Coastal ocean forecasting on the GPU using a two-dimensional finite-volume scheme

André R. Brodtkorb & HÅvard Heitlo Holm

Published online: 17 May 2021.

Submit your article to this journal ⬀

Article views: 3994

View related articles ⬀

View Crossmark data ⬀

Coastal ocean forecasting on the GPU using a two-dimensional finite-volume scheme

By ANDRÉ R. BRODTKORB[1]*, and HÅVARD HEITLO HOLM[2], [1]*Oslo Metropolitan University, Oslo, Norway;* [2]*SINTEF Digital, Oslo, Norway*

ABSTRACT

In this work, we take a modern high-resolution finite-volume scheme for solving the rotational shallow-water equations and extend it with features required to run real-world ocean simulations. Our contributions include a spatially varying north vector and Coriolis term required for large scale domains, moving wet-dry fronts, a static land mask, bottom shear stress, wind forcing, boundary conditions for nesting in a global model, and an efficient model reformulation that makes it well-suited for massively parallel implementations. Our model order is verified using a grid convergence test, and we show numerical experiments using three different sections along the coast of Norway based on data originating from operational forecasts run at the Norwegian Meteorological Institute. Our simulation framework shows perfect weak scaling on a modern P100 GPU, and is capable of providing tidal wave forecasts that are very close to the operational model at a fraction of the cost. All source code and data used in this work are publicly available under open licenses.

Keywords: shallow-water equations, oceanography, GPU computing, realistic use cases, high-resolution finite-volume methods

1. Introduction

The aim of this paper is to simulate realistic oceanographic scenarios using a modern finite-volume scheme on GPUs. Modern operational ocean models, such as the Regional Ocean Modeling System (ROMS) (Shchepetkin and McWilliams, 2005) and the Nucleus for European Modelling of the Ocean (NEMO) (the NEMO team, 2008), are based on solving the primitive equations that describe conservation of mass, momentum, temperature, and salinity. Even though these ocean models are heavily optimised, their complex physical and mathematical properties make them very computationally demanding. The NorKyst-800 model system (Albretsen et al., 2011), for example, is based on the ocean model ROMS (using the Rutgers-ROMS 3.7 dynamic kernel) and covers the Norwegian coast with 800 m horizontal resolution. It is run operationally by the Norwegian Meteorological Institute on a daily basis. The complete model consists of 42 vertical layers with 2600×900 grid cells each, and a 24 hour forecast takes 45 minutes when using 512 CPU cores (Christensen, 2020). It is unfeasible to use such models for flexible on-demand simulations during search-and-rescue operations, oil spills, heavy lifting at sea, storm surges, etc. In these cases, fast on-demand simulations based on the shallow-water equations can have a significant value for decision makers. Furthermore, such fast simulations enables running ensembles with thousands of members required for fully non-linear data assimilation (Holm et al., 2020).

We simulate the shallow-water equations in a rotating frame of reference, targeting efficient simulation of ocean currents based on the operational NorKyst-800 data. Our model equations are essentially vertical integrations of the primitive equations and assume constant density and a vertically integrated momentum (see more detailed derivations in, e.g. Røed (2019)). Our work starts with the recent high-resolution finite-volume scheme by Chertock et al. (2018), and extends this scheme to support bottom shear stress, wind forcing, projection-free spatial variations in the north vector, spatially varying Coriolis parameter, land mask, and a moving wet-dry boundary. We present how to reconstruct a piecewise bilinear bathymetry from the piecewise constant bathymetry in the NorKyst-800 data, nesting of the model into the NorKyst-800 data using linearly interpolated boundary conditions, and generation of both higher and lower spatial resolution of the initial conditions.

We simulate increasingly challenging scenarios along the Norwegian coast, and compare with reference results from

---

*Corresponding author. Email: Andre.Brodtkorb@oslomet.no

1

the operational model. As the shallow-water equations are particularly well suited for tidal and storm-surge predictions, we also compare our tidal-wave predictions at several locations with the reference dataset and observations. All datasets and source code used in this work are publicly available under open licenses (Kluyver et al., 2016).[1]

The rest of the paper is organised as follows: Section 2 gives a high-level description of the shallow-water equations and how they can be solved numerically. In Section 3, we present all the extensions that are required to tackle relevant oceanographic problems. The performance and numerical accuracy of our scheme are assessed in Section 4, where we show perfect weak scaling of the GPU implementation and run a grid convergence test with the new terms introduced in the scheme. We present operational-grade simulations in Section 5, and finally summarise the work in Section 6.

1.1. Related work

The aim of this work is to develop a coastal ocean forecasting system that is useful for short-term predictions of real-world ocean currents and sea-surface level in coastal waters. To this end, we use a shallow-water model in a rotating frame of reference. Such models are used operationally for forecasting of storm surges (Flowerdew et al., 2010; Guide to storm surge forecasting, 2011; Bertin, 2016) and tsunamis (Rakowsky et al., 2013; Castro et al., 2015).

Shallow-water models can be discretised and solved in different ways. They are often discretised using classical finite-difference methods on staggered C-grids within operational meteorology and oceanography (see, e.g. Shchepetkin and McWilliams, 2005; the NEMO team, 2008; Røed, 2019) as they are especially well suited for consistent specifications of boundary conditions and nested models. Finite-volume methods have also been used, but require more care when implementing boundary conditions (Natvig, 2006).

Finite-volume schemes have traditionally required computationally expensive Riemann solvers that resolve the Riemann problem at the interface between two cells, but many modern schemes use solvers that replace the Riemann solver by an efficient approximation. One example is the family of schemes based on the work of Kurganov and Tadmor (2000) who proposed a numerical scheme that was well balanced for lake at rest steady-state solutions. This scheme was further developed to handle dry states (Kurganov and Levy, 2002; Kurganov and Petrova, 2007), and recently Chertock et al. (2018) adapted the scheme to rotating shallow-water flow in which steady-states in geostrophic balance are represented correctly. We have previously demonstrated that this scheme captures important properties required for oceanographic applications and compared it against classical finite-difference approaches (Holm et al., 2020).

Hyperbolic conservation laws, such as the shallow-water equations, are often solved using explicit discretizations in time. Such schemes can be represented using stencil computations to compute the numerical fluxes, which is an embarrassingly parallel algorithmic primitive (Asanovic et al., 2006). Stencil computations are perfectly suited for implementation on massively parallel accelerator hardware, such as GPUs (see, e.g. Hagen et al., 2007; de la Asunción et al., 2011; Brodtkorb et al., 2012; Parna et al., 2018), and when implemented correctly, the use of GPUs not only increases the computational efficiency (Brodtkorb et al., May 2010), but also typically the energy efficiency (see, e.g. Huang et al., 2009; Qi et al., 2014; Dong et al., 2014; Holm et al., 2020). The use of GPUs for shallow water and oceanographic simulations is currently also included in large commercial and academic software packages such as ClawPack (Qin et al., 2019), Telemac (Grasset et al., 2019), TUFLOW (Huxley and Syme, 2016) and MIKE (MIKE Powered by DHI, 2019).

In an operational setting, it is important to have a sound treatment of boundary conditions and source terms. In this work, we apply the flow relaxation scheme (Martinsen and Engedahl, 1987) for boundary conditions, but a more thorough discussion on the state-of-the-art on the topic can be found in Klingbeil et al. (2018).

2. Mathematical formulation

We use the shallow-water equations with source terms to simulate ocean dynamics. Since the equations are purely barotropic, they are not expected to generate or preserve features such as mesoscale eddies in the long term. These features arise from baroclinic instabilities, typically caused by variations in temperature and salinity that are not included in the present mathematical model, but the equations are nevertheless capable of capturing the most important physics required to simulate short-term ocean dynamics. We can write the equations with source terms as

$$
\begin{bmatrix} h \\ hu \\ hv \end{bmatrix}_t + \begin{bmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \\ huv \end{bmatrix}_x + \begin{bmatrix} hv \\ huv \\ hv^2 + \frac{1}{2}gh^2 \end{bmatrix}_y
$$
$$
= \begin{bmatrix} 0 \\ ghH_x \\ ghH_y \end{bmatrix} + \begin{bmatrix} 0 \\ -ru\sqrt{u^2+v^2}/h \\ -rv\sqrt{u^2+v^2}/h \end{bmatrix}
$$
$$
+ \begin{bmatrix} 0 \\ fhv \\ -fhu \end{bmatrix} + \begin{bmatrix} 0 \\ \tau_x \\ \tau_y \end{bmatrix}.
$$
(1)

Here, h is water depth, hu momentum along the x-axis, and hv momentum along the y-axis (see also Fig. 1). Furthermore, g is the gravitational acceleration, H is the
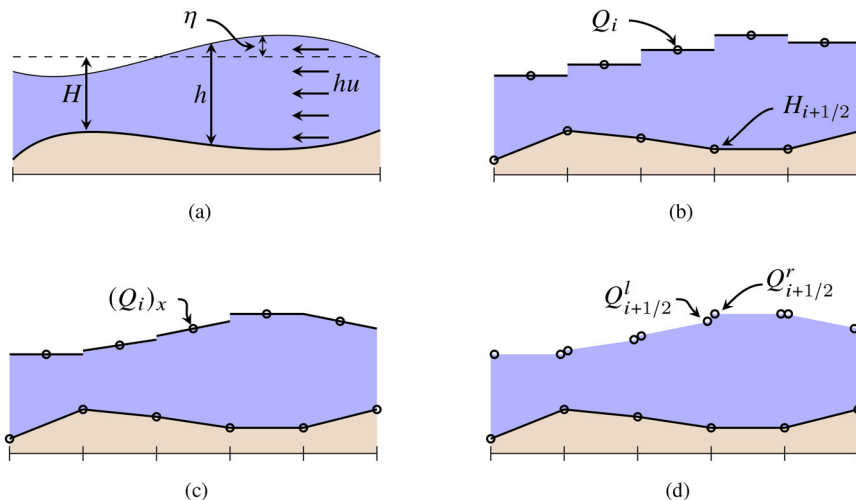
Fig. 1. a) The relationship between the physical variables h, H, η, and hu used in the shallow-water equations, here shown in one dimension. b) The ocean state is discretised in terms of cell average values, whereas the bathymetry is defined at the cell intersections. c) We find the slopes of the ocean state within each cell. d) The flux between the cells is found from two one-sided point values at each interface, as seen from the two adjacent cells.

equilibrium ocean depth measured from a reference sea-surface level, r is the bottom drag coefficient, f is the Coriolis parameter, and τ_x and τ_y are the wind stress along the x- and y-axis, respectively. In vector form, we can write the equations as

$$Q_t + F(Q)_x + G(Q)_y = B(Q) + S(Q) + C(Q) + W(Q). \tag{2}$$

Here, $Q = [h, hu, hv]^T$ is the vector of conserved variables and B, S, C, and W are source terms representing varying bathymetry, bed shear stress, Coriolis force, and wind drag, respectively. Note that this model does not include the atmospheric pressure gradient, which is important in, e.g., storm surge modelling.

By using a finite-volume discretisation on a Cartesian grid to solve (2), we get the following semi-discrete formulation

$$\frac{\partial Q}{\partial t} = -\frac{1}{\Delta x}\left[F_{i+1/2,j} - F_{i-1/2,j}\right] - \frac{1}{\Delta y}\left[G_{i,j+1/2} - G_{i,j-1/2}\right]$$
$$+ B_{i,j} + S_{i,j} + C_{i,j} + W_{i,j}. \tag{3}$$

To evolve the solution in time, we use a second-order strong stability-preserving Runge-Kutta scheme (Gottlieb et al., 2001),

$$Q^*_{i,j} = Q^n_{i,j} + \Delta t M(Q^n)_{i,j}$$
$$Q^{n+1}_{i,j} = \frac{1}{2}(Q^n_{i,j} + (Q^*_{i,j} + \Delta t M(Q^*)_{i,j})), \tag{4}$$

in which M represents the right hand side of (3). The time step is restricted by the CFL condition given by

$$\Delta t \leq \frac{1}{4}\min\left\{\frac{\Delta x}{\max|u \pm \sqrt{gh}|}, \frac{\Delta y}{\max|v \pm \sqrt{gh}|}\right\}, \tag{5}$$

if we assume no Coriolis, bottom drag, and wind drag.[2] The dominant term for oceanographic applications is typically the speed of gravity waves $\sqrt{gh}$, and we scale Δt by a factor $C_{CFL} \in (0, 1]$ in our simulations to account for Coriolis and wind drag.

2.1. Spatial discretisation

The starting point for our simulator is the scheme proposed by Chertock et al. (2018), from now on referred to as CDKLM. It is a modern central-upwind scheme that uses a reconstruction of the physical variables to be able to capture steady-state solutions important for physical oceanography, and includes source terms for varying bathymetry and Coriolis. The original paper includes several idealised test cases and examples demonstrating that the steady states in geostrophic balance are well captured. However, there are several shortcomings that need to be addressed before the scheme can be used for real-world simulations.

