Data Assimilation for Ocean Drift Trajectories Using Massive Ensembles and GPUs

Håvard H. Holm, Martin L. Sætra, and André R. Brodtkorb

Abstract In this work, we perform fully nonlinear data assimilation of ocean drift trajectories using multiple GPUs. We use an ensemble of up to 10000 members and the sequential importance resampling algorithm to assimilate observations of drift trajectories into the underlying shallow-water simulation model. Our results show an improved drift trajectory forecast using data assimilation for a complex and realistic simulation scenario, and the implementation exhibits good weak and strong scaling.

Key words: Particle filters, finite-volume methods, shallow-water simulations

MSC (2010)**:** 62M99, 60G35, 35L65, 65M08, 76B15, 68N