusion is that automatic mesh generation for coastal modeling has reached a new level of quality, efficiency, and automation. Perhaps the most powerful result is that the usage and modification of the *DistMesh2D* algorithm automatically simplifies polygonal boundaries that approximate arbitrarily complex shorelines. The resulting simplified shoreline boundaries can be used for a variety of purposes outside of mesh generation. In combination with the feature-size technique that was developed and improved upon to size elements nearshore, this technology enables users to develop incredibly fine-detailed coastal ocean circulation models that can resolve high-aspect ratio shoreline geometry features extending far inland. Inevitably, this capability will make it easier to develop models that can support new coupling paradigms between storm surge calculations and river discharges from hydrologic models for more physically complete total water level simulations.

Another key result was the multiscale meshing technique, which can be used to construct a single seamless unstructured mesh with smooth mesh size transitions spanning large gaps in element sizes. For the purpose of regional coastal modeling, this approach can be considered beneficial over traditional structured grid nesting approaches as it avoids the need for a coupling paradigm in the numerical solver as well as issues associated with interpolation and smoothing at the interfaces between disparate resolution nests. Further, it makes it easier to incorporate the information contained within Light RAnging and Detection (LiDAR) datasets that have relatively small horizontal resolutions ($\mathcal{O}(1$ m) horizontal resolution). The meshing capability can be used to rapidly update existing meshes objectively and largely automatically as new LiDAR datasets become available. Given the utility of these techniques and procedures, organizations such as NOAA, the Army Corp. of Engineers, F.M. Global, and Berkshire Hathaway are generating their own modeling systems with it.

Using the developed mesh generation technology, I studied the effect of variable unstructured mesh resolution based solely on *a priori* information such as shoreline geometry and seabed topographic features on the forward simulation of barotropic tides. The purpose of this work was to increase confidence in our design of high-fidelity coastal ocean models, to quantify the efficacy of current mesh design practices, and to develop new meshes of the coastal ocean with fewer vertices. Through the combination of the constraints imposed by a set of documented mesh size functions, the vertex count was reduced by nearly an order of magnitude (1.3M) from an 10.8M vertex reference mesh and had a converged solution with tidal error metrics in 99% of the East and Gulf Coast waters ranging from -2% to +1%.

Pre-existing models use nearly uniform resolution nearshore and apply a wavelength-to-gridscale sizing heuristic along the inner and outer shelf, which was criticized due to its simplistic assumptions and associated inefficiency. For example, the hand-crafted mesh used in the Hurricane Surge Operational Forecasting system (HSOFS) [146] for real-time predictions employs a minimum uniform shoreline resolution of 250 m and contains 0.75 million underwater vertices. In contrast, the converged lightweight mesh created via *OceanMesh2D* that spans the same ECGC study region as HSOFS, utilizes up to five times finer resolution nearshore (50 m compared to 250 m) and up to ten times finer resolution along the continental slope (1 km compared to 10 km), with only 1.6 times the total number of underwater vertices than HSOFS. Further, unlike HSOFS all the meshes documented in this work can be reproduced within hours following the parameters listed in the text with the software.

I highlighted that an important first step in the coastal model development procedure is to construct a mesh that minimizes the physical domain approximation error before model tuning occurs vis-a-vis varying bottom friction, other dissipative coefficients, viscous models, and manually altering ocean depths and shoreline form. As was evident in this work, by improving the accuracy of the approximate problem (i.e., the representation of the shoreline and seabed topography as per the available geospatial data used), the tidal solutions exhibited convergence towards a reference solution. The predominate improvements in the accuracy of the tidal harmonics as compared to measured data stemmed from 1) incorporating finer resolution following seabed gradients and 2) refining areas in the computational domain co-located with narrow geometries or large values of upslope area (e.g., estuarine channels). Together these two aspects of the mesh design enabled the usage of a steeper element size expansion rate that largely reduced the overall vertex count in the mesh but did not highly distort the solution. Another key result is that uniform shoreline resolution is not necessary to capture the geometric form of the shoreline and the feature-size algorithm should be used instead to produce significantly more efficient discretizations. Therefore, it is a sensible conclusion to design coastal models with the feature-size algorithm described in Chapter 2 and implemented in the *OceanMesh2D* software and to fully move away from manually and uniformly resolving the shoreline for tides, coastal flooding, and storm surge calculations.

A complementary aspect of this work was dedicated to developing technology that could effectively reduce the cost of modeling coastal floodplains with high-resolution unstructured triangles. In coastal ocean simulations, these large floodplains are allowed to wet and dry during the storm, and thus the workload for wet regions may not be distributed evenly over the computational resources. I developed a methodology and an implementation within a popular shallow-water equation solver called ADvanced CIRCulation model+Dynamic Load Balancing (ADCIRC+DLB) to dynamically load balance the floodplain component of a coastal mesh. The idea was that by producing parallel decompositions that were better load balanced, it would lead to more efficient and thus faster parallel computations. In both and idealized realistic case study, I demonstrated the application was capable of speed ups (45% and 47%) close to their theoretical maximums over the static ADCIRC. My solution to the load balancing problem involved efficiently re-decomposing the unstructured mesh in parallel based on the location of the moving free-surface boundary. A variety of well-developed software packages were integrated into the solver to produce the parallel re-decomposition of the mesh, such as Zoltan toolkit and the ParMETIS graph partitioner. A simple approach was used to reduce the cost of the dry-state DoFs by rearranging the vertex and elemental data in memory based on an online/offline status. I demonstrated that this loop-rearrangement methodology could effectively eliminate the calculation cost of the dry-state DoFS if the graph of the unstructured mesh csould be well-balanced via the graph partitioner.

The success of ADCIRC+DLB in producing speed ups for the problems studied depended in part on the selection of criteria that determined when to rebalance the mesh data. Overall, the rebalance criteria based on topographic elevation above sea level was successful in both producing speedups and minimizing the time spent in the rebalancing operation. In the realistic example studied, ADCIRC+DLB produced between 20-80 rebalance events cumulatively representing 0.03 to 2.33% of the total static simulation. Due to the regional-scale nature of a tropical cyclone event, a simulation must be run for a sufficiently long period of time pre-event to ensure the water levels equilibrate and the application of forcing functions is smoothly varying in time to avoid exciting numerical instabilities. Thus, a significant component of the simulation time is often spent pre-event when the water levels do not flood overland. Considering this, my recommendation was to configure the rebalance criteria so that as much of the coastal floodplain is set to an offline-state and let the program automatically lift the zone that trigger rebalances as coastal flooding occurs throughout the domain.

Overall, this technology represents a new direction to reduce the cost of high resolution unstructured, mesh-based, modeling approaches that include extensive floodplains for coastal flooding simulations. ADCIRC+DLB does not sacrifice the validated accuracy and the majority of functionality available in the program. However, future work is required to enable more of the nuanced functionality in the ADCIRC suite with the DLB capability.

5.2 Future work

The topics focused on in this work have led to many new questions and ideas. The following list some of the most relevant directions that could be pursued in future works:

- **Automatic extraction of shoreline datasets from satellite imagery**

  The existing approach to coastal mesh generation involves extracting a geometric elevation contour that approximates the location of the shoreline (e.g., 0-m isoline) using the methods encapsulated in *OceanMesh2D* from a DEM. However, as we begin to couple and integrate hydraulic processes into our calculations of coastal flooding, such as river discharge and precipitation run-off, we realize that important shoreline features may no longer be well-approximated by the 0-m elevation contour. This is especially the case for rivers as they often extend far above the local mean sea level. Thus, additional methods are required to correctly and automatically extract the boundary description that correctly identifies features that always reamin underwater from land-based features. Particularly, the methods documented in [20] could be implemented based on a texture-based analysis of visible satellite imagery and embedded within the geospatial processing unit of *OceanMesh2D*.

- **Anisotropic mesh generation**

  Coastal flows often enter narrow and restricted waterways in which the flow direction can become highly anisotropic. In these situations, it is sensible to incorporate such anisotropy into the underlying mesh design. In particular, elongating the element sizes in the direction of the flow can help enhance the model's numerical stability and accuracy. One possible way to create anisotropy is to generalize the pre-existing isotropic mesh size function using a metric tensor [25]. This would involve the calculation of the three additional mesh size functions (minor, major, and angle of major axis) that would be necessary for the calculation of this metric tensor. These additional components of the mesh size function could be extracted automtically from the form of the boundary description (i.e., width, sinuosity, length) and the seabed depth.