The CDKLM scheme is based on the same principles as the MUSCL scheme (van Leer, 1979) to obtain a high-order, total variation diminishing discretisation. For each cell in the domain, we first reconstruct a piecewise (linear) function for our physical variables (see Fig. 1). At the interface between two cells, we evaluate the slope limited function from the right and left cell, and use these two values to compute the flux across the interface. The

scheme uses the Riemann-solver-free central-upwind flux function,

$$F_{i+1/2} = \frac{a^+ F(Q^l_{i+1/2}) - a^- F(Q^r_{i+1/2})}{a^+ - a^-}$$
$$+ \frac{a^+ a^-}{a^+ - a^-} \left[Q^r_{i+1/2} - Q^l_{i+1/2} \right], \qquad (6)$$

to evaluate the numerical flux (Kurganov et al., 2001). Here, a^+ and a^- are the largest positive and negative wave speeds at the interface, respectively.

When adding source terms, it becomes challenging to preserve steady states such as 'lake at rest', i.e.

$$\frac{\partial Q}{\partial t} = 0 \qquad \text{if} \qquad \frac{\partial \eta}{\partial x} = \frac{\partial \eta}{\partial y} = hu = hv = 0, \qquad (7)$$

in which η is the sea-surface deviation from a mean equilibrium level, given by

$$\eta = h - H. \qquad (8)$$

Kurganov and Levy (2002) presented a discretisation of the bottom slope source term $B(Q)$ and a matching reconstruction based on water elevation to capture such steady states. For oceanographic simulation, an important steady state is the so-called geostrophic balance, in which the angular momentum caused by the rotational reference frame is balanced by the pressure gradient:

$$fhv - gh\frac{\partial \eta}{\partial x} = 0, \qquad -fhu - gh\frac{\partial \eta}{\partial y} = 0. \qquad (9)$$

The novel idea in the CDKLM scheme is to reconstruct the face values $Q^l_{i+1/2}$ and $Q^r_{i+1/2}$ based on this geostrophic balance so that the resulting flux $F_{i+1/2}$ also balances the Coriolis force for steady-state solutions. This makes the scheme suitable for simulating ocean currents in a rotating frame of reference.

3. Efficient simulation of real-world ocean currents

Our simulator is implemented using CUDA (NVIDIA, 2019) coupled with PyCUDA (Klöckner et al., 2012) to access the GPU from Python, and we have previously shown that using Python instead of C++ has a negligible performance impact when used like this (Holm et al., 2020). Using the Python ecosystem for tasks such as pre-processing input data and post-processing results has significantly increased our productivity, while still maintaining high performance.

A simulation starts by reading initial conditions, forcing terms, and other grid data from NetCDF files hosted by *thredds* servers at the Norwegian Meteorological Institute. We continue by initialising the simulator, which internally allocates memory and uploads initial conditions

on the GPU. Finally, the main loop steps the simulator forward in time, and occasionally (e.g., every simulated hour) downloads and writes results to NetCDF files on disc.

The rest of this section covers additions and improvements to the CDKLM scheme required for simulating real-world scenarios.

3.1. Efficient parallel formulation of CDKLM

The original CDKLM scheme uses a recursive formulation of the potential energies used for the well-balanced flux reconstruction. This turns the reconstruction procedure into a global operation with a data-dependency spanning the whole domain, making the scheme prohibitively expensive on parallel architectures. The recursive expression can be calculated somewhat efficiently using a prefix sum, as suggested by the authors themselves, but by carefully reformulating the recursive terms it can also be rephrased as a local operation.

The scheme reconstructs h, hu, and hv on each side of a face to calculate the fluxes in (6). However, to capture the rotating steady-state solutions in geostrophic balance, the reconstruction is based on the potential energies,

$$K = g(\eta - V), \qquad L = g(\eta + U), \qquad (10)$$

instead of physical variables. Note that K is the potential energy due to the Coriolis force along the abscissa and that L is the potential along the ordinate. U and V are the primitives of the Coriolis force, given by

$$U_y = \frac{f}{g}u, \qquad V_x = \frac{f}{g}v. \qquad (11)$$

Note that setting $K_x = 0$ and $L_y = 0$ is equivalent to the geostrophic balance from (9). In the following, we detail the reconstruction along the abscissa, but the same derivations apply in the ordinate. The recursive terms appear from the discretisation of U and V, which are defined on the cell faces as

$$V_{i+1/2,j} = V_{i-1/2,j} + \frac{\Delta x}{g}f_{i,j}v_{i,j}, \qquad V_{-1/2,j} = 0. \qquad (12)$$

The Coriolis parameter $f_{k,j}$ is allowed to change both along the x- and y-axis, even though f only varies with latitude (for example, the NorKyst-800 grid is rotated to follow the Norwegian coastline.) This is because we will use a spatially varying latitude, as described in more detail in Section 3.7. Values of V in the cell centres are obtained by taking the mean of the face values, so that the cell-average value of K from (10) becomes

$$K_{i,j} = g\left(\eta_{i,j} - \frac{1}{2}\left(V_{i+1/2,j} + V_{i-1/2,j}\right)\right). \quad (13)$$

By reconstructing the (limited) slopes of K within each cell, the face values of h are found by combining (13) and (8), so that

$$h_{i,j}^E = \frac{1}{g}\left(K_{i,j} + \frac{\Delta x}{2}(K_x)_{i,j}\right) + V_{i+1/2,j} + H_{i+1/2,j}, \quad (14)$$

and

$$h_{i,j}^W = \frac{1}{g}\left(K_{i,j} - \frac{\Delta x}{2}(K_x)_{i,j}\right) + V_{i-1/2,j} + H_{i-1/2,j}. \quad (15)$$

The reconstructed values (14) and (15) of h at the cell faces quickly become a performance bottleneck as long as they depend on calculating the recursive relation in (12). As it turns out, however, we never need to explicitly compute the recursive terms. First of all, notice that (14) and (15) only depend on derivatives of K and not on L. Similarly, the reconstructions of $h_{i,j}^N$ and $h_{i,j}^S$ depend on L and L_y. This means that we are only interested in derivatives of K (and L) in the same direction as the recursive terms for V (and U). The derivatives are limited using the generalised minmod function using the backward, central, and forward differences. The backward difference is given by

$$\frac{K_{i,j} - K_{i-1,j}}{\Delta x} = \frac{g}{\Delta x}\left(\eta_{i,j} - \frac{1}{2}\left(V_{i+1/2,j} + V_{i-1/2,j}\right)\right.$$
$$\left. -\eta_{i-1,j} + \frac{1}{2}\left(V_{i-1/2,j} + V_{i-3/2,j}\right)\right). \quad (16)$$

We see that $V_{i-1/2,j}$ cancels, and focus on the remaining V values. By applying the recursive expression in (12) twice on $V_{i+1/2,j}$ we get

$$\frac{1}{2}\left(-V_{i+1/2,j} + V_{i-3/2,j}\right)$$
$$= \frac{1}{2}\left(-V_{i-1/2,j} - \frac{\Delta x}{g}f_{i,j}v_{i,j} + V_{i-3/2,j}\right)$$
$$= \frac{1}{2}\left(-V_{i-3/2,j} - \frac{\Delta x}{g}f_{i-1,j}v_{i-1,j} - \frac{\Delta x}{g}f_{i,j}v_{i,j} + V_{i-3/2,j}\right)$$
$$= -\frac{\Delta x}{2g}(f_{i-1,j}v_{i-1,j} + f_{i,j}v_{i,j}). \quad (17)$$

Inserted back into (16), the backward difference needed to evaluate $(K_x)_{i,j}$ can be written as

$$\frac{K_{i,j} - K_{i-1,j}}{\Delta x} = \frac{g}{\Delta x}\left(\eta_{i,j} - \eta_{i-1,j} - \frac{\Delta x}{2g}(f_{i-1,j}v_{i-1,j} + f_{i,j}v_{i,j})\right), \quad (18)$$

which no longer contains any recursive terms. Similar derivations can be used for the forward and central differences, and equivalently to obtain $(L_y)_{i,j}$.

After obtaining $(K_x)_{i,j}$ we look at the reconstruction of $h_{i,j}^E$ from (14). We start by inserting the expression for K from (13) into (14) and gather all V-terms,

$$h_{i,j}^E = \eta_{i,j} - \frac{1}{2}\left(V_{i+1/2,j} + V_{i-1/2,j}\right) + \frac{\Delta x}{2g}(K_x)_{i,j}$$
$$+ V_{i+1/2,j} + H_{i+1/2,j}$$
$$= \eta_{i,j} + H_{i+1/2,j} + \frac{\Delta x}{2g}(K_x)_{i,j} + \frac{1}{2}\left(V_{i+1/2,j} - V_{i-1/2,j}\right). \quad (19)$$

Here, the values for $\eta_{i,j}, H_{i,j}$ and $(K_x)_{i,j}$ are known, and by using the recursive expression in (12) we get that

$$\frac{1}{2}\left(V_{i+1/2,j} - V_{i-1/2,j}\right) = \frac{1}{2}\left(V_{i-1/2,j} + \frac{\Delta x}{g}v_{i,j}f_{i,j} - V_{i-1/2,j}\right)$$
$$= \frac{\Delta x}{2g}f_{i,j}v_{i,j}. \quad (20)$$

Inserting (20) back into (19) we see that we get

$$h_{i,j}^E = \eta_{i,j} + H_{i+1/2,j} + \frac{\Delta x}{2g}\left[(K_x)_{i,j} + f_{i,j}v_{i,j}\right], \quad (21)$$

Again, the same derivation can be applied on the other three face values as well. This enables us to express the reconstruction step in terms of only local variables, which unlocks an embarrassingly parallel numerical algorithm.

3.2. Accuracy, precision and GPUs

Today's GPUs can be categorised in two major classes: GPUs meant for the consumer market, and GPUs meant for supercomputers and other professional users. For the most part, these GPUs are close to identical, but there are some key differences. One is price: professional GPUs can cost over ten times more than equivalent consumer market GPUs, making the latter tractable from a financial stand point. Another difference is in the feature set. Professional GPUs offer more memory, error-correcting code (ECC) memory, and significantly higher performance for features such as double precision.

Double precision is not an important feature on consumer market GPUs, and typically offer less than 5% of the performance of single-precision arithmetics. On professional GPUs, on the other hand, the performance is 50% of single precision, which is the same ratio as on CPUs.[3] If we are able to utilise single-precision calculations, we should therefore be able to get the highest performance at minimum monetary cost. We have previously shown that using single precision is sufficiently accurate for most simulation scenarios for the shallow-water equations (Brodtkorb et al., 2010), and recent studies have shown a 40% increase in performance for complex forecasting models when using single precision (Váňa et al., 2017).
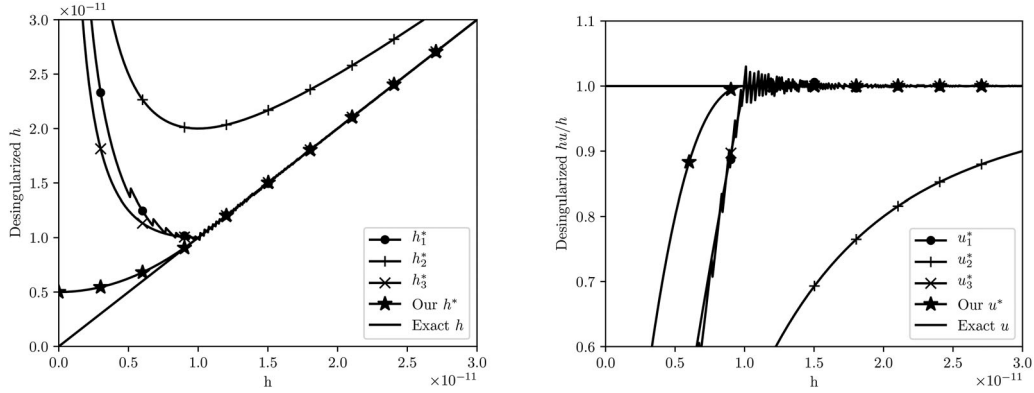
Fig. 2. Desingularization of the quantity $u = h/hu$ using existing and our proposed method with $\kappa = 1.0 \cdot 10^{-11}$ for all cases. We compute the value of hu / h using the different approaches for $u = 1$ and different values of h. Notice how the single precision floating-point errors are clearly visible for u_1^*, that u_2^* has a distinct kink, and that u_3^* significantly underestimates the true magnitude of u.

It is well known that floating-point numbers have a round-off error that increases for larger numbers. A single-precision floating-point number is represented as

$$(-1)^s \cdot 2^{e-127} \cdot (1.0 + f), \tag{22}$$

in which s is the sign bit, e is represented using 8 bits, and f is the fractional part, represented by 23 bits. This formulation means that the distance between two floating-point numbers is smallest close to zero, and increases as e increases. This is important to note in oceanographic simulations, as the water depth, h, can often be thousands of metres, whereas the relevant tidal wave height is perhaps 0.5 metres. The loss in precision due to the water depth makes floating point prohibitively inaccurate if we implement the shallow-water equations as formulated in (1). To counter this loss in precision, we can possibly use double precision (at twice the computational and memory cost), but a better alternative is to reformulate the problem to represent the relevant physical quantities. This can be done by basing our simulation on η from (8) instead of h and use $[\eta, hu, hv]^T$ as our vector of conserved variables. By combining this approach with single-precision floating-point numbers, we get more than sufficient accuracy for the relevant oceanographic simulations.

