https://doi.org/10.1038/s44304-024-00036-5

SWEMniCS: a software toolbox for modeling coastal ocean circulation, storm surges, inland, and compound flooding

Clint Dawson[1], Mark Loveland[2] ✉, Benjamin Pachev[1], Jennifer Proft[1] & Eirik Valseth[1,3,4]

Flooding from storm surges, rainfall-runoff, and their interaction into compounding events are major natural hazards in coastal regions. To assess risks of damages to life and properties alike, numerical models are needed to guide emergency responses and future assessments. Numerical models, such as ADCIRC have over many decades shown their usefulness in such assessments. However, these models have a high threshold in terms of new user engagement as development and compilation is not trivial for users trained in compiled programming languages. Here, we develop a new open-source finite element solver for the numerical simulation of flooding. The numerical solution of the underlying PDEs is developed using the finite element framework FEniCSx. The goal is a framework where new methods can be rapidly tested before time-consuming development into codes like ADCIRC. We validate the framework on several test cases, including large-scale computations in the Gulf of Mexico for Hurricane Ike (2008).

Flooding in coastal and estuarine areas, due to the impacts of tropical cyclones, poses significant risks due to the resulting storm surge and intense rainfall runoff. This devastating phenomenon not only threatens human life and property but also significantly impacts wildlife, warranting constant scrutiny from researchers, policymakers, emergency managers and responders, industries, and governments. The impacts of these events in the United States are tracked and documented in The "U.S. Billion-dollar Weather and Climate Disasters" dataset from the United States National Oceanographic and Atmospheric Administration (NOAA) National Centers for Environmental Information[1,2]. The title alone gives an indication of the financial impacts in addition to the resulting human and animal deaths from these events. Hence, the capability to understand and simulate the dynamics of flooding from tropical cyclone events, including fluvial, pluvial, and coastal, is critical to all those affected by these events.

To develop a new, open-source numerical model for flooding induced by tropical cyclones, we apply the shallow water equations (SWE). The SWE is a set of nonlinear conservation laws that describe the large-scale physics of a fluid layer and have proved useful in predicting the behavior of fluids in the atmosphere, oceans, rivers, and estuaries[3,4]. In recent history, the use of the SWE has been successfully used in numerical models forecasting water levels and water velocities on coastlines during hurricanes and remains a highly active area of research[5–12]. As nonlinear conservation laws, the SWE in practice needs to be solved approximately using numerical techniques. Historically, this has been achieved by finite volume methods[13], finite difference methods[14], finite element methods[15,16], and spectral[17] approaches.

Over the last several decades, numerous numerical SWE models have been developed that use finite element methods[18]. These methods are favored by researchers because they allow straightforward use of unstructured mesh discretizations of the computational domain. This is a significant advantage in representing irregular geometries encountered in modeling coastal regions accurately. Finite element methods also offer extensive flexibility in the type of approximation (e.g., continuous or discontinuous), possibility for adaptive mesh refinement[19,20], and use of parallel computing environments.

Several numerical models utilize a Bubnov-Galerkin or continuous Galerkin (CG) finite element approach such as in the ADvanced CIRCulation (ADCIRC) model[5,15], Adaptive Hydraulics (AdH)[16], and TELEMAC[21]. Some models implement a discontinuous Galerkin (DG) finite element approach such as DG-SWEM[22,23] and Thetis[24]. The hybridized version of the DG method is also used for the SWE as it has the potential to reduce the computational cost incurred in the pure DG methods[25]. Furthermore, there are models employing combined continuous and discontinuous finite element approaches[26,27], as well as mixed finite element-finite volume schemes such as CaMel[28] and SCHISM[29]. We also

[1]Oden Institute for Computational Engineering and Sciences, The University of Texas at Austin, Austin, TX, USA. [2]Information and Technology Laboratory, Engineering Research and Development Center, US Army Corps of Engineers, Vicksburg, MS, USA. [3]The Department of Data Science, The Norwegian University of Life Science, Ås, Norway. [4]Department of Scientific Computing and Numerical Analysis, Simula Research Laboratory, Oslo, Norway.
✉e-mail: markloveland6@gmail.com

note that finite element methods may suffer from loss of discrete numerical stability due to the structure of the SWE and its corresponding weak formulations. Hence, particular care in the design of finite element models for the SWE must be taken, e.g., through upwind fluxes in DG-based methods or Streamlined Upwind Petrov-Galerkin (SUPG) stabilization in CG-based methods[30].

The vast majority of SWE finite element models are developed from the ground up and include custom implementations of finite element assembly routines, quadrature, and meshing. This is extremely time-intensive and results in codes that are difficult to modify. In recent years, finite element frameworks such as MFEM[31], deal.II[32], libCEED[33], DUNE[34], FreeFem++[35], libMesh[36], NGSolve[37] and FEniCSx[38] have gained prominence. These frameworks automate functions common to all finite-element-based codes and enable faster development of more flexible codes. Among these, FEniCSx[38–41] is uniquely suited for rapid prototyping and extensibility. It allows variational forms to be specified symbolically in Python using the Unified Form Language (UFL)[40]. These symbolic representations are then compiled into optimized, MPI-parallel C finite element assembly routines, which can be invoked using either the C++ or Python bindings provided by FEniCSx. This enables researchers to develop scalable and performant finite element solvers without needing to write C code. The overwhelming majority of the work is offloaded to compiled languages, with minimal overhead from Python. The use of Python wrappers additionally simplifies integration of a finite element solver with invasive reduced order methods, derivative-informed inverse methods, or machine-learning algorithms.

In this study, we describe our software SWEMniCS, a FEniCSx-based finite element solver for the 2D depth-integrated SWE. In "Results" section, we describe five test cases and numerical results used to validate the accuracy and performance of SWEMniCS. The first test case validates the convergence of our numerical schemes to an analytic solution, the second test case demonstrates the behavior of our methods in the presence of a discontinuity, the third test case demonstrates the ability of our model to handle rainfall, the fourth demonstrates SWEMniCS' ability to handle wetting and drying, and the final test case is of storm surge during a hurricane in the Gulf of Mexico which we use to demonstrate both scalability and performance. Following these results, in "Discussion" section we discuss our findings and consider opportunities for further research. In "Methods" section we describe the specific finite element techniques currently available in SWEMniCS. Through the use of FEniCSx, we are able to create solvers that utilize novel numerical schemes including a novel DG solver that is fully implicit in time and capable of handling wetting and drying, as well as a combined DG-CG scheme. Additionally, we are able to replicate a SUPG-based scheme as in ref. 16.

To the knowledge of the authors, the combination of the fully implicit BDF2 time-step scheme along with the DG scheme in space using Lax-Friedrichs numerical flux is novel. The combined DG-CG scheme for the full shallow water equations is novel as is the implementation of the SUPG scheme from[16] with an exact Newton nonlinear solver.

The developed software framework and utility programs are all available from https://github.com/UT-CHG/SWEMniCS.

Results

To numerically verify and validate our developed software framework, we investigate its performance using several shallow water test cases. We first consider a case in which a smooth analytical solution is defined to assess the convergence properties of each of the three implemented finite element schemes described in "Methods" section. Following this verification, we present physically relevant tests, including a test of the well-balanced property, a dam break problem, wetting and drying, and a hurricane in the Gulf of Mexico. In this section, we describe details of the various cases and present results.

In all cases where we report solver times and scaling results, we use the Frontera supercomputer at The Texas Advanced Computing Center[42]. Otherwise, we use a MacBook Air 13 laptop with M1 chip. All test cases with instructions on how to run them are available from the git repository (https://github.com/UT-CHG/SWEMniCS).

Smooth analytical solution

To validate the convergence rates of the numerical methods with respect to mesh refinement, we use a simple test case that describes the propagation of smooth harmonic waves through a rectangular domain. This test case was originally formulated in a paper by Lynch and Gray where they derive analytic solutions to a linearization of the shallow water equations under various conditions[43].

We employ this same test case from Lynch and Gray with linear bottom friction and flat bathymetry that is thoroughly described in a paper by Kubatko et al.[22]. In this case, a wave height of 0.3 m with a period that is the same as the M2 tide is forced on the right side boundary of a rectangular domain that is 90 km in the x direction and 45 km in the y direction with a uniform depth of 3 m. The remainder of the boundary is treated with a wall boundary condition. The test case is run from $t = 0$ days to $t = 2$ days with 5 s time steps in order to minimize the impact of discretization errors with respect to time. These analytical solutions are given by:

$$\begin{aligned} \zeta &= \mathrm{Re}\left(\alpha_0 e^{i\omega t}\frac{\cos(\beta x)}{\cos(\beta L)}\right) \\ u &= \mathrm{Re}\left(\frac{-i\omega\alpha_0}{\beta H_0} e^{i\omega t}\frac{\sin(\beta x)}{\cos(\beta L)}\right) \end{aligned} \tag{1}$$

where $\alpha_0 = 0.3\ m$ the tidal wave height, i the complex unit, $L = 90$ km, $\beta = \sqrt{\frac{\omega^2 - \omega\tau i}{gH_0}}$, $\omega = 0.00014051891708$ the frequency of the M2 tide, $\tau = 0.0001$ a damping parameter, and $H_0 = 3$ m the uniform bathymetric depth.

We compare the numerical solution to the analytic solution at $t = 2$ days and compute L^2 and L^∞ errors as we evaluate the solution with element sizes of 15,000 m, 7500 m, 3750 m, and 1875 m. The L^2 error is defined as:

$$\| e\|_{L^2} = \sqrt{\int_\Omega (s - s_h)^2 d\mathbf{x}}, \tag{2}$$

where s represents an analytic solution, e.g., either solution given in (1), s_h an approximate FE solution, and Ω the computational domain. The L^∞ error is defined as:

$$\| e\|_{L^\infty} = \max_{i \in \mathcal{N}} |s(x_i, y_i) - s_{h_i}|, \tag{3}$$

where the indexed variables $s(x_i, y_i)$ and s_{h_i} represent function values at the finite set of degrees of freedom in the computational domain, $\mathcal{N}$. The meshes employ triangular elements that are uniform in shape and size on the entire rectangular domain. Lagrange finite elements are used for all schemes; for each level of mesh refinement, the orders of polynomial $p = 1, 2, 3$ are run, resulting in a total of 12 simulations per numerical scheme.

Each of the L^2 and L^∞ errors are summarized in the log-log plots of Fig. 1. Beginning with the top left plot, which depicts all L^2 error values for the velocity field, we demonstrate that for order $p = 1$ the solution converges at the optimal rate of $O(h^2)$ for all schemes. For $p = 2$, the DG and SUPG schemes converge near the optimal rate of $O(h^3)$ while the combined DG-CG scheme converges sub-optimally. For $p = 3$ we demonstrate optimal rates for all three schemes until a limit is reached due to accuracy limitations in the time-stepping scheme. For the velocity field, in general L^2 error is lowest in the DG scheme followed by SUPG. For the L^2 error with respect to h, as seen on the top right plot in Fig. 1, similar behavior in terms of convergence rates is observed; in general SUPG error is the lowest followed by the DG scheme. With

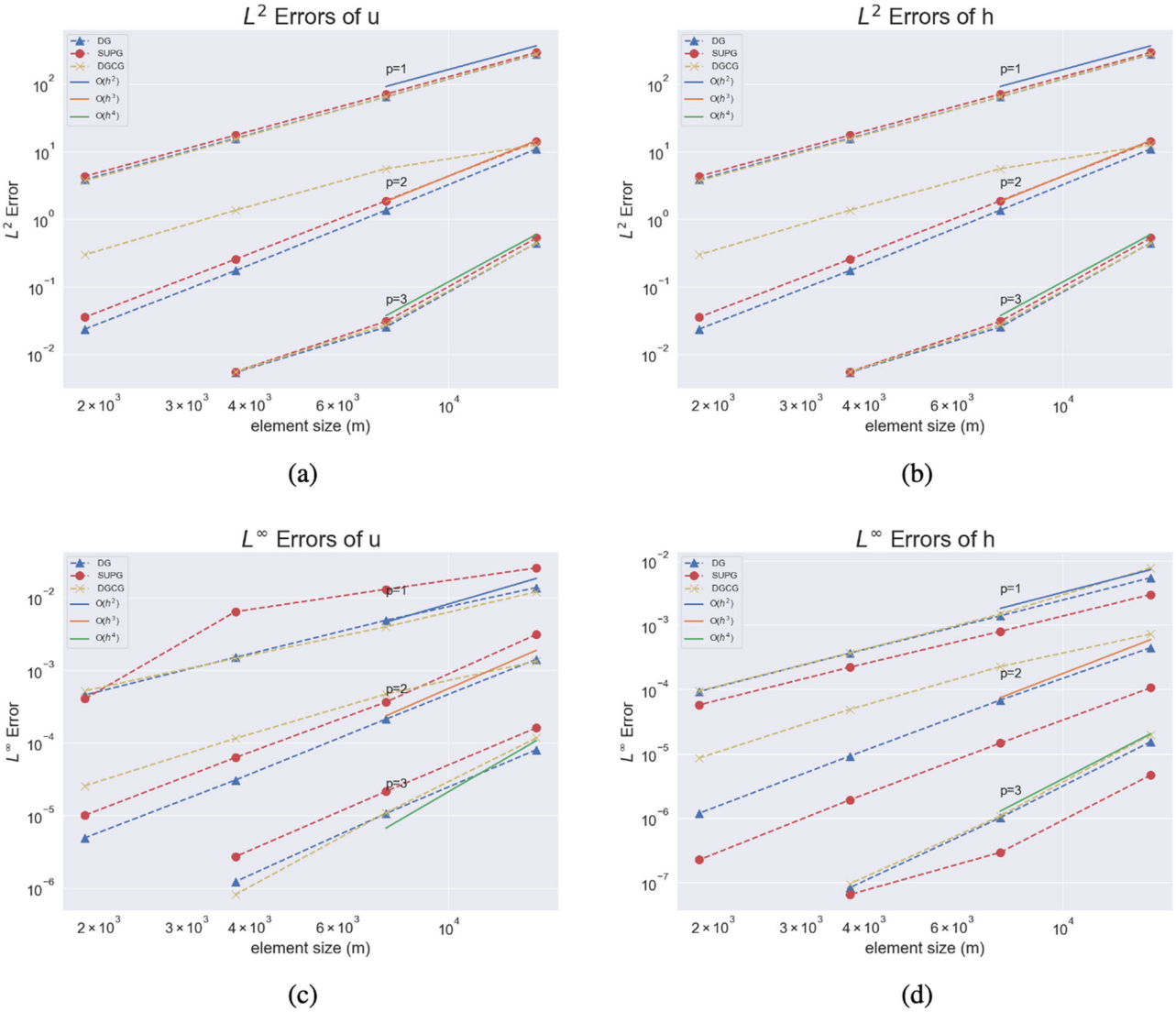

Fig. 1 | L^2 and L^∞ errors compared to an exact solution. Error at $t = 2$ days for the smooth analytic test for each scheme subject to mesh refinement. **a** L^2 errors for velocity. **b** L^2 errors for depth. **c** L^∞ errors for velocity. **d** L^∞ errors for depth.1

regards to the L^∞ errors (bottom two plots in Fig. 1), we see similar properties to the L^2 errors in that the error for velocity is generally lowest for DG followed by SUPG, while the error for h is generally lowest for SUPG followed by DG.