- **Integrate *a posterioi* mesh improvement techniques**

  Hagen et al. [72, 73] explored the distribution of mesh resolution for 1D models of the continental shelf and 2D tidal models through the LTEA method with the goal to reduce the overall vertex count of a tidal model using *a posteriori* truncation error indicators. An interesting future direction of some of my work would be to take a mesh created via *OceanMesh2D* that produces a low approximate error and then apply the LTEA method to further reduce the vertex count and/or reduce the numerical truncation error. This would also enable the necessary work to study how the convergence of the LTEA method is effected by the various initial point configurations.

- **Improved rebalance criteria**

  Currently in ADCIRC+DLB, the rebalance criteria is defined globally over the entire computational domain. The global definition of the rebalance criteria can make the algorithm ineffective for modeling total water levels in regional domains that may contain substantial spatial variations in the tide range. Thus, future work with ADCIRC+DLB should improve the definition of the rebalance criteria based on the local water surface conditions only. The local rebalance criteria could increase the measured speed ups for models with more regional floodplain coverage by enabling more dry-state DoFs to be dynamically removed from the calculation. One difficulty with the local rebalance criteria is defining an appropriate space around each vertex in which a specific rebalance criteria is defined while avoiding searching in parallel during run-time. One could perhaps rely on a pre-computed solution of tides to determine the local inter-tidal range and set the buffer height accordingly.

- **Dedicated processor group to perform rebalancing**

  Future modeling systems may contain even more than 50% of their total DoFS overland. When ADCIRC+DLB is used with these future modeling systems, it may create prohibitive memory imbalances between processors that may exceed the virtual memory on the processor. One way to circumvent this is to introduce an additional graph weight proportional to the amount of memory over a threshold. Further, the distributed data directories [120] used in our approach to locate off-processor data will become inefficient given excessives memory imbalances. This can slow down rebalancing operations as the hash table's performance is negatively impacted. A potential solution to the potentially poor performance of the distributed data directories is to use a subset of the processors for the rebalancing operation; analogous to the dedicated writer processors that were described in [52]. In the subset of processors involved with only rebalancing the mesh, the ownership of mesh data can remain equally distributed ensuring the memory load is well-balanced and thus the search-speed of the distributed data directories can be much more efficient.

APPENDIX A

PARALLEL WEIR PARTITIONING

Here I document the algorithm that is implemented in ADCIRC+DLB to decompose the internal-type (weir) barrier conditions. These boundary conditions represent sub-gridscale barriers such as levees and walls that are critical for modeling coastal circulations but cannot be expliclty represented at the gridscales used. The weirs in ADCIRC are a pair of vertices separated by a small distance (i.e., $<$ 10-m). The model calculates a flux across the vertices using the elevations specified on either vertex of the weir. There are also two coefficients that describe the nature of the flow overtop the weir (either subcritical or supercritical flow).

ADCIRC requires that both pair of vertices of each weir be localized on the same processor to correctly compute the flux, which creates a challenging data decomposition problem in a parallel computing environment since the graph decomposition software does cannot guarantee these constraints are enforced through weights.

In the following algorithm, a weir is composed of two vertices: one front-facing and one back-facing. Note that either the front or back facing weirs are a member of only *one* weir, in other words, either vertex cannot be shared between multiple weirs in the following algorithm. ADCIRC+DLB will strip weirs owned by multiple pairs automatically. In the following, a weir is *fixed* if both the front and back facing veritces exist locally on the processor and *cut* if either the front facing or back facing vertex is not local to the processor. An element is *connected* to another element if it shares an edge (i.e., two vertices). Further, an element is *local* to a processor if it exists in that processor's memory.

The following algorithm takes place after the parallel mesh partitioning algorithm has finished. Thus, the elements and vertices that are needed to fix the weirs are added locally as ghost elements and vertices (i.e., they are duplicated between processors and exchange information through message passing).

```
Result: A data decomposition with all weirs fixed.
while If any weir is cut do
    1. Add non-local elements connected to local elements that contain at least
    one weir vertex;
    2. If any to-be-added element cuts another weir and is not flagged, do not
    localize this element;
    3. Migrate elements found in step 2 between processors using message
    passing;
    4. Migrate vertices to support newly added elements from step 3;
    5. Check if all weirs are still cut and exit if they are all fixed. If an element
    has a weir that's still cut, flag it and repeat algorithm;
end
```

Algorithm 2: Parallel weir partitioning algorithm.

Algorithm 2 is iterative and in practice takes between 3 to 4 iterations to converge on meshes with thousands of weirs. The pre-existing algorithm used to partition weirs is largely inefficient. In fact, when decomposing on a mesh with approximately 8K weirs, the time required to produce a decomposition with all fixed weirs is between 3 to 6 times faster than the static version of the code (Figure A.1).

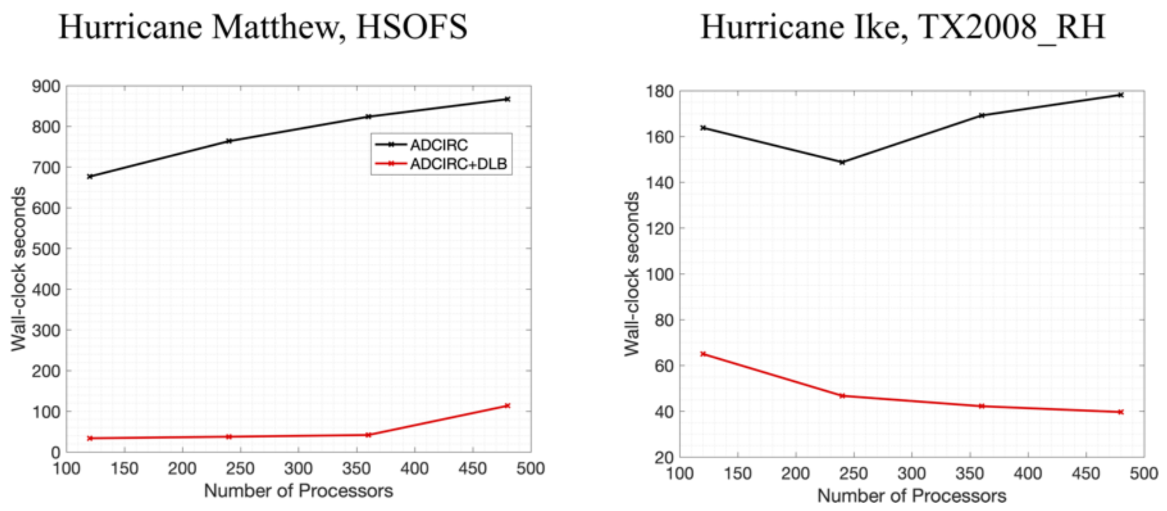


Figure A.1. The time required to initially partition a mesh for the number of processors labeled on the x-axis with 14k weirs (a) and 8K weirs (b) using algorithm 2 as compared to the serial algorithm.

BIBLIOGRAPHY

1. Coastal flooding in Scituate (MA): A FVCOM study of the 27 December 2010 nor'easter. Journal of Geophysical Research: Oceans, 118(11):6030–6045, 2013. ISSN 21699291. doi: 10.1002/2013JC008862.

2. BatTri: A two-dimensional bathymetry-based unstructured triangular grid generator for finite element circulation modeling. Computers and Geosciences, 32 (5):632–642, 2006. ISSN 00983004. doi: 10.1016/j.cageo.2005.09.007.

3. Extratropical storm inundation testbed: Intermodel comparisons in Scituate, Massachusetts. Journal of Geophysical Research: Oceans, 118(10):5054–5073, 2013. ISSN 21699291. doi: 10.1002/jgrc.20397.

4. Mesh size functions for implicit geometries and PDE-based gradient limiting. Engineering with Computers, 22(2):95–109, 2006. ISSN 01770667. doi: 10.1007/s00366-006-0014-1.

5. C. Amante and B. Eakins. ETOPO1 1 Arc-Minute Global Relief Model: Procedures, Data Sources and Analysis. NOAA Technical Memorandum NESDIS NGDC-24. National Geophysical Data Center, NOAA., 2009.

6. A. Androsov, V. Fofonova, I. Kuznetsov, S. Danilov, N. Rakowsky, S. Harig, H. Brix, and K. H. Wiltshire. Fesom-c v.2: coastal dynamics on hybrid unstructured meshes. Geoscientific Model Development, 12(3):1009–1028, 2019. doi: 10.5194/gmd-12-1009-2019.

