Environmental Modelling and Software 140 (2021) 105045

Contents lists available at ScienceDirect

Environmental Modelling and Software

journal homepage: http://www.elsevier.com/locate/envsoft

Dynamic load balancing for predictions of storm surge and coastal flooding

Keith J. Roberts [a,1,*], J. Casey Dietrich [b], Damrongsak Wirasaet [a], William J. Pringle [a], Joannes J. Westerink [a]

[a] Dept. of Civil and Environmental Engineering and Earth Sciences, University of Notre Dame, 156 Fitzpatrick Hall, Notre Dame, IN, USA
[b] Dept. of Civil, Construction and Environmental Engineering, North Carolina State University, Mann Hall, USA

ARTICLE INFO

Keywords:
Storm surge
Coastal flooding
Dynamic load balancing
Finite element modeling
Zoltan toolkit
ParMETIS

ABSTRACT

As coastal circulation models have evolved to predict storm-induced flooding, they must include progressively more overland regions that are normally dry, to where now it is possible for more than half of the domain to be needed in none or only some of the computations. While this evolution has improved real-time forecasting and long-term mitigation of coastal flooding, it poses a problem for parallelization in an HPC environment, especially for static paradigms in which the workload is balanced only at the start of the simulation. In this study, a dynamic rebalancing of computational work is developed for a finite-element-based, shallow-water, ocean circulation model of extensive overland flooding. The implementation has a low overhead cost, and we demonstrate a realistic hurricane-forced coastal flooding simulation can achieve peak speed-ups near 45% over the static case, thus operating now at $_{80-90}$% efficiency.

1. Software availability

The code, instructions, and a test to run the software detailed in this work can be found by this private link: https://figshare.com/s/41827afadc318047e2ea. The ADvanced Hydrodynamic CIRCulation (ADCIRC) code was originally developed in FORTRAN77 by Joannes Westerink, Rick Luettich, and Clint Dawson. Since then it has been continuously updated using FORTRAN90 and FORTRAN77 and maintained by a community of developers. ADCIRC is a commercial code (www.adcirc.org) that is otherwise free for research and academic purposes. ADCIRC + DLB, which is a branch of ADCIRC, requires a Linux-based system with distributed memory architecture that uses the message passing interface. ADCIRC + DLB must be compiled with an Intel Fortran 16.0 or more recent Intel Fortran compiler), either MPICH or MVAPICH, and it requires the Zoltan Toolkit part of the Trilinos package https://cs.sandia.gov/Zoltan/and ParMETIS http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview.

2. Introduction

In two-dimensional (2D), 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. To obtain a high fidelity model while keeping computing cost relatively low, an unstructured mesh composed of elements with highly-variable sizes is used to represent the large horizontal scale separation (i.e., $\mathcal{O}$(10 m)-$\mathcal{O}$(100 km)) of tide, storm surge, and coastal flooding processes (Gorman et al., 2008; Marks et al., 2017; Roberts et al., 2019b). Predictions with these unstructured meshes are used to provide crucial information for coastal hazard assessment and design (e.g., CSTORM Project; Cialone et al., 2017; Quetzalcóatl et al., 2019), water level guidance (e.g., iFLOOD; Khalid and Ferreira, 2020) and emergency management operations (Staneva et al., 2016; Blanton et al., 2012, 2018; Dresback et al., 2013).

To simulate coastal flooding in the region of focus, the mesh often contains an extensive floodplain extending up to an elevation of 10 m–15 m above local mean sea level (Fig. 1; Bunya et al., 2010; Blanton et al., 2012; Teng et al., 2017). On the floodplain, high-resolution elements of size 10–50 m enable a representation of the fine horizontal geometric length scales and complex land cover variability that control coastal inundation patterns (e.g., Dietrich et al., 2010; Forbes et al., 2010; Hope et al., 2013; Sebastian et al., 2014). Some works demonstrate that relatively coarse mesh resolution (e.g., >50 m overland) may result in inaccurate inundation area (Bilskie and Hagen, 2013; Bilskie et al., 2015; Cobell et al., 2013) and inaccurate velocities (Neelz and Pender, 2008) as coarser mesh resolution may alias vertical features

* Corresponding author.
E-mail addresses: krober@usp.br, keithrbt0@gmail.com (K.J. Roberts).
[1] Present address: Escola Politécnica, Dept. Mechanical Engineering, Av. Professor Mello Moraes, 2231, Cidade Universitária, São Paulo - SP, Brazil.

https://doi.org/10.1016/j.envsoft.2021.105045
Accepted 25 March 2021
Available online 2 April 2021

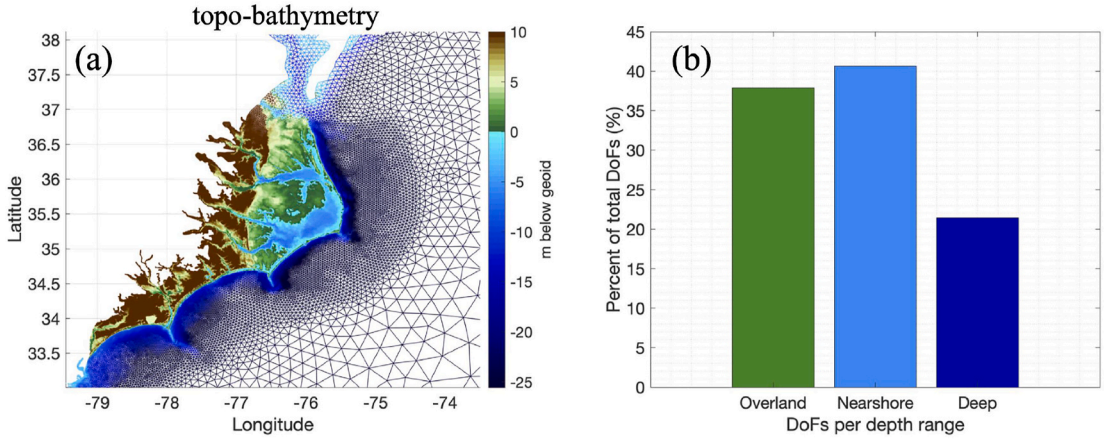

Fig. 1. (a) is an illustration of a high-resolution finite element mesh used in the computation of coastal flooding. In (b), a histogram illustrates the number of vertices otherwise referred to as degrees-of-freedom (DoF) per bathymetric depth range below sea level in the domain shown in (a). For the purpose of this figure, the overland category is classified as vertices with elevations higher than 0.10 m above local mean sea level, nearshore is vertices with elevations less than or equal to elevation of 0.10 m and greater than a elevation $_{-50}$ m, and "deep" is less than or equal vertices with elevations of $_{-50}$ m.

such as elevated roadways and levees that can largely alter inundation patterns. As a consequence of the high-resolution elements that are used to represent overland features, an often disproportionate number of degrees-of-freedom (DoFs) in the modeling system become located overland. For example, in a modeling system used for operational coastal flooding predictions (Fleming et al., 2008; Technology Riverside Inc. and Aecom, 2015), approximately 55% percent of the vertices are above the LMSL state at initialization. The transient nature of coastal flooding further implies 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 a broad range of numerical strategies (Chen et al., 2003; Casulli, 2019; Behrens and Bader, 2009; LeVeque et al., 2011a; Taeb and Weaver, 2019; Androsov et al., 2019; Wittmann et al., 2017; Ginting and Mundani, 2019; Sanders et al., 2010). Traditionally, coastal flooding is modeled using a pre-determined highly refined mesh with a logic based wetting/drying approach (Medeiros and Hagen, 2012a; Luettich and J. Westerink, 1999; Candy, 2017; Sanders et al., 2010). Another approach is to start the calculation with an initially coarse mesh and then adaptively refine the mesh as the event propagates (LeVeque et al., 2011b). Adaptive methods perform well for the simulation of transoceanic tsunamis, storm surges, and the associated coastal flooding, in which the background state is quiescent before the event occurs (Behrens and Bader, 2009; LeVeque et al., 2011a). However, for predictions of total water levels, the background tide and wind-driven signal continuously interact complicating mesh refinement strategies. Multi-grid approaches (e.g., Taeb and Weaver, 2019) also have been shown to reduce wall-clock times for detailed coastal flooding simulations by simulating with a hierarchy of meshes, but require an extensive mesh development and coupling approaches to merge solutions. Instead of using a pre-determined sufficiently fine mesh, sub-grid scale modeling techniques (e.g., Casulli and Walters, 2000; Casulli, 2019; Candy, 2017; Kennedy et al., 2019) aim at improving the accuracy of an under-resolved mesh by incorporating these geometries through porosity functions. This methodology is promising but there are limitations on the practical ability to provide closures for all subgrid physics (Kennedy et al., 2019).

The topic of dynamic load balancing has also been applied for the prediction of overland flooding to improve computational efficiency (Ginting and Mundani, 2019; Sanders et al., 2010; Wittmann et al., 2017; Ginting et al., 2020). Dynamic load balancing in the context of flooding is important for computational efficiency as the computational costs of wet and dry cells are often nonequivalent (Wittmann et al., 2017; Ginting and Mundani, 2019; Ginting et al., 2020) and inherently time varying as a flood event progresses (Sanders et al., 2010). One strategy presented in Wittmann et al. (2017) was to vectorize the flux calculation and mask out the dry-cell calculations resulting from wetting and drying in a shared memory parallelism finite volume solver. Ginting and Mundani (2019) using a shared-memory parallelism finite volume approach, with a space-filling curve to adaptively distribute meshes with a weighted dynamic load balancing strategy. Ginting et al. (2020) further advanced by performing dynamic load balancing with a static domain decomposition in a hybrid OpenMP/MPI parallel computing environment. A finite volume solver called ParBreZo with domain decomposition MPI parallelism showed a 97% reduction in execution time in a simulation of regional hurricane-driven storm surge on an unstructured grid by using a weighted domain decomposition approach and a static grid partitioning technique (Sanders et al., 2010).

By considering the challenges and recognizing the complexity involved in mesh design for coastal ocean modeling (Bilgili et al., 2006; Gorman et al, 2007, 2008; Roberts et al., 2019a), a dynamic load balancing approach is developed to reduce the computational cost of modeling wind-induced inundation of the floodplain using a pre-determined mesh with a widely used finite element model called the ADvanced CIRCulation model (Luettich and Westerink, 2004). Our approach relies on a cell-weighting approach similar to that of Ginting and Mundani (2019) and Sanders et al. (2010). The approach uses dynamic grid partitioning and is capable of redistributing all components of the mesh during execution between cores to reflect the time-varying movement of the wet/dry boundary. We show how our approach more efficiently utilizes computational resources and is minimally invasive in the sense that it does not require any modifications to pre-existing models to obtain speedups. To facilitate dynamic load balancing in a highly-scalable parallel element solver such as ADCIRC, we rely on the Zoltan toolkit (Boman et al., 2012) and ParMETIS (Karypis and Kumar, 1998) to provide essential functionality to the application.

The rest of this article is organized as follows: first, we describe our strategy to reduce the computational cost associated with the floodplain by integrating the Zoltan toolkit within a finite element solver and developing an algorithm to dynamically remove the floodplain from the computational problem. Then we describe the effect and behavior of the load balancing approach when it is applied to an idealized case study. Finally, we test our implementation in a real-world coastal flooding application in North Carolina, U.S.A. The paper concludes with a discussion on the key findings.

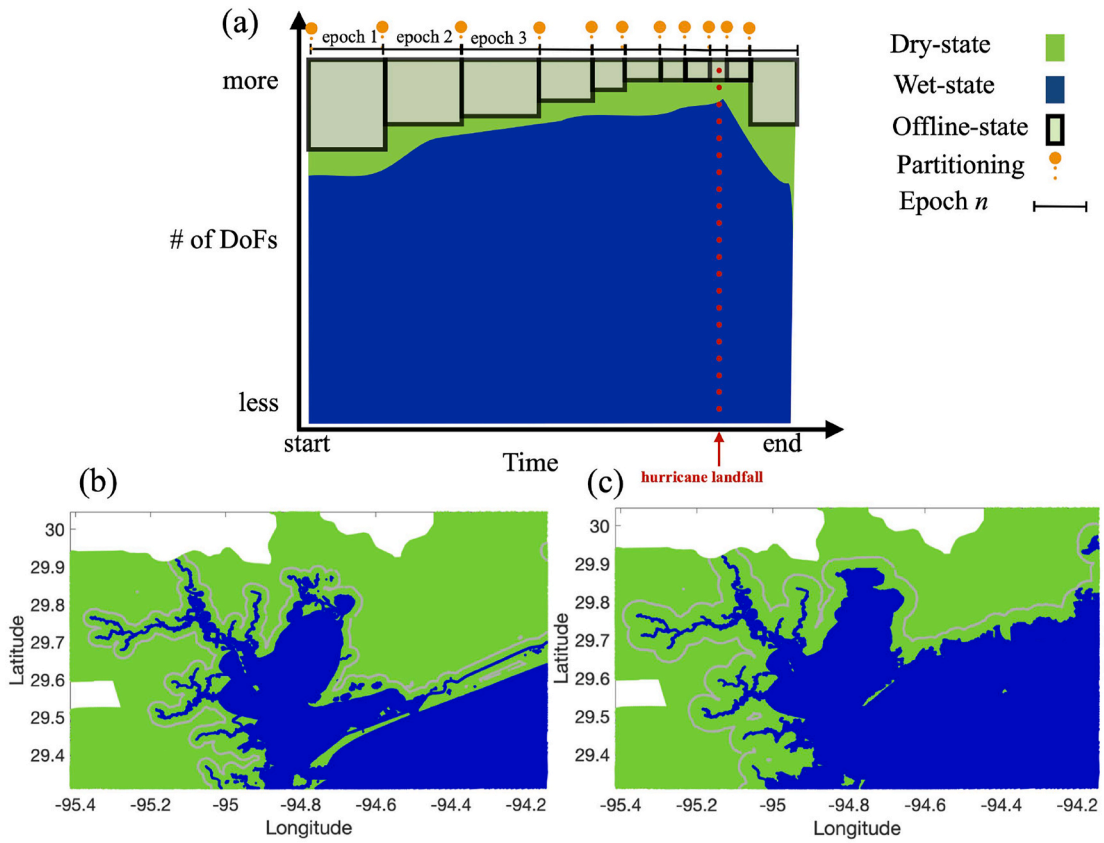

Fig. 2. A cartoon illustrating key concepts in the load balancing strategy. In panel (a), the computational load as a function of time is shown for a hypothetical hurricane-driven coastal flooding event. The shaded off-green color bars indicate epochs in which a fraction of the total number of DoFs are set to an offline state. In panels (b) and (c) the computational domain is partitioned with the color blue denoting portions of the domain that are underwater and the color green denoting parts that are overland before (b) and after (c) the storm makes landfall. The grey dots depicted in panels (b) and (c) represent checkpoint vertices, as explained in the text.