The error rates are contained in tables for each scheme and each order of p. In Table 1 the L^2 error rates for h are given. In general, near-optimal rates are shown for all schemes except for the combined DG-CG scheme for polynomial order higher than 1, which intuitively matches error plots depicted in Fig. 1. In Table 2 the L^2 error rates for velocity are shown. Similar to h, near-optimal rates occur for all schemes at each order, with the exception of DG-CG scheme for p of order higher than 1. The L^∞ error convergence rates are shown in Tables 3 and 4 for depth and velocity respectively.

Modified lake-at-rest

To test the well-balanced properties of our finite element schemes and their applicability in the presence of nonzero source terms, we consider a modification of the classical lake-at-rest test case[44] as we have previously done in ref. 23. To this end, we modify the right-hand side of the continuity equation in the conservative SWE (6) with a positive nonzero rainfall source. This results in the addition of a rainfall source in definition of the forcing vector of (9) so that it now becomes:

$$\mathbf{f}(\mathbf{U}) = \begin{pmatrix} -r \\ -g\zeta\frac{\partial h_B}{\partial x} + \tau_B uH - fvH - C_D\frac{\rho_{\text{air}}}{\rho_{water}} \parallel \mathbf{w} \parallel w_x + \frac{H}{\rho_{\text{water}}}\frac{\partial P}{\partial x} \\ -g\zeta\frac{\partial h_B}{\partial y} + \tau_B vH + fuH - C_D\frac{\rho_{\text{air}}}{\rho_{\text{water}}} \parallel \mathbf{w} \parallel w_y + \frac{H}{\rho_{\text{water}}}\frac{\partial P}{\partial y} \end{pmatrix}, \tag{4}$$

where r is the rainfall rate in units of m/s. The rainfall rate may vary in space and time. We consider a rectangular domain (0, 50,000 m) × (0, 8000 m) discretized with 200 triangular elements (25 in the horizontal direction, 4 in the vertical), and set the bathymetry to be:

$$h_b(x, y) = 5 - e^{-100(x-25000m)^2}, \tag{5}$$

creating a "bump" at the center as shown in Fig. 2.

We assume that the bottom has a constant quadratic friction coefficient of 0.001, set the time step size to 5 s, and apply zero-flow boundary conditions on all domain edges. The initial conditions are set uniformly throughout the domain with $\zeta = 0$ m and $u = v = 0$ m/s everywhere. We incorporate a constant rainfall rate of 7.0556×10^{-6} m/s (1 inch/hour) for

Table 1 | L^2 error convergence rates for depth

	DG			SUPG			DG-CG		
elem length	$p=1$	$p=2$	$p=3$	$p=1$	$p=2$	$p=3$	$p=1$	$p=2$	$p=3$
15,000	–	–	–	–	–	–	–	–	–
7500	2.0362	2.8676	3.7588	2.1109	3.2057	3.2952	2.1220	1.6421	3.9450
3750	2.0188	2.9531	–	2.0471	2.9939	–	2.0315	2.0307	–
1875	2.0089	2.8508	–	2.0236	2.0874	–	2.0118	2.4139	–

Table 2 | L^2 error convergence rates for velocity

	DG			SUPG			DG-CG		
elem length	$p=1$	$p=2$	$p=3$	$p=1$	$p=2$	$p=3$	$p=1$	$p=2$	$p=3$
15,000	–	–	–	–	–	–	–	–	–
7500	2.0958	3.0154	4.1468	2.0464	2.9138	4.1037	2.1067	1.1637	4.0119
3750	2.0452	2.9714	–	2.0239	2.8771	–	2.0917	2.0458	–
1875	2.0252	2.8742	–	2.0212	2.8466	–	2.0461	2.1860	–

Table 3 | L^∞ error convergence rates for depth

	DG			SUPG			DG-CG		
elem length	$p=1$	$p=2$	$p=3$	$p=1$	$p=2$	$p=3$	$p=1$	$p=2$	$p=3$
15,000	–	–	–	–	–	–	–	–	–
7500	1.9677	2.7097	3.9236	1.9057	2.8586	3.9964	2.3830	1.6829	4.2024
3750	1.9293	2.8882	–	1.8522	2.9363	2.1761	2.0056	2.2094	–
1875	1.9723	2.9514	–	1.9415	3.0927	–	1.9591	2.5070	–

Table 4 | L^∞ error convergence rates for velocity

	DG			SUPG			DG-CG		
elem length	$p=1$	$p=2$	$p=3$	$p=1$	$p=2$	$p=3$	$p=1$	$p=2$	$p=3$
15,000	–	–	–	–	–	–	–	–	–
7500	1.4998	2.7332	2.9185	0.9749	3.0904	2.9124	1.5965	1.5065	3.4239
3750	1.6941	2.7852	–	1.0256	2.5376	–	1.4405	2.0247	–
1875	1.7210	2.6428	–	3.9851	2.6365	–	1.4860	2.1659	–

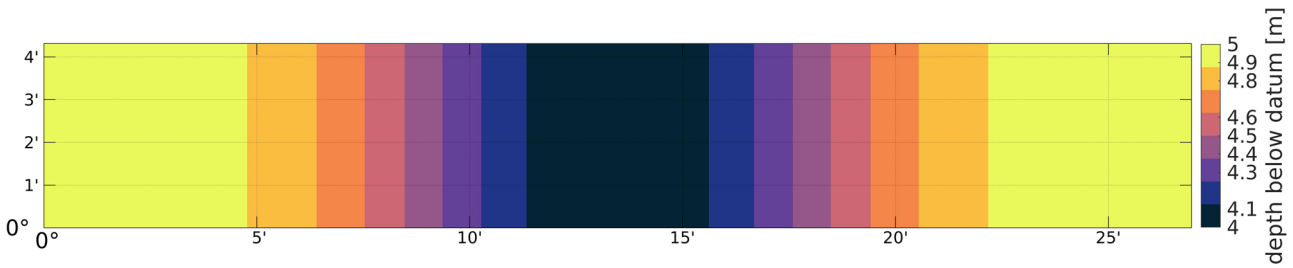

Fig. 2 | Domain of modified lake-at-rest case. Bathymetry of the bathtub mesh reproduced from ref. 23. The bathymetry is flat on either side and contains a smooth symmetric bump in the middle of the domain.

24 h, followed by 48 hours of no forcing. At the conclusion of the simulation, we record the surface elevation and velocity field and compare to the exact solutions, $\zeta = 0.6096$ m and $(u, v) = (0, 0)$ m/s.

At the conclusion of this test case, the exact water surface elevation at the final point in time is 0.60960384 m and the exact velocity field is 0 m/s in both directions. In Table 5 we display the final solution value for each scheme averaged across the entire grid. In all cases, the nonlinear solver tolerance is set to 1×10^{-6} and therefore we would expect the error in the final solution to be solely due to this tolerance level. For each scheme, the final water surface elevation all have errors less than 2×10^{-5} and the velocity values are well below the nonlinear solver tolerance. Hence, we conclude that our solver is well-balanced.

Dam break problem

Next, we apply the numerical schemes to a test case involving a shock in the initial condition, often referred to as a dam break problem in the SWE

literature. The specific dam break case is set up according to the SWASHES test bed in ref. 45. This setup involves a dam break on a fully wet domain without friction. Our motivation for selecting this specific dam break problem is due to the existence of an analytic solution to compare to. The domain is a square of 1000 m by 1000 m and is discretized uniformly by 20,000 triangular elements. The initial condition is set so that left of $x = 500$ m, the water height is 2 m, and right of $x = 500$ m, the water height is 1.5 m. The initial velocity field is set to 0 m/s. The left and right boundaries are set as open boundaries with $h = 2.0$ m and $h = 1.5$ m, respectively. For each scheme, time begins at $t = 0$ s and is advanced to $t = 40$ s with a time step of 0.5 s. The solution of a cross-section along the x direction at $y = 500$ m is recorded for each scheme at $t = 20$ m and $t = 40$ s. As a reference, the continuous Galerkin method without any stabilization terms is included. The solutions at the aforementioned cross sections are plotted for comparison purposes and the root mean square error between the analytic solution and the numerical solutions are tabulated for both $t = 20$ s and $t = 40$ s.

Table 5 | Nodally averaged solution values for each scheme at final point in time for the modified lake-at-rest case

variable	CG	SUPG	DG	DGCG
ζ	0.60962149	0.60962147	0.60962148	0.60962148
u	3.7543e-09	−3.7539e-09	6.0375e-10	−1.6228e-09
v	−6.1010e-09	−5.1234e-09	−5.6806e-09	−7.8594e-10

Figures 3 and 4 depict the solution of all three schemes at $t = 0$ s, $t = 20$ s, and $t = 40$ s along with the relevant analytic solution. In Fig. 3 we plot the solution for the water depth and in Fig. 4 the solution for the velocity in the x-direction. We demonstrate that all three stabilized schemes do suppress oscillations in the solution (due to the discontinuity in the initial condition) in contrast to the continuous Galerkin scheme without stabilization (top-right in both Figs. 3 and 4). We also note that in the solutions for both water depth and velocity, the schemes employing DG discretizations appear to have lower magnitude oscillations local to the discontinuity compared to the SUPG scheme. The oscillations present in the solution of the SUPG scheme can be modified based on the choice of the arbitrary constant α that appears in the definition of τ from (21). For the purposes of this test case, we define $\alpha = 0.5$. The RMSE for each cross-section with respect to the analytic solution is tabulated in Table 6. In general, the purely DG scheme most closely agrees with the analytic solution which is unsurprising as the DG scheme contains the most degrees of freedom compared to the other schemes.

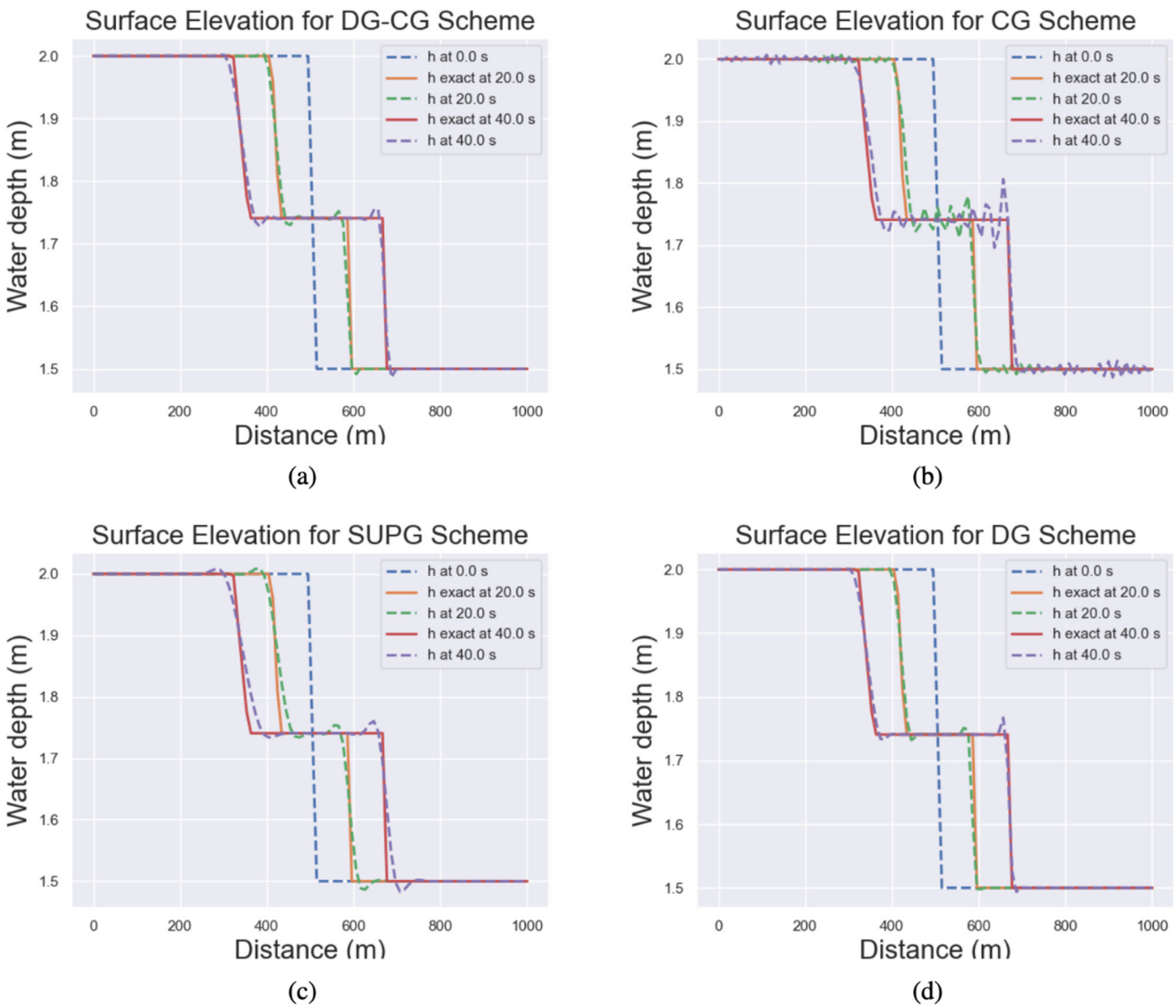

Fig. 3 | Water depths during a dam break. The water height along the x-axis at t = 0, 20, 40 s for all schemes. **a** solution from DG-CG scheme, (**b**) solution from standard CG scheme, (**c**) solution from SUPG scheme, (**d**) solution from DG scheme.