7. S. Arya and D. M. Mount. Approximate nearest neighbor queries in fixed dimensions. In Proc. 4th Ann. ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 271–280, 1993.

8. A. Avdis, A. S. Candy, J. Hill, S. C. Kramer, and M. D. Piggott. Efficient unstructured mesh generation for marine renewable energy applications. Renewable Energy, 116:842 – 856, 2018. ISSN 0960-1481. doi: https://doi.org/10.1016/j.renene.2017.09.058.

9. I. Babuka and A. Aziz. On the angle condition in the finite element method. SIAM Journal on Numerical Analysis, 13(2):214–226, 1976. doi: 10.1137/0713021. URL `https://doi.org/10.1137/0713021`.

10. P. Bacopoulos and S. C. Hagen. The intertidal zones of the south atlantic bight and their local and regional influence on astronomical tides. Ocean Modelling, 119:13 – 34, 2017. ISSN 1463-5003. doi: https://doi.org/10.1016/j.ocemod.2017.09.002.

11. S. Bagon. Matlab class for ann, February 2009. available at `http://www.wisdom.weizmann.ac.il/~bagon/matlab.html`.

12. S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, D. A. May, L. C. McInnes, R. T. Mills, T. Munson, K. Rupp, P. Sanan, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang. PETSc Web page. `http://www.mcs.anl.gov/petsc`, 2018. URL `http://www.mcs.anl.gov/petsc`.

13. B. Balendran. A Direct Smoothing Method for Surface Meshes. In Proceedings of the 8th International Meshing Roundtable, pages 189—-193, South Lake Tahoe, California, USA, 1999.

14. D. Balk, M. Montgomery, G. Mcgranahan, D. Kim, V. Mara, M. Todd, T. Buettner, and A. Dorlien. Mapping urban settlements and the risks of climate change in africa, asia and south america. 01 2009.

15. J. Behrens. Atmospheric and ocean modeling with an adaptive finite element solver for the shallow-water equations. Applied Numerical Mathematics, 26(1): 217 – 226, 1998. ISSN 0168-9274. doi: https://doi.org/10.1016/S0168-9274(97)00090-1.

16. J. Behrens and M. Bader. Efficiency considerations in triangular adaptive mesh refinement. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 367(1907):4577–4589, 2009. doi: 10.1098/rsta.2009.0175.

17. A. Bilgili, K. W. Smith, and D. R. Lynch. Battri: A two-dimensional bathymetry-based unstructured triangular grid generator for finite element circulation modeling. Computers and Geosciences, 32(5):632 – 642, 2006. ISSN 0098-3004. doi: 10.1016/j.cageo.2005.09.007. URL `http://www.sciencedirect.com/science/article/pii/S0098300405001950`.

18. C. A. Blain, J. J. Westerink, and R. A. Luettich. The influence of domain size on the response characteristics of a hurricane storm surge model. Journal of Geophysical Research, 99:467–479, 1994. ISSN 0148-0227. doi: 10.1029/94JC01348.

19. C. A. Blain, J. J. Westerink, and R. A. Luettich. Grid convergence studies for the prediction of hurricane storm surge. International Journal for Numerical Methods in Fluids, 26(4):369–401, 1998. ISSN 02712091 (ISSN). doi: 10.1002/(SICI)1097-0363(19980228)26:

4⟨369::AID-FLD624⟩3.0.CO;2-0. URL `http://www.scopus.com/inward/record.url?eid=2-s2.0-0032005681{&}partnerID=40{&}md5=c0c10936cce1dfadc5202b6ac71c6405`.

20. C. A. Blain, R. Linzell, and P. McKay. Simple methodology for deriving continuous shorelines from imagery: Application to rivers. Journal of Waterway, Port, Coastal, and Ocean Engineering, 139(5):365–382, 2013. doi: 10.1061/(ASCE)WW.1943-5460.0000189.

21. B. Blanton, J. McGee, J. Fleming, C. Kaiser, H. Kaiser, H. Lander, R. Luettich, K. Dresback, and R. Kolar. Urgent computing of storm surge for north carolina's coast. Procedia Computer Science, 9:1677 – 1686, 2012. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2012.04.185. Proceedings of the International Conference on Computational Science, ICCS 2012.

22. B. Blanton, K. Dresback, B. Colle, R. Kolar, H. Vergara, Y. Hong, N. Leonardo, R. Davidson, L. Nozick, and T. Wachtendorf. An integrated scenario ensemble-based framework for hurricane evacuation modeling: Part 2 hazard modeling. Risk Analysis, 0(0), 2018. doi: 10.1111/risa.13004.

23. B. O. Blanton, F. E. Werner, H. E. Seim, R. A. Luettich, D. R. Lynch, K. W. Smith, G. Voulgaris, F. M. Bingham, and F. Way. Barotropic tides in the South Atlantic Bight. Journal of Geophysical Research C: Oceans, 109(12):1–17, 2004. ISSN 01480227. doi: 10.1029/2004JC002455.

24. E. G. Boman, U. V. Catalyurek, C. Chevalier, and K. D. Devine. The Zoltan and Isorropia parallel toolkits for combinatorial scientific computing: Partitioning, ordering, and coloring. Scientific Programming, 20(2):129–150, 2012.

25. S. P. Bossens Frank, Heckbert. A pliant method for anisotropic mesh generation. 5th Intl. Meshing Roundtable, 1996.

26. A. Bowyer. Computing Dirichlet tessellations*. The Computer Journal, 24(2): 162–166, 01 1981. ISSN 0010-4620. doi: 10.1093/comjnl/24.2.162.

27. J. M. Brown, D. L. Norman, L. O. Amoudry, and A. J. Souza. Impact of operational model nesting approaches and inherent errors for coastal simulations. Ocean Modelling, 107:48–63, 2016. ISSN 14635003. doi: 10.1016/j.ocemod.2016.10.005. URL `http://dx.doi.org/10.1016/j.ocemod.2016.10.005`.

28. P. Brufau, P. Garcã, and M. E. Vã. Zero mass error using unsteady wetting-drying conditions in shallow flows over dry irregular topography. International Journal for Numerical Methods in Fluids, 45:1047–1082, 2004. doi: 10.1002/fld.729.

29. S. Bunya, J. C. Dietrich, J. J. Westerink, B. A. Ebersole, J. M. Smith, J. H. Atkinson, R. Jensen, D. T. Resio, R. A. Luettich, C. Dawson, V. J. Cardone, A. T. Cox, M. D. Powell, H. J. Westerink, and H. J. Roberts. A high-resolution

coupled riverine flow, tide, wind, wind wave, and storm surge model for southern louisiana and mississippi. part i: Model development and validation. Monthly Weather Review, 138(2):345–377, 2010. doi: 10.1175/2009MWR2906.1. URL https://doi.org/10.1175/2009MWR2906.1.

30. S. A. Canann, M. B. Stephenson, and T. Blacker. Optismoothing: An optimization-driven approach to mesh smoothing. Finite Elements in Analysis and Design, 13(2):185 – 190, 1993. ISSN 0168-874X. doi: 10.1016/0168-874X(93)90056-V. URL http://www.sciencedirect.com/science/article/pii/0168874X9390056V.

31. A. Candy and J. Pietrzak. Shingle 2.0: Generalising self-consistent and automated domain discretisation for multi-scale geophysical models. Geoscientific Model Development, 11:213–234, 2018. ISSN 19919603. doi: 10.5194/gmd-11-213-2018.

32. A. S. Candy. An implicit wetting and drying approach for non-hydrostatic baroclinic flows in high aspect ratio domains. Advances in Water Resources, 102:188 – 205, 2017. ISSN 0309-1708. doi: https://doi.org/10.1016/j.advwatres.2017.02.004. URL http://www.sciencedirect.com/science/article/pii/S030917081730115X.

33. V. Casulli. Computational grid, subgrid, and pixels. International Journal for Numerical Methods in Fluids, 0(0). doi: 10.1002/fld.4715. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/fld.4715.

34. V. Casulli and R. A. Walters. An unstructured grid, three-dimensional model based on the shallow water equations. International Journal for Numerical Methods in Fluids, 32(3):331–348, 2000. doi: 10.1002/(SICI)1097-0363(20000215)32:3⟨331::AID-FLD941⟩3.0.CO;2-C.