3. Methods

3.1. ADCIRC hydrodynamic model

The dynamic load balancing application is built around the ADvanced hydrodynamic CIRculation model (Luettich and Westerink, 2004) to improve the computational performance of both existing and future modeling systems that rely on ADCIRC. ADCIRC (http://adcirc.org) has become one of the most widely used community modeling platforms for storm surge/coastal flooding predictions across academia, United States governmental agencies and the private sector. This is due to its inclusion of critical physics (e.g., sub-grid scale features such as levees and floodwalls; explicit inclusion of spatially varying land cover and its influence on both surface wind conditions and bottom stress; coupled waves, surge, tides and runoff; interfaces to multiple meteorological model forcings), accurate numerics and optimization for high performance computing. Furthermore, an active development community continues to advance the model's capabilities.

ADCIRC solves the Shallow Water Equations (SWEs) using the Generalized Wave Continuity Equation (GWCE) (Kellogg, 1988; Lynch and Gray, 1979). 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 (Luettich and Westerink, 2004). A parallel version of ADCIRC is implemented for a distributed memory parallel system with the message passing interface (MPI). The solver is scalable and demonstrates linear speed up at CPU cores of 10,000 and greater (Tanaka et al., 2010).

ADCIRC represents the wetting and drying process on an elemental basis, in which elements must either be fully wet or fully dry through logic-based conditions (Luettich and J. Westerink, 1999; Dietrich et al., 2004). The wetting and drying process creates a moving boundary in the computational domain that can move at most one element per time step due to the order of the wet-dry logical operations. While many different and potentially more advanced wetting and drying methods exist (Greenberg et al., 2005; Bates and Hervouet, 1999; Kärnä et al., 2011; Candy, 2017; Warner et al., 2013; Medeiros and Hagen, 2012b), the ADCIRC model with its wetting and drying approach is used by various United States government and state entities for coastal design and risk assessment among other use cases (e.g., Cobell et al., 2013).

Both ADCIRC v53 (ADCIRC) and ADCIRC + Dynamic Load Balancing (ADCIRC + DLB) source codes are compiled identically using Intel Fortran 17.1 compiler with the -O2 optimization strategy and the MVAPICH implementation of message-passing. Double-precision arithmetic is used for all calculations. Besides the DLB capability, there are no differences in the source code implementation between the two versions of the code.

3.2. Dependencies

The Zoltan toolkit is used as a data-parallel programming library that we built the load balancing algorithm upon (Devine et al., 2002; Boman et al., 2012). Built with MPI, Zoltan provides configurable MPI-based unstructured-grid data partitioning through an interface to several state-of-the-art graph and hypergraph partitioners. It also provides a

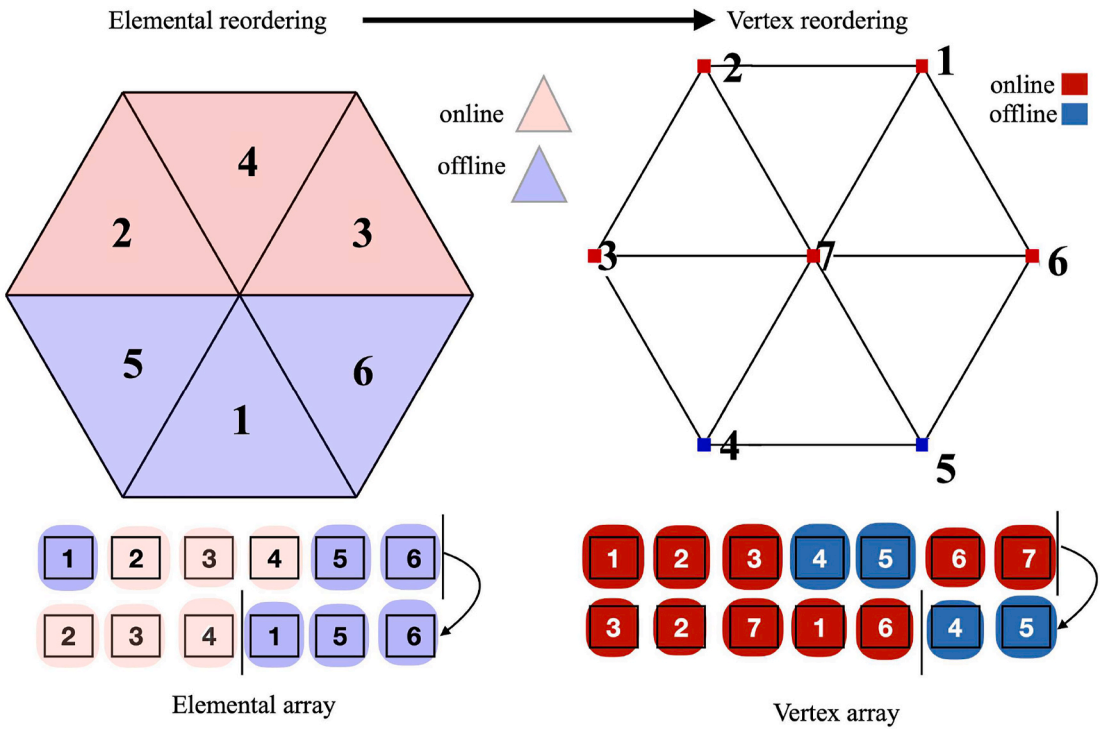

Fig. 3. A schematic illustrating array rearrangement on a patch of elements. Color highlights indicate the online and offline portion of the mesh. The portion of the array that is not iterated in the new reordering is referred to as "clipped". The vertical black bar denotes the end of the iterable space of the array.

scalable data exchange algorithm that uses distributed data directories to locate non-local data (Pinar and Hendrickson, 2001; Devine et al., 2002). Zoltan requires users to define a set of call-back functions to facilitate the neighbor exchange of enqueued data.

In this work, we choose to work with a graph-based partitioner ParMETIS (Karypis and Kumar, 1998) due to its widespread usage and its integration with the Zoltan toolkit. Specifically, we use ParMETIS V4.0.3 program with the implementation of the K-way Multi-level graph decomposition algorithm. ParMETIS's Part K-way graph decomposition algorithm is called via Zoltan's interface.

3.3. Load balancing strategy

Our strategy to reduce the cost of modeling the coastal floodplain involves trading a computational workload balance for a memory imbalance. In this strategy, the decomposition is done in such a way that the load is 'balanced' with respect to the wet-state component of the mesh. As a consequence, the majority of the dry-state component of the mesh belong only to a few subdomains hence introducing memory imbalances. As shall be explained below in Section 3.5, when the majority of dry-state mesh are grouped together, the computing time can be reduced through a rearrangement of data in memory. The data rearrangement leads to an offline portion of the mesh that is not involved in timestepping and an online portion that is (Fig. 2).

At the start of the simulation, the wet/dry boundary is located near the cold-start water level. Depending on the extent of the floodplain, this enables the majority of dry-state mesh data to be located in the offline portion of the subdomains and a theoretical work savings to be obtained through our load balancing approach (Fig. 2). As the movement of the wet/dry boundary begins to reach the boundary of the offline-state subdomains, the computational workload may need to be rebalanced either to maintain a target performance level and/or to satisfy constraints imposed by our approach. In this way, the simulation can be thought of as a sequence of epochs each composed of a rebalancing phase and a subsequent calculation phase. In the case that all mesh data are wetted, the load and thus the theoretical speed of the parallel application ideally should become approximately equivalent to that of static ADCIRC.

3.4. Mesh partitioning

To form the decomposition, the mesh is represented as an undirected graph. Vertices of the graph are assigned a non-dimensional weight proportional to its anticipated computational expense in a similar manner to (Ginting and Mundani, 2019; Ginting et al., 2020). To reflect communication costs and data dependencies, the edges of the graph are given a non-dimensional weight as well.

Elements and vertices are weighted depending on an offline or online status, which is based on a set of user-defined criteria. Elements that have an offline status are weighted with a value of 0, whereas online elements are weighted with a value of 1. Graph edges that connect two offline elements are weighted with the value 1, whereas graph edges that connect either two online elements or one online element to an offline element are weighted with a relatively larger value of $_{1,\ 000}$. Graph edge weights with a value of 0 are not permitted in ParMETIS. ParMETIS then uses these weights in partitioning the graph into a number of subdomain. The disparity in graph weights between the online and offline portion of the mesh represent their true cost.

Algorithm 1. Mesh partitioning algorithm

Algorithm 1: Mesh partitioning algorithm

Result: Parallel decomposition of mesh across N cores with ghost zones.

Call ParMETIS to produce element-core ownership;

if *Element ownership changes* **then**

- Update element data directory;
- Exchange elements;
- Query element data directory for updated element adjacency ownership;
- Elements form vertex ownership;
- Update vertex data directory;
- Exchange vertices;
- Query vertex data directory for updated vertex adjacency ownership;
- Exchange ghost vertices;
- Exchange elements that contain ghost vertices (ghost elements);

end

To decompose the mesh, a sequence of migration and query operations are performed (Algorithm 1). We stress here that the entire mesh (both dry and wet state components) can be redistributed to new subdomain configurations each rebalancing epoch.

To start the application, slices of the finite element mesh along with their weights are passed to a call-back function that makes calls to ParMETIS within the Zoltan toolkit. Elements are first migrated so that each element of the mesh is owned by only a core. Query operations are performed using the distributed data directories provided in Zoltan. Note that data directories for both vertices and elements are over allocated by a factor of three times the initial number of vertices or the number of elements on each core, respectively, to avoid reallocation operations during dynamic load balancing. This over allocation factor was found to be sufficient to avoid reallocations of data directories while avoiding hash collisions.

Vertex ownership is formed implicitly by the element ownership since three vertices of the element belong to the element's owner. On the border of the subdomain, only the relatively higher-numbered core is designated an owner of the vertices. Thereafter, a single layer of vertices are imported on the relatively lower-numbered core in each pair of adjacent subdomains. The duplicated vertices that border all subdomains are termed halo vertices.

3.5. Array rearrangement

After Algorithm 1, the arrays representing the mesh are rearranged according to their online/offline status (Fig. 3). Online elements are placed contiguously in memory at the start of the array and offline elements are then located in memory after the last online element. In this memory arrangement, the number of loop iterations can be reduced by the number of offline elements on each core by reducing the loop range. In this way, a speed-up of the simulation can be achieved provided work is balanced across all cores. The reduction in loop iterations by array rearrangement is referred to as "loop-clipping".

When the elemental loop range is reduced, all of the online element's vertices must be accessible. This is accomplished by iterating through the online-state elements and placing their associated vertices at the start of the array while ensuring that vertices are not duplicated. The process is repeated for the offline-state elements array, placing these offline vertices after the last online-state vertex (Fig. 3).

Loop extents are set to the memory location of the online element and online vertex with the respective adjacent memory location labeled offline.

3.6. Rebalancing strategies

To trigger a rebalance, a set of vertices called checkpoints shared between the boundary of the online- and offline-state portion of the mesh are checked for wetting each time step (Fig. 2 (b),(c)). If any checkpoint vertex is wetted, the simulation must pause and rebalance by executing Algorithm 1. Note that at a minimum, the buffer zone is one element wide and since the wetting/drying algorithm can, at most, only move the wet/dry boundary one element per timestep (Section 3.1), this avoids a scenario where the wetting/drying front would not advance landward in the same manner as ADCIRC.