When the water depth becomes close to zero, the calculation of the particle velocity, $u = hu/h$ can become severely ill conditioned as the expression is prone to large round-off errors. To counter this, Kurganov (2018) and Kurganov and Petrova (2007) have proposed to *desingularize* the term using expressions such as

$$u_1^* = \frac{\sqrt{2}h(hu)}{\sqrt{h^4 + \max(h^4, \kappa^4)}}, \tag{23}$$

$$u_2^* = \frac{h \cdot hu}{h^2 + \kappa^2}, \qquad \text{or} \tag{24}$$

$$u_3^* = \frac{2h \cdot hu}{h^2 + \max(h^2, \kappa^2)}. \tag{25}$$

The suggestion of using $\kappa = \max(\Delta x^4, \Delta y^4)$ has some obvious problems for real-world simulations with $\Delta x > 1$, as pointed out in (Brodtkorb et al., 2012). Furthermore, the desingularization of u here is not very well-behaved. Instead, we propose to use the following formulation

$$\begin{aligned} h^* &= \text{sign}(h)\max(|h|, \min(h^2/(2\kappa) + \kappa/2, \kappa)) \\ u^* &= hu/h^*. \end{aligned} \tag{26}$$

The advantages of this formulation is that (i) κ now controls directly what magnitudes of h to desingularize, (ii) it improves the numerical stability as it avoids the square root and the problematic h^4 term, (iii) it yields a smooth transition between normal and desingularized values, and (iv) the effective magnitude of h^* is controlled. Figure 2 shows the three desingularization strategies and clearly indicates that the desingularized quantity h is better behaved with our proposed approach.

3.3. Global wall boundary conditions

We have implemented wall and periodic boundary conditions to be able to run test cases to assess the numerical correctness of our simulator. For reflective von Neumann-type wall boundary conditions, we use so-called ghost cells that mirror the ocean state across the boundary to enforce a zero flux out of the domain. In the absence of Coriolis forces, wall boundary conditions at $x = 0$ are constructed by setting the ghost cell values according to

$$\begin{aligned} \eta(-x, y) &= \eta(x, y), \qquad hu(-x, y) = -hu(x, y), \\ hv(-x, y) &= hv(x, y). \end{aligned} \tag{27}$$

Naïve implementation of these conditions with the CDKLM scheme leads to gravitational waves along the boundary. Special care is therefore required to balance the Coriolis potential energies in the reconstruction of h,

so that

$$
\begin{aligned}
0 &= h_{0,j}^W - h_{-1,j}^E \\
&= \left(\eta_{0,j} + H_{-1/2,j} - \frac{\Delta x}{2g} \left[(K_x)_{0,j} + f_{0,j} v_{0,j} \right] \right) \\
&\quad - \left(\eta_{-1,j} + H_{-1/2,j} + \frac{\Delta x}{2g} \left[(K_x)_{-1,j} + f_{-1,j} v_{-1,j} \right] \right),
\end{aligned}
\tag{28}
$$

using (21). Here, we see that H cancels immediately, whereas η cancels by the use of (27). To make the last terms cancel, we need to set $f_{-1,j} v_{-1,j} = -f_{0,j} v_{0,j}$, which means that in addition to (27), we need to enforce

$$
f(-x, y) = -f(x, y). \tag{29}
$$

Note also that this results in $(K_x) = 0$ across the boundary from (18).

3.4. Moving wet-dry boundary and land mask

The original CDKLM scheme breaks down on land as the gravitational wave speed $\sqrt{gh}$ becomes imaginary with negative depths. To support a moving wet-dry boundary, we use a similar approach as Kurganov and Petrova (2007) by adjusting the slope of h so that all the reconstructed point values become non-negative. For example, if we get $h_{i,j}^E < 0$ from (21), we adjust the slope to become

$$
(K_x)_{i,j} = -f_{i,j} v_{i,j} - \frac{2g}{\Delta x} \left[\eta_{i,j} + H_{i+1/2,j} \right], \tag{30}
$$

and thus forcing $h_{i,j}^E = 0$ instead.

The moving wet-dry boundary is required for simulating, e.g. tsunamis and other phenomena in which the run-up is important. The approach nonetheless violates the steady-state reconstruction, causing non-physical waves and often also small time step sizes. For static wet-dry boundaries, we have therefore also implemented wall boundaries following a static land mask, i.e. ensuring a zero flux for the shore-line. In our approach, we explicitly set the flux between two neighbouring cells to zero if any one of them is masked. The mask is represented by a unique no-data value in the bathymetry, thereby having no increase in the memory footprint. It should be noted that both of these approaches reduce the reconstruction to first order along the wet-dry boundary.

3.5. Wind forcing and bed friction source terms

The original formulation of CDKLM includes the bed slope source term, $B(Q)$, and Coriolis force, $C(Q)$, and we extend the scheme to also include the bed shear stress, $S(Q)$, and wind forcing, $W(Q)$. Bed shear stress is

discretised using a semi-implicit formulation in the strong stability preserving Runge-Kutta scheme,

$$
\begin{aligned}
Q_{i,j}^* &= \left(Q_{i,j}^n + \Delta t M(Q^n)_{i,j} \right) / \left(1 + \Delta t S(Q_{i,j}^n) \right) \\
Q_{i,j}^{n+1} &= \frac{1}{2} \left(Q_{i,j}^n + (Q_{i,j}^* + \Delta t M(Q^*)_{i,j}) \right) / \left(1 + \frac{1}{2} \Delta t S(Q_{i,j}^*) \right),
\end{aligned}
\tag{31}
$$

instead of (4). This essentially means that we apply half of the bed friction using Q^n, and half of the friction using the predictive step Q^*.

The wind stress comes from the atmospheric wind transferring momentum to the water column. We take the approach of Large and Pond (1981) and use

$$
\vec{\tau} = \frac{\rho_a}{\rho_w} C_D \; |W_{10}| \; W_{10}, \tag{32}
$$

in which $\rho_a = 1.225$ kg/m³ is the specific density of air, $\rho_w = 1025$ kg/m³ is the specific density of sea water, W_{10} is the wind speed at 10 metres, and C_D is the drag coefficient computed as

$$
C_D = 10^{-3} \begin{cases} 1.2 & \text{if } |W_{10}| < 11 \text{ m/s}, \\ 0.49 + 0.065 \; |W_{10}| & \text{if } |W_{10}| \geq 11 \text{ m/s}. \end{cases} \tag{33}
$$

The NorKyst-800 model stores its output, including the wind forcing, every hour. Because the time step of our simulator is significantly smaller than one hour, we use linear interpolation in time to get the most accurate wind stress source term for each time step. Given the wind stresses W_{10}^a at time t_a and W_{10}^b at time t_b, we compute the wind stress at time $t \in [t_a, t_b]$ as a convex combination of the two

$$
W_{10}^n = (1-s) W_{10}^a + s W_{10}^b, \qquad s = (t - t_a)/(t_b - t_a). \tag{34}
$$

In space, we use specialised GPU texture hardware for bilinear interpolation (see e.g. Brodtkorb et al., May 2010), resulting in a trilinearly interpolated wind stress. This is highly efficient, as the GPU has dedicated hardware for bilinear spatial interpolation that also includes caching. An added benefit is that the grid size of the wind stress data does not need to match up with the underlying simulation grid. This further provides us with a choice to reduce the accuracy and memory footprint of the forcing fields, and thus trade better performance against accuracy.

3.6. Nesting the model into NorKyst-800

A common technique in operational oceanography is to nest high-resolution local models within lower-resolution models that cover a larger domain. For instance, the NorKyst-800 model is nested within TOPAZ, which is a
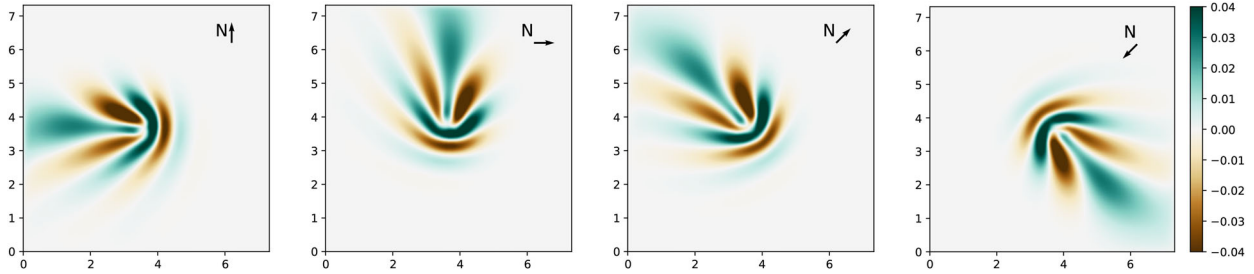
Fig. 3. Planetary Rossby waves generated by a beta plane model for the Coriolis force and with different directions for north. The simulations are initialised by a rotating bump in the middle of the domain, which develops into Rossby waves that propagates westwards with wave energy slowly propagating eastwards. The different figures show that we get the same Rossby waves for arbitrary orientation of our domain. Axes are given in 1000 km, and the colormap shows η in m.

12-16 km resolution HYCOM-based coupled ocean and sea ice model with data assimilation (Xie et al., 2017). In this work, we nest our simulations into the NorKyst-800 model by setting initial ocean state and boundary conditions appropriately. We use the so-called flow relaxation scheme (Davies, 1976), in which we have a relaxation zone,[4] between the internal computational domain and the external boundary conditions. In this region, we solve the shallow-water equations as normal to obtain an internal solution $Q_{i,j}^{int}$, and then gradually nudge it towards an external solution Q^{ext} using

$$Q_{i,j} = (1-\alpha)Q_{i,j}^{int} + \alpha Q^{ext}, \quad \alpha = 1 - \tanh(d_{i,j}/d_0). \tag{35}$$

Here, $d_{i,j}$ is distance in number of cells to the external boundary, and the parameter d_0 is typically set to 2 or 3 to control the fall-off of the tanh relaxation function.

The implementation of the boundary conditions follows the same ideas as for the wind stress, using textures for linear interpolation in time and space. We use float4[5] textures to hold our physical variables, η, hu, hv, because float3 is not supported as a texture format. We furthermore only use two textures, as we pack the north and south boundaries into one texture, and similarly for the east and west.

3.7. Coriolis force

The original CDKLM scheme requires that the grid is aligned with the cardinal directions, which significantly restricts the use of the scheme. For higher latitudes this becomes especially pronounced, as both the latitude and direction towards north will vary across the domain. We have therefore extended the scheme to support both a spatially varying north-vector and a spatially varying latitude, which is required to be able to run realistic simulations for the areas covered by the NorKyst-800. To account for the varying north vector,

we project the momentum onto the local north and east vectors,

$$hu^e = \vec{e} \cdot [hu, hv]^T, \qquad \vec{e} = [\cos(\theta), -\sin(\theta)],$$
$$hv^n = \vec{n} \cdot [hu, hv]^T, \qquad \vec{n} = [\sin(\theta), \cos(\theta)], \tag{36}$$

and use these vectors when computing the source term $C(Q)$ so that

$$C(Q) = f \cdot \begin{bmatrix} 0 \\ \vec{x} \cdot [hv^n, -hu^e]^T \\ \vec{y} \cdot [hv^n, -hu^e]^T \end{bmatrix}. \tag{37}$$

in which the vectors $\vec{x} = [\cos(\theta), \sin(\theta)]$ and $\vec{y} = [-\sin(\theta), \cos(\theta)]$ project the result back into the (x, y)-coordinate system. Note that we also need to use the same approach in the reconstruction of h as discussed in Section 3.1 to maintain the rotational steady states.

Our spatially varying Coriolis parameter is computed from the latitude, $l_{i,j}$, of a cell

$$f_{i,j} = 2\omega \sin(l_{i,j}), \qquad \omega = 7.2921 \cdot 10^{-5} \text{rad/s}, \tag{38}$$

or using a beta-plane model,

$$f_{i,j} = f_{ref} + \beta \cdot (\vec{n} \cdot [(x_i - x_{ref}), (y_j - y_{ref})]^T). \tag{39}$$

Here, f is linearised around a reference point, (x_{ref}, y_{ref}), with a slope β in the direction of $\vec{n}$. Setting β to zero yields a constant Coriolis parameter throughout the domain.