35. C. Chen, H. Liu, and R. C. Beardsley. An unstructured grid, finite-volume, three-dimensional, primitive equations ocean model: Application to coastal ocean and estuaries. Journal of Atmospheric and Oceanic Technology, 20(1):159–186, 2003. doi: 10.1175/1520-0426(2003)020⟨0159:AUGFVT⟩2.0.CO;2. URL https://doi.org/10.1175/1520-0426(2003)020<0159:AUGFVT>2.0.CO;2.

36. C. Chen, H. Huang, R. C. Beardsley, Q. Xu, R. Limeburner, G. W. Cowles, Y. Sun, J. Qi, and H. Lin. Tidal dynamics in the Gulf of Maine and New England Shelf: An application of FVCOM. Journal of Geophysical Research: Oceans, 116(12):1–14, 2011. ISSN 21699291. doi: 10.1029/2011JC007054.

37. C. Chen, G. Gao, Y. Zhang, R. C. Beardsley, Z. Lai, J. Qi, and H. Lin. Circulation in the Arctic Ocean: Results from a high-resolution coupled ice-sea nested Global-FVCOM and Arctic-FVCOM system. Progress in Oceanography, 141:60–80, 2016. ISSN 00796611. doi: 10.1016/j.pocean.2015.12.002. URL http://dx.doi.org/10.1016/j.pocean.2015.12.002.

38. M. A. Cialone, A. S. Grzegorzewski, D. J. Mark, M. A. Bryant, and T. C. Massey. Coastal-storm model development and water-level validation for the north atlantic coast comprehensive study. Journal of Waterway, Port, Coastal, and Ocean Engineering, 143(5):04017031, 2017. doi: 10.1061/(ASCE)WW.1943-5460.0000408.

39. A. J. Clarke and D. S. Battisti. The effect of continental shelves on tides. Deep Sea Research Part A. Oceanographic Research Papers, 28(7):665 – 682, 1981. ISSN 0198-0149. doi: 10.1016/0198-0149(81)90128-X.

40. B. Colle, M. Bowman, K. Roberts, M. Bowman, C. Flagg, J. Kuang, Y. Weng, E. Munsell, and F. Zhang. Exploring Water Level Sensitivity for Metropolitan New York during Sandy (2012) Using Ensemble Storm Surge Simulations. Journal of Marine Science and Engineering, 3(2):428–443, jun 2015. ISSN 2077-1312. doi: 10.3390/jmse3020428. URL `http://www.mdpi.com/2077-1312/3/2/428/`.

41. C. J. Conroy, E. J. Kubatko, and D. W. West. ADMESH: An advanced, automatic unstructured mesh generator for shallow water models. Ocean Dynamics, 62(10-12):1503–1517, dec 2012. ISSN 1616-7341. doi: 10.1007/s10236-012-0574-0. URL `http://link.springer.com/10.1007/s10236-012-0574-0`.

42. R. Cyriac, J. Dietrich, J. Fleming, B. Blanton, C. Kaiser, C. Dawson, and R. Luettich. Variability in Coastal Flooding predictions due to forecast errors during Hurricane Arthur. Coastal Engineering, 137:59 – 78, 2018. ISSN 0378-3839. doi: https://doi.org/10.1016/j.coastaleng.2018.02.008.

43. S. Dangendorf, M. Marcos, G. W'Oppelmann, C. P. Conrad, T. Frederikse, and R. Riva. Reassessment of 20th century global mean sea level rise. Proceedings of the National Academy of Sciences, 114(23):5946–5951, 2017. ISSN 0027-8424. doi: 10.1073/pnas.1616007114.

44. L. Debreu, P. Marchesiello, P. Penven, and G. Cambon. Two-way nesting in split-explicit ocean models: Algorithms, implementation and validation. Ocean Modelling, 49-50:1–21, 2012. ISSN 14635003. doi: 10.1016/j.ocemod.2012.03.003. URL `http://dx.doi.org/10.1016/j.ocemod.2012.03.003`.

45. K. Devine, E. Boman, R. Heaphy, B. Hendrickson, and C. Vaughan. Zoltan data management services for parallel dynamic applications. Computing in Science Engineering, 4(2):90–96, March 2002. ISSN 1521-9615. doi: 10.1109/5992.988653.

46. K. D. Devine, E. G. Boman, R. T. Heaphy, B. A. Hendrickson, J. D. Teresco, J. Faik, J. E. Flaherty, and L. G. Gervasio. New challenges in dynamic load balancing. Applied Numerical Mathematics, 52(2):133 – 152, 2005. ISSN 0168-9274. doi: https://doi.org/10.1016/j.apnum.2004.08.028. URL `http://www.`

sciencedirect.com/science/article/pii/S0168927404001631. ADAPT '03: Conference on Adaptive Methods for Partial Differential Equations and Large-Scale Computation.

47. J. Dietrich, R. Kolar, and R. Luettich. Assessment of adcirc's wetting and drying algorithm. In C. T. Miller and G. F. Pinder, editors, Computational Methods in Water Resources: Volume 2, volume 55 of Developments in Water Science, pages 1767 – 1778. Elsevier, 2004. doi: https://doi.org/10.1016/S0167-5648(04)80183-7. URL http://www.sciencedirect.com/science/article/pii/S0167564804801837.

48. J. Dietrich, M. Zijlema, J. Westerink, L. Holthuijsen, C. Dawson, R. Luettich, R. Jensen, J. Smith, G. Stelling, and G. Stone. Modeling hurricane waves and storm surge using integrally-coupled, scalable computations. Coastal Engineering, 58(1):45 – 65, 2011. ISSN 0378-3839. doi: https://doi.org/10.1016/j.coastaleng.2010.08.001.

49. J. C. Dietrich, R. L. Kolara, and R. A. Luettichb. Assessment of adcirc s wetting and drying algorithm. 2004.

50. J. C. Dietrich, S. Bunya, J. J. Westerink, B. A. Ebersole, J. M. Smith, J. H. Atkinson, R. Jensen, D. T. Resio, R. A. Luettich, C. Dawson, V. J. Cardone, A. T. Cox, M. D. Powell, H. J. Westerink, and H. J. Roberts. A high-resolution coupled riverine flow, tide, wind, wind wave, and storm surge model for southern louisiana and mississippi. part ii: Synoptic description and analysis of hurricanes katrina and rita. Monthly Weather Review, 138(2):378–404, 2010. doi: 10.1175/2009MWR2907.1. URL https://doi.org/10.1175/2009MWR2907.1.

51. J. C. Dietrich, J. J. Westerink, A. B. Kennedy, J. M. Smith, R. E. Jensen, M. Zijlema, L. H. Holthuijsen, C. Dawson, R. A. Luettich, M. D. Powell, V. J. Cardone, A. T. Cox, G. W. Stone, H. Pourtaheri, M. E. Hope, S. Tanaka, L. G. Westerink, H. J. Westerink, and Z. Cobell. Hurricane gustav (2008) waves and storm surge: Hindcast, synoptic analysis, and validation in southern louisiana. Monthly Weather Review, 139(8):2488–2522, 2011. doi: 10.1175/2011MWR3611.1.

52. J. C. Dietrich, S. Tanaka, J. J. Westerink, C. N. Dawson, R. A. Luettich, M. Zijlema, L. H. Holthuijsen, J. M. Smith, L. G. Westerink, and H. J. Westerink. Performance of the unstructured-mesh, swan+adcirc model in computing hurricane waves and surge. Journal of Scientific Computing, 52(2): 468–497, Aug 2012. ISSN 1573-7691. doi: 10.1007/s10915-011-9555-6. URL https://doi.org/10.1007/s10915-011-9555-6.

53. K. M. Dresback, J. G. Fleming, B. O. Blanton, C. Kaiser, J. J. Gourley, E. M. Tromble, R. A. Luettich, R. L. Kolar, Y. Hong, S. V. Cooten, H. J. Vergara, Z. L. Flamig, H. M. Lander, K. E. Kelleher, and K. L. Nemunaitis-Monroe. Skill assessment of a real-time forecast system utilizing a coupled hydrologic

and coastal hydrodynamic model during hurricane irene (2011). Continental Shelf Research, 71:78 – 94, 2013. ISSN 0278-4343. doi: https://doi.org/10.1016/j.csr.2013.10.007.

54. J. Dronkers. Tidal asymmetry and estuarine morphology. Netherlands Journal of Sea Research, 20(2):117 – 131, 1986. ISSN 0077-7579. doi: https://doi.org/10.1016/0077-7579(86)90036-0.