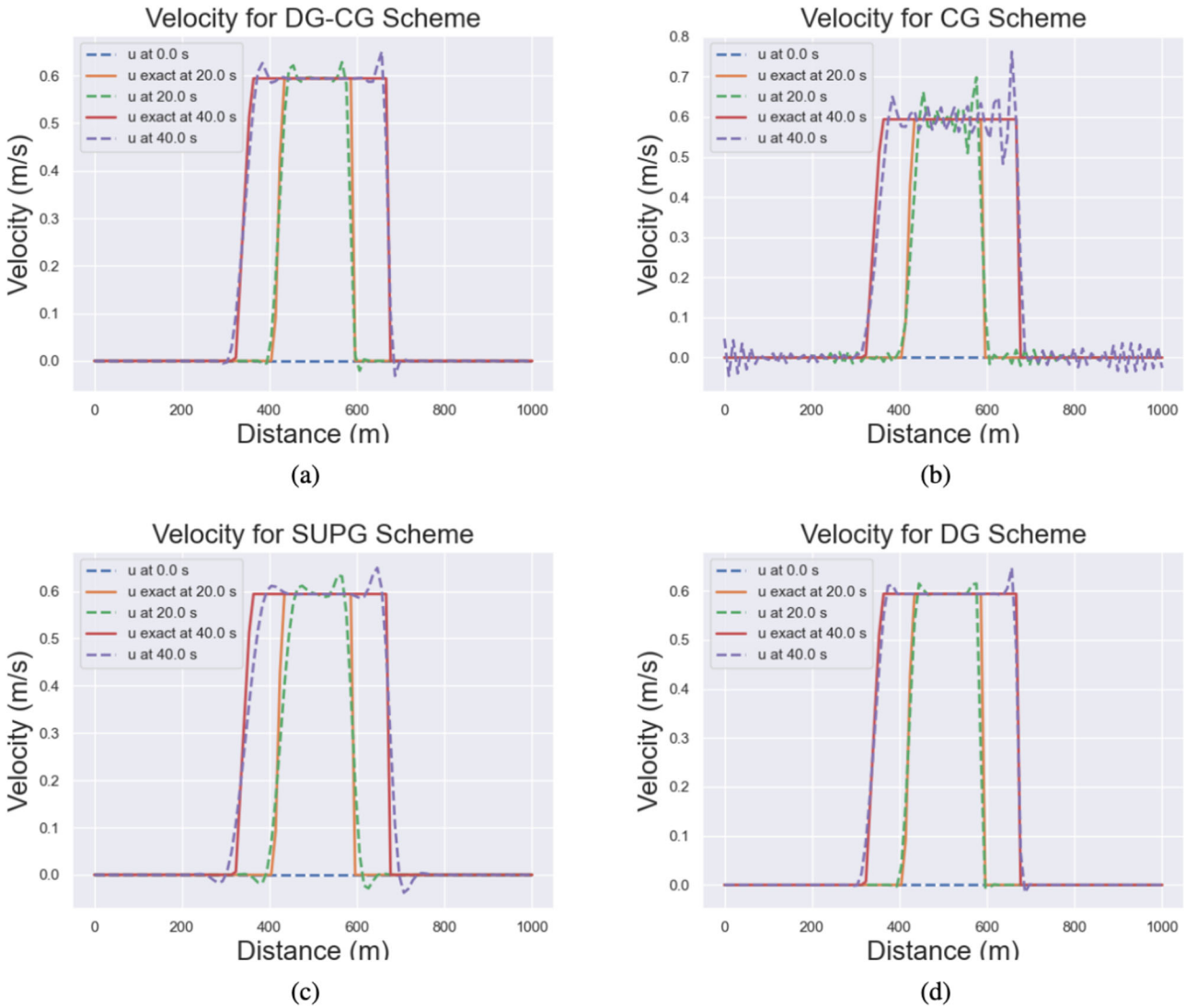

Fig. 4 | Velocities during a dam break. The water velocity along the x-axis at t = 0, 20, 40 s for all schemes. **a** solution from DG-CG scheme, (**b**) solution from CG scheme, (**c**) solution from SUPG scheme, (**d**) solution from DG scheme.

Wetting and drying

This test case demonstrates SWEMniCS' capability of handling wetting and drying by adapting the α scheme of Kärna[46] to work with the DG scheme described in (22). The test case comes originally from an inter-comparison paper by Balzano[47] and subsequently used for testing in ref. 46. The domain is a rectangle that is 13,800 m in the x direction and 7200 m in the y direction. The rectangle is discretized uniformly by 144 triangular elements that have lengths 1150 m in the x direction and 1200 m in the y direction, as in Fig. 5. The bathymetry is a uniformly sloping beach in the x direction that begins 5m below the geoid at $x = 0$ and ends at 0 m relative to the geoid at $x = 13{,}800$ m. The initial conditions are set so that the water is flat at the geoid $\zeta(x, y, 0) = 0$ m and still ($u(x, y, 0) = v(x, y, 0) = 0$m/s). The left boundary is set as an open boundary that is harmonic in time with an amplitude of 2 m and period of 12 h. All other boundaries are set as wall conditions. The simulation is run from $t = 0$ days until $t = 7$ days. The Manning's formula as in (19) is used for the bottom friction term and the coefficient is uniform in the domain with a value of 0.02 s/m$^{1/3}$. Elevations and velocities are recorded over time at three stations along a cross-section within the domain as shown in Fig. 5 at $x = 9000$ m, $x = 11{,}000$ m, and $x = 13{,}500$ m.

SWEMniCS is run with a time step of 600 s with the DG formulation of (22) modified for the wetting and drying scheme described in section "Wetting and drying" with $\alpha = 0.36$. Furthermore, SWEMniCS is run both with the conservative and non-conservative versions of the momentum equations. In order to compare the results from SWEMniCS, the extensively validated SWE model, ADCIRC, is also run for this case but with a time step of 60s due to CFL constraints.

Figures 6–8 show the timeseries of the surface elevations and water velocities at each of the three stations at $x = 9000$ m, $x = 11{,}000$ m, and $x = 13{,}500$ m respectively. Across all three figures, we can see that the difference in results for water depth between the two SWEMniCS configurations is minimal, with a root mean square difference of 9.984×10^{-4} m between conservative and non-conservative momentum formulations. As

Table 6 | RMSE error for each scheme for dam case compared to analytic solution

variable	t(s)	CG	SUPG	DG	DGCG
H	20	0.0156	0.0168	0.0144	0.0138
H	40	0.0168	0.0179	0.0082	0.0098
u	20	0.0387	0.0406	0.0348	0.0346
u	40	0.0439	0.0428	0.0188	0.0231

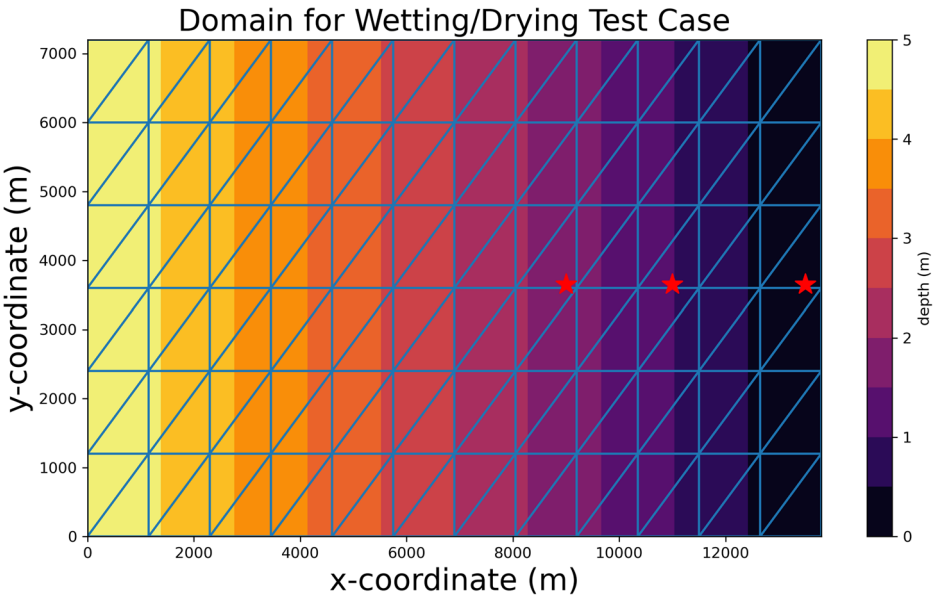

Fig. 5 | Domain of wet/dry case. The recording stations are marked with stars, the bathymetry contours are plotted in color. The left-hand side of the domain begins at a depth of 5 m while the right-hand side has a depth of 0 m. The open boundary that forces the tides is on the left side.

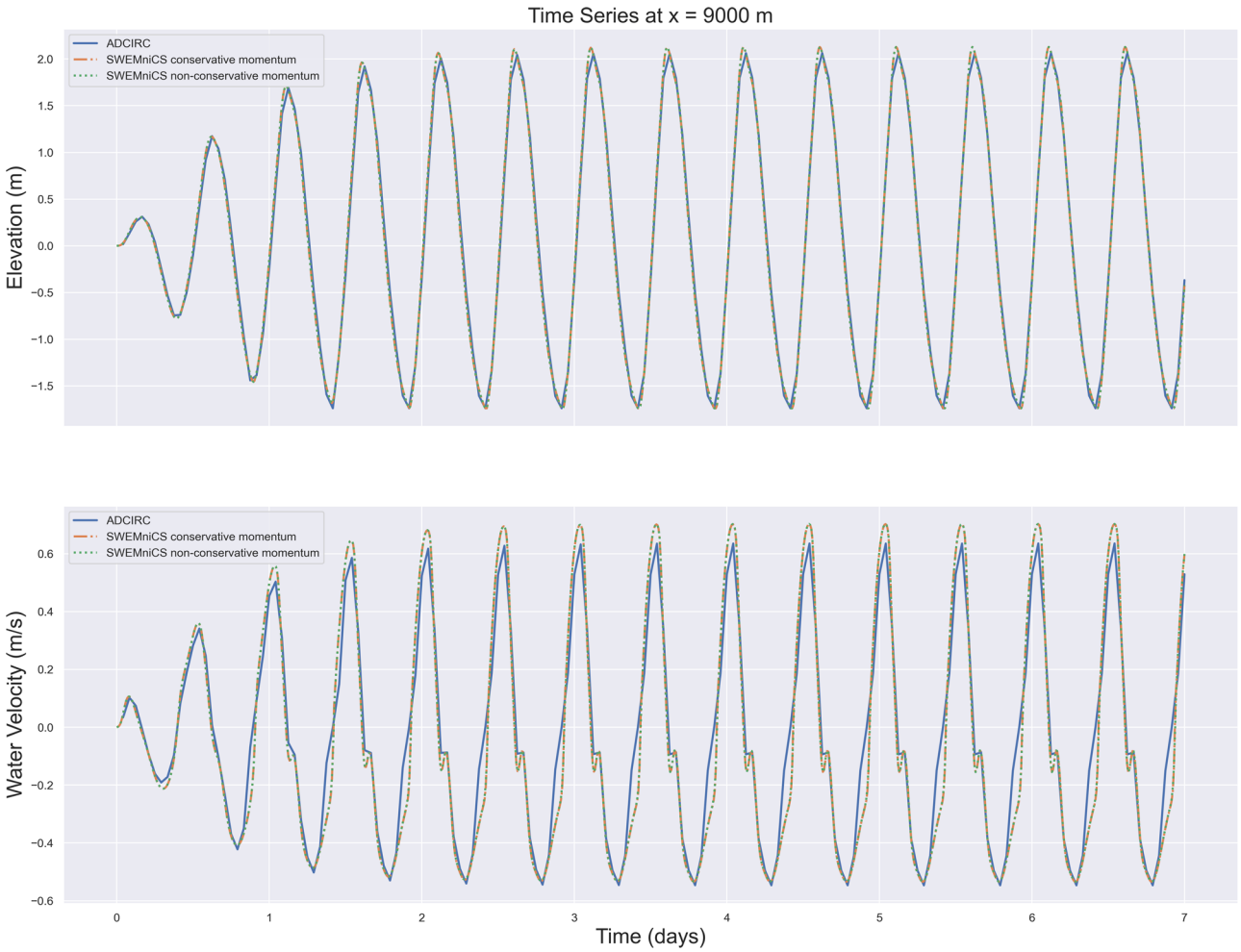

Fig. 6 | Comparison between SWEMniCS and ADCIRC solutions at $x = 9000$ m for wet/dry case. The resulting water depths and velocities for the wet/dry case at the leftmost recording station. The water at this point nearly becomes dry.