We've found that it is not possible to set all dry elements and vertices offline. Instead, a zone of elements buffering the region between the checkpoint vertices and the online component of the domain is created to control the frequency of rebalance events. As is shown later, the buffer zone enables this approach to be practical for coastal flooding predictions. We implemented the following rebalance heuristics:

1. Distance: If a vertex's nearest distance from the wet/dry boundary exceeds a user-defined threshold $_{db}$ in meters, the vertex is set to an offline-state. If the three vertices of an element all exceed $_{db}$, then the element is set to an offline-state. An assumption here is that locations in close proximity to the shoreline are most likely to be flooded. We stress that the distance to the shoreline front does not necessarily imply that the element will be flooded as the wetting/drying algorithm advances the front based on a balance between the hydrodynamic pressure gradient force across an element and friction, which is related to the forcings, topography, and landcover nearby the element (Luettich and J. Westerink, 1999; Dietrich et al., 2004).
2. Topographical: A vertex's elevation must be located above a user-defined value $_{hb}$ in meters above the model's reference datum to be set to an offline-state. If the three vertices of an element all exceed $_{hb}$, then the element is set to an offline-state. In practice, the $_{hb}$ can be selected based on the local inter-tidal range.

The definition of the criteria to determine online and offline state vertices are globally defined among cores.

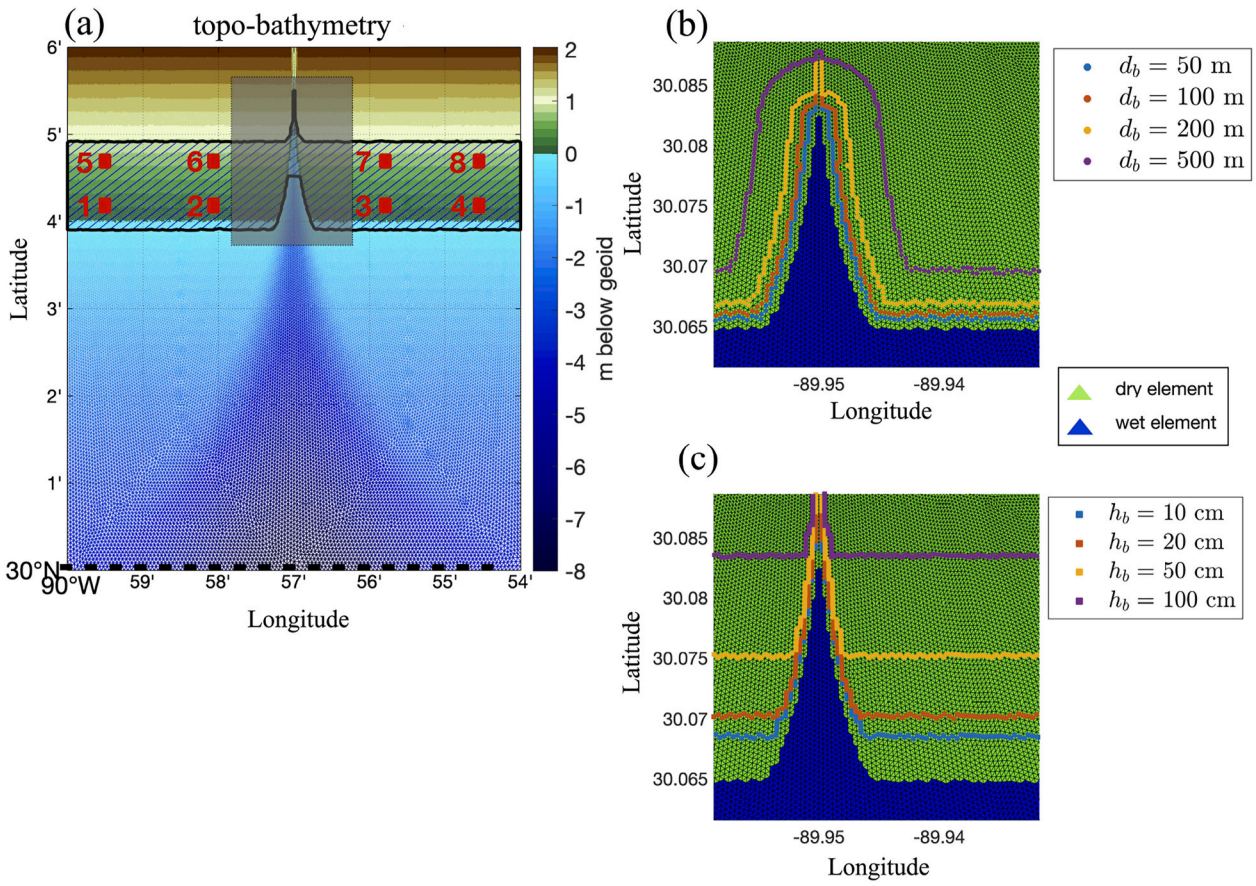

Fig. 4. The 'ideal channel' problem. Panel (a) shows the model's topography and bathymetry interpolated onto the triangular elements. The elevation specified boundary is indicated by a thick dashed black line and eight stations where water elevations are recorded are annotated as red squares with the station number. The hatched region indicates the areal extent of the inundation. Panels (b) and (c) show close-ups of the location of different checkpoint vertices for the distance- ($_{db}$) and depth-based ($_{hb}$) buffer configurations that were tested.

To prevent excessively frequent rebalance events, for example every simulation timestep, each time the checkpoints of the buffer are wetted, $_{hb}$ is elevated by $_{10\%}$ of its existing value. This is not necessary for $_{db}$ since the nearest distance is calculated from a wet/dry boundary which is moving in space. The selection of increasing $_{10\%\ hb}$ each rebalance event was based on the idea that we want to gradually lift the checkpoint vertices away from the wet/dry front as flooding occurs to maximize the amount of DoFs set to an offline state. As more flooding occurs, $_{hb}$ elevates a larger increment and avoids a rebalancing every timestep, which would otherwise occur during a highly energetic flooding event.

A key assumption in the formation of the buffer is that the solution does not exhibit numerical artifacts and/or instabilities. It is noted that the shoreward movement of the wet/dry boundary was not used as a factor in determining rebalance events; however, the user does have the option to schedule rebalance events, which can be used to lower the buffer zone shoreward after a flooding event has passed. Note that improving the computational efficiencies post-storm through lowering the buffer shoreward could become more important for real-time predictions due to operational constraints.

3.7. 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 the sum of the time spent computing $_{TC}$ and the time spent rebalancing $_{TRB}$:\

$$T = T_C + T_{RB} \tag{1}$$

where $_{TC}$ and $_{TRB}$ are summed over the entire simulation. Specifically, $_{TRB}$ represents the cumulative time spent performing the parallel partitioning algorithm, while $_{TC}$ represents the cumulative time spent in timestepping.

The speed-up over the static version of the code is calculated as:

$$SU_{obs} = \frac{T_{Static}}{T_{DLB}} \tag{2}$$

in which $_{TDLB}$ is the wall-clock time spent timestepping for a given simulation computed with ADCIRC + DLB and $_{Tstatic}$ is the wall-clock time spent timestepping using ADCIRC.

The theoretical maximum speed-up ($_{SUmax}$) is calculated by dividing the total number of vertices $_{VTotal}$ by the average number of wetted vertices $_{VWet}$ for a simulation,

$$SU_{max} = \frac{V_{Total}}{\frac{1}{N_t}\sum_{t=1}^{t=N_t} V_{Wet}^t} \tag{3}$$

where V_{wet}^t is the number of wet vertices at timestep t, $_{Nt}$ is the number of simulation time steps. Following this, if 50% of the domain is wet throughout the entire simulation, the simulation would have a maximum speed-up factor of 2.0. The $_{SUmax}$ represents a maximum potential speed-up assuming: 1) the problem is perfectly load balanced, 2) it has zero communication overhead, 3) dry-state vertices can be completely ignored from the computational problem and 4) there is zero rebalancing cost. Following this, the speed-up efficiency ($_{SUE}$) is computed as the ratio of the $_{SUobs}$ to $_{SUmax}$

$$SU_E = \frac{SU_{obs}}{SU_{max}} \tag{4}$$

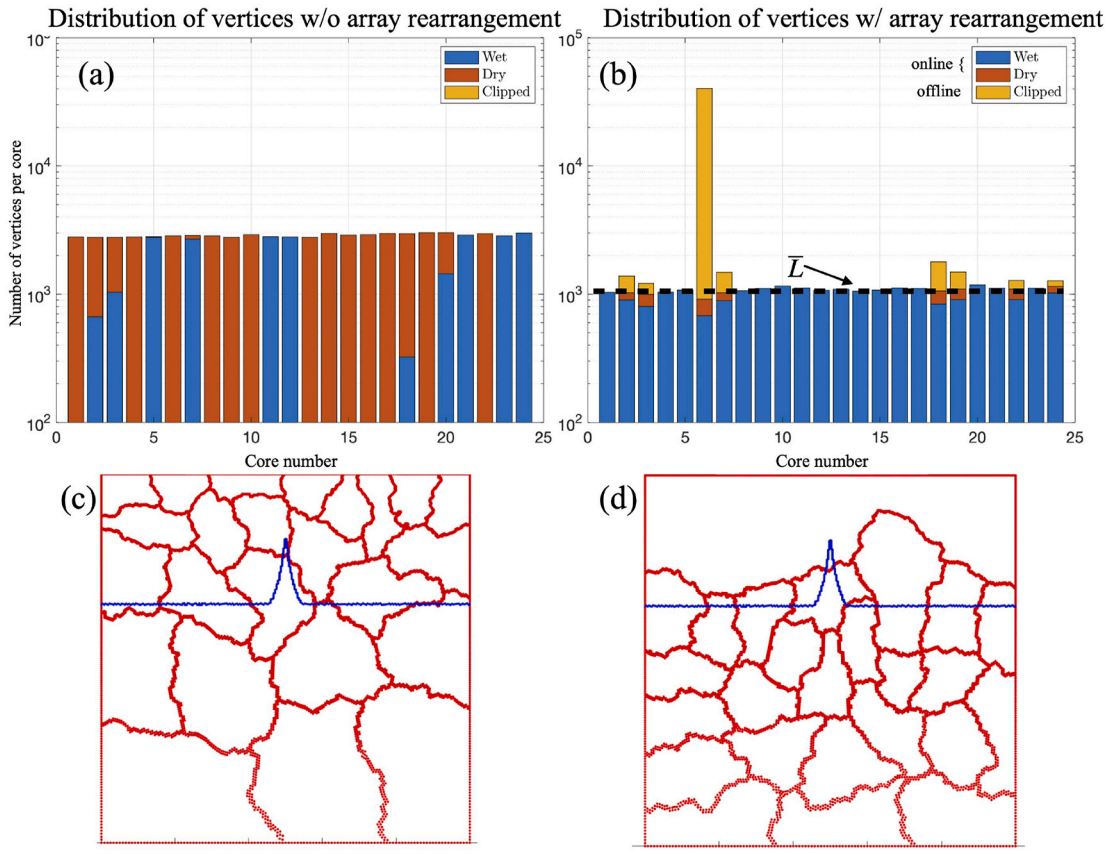

Fig. 5. Panel (a) depicts the vertex distribution on a 24 core decomposition with $d_b = \infty$ configuration and panel (b) depicts vertex distribution with the $_{db=50}$ m configuration at the initial epoch. In panel (b), the mean computational load $\overline{L}$ is annotated. Panel (c) and (d) indicate the extent of all subdomains in red and the location of the wet/dry front at the first timestep is indicated in blue.

where $_{SUE}$ =1 ($_{100\%}$) implies a perfectly efficient calculation. Note that this efficiency metric is subjective to the modeling scenario which influences the amount of wetting and drying in the domain. Nevertheless, if there are fewer dry vertices, then there will be a smaller potential for speed-up, and thus a smaller $_{SUmax}$. However this doesn't necessarily translate to a smaller $_{SUE}$ – that has to be earned by minimizing the costs of the dry vertices and the time spent in the dynamic load balancing (e.g., $_{TDLB}$) operation as much as possible.

To investigate the performance of the application, the load imbalance is calculated. Load imbalance is defined as the ratio of the maximum load L^k_{max} calculated over all cores divided by the average load $L^k_{average}$ calculated over all cores:

$$\overline{R}_{IMB} = \frac{1}{K}\sum_{k=1}^{K} R^k_{IMB}, R^k_{IMB} = \frac{L^k_{max}}{L^k_{average}} - 1 \tag{5}$$

for all K epochs. The quantity L is calculated as the number of online-state vertices per core. All cores ideally should maintain an equal load with an $R^k_{IMB} = 0\%$ in order to ensure that the idle computing time is at a minimum.

For efficient parallelism, inter-processor communication volume should be minimized. In ADCIRC, messages are exchanged several times during a given timestep at the halo vertices of the subdomain. The inter-processor communication volume is estimated through the surface-area-to-volume ($_{SV}$) ratio per k^{th} epoch, which approximates the relative amount of communication cost as compared to the computing cost:

$$\overline{SV} = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{N_s}\sum_{n=1}^{N_s}\left[SV^k_n = \frac{\text{number of halo vertices on subdomain } n}{\text{total number of vertices on subdomain } n}\right] \tag{6}$$

for $_{Ns}$ subdomains over K epochs. The number of halo vertices on a given subdomain in Eq. (6) is influenced by the shape of subdomains, the number of vertices, and whether the subdomains are connected in the computational domain. As the number of vertices per core is reduced with the usage of more processors, $\overline{SV}$ increases, implying more communication over computation.