55. A. P. Engsig-Karup, J. S. Hesthaven, H. B. Bingham, and T. Warburton. DG-FEM solution for nonlinear wave-structure interaction using Boussinesq-type equations. Coastal Engineering, 55(3):197 – 208, 2008. ISSN 0378-3839. doi: 10.1016/j.coastaleng.2007.09.005. URL `http://www.sciencedirect.com/science/article/pii/S0378383907001081`.

56. D. Engwirda. Locally optimal Delaunay-refinement and optimisation-based mesh generation. PhD thesis, University of Sydney, 2014.

57. D. Engwirda. JIGSAW-GEO (1.0): Locally orthogonal staggered unstructured grid generation for general circulation modelling on the sphere. Geoscientific Model Development, 10(6):2117–2140, 2017. ISSN 19919603. doi: 10.5194/gmd-10-2117-2017.

58. A. Feldmann and L. Foschini. Balanced partitions of trees and applications. volume 71, pages 100–111, 02 2012. doi: 10.1007/s00453-013-9802-3.

59. J. Fleming, C. W. Fulcher, R. Luettich, Jr, B. D. Estrade, G. Allen, and H. S. Winer. A real time storm surge forecasting system using adcirc. Estuarine and Coastal Modeling X, 08 2008. doi: 10.1061/40990(324)48.

60. C. Forbes, R. A. Luettich, C. A. Mattocks, and J. J. Westerink. A retrospective evaluation of the storm surge produced by hurricane gustav (2008): Forecast and hindcast results. Weather and Forecasting, 25(6):1577–1602, 2010. doi: 10.1175/2010WAF2222416.1. URL `https://doi.org/10.1175/2010WAF2222416.1`.

61. C. T. Friedrichs. Barotropic tides in channelized estuaries, pages 27–61. Cambridge University Press, 2010. doi: 10.1017/CBO9780511676567.004.

62. O. Fringer, M. Gerritsen, and R. Street. An unstructured-grid, finite-volume, nonhydrostatic, parallel coastal ocean simulator. Ocean Modelling, 14(3):139 – 173, 2006. ISSN 1463-5003. doi: https://doi.org/10.1016/j.ocemod.2006.03.006.

63. C. Garrett and E. Kunze. Internal Tide Generation in the Deep Ocean. Annual Review of Fluid Mechanics, 39(1):57–87, jan 2007. ISSN 0066-4189. doi: 10.1146/annurev.fluid.39.050905.110227. URL `http://www.annualreviews.org/doi/10.1146/annurev.fluid.39.050905.110227`.

64. G. Gorman, M. Piggott, C. Pain, C. de Oliveira, A. Umpleby, and A. Goddard. Optimisation based bathymetry approximation through constrained unstructured mesh adaptivity. Ocean Modelling, 12(3):436 – 452, 2006. ISSN 1463-5003. doi: 10.1016/j.ocemod.2005.09.004. URL http://www.sciencedirect.com/science/article/pii/S1463500305000764.

65. G. Gorman, M. Piggott, M. Wells, C. Pain, and P. Allison. A systematic approach to unstructured mesh generation for ocean modelling using gmt and terreno. Computers and Geosciences, 34(12):1721–1731, 2008. ISSN 0098-3004. doi: 10.1016/j.cageo.2007.06.014. URL http://www.sciencedirect.com/science/article/pii/S0098300408001003.

66. G. J. Gorman, M. D. Piggott, and C. C. Pain. Shoreline approximation for unstructured mesh generation. Computers and Geosciences, 33:666–677, 2007. ISSN 00983004. doi: 10.1016/j.cageo.2006.09.007.

67. GRASS Development Team. Geographic Resources Analysis Support System (GRASS GIS) Software, Version 7.2. Open Source Geospatial Foundation, 2017. URL http://grass.osgeo.org.

68. J. A. M. Green and J. Nycander. A Comparison of Tidal Conversion Parameterizations for Tidal Models. Journal of Physical Oceanography, 43(1): 104–119, jan 2013. ISSN 0022-3670. doi: 10.1175/JPO-D-12-023.1. URL http://journals.ametsoc.org/doi/abs/10.1175/JPO-D-12-023.1.

69. D. A. Greenberg, F. Dupont, F. H. Lyard, D. R. Lynch, and F. E. Werner. Resolution issues in numerical models of oceanic and coastal circulation. Continental Shelf Research, 27(9):1317 – 1343, 2007. ISSN 0278-4343. doi: 10.1016/j.csr.2007.01.023. Recent Developments in Physical Oceanographic Modelling: Part IV.

70. L. A. Hageman and D. M. Young. Chapter 10 - the use of iterative methods in the solution of partial differential equations. In Applied Iterative Methods, pages 259 – 286. Academic Press, San Diego, 1981. ISBN 978-0-12-313340-3. doi: https://doi.org/10.1016/B978-0-12-313340-3.50015-7. URL http://www.sciencedirect.com/science/article/pii/B9780123133403500157.

71. S. C. Hagen, J. J. Westerink, and R. L. Kolar. One-dimensional finite element grids based on a localized truncation error analysis. International Journal for numerical methods in fluids, 32:241–261, 2000. doi: 10.1002/(SICI)1097-0363(20000130)32:2⟨241::AID-FLD947⟩3.0.CO;2-#.

72. S. C. Hagen, J. J. Westerink, R. L. Kolar, and O. Horstmann. Two-dimensional, unstructured mesh generation for tidal models. International Journal for Numerical Methods in Fluids, 35(6):669–686, 2001. ISSN 02712091. doi: 10.1002/1097-0363(20010330)35:6⟨669::AID-FLD108⟩3.0.CO;2-#.

73. S. C. Hagen, O. Horstmann, and R. J. Bennett. An unstructured mesh generation algorithm for shallow water modeling. International Journal of Computational Fluid Dynamics, 16(2):83–91, 2002. doi: 10.1080/10618560290017176. URL `https://doi.org/10.1080/10618560290017176`.

74. C. Hannah and D. Wright. Depth dependent analytical and numerical solutions for winddriven flow in the coastal ocean. Quantitative Skill Assessment for Coastal Ocean Models, 47:125–152, 1995.

75. T. J. Heinzer, M. D. Williams, E. C. Dogrul, T. N. Kadir, C. F. Brush, and F. I. Chung. Implementation of a feature-constraint mesh generation algorithm within a gis. Computers and Geosciences, 49:46 – 52, 2012. ISSN 0098-3004. doi: 10.1016/j.cageo.2012.06.004. URL `http://www.sciencedirect.com/science/article/pii/S0098300412001999`.

76. M. C. Hendershott. The Effects of Solid Earth Deformation on Global Ocean Tides. Geophysical Journal International, 29(4):389–402, oct 1972. ISSN 0956-540X. doi: 10.1111/j.1365-246X.1972.tb06167.x. URL `https://academic.oup.com/gji/article-lookup/doi/10.1111/j.1365-246X.1972.tb06167.x`.

77. B. Hendrickson and T. G. Kolda. Graph partitioning models for parallel computing. Parallel Computing, 26(12):1519–1534, November 2000. doi: 10.1016/S0167-8191(00)00048-X.

78. M. E. Hope, J. J. Westerink, A. B. Kennedy, P. C. Kerr, J. C. Dietrich, C. Dawson, C. J. Bender, J. M. Smith, R. E. Jensen, M. Zijlema, L. H. Holthuijsen, R. A. Luettich Jr., M. D. Powell, V. J. Cardone, A. T. Cox, H. Pourtaheri, H. J. Roberts, J. H. Atkinson, S. Tanaka, H. J. Westerink, and L. G. Westerink. Hindcast and validation of hurricane ike (2008) waves, forerunner, and storm surge. Journal of Geophysical Research: Oceans, 118(9):4424–4460, 2013. doi: 10.1002/jgrc.20314.

79. J. M. Huthnance. Circulation, exchange and water masses at the ocean margin: the role of physical processes at the shelf edge. Progress in Oceanography, 35 (4):353 – 431, 1995. ISSN 0079-6611. doi: 10.1016/0079-6611(95)80003-C. URL `http://www.sciencedirect.com/science/article/pii/007966119580003C`.

80. B. Joyce, J. Gonzalez-Lopez, A. J. van der Westhuysen, D. Yang, W. J. Pringle, J. J. Westerink, and A. T. Cox. U.S. IOOS Coastal and Ocean Modeling Testbed: Hurricane-induced Winds, Waves and Surge for Deep-ocean, Reef Fringed Islands in the Caribbean. Journal of Geophysical Research C: Oceans, 2019. doi: 10.1029/2018JC014687.