Figure 3 demonstrates how the angle to north plays an important role in the generation of planetary Rossby waves, caused by variations in the Coriolis force using (39). The figure shows the resulting η after long simulations initialised with a rotating Gaussian bump in geostrophic balance in the middle of the domain, similar to the case presented in Holm et al. (2020). Here, we have used a domain consisting of 350×350 cells with $\Delta x = \Delta y = 20$km, a depth of $H = 50$m, and the centre of the domain corresponding to approximately $33°$ north, with $f = 8 \cdot 10^{-4}$ and $\beta = 2 \cdot 10^{-11}$. The simulation ends after approximately 35 days. The figure clearly shows the effect
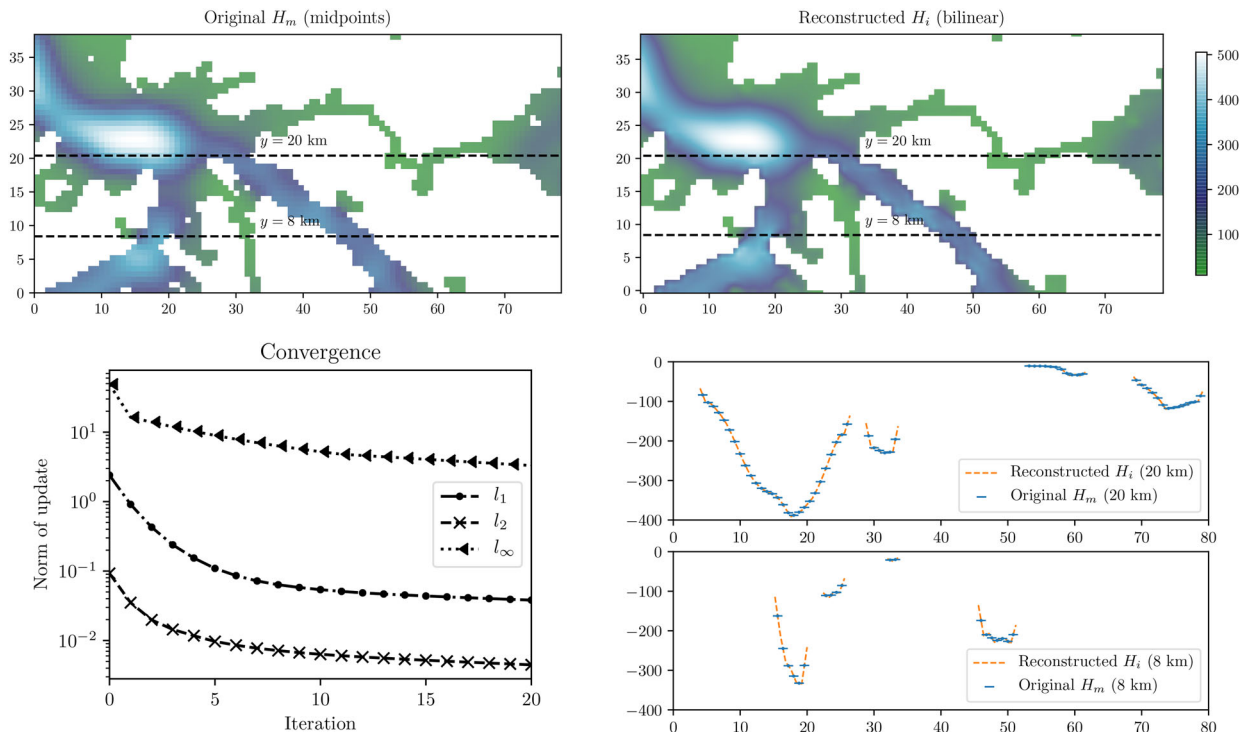
Fig. 4. Reconstruction of the bathymetry at intersection points. Starting with the initial bathymetry H_m defined at cell midpoints (upper left), we compute the H_i values at intersections (upper right). The lower-left figure shows the convergence of our reconstruction algorithm, and the reconstructed bathymetry is plotted together with the original in the lower-right figure. The top two figures have axes units in number of cells, and the *x*-axis in the bottom right is similarly in number of cells. The *y*-axis in the bottom right figure shows the depth. Notice that small features such as islands can disappear in the top right figure, but that the bathymetry does not appear to suffer from smoothing caused by the interpolation procedure.

of varying the north vector and hence the Coriolis source term.

3.8. Bathymetry

The bathymetry in the NorKyst-800 model is given as cell midpoint values, and can not be used directly in our framework as CDKLM requires the bathymetry to be defined as a piecewise bilinear surface specified by point values at the cell intersections. It is still desirable that the averages of the resulting four intersection values end up to be equal to the NorKyst-800 cell average values, as large deviations can disturb the balanced relations in the provided ocean state. Constructing such a piecewise bilinear surface from the cell averages is an ill-posed problem, and a simple averaging of cell values excessively smears important features such as deep fjords and straits.

We have therefore devised a pragmatic iterative algorithm, which is based on reconstruction of slopes for each cell, similar to the reconstruction procedure in our numerical shallow-water scheme. Our initial guess is computed as follows:

1. Compute the local gradient in each cell so that

$$\left[H_{i,j}^x, H_{i,j}^y\right] = \nabla H_{i,j}. \tag{40}$$

2. Evaluate the piecewise planar function at the four corners of each cell,

$$
\begin{aligned}
H_{i-1/2,j-1/2}^{i,j} &= H_{i,j} - \frac{1}{2}H_{i,j}^x - \frac{1}{2}H_{i,j}^y, \\
H_{i+1/2,j-1/2}^{i,j} &= H_{i,j} + \frac{1}{2}H_{i,j}^x - \frac{1}{2}H_{i,j}^y, \\
H_{i-1/2,j+1/2}^{i,j} &= H_{i,j} - \frac{1}{2}H_{i,j}^x + \frac{1}{2}H_{i,j}^y, \\
H_{i+1/2,j+1/2}^{i,j} &= H_{i,j} + \frac{1}{2}H_{i,j}^x + \frac{1}{2}H_{i,j}^y.
\end{aligned}
\tag{41}
$$

3. Average the estimate from the four cells meeting at one intersection point so that

$$
H_{i+1/2,j+1/2}^* = \frac{1}{4}\left(H_{i+1/2,j+1/2}^{i,j} + H_{i-1/2,j+1/2}^{i+1,j} \right.
\left. + H_{i-1/2,j-1/2}^{i+1,j+1} + H_{i+1/2,j-1/2}^{i,j+1}\right).
\tag{42}
$$

After obtaining the initial guess, we start an iterative procedure consisting of two stages. The first stage minimises the difference between computed and true midpoint
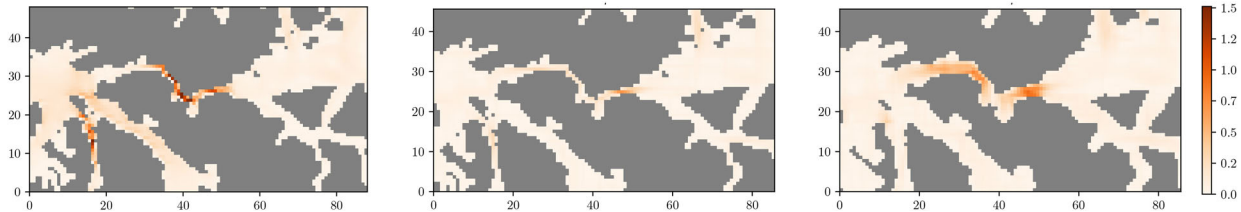
Fig. 5.   Widening of narrow straits, in which the land mask is eroded using binary erosion. The left figure shows particle velocities in the Tjeldsund strait in Lofoten/Vesterålen between the main land and Hinnøya (the bathymetry is shown in Fig. 4). Left shows the NorKyst-800 reference, centre shows original (reconstructed) bathymetry, and the right figure shows the result with one grid cell land erosion. The strait width and geometry limits the amount of water passing through it in the original bathymetry, but by eroding one cell of the land mask the amount of water is much closer to the reference. The axis units are in number of grid cells.

values, and the second dampens oscillations created by the first stage. We start by computing the difference between our estimate and the target bathymetry as

$$\Delta H_{i,j} = H_{i,j} - H^*(i,j), \tag{43}$$

in which $H^*(i,j)$ means that we evaluate the piecewise bilinear estimate at midpoints. We then apply (40)–(42) to compute an estimate of the difference at intersections, $\Delta H^*_{i+1/2,j+1/2}$, and subtract this difference from our initial guess,

$$H^*_{i+1/2,j+1/2} := H^*_{i+1/2,j+1/2} + \Delta H^*_{i+1/2,j+1/2}. \tag{44}$$

This brings our updated midpoints of $H^*(i,j)$ closer to the target $H_{i,j}$, but it also results in unwanted oscillations, as there is nothing that limits the slopes of this piecewise bilinear surface. We therefore smooth H^*, and repeat the procedure until the update between two subsequent iterations is sufficiently small. Around ten iterations is typically sufficient to create a piecewise bilinear surface that lies close to the target. Figure 4 shows the result of our approach applied to the bathymetry from NorKyst-800 in the innermost parts of Vestfjorden (see also Section 5.2).

CDKLM is a full two-dimensional scheme, yet it exhibits a grid orientation bias close to the land mask. For example, we have experienced that the scheme has difficulties in propagating the momentum through narrow straits and canals that are only one or two cells wide. To ameliorate this, we can use binary erosion on the land mask to erode one row of cells at a time as shown in Fig. 5. We extrapolate η into these areas using grey dilation,[6] and set the water depth equal the land value, which in the NorKyst-800 model is 5 metres. The process can be repeated to erode the land mask even more, and is required for tidal waves to properly propagate in features such as narrow fjords and straits. We have found that using single iteration is sufficient to propagate tides in and out of the fjords in the NorKyst-800 model.

3.9. Refinement and coarsening of grid

The possibility to run simulations at different resolutions is attractive to quickly obtain low-resolution preliminary results, and also to resolve more of the dynamics for more detailed and accurate results with higher resolution. We have included functionality in our simulation framework to increase and decrease the grid resolution with factors of two to support both of these scenarios. We use a static grid refinement/coarsening performed on the initial conditions themselves, but the same approach can be used for efficient adaptive mesh refinement following the same approach as Saetra, Brodtkorb and Lie (Sætra et al., 2015).

The physical variables are represented by cell average values, and refinement is based on a slope-limited reconstruction within each cell, which we evaluate at the refined grid cell centres. As the bathymetry is specified by a piecewise bilinear surface, the high-resolution representation of the bathymetry evaluates the surface on the refined cell intersections. Low-resolution representation of the physical variables on a coarsened grid is simply the average of the original cell values, and intersection values for the bathymetry are sampled from the corresponding original intersection values. With a factor two coarsening, both these operations are well-defined.

Note that the boundary conditions and wind forcing terms are independent of the mesh resolution due to our use of GPU textures.

4. Performance and accuracy of GPU implementation

We have previously evaluated the standard CDKLM scheme on several test cases which target oceanographic dynamics relevant for the shallow-water equations when implemented on a Cartesian grid (Holm et al., 2020). In this work, however, we have reformulated the scheme in terms of η instead of h, added several new source terms, and enabled varying latitude and north vector. It is
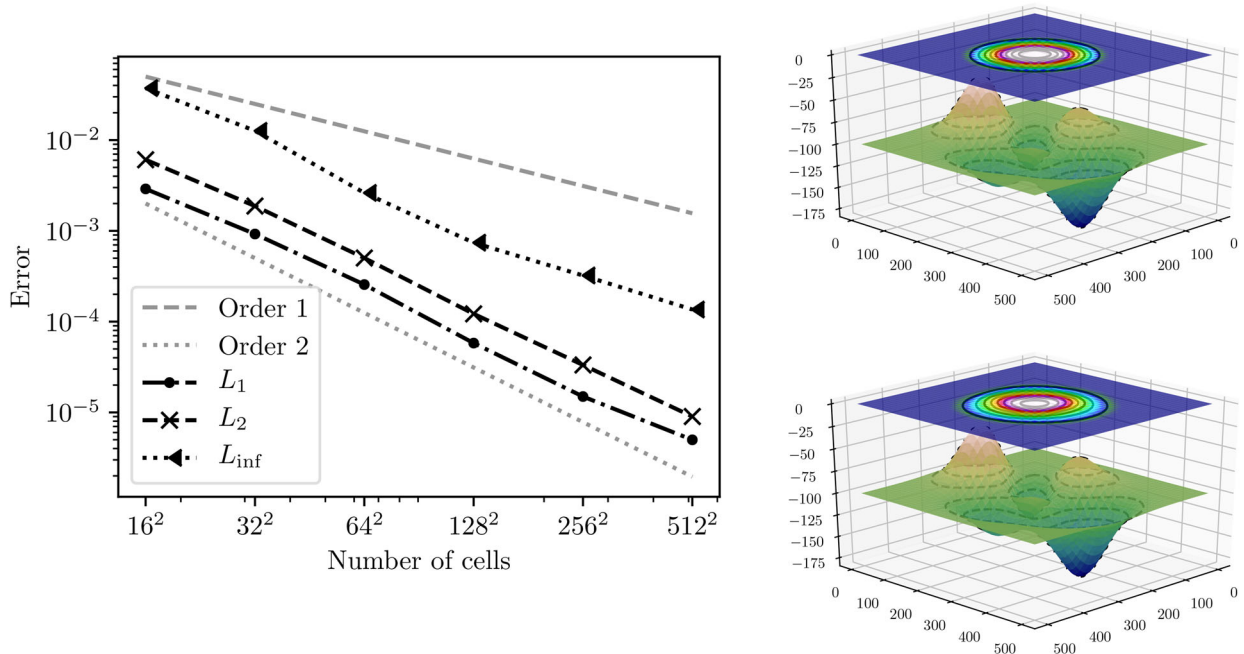
Fig. 6. Convergence plot of our numerical scheme. The numerical case has a water disturbance consisting of a cosine bump at the centre of the domain, and the bathymetry consists of the smooth Matlab peaks function. The Coriolis force varies spatially across the domain according to variations in latitude and direction towards north. The left figure shows the self convergence of the numerical error compared to the reference solution with 1024^2 cells. To the right we show the bathymetry with the sea-surface level initially (top) and at the end of the simulation (bottom). The horizontal axis are in kilometres, and the vertical axis is in metres. Notice that the height of the bump is slightly reduced in the lower figure (smaller white area).

therefore important to revisit the grid convergence test. We also measure computational performance of our simulation framework through assessing its weak scaling.