4. Results

Simulations were executed on the Computational Hydraulics Laboratory's computer cluster called Aegaeon (https://coast.nd.edu/). Aegaeon contains 83 compute nodes of dual 12 core E5-2680, 2.50 GHz Haswell processors ($_{1,\ 992}$ cores in total). Each node contains 64 GB of random access memory that is shared among each node's 24 cores. The nodes are connected via a high-speed 56 GB Infiniband network. File input/output was disabled (i.e., no logging and no output file writing) however, for the Hurricane Irene test case, meteorological forcing files (winds and surface pressure data) needed to be read into memory every 15 simulation minutes. All timing results were repeated three times and the average is reported.

4.1. Simple tide on an idealized floodplain

The DLB capability is evaluated first for an idealized problem with a channel and floodplain. This problem is referred to as the "ideal channel" and is selected because it allows for a tidal signal to inundate and recede on a gradual linear sloping beach with a narrowing channel along its centerline to provide variations in horizontal directions. While the problem has a predictable wet/dry boundary, the point of the experiment is to demonstrate the application of dynamic load balancing and

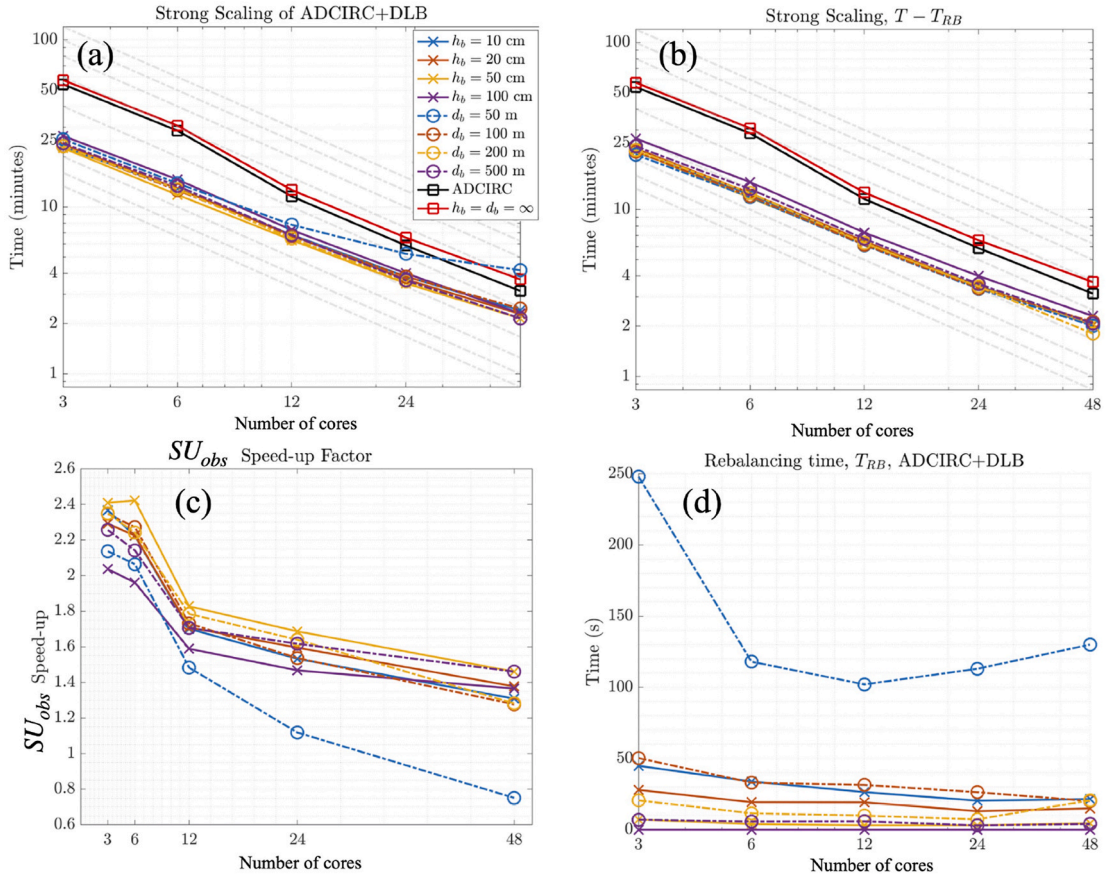

Fig. 6. (a,b) Strong scaling curves for the ideal channel problem (c) the speed-up factor (i.e., Eq. (2)), and (d) the time spent rebalancing $_{TRB}$.

some of the associated concepts.

4.1.1. Model and setup

A channel and floodplain (Fig. 4) 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 at a cold-start state. Bathymetry varies linearly and gradually with a constant slope of approximately 0.002 from $_{-8}$ m to$_{+2}$ m above the model's LMSL onto the floodplain. Element sizes vary from 75 m near the open boundary to 15 m along the shoreline and then coarsens to 35 m overland.

A tide with an amplitude of 1 m and a period of 12.42 h (i.e., semi-diurnal M_2 frequency) and a ramp of 0.5 simulation days is prescribed along the open boundary (Fig. 4). This boundary forcing generates a fluctuation of 1 m above and below LMSL with episodic inundation covering $_{13,\ 240}$ vertices (20.55% of the vertices) (hatched area in Fig. 4). This fluctuation of water levels leads to a maximum of 54.30% of total vertices inundated in the mesh. The application of the elevation specified boundary with a 0.5 simulation day ramp generates a wet/dry boundary that rises in a "rigid" fashion and mimics a cresting tidal inundation in a small enclosed estuary. The simulation uses a time step of 0.5 s with a maximum Courant number less than 0.25 and the GWCE is solved with an explicit time scheme with a mass-lumping approach. For this problem, the maximum theoretical speed-up (c.f., Eq. (3)) is 2.56.

The ideal channel problem is run using five different core configurations (i.e., 3, 6, 12, 24, 48 cores). Each core configuration is run with a sequence of progressively wider buffer configurations. Four configurations based on nearest distance to the wet-dry boundary are considered: $_{db}$=50 m, $_{db}$=100 m, $_{db}$=200 m, $_{db}$=500 m (Fig. 4). Likewise, four configurations are based on topography (i.e., meters above the LMSL): $_{hb}$=0.10 m, $_{hb}$=0.20 m, $_{hb}$=0.50 m, and $_{hb}$=1.0 m. An infinite size $d_b = h_b = \infty$ configuration that represents the static case using the decomposition algorithms was also tested. Each time a rebalance event is triggered in the depth-based configurations, $_{hb}$ is increased by $_{10\%}$ of its current value, while the $_{db}$ remains fixed (c.f., Section 3.6).

4.1.2. Effect on decomposition

Compared to the case when all elements/vertices are weighted equally, the $_{db}$=50 m configuration greatly increases the number of vertices in subdomains located in overland regions (Fig. 5(c)-(d)). Consequently, the number of total vertices in subdomains with predominately online vertices are reduced, which reduces the computational problem size. The number of dry-state online vertices are greatly reduced per core while the distribution of online vertices remains well balanced (Fig. 5(b)). As $_{db}$ is increased, more vertices are set offline and this reduces the number of vertices in subdomains with predominately online vertices.

4.1.3. Timing improvements

ADCIRC + DLB demonstrates approximately linear scaling in all cases considered (Fig. 6(a)). Significant speed-ups were measured (e.g., up to $_{SUobs=2.44}$) but became less significant and more inefficient with the usage of more cores. In configurations where a small value of $_{db}$ is considered, the time spent rebalancing became similar in value to the total simulation time diminishing the speed-up. Overall, ADCIRC + DLB performed slightly slower than ADCIRC in the $_{db}$ = $_{hb}$ = ∞ configurations. It was found that because ADCIRC + DLB decomposes the dual graph whereas ADCIRC decomposes the nodal graph, the decomposition of the dual graph produced slightly greater load imbalances resulting in

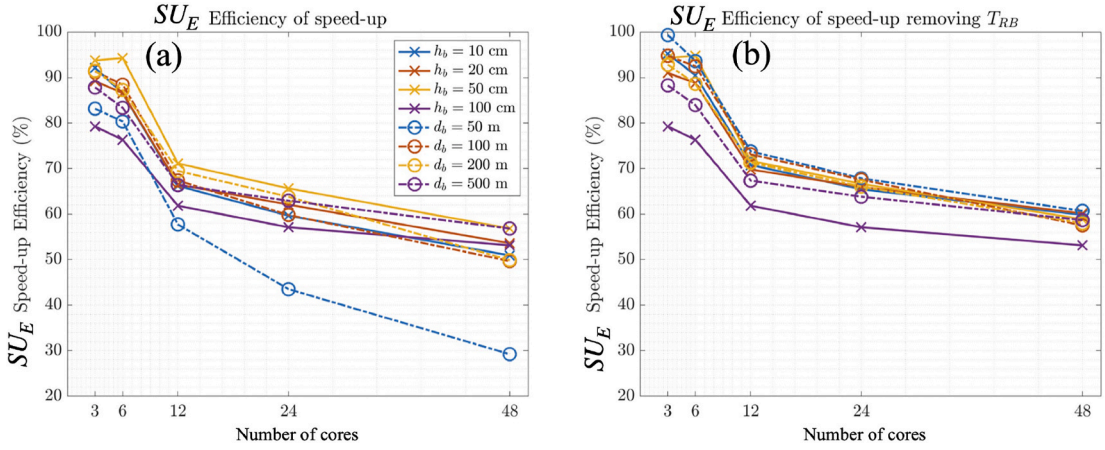

Fig. 7. The speed-up efficiency $_{SUE}$ (Eq. (4)) for the ideal channel problem.

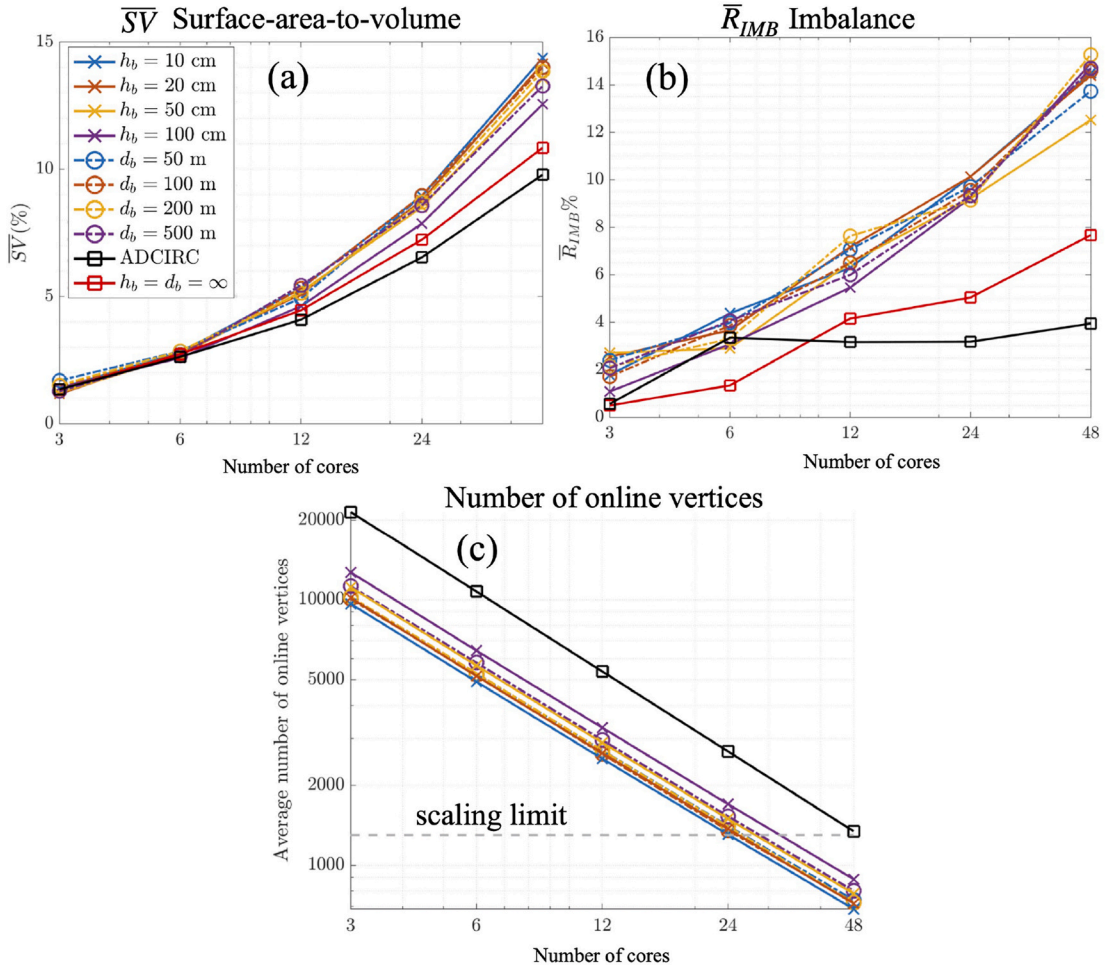

Fig. 8. Panel (a) shows the simulation average surface-area-to-volume $\overline{SV}$ (Eq. (6)). Panel (b) shows the simulation average load imbalance $\overline{R}_{IMB}$ (Eq. (5)). Panel (c) shows the simulation average number of online vertices. For ADCIRC, this statistic is calculated by dividing the number of vertices by the number of cores. The scaling limit is indicated on panel (c) from Tanaka et al. (2010).

marginally slower execution times.