81. L. V. Kale and S. Krishnan. Charm++: A portable concurrent object oriented system based on c++. SIGPLAN Not., 28(10):91–108, Oct. 1993. ISSN 0362-1340. doi: 10.1145/167962.165874. URL `http://doi.acm.org/10.1145/167962.165874`.

82. J. Kappraff. The geometry of coastlines: a study in fractals. Computers Mathematics with Applications, 12(3, Part 2):655 – 671, 1986. ISSN 0898-1221. doi: https://doi.org/10.1016/0898-1221(86)90417-7. URL http://www.sciencedirect.com/science/article/pii/0898122186904177.

83. G. Karypis and V. Kumar. A parallel algorithm for multilevel graph partitioning and sparse matrix ordering. Journal of Parallel and Distributed Computing, 48(1):71 – 95, 1998. ISSN 0743-7315. doi: https://doi.org/10.1006/jpdc.1997.1403. URL http://www.sciencedirect.com/science/article/pii/S0743731597914039.

84. A. B. Kennedy, J. C. Dietrich, and J. J. Westerink. The surge standard for "events of katrina magnitude". Proceedings of the National Academy of Sciences, 110(29):E2665–E2666, 2013. ISSN 0027-8424. doi: 10.1073/pnas.1305960110.

85. P. C. Kerr, R. C. Martyr, A. S. Donahue, M. E. Hope, J. J. Westerink, R. A. Luettich, A. B. Kennedy, J. C. Dietrich, C. Dawson, and H. J. Westerink. U.S. IOOS coastal and ocean modeling testbed: Evaluation of tide, wave, and hurricane surge response sensitivities to mesh resolution and friction in the Gulf of Mexico. Journal of Geophysical Research: Oceans, 118(9):4633–4661, sep 2013. ISSN 21699275. doi: 10.1002/jgrc.20305. URL http://doi.wiley.com/10.1002/jgrc.20305.

86. I. Kinnmark. The shallow water wave equations: Formulation, analysis and application (i. kinnmark). SIAM Review, 30(3):517–518, 1988. doi: 10.1137/1030116. URL https://doi.org/10.1137/1030116.

87. J. Koko. A Matlab mesh generator for the two-dimensional finite element method. Applied Mathematics and Computation, 250:650–664, 2015. ISSN 00963003. doi: 10.1016/j.amc.2014.11.009. URL http://dx.doi.org/10.1016/j.amc.2014.11.009.

88. S. K. Kumar, K. Schloegel, G. Karypis, V. K. R. Biswas, and L. Oliker. A performance study of diffusive vs. remapped load-balancing schemes. In ISCA 11th Intl. Conf. on Parallel and Distributed Computing Systems, pages 59–66, 1998.

89. J. Lambrechts, R. Comblen, V. Legat, C. Geuzaine, and J.-F. Remacle. Multiscale mesh generation on the sphere. Ocean Dynamics, 58(5-6):461–473, dec 2008. ISSN 1616-7341. doi: 10.1007/s10236-008-0148-3. URL http://link.springer.com/10.1007/s10236-008-0148-3.

90. C. Le Provost and F. Lyard. Energetics of the M2 barotropic ocean tides: an estimate of bottom friction dissipation from a hydrodynamic model. Progress in Oceanography, 40(1):37–52, 1997. ISSN 00796611. doi: 10.1016/S0079-6611(97)00022-0.

91. P. H. LeBlond. Tides and their Interactions with Other Oceanographic Phenomena in Shallow Water (Review). In B. B. Parker, editor, Tidal hydrodynamics, pages 357–378. John Wiley & Sons, Inc., New York, USA, 1991.

92. R. LeVeque, D. L. George, and M. J. Berger. Adaptive mesh refinement techniques for tsunamis and other geophysical flows over topography. Acta Numerica, pages 211–289, 2011.

93. R. J. LeVeque, D. L. George, and M. J. Berger. Tsunami modelling with adaptively refined finite volume methods. Acta Numerica, 20:211289, 2011. doi: 10.1017/S0962492911000043.

94. J. Liu. Open and traction boundary conditions for the incompressible navier-stokes equations. Journal of Computational Physics, 228(19):7250 – 7267, 2009. ISSN 0021-9991. doi: 10.1016/j.jcp.2009.06.021. URL `http://www.sciencedirect.com/science/article/pii/S0021999109003453`.

95. J. W. Loder. Topographic Rectification of Tidal Currents on the Sides of Georges Bank. Journal of Physical Oceanography, 10(9):1399–1416, 1980. doi: 10.1175/1520-0485(1980)010⟨1399:TROTCO⟩2.0.CO;2.

96. R. Luettich and J. J. Westerink. Formulation and Numerical Implementation of the 2D/3D ADCIRC Finite Element Model Version 44.XX. Technical report, 2004.

97. R. Luettich, Jr. and J. J. Westerink. Elemental wetting and drying in the adcirc hydrodynamic model: Upgrades and documentation for adcirc version 34.xx. 01 1999.

98. R. Luettich, Jr and J. J. Westerink. Continental shelf scale convergence studies with a barotropic tidal model. Quantitative Skill Assessment for Coastal Ocean Models, A.G.U., 47, 01 1995. doi: 10.1029/CE047p0349.

99. F. Lyard, F. Lefevre, T. Letellier, and O. Francis. Modelling the global ocean tides: modern insights from FES2004. Ocean Dynamics, 56(5-6):394–415, dec 2006. ISSN 1616-7341. doi: 10.1007/s10236-006-0086-x. URL `http://link.springer.com/10.1007/s10236-006-0086-x`.

100. D. R. Lynch and W. G. Gray. A wave equation model for finite element tidal computations. Computers Fluids, 7(3):207 – 228, 1979. ISSN 0045-7930. doi: https://doi.org/10.1016/0045-7930(79)90037-9. URL `http://www.sciencedirect.com/science/article/pii/0045793079900379`.

101. R. Marsooli and N. Lin. Numerical Modeling of Historical Storm Tides and Waves and Their Interactions Along the U.S. East and Gulf Coasts. Journal of Geophysical Research: Oceans, pages 3844–3874, 2018. ISSN 21699291. doi: 10.1029/2017JC013434.

102. T. C. Massey. Locally constrained nodal connectivity refinement procedures for unstructured triangular finite element meshes. pages 375–386, 2015. doi: 10.1007/s00366-014-0357-y.

103. S. C. Medeiros and S. C. Hagen. Review of wetting and drying algorithms for numerical tidal flow models. International Journal for Numerical Methods in Fluids, 71(4):473–487, 2012. doi: 10.1002/fld.3668.

104. H. R. Moftakhari, G. Salvadori, A. AghaKouchak, B. F. Sanders, and R. A. Matthew. Compounding effects of sea level rise and fluvial flooding. Proceedings of the National Academy of Sciences, 114(37):9785–9790, 2017. ISSN 0027-8424. doi: 10.1073/pnas.1620325114.

105. J. Molines, M. Fornerino, and C. L. Provost. Tidal spectroscopy of a coastal area: observed and simulated tides of the lake maracaibo system. Continental Shelf Research, 9(4):301 – 323, 1989. ISSN 0278-4343. doi: https://doi.org/10.1016/0278-4343(89)90036-8.

106. D. M. Mount and S. Arya. Ann: A library for approximate nearest neighbor searching, August 2006. version 1.1.1, available at `http://www.cs.umd.edu/~mount/ANN/`.

107. S. Muis, N. Lin, M. Verlaan, H. C. Winsemius, P. J. Ward, and J. C. J. H. Aerts. Spatiotemporal patterns of extreme sea levels along the western north-atlantic coasts. 9(1):3391. ISSN 2045-2322. doi: 10.1038/s41598-019-40157-w.

108. S. Muis, N. Lin, M. Verlaan, H. C. Winsemius, P. J. Ward, and J. C. J. H. Aerts. Spatiotemporal patterns of extreme sea levels along the western North-Atlantic coasts. Scientific Reports, 9(1):3391, 2019. ISSN 2045-2322. doi: 10.1038/s41598-019-40157-w. URL `http://www.nature.com/articles/s41598-019-40157-w`.

109. R. S. Nerem, B. D. Beckley, J. T. Fasullo, B. D. Hamlington, D. Masters, and G. T. Mitchum. Climate-change driven accelerated sea-level rise detected in the altimeter era. Proceedings of the National Academy of Sciences, 115(9):2022–2025, 2018. ISSN 0027-8424. doi: 10.1073/pnas.1717312115.