4.1. Convergence of the numerical scheme

To evaluate numerical convergence, we define a benchmark case covering $500 \text{ km} \times 500 \text{ km}$. We model the bathymetry after the Matlab peaks function,

$$\text{peaks}(s,t) = 3(1-s)^2 e^{-s^2-(t+1)^2} - 10(s/5 - s^3 - t^5)e^{-s^2-t^2}$$
$$- 1/3 e^{-(s+1)^2-t^2}, \qquad s,t \in [-3,3]$$

$$(45)$$

which yields a non-symmetric smooth surface. We then define the equilibrium ocean depth as

$$H(x,y) = 100 + 10 \cdot \text{peaks}\left(\frac{6x}{500} - 3, \frac{6y}{500} - 3\right), \qquad (46)$$

with x and y given in km, so that the depth varies between 34 and 181 metres (see Fig. 6). We initialise the sea-surface level with a bump centred in the domain,

$$\eta(r) = \begin{cases} \frac{1}{2}\left(1 + \cos\left(\frac{r}{c}\pi\right)\right), & \text{if } r \le c, \\ 0, & \text{if } r > c, \end{cases} \qquad (47)$$

in which $r = \sqrt{(x-250)^2 + (y-250)^2}$ is the distance from the centre of the domain, and c controls the size of the bump, set to 300 km in our case. We assume that the North pole is located at $(3000 \text{ km}, 3000 \text{ km})$, which means that the angle between the y-axis and the north vector varies from $33.7°$ to $56.3°$ across the domain and that the latitude varies from $56.0°$ to $67.4°$ north. These parameters roughly correspond to the North Sea. Including land inevitably yield only first-order accurate fluxes, and we have thus deliberately not included such areas in our grid convergence test.

The simulation is run for 800 seconds, after which we compute the difference for the whole domain between the solution and a reference computed on a 1024×1024 grid. Figure 6 and Table 1 show that we achieve second-order accuracy in both the L_1 and the L_2 norm, whilst the convergence in L_{inf} is somewhat slower.

4.2. Computational performance

To assess the computational performance, we continue to use the same benchmark, but run the simulation twice as long. We also measure the cost of downloading the results from the GPU to the CPU. Figure 7 shows the

Table 1. Error of computed solutions at different resolutions and numerical reduction order for each refinement. The time step is fixed throughout these simulations.

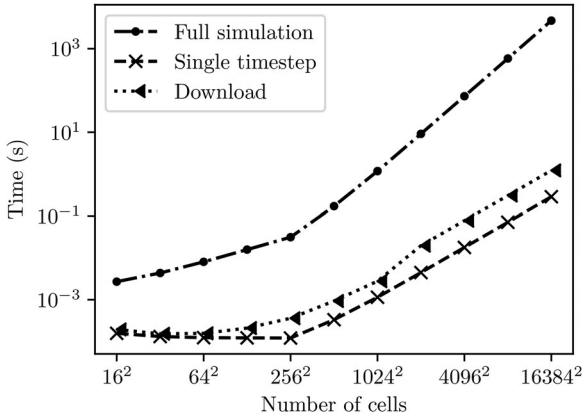| Resolution	Δt	L_1	Order	L_2	Order	L_{inf}	Order
16^2	1600/16	0.002898	–	0.006072	–	0.037367	–
32^2	1600/32	0.000923	1.65	0.001877	1.69	0.012651	1.56
64^2	1600/64	0.000256	1.85	0.000502	1.90	0.002621	2.27
128^2	1600/128	0.000058	2.14	0.000122	2.04	0.000741	1.82
256^2	1600/256	0.000015	1.96	0.000033	1.87	0.000322	1.20
512^2	1600/512	0.000005	1.58	0.000009	1.89	0.000136	1.25

Fig. 7. Weak scaling of our simulator. For small domain sizes, the runtime is dominated by overheads, but for domain sizes larger than approximately 512×512, we see that the computational time scales with the number of cells. For each doubling of our domain, we perform approximately eight times as many operations (four times as many cells, and twice the number of time steps).

Table 2. Performance for our simulator. The simulation time is approximately eight times larger when we double the resolution. This is because we have four times as many cells and twice as many timesteps to reach the same end time. The download increases by a factor four, which is to be expected as we transfer four times as many data elements.

Resolution	Simulation time	Factor	Download time	Factor
512^2	0.169710	–	0.000929	–
1024^2	1.170814	6.9	0.002773	3.0
2048^2	9.133564	7.8	0.019642	7.1
4096^2	72.832028	8.0	0.077541	3.9
8192^2	582.599290	8.0	0.305405	3.9
16384^2	4695.904546	8.0	1.218880	4.0

results for domains with 16^2 to 16384^2 cells, running on a Tesla P100 GPU. We see that for the larger domain sizes, we obtain perfect weak scaling, as a factor two grid refinement leads to a factor eight increase in computational complexity (the time step size is cut in half due to the CFL condition, and we have four times the number

of cells). Cases with less than 512^2 cells are dominated by overheads and hence, the cost of a single time step is close to constant. The problem size required to make overheads negligible and obtain perfect weak scaling varies between different GPUs and their configurations. Time for downloading data scales linearly with the problem size, as is expected. Again, we see the same behaviour for small domains as for the computational performance (Table 2).

Our current implementation is designed for running massive ensembles (Holm et al., 2020), and therefore does not utilise multiple GPUs for a single simulation. However, we have previously shown that finite-volume schemes such as the one discussed here are very well suited for multi-GPU simulations (Sætra and Brodtkorb, 2010).

5. Real-world simulations

Using the NorKyst-800 model system, the Norwegian Meteorological Institute issues daily ocean forecasts for the complete Norwegian coast and makes them publicly available through thredds servers and various web pages, as well as transferring to other authorities. In addition to the 3 D velocities of the ocean currents, these forecasts also provide the 2 D depth-averaged currents, which we use as initial conditions along with the sea-surface level. The forecasts also contain wind forcing, bathymetry, land masks, longitudes, latitudes, and angle towards north for each cell. This means that it is simply a matter of accessing the file and cutting out the correct sections to be able to initialise our simulator. To start a simulation within our framework, we simply need the full url to the specific forecast we want to nest our simulation within, and specify a desirable sub-domain.

We run our simulations on the three different sub-domains shown in Fig. 8. The simulation cases run independently, but use the same global data set for initialisation and forcing. Our first simulation is in the Norwegian Sea with no land values within the domain, and is therefore a fairly simple test for using real initial and boundary conditions. The second case is centred on the Lofoten
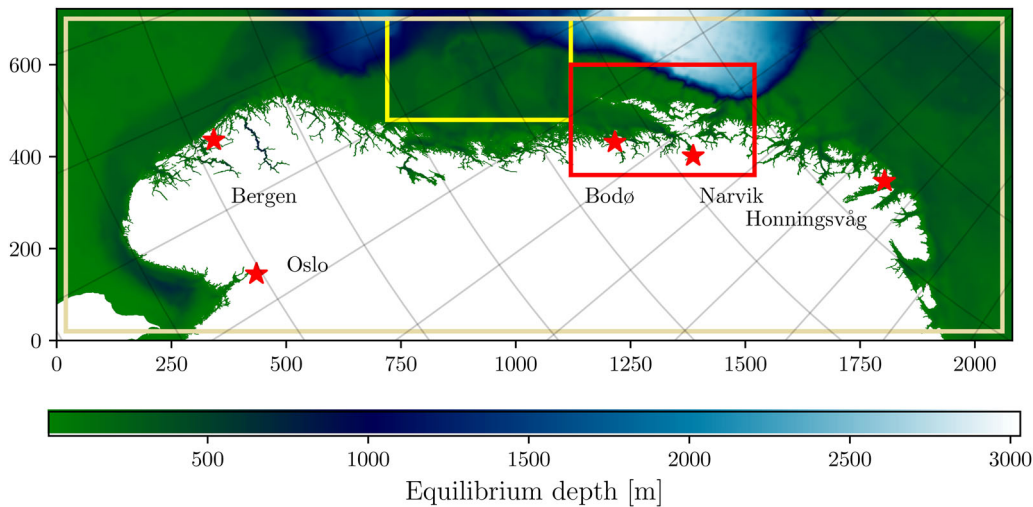
Fig. 8. The domain covered by the NorKyst-800 model showing the equilibrium depth off shore. The location of Case 1 in the Norwegian Sea (yellow rectangle), Case 2 around the Lofoten archipelago (red rectangle), and Case 3 (beige rectangle) within the NorKyst-800 model domain, with red stars marking the locations for sea-level predictions. The values on the *x*- and *y*-axes are given in km, and the latitude-longitude grid shows how the direction towards north changes throughout the domain.

archipelago in Northern Norway, which is dominated by a complex coastline consisting of narrow fjords and a large number of islands, and serves as a challenging test for our reconstruction of the bathymetry and land mask. Finally, we run almost the complete domain, leaving only 25 cells in each direction to serve as boundary conditions. With the entire Norwegian coastline, it becomes important to account for the difference in the north vector throughout the domain (see the longitudinal lines and north arrows in Fig. 8), and we should see the tides following the coast as Kelvin waves. The last case is also used to make sea-level predictions at five different locations, indicated as stars in Fig. 8.

All three cases are run with three different resolutions; a high resolution with a grid refined to 400 m cells, the original 800 m resolution of NorKyst-800, and a low resolution grid consisting of 1600 m cells. We use an adaptive time step, and update the time step every 20 minutes according to the CFL condition in (5) scaled by $C_{CFL} = 0.8$. Each dataset provided from the NorKyst-800 forecasts covers 23 hours, which means that we only have boundary conditions to run simulations for the same time range.

5.1. Case 1: Norwegian sea

The first case runs an open water domain in the Norwegian Sea and tests our ability to use the NorKyst data as initial and boundary conditions. The sea-surface level changes significantly during the simulation time range as the tidal waves enter and exit the domain, and this dynamic is largely dependent on the boundary conditions. The velocity field on the other hand, is dominated by already present features of slowly rotating dynamics, and will not change much during the simulation time range. This enables us to test both that the initial conditions are correctly captured, and how well our simulator maintains complex rotating structures over time.