ADCIRC + DLB produced 31 out of 32 faster simulations with speed-up factors that ranged between $_{SUobs=1.10}$ to $_{SUobs=2.44}$ (Fig. 6(c)). Distance and depth-based buffer performed similarly although changes to $_{db}$ produced greater changes in timing results than changes to $_{hb}$. We observed that configurations using small distance-based buffers experienced far more rebalance events as the wet/dry boundary initially advanced shoreward from its cold-start state than the depth-based buffer configurations. This resulted in more variation in timing results. The fastest simulation was $_{hb=50}$ cm and produced a speed-up factor that ranged from 2.44 to 1.42 when considering 3 to 48 core configurations, respectively. The highest speed-up of $_{SUobs=2.44}$ was achieved as a result of minimizing $_{TRB}$ while maximizing the $_{SUmax}$. For instance, configurations with a $_{hb<0.50}$ m had greater $_{TRB}$, which resulted in slower

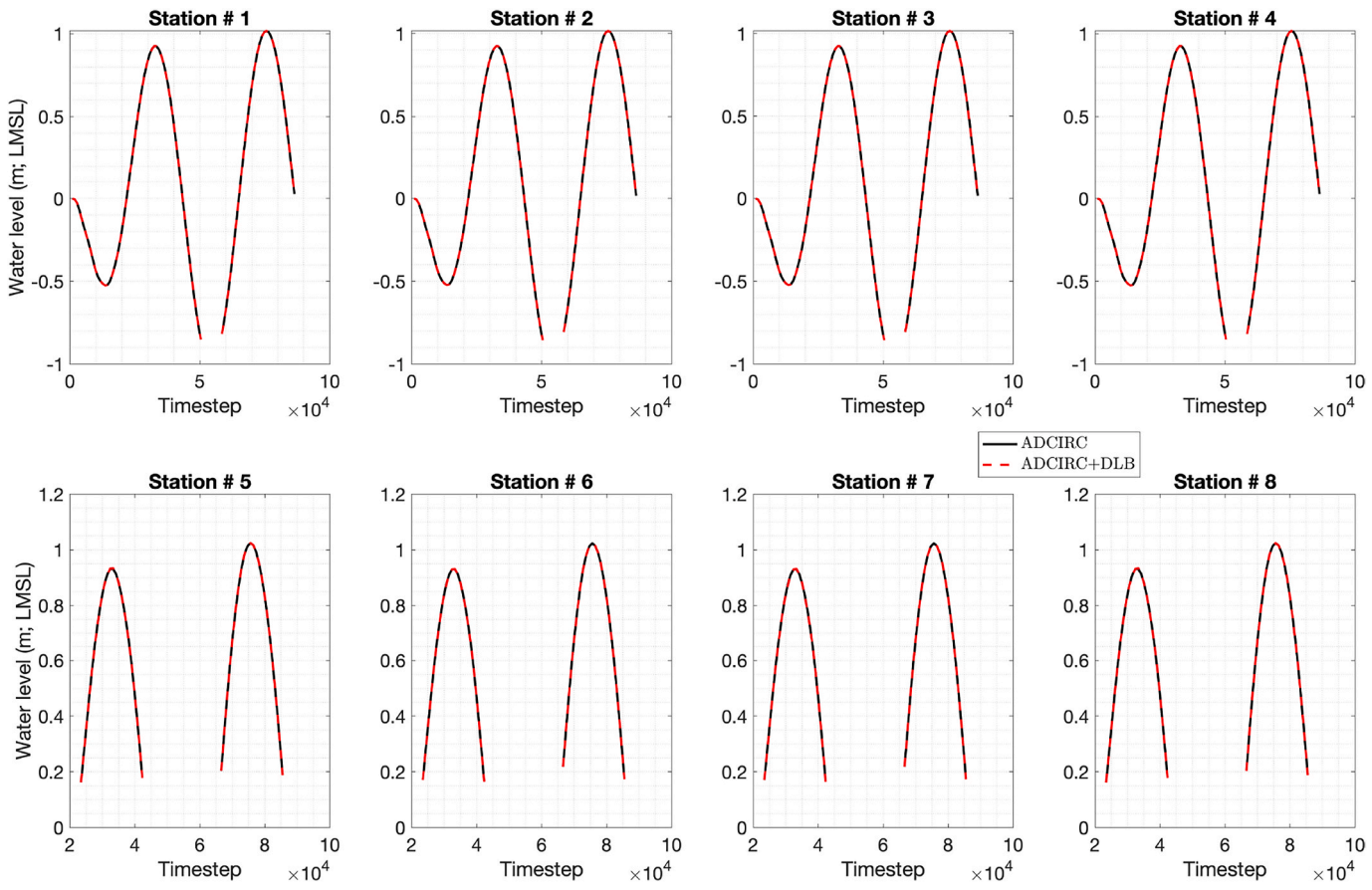

Fig. 9. A comparison of water levels between ADCIRC and ADCIRC + DLB at the eight stations annotated in Fig. 4 for the best performing experiment in the ideal channel case (i.e.,6 cores $_{hb=50}$ cm).

simulations despite that their theoretical maximum speed-up $_{SUmax}$ was greater.

The time spent rebalancing represented maximally 16% of the total simulation time while the majority of simulations (31 out of 32) spent $_{<}$3% of the total time in rebalancing operations (Fig. 6(c)). While the time spent in rebalancing operations can be considered relatively small, $_{TRB}$ does not exhibit scalability (Fig. 6(d)) as rebalancing requires a number of communication and query operations (c.f., Algorithm 1). The slowest performing experiments used buffer configurations that initially set the most data offline, which leads to more rebalancing events and thus far greater $_{TRB}$. For instance, the $_{db=10}$ m corresponded to the narrowest buffer and resulted in a slow down of 1.42 using 48 cores due to $_{TRB}$ (Fig. 6(b)).

For all simulations that were faster than ADCIRC, their efficiency $_{SUE}$ ranged from 53% to 96% (Fig. 7). The 3 and 6 core configurations were consistently more efficient than the 12 and 24 core configurations. The most efficient ($_{SUE=94\%}$) simulation was measured using 6 cores with the $_{hb=50}$ cm buffer, while the $_{db=200}$ m configuration produced the second most efficient simulation with a maximum $_{SUE=92}$%. The rest of the configurations produced lower $_{SUE}$ as the $_{hb}$ elevated and $_{db}$ increased as less of the theoretical speed-up was realized.

Subtracting $_{TRB}$ from T demonstrates that the theoretical speed-up increases as more data is set offline, but at the cost of more time spent rebalancing (Fig. 7(a and b)). When $_{TRB}$ is removed from the timing statistics, the most efficient simulations are those that initially set the greater component of the mesh offline (i.e., $_{db=50}$ m and $_{hb=10}$ cm) (Fig. 7 (a)). However, the opposite is observed when $_{TRB}$ is included, which indicates that a configuration that minimizes both T and $_{TRB}$ exists with a buffer configuration near to the $_{hb=50}$ cm and $_{db=200}$ m (Fig. 7(b)).

In all experiments compared to ADCIRC, decompositions generated by ADCIRC + DLB had larger $\overline{SV}$ by approximately $_{1\%}$ whereas less consistent differences for $\overline{R}_{IMB}$ were measured (Fig. 8). Configurations that set more data to an offline state produced a consistent increase to the $\overline{SV}$ by maximally 3–5%. The increase in $\overline{SV}$ is expected considering that the size of the online component of the problem is reduced through array rearrangement and shortening of loops (Fig. 8(c)). The 48 core setup leads to buffers configurations falling below the scaling limit as was reported in Tanaka et al. (2010). Similarly to $\overline{SV}$, a clear increase to $\overline{R}_{IMB}$ is measured by 6–8% as compared to ADCIRC + DLB and also increases as more cores are used. Unlike $\overline{SV}$ however, the disparity in $\overline{R}_{IMB}$ between different buffer configurations is greater. There is no clear relationship between the number of vertices set to an offline state and $\overline{R}_{IMB}$.

4.1.4. Comparison with static ADCIRC

A comparison of water levels between ADCIRC and ADCIRC + DLB is shown in Fig. 9 for the best performing experiment (i.e., 6 cores, hb = 50 cm) in the ideal channel test case. ADCIRC + DLB solutions are numerically identical to that of ADCIRC, and the timing of wetting and drying is unaltered by the dynamic load balancing (Fig. 9).

4.1.5. Summary of experiment

The best performance ($_{SUobs=2.44}$) was obtained using a depth based buffer of $_{hb=50}$ cm. Speed-up factors had relatively larger variation from $_{SUobs=1.10}$ to $_{SUobs=2.44}$ depending on the configuration of the buffer. $_{TRB}$ represented less than 3% of the total simulation time with the exception of $_{db=50}$ m where it created a significant slow-down. Overall, $_{db}$ had more variability than $_{hb}$ in $_{TRB}$ to its configuration than the depth-based buffer. The application of DLB produced decompositions that had greater

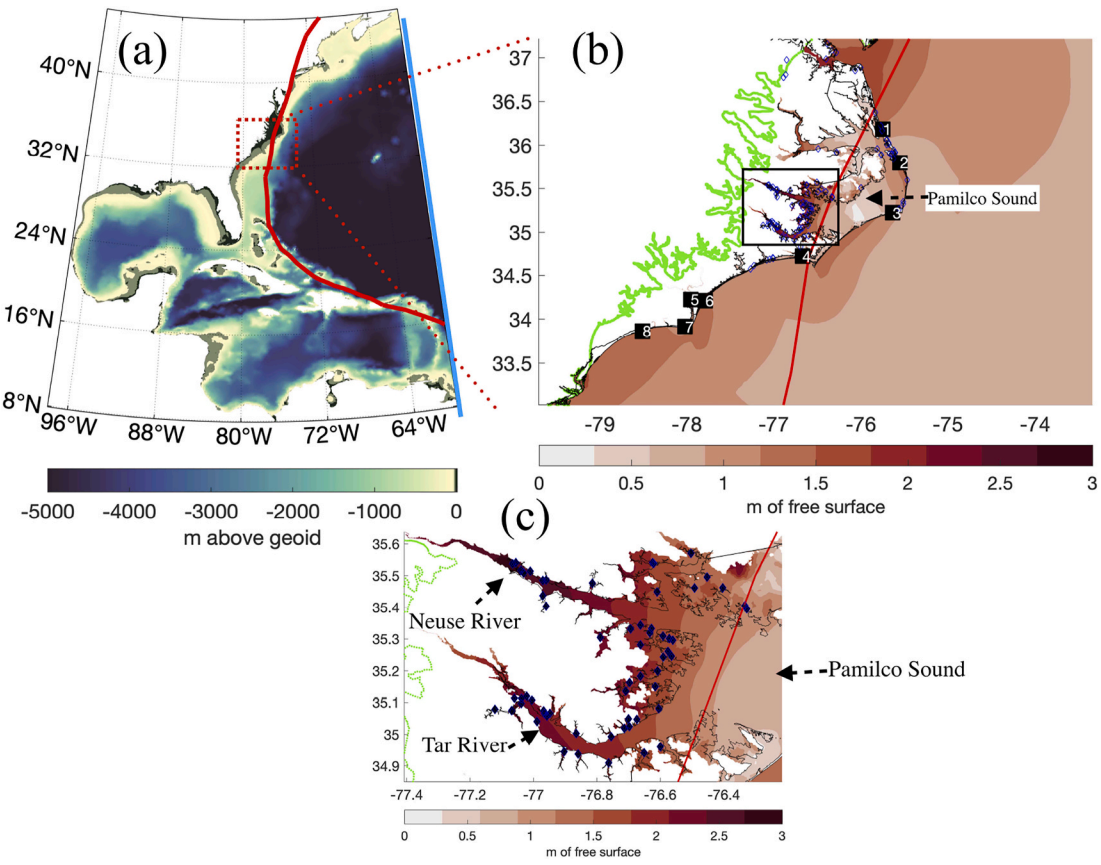

Fig. 10. The (a) NC9 mesh's topo-bathymetry with the region of interest around the Outer Banks of North Carolina indicated by the red dotted box. In panel (a) the solid blue indicates the open ocean boundary where elevation forcing is prescribed. In panel (b), the simulated maximum free surface elevation above the geoid during the simulation of Hurricane Irene is shown along with the location of eight NOAA gauges (black squares) and 74 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. (c) Depicts the black region indicated in (b).

surface-area-to-volume $\overline{SV}$ and greater imbalance factors $\overline{R}_{IMB}$ by 1–3% but this did not appear to impact parallel performance significantly. In general, the application of DLB reduced the problem size (Fig. 8(c)) and this can lead to higher levels $_{SV}$ and more $_{RIMB}$.

Based on these findings, for applications our recommendation is to use a depth-based buffer configuration with an initial height of 0.50 cm, which was the best performing experiment. Further, when selecting the buffer, it is important to ensure that the application of DLB does not reduce the problem size below the scaling limit of ADCIRC.