110. V.-T. Nguyen, J. Peraire, B. C. Khoo, and P.-O. Persson. A discontinuous galerkin front tracking method for two-phase flows with surface tension. Computers & Fluids, 39(1):1 – 14, 2010. ISSN 0045-7930. doi: 10.1016/j.compfluid.2009.06.007. URL `http://www.sciencedirect.com/science/article/pii/S0045793009000899`.

111. R. J. Nicholls and A. Cazenave. Sea-level rise and its impact on coastal zones. Science, 328(5985):1517–1520, 2010. ISSN 0036-8075. doi: 10.1126/science.1185782.

112. J. F. O'Callaghan and D. M. Mark. The extraction of drainage networks from digital elevation data. Computer Vision, Graphics, and Image Processing, 28 (3):323 – 344, 1984. ISSN 0734-189X. doi: 10.1016/S0734-189X(84)80011-0.

113. J. Oden, L. Demkowicz, W. Rachowicz, and T. Westermann. A posteriori error analysis in finite elements: The element residual method for symmetrizable problems with applications to compressible euler and navier-stokes equations. Computer Methods in Applied Mechanics and Engineering, 82(1):183 – 203, 1990. ISSN 0045-7825. doi: https://doi.org/10.1016/0045-7825(90)90164-H. Proceedings of the Workshop on Reliability in Computational Mechanics.

114. B. B. Parker. The relative importance of the various nonlinear mechanisms in a wide range of tidal interactions (Review). In B. B. Parker, editor, Tidal hydrodynamics, pages 237–268. John Wiley & Sons, Inc., New York, USA, 1991.

115. D. M. Parrish and S. C. Hagen. Incorporating spatially variable bottom stress and Coriolis force into 2D, a posteriori , unstructured mesh generation for shallow water models. International Journal for Numerical Methods in Fluids, 60(3):237–261, may 2009. ISSN 02712091. doi: 10.1002/fld.1882. URL `http://doi.wiley.com/10.1002/fld.1882`.

116. R. Pawlowicz. M_Map: A mapping package for MATLAB, version 1.4j. `www.eoas.ubc.ca/~rich/map.html`, 2018.

117. F. Pellegrini and J. Roman. Scotch: A software package for static mapping by dual recursive bipartitioning of process and architecture graphs. In H. Liddell, A. Colbrook, B. Hertzberger, and P. Sloot, editors, High-Performance Computing and Networking, pages 493–498. Springer Berlin Heidelberg, 1996. ISBN 978-3-540-49955-8.

118. P.-o. Persson and G. Strang. A Simple Mesh Generator in MATLAB. SIAM Rev., 46, 2004. doi: 10.1137/S0036144503429121. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.84.7905`.

119. M. D. Piggott, P. E. Farrell, C. R. Wilson, G. J. Gorman, and C. C. Pain. Anisotropic mesh adaptivity for multi-scale ocean modelling. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 367(1907):4591–4611, 2009. ISSN 1364503X. doi: 10.1098/rsta.2009.0155.

120. A. Pinar and B. Hendrickson. Communication support for adaptive computation, 2001.

121. D. Prandle. Relationships between tidal dynamics and bathymetry in strongly convergent estuaries. Journal of Physical Oceanography, 33(12):2738–2750, 2003. doi: 10.1175/1520-0485(2003)033⟨2738:RBTDAB⟩2.0.CO;2.

122. W. J. Pringle, D. Wirasaet, A. Suhardjo, J. Meixner, J. J. Westerink, A. B. Kennedy, and S. Nong. Finite-element barotropic model for the indian and western pacific oceans: Tidal model-data comparisons and sensitivities. Ocean Modelling, 129:13 – 38, 2018. ISSN 1463-5003. doi: https://doi.org/10.1016/j.ocemod.2018.07.003.

123. W. J. Pringle, D. Wirasaet, and J. J. Westerink. Modifications to Internal Tide Conversion Parameterizations and Implementation into Barotropic Ocean Models. EarthArXiv, page 9, 2018. doi: 10.31223/osf.io/84w53.

124. W. J. Pringle, N. Yoneyama, and N. Mori. Multiscale coupled three-dimensional model analysis of the tsunami flow characteristics around the Kamaishi Bay offshore breakwater and comparisons to a shallow water model. Coastal Engineering Journal, 60(2):200–224, 2018. ISSN 2166-4250. doi: 10.1080/21664250.2018.1484270. URL `https://www.tandfonline.com/doi/full/10.1080/21664250.2018.1484270`.

125. W. J. Pringle, J. Gonzalez-lopez, B. Joyce, J. J. Westerink, and A. J. van der Westhuysen. Baroclinic Coupling Improves Depth-Integrated Modeling of Coastal Sea Level Variations around Puerto Rico and the U.S . Virgin Islands. Journal of Geophysical Research: Oceans, 2019. doi: 10.1029/2018JC014682.

126. J.-F. Remacle and J. Lambrechts. Fast and Robust Mesh Generation on the Sphere Application to Coastal Domains. Procedia Engineering, 163:20–32, 2016. ISSN 18777058. doi: 10.1016/j.proeng.2016.11.011.

127. D. Resio and J. Westerink. Modelling the physics of storm surges. Physics Today, 61:33, 2008. doi: https://doi.org/10.1063/1.2982120.

128. P. J. Roache. Perspective: A Method for Uniform Reporting of Grid Refinement Studies. Journal of Fluids Engineering, 116(3):405, sep 1994. ISSN 00982202. doi: 10.1115/1.2910291. URL `http://fluidsengineering.asmedigitalcollection.asme.org/article.aspx?articleid=1427780`.

129. K. J. Roberts and W. J. Pringle. OceanMesh2D: User guide - Precise distance-based two-dimensional automated mesh generation. Technical Report June, University of Notre Dame, 2018.

130. K. J. Roberts, B. A. Colle, and N. Korfe. Impact of simulated twenty-first-century changes in extratropical cyclones on coastal flooding at the battery, new york city. Journal of Applied Meteorology and Climatology, 56(2):415–432, 2017. doi: 10.1175/JAMC-D-16-0088.1.

131. K. J. Roberts, W. J. Pringle, and J. J. Westerink. OceanMesh2D 1.0: MATLAB-based software for two-dimensional unstructured mesh generation in coastal ocean modeling. Geoscientific Model Development Discussions, page in review, 2018. doi: 10.5194/gmd-2018-203.

132. D. T. Sandwell, J. J. Becker, C. Olson, and A. Jackson. SRTM15_PLUS: Data Fusion of SRTM Land Topography with Measured and Estimated Seafloor topography, 2014. URL ftp://topex.ucsd.edu/pub/srtm15{_}plus/.

133. K. Schloegel, G. Karypis, and V. Kumar. Multilevel diffusion schemes for repartitioning of adaptive meshes. Journal of Parallel and Distributed Computing, 47(2):109 – 124, 1997. ISSN 0743-7315. doi: https://doi.org/10.1006/jpdc.1997.1410. URL http://www.sciencedirect.com/science/article/pii/S0743731597914106.

134. A. Sebastian, J. Proft, J. C. Dietrich, W. Du, P. B. Bedient, and C. N. Dawson. Characterizing hurricane storm surge behavior in Galveston Bay using the SWAN+ADCIRC model. Coastal Engineering, 88:171–181, 6 2014. ISSN 03783839. doi: 10.1016/j.coastaleng.2014.03.002. URL http://linkinghub.elsevier.com/retrieve/pii/S0378383914000556.

135. D. V. Sein, S. Danilov, A. Biastoch, J. V. Durgadoo, D. Sidorenko, S. Harig, and Q. Wang. Designing variable ocean model resolution based on the observed ocean variability. Journal of Advances in Modeling Earth Systems, 8:904–916, 2016. doi: 10.1002/2016MS000650.

136. D. V. Sein, N. V. Koldunov, S. Danilov, Q. Wang, D. Sidorenko, I. Fast, T. Rackow, W. Cabos, and T. Jung. Ocean Modeling on a Mesh With Resolution Following the Local Rossby Radius. Journal of Advances in Modeling Earth Systems, 9:2601–2614, 2017. ISSN 19422466. doi: 10.1002/2017MS001099.

137. G. Seroka, T. Miles, Y. Xu, J. Kohut, O. Schofield, and S. Glenn. Hurricane irene sensitivity to stratified coastal ocean cooling. Monthly Weather Review, 144(9):3507–3530, 2016. doi: 10.1175/MWR-D-15-0452.1.