the stations get closer to the right of the domain, we do see differences in the velocity profiles between the SWEMniCS configurations. In particular, we can see that in the rightmost station of Fig. 8, the root mean square difference in the velocities is 7.583×10^{-3} m/s. The differences in the other two stations were less at 5.61209×10^{-3} m/s and 4.84593×10^{-3} m/s. In terms of computational performance, it was observed that the usage of the non-conservative momentum equations resulted in an approximately 27 percent faster run time than the conservative momentum equations. This was

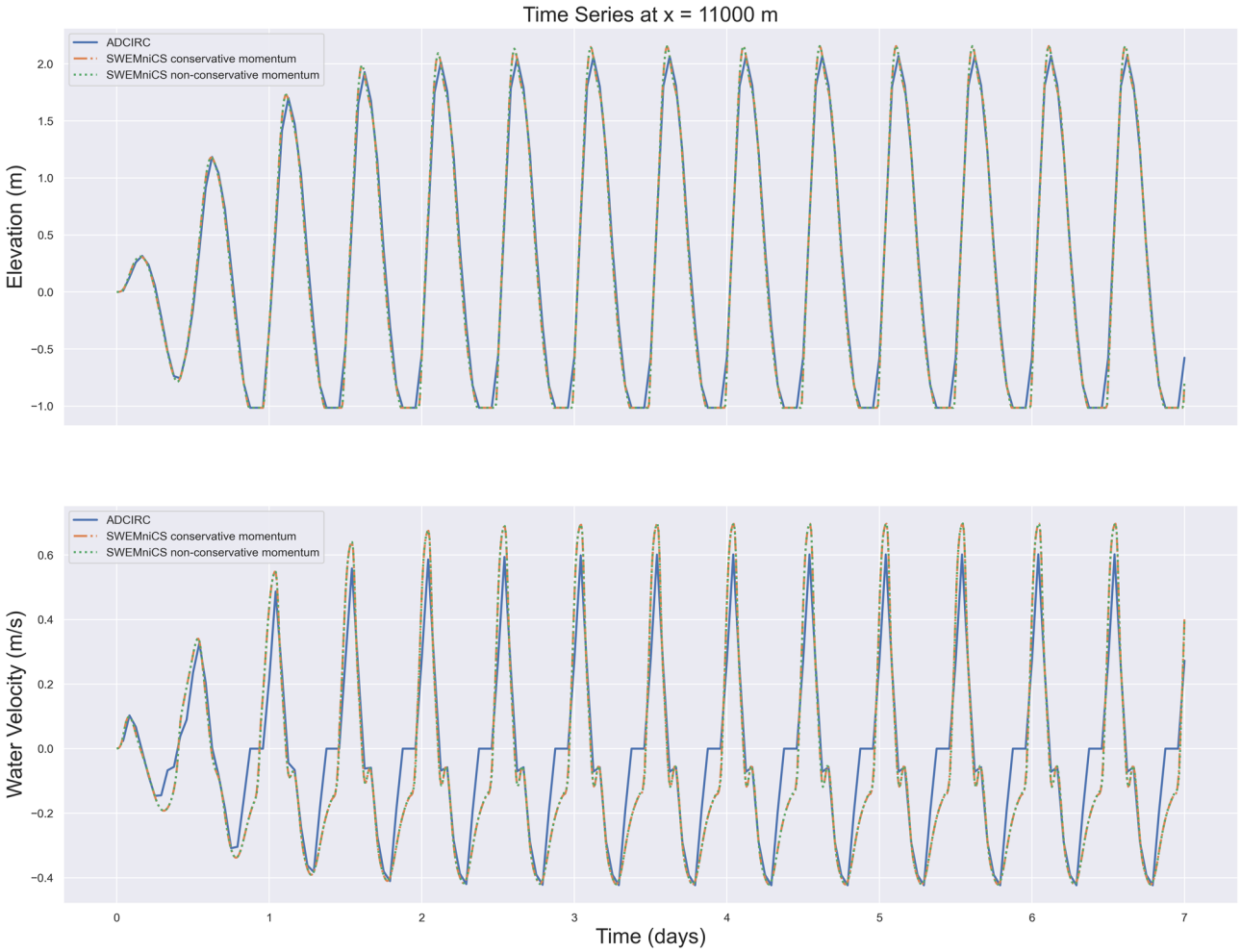

Fig. 7 | Comparison between SWEMniCS and ADCIRC solutions at $x = 11{,}000$ m for wet/dry case. The resulting water depths and velocities for the wet/dry case at the middle recording station. The water at this point does change from wet to dry over the course of the simulation.

because the Newton solver converged with a quarter less iterations on average with the non-conservative momentum equations than with the conservative. Additionally, it was observed that the conservative formulation failed to converge to a solution with time steps larger than 600 s while the non-conservative formulation was able to converge to a solution at time steps as large as 3600 s. This is likely owed to the fact that the formulation with the non-conservative momentum equations reduces the amount of nonlinearity in the problem as seen in "Wetting and drying" section.

When comparing the SWEMniCS results to ADCIRC, the timing of the wetting and the drying at each station appear to be in agreement at all stations. In Fig. 8, we can see that the max water elevations for SWEMniCS are ≈ 0.1 m higher than ADCIRC. This could be due to the required presence of viscosity within the ADCIRC formulation as well as from damping due to ADCIRC's wetting and drying. The differences in the velocity values as a particular station becomes dry are likely due to the fact that ADCIRC's scheme involves activating/deactivating elements depending on their wet or dry state while SWEMniCS operates more like a minimal depth algorithm that always computes a velocity. Despite this, the timings of the maximum/minimum velocities at each station match and the maximum values are all within 7 percent at the leftmost two stations. For the rightmost station, the difference in maximum velocities is approximately 30 percent for the conservative SWEMniCS scheme and ADCIRC while it is approximately 40 percent for the non-conservative SWEMniCS and ADCIRC. This is likely due to a combination of the required inclusion of viscosity in ADCIRC as well as the differences in the wetting/drying scheme.

Hurricane test case

To verify that our models can accurately reproduce real-world physics, we implement a simulation of Hurricane Ike (2008), which peaked as a category 4 storm in the open ocean, and made landfall as a category 2 storm near Galveston, Texas[7]. Hurricane Ike generated peak storm surges of over 5 m and is of particular interest due to the availability of observational data[48]. The mesh used was originally developed for testing ADCIRC, and contains 58,369 triangular elements, see Fig. 9. It covers the entirety of the Gulf of Mexico, as well as the eastern coast of the United States. Mesh resolution is variable with the greatest detail near the coast. Floodplains are not included and the coastal boundary is treated as a wall. Future work will investigate the use of the SWEMniCS model with operational-grade meshes which include detailed floodplains. Such meshes typically have millions of elements and represent the gold standard in hurricane modeling[7].

Due to the size of the physical domain, the numerical schemes must solve the spherically projected shallow water equations as in "Spherical Coordinates" section. For both tidal boundary conditions and the tidal potential forcing term, eight tidal constituents are used (M2, S2, N2, K2, K1, O1, P1, and Q1). Finally, we use Oceanweather Inc. (OWI) meteorological forcing data for Hurricane Ike[49]. The data has a spatial resolution of one-tenth of a degree and a temporal resolution of 15 min over 8.5 days. Despite the lack of wetting and drying, this simulation represents a thorough test of the scalability and physical modeling capabilities of our software.

Quantitative validation is provided by comparing our predicted results to observational data at several locations on the coast. Because the mesh does

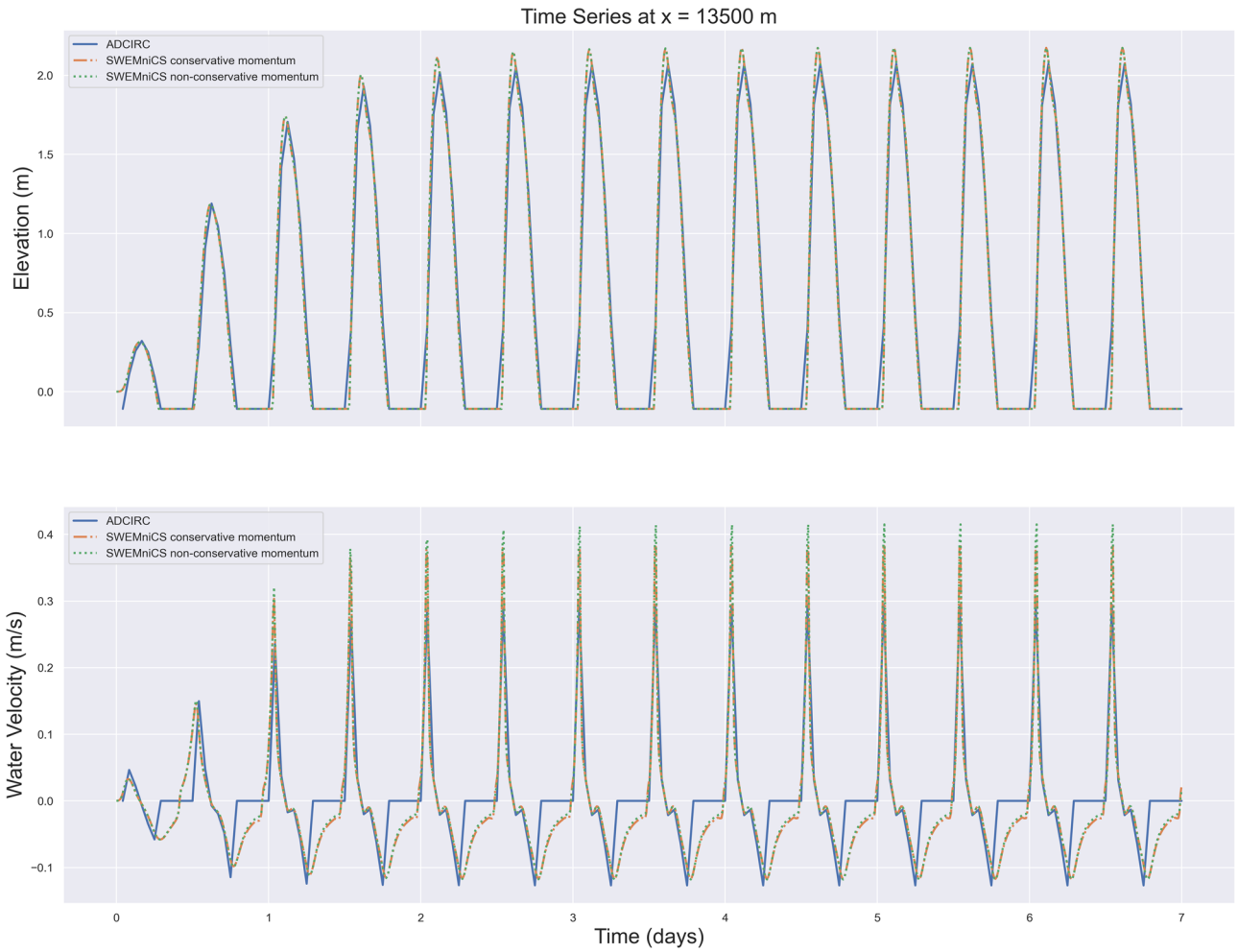

Fig. 8 | Comparison between SWEMniCS and ADCIRC solutions at $x = 13{,}800$ m for wet/dry case. The resulting water depths and velocities for the wet/dry case at the right-most recording station. The water at this point does change from wet to dry over the course of the simulation and spends considerable time in a dry state.

not contain rivers or a significant inland floodplain, much of the available observational data for Hurricane Ike is not applicable. However, we do include a comparison with ADCIRC on the same mesh, to help differentiate between errors in our physical parameterizations and those arising from the limited mesh resolution. The inclusion of ADCIRC also enables us to assess the speed of our shallow water solvers, developed using a general-purpose Python framework, to an optimized, Fortran implementation. We have additionally included video of the Hurricane Ike simulation in the online supplemental material, which provides a further, albeit qualitative validation of our model.

From our extensive numerical testing, we observe that the DG scheme from section "DG Finite Elements" performs the best for this test case and therefore presents only results using the DG scheme in this section. A highly attractive feature of the FEniCSx framework is its built-in MPI parallelism. Assembly of the stiffness matrices is done in parallel by FEniCSx, and the solution of linear systems is handled by PETSc. We observe excellent scaling results for the Hurricane Ike simulation up to 32 processors in Fig. 10.

A major motivation for using an implicit time scheme is the ability to take large timesteps. We find that in this case the timestep can be increased up to 3600 s without adverse impacts on physical accuracy or numerical stability. Although our implementation is stable for larger timesteps, further increasing the timestep causes the tidal signal to be under-resolved. While larger timesteps reduce the overall simulation time, the relationship is not linear, as demonstrated in Fig. 11. Increasing the timestep also results in more poorly conditioned linear systems during the Newton iteration, which therefore requires more iterations to solve.