Figure 9 shows simulation results of the Norwegian Sea for all three different grid resolutions after 23 hours, compared to the reference solution from NorKyst-800. First, we see that the contour lines for the sea-surface level in the left column show the same values at similar places for all simulations. This shows that the boundary conditions allow tidal waves to propagate correctly in and out of the domain as intended. The right column shows the corresponding particle velocities. Note that the initial state for both the high-resolution and original runs are visually inseparable from the NorKyst-800 model state, whereas the low-resolution initial conditions slightly smear out the sharper features for the velocity. At 23 hours, we see that the high-resolution model to a large degree maintains the sharp features from NorKyst-800. This shows that even though we run a barotropic model, we are able to maintain some of the baroclinic signals present in the reference model. We suggest that this is due to a combination of initialisation of the baroclinic motion and a well-balanced numerical scheme for geostrophic flow. With the original resolution, however, the particle velocities resemble a slightly blurred image of the reference solution, and in the low-resolution simulation, the features are blurred even more. Also, we are starting to see clear signs of grid effects both internally and from the boundary.
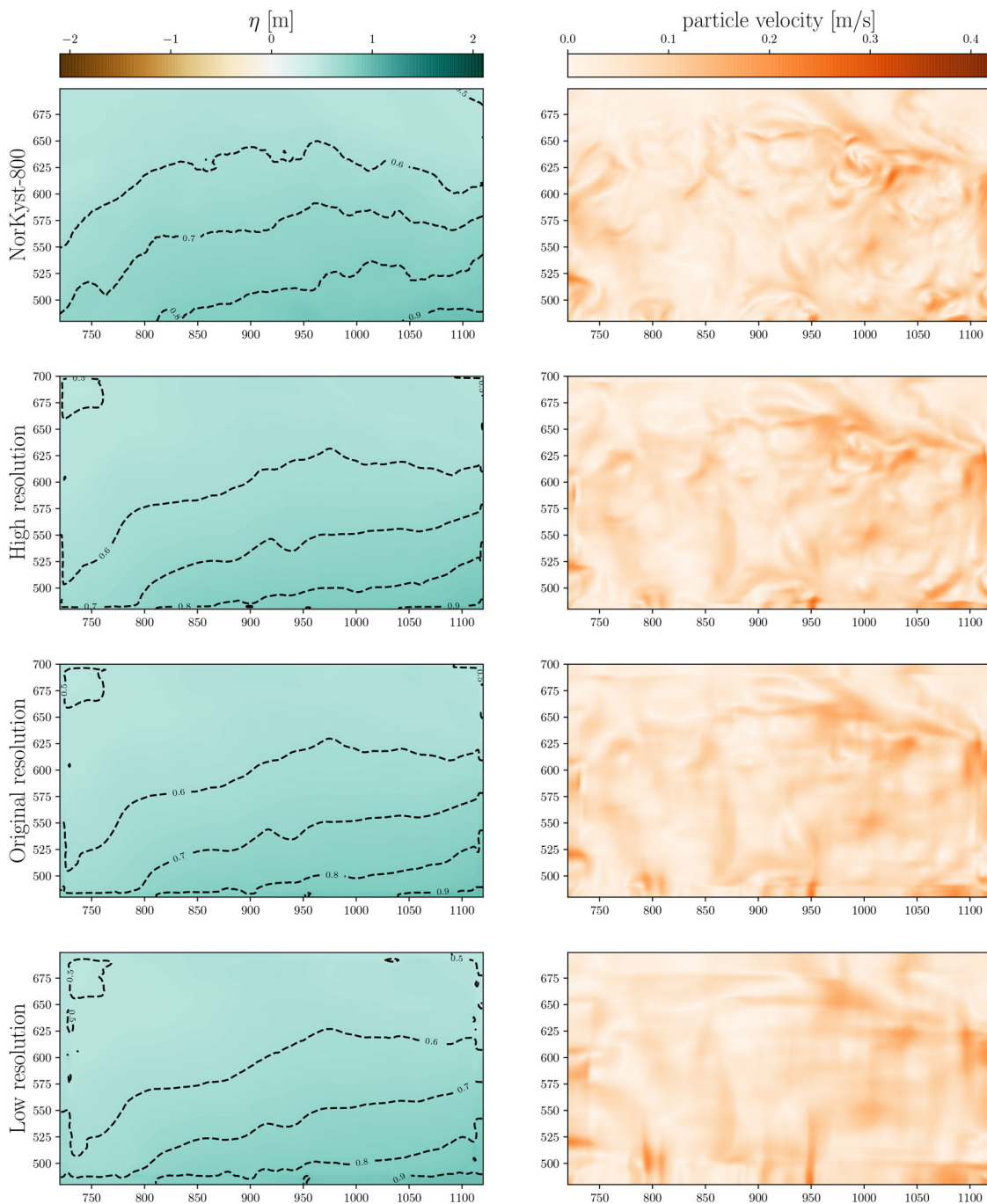
Fig. 9. Case 1 of the Norwegian Sea after 23 hours, comparing the reference solution from NorKyst-800 with our simulations using three different grid resolutions. All simulations obtain sea-surface level (left) similar to the reference solution, meaning that the boundary conditions allow the tides to correctly enter and exit our domain. From the particle velocities (right), we see that only the high-resolution model is able to maintain sharp features, whereas the low-resolution simulation has strong signs of grid effects. The values on the x- and y-axes are in km relative to the location in the complete domain.

5.2. Case 2: Lofoten

The second case is centred around the Lofoten archipelago in Northern Norway. This region has a complex coast line consisting of a large number of fjords, straits and islands. In addition to the aspects tested in the first case, this case tests the reconstruction of the bathymetry at intersections and the handling of dry zones through
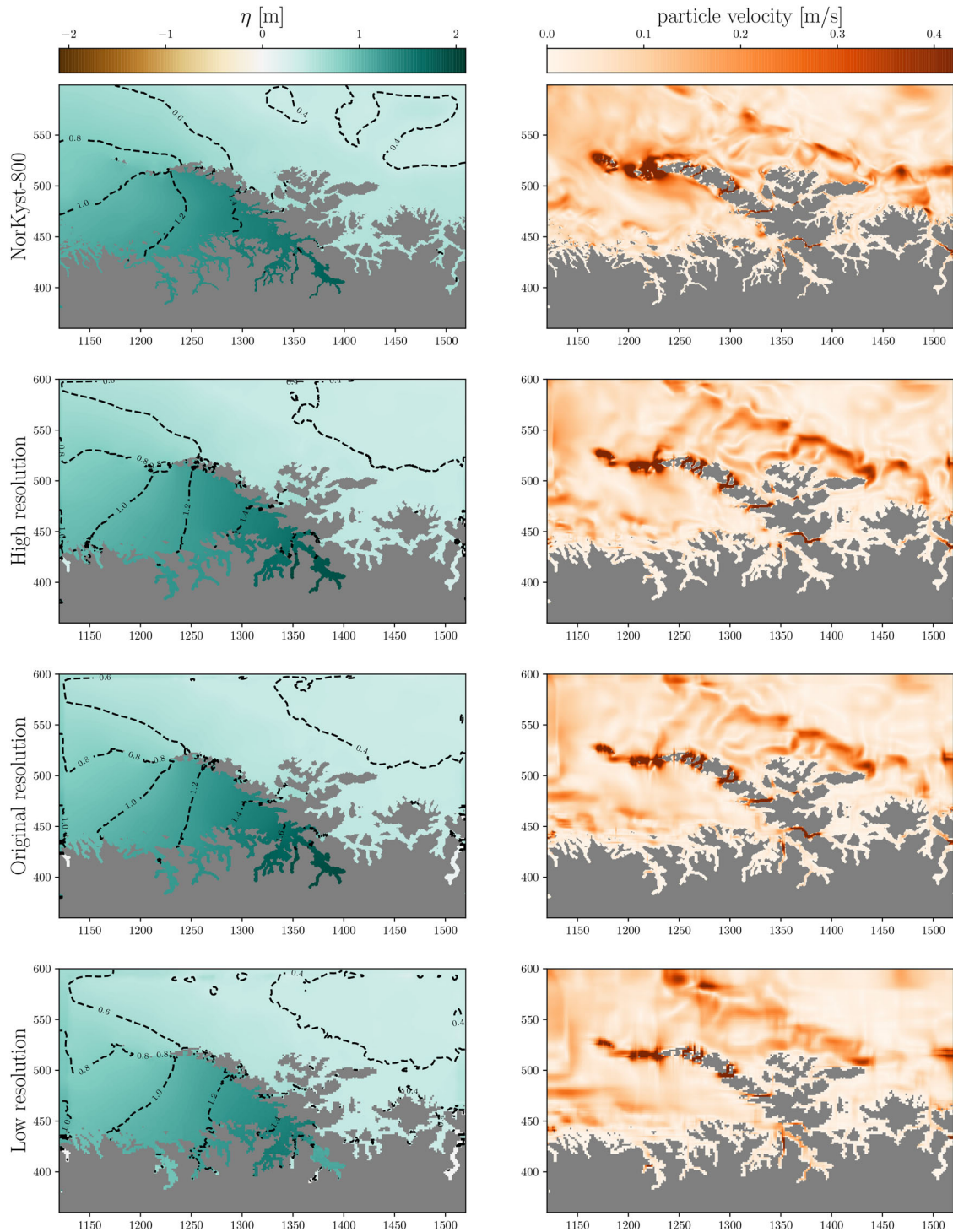
Fig. 10. Case 2 around the Lofoten archipelago after 23 hours, comparing the reference solution from NorKyst-800 with our simulations using three different grid resolutions. The values on the *x*- and *y*-axes are in km relative to the location in the complete domain. Our sea-surface levels (left column) are similar but slightly higher within Vestfjorden compared to the reference solution. From the particle velocities (right), we see that we get stronger currents than the reference. The level of details in our simulations are stronger with higher grid resolution, whereas the low-resolution simulation blurs the structures and are influenced by grid effects.
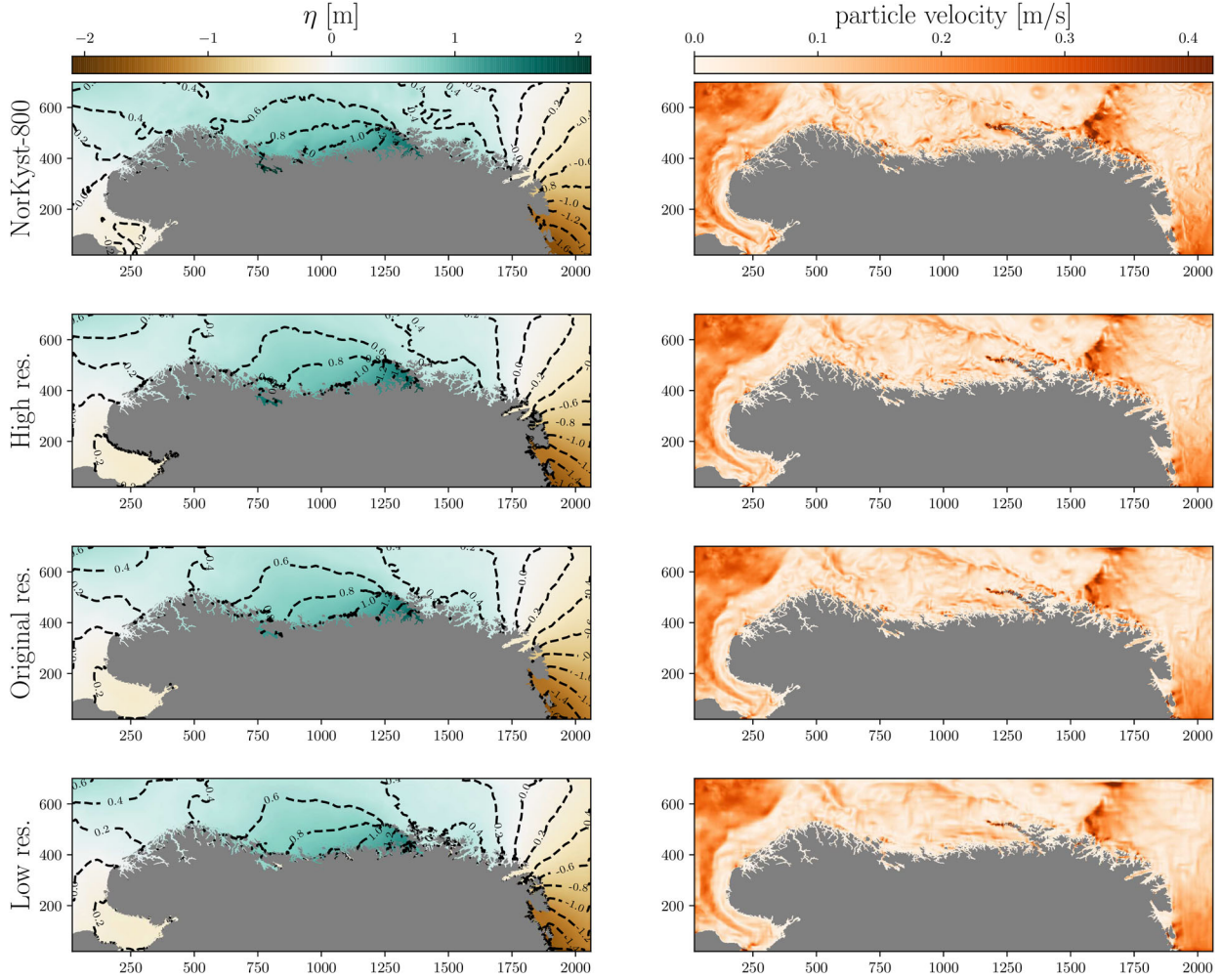
Fig. 11. Simulation result for Case 3 containing almost the complete domain used by NorKyst-800 after 23 hours. The top row shows the reference solution from NorKyst-800, followed by our simulation results using three different grid resolutions below. The values on the x- and y-axes are in km relative to the location in the complete domain. The sea-surface levels (left) show how the tide varies along the coast, with low tide in the Oslofjord (lower left corner), high tide in the middle of the domain, and low tide again towards the border with Russia (lower right corner). Our results are consistent with the reference solution for all grid resolutions. The particle velocities (right column) are qualitatively similar for the reference solution (NorKyst-800), and our simulations with high and original resolution. Even at this scale, it is apparent that the low resolution grid smears the features in the ocean currents.

the complex land mask. We expect to see a Kelvin wave following the coast, but because the Lofoten archipelago protrudes into the Atlantic, some of the wave will be partially trapped and amplified in Vestfjorden between Lofoten and the main land. Furthermore, as can be seen from the depth map in Fig. 8, the edge of the continental shelf crosses through the upper right corner of the subdomain, and we can here see a line of eddies driven by baroclinic instabilities.