138. J. R. Shewchuk. What is a good linear finite element? - interpolation, conditioning, anisotropy, and quality measures. Technical report, In Proc. of the 11th International Meshing Roundtable, 2002.

139. D. Stammer, R. D. Ray, O. B. Andersen, B. K. Arbic, W. Bosch, L. Carrère, Y. Cheng, D. S. Chinn, B. D. Dushaw, G. D. Egbert, S. Y. Erofeeva, H. S. Fok, J. A. M. Green, S. Griffiths, M. A. King, V. Lapin, F. G. Lemoine, S. B. Luthcke, F. Lyard, J. Morison, M. Müller, L. Padman, J. G. Richman, J. F. Shriver, C. K. Shum, E. Taguchi, and Y. Yi. Accuracy assessment of global barotropic ocean tide models. Reviews of Geophysics, 52(3):243–282, sep 2014. ISSN 87551209. doi: 10.1002/2014RG000450. URL http://doi.wiley.com/10.1002/2014RG000450.

140. J. Staneva, K. Wahle, H. Günther, and E. Stanev. Coupling of wave and circulation models in coastal–ocean
predicting systems: a case study for the german bight. Ocean Science, 12(3): 797–806, 2016. doi: 10.5194/os-12-797-2016.

141. S.-W. Suh and H.-J. Kim. Simulation of wave overtopping and inundation over a dike caused by typhoon chaba at marine city, busan, korea. Journal of Coastal Research, pages 711–715, 2018. doi: 10.2112/SI85-143.1.

142. B. K. Sullivan. The most expensive u.s. hurricane season ever: By the numbers. November 2017.

143. C. Szpilka, K. Dresback, R. Kolar, J. Feyen, and J. Wang. Improvements for the Western North Atlantic, Caribbean and Gulf of Mexico ADCIRC Tidal Database (EC2015). Journal of Marine Science and Engineering, 4(4), 2016. ISSN 2077-1312. doi: 10.3390/jmse4040072.

144. P. Taeb and R. J. Weaver. An operational coastal forecasting tool for performing ensemble modeling. Estuarine, Coastal and Shelf Science, 217:237 – 249, 2019. ISSN 0272-7714. doi: https://doi.org/10.1016/j.ecss.2018.09.020. URL `http://www.sciencedirect.com/science/article/pii/S0272771418302956`.

145. S. Tanaka, S. Bunya, J. J. Westerink, C. Dawson, and R. A. Luettich. Scalability of an Unstructured Grid Continuous Galerkin Based Hurricane Storm Surge Model. Journal of Scientific Computing, 46(3):329–358, 3 2011. ISSN 0885-7474. doi: 10.1007/s10915-010-9402-1. URL `http://link.springer.com/10.1007/s10915-010-9402-1`.

146. Technology Riverside Inc. and AECOM. Mesh Development, Tidal Validation, and Hindcast Skill Asessment of an ADCIRC Model for the Hurricane Storm Surge Operational Forecast System on the US Gulf-Atlantic Coast. Technical report, National Oceanic and Atmospheric Administration/Nation Ocean Service, Coast Survey Development Laboratory, Office of Coast Survey, 2015.

147. K. M. Thyng, C. A. Greene, R. D. Hetland, H. M. Zimmerle, and S. F. DiMarco. True colors of oceanography: Guidelines for effective and accurate colormap selection. Oceanography, 29(3):9–13, 2016. doi: 10.5670/oceanog.2016.66.

148. H. Wang, J. Loftis, Z. Liu, D. Forrest, and J. Zhang. The Storm Surge and Sub-Grid Inundation Modeling in New York City during Hurricane Sandy. Journal of Marine Science and Engineering, 2(1):226–246, mar 2014. ISSN 2077-1312. doi: 10.3390/jmse2010226. URL `http://www.mdpi.com/2077-1312/2/1/226/`.

149. Q. Wang, S. Danilov, D. Sidorenko, R. Timmermann, C. Wekerle, X. Wang, T. Jung, and J. Schröter. The finite element sea ice-ocean model (fesom) v.1.4: formulation of an ocean general circulation model. Geoscientific Model Development, 7(2):663–693, 2014. doi: 10.5194/gmd-7-663-2014. URL `https://www.geosci-model-dev.net/7/663/2014/`.

150. P. Wessel and W. H. F. Smith. A global, self-consistent, hierarchical, high-resolution shoreline database. Journal of Geophysical Research: Solid Earth, 101(B4):8741–8743, apr 1996. ISSN 01480227. doi: 10.1029/96JB00104. URL `http://doi.wiley.com/10.1029/96JB00104`.

151. J. J. Westerink, R. A. Luettich, A. M. Baptists, N. W. Scheffner, and P. Farrar. Tide and Storm Surge Predictions Using Finite Element Model. Journal of Hydraulic Engineering, 118(10): 1373–1390, oct 1992. ISSN 0733-9429. doi: 10.1061/(ASCE) 0733-9429(1992)118:10(1373). URL `http://ascelibrary.org/doi/10.1061/{%}28ASCE{%}290733-9429{%}281992{%}29118{%}3A10{%}281373{%}29`.

152. J. J. Westerink, R. A. Luettich, and J. Muccino. Modelling tides in the western North Atlantic using unstructured graded grids. Tellus A, 46(2):178–199, 1994. ISSN 16000870. doi: 10.1034/j.1600-0870.1994.00007.x.

153. J. J. Westerink, R. A. Luettich, J. C. Feyen, J. H. Atkinson, C. Dawson, H. J. Roberts, M. D. Powell, J. P. Dunion, E. J. Kubatko, and H. Pourtaheri. A Basin-to Channel-Scale Unstructured Grid Hurricane Storm Surge Model Applied to Southern Louisiana. Monthly Weather Review, 136(3):833–864, mar 2008. ISSN 0027-0644. doi: 10.1175/2007MWR1946.1. URL `http://journals.ametsoc.org/doi/abs/10.1175/2007MWR1946.1`.

154. S. White and K. Hess. An Assessment of the Revised VDatum for Eastern Florida, Georgia, South Carolina, and North Carolina. NOAA Technical Memorandum NOS CS 38., 2016.

155. S. J. Williams. Sea-level rise implications for coastal regions. Journal of Coastal Research, pages 184–196, 2013. doi: 10.2112/SI63-015.1.

156. D.-m. Xie, Q.-p. Zou, and J. W. Cannon. Application of SWAN+ADCIRC to tide-surge and wave simulation in Gulf of Maine during Patriot's Day storm. Water Science and Engineering, 9(1):33–41, 2016. ISSN 16742370. doi: 10.1016/j.wse.2016.02.003.

157. J. Xing and A. M. Davies. A three-dimensional model of internal tides on the Malin-Hebrides shelf and shelf edge. Journal of Geophysical Research: Oceans, 103(C12):27821–27847, 1998. doi: 10.1029/98JC02149.

158. Q. Xu, C. Chen, J. Qi, H. Lin, R. C. Beardsley, and Y. Sun. Impact of current-wave interaction on storm surge simulation: A case study for Hurricane Bob. Journal of Geophysical Research: Oceans, 118(5):2685–2701, 2013. doi: 10.1002/jgrc.20207.

159. Y. Zhang and A. M. Baptista. SELFE: A semi-implicit Eulerian-Lagrangian finite-element model for cross-scale ocean circulation. Ocean Modelling, 21(3-4):71–96, 2008. ISSN 14635003. doi: 10.1016/j.ocemod.2007.11.005.

160. Y. J. Zhang, F. Ye, E. V. Stanev, and S. Grashorn. Seamless cross-scale modeling with SCHISM. Ocean Modelling, 102:64–81, 2016. ISSN 14635003. doi: 10.1016/j.ocemod.2016.05.002.

161. L. Zheng, R. H. Weisberg, Y. Huang, R. A. Luettich, J. J. Westerink, P. C. Kerr, A. S. Donahue, G. Crane, and L. Akli. Implications from the comparisons between two- and three- dimensional model simulations of the Hurricane Ike storm surge. Journal of Geophysical Research C: Oceans, 118:3350–3369, 2013. doi: 10.1002/jgrc.20248.

162. A. K. Zundel. Users manual for the surface-water modelling system, version 9.0. 2005.

This document was prepared & typeset with pdfLaTeX*, and formatted with* NDdiss2$_{\varepsilon}$ *classfile (v3.2013[2013/04/16]) provided by Sameer Vijay and updated by Megan Patnott.*