Between NOAA tidal gauges and other temporary gauges deployed during Hurricane Ike, we are able to obtain observations of water levels at nine locations that lie within the computational domain. In Fig. 12, we plot the hydrographs of our SWEMniCS implementation relative to an ADCIRC simulation run on the same mesh against observational data. Our model attains roughly equivalent accuracy as ADCIRC does. There is a trend towards underprediction of the peak water levels; as these simulations were carried out on a limited-resolution mesh, these results are unsurprising. The main takeaway is that SWEMniCS is capable of carrying out physically realistic simulations.

When using the quadratic friction parameter for SWEMniCS from section "Bottom Friction", a timestep of 3600 s, and 32 MPI processes, the simulation completes in 90 s. In comparison, the ADCIRC simulation uses a timestep of 6 s and completes in 141 s with 32 MPI processes. Increasing the ADCIRC time step further results in numerical instabilities. Visually, the produced maximum elevation profiles are similar (Fig. 13).

Discussion

We have introduced a scalable, open-source finite element solver for the SWE within Python based on the FEniCSx library: SWEMniCS. The solver currently contains multiple novel FE schemes including an SUPG-based scheme similar to ref. 16, a DG scheme with Lax-Friedrichs flux, and an experimental combined DG-CG scheme. All schemes use fully implicit time-stepping schemes based on a generalized BDF2 formula. The resulting nonlinear solver uses the Newton method, taking advantage of FEniCSx' ability to perform symbolic differentiation.

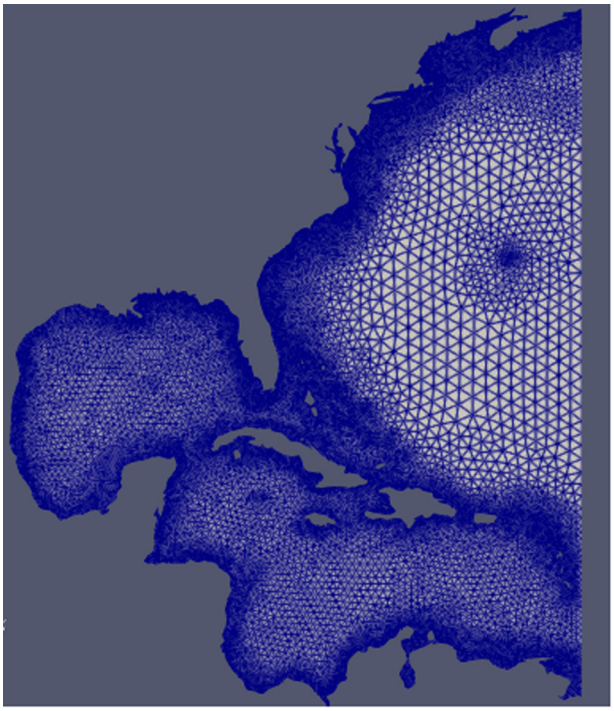

Fig. 9 | Finite element mesh used for the Hurricane Ike simulation test case. The mesh is fully unstructured and comprised of 58,369 triangular elements. The mesh is designed so that the element size decreases as the coastline is approached from the Atlantic Ocean or Gulf of Mexico.

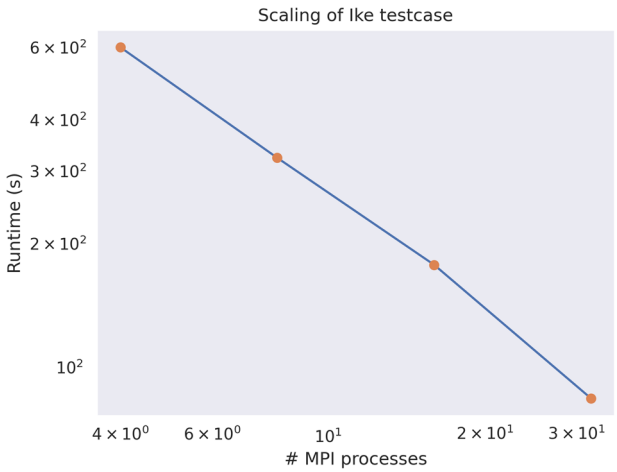

Fig. 10 | Scaling results for the Ike test case. The plot compares MPI processes to overall run time. Ideal scaling was observed up until 30 processes, at which point communication overhead limits performance.

The solver was tested against several test cases that included a simple case with an analytic solution, a modified lake-at-rest, a dam break, a wetting and drying test case, and a simulation of Hurricane Ike. Expected convergence rates toward the analytic solution with respect to h refinement and p enrichment for all schemes were shown. All three numerical schemes were shown to be well-balanced and capable of accounting for on-grid rainfall. Each of the stabilized schemes showed the ability to handle discontinuities by reducing oscillations in the solution. It was demonstrated that the DG scheme can handle wetting and drying cases via the α scheme of ref. 46, with either fully conservative momentum or non-conservative momentum equations. Lastly, the DG scheme showed promising scalability as well as stability and fidelity while taking very large time steps when simulating

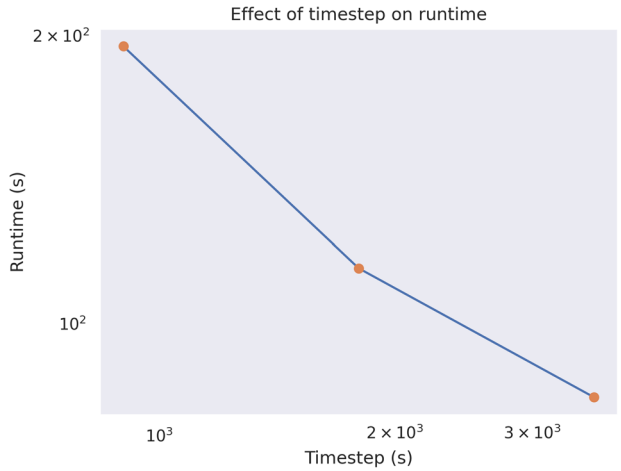

Fig. 11 | Effect of timestep on runtime for the Ike test case. Runtime decreases up until the timestep size of 3600 s. At that point, the underlying Newton solver has difficulty converging from one point in time to the next.

Hurricane Ike in the Gulf of Mexico. Notably, the SWEMniCS model was able to produce results using a time step of 3600 s while ADCIRC required a time step of 6 seconds to remain stable.

A near-term goal is testing the SWEMniCS model on even larger test cases involving wetting and drying. Future work includes investigation into better preconditioners, as well as inclusion of other finite element schemes such as Hybridized DG or discontinuous Petrov-Galerkin (DPG)[50] schemes, and inclusion of explicit or semi-implicit time-stepping schemes. Because this code is written in Python, it could also be a useful tool in exploring novel techniques in inverse problems using e.g., ref. 51, uncertainty quantification, or even augmentation with machine learning libraries and techniques e.g., ref. 52 because there can be direct access to the derivatives of the operators within SWEMniCS via FEniCSx support for automatic differentiation.

Methods

The 2D depth-integrated SWE arises as a consequence of significantly larger horizontal wavelength scales than vertical, a hydrostatic pressure distribution in the water column, and application of boundary conditions on the sea floor and surface. This leads to a modification of the compressive Navier-Stokes equations and can be found in, e.g., the famous text of Vreugdenhil[3]. In this section, we present the forms of the SWE that are used throughout within SWEMniCS, as well as some key physics and their parameterizations, and finally, the finite element methods used.

Conservative form

The conservative 2D depth-integrated SWE on a Cartesian coordinate system can be defined as follows[22]. The specific form is chosen so that any resulting DG finite element schemes are well-balanced in the presence of discontinuous bathymetry.:

$$\begin{aligned}
&\tfrac{\partial H}{\partial t} + div(H\mathbf{u}) = 0,\\
&\tfrac{\partial uH}{\partial t} + \tfrac{\partial}{\partial x}(Hu^2 + \tfrac{1}{2}g(H^2 - h_B^2)) + \tfrac{\partial}{\partial y}(Huv) =\\
&g\zeta\tfrac{\partial h_B}{\partial x} - \tau_B uH + fvH + C_D\tfrac{\rho_{\text{air}}}{\rho_{\text{water}}} \parallel \mathbf{w} \parallel w_x - \tfrac{H}{\rho_{\text{water}}}\tfrac{\partial P}{\partial x},\\
&\tfrac{\partial vH}{\partial t} + \tfrac{\partial}{\partial x}(Huv) + \tfrac{\partial}{\partial y}\left(Hv^2 + \tfrac{1}{2}g(H^2 - h_B^2)\right) =\\
&g\zeta\tfrac{\partial h_B}{\partial y} - \tau_B vH - fuH + C_D\tfrac{\rho_{\text{air}}}{\rho_{\text{water}}} \parallel \mathbf{w} \parallel w_y - \tfrac{H}{\rho_{\text{water}}}\tfrac{\partial P}{\partial y},
\end{aligned} \tag{6}$$

H, u, v are the primitive dependent variables which are respectively water depth, depth-averaged velocity in the x-coordinate, and depth-averaged velocity in the y-coordinate. We define $\zeta = H - h_b$ as the water surface elevation relative to some fixed geoid, h_b the bathymetry of the land surface that remains fixed in time, g as the constant of gravitational acceleration, τ_B

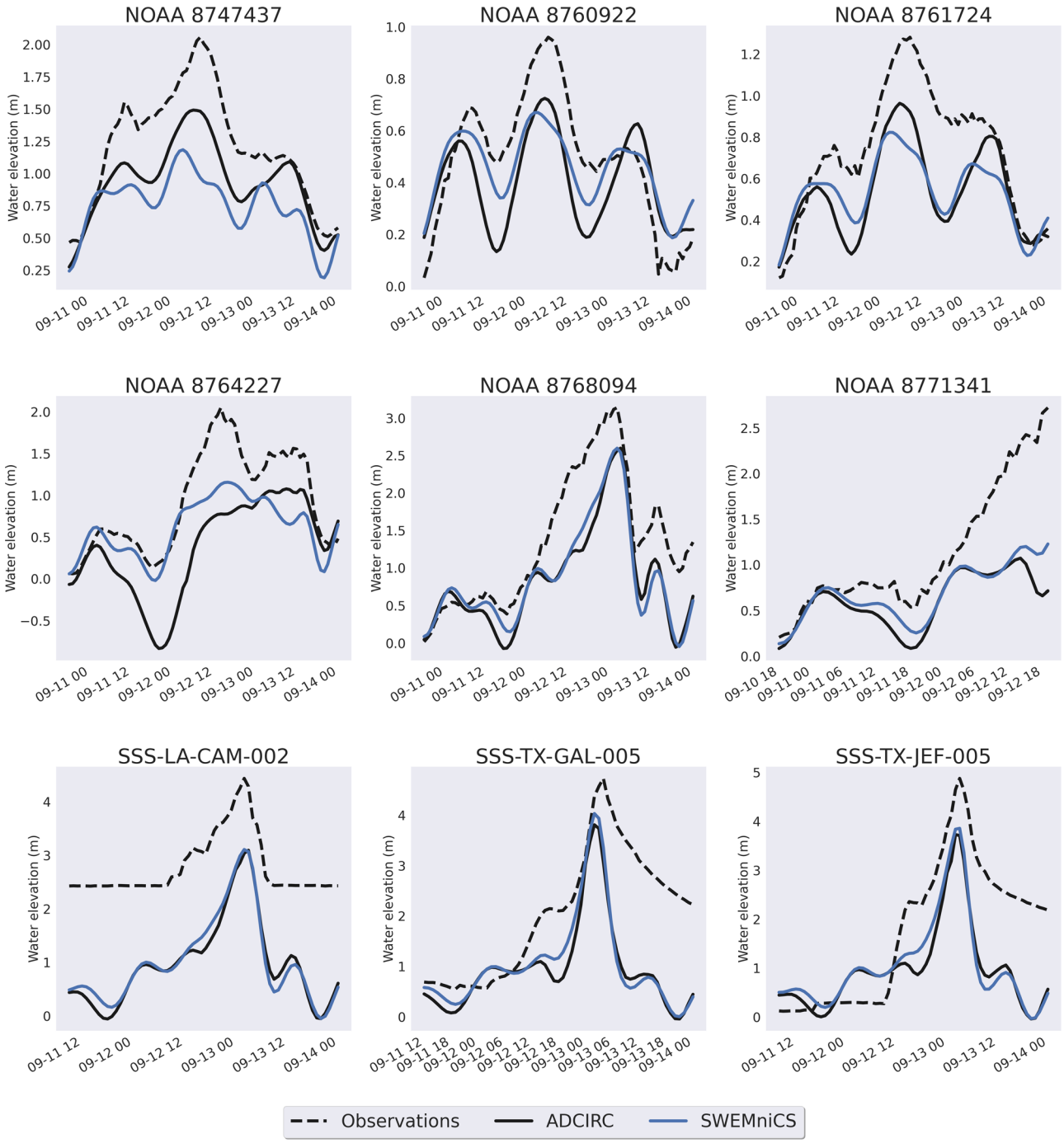

Fig. 12 | Comparison of predicted and observed water levels for Hurricane Ike. The water surface elevation from nine NOAA gauges that were recorded during Ike is plotted in the dashed lines. The water surface elevations from ADCIRC are plotted with black solid lines, and the water surface elevations from SWEMniCS are plotted with blue solid lines.