We again run simulations with the three different grid resolutions, and Fig. 10 shows the final simulation state after 23 hours, with a high tide in Vestfjorden. All three simulations show that the contour lines for the sea-

surface level are slightly off compared to the reference solution, with marginally more water into the fjord. Similarly as in Case 1, the low-resolution simulation is unable to preserve the fine structures in the currents and shows grid effects. The original and high grid resolutions are also unable to preserve the exact eddy structure as found in the reference solution, but we can instead more clearly see a consistent coastal current with natural irregularities. This is consistent with what we can expect using a barotropic model. These results also show a larger area with high particle velocity around the smallest islands in our simulations, possibly due to the reconstructed

Table 3. Timing results different resolutions of running the complete Norwegian coast (Case 3), excluding initialisation, with the original, high and low resolution on different computer types.

Resolution	Num cells	Average Δt	Laptop GeForce 840M Maxwell (2014)	Desktop GeForce GTX780 Kepler (2013)	Server Tesla P100 Pascal (2016)
Low	541 875	1.85 s	9m 22 s	1 m 29 s	47 s
Original	2 167 500	0.92 s	53m 49 s	10 m 42 s	3 m 13 s
High	8 670 000	0.46 s	6h 10 m 09 s	1 h 23 m 22 s	23 m 21 s

Table 4. Breakdown of run time for simulating 23 hours of the complete Norwegian coast (Case 3) on the desktop with a GeForce GTX780 GPU. The column with time spent writing to disc includes downloading from the GPU and all over overhead required for this operation.

Resolution	Initialisation	Simulation 1 hour	Write one time step to disc (fraction of simulation time)
Low	15.6 s	3.4 s (85.3%)	0.5 s (14.7%)
Original	11.8 s	25.9 s (92.3%)	2.0 s (7.7%)
High	18.3 s	209.5 s (96.2%)	8.0 s (3.8%)

bathymetry, which widens some straits by the land-mask erosion.

5.3. Case 3: Norway

Our third case consists of almost the complete domain covered by the NorKyst-800 model. We leave out the 25 outermost cells and use these as boundary conditions. With the complete domain, we cover a large enough area to see the Kelvin waves entering in the southwestern part of the domain and travelling north.

Figure 11 shows the simulation results after 23 hours in the same manner as the earlier figures. We now see from the sea-surface level (left column) that there is a low tide in the middle of the domain, whereas there are high tides both in the north at the Russian border and in the Oslo fjord in the lower left corner. The contour lines are similar for all three grid-resolutions, and the particle velocities (right column) are in general consistent with the reference solution from NorKyst-800. Due to the scale of the domain, it is harder to compare the fine details in Fig. 11, but the large scale features from NorKyst-800 are found in both the high and original grid resolutions. As we have seen in the other cases, the low-resolution simulation maintains less details. It should be noted that by zooming into the same areas simulated in Case 1 and 2, we have seen that the results from running the complete domain are similar within each resolution as to the results shown in Figs. 9 and 10.

Table 3 reports the run time for simulating this case on a laptop, a desktop, and a server GPU. The laptop contains a dedicated GeForce 840 M GPU, which is powerful for a laptop but on the low end of the GPU performance spectrum. It completes the simulation in slightly less than 54 minutes for the original grid resolution of NorKyst-800, which is comparable to the time required for 512 CPU cores to run the NorKyst-800 simulation (Christensen, 2020).[7] On a desktop with a slightly old GeForce 780GTX gaming GPU, the same simulation is carried out in approximately 10 minutes. Tesla P100 is a GPU often found in supercomputers, thus belonging to the very high-end of the performance (and price) spectrum, and runs the 23-hour forecast in approximately 3 minutes. Note that for the high-resolution scheme, the time step is on average just 0.46 seconds, meaning that the results shown in Figs. 9–11 are obtained after 180 000 time steps, using only single-precision floating-point operations.

Table 4 takes a closer look at the simulation time of Case 3 by comparing the time for spent initialising, running the actual simulation for one hour, and writing the result to disc for one time step. The initialisation time is measured under the assumption that the NetCDF file from NorKyst-800 is available on disc locally, as downloading from the thredds server depends on your internet connection and therefore of less interest. First of all, we see that initialising the original resolution costs approximately the same as half an hour of simulation. With higher or lower resolution, initialisation is slightly more expensive as the initial currents, sea-surface level and bathymetry need to undergo pre-processing. Furthermore, we see that initialisation constitutes a considerable amount of total run time when using a low grid
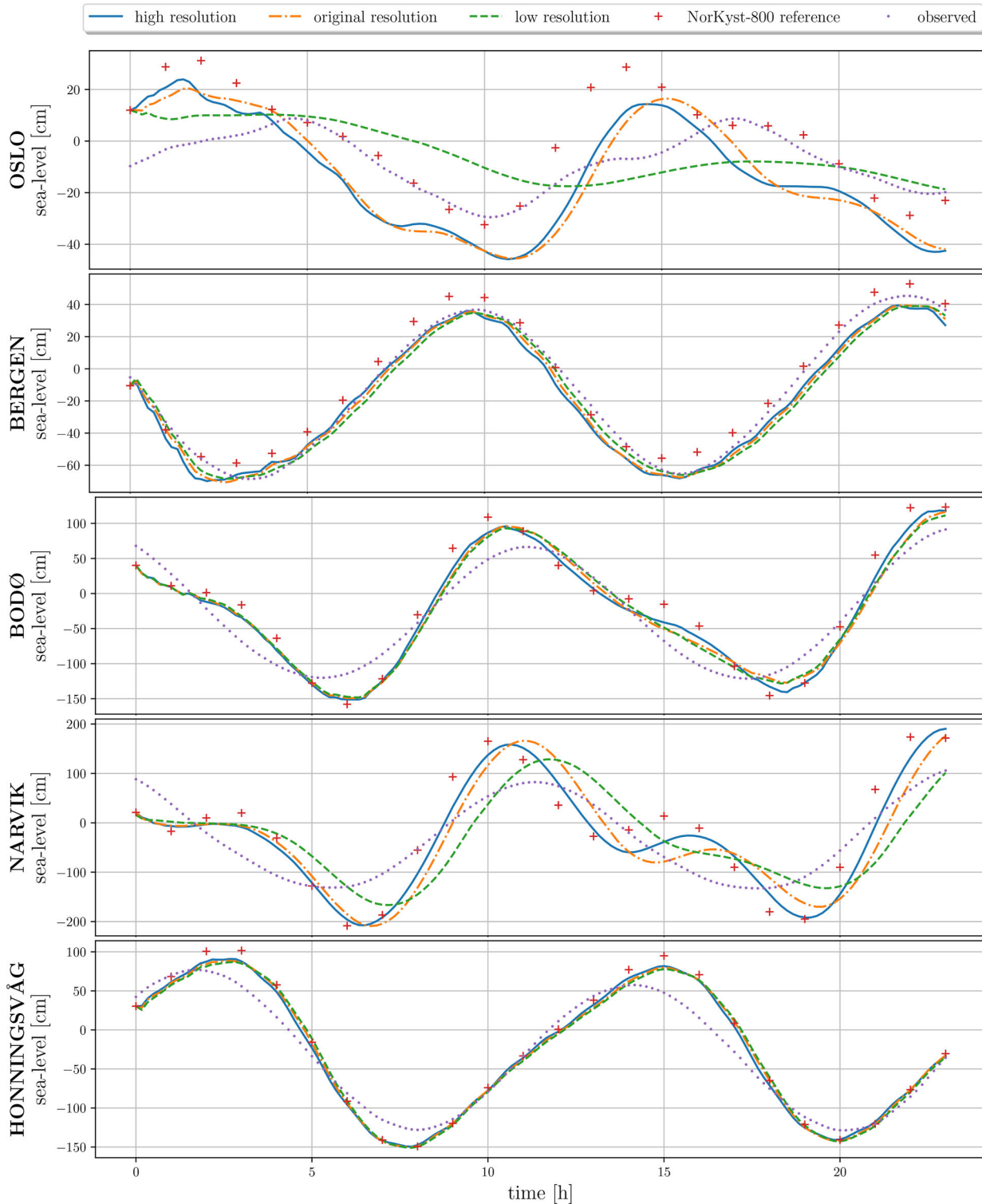
Fig. 12.   Tidal predictions, reference NorKyst-800 results, and official gauge measurements every five minutes at selected locations generated by the simulations discussed in Case 3. All grid resolutions capture the tide as predicted by NorKyst-800 with only minor discrepancies at coastal locations, such as Bergen, Bodø and Honningsvåg. The grid resolution comes much more into play deep inside the fjords, such as at Oslo and Narvik. Here, the low-resolutions grid no longer manages to represent the bathymetry well enough for the tide to enter the fjord correctly (especially in the very narrow Oslo fjord), but we still get fair results with the high-resolution grid. The weakly dotted line is actual observed sea-surface level at the respective locations in the time range of the forecast. It should be noted that official tidal forecasts have additional physical terms, which is the main reason why the NorKyst-800 is off.

resolution, but less so with high resolution. The table also shows us that simulation dominates over writing results to disc for all resolutions. Note that the cost of writing to file scales with the number of cells, meaning that the fraction of total run time spent in writing the results to file is reduced by the square root of the number of cells. This matches the measurements presented in 4.2.

5.4. Tidal forecast verification and validation

The oceanographic phenomena best captured and maintained by the shallow-water equations are arguably tides and storm surges. We therefore use Case 3 to generate tidal forecasts for five locations along the Norwegian coast (see Fig. 8) and compare these with the hourly values from the NorKyst-800 dataset. The locations have been chosen based on the availability of actual sea-level observation stations, which record data every 5 minutes, and we use these to show the realism in the forecasts generated by both NorKyst-800 and the current model. It should be noted that the sea level variation depends on more physical processes than those modelled by both NorKyst-800 and our code, such as atmospheric pressure and other friction terms (see also e.g. (Döös et al., 2004),). We therefore expect a discrepancy between the forecasts and the observations. For us, it is therefore most relevant to use the NorKyst-800 result as a reference solution. The five locations (see Fig. 8) are Bergen and Honningsvåg close to the open sea; Oslo and Narvik located in the innermost parts of long fjords; and Bodø, which is affected by how the Lofoten archipelago catches the tidal waves.

Figure 12 shows the tidal forecasts generated from Case 3 for each of the five locations, sorted from south to north. First of all, we see that our forecasts at Honningsvåg are very well in accordance with NorKyst-800. The same is the case in Bergen, even though our simulations here consistently give slightly too low values for η. At Bodø, we see that NorKyst-800 gives a delay in the transition from high to low tide between hours 14 and 16, and we can see weak indications of this in our high-resolution simulation as well. All these locations are at the coast, which makes the tides here relatively easy to capture, but it should still be noted that they are all protected by islands. The tides in Oslo and Narvik are seen to be harder to capture, as these two cities are located at the innermost parts of two long fjords, and we get larger differences between the different grid resolutions. In Narvik, we get a large improvement by using the high-resolution grid over the original and low resolutions, but overall, our results are fairly close to the reference solution. Oslo is a particularly hard case, as it is positioned within a very narrow and shallow fjord. This leads to

larger relative discrepancies here, even though the absolute difference is roughly the same as in Narvik. We see that the shape of our high-resolution result is a bit different than for the reference solution. The tidal signal predicted by the low-resolution grid in Oslo is very weak, as the fjord almost becomes closed off at this resolution. Finally, note that our solutions are well behaved already from the start, meaning that we manage to initialise our simulations in a balanced state.

6. Summary

We have presented a GPU simulation framework suitable for real-world oceanographic applications. The framework is based on a modern high-resolution finite-volume method for the shallow-water equations (Chertock et al., 2018) with new adaptations and extensions to handle moving wet-dry fronts, land mask, bottom friction, wind forcing, varying north vector, varying latitude, and external boundary conditions. The numerical algorithm is also improved to facilitate for efficient implementation on GPUs using single-precision floating-point operations and includes an efficient reformulation of a problematic recursive term. The framework is designed to be initialised from operational ocean forecasts issued by the Norwegian Meteorological Institute from a three-dimensional ocean model.

We have presented second-order grid convergence for a synthetic but challenging benchmark containing complex topography and varying north vector. We have also shown that the computational performance follows the expected weak scaling.

Finally, we have validated the simulation framework through three different real-world cases along the Norwegian coastline, initialised from ocean forecasts produced by the NorKyst-800 operational model with an 800 m × 800 m horizontal resolution. Our results demonstrate that the framework manages to maintain fine rotational structures, especially when increasing the grid resolution to 400 m × 400 m. On a coarse grid (1600 m grid cells), however, the features are lost and the simulations get dominated by grid effects. The tidal forecasts show that we are able to predict the observed tides relatively well at coastal locations on any of the three grid resolutions used, when compared to the NorKyst-800 reference data. We see, however, that within long fjords, our results are improved when increasing the grid resolution, compared to using the original horizontal resolution of the NorKyst-800 model.

CRediT author statement

A. R. Brodtkorb: Conceptualization, Methodology, Software, Investigation, Writing - Original Draft, Writing - Review & Editing, Visualization, Project administration, Supervision, H. H. Holm: Conceptualization, Methodology, Software, Investigation, Writing - Original Draft, Writing - Review & Editing, Visualization.