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 an extensively validated mesh of the mid-Atlantic United States region. This mesh was validated during Hurricane Isabel (2003) (Blanton et al., 2018) and Hurricane Irene (Dresback et al., 2013) and the model is used for real-time predictions for the North Carolina Forecasting System (Blanton et al., 2012). Additionally, it contains an extensive floodplain. Unlike the ideal channel problem, this test problem features an irregularly moving wet/dry boundary forced by a combination of meteorological forcings and astronomical tides interacting with observed topography/bathymetry datasets.

Hurricane Irene impacted the mid-Atlantic region as a Category 1 hurricane with 10 min sustained winds of approximately 78 mph on August, 27–28 2011 (Seroka et al., 2016). Measured water levels of approximately 3 m above LMSL were observed on the westward side of the cyclone's track near the Tar/Pamlico Sound and Neuse River basins in North Carolina (Fig. 10; Dresback et al., 2013). Elsewhere in the Pamlico Sound, peak water levels of 1 m–2 m above LMSL were observed.

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 (Dresback et al., 2013). Along the South Atlantic Bight, a patch of elements were modified to improve their quality, otherwise the mesh was identical to the original. At a cold-start state, approximately 56% of the mesh vertices are dried. In the area impacted by Hurricane Irene, the tides range between 0.40 and 0.60 m above LMSL. Approximately 10–12% of the floodplain vertices are flooded during an average tidal cycle (Fig. 10). 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 (Blanton et al., 2012). In areas nearby large rivers such as the Tar and Neuse Rivers, 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 $_{SUmax}$ for this particular setup was 1.97.

The setup used in this section resembles the operational simulation used in (Fleming et al., 2008): a simulation of 8 days, explicit numerical scheme, 0.5 s time step, atmospheric and tidal forcing, and point-based outputs of free surface elevation. Wind and pressure hindcast fields from Ocean Weather Inc. (OWI) were used to force the model between the

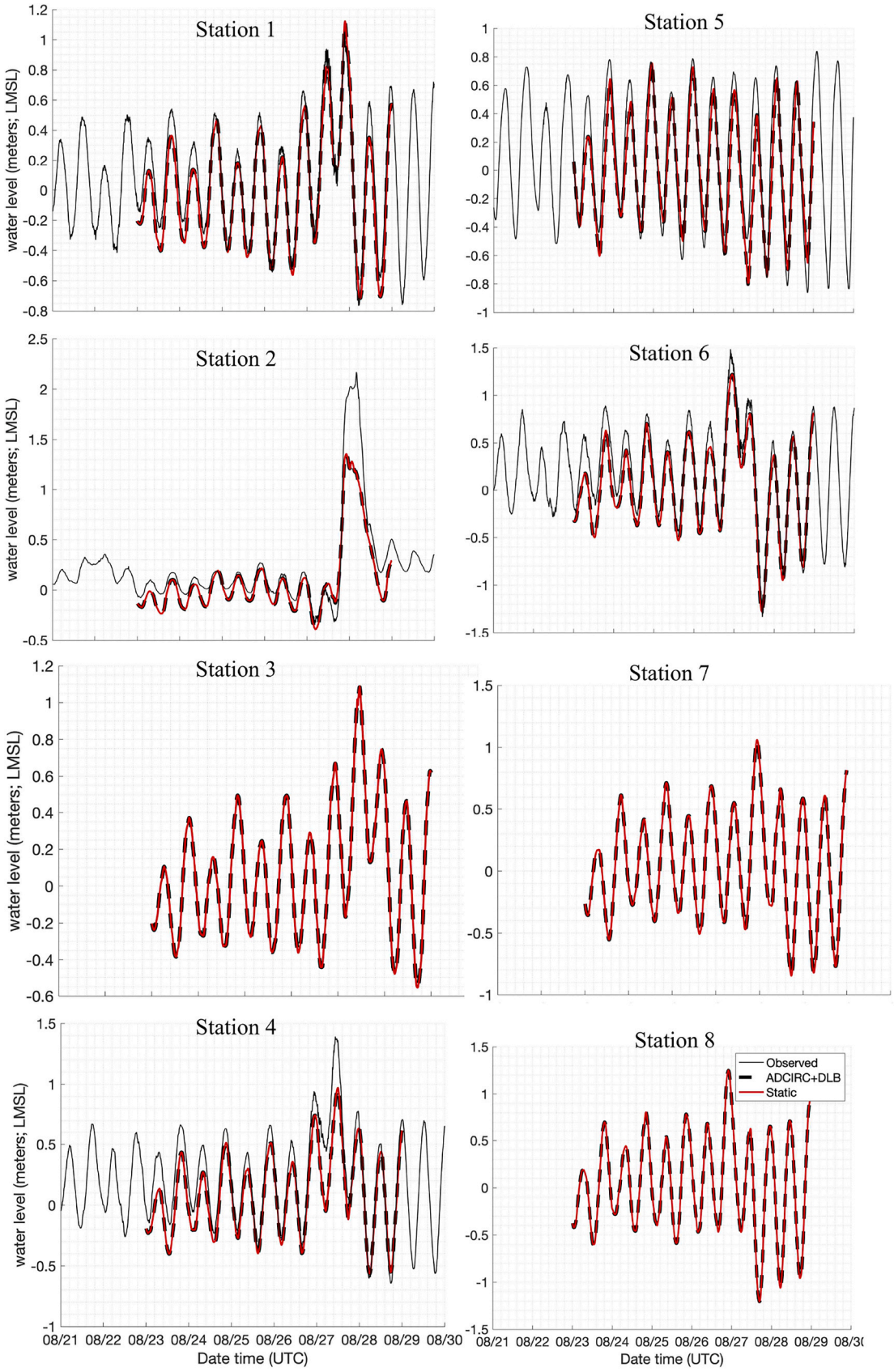

Fig. 11. Time series comparisons of total water levels at the eight National Oceanic Service (NOS) gauges (Fig. 10). Observations are shown at a solid black line where available.

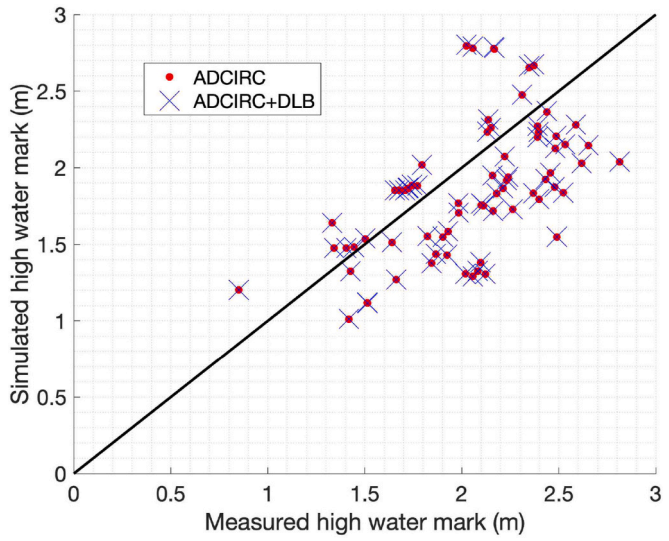

Fig. 12. Agreement in computed high water marks between ADCIRC and ADCIRC + DLB as compared to 74 USGS high water marks (m).

dates of August 21, 2011 12:00:00 UTC to August 29, 2008 00:00:00 UTC at a time increment of 15 min.

Temporally consistent tidal processes are included in the simulation by specifying elevation boundary conditions on an open ocean segment using the TPXO9.1 atlas (Egbert and Erofeeva, 2019) 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). Forcings are ramped from a cold-start state using a hyperbolic tangent function over the first two days of the simulation to avoid exciting transient modes in the study region. Free surface elevation are recorded every 6 simulation minutes from simulation days 2–8 at 74 rapidly deployed gauges by the United States Geological Survey Service (USGS) and an additional 8 National Oceanic Service (NOS) gauges depicted in Fig. 10.

4.2.2. Comparison with measured data

ADCIRC + DLB solutions are effectively identical to that of ADCIRC to within small differences often seen when running ADCIRC with different core configurations (Figs. 11 and 12). These differences are possibly introduced through the threshold-based wet/dry logic (Dietrich et al., 2004). 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 under-predicted the water level response (Fig. 11). Further, good agreement between simulated results and high water mark observations at 74 USGS sensor in the Neuse, Tar, and Pamlico river basins are measured.

4.2.3. Timing improvements

The model setup is executed on five core configurations, more specifically 60, 120, 240, 360, 480 cores. For ADCIRC + DLB, we use the best performing buffer configuration in terms of $_{SUE}$ and $_{SUobs}$ from Section 4.1, which was $_{hb=50}$ cm.

Significant speed-ups that ranged from 1.32 to 1.84 with 480 to 60 cores were measured, respectively (Fig. 13). This implies a reduction in wall-clock times between 10 and 190 min with the usage of 480 and 60 cores, respectively. $_{TRB}$ were relatively negligible occupying less than 58

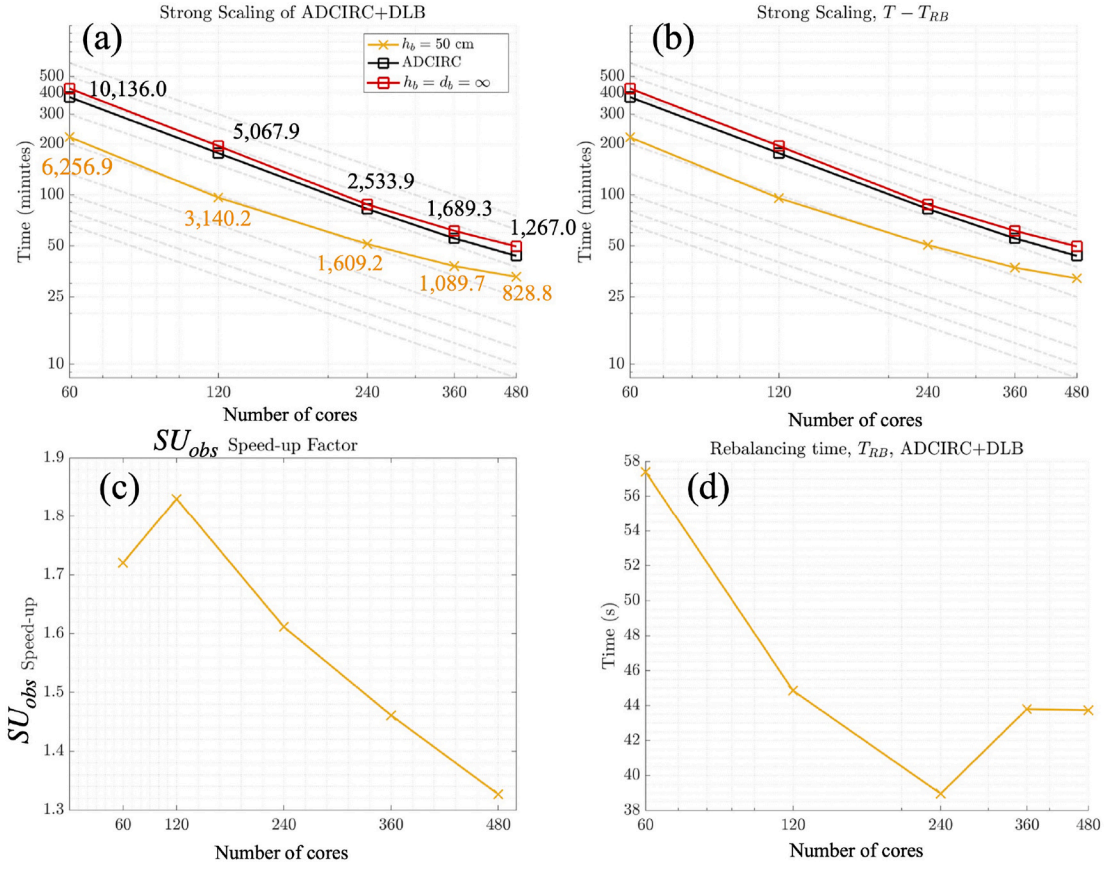

Fig. 13. (a,b) Scaling curves for the Hurricane Irene problem (c) the speed-up factor (i.e., Eq. (2)), and (d) the time spent rebalancing $_{TRB}$. The simulation average (over all k epochs) number of online vertices is annotated for $_{hb}$ and for ADCIRC. For ADCIRC, this statistic is calculated by dividing the number of vertices by the number of cores.

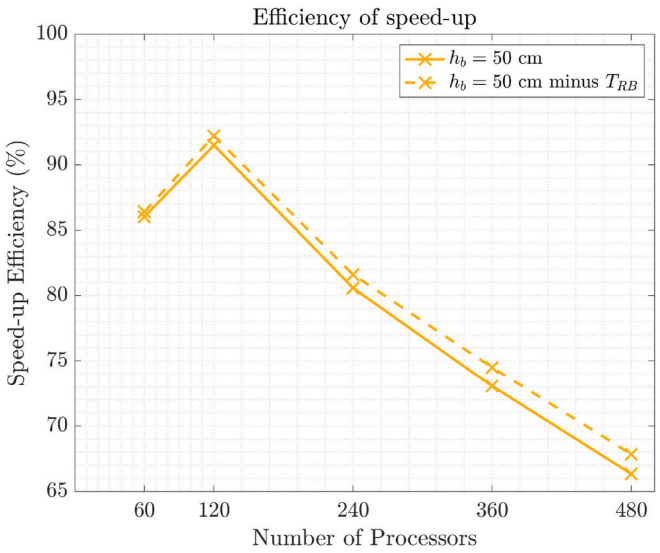

Fig. 14. The speed-up efficiency $_{SUE}$ (Eq. (4)) for the Hurricane Irene problem with (solid line) and without rebalancing time ($_{TRB}$) considered (dashed line).

s of the total simulation time. All simulations rebalanced for a total of 22 times as water levels flooded the floodplain. $_{TRB}$ demonstrated weak scalability and reduced from $_{TRB=58}$ s to approximately $_{TRB=40}$ s using 60 to 480 cores, respectively.