the bottom friction factor, f the Coriolis parameter, C_D the water surface drag coefficient, ρ_{water} the reference density of water, ρ_{air} the reference density of air, $\mathbf{w} = (w_x, w_y)^T$ the wind velocity vector, and $\frac{\partial P}{\partial x}, \frac{\partial P}{\partial y}$ as the atmospheric pressure gradient. Note that (6) can be rewritten more concisely if we define a tensor for the flux, $\mathbf{F}(\mathbf{U})$, and write the primitive variables as the vector $\mathbf{U} = (H, u, v)^T$:

$$\partial_t \mathbf{Q} + div(\mathbf{F}(\mathbf{U})) + \mathbf{f}(\mathbf{U}) = 0, \tag{7}$$

where $\mathbf{Q} = (H, uH, vH)^T$ is the vector of conserved variables, $\partial_t \mathbf{Q}$ denotes time derivatives of the components of $\mathbf{Q}$, and $div(\mathbf{F}) := \frac{\partial \mathbf{F}_{ij}}{\partial x_j}$. $\mathbf{F}(\mathbf{U})$ comes directly from (6):

$$\mathbf{F}(\mathbf{U}) = \begin{pmatrix} Hu & Hv \\ Hu^2 + \frac{1}{2}g(H^2 - h_b^2) & Huv \\ Huv & Hv^2 + \frac{1}{2}g(H^2 - h_b^2) \end{pmatrix} \tag{8}$$

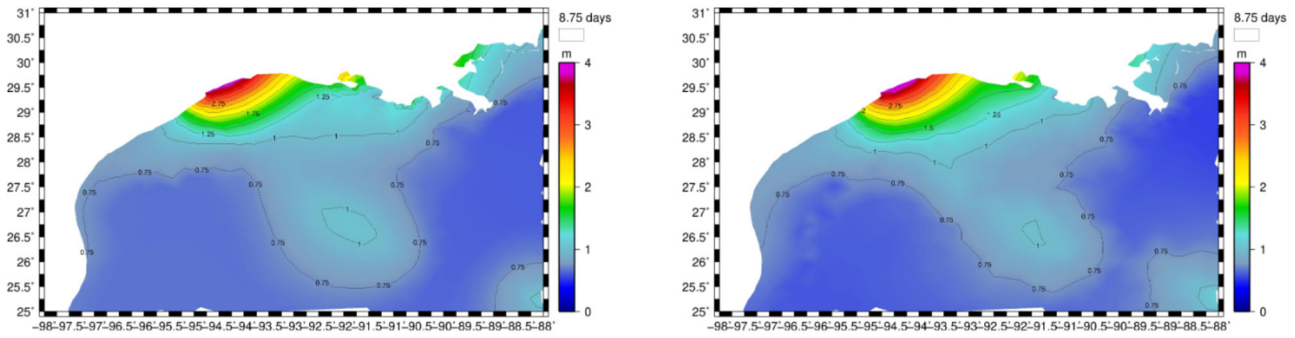

Fig. 13 | Hurricane Ike maximum elevation profiles. a The maximum surface elevations from the ADCIRC simulation. **b** The maximum surface elevations from the SWEMniCS simulation.

with forcing vector $\mathbf{f}(\mathbf{U})$:

$$\mathbf{f}(\mathbf{U}) = \begin{pmatrix} 0 \\ -g\zeta\frac{\partial h_B}{\partial x} + \tau_B uH - fvH - C_D\frac{\rho_{\text{air}}}{\rho_{water}} \parallel \mathbf{w} \parallel w_x + \frac{H}{\rho_{\text{water}}}\frac{\partial P}{\partial x} \\ -g\zeta\frac{\partial h_B}{\partial y} + \tau_B vH + fuH - C_D\frac{\rho_{\text{air}}}{\rho_{\text{water}}} \parallel \mathbf{w} \parallel w_y + \frac{H}{\rho_{\text{water}}}\frac{\partial P}{\partial y} \end{pmatrix} \quad (9)$$

We will also use the definition of a residual, $\mathbf{R}$, that can compactly be written if given a definition of the relevant shallow water variables, $\mathbf{Q}$, $\mathbf{F}$, $\mathbf{f}$:

$$\mathbf{R}(\mathbf{U}) = \partial_t \mathbf{Q} + div(\mathbf{F}) + \mathbf{f} = 0 \quad (10)$$

Non-conservative form

When working with continuous finite elements as well as with wetting and drying algorithms, the non-conservative form of the SWE may have more favorable numerical properties. This formulation is not valid in the presence of discontinuities. The non-conservative form may be obtained from the conservative form by applying the chain rule along with some algebraic manipulations. There are several works which carefully derive the non-conservative form of the shallow water equations[3,4]. We may write the non-conservative form of the SWE as:

$$\mathbf{R}_{nc} = \partial_t \mathbf{U} + \mathbf{A_1}\partial_x \mathbf{U} + \mathbf{A_2}\partial_y \mathbf{U} + \mathbf{f}/H = 0, \quad (11)$$

where the tensors are defined as:

$$\mathbf{A_1} = \begin{pmatrix} u & H & 0 \\ g & u & 0 \\ 0 & 0 & u \end{pmatrix}, \quad \mathbf{A_2} = \begin{pmatrix} v & 0 & H \\ 0 & v & 0 \\ g & 0 & v \end{pmatrix}, \quad (12)$$

and $\mathbf{f}$ is defined in (9). The non-conservative forms of the SWE will be used in both the SUPG formulation in (20) and in the momentum equations for one of the options with the wetting and drying scheme in (29).

Spherical coordinates

In many practical applications where larger spatial domains are considered, the SWE needs to be solved in spherical coordinates. Within this paper, we will employ projected spherical coordinates to simulate the storm surge from Hurricane Ike. Because FEniCSx allows for specification of the PDE using symbolic notation, we are able to include both the SWE in full spherical coordinates as well as projected spherical coordinates. For the SWE using projected spherical coordinates, we use the equirectangular projection formulation similar to refs. 15,16,53. Here we denote latitude by ϕ and longitude by λ. A reference coordinate, denoted (ϕ_o, λ_o), is selected and is assumed to be central in the given domain. We define R to be the radius of the earth in meters. The equations are solved in Cartesian coordinates (x, y) and the coordinates are transformed from the original spherical coordinates (λ, ϕ) via:

$$\begin{aligned} x &= R(\lambda - \lambda_o)\cos(\phi_o), \\ y &= R\phi, \end{aligned} \quad (13)$$

The flux tensor written for Cartesian coordinates in (8), becomes:

$$\mathbf{F}_s(\mathbf{U}) = \begin{pmatrix} \frac{\cos(\phi_o)}{\cos(\phi)}Hu & Hv \\ \frac{\cos(\phi_o)}{\cos(\phi)}(Hu^2 + \frac{1}{2}g(H^2 - h_b^2)) & Huv \\ \frac{\cos(\phi_o)}{\cos(\phi)}Huv & Hv^2 + \frac{1}{2}g(H^2 - h_b^2) \end{pmatrix}. \quad (14)$$

The forcing vector written in (9) for Cartesian coordinates now becomes:

$$\mathbf{f}_s(\mathbf{U}) = \begin{pmatrix} \frac{-Hv\tan(\phi)}{R} \\ -\frac{\cos(\phi_o)}{\cos(\phi)}g\zeta\frac{\partial h_B}{\partial x} + \tau_B uH - fvH - C_D\frac{\rho_{\text{air}}}{\rho_{\text{water}}} \parallel \mathbf{w} \parallel w_x + \frac{\cos(\phi_o)}{\cos(\phi)}\frac{H}{\rho_{\text{water}}}\frac{\partial P}{\partial x} - \frac{uvH\tan(\phi)}{R} \\ -g\zeta\frac{\partial h_B}{\partial y} + \tau_B vH + fuH - C_D\frac{\rho_{\text{air}}}{\rho_{\text{water}}} \parallel \mathbf{w} \parallel w_y + \frac{H}{\rho_{\text{water}}}\frac{\partial P}{\partial y} + \frac{u^2 H\tan(\phi)}{R} \end{pmatrix}. \quad (15)$$

The residual for the strong form of the conservative SWE in spherical projected coordinates can be written compactly as:

$$\mathbf{R}_s = \partial_t \mathbf{Q} + div(\mathbf{F}_s) + \mathbf{f}_s = 0. \quad (16)$$

Bottom friction

The specific form of the bottom friction term, τ_B, is important in determining the final closed form of the SWE which is all defined via constitutive laws. We currently have three implementations: a linear drag law, a quadratic drag law, and a Manning's law. The linear friction law can be written as:

$$\tau_B = c_f/H \quad (17)$$

where a typical value of the constant $c_f = 0.03$. The quadratic law can be written as:

$$\tau_B = c_f\sqrt{u^2 + v^2}/H. \quad (18)$$

As in ref. 22, an example value is 0.003. Lastly, the Manning's law can be written as:

$$\tau_B = gn^2 H^{-4/3}\sqrt{u^2 + v^2} \quad (19)$$

where g is the acceleration of gravity, and n is the Manning's roughness coefficient which depends on the material of the land surface. In SI units, Manning's n can vary from 0.01 to 0.07 $\text{m}^{-1/3}/\text{s}$.

In the next three sub-sections, we present our currently available finite element schemes defined for the SWE.

CG finite elements with SUPG stabilization

The SWE is a hyperbolic system of equations, so it is necessary to utilize techniques that provide numerical stability in order to use continuous finite elements effectively. The SUPG method[54] is a popular method and has been used for solving SWE in multiple cases[55–57]. For the test cases presented in the next section, we will use the SUPG formulation from the AdH[16]. It is noted that the following numerical scheme will be identical to that of[16] but with one main exception that is the nonlinear solver in SWEMniCS, described in the section "Time Stepping", utilizes an exact Newton method while AdH uses a finite difference approximation of the Newton method.

To briefly summarize, we begin with the standard Galerkin form which arises from multiplying by a test function $\mathbf{p}$ for each of the three equations, integrating across the domain, and then integrating by parts the term containing the flux tensor $\mathbf{F}$. The SUPG term is added to the standard Galerkin form and is a product of the strong form of the residual with the derivative of the test function in the upwind direction. We only consider the SUPG additions in the discrete setting so that these terms are added on an elemental basis. In general, the SUPG weak form for the conservative SWE can be written the following way. Find $\mathbf{U} \in U^h$ or find $\mathbf{Q} \in Q^h$:

$$(\partial_t \mathbf{Q}, \mathbf{p})_\Omega - (\mathbf{F}, \nabla(\mathbf{p}))_\Omega + (\mathbf{F}\cdot\mathbf{n}, \mathbf{p})_{\partial\Omega} + \left(\mathbf{R}, \tau_i \odot \mathbf{A}_i \frac{\partial \mathbf{p}}{\partial x_i}\right)_{\Omega_e} + (\mathbf{f}, \mathbf{p})_\Omega = 0 \quad \forall \mathbf{p} \in V^h, \tag{20}$$

where $\mathbf{n}$ is the unit normal vector pointing outward, τ_i is a 3x3 matrix for each i (i is defined for each spatial dimension, which in this case is two) that contains the coefficients for the SUPG term, $\odot$ is elementwise matrix-matrix multiplication, and $\mathbf{R}$ is usually the conservative form of the residual[30]. It is most common to take the conserved variable $\mathbf{Q}$ as the solution in which case $\mathbf{A_i} = \frac{\partial \mathbf{F}_{:,i}}{\partial \mathbf{Q}}$ [55,57]. Although this more standard formulation is available in our software, we observed better runtimes with the formulation from AdH and therefore elected to use it for the test cases in the next section.

In the AdH formulation, the primitives $\mathbf{U}$ are the solution variable, and the residual term in the SUPG term is in non-conservative form: $\mathbf{R} = \mathbf{R}_{nc}$. As a result, $\mathbf{A}_i$ in (20) is $\mathbf{A}_1$ and $\mathbf{A}_2$ from (12). It is noted that there has also been successful usage of the SUPG with the primitives, $\mathbf{U}$, as the solution variable in other equations besides SWE[58,59].

We use the following definition of the stabilization parameter τ_i from[16]:

$$\tau_i = \tau = \alpha \frac{A_e}{\sqrt{u^2 + v^2 + gH}}, \tag{21}$$

where A_e is the radius of the element, and α is a positive constant typically chosen to be between 0 and 0.5.

DG finite elements

In DG finite element methods, the solutions are allowed to be discontinuous at element interfaces and are continuous in a weak sense. In order for a closed system of equations to be formed, a unique flux must be specified at the element interfaces. DG methods have been used extensively to solve many conservation laws, including the SWE[22,24,26,60–62]. However, the vast majority of these methods with a few exceptions[24,62] use fully explicit time integration schemes that require a CFL constraint in order to preserve stability. The following scheme available within SWEMniCS is fully implicit in time and uses an exact Newton method which is described in Section 4.9. The weak form is: Find $\mathbf{U} \in U^h$ or find $\mathbf{Q} \in Q^h$:

$$(\partial_t \mathbf{Q}, \mathbf{p})_{\Omega_e} - (\mathbf{F}, \nabla(\mathbf{p}))_{\Omega_e} + (\hat{\mathbf{F}}\cdot\mathbf{n}, \mathbf{p})_{\partial\Omega_e} + (\mathbf{f}, \mathbf{p})_{\Omega_e} = 0 \quad \forall \mathbf{p} \in V^h, \tag{22}$$

where $\hat{\mathbf{F}}$ denotes the numerical flux. For the flux on the element boundary, we choose to use Lax-Friedrichs upwinding defined as:

$$(\hat{\mathbf{F}}\cdot\mathbf{n}, \mathbf{p})_{\partial\Omega_e} = (\mathrm{avg}(\mathbf{F})\cdot\mathbf{n}^+ + \lambda \mathrm{jump}(\mathbf{Q}), \mathrm{jump}(\mathbf{p}))_{\partial\Omega_e}. \tag{23}$$

In this case, $^+$ denotes the function value on one side of an inter-element edge $\partial\Omega_e$ while $^-$ denotes the value on the opposite side of that edge. The average and jump operators are defined respectively as $\mathrm{avg}(\cdot) = \frac{1}{2}(\cdot^+ + \cdot^-)$ and $\mathrm{jump}(\cdot) = \cdot^+ - \cdot^-$. The value λ comes from the eigenvalues of the Jacobian of the flux tensor and is defined as:

$$\lambda = \frac{1}{2}\max\left(\sqrt{(u^+)^2 + (v^+)^2} + \sqrt{gH^+}, \sqrt{(u^-)^2 + (v^-)^2} + \sqrt{gH^-}\right). \tag{24}$$

It is most common to use the conserved variables, $\mathbf{Q}$, as the solution variable with the DG approach as this approach keeps terms that are differentiated by time linear. This allows for the discretized finite element system to be cast more readily into a system of ODE's to be solved forward in time. Our code allows for the choice of either the solution variable to be the primitives, $\mathbf{U}$ or the conserved variables $\mathbf{Q}$. The usage of this DG scheme along with the fully implicit BDF2 time step scheme and exact Newton solver described in section "Time Stepping" has not been implemented before to the best knowledge of the authors. In SWEMniCS, this DG scheme also can handle wetting and drying as described in section "Wetting and drying".

DG-CG

To demonstrate the flexibility of our software, we have also employed a novel combined FE method where we use a discontinuous finite element method for the continuity equation and a continuous finite element method for the momentum equations. The motivation here is to take advantage of the ability of the DG scheme to capture shocks and maintain local conservation with respect to mass while also reducing the total number of degrees of freedom by using a CG scheme for the two momentum equations.

The scheme is essentially the DG scheme from section "DG finite elements" but with CG solution/test space for the momentum equations. Hence, in the momentum equations there is no numerical flux applied. Here we will note that our finite element space is now a combined finite element space where $\mathbf{U} \in U$ represents a vector function of length three where the first component is a DG finite element function, and the second and third components are CG finite element functions: $\mathbf{U} = (H_{dg}, u_{cg}, v_{cg})$. This implies that $\mathbf{Q} = (H_{dg}, H_{dg}u_{cg}, H_{dg}v_{cg})$. Additionally, the test functions come from the combined space so we will denote $\mathbf{p} = (p_1, p_2, p_3)$ where p_1 comes from the DG finite element space and p_2, p_3 comes from the CG finite element space.

The combined formulation is in fact identical to the DG formulation (22) but applied to the combined space. The only simplification is that the inter-element Lax-Friedrichs flux is trivially zero for the two momentum equations because the jump of the now continuous test functions, p_1, p_2, for the momentum equations are trivially zero. Formal analysis of the properties of this scheme is a work in progress, but for the purposes of this research, it is intended as a proof of concept of how simple it is to create new finite element schemes within our FEniCSx based software.

Boundary conditions

All of the finite element schemes described require weak or strong enforcement of boundary conditions. The available boundary conditions currently in the software that will be used in the test cases include open boundaries, and wall boundaries which will both be enforced weakly. For a discussion on which boundary conditions are appropriate for a given problem, we refer to ref. 3 and only state the types of boundaries here.

The open boundary condition is defined by enforcing a Dirichlet condition weakly on the water depth, $H = H_o$, while leaving the velocities free. This type of boundary arises when tidal heights are used as a forcing function onto a domain. The wall boundary condition is designed to behave as an impenetrable barrier and is implemented by setting the flux normal to

the boundary to zero. In our software, the wall boundary condition is enforced weakly. For the CG finite elements, both the wall and open boundaries are accomplished by modifying the definition of **F** on the appropriate segment of the boundary. If we assume $\mathbf{u}\cdot\mathbf{n}=0$ on any wall boundary then it must be true that the only non-zero components in **F** are:

$$\mathbf{F}=\begin{pmatrix}0 & 0\\ \frac{1}{2}g(H^2-h_b^2) & 0\\ 0 & \frac{1}{2}g(H^2-h_b^2)\end{pmatrix}. \tag{25}$$

Similarly, for weak enforcement of the open boundary, we replace the definition of **F(U)** on the open boundary with $\mathbf{F}(\mathbf{U_o})$. Where $\mathbf{U_o}=(H_o,u,v)^T$ are the primitive variables but with the depth variable replaced with the Dirichlet condition for H on the open boundary. For the DG finite elements, we use the standard weak boundary enforcement for walls and open boundaries as described in ref. 22.

Time stepping

The finite element schemes in Sections "CG finite elements with SUPG stabilization", "DG finite elements", and "DG-CG" are all written in semi-discrete form. In order to have a fully discrete system, the temporal derivative is approximated via finite difference. Currently, the implemented time stepping scheme is the fully implicit generalized BDF2 as in ref. 16. The fully implicit time-stepping scheme allows for SWEMniCS to take larger time steps than the typical CFL constraints of explicit time-stepping schemes[5,26,61] and could therefore bring potential performance benefits when the CFL condition is particularly restrictive. The scheme operating on flux **Q** is defined as:

$$\partial_t\mathbf{Q}\approx\alpha\left[\frac{\frac{3}{2}\mathbf{Q}^{n+1}-2\mathbf{Q}^{n}+\frac{1}{2}\mathbf{Q}^{n-1}}{\Delta t}\right]+(1-\alpha)\left[\frac{\mathbf{Q}^{n+1}-\mathbf{Q}^{n}}{\Delta t}\right]. \tag{26}$$

The time step size is defined as Δt, with n being an integer that represents the nth discrete time level, and the parameter α is a scalar constant that can vary between zero and one. A value of $\alpha=0$ would result in the first-order implicit Euler scheme while a value of $\alpha=1$ would result in the second-order BDF2 scheme. This scheme is fully implicit, meaning all of the terms outside of the finite difference approximation to the time derivative are evaluated at time level $n+1$. This results in a nonlinear system of equations at each time step that is solved via Newton's method. An advantage of using the FEniCSx software is that it is capable of automatically computing the Jacobian necessary to define the Newton method via symbolic differentiation. The underlying linear solver uses the PETSc library and by default uses the GMRES iterative solver with block Jacobi preconditioning.

Wetting and drying

In many practical applications, the domain of interest may include areas that transition between wet ($H>0$) or dry ($H=0$) over the course of a simulation. The SWE becomes ill-defined as $H\to 0$ and so in practice an empirical treatment must be used in order to allow for wetting and drying. In SWEMniCS, we adapt the α scheme of Kärna[46] so that it is compatible with the finite element and time-stepping schemes from the previous sections. This scheme was chosen because it is designed to work with implicit time-stepping schemes (no required time-step restriction) and it is convenient to implement within the FEniCSx library because it can be directly written in Unified Form Language. For the purposes of this study, the α scheme is only applied to the DG scheme from section "DG finite elements" although the application of the α scheme towards the SUPG-based scheme is planned future work.

The main idea of the α scheme is to introduce a special function $f(H)$ to enforce positive water depths. This can be interpreted as allowing for the bathymetry to move over time to enforce a very small minimal depth. For more details on the α scheme, readers are referred to ref. 46.

The α scheme is applied to the scheme in section "DG finite elements" by replacing any instances of the variables within **U** with $\tilde{\mathbf{U}}=(\tilde{H},u,v)^T$, **Q** with $\tilde{\mathbf{Q}}=(\tilde{H},u\tilde{H},v\tilde{H})^T$, and h_b with $\tilde{h_b}$. The modified variable, $\tilde{H}$, is defined by adding the aforementioned function $f(H)$ so that $\tilde{H}$ must always be positive. The definitions for $\tilde{H}$, $\tilde{h_b}$ along with the choice for $f(H)$ in SWEMniCS (same as in ref. 46) are:

$$\begin{aligned}\tilde{H}&=H+f(H)\\ \tilde{h_b}&=h_b+f(H)\\ f(H)&=\tfrac{1}{2}(\sqrt{H^2+\alpha^2}-H).\end{aligned} \tag{27}$$

It is noted that the choice for $f(H)$ is arbitrary so long as: $\tilde{H}>0$ for any H, $f(H)\approx -H$ as H decreases below 0, $f(H)\approx 0$ as H increases above 0, and $f(H)$ must be continuously differentiable for any H. An appropriate choice of α is one that is as small as possible but must grow with the change in bathymetry from one element to the other. It is noted that even though $\tilde{H}$ is replaced into the FE formulation, the solution variable remains H (therefore allowing for negative H). One convenient property of this scheme is that the water surface elevation from the solution remains unchanged regardless of the swapped variables: $\zeta=H-h_b=\tilde{H}-\tilde{h_b}$.

The test case in section "Wetting and drying" demonstrates the usage of the α scheme within SWEMniCS for both a conservative and non-conservative form of momentum equations. The FE form with wetting-drying for the conservative momentum equations is identical to the FE form without wetting-drying from section "DG finite elements" except for two modifications: the first is the replacement of any occurrences of H or h_b with the positivity-enforcing $\tilde{H}$ and $\tilde{h_b}$ from (27), and the second being a slight manipulation of the definition of the flux tensor **F** to minimize the number of occurrences of the nonlinear additions to $\tilde{H}$ and $\tilde{h_b}$. We note that this modification is exact in the formal setting but practically will reduce the number of occurrences of the computation of the nonlinear function $f(H)$ in the weak form due to the fact that ζ doesn't require the usage of $f(H)$. The new flux definition with wetting and drying can be written as:

$$\begin{aligned}\mathbf{F}&=\begin{pmatrix}\tilde{H}u & \tilde{H}v\\ \tilde{H}u^2+\frac{1}{2}g(\tilde{H}^2-\tilde{h_b}^2) & \tilde{H}uv\\ \tilde{H}uv & \tilde{H}v^2+\frac{1}{2}g(\tilde{H}^2-\tilde{h_b}^2)\end{pmatrix}\\ &=\begin{pmatrix}\tilde{H}u & \tilde{H}v\\ \tilde{H}u^2+\frac{1}{2}g(\zeta^2+2\zeta\tilde{h_b}) & \tilde{H}uv\\ \tilde{H}uv & \tilde{H}v^2+\frac{1}{2}g(\zeta^2+2\zeta\tilde{h_b})\end{pmatrix}.\end{aligned} \tag{28}$$

We can see that the manipulation in (28) to the form on the right side makes the flux only linearly dependent on the positivity-enforcing function $f(H)$ while also reducing the total number of occurrences of $f(H)$ in the definition of **F**.

Within SWEMniCS, we also allow for the usage of the DG scheme with wetting and drying for the non-conservative momentum equations as in section "Non-conservative form" (still using conservative form of continuity equation). The motivation for including this is two-fold. First, it is to compare results from cases involving wetting and drying with the fully conservative momentum equations. The combination of the usage of the conservative continuity equation and the non-conservative momentum equations is the same as what the α wetting and drying scheme was originally applied to in Kärna[46]. The main difference with its usage in SWEMniCS is the choice of numerical flux and time step scheme.

Second, the usage of the non-conservative momentum equations eliminates any definition of $\tilde{H}$ or $\tilde{h_b}$ outside of the potential usage within the source term. This could improve computational performance of SWEMniCS by reducing the amount of nonlinearity present in the conservative momentum equations.

The application of the α scheme to the conservative continuity equations and non-conservative momentum equations to the FE scheme in section

"DG finite elements" can be written in the following way. Find $\mathbf{U} \in U^h$:

$$(\partial_t \tilde{\mathbf{U}}, \mathbf{p})_{\Omega_e} - (\mathbf{F}_c, \nabla(\mathbf{p}))_{\Omega_e} + (\hat{\mathbf{F}}_c \cdot \mathbf{n}, \mathbf{p})_{\partial\Omega_e} - (div(\mathbf{p}_m \otimes \mathbf{u}), \mathbf{u})_{\Omega_e} + (\hat{\mathbf{u}} \cdot \mathbf{n}, \mathbf{u} \cdot \mathbf{p}_m)_{\partial\Omega_e} + (\mathbf{f}_*, \mathbf{p})_{\Omega_e} = 0 \quad \forall \mathbf{p} \in V^h, \tag{29}$$

Here, $\mathbf{u} = (u, v)^T$ is just the velocity components from the solution, $\mathbf{p}_m = (p2, p3)^T$ are the test functions for the x and y momentum equations. $\mathbf{F}_c$ is the flux tensor as in (28) but with the second and third rows containing the gravity terms from the non-conservative momentum equations as in (11). The source term also changes from the previous definition because we elect to move all gravity terms within the flux tensor to avoid extra definitions of the nonlinear $f(H)$. $\mathbf{F}_c$ and $\mathbf{f}_*$ are defined as:

$$\mathbf{F}_c = \begin{pmatrix} \tilde{H}u & \tilde{H}v \\ g\zeta & 0 \\ 0 & g\zeta \end{pmatrix}, \quad \mathbf{f}_* = \begin{pmatrix} 0 \\ \tau_B u - fv - C_D \frac{\rho_{\text{air}}}{\rho_{water}} \| \mathbf{w} \| \frac{w_x}{\tilde{H}} + \frac{1}{\rho_{\text{water}}} \frac{\partial P}{\partial x} \\ \tau_B v + fu - C_D \frac{\rho_{\text{air}}}{\rho_{\text{water}}} \| \mathbf{w} \| \frac{w_y}{\tilde{H}} + \frac{1}{\rho_{\text{water}}} \frac{\partial P}{\partial y} \end{pmatrix}. \tag{30}$$

We employ the same numerical flux as in (23) to uniquely define any variables on the interface. We can see now that this non-conservative formulation does not include any usage of $f(H)$ within the flux terms or time derivatives of the momentum equations.