Notes

1. The source code presented in this paper, along with Jupyter Notebooks (Kluyver et al. 2016) used to create the plots and results, is available as open source under GPL version 3, and has been published on Zenodo under DOI: 10.5281/zenodo.4056327. We have used the NorKyst-800 forecast for 2019-07-16 from the Norwegian Meteorological Institute in this work, available on https://thredds.met.no/thredds/dodsC/fou-hi/norkyst800m-1h/NorKyst-800m_ZDEPTHS_his.an.2019071600.nc. The observed sea-surface levels are obtained from Se Havnivå, https://www.kartverket.no/sehavniva/, provided by the Norwegian Mapping Authority, Hydrographic Service. The data sets are available under the Creative Commons 4.0 BY International license as described on https://www.met.no/en/free-meteorological-data/Licensing-and-crediting and https://www.kartverket.no/en/data/Terms-of-use/.
2. The spatial discretization of the numerical fluxes in (Chertock et al., 2018) reduces to the fluxes presented in (Kurganov and Levy, 2002) when friction terms and the Coriolis parameter is set to zero.
3. The Nvidia GeForce GTX780, used as an example of a consumer GPU in this study, has a performance of 4.1 teraFLOPS in single precision, and 0.17 teraFLOPS in double precision. The recent Nvidia TITAN RTX has a performance of 16.3 teraFLOPS in single precision, and 0.51 teraFLOPS in double precision. The professional GPU used in this study, the Nvidia Tesla P100, has a performance of 9.5 teraFLOP in single precision, and 4.8 teraFLOPS in double precision. Other consumer grade and professional grade GPUs follow the same trend.
4. Also known as a sponge layer.
5. Textures on GPUs are originally designed to hold the colour of one pixel as the colour channels red, green, blue, and alpha, hence float4.
6. Grey dilation computes the maximum value of a small neighbourhood (typically 3×3 cells) for each location.
7. Please note that the ROMS model is far more complex and can properly represent the ocean state. Our simplified model is only capable of capturing the barotropic dynamics and, e.g., not capable of creating long-term forecasts or capture salinity and temperature driven dynamics.

Acknowledgements

References

Albretsen, J., Sperrevik, A., Staalstrøm, A., Sandvik, A. and Vikebø, F. 2011. NorKyst-800 report no. 1: User manual and technical descriptions. Technical Report 2, Fisken og Havet, Institute of Marine Research

Asanovic, K., Bodik, R., Catanzaro, B. C., Gebis, J. J. et al. 2006. *Parry Husbands, Kurt Keutzer, David a Patterson, William Lester Plishker, John Shalf, Samuel Webb Williams.* The landscape of parallel computing research: A view from berkeley. New York, NY: Association for Computing Machinery.

Bertin, X. 2016. Storm surges and coastal flooding: status and challenges. *La Houille Blanche* **2**, 64–70. doi:10.1051/lhb/2016020

Brodtkorb, A., Dyken, C., Hagen, T., Hjelmervik, J. and Storaasli, O. May 2010. State-of-the-art in heterogeneous computing. *J. Sci. Program.* **18**, 1–33.

Brodtkorb, A., Hagen, T., Lie, K.-A. and Natvig, J. 2010. Simulation and visualization of the Saint-Venant system using GPUs. *Comput. Visual. Sci.* **13**, 341–353. doi:10.1007/s00791-010-0149-x

Brodtkorb, A., Sætra, M. and Altinakar, M. 2012. Efficient shallow water simulations on GPUs: Implementation,

visualization, verification, and validation. *Comput. Fluids* **55**, 1–12. doi:10.1016/j.compfluid.2011.10.012

Castro, M., González-Vida, J., Macías, J. and Ortega, S. 2015. and M de la Asunción. Tsunami-hysea: a gpu-based model for tsunami early warning systems. In *Proc XXIV Congress on Differential Equations and Applications, June, Cádiz, Spain*, 8–12.

Chertock, A., Dudzinski, M., Kurganov, A. and Lukácová-Medvidová, M. 2018. Well-balanced schemes for the shallow water equations with Coriolis forces. *Numer. Math.* **138**, 939–973. Dec doi:10.1007/s00211-017-0928-0

Christensen, K. 2020. Head of Division for Ocean and Ice, Norwegian Meteorological Institute. [personal email communication].

Davies, H. 1976. A lateral boundary formulation for multi-level prediction models. *Q. J. R. Meteorol. Soc.* **102**, 405–418.

de la Asunción, M., Mantas, J. and Castro, M. 2011. Simulation of one-layer shallow water systems on multicore and CUDA architectures. *J. Supercomput.* **58**, 206–214. Nov doi:10.1007/s11227-010-0406-2

Dong, T., Dobrev, V., Kolev, T., Rieben, R., Tomov, S. and co-authors. 2014. A step towards energy efficient computing: Redesigning a hydrodynamic application on CPU-GPU. In *2014 IEEE 28th International Parallel and Distributed Processing Symposium*, IEEE, pp. 972–981.

Döös, K., Nycander, J. and Sigray, P. 2004. Slope-dependent friction in a barotropic model. *J. Geophys. Res* **109**,

Flowerdew, J., Horsburgh, K., Wilson, C. and Mylne, K. 2010. Development and evaluation of an ensemble forecasting system for coastal storm surges. *Q. J. R. Meteorol. Soc.* **136**, 1444–1456. doi:10.1002/qj.648

Gottlieb, S., Shu, C.-W. and Tadmor, E. 2001. Strong stability-preserving high-order time discretization methods. *SIAM Rev.* **43**, 89–112. doi:10.1137/S003614450036757X

Grasset, J., Audouin, Y., Longshaw, S., Moulinec, C. and Emerson, D. R. 2019. Porting and optimising telemac-mascaret for the openpower ecosystem. In *Proceedings of the 2019 Emerging Technology Conference, Editors: MK Bane and V. Holmes*.

Guide to storm surge forecasting. 2011. Technical Report WMO-No 2011. 1076, World Meteorological Organisation.

Hagen, T., Henriksen, M., Hjelmervik, J., and Lie, K.-A. 2007. *How to Solve Systems of Conservation Laws Numerically Using the Graphics Processor as a High-Performance Computational Engine*. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 211–264.

Holm, H., Brodtkorb, A. and Sætra, M. 2020. GPU computing with Python: Performance, energy efficiency and usability. *Computation* **8**, 4. doi:10.3390/computation8010004

Holm, H., Brodtkorb, A., Broström, G., Christensen, K. and Sætra, M. 2020. Evaluation of selected finite-difference and finite-volume approaches to rotational shallow-water flow. *CiCP* **27**, 1234–1274. doi:10.4208/cicp.OA-2019-0033

Holm, H., Sætra, M. and Brodtkorb, A. 2020. Data assimilation for ocean drift trajectories using massive ensembles and GPUs. In *International Conference on Finite Volumes for Complex Applications*, Springer, pp. 715–723.

Huang, S., Xiao, S., and Feng, W.-C. 2009. On the energy efficiency of graphics processing units for scientific computing. In *2009 IEEE International Symposium on Parallel & Distributed Processing* IEEE. pp. 1–8.

Huxley, C. and Syme, B. 2016. TUFLOW GPU-best practice advice for hydrologic and hydraulic model simulations. In *37th Hydrology & Water Resources Symposium 2016: Water, Infrastructure and the Environment*, Engineers Australia, pp. 195–203.

Klingbeil, K., Lemarié, F., Debreu, L. and Burchard, H. 2018. The numerics of hydrostatic structured-grid coastal ocean models: State of the art and future perspectives. *Ocean Modell.* **125**, 80–105. doi:10.1016/j.ocemod.2018.01.007

Klöckner, A., Pinto, N., Lee, Y., Catanzaro, B., Ivanov, P. and co-authors. 2012. PyCUDA and PyOpenCL: A scripting-based approach to GPU run-time code generation. *Parallel Comput.* **38**, 157–174. doi:10.1016/j.parco.2011.09.001

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M. and co-authors. 2016. Jupyter notebooks - a publishing format for reproducible computational workflows. In *Positioning and Power in Academic Publishing: Players, Agents and Agendas* (eds. F. Loizides and B. Schmidt), Netherlands: IOS Press, pp. 87–90.

Kurganov, A. 2018. Finite-volume schemes for shallow-water equations. *Acta Numer.* **27**, 289–351. doi:10.1017/S0962492918000028

Kurganov, A. and Levy, D. 2002. Central-upwind schemes for the Saint-Venant system. *Esaim: M2an.* **36**, 397–425. doi:10.1051/m2an:2002019

Kurganov, A. and Petrova, G. 2007. A second-order well-balanced positivity preserving central-upwind scheme for the Saint-Venant system. *Communications in Mathematical Sciences* **5**, 133–160. 03 doi:10.4310/CMS.2007.v5.n1.a6

Kurganov, A. and Tadmor, E. 2000. New high-resolution central schemes for nonlinear conservation laws and convection–diffusion equations. *Comput. Phys.* **160**, 241–282. doi:10.1006/jcph.2000.6459

Kurganov, A., Noelle, S. and Petrova, G. 2001. Semidiscrete central-upwind schemes for hyperbolic conservation laws and Hamilton–Jacobi equations. *SIAM J. Sci. Comput.* **23**, 707–740. doi:10.1137/S1064827500373413

Large, W. and Pond, S. 1981. Open ocean momentum flux measurements in moderate to strong winds. *J. Phys. Oceanogr.* **11**, 324–336. doi:10.1175/1520-0485(1981)011<0324:OOMFMI>2.0.CO;2

Madec, G. and the NEMO team. 2008. *NEMO ocean engine*. Note du Pôle de modélisation, Institut Pierre-Simon Laplace (IPSL), France, No 27, ISSN No 1288–1619.

Martinsen, E. A. and Engedahl, H. 1987. Implementation and testing of a lateral boundary scheme as an open boundary condition in a barotropic ocean model. *Coastal Eng.* **11**, 603–627. doi:10.1016/0378-3839(87)90028-7

MIKE Powered by DHI. 2019. MIKE 21 graphical processing units (GPU) benchmarking report 2019. Technical report, DHI.

Natvig, J. 2006. *High-resolution methods for conservation laws in the geosciences*. PhD thesis, University of Oslo.

NVIDIA. 2019. NVIDIA CUDA C programming guide version 10.1, 2019.

Parna, P., Meyer, K. and Falconer, R. 2018. GPU driven finite difference WENO scheme for real time solution of the shallow water equations. *Comput. Fluids* **161**, 107–120. doi:10.1016/j.compfluid.2017.11.012

Qi, Z., Wen, W., Meng, W., Zhang, Y. and Shi, L. 2014. An energy efficient OpenCL implementation of a fingerprint verification system on heterogeneous mobile device. In *2014 IEEE 20th International Conference on Embedded and Real-Time Computing Systems and Applications*, IEEE, pp. 1–8.

Qin, X., LeVeque, R. and Motley, M. 2019. Accelerating an adaptive mesh refinement code for depth-averaged flows using GPUs. *J. Adv. Model. Earth Syst.* **11**, 2606–2628. doi:10.1029/2019MS001635

Rakowsky, N., Androsov, A., Fuchs, A., Harig, S., Immerz, A. and co-authors. 2013. Operational tsunami modelling with TsunAWI – recent developments and applications. *Nat. Hazards Earth Syst. Sci.* **13**, 1629–1642. doi:10.5194/nhess-13-1629-2013

Røed, L. 2019. *Atmospheres and Oceans on Computers*. Switzerland: Springer International Publishing.

Sætra, M. and Brodtkorb, A. 2010. Shallow water simulations on multiple GPUs. In *International Workshop on Applied Parallel Computing*. Springer, pp. 56–66.

Sætra, M., Brodtkorb, A. and Lie, K.-A. 2015. Efficient GPU-implementation of adaptive mesh refinement for the shallow-water equations. *J. Sci. Comput.* **63**, 23–48. doi:10.1007/s10915-014-9883-4

Shchepetkin, A. and McWilliams, J. 2005. The regional oceanic modeling system (ROMS): a split-explicit, free-surface, topography-following-coordinate oceanic model. *Ocean Modell.* **9**, 347–404. doi:10.1016/j.ocemod.2004.08.002

van Leer, B. 1979. Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method. *J. Comput. Phys.* **32**, 101–136. doi:10.1016/0021-9991(79)90145-1

Váňa, F., Düben, P., Lang, S., Palmer, T., Leutbecher, M. and co-authors. 2017. Single precision in weather forecasting models: An evaluation with the IFS. *Mon. Wea. Rev.* **145**, 495–502. doi:10.1175/MWR-D-16-0228.1

Xie, J., Bertino, L., Counillon, F., Lisaeter, K. and Sakov, P. 2017. Quality assessment of the TOPAZ4 reanalysis in the Arctic over the period 1991–2013. *Ocean Sci.* **13**, 123–144. doi:10.5194/os-13-123-2017