The measured speed ups can be considered highly efficient with a $_{SUE}$ ranging from 69% to 93%. Our most efficient configuration ($_{SUE=93\%}$) occurred with the 120 core setup. The 60 core configuration performed similarly with a $_{SUE=87.8\%}$. Configurations that used more than 120 cores are less efficient. The 480 cores setup is the least efficient simulation with a $_{SUE=69\%}$. Similar to the timing results obtained in Section 4.1, $_{TRB}$ did not negatively affect the model efficiency and only a small reduction of efficiency ($_{0.1–0.3\%}$) was measured that could be attributed to the rebalancing (Fig. 14).

In Tanaka et al. (2010), the scaling limit for the ADCIRC solver with a similar core configuration to ours occurred when there were on average $_{1,\,000}$ vertices per core. Our results reflect a similar pattern with a deviation in the scaling curve (Fig. 13). The application of array rearrangement and loop clipping reduces the problem size to below the scaling limit in the 360 and 480 core configurations, which explains the measured deviation from an optimal scaling rate. (Fig. 13(a)).

In contrast to the ideal channel case (c.f., Section 4.1), the movement of the wet/dry front is complex during the coastal flooding event with substantial shoreward movement of the wet/dry front in the wetland environments immediately north of the Neuse River, and in contrast, limited shoreward advancement of the wet/dry front along the steeper banks and tributaries of the Neuse River (Fig. 15). In spite of the irregular movement of the wet/dry front, the depth-based buffer configuration criteria remains robust and enables a significant speed-up of the overall calculation while triggering relatively few rebalancing events.

5. Discussion and conclusion

The aim of this work is to reduce the wall-clock times spent modeling wind-driven coastal flooding on unstructured triangular meshes using the ADCIRC solver. Regional coastal ADCIRC meshes often contain relatively large amounts of dry-state vertices to represent the finely-detailed nature of the coastal floodplains. Considering variable resolution unstructured mesh/model development is both challenging and time-consuming, modelers cannot design a stable and robust modeling/mesh system that is hand-crafted for each storm that may occur in a given area. As a result, all dry-state floodplain vertices are not actively solved for and this make the model's parallel execution susceptible to large work imbalances and inefficiencies.

Our solution involves in-memory re-decomposing the unstructured mesh in parallel based on the moving domain boundary determined by the wet/dry elemental state. Each time the problem is re-decomposed, local arrays of vertex and elemental data are rearranged to reduce the extent of loops involved in the calculation leading to an acceleration of the program. The third-party libraries Zoltan (Boman et al., 2012) and ParMETIS (Karypis and Kumar, 1998) were used to implement ADCIRC + DLB.

The performance of ADCIRC + DLB as compared to static ADCIRC was studied given a range of computational resources (3–480 cores) in both an idealized problem and a hurricane scenario. Overall, ADCIRC + DLB exhibited similar scaling behavior to ADCIRC with a linear reduction in simulation time. Reductions between approximately $_{20\%–50\%}$ to the total wall-clock time of a realistic coastal flooding simulation were achieved with approximately $_{70\%}$–90% efficiency compared to a theoretical speed-up estimate.

However, the reduction in the problem size associated with DLB given the same amount of computational resources can lead to relative calculation slow-downs to the same calculation computed on ADCIRC

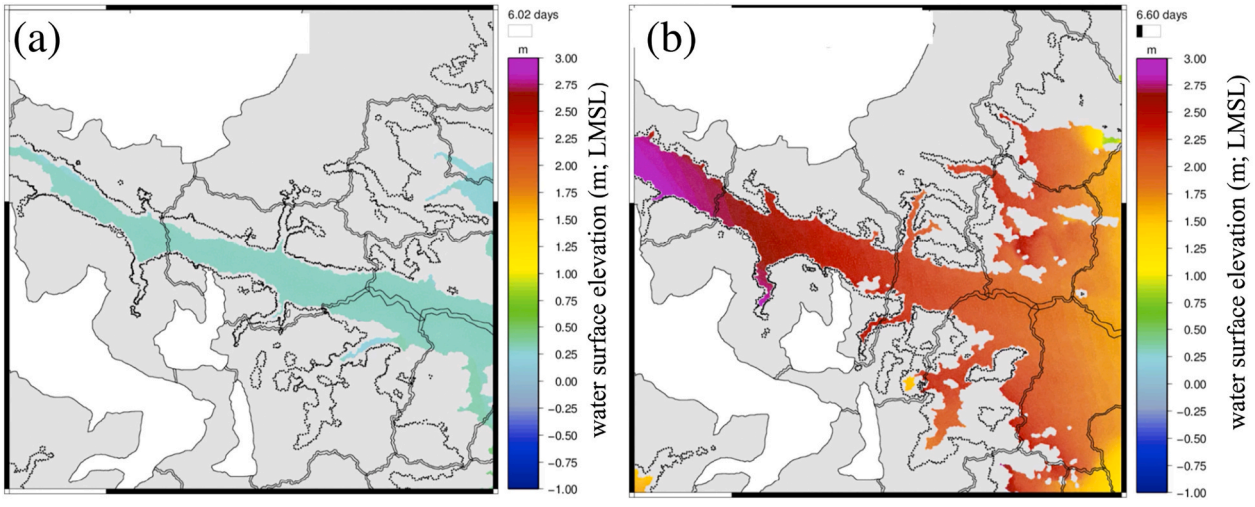

Fig. 15. The water surface elevation along the Neuse River (c.f., Fig. 10(c)) before (a) and after (b) after landfall of Hurricane Irene. Here we highlight the complex progression of the wet/dry boundary during the coastal flood event. Solid lines black lines indicate subdomain extents, while the black dots indicate checkpoint vertices.

when the minimum number of vertices-per-core falls below $_{1,\,000}$. In this part of the scaling regime, the communication cost dominates the computing cost and dynamic load balancing offers little gain.

Performance was also significantly affected by the criteria used to determine what portion of the mesh was set as offline. Criteria based on elevation $_{hb}$ was more efficient than using a minimum-distance criteria $_{db}$ from the wet/dry boundary. Minimum-distance criteria produced more rebalance events as the tide advanced shoreward compared to the depth-based criteria, which leads to more time spent rebalancing and overall slower performance. Based on the results, our recommendation is to remove as much of the inter-tidal zone using a topographic criteria (i.e., $_{hb=0.50}$ m or a depth similar to the local tidal range) and let the program automatically lift $_{hb}$ as wind-driven coastal flooding occurs.

We demonstrated that ADCIRC + DLB does not sacrifice the well-established accuracy available in the ADCIRC solver, parallel scalability, or require new meshes/models to be developed. Future work intends to apply ADCIRC + DLB to operational storm tide forecasting systems, including regional models (e.g., Fleming et al., 2008) as well as emerging global ones (e.g., Pringle et al., 2021; Seroka et al., 2020). While some current research is focused on more efficiently incorporating coastal floodplains into modeling systems to avoid components of the mesh that are rarely flooded through mesh decimation techniques (Bilskie et al., 2020), our dynamic load balancing concept may provide a different approach to coastal ocean model development. For instance, dynamic load balancing could reduce the need to develop many regional models with carefully designed floodplains by instead enabling the user to develop one substantial modeling domain with a large overland extent while incurring minimal computational overhead. For example, a next-generation comprehensive modeling system of the East and Gulf Coasts of the United States with fine resolution overland would imply that the vast majority of vertices would remain in a dry-state for any given event. In this case, the size of the problem and the number of dry-state vertices would make it a suitable candidate to take advantage of ADCIRC + DLB to accelerate the calculation.

Funding

This material is based upon work supported by the U.S. Department of Homeland Security under Grant Award Number 2015-ST-061-ND0001-01 and National Science Foundation Grant Award Number NSF ACI-1339738. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security or the National Science Foundation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We thank the two anonymous reviewers who helped improve the quality of the manuscript. We thank Ocean Weather Inc. for allowing us to use their meteorological forcing inputs for the Hurricane Irene test problem. We thank Dr. Brian Blanton at Renaissance Computing Institute at the University of North Carolina at Chapel Hill for providing the mesh and input files used in the Hurricane Irene test problem. KR prepared the manuscript, designed and implemented the coding upgrades into ADCIRC, designed and performed the experiments, and conducted the analysis of the results. JCD, DW, and JJW provided feedback throughout the project, supervised the research and improved the manuscript. WP improved the manuscript presentation and provided critical feedback.

References

Androsov, A., Fofonova, V., Kuznetsov, I., Danilov, S., Rakowsky, N., Harig, S., Brix, H., Wiltshire, K.H., Mar. 2019. FESOM-c v.2: coastal dynamics on hybrid unstructured meshes. Geosci. Model Dev. (GMD) 12 (3), 1009–1028. https://doi.org/10.5194/gmd-12-1009-2019. URL.

Bates, P.D., Hervouet, J.-M., 1999. A new method for moving–boundary hydrodynamic problems in shallow water. Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences 455, 3107–3128, 1988.

Behrens, J., Bader, M., Nov. 2009. Efficiency considerations in triangular adaptive mesh refinement, 1907 Phil. Trans. R. Soc. A. 367, 4577–4589. https://doi.org/10.1098/rsta.2009.0175. URL.

Bilgili, A., Smith, K.W., Lynch, D.R., Jun. 2006. BatTri: a two-dimensional bathymetry-based unstructured triangular grid generator for finite element circulation modeling. Comput. Geosci-uk. 32 (5), 632–642. https://doi.org/10.1016/j.cageo.2005.09.007. URL.

Bilskie, M.V., Coggin, D., Hagen, S.C., Medeiros, S.C., 2015. Terrain-driven unstructured mesh development through semi-automatic vertical feature extraction. Adv. Water Resour. 86, 102–118. URL. http://www.sciencedirect.com/science/article/pii/S0309170815002274.

Bilskie, M.V., Hagen, S.C., 2013. Topographic accuracy assessment of bare earth lidar-derived unstructured meshes. Adv. Water Resour. 52, 165–177. URL. http://www.sciencedirect.com/science/article/pii/S0309170812002503.

Bilskie, M.V., Hagen, S.C., Medeiros, S.C., 2020. Unstructured finite element mesh decimation for real-time hurricane storm surge forecasting. Coast Eng. 156, 103622.

Blanton, B., Dresback, K., Colle, B., Kolar, R., Vergara, H., Hong, Y., Leonardo, N., Davidson, R., Nozick, L., Wachtendorf, T., Apr. 2018. An integrated scenario Ensemble-Based framework for hurricane evacuation modeling: Part 2—hazard modeling. Risk Anal. 40 (1), 117–133. https://doi.org/10.1111/risa.13004. URL.

Blanton, B., McGee, J., Fleming, J., Kaiser, C., Kaiser, H., Lander, H., Luettich, R., Dresback, K., Kolar, R., 2012. Urgent computing of storm surge for North Carolina's coast. In: Procedia Comput. Sci. 9, 1677–1686, Proceedings of the International Conference on Computational Science. ICCS. https://doi.org/10.1016/j.procs.2012.04.185, 2012. URL.

Boman, E.G., Çatalyürek, m.V., Chevalier, C., Devine, K.D., 2012. The zoltan and isorropia parallel toolkits for combinatorial scientific computing: partitioning, ordering and coloring. Sci. Programming-neth. 20 (2), 129–150. https://doi.org/10.1155/2012/713587. URL.

Bunya, S., Dietrich, J., Westerink, J., Ebersole, B., Smith, J., Atkinson, J., Jensen, R., Resio, D., Luettich, R., Dawson, C., Cardone, V., Cox, A., Powell, M., Westerink, H., Roberts, H., Feb. 2010. 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. Mon. Weather Rev. 138 (2), 345–377. https://doi.org/10.1175/2009mwr2906.1. URL.

Candy, A.S., Apr. 2017. An implicit wetting and drying approach for non-hydrostatic baroclinic flows in high aspect ratio domains. Adv. Water Resour. 102, 188–205. https://doi.org/10.1016/j.advwatres.2017.02.004. URL.

Casulli, V., Jan. 2019. Computational grid, subgrid, and pixels. Int. J. Numer. Methods Fluid. 90 (3), 140–155. https://doi.org/10.1002/fld.4715. URL.

Casulli, V., Walters, R.A., Feb. 2000. An unstructured grid, three-dimensional model based on the shallow water equations. Int. J. Numer. Methods Fluid. 32 (3), 331–348. https://doi.org/10.1002/(sici)1097-0363(20000215)32:3<331::aid-fld941>3.0.co;2-c. URL.