Results involving wetting and drying from the fully conservative scheme and the non-conservative momentum scheme will be compared in section "Wetting and drying" against a validated SWE model ADCIRC.

Additional details

Mixed order p elements are allowed, meaning that for any weak form mentioned above, we can select a different polynomial order for each solution variable. However, in all of our test cases, we use uniform-order polynomials. Although any appropriate element type is allowable in FEniCSx, for the test cases described below we simply use Lagrange elements defined on triangles.

To ensure that SWEMniCS can be a potentially competitive alternative to existing software and moreover can access high-quality inputs from ADCIRC, SWEMniCS is supplied with conversion scripts for ADCIRC inputs.

Data availability

All data required to reproduce all results are openly available at https://github.com/UT-CHG/SWEMniCS.

Code availability

Source code used in the simulations is openly available at https://github.com/UT-CHG/SWEMniCS.

Received: 16 May 2024; Accepted: 5 November 2024;
Published online: 18 December 2024

References

1. Smith, A. B. U.S. billion-dollar weather and climate disasters, 1980–present https://www.ncei.noaa.gov/archive/accession/0209268 (2020).
2. Smith, A. B. & Katz, R. W. US billion-dollar weather and climate disasters: data sources, trends, accuracy and biases. *Nat. Hazards* **67**, 387–410 (2013).
3. Vreugdenhil, C. B. *Numerical methods for shallow-water flow* 13 (Springer, Netherlands, 1994).
4. Tan, W.-Y. *Shallow water hydrodynamics: Mathematical theory and numerical solution for a two-dimensional system of shallow-water equations*. (Elsevier, China, 1992).
5. Westerink, J. J. et al. A basin- to channel-scale unstructured grid hurricane storm surge model applied to Southern Louisiana. *Monthly Weather Rev.* **136**, 833–864 (2008).
6. Zhang, Y. J. et al. Simulating compound flooding events in a hurricane. *Ocean Dyn.* **70**, 621–640 (2020).
7. Hope, M. E. et al. Hindcast and validation of Hurricane Ike (2008) waves, forerunner, and storm surge. *J. Geophys. Res.: Oceans* **118**, 4424–4460 (2013).
8. Burgan, H. I. & Icaga, Y. Flood analysis using adaptive hydraulics (AdH) model in Akarcay Basin. *Tek. Dergi* **30**, 9029–9051 (2019).
9. Jones, J. E. & Davies, A. M. Application of a finite element model (TELEMAC) to computing the wind induced response of the Irish Sea. *Cont. shelf Res.* **26**, 1519–1541 (2006).
10. Beisiegel, N., Vater, S., Behrens, J. & Dias, F. An adaptive discontinuous Galerkin method for the simulation of hurricane storm surge. *Ocean Dyn.* **70**, 641–666 (2020).
11. Woodruff, J., Dietrich, J., Wirasaet, D., Kennedy, A. & Bolster, D. Storm surge predictions from ocean to subgrid scales. *Nat. Hazards* **117**, 2989–3019 (2023).
12. Arpaia, L., Ricchiuto, M., Filippini, A. G. & Pedreros, R. An efficient covariant frame for the spherical shallow water equations: well balanced DG approximation and application to tsunami and storm surge. *Ocean Model.* **169**, 101915 (2022).
13. Chen, C., Beardsley, R. & Cowles, G. An unstructured grid, finite-volume coastal ocean model (FVCOM) system. *Oceanography* **19**, 78–89 (2006).
14. Shchepetkin, A. F. & McWilliams, J. C. The regional oceanic modeling system (ROMS): a split-explicit, free-surface, topography-following-coordinate oceanic model. *Ocean Model.* **9**, 347–404 (2005).
15. Luettich, R. A. & Westerink, J. J. *Formulation and numerical implementation of the 2D/3D ADCIRC finite element model version 44. XX* 20 (Chapel Hill, NC, USA, 2004).
16. Savant, G., Berger, R. C., Trahan, C. J. & Brown, G. L. Theory, formulation, and implementation of the cartesian and spherical coordinate two-dimensional depth-averaged module of the adaptive hydraulics (AdH) finite element numerical code (2020).
17. Taylor, M., Tribbia, J. & Iskandarani, M. The spectral element method for the shallow water equations on the sphere. *J. Comput. Phys.* **130**, 92–108 (1997).
18. Navon, I. A review of finite-element methods for solving the shallow-water equations. *Comput. Model Ocean Eng*. **727** (1988).
19. Behrens, J. Atmospheric and ocean modeling with an adaptive finite element solver for the shallow-water equations. *Appl. Numer. Math.* **26**, 217–226 (1998).
20. Valseth, E. & Dawson, C. A stable space-time FE method for the shallow water equations. *Comput. Geosci.* **26**, 1–18 (2022).
21. Galland, J.-C., Goutal, N. & Hervouet, J.-M. TELEMAC: a new numerical model for solving shallow water equations. *Adv. Water Resour.* **14**, 138–148 (1991).
22. Kubatko, E. J., Westerink, J. J. & Dawson, C. hp discontinuous Galerkin methods for advection dominated problems in shallow water flow. *Comput. Methods Appl. Mech. Eng.* **196**, 437–451 (2006).
23. Wichitrnithed, C. et al. A discontinuous Galerkin finite element model for compound flood simulations. *Comput. Methods Appl. Mech. Eng.* **420**, 116707 (2024).
24. Kärnä, T. et al. Thetis coastal ocean model: discontinuous Galerkin discretization for the three-dimensional hydrostatic equations. *Geosci. Model Dev.* **11**, 4359–4382 (2018).
25. Samii, A., Kazhyken, K., Michoski, C. & Dawson, C. A comparison of the explicit and implicit hybridizable discontinuous Galerkin methods for nonlinear shallow water equations. *J. Sci. Comput.* **80**, 1936–1956 (2019).
26. Dawson, C. & Proft, J. Discontinuous and coupled continuous/discontinuous Galerkin methods for the shallow water equations. *Comput. Methods Appl. Mech. Eng.* **191**, 4721–4746 (2002).
27. Dawson, C., Westerink, J. J., Feyen, J. C. & Pothina, D. Continuous, discontinuous and coupled discontinuous–continuous Galerkin finite element methods for the shallow water equations. *Int. J. Numer. Methods Fluids* **52**, 63–88 (2006).

28. Akbar, M. & Aliabadi, S. Hybrid numerical methods to solve shallow water equations for hurricane induced storm surge modeling. *Environ. Model. Softw.* **46**, 118–128 (2013).
29. Zhang, Y. J., Ye, F., Stanev, E. V. & Grashorn, S. Seamless cross-scale modeling with SCHISM. *Ocean Model.* **102**, 64–81 (2016).
30. Hughes, T. J. & Mallet, M. A new finite element formulation for computational fluid dynamics: III. the generalized streamline operator for multidimensional advective-diffusive systems. *Comput. Methods Appl. Mech. Eng.* **58**, 305–328 (1986).
31. Anderson, R. et al. MFEM: a modular finite element methods library. *Comput. Math. Appl.* **81**, 42–74 (2021).
32. Arndt, D. et al. The deal. II finite element library: design, features, and insights. *Comput. Math. Appl.* **81**, 407–422 (2021).
33. Brown, J. et al. libCEED: fast algebra for high-order element-based discretizations. *J. Open Source Softw.* **6**, 2945 (2021).
34. Blatt, M. et al. The distributed and unified numerics environment, version 2.4. *Arch. Numer. Softw.* **4**, 13–29 (2016).
35. Hecht, F. New development in FreeFem++. *J. Numer. Math.* **20**, 251–266 (2012).
36. Kirk, B. S., Peterson, J. W., Stogner, R. H. & Carey, G. F. libMesh: a C++ library for parallel adaptive mesh refinement/coarsening simulations. *Eng. Comput.* **22**, 237–254 (2006).
37. Schöberl, J. C++ 11 implementation of finite elements in NGSolve. *Institute for analysis and scientific computing, Vienna University of Technology* **30** (2014).
38. Scroggs, M. W., Dokken, J. S., Richardson, C. N. & Wells, G. N. Construction of arbitrary order finite element degree-of-freedom maps on polygonal and polyhedral cell meshes. *ACM Trans. Math. Softw.* **48**, 18:1–18:23 (2022).
39. Baratta, I. A. et al. DOLFINx: the next generation FEniCS problem solving environment. preprint (2023).
40. Alnaes, M. S., Logg, A., Ølgaard, K. B., Rognes, M. E. & Wells, G. N. Unified form language: a domain-specific language for weak formulations of partial differential equations. *ACM Transactions Math. Softw.* **40**, 1–37 (2014).
41. Scroggs, M. W., Baratta, I. A., Richardson, C. N. & Wells, G. N. Basix: a runtime finite element basis evaluation library. *J. Open Source Softw.* **7**, 3982 (2022).
42. Stanzione, D. et al. *Frontera: The evolution of leadership computing at the National Science Foundation* (2020).
43. Lynch, D. R. & Gray, W. G. Analytic solutions for computer flow model testing. *J. Hydraulics Div.* **104**, 1409–1428 (1978).
44. LeVeque, R. J. Balancing source terms and flux gradients in high-resolution Godunov methods: the quasi-steady wave-propagation algorithm. *J. Comput. Phys.* **146**, 346–365 (1998).
45. Delestre, O. et al. SWASHES: a compilation of shallow water analytic solutions for hydraulic and environmental studies. *Int. J. Numer. Methods Fluids* **72**, 269–300 (2013).
46. Kärnä, T. et al. 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**, 509–524 (2011).
47. Balzano, A. Evaluation of methods for numerical simulation of wetting and drying in shallow water flow models. *Coast. Eng.* **34**, 83–107 (1998).
48. East, J. W., Turco, M. J. & Mason Jr, R. R. Monitoring inland storm surge and flooding from Hurricane Ike in Texas and Louisiana, September 2008. *Surge* **29**, 95–20833 (2008).
49. Cardone, V. & Cox, A. Tropical cyclone wind field forcing for surge models: critical issues and sensitivities. *Nat. Hazards* **51**, 29–47 (2009).
50. Demkowicz, L. & Gopalakrishnan, J. A class of discontinuous Petrov–Galerkin methods. II. Optimal test functions. *Numer. Methods Partial Differ. Equ.* **27**, 70–105 (2011).
51. Villa, U., Petra, N. & Ghattas, O. Hippylib: an extensible software framework for large-scale inverse problems. *J. Open Source Softw.* **3**, 940 (2018).
52. Pachev, B., Arora, P., del Castillo-Negrete, C., Valseth, E. & Dawson, C. A framework for flexible peak storm surge prediction. *Coast. Eng.* **186**, 104406 (2023).
53. Kolar, R., Gray, W., Westerink, J. & Luettich Jr, R. Shallow water modeling in spherical coordinates: equation formulation, numerical implementation, and application. *J. Hydraulic Res.* **32**, 3–24 (1994).
54. Hughes, T. & Tezduyar, T. Finite element methods for first-order hyperbolic systems with particular emphasis on the compressible Euler equations. *Comput. Methods Appl. Mech. Eng.* **45**, 217–284 (1984).
55. Takase, S., Kashiyama, K., Tanaka, S. & Tezduyar, T. E. Space–time SUPG formulation of the shallow-water equations. *Int. J. Numer. Methods Fluids* **64**, 1379–1394 (2010).
56. Bova, S. & Carey, G. A symmetric formulation and SUPG scheme for the shallow-water equations. *Adv. Water Resour.* **19**, 123–131 (1996).
57. Cengizci, S. & Uğur, Ö. SUPG formulation augmented with yzβ shock-capturing for computing shallow-water equations. *ZAMM-J. Appl. Math. Mech./Z. f.ür. Angew. Mathematik und Mech.* **103**, e202200232 (2023).
58. Hauke, G. & Hughes, T. J. A comparative study of different sets of variables for solving compressible and incompressible flows. *Comput. methods Appl. Mech. Eng.* **153**, 1–44 (1998).
59. Hauke, G. & Hughes, T. A unified approach to compressible and incompressible flows. *Comput. Methods Appl. Mech. Eng.* **113**, 389–395 (1994).
60. Xing, Y., Zhang, X. & Shu, C.-W. Positivity-preserving high order well-balanced discontinuous Galerkin methods for the shallow water equations. *Adv. Water Resour.* **33**, 1476–1493 (2010).
61. Ern, A., Piperno, S. & Djadel, K. A well-balanced Runge–Kutta discontinuous Galerkin method for the shallow-water equations with flooding and drying. *Int. J. Numer. Methods fluids* **58**, 1–25 (2008).
62. Meister, A. & Ortleb, S. On unconditionally positive implicit time integration for the DG scheme applied to shallow water flows. *Int. J. Numer. Methods Fluids* **76**, 69–94 (2014).

Acknowledgements

This work has been supported by the United States National Science Foundation NSF PREEVENTS Track 2 Program, under NSF Grant Numbers. 1855047. This material is also based on work supported by the US Department of Homeland Security under Grant No. 2015-ST-061-ND0001-01. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the US Department of Homeland Security. The authors also would like to gratefully acknowledge the use of the "ADCIRC", "DMS23001", "DMS23023", and "DMS21031" allocations on the Frontera supercomputer at the Texas Advanced Computing Center at the University of Texas at Austin.

Author contributions

M.L., B.P., E.V., and J.P. wrote the main manuscript text. M.L. and B.P. developed software and conducted simulations. All authors contributed to mathematical derivations and all authors reviewed the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s44304-024-00036-5.

Correspondence and requests for materials should be addressed to Mark Loveland.