Chen, C., Liu, H., Beardsley, R.C., Jan. 2003. An unstructured grid, finite-volume, three-dimensional, primitive equations ocean model: application to coastal ocean and estuaries. J. Atmos. Ocean. Technol. 20 (1), 159–186. https://doi.org/10.1175/1520-0426(2003)020<0159:augfvt>2.0.co;2. URL.

Cialone, M.A., Grzegorzewski, A.S., Mark, D.J., Bryant, M.A., Massey, T.C., Sep. 2017. Coastal-storm model development and water-level validation for the north Atlantic coast comprehensive study. J. Waterw. Port, Coast. Ocean Eng. 143 (5), 04017031 https://doi.org/10.1061/(asce)ww.1943-5460.0000408. URL.

Cobell, Z., Zhao, H., Roberts, H.J., Clark, F.R., Zou, S., 2013. Surge and wave modeling for the Louisiana 2012 coastal master plan. J. Coast Res. (67), 88–108.

Devine, K., Boman, E., Heaphy, R., Hendrickson, B., Vaughan, C., Mar. 2002. Zoltan data management services for parallel dynamic applications. Comput. Sci. Eng. 4 (2), 90–96. https://doi.org/10.1109/5992.988653. URL.

Dietrich, J., Bunya, S., Westerink, J., Ebersole, B., Smith, J., Atkinson, J., Jensen, R., Resio, D., Luettich, R., Dawson, C., Cardone, V., Cox, A., Powell, M., Westerink, H., Roberts, H., Feb. 2010. 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. Mon. Weather Rev. 138 (2), 378–404. https://doi.org/10.1175/2009mwr2907.1. URL.

Dietrich, J., Kolar, R., Luettich, R., 2004. Assessment of ADCIRC's wetting and drying algorithm. of Developments in Water Science. In: Miller, C.T., Pinder, G.F. (Eds.), Computational Methods in Water Resources: Volume 2, Proceedings of the XVth International Conference on Computational Methods in Water Resources, vol. 55. Elsevier, pp. 1767–1778. https://doi.org/10.1016/s0167-5648(04)80183-7. URL.

Dresback, K.M., Fleming, J.G., Blanton, B.O., Kaiser, C., Gourley, J.J., Tromble, E.M., Luettich, R.A., Kolar, R.L., Hong, Y., Van Cooten, S., Vergara, H.J., Flamig, Z.L., Lander, H.M., Kelleher, K.E., Nemunaitis-Monroe, K.L., Dec. 2013. Skill assessment of a real-time forecast system utilizing a coupled hydrologic and coastal hydrodynamic model during hurricane Irene (2011). Continent. Shelf Res. 71, 78–94. https://doi.org/10.1016/j.csr.2013.10.007. URL.

Egbert, G.D., Erofeeva, S.Y., 2019. TPXO9-Atlas. URL. http://volkov.oce.orst.edu/tides/tpxo9_atlas.html.

Fleming, J., Fulcher, C.W., Luettich Jr., R., Estrade, B.D., Allen, G., Winer, H.S., 2008. A real time storm surge forecasting system using ADCIRC. Estuar. Coast. Model. 2009, 893–912.

Forbes, C., Luettich, R.A., Mattocks, C.A., Westerink, J.J., Dec. 2010. A retrospective evaluation of the storm surge produced by hurricane Gustav (2008): forecast and hindcast results. Weather Forecast. 25 (6), 1577–1602. https://doi.org/10.1175/2010waf2222416.1. URL.

Ginting, B.M., Bhola, P.K., Ertl, C., Mundani, R.-P., Disse, M., Rank, E., 2020. Hybrid-parallel simulations and visualisations of real flood and tsunami events using unstructured meshes on high-performance cluster systems. In: Gourbesville, P., Caignaert, G. (Eds.), Advances in Hydroinformatics. Springer Singapore, Singapore, pp. 867–888.

Ginting, B.M., Mundani, R.-P., 2019. Parallel flood simulations for wet–dry problems using dynamic load balancing concept. J. Comput. Civ. Eng. 33 (3), 04019013.

Gorman, G., Piggott, M., Pain, C., May 2007. Shoreline approximation for unstructured mesh generation. Comput. Geosci-uk 33 (5), 666–677. https://doi.org/10.1016/j.cageo.2006.09.007.

Gorman, G., Piggott, M., Wells, M., Pain, C., Allison, P., Dec. 2008. A systematic approach to unstructured mesh generation for ocean modelling using GMT and terreno. Comput. Geosci-uk 34 (12), 1721–1731. https://doi.org/10.1016/j.cageo.2007.06.014. URL.

Greenberg, D.A., Shore, J.A., Page, F.H., Dowd, M., 2005. A finite element circulation model for embayments with drying intertidal areas and its application to the quoddy region of the bay of fundy. Ocean Model. 10 (1), 211–231 (the Second International Workshop on Unstructured Mesh Numerical Modelling of Coastal, Shelf and Ocean Flows).

Hope, M., Westerink, J., Kennedy, A., Kerr, P., Dietrich, J., Dawson, C., Bender, C., Smith, J., Jensen, R., Zijlema, M., Holthuijsen, L., Luettich, R., Powell, M., Cardone, V., Cox, A., Pourtaheri, H., Roberts, H., Atkinson, J., Tanaka, S., Westerink, H., Westerink, L., Sep. 2013. Hindcast and validation of hurricane Ike (2008) waves, forerunner, and storm surge. J. Geophys. Res. Oceans 118 (9), 4424–4460. https://doi.org/10.1002/jgrc.20314. URL.

Karypis, G., Kumar, V., Jan. 1998. A parallel algorithm for multilevel graph partitioning and sparse matrix ordering. J. Parallel Distr. Comput. 48 (1), 71–95. https://doi.org/10.1006/jpdc.1997.1403. URL.

Kellogg, R., Sep. 1988. The shallow water wave equations: formulation, analysis and application (I. kinnmark). SIAM Rev. 30 (3), 517–518. https://doi.org/10.1137/1030116. URL.

Kennedy, A.B., Wirasaet, D., Begmohammadi, A., Sherman, T., Bolster, D., Dietrich, J., Dec. 2019. Subgrid theory for storm surge modeling. Ocean Model. 144 (October), 101491. https://doi.org/10.1016/j.ocemod.2019.101491. URL.

Khalid, A., Ferreira, C.M., Sep. 2020. Advancing real-time flood prediction in large estuaries: iflood a fully coupled surge-wave automated web-based guidance system. Environ. Model. Software 131, 104748. https://doi.org/10.1016/j.envsoft.2020.104748. URL.

Kärnä, T., de Brye, B., Gourgue, O., Lambrechts, J., Comblen, R., Legat, V., Deleersnijder, E., 2011. A fully implicit wetting–drying method for dg-fem shallow water models, with an application to the scheldt estuary. Comput. Methods Appl. Mech. Eng. 200 (5), 509–524. URL. http://www.sciencedirect.com/science/article/pii/S0045782510002136.

LeVeque, R.J., George, D.L., Berger, M.J., 2011a. Tsunami modelling with adaptively refined finite volume methods. Apr Acta Numer. 20, 211–289. https://doi.org/10.1017/s0962492911000043. URL.

LeVeque, R.J., George, D.L., Berger, M.J., 2011b. Tsunami modelling with adaptively refined finite volume methods. Apr Acta Numer. 20, 211–289. https://doi.org/10.1017/s0962492911000043. URL.

Luettich, R., Westerink, J.J., 2004. Formulation and Numerical Implementation of the 2D/3D ADCIRC Finite Element Model Version 44.XX. Tech. Rep. URL. https://adcirc.org/files/2018/11/adcirc%5c_theory%5c_2004%5c_12%5c_08.pdf.

Luettich Jr., R., Westerink, J., Jan. 1999. Elemental Wetting and Drying in the ADCIRC Hydrodynamic Model: Upgrades and Documentation for ADCIRC version 34.xx.

Lynch, D.R., Gray, W.G., Sep. 1979. A wave equation model for finite element tidal computations. Comput. Fluids 7 (3), 207–228. https://doi.org/10.1016/0045-7930(79)90037-9. URL.

Marks, D., Elmore, P., Blain, C.A., Bourgeois, B., Petry, F., Ferrini, V., Dec. 2017. A variable resolution right TIN approach for gridded oceanographic data. Comput. Geosci-uk. 109, 59–66. https://doi.org/10.1016/j.cageo.2017.07.008. URL.

Medeiros, S.C., Hagen, S.C., 2012a. Review of wetting and drying algorithms for numerical tidal flow models. Mar Int. J. Num. Methods Fluid. 71 (4), 473–487. https://doi.org/10.1002/fld.3668. URL.

Medeiros, S.C., Hagen, S.C., 2012b. Review of wetting and drying algorithms for numerical tidal flow models. Mar Int. J. Num. Methods Fluid. 71 (4), 473–487. https://doi.org/10.1002/fld.3668. URL.

Neelz, S., Pender, G., 10 2008. Grid Resolution Dependency in Inundation Modelling, pp. 109–117.

Pinar, A., Hendrickson, B., 2001. Communication Support for Adaptive Computation.

Pringle, W.J., Wirasaet, D., Roberts, K.J., Westerink, J.J., 2021. Global storm tide modeling with ADCIRC v55: unstructured mesh design and performance. Geosci. Model Dev. 14, 1125–1145. https://doi.org/10.5194/gmd-14-1125-2021.

Quetzalcóatl, O., González, M., Cánovas, V., Medina, R., Espejo, A., Klein, A., Tessler, M., Almeida, L., Jaramillo, C., Garnier, R., Kakeh, N., González-Ondina, J., Jun. 2019. SMC, a coastal modeling system for assessing beach processes and coastal interventions: application to the Brazilian coast. Environ. Model. Software 116, 131–152. https://doi.org/10.1016/j.envsoft.2019.03.001. URL.

Roberts, K.J., Pringle, W.J., Westerink, J.J., 2019a. OceanMesh2D 1.0: matlab-based software for two-dimensional unstructured mesh generation in coastal ocean modeling. May Geosci. Model Dev. (GMD) 12 (5), 1847–1868. https://doi.org/10.5194/gmd-12-1847-2019. URL.

Roberts, K.J., Pringle, W.J., Westerink, J.J., Contreras, M.T., Wirasaet, D., 2019b. On the automatic and a priori design of unstructured mesh resolution for coastal ocean circulation models. Dec Ocean Model. 144, 101509. https://doi.org/10.1016/j.ocemod.2019.101509. URL.

Sanders, B.F., Schubert, J.E., Detwiler, R.L., 2010. Parbrezo: a parallel, unstructured grid, godunov-type, shallow-water code for high-resolution flood inundation modeling at the regional scale. Adv. Water Resour. 33 (12), 1456–1467.

Sebastian, A., Proft, J., Dietrich, J.C., Du, W., Bedient, P.B., Dawson, C.N., Jun. 2014. Characterizing hurricane storm surge behavior in Galveston bay using the SWAN+ADCIRC model. Coast. Eng. 88, 171–181. https://doi.org/10.1016/j.coastaleng.2014.03.002. URL.

Seroka, G., Miles, T., Xu, Y., Kohut, J., Schofield, O., Glenn, S., Sep. 2016. Hurricane Irene sensitivity to stratified coastal ocean cooling. Mon. Weather Rev. 144 (9), 3507–3530. https://doi.org/10.1175/mwr-d-15-0452.1. URL.

Seroka, G., Vinogradov, S., Funakoshi, Y., Myers, E., Moghimi, S., Contreras, M.T., Pringle, W., Westerink, J., Calzada, J., Tang, L., Pe'eri, S., Britzolakis, G., 2020. Extratropical surge tide operational forecast system (ESTOFS): global upgrade, future development, pacific enhancement. In: AGU Fall Meeting 2020. URL. https://agu2020fallmeeting-agu.ipostersessions.com/?s=6E-47-F5-D2-36-73-9A-97-6B-C3-9E-34-E7-04-9B-5A.

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

Taeb, P., Weaver, R.J., Feb. 2019. An operational coastal forecasting tool for performing ensemble modeling. Estuar. Coast Shelf Sci. 217, 237–249. https://doi.org/10.1016/j.ecss.2018.09.020. URL.

Tanaka, S., Bunya, S., Westerink, J., Dawson, C., Luettich, R., Jul. 2010. Scalability of an unstructured grid continuous galerkin based hurricane storm surge model. J. Sci. Comput. 46 (3), 329–358. https://doi.org/10.1007/s10915-010-9402-1. URL.

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

Teng, J., Jakeman, A., Vaze, J., Croke, B., Dutta, D., Kim, S., Apr. 2017. Flood inundation modelling: a review of methods, recent advances and uncertainty analysis. Environ. Model. Software 90, 201–216. https://doi.org/10.1016/j.envsoft.2017.01.006. URL.

Warner, J.C., Defne, Z., Haas, K., Arango, H.G., Aug. 2013. A wetting and drying scheme for ROMS. Comput. Geosci-uk 58, 54–61. https://doi.org/10.1016/j.cageo.2013.05.004. URL.

Wittmann, R., Bungartz, H.-J., Neumann, P., 2017. High performance shallow water kernels for parallel overland flow simulations based on fullswof2d. Comput. Math. Appl. 74 (1), 110–125, 5th European Seminar on Computing ESCO 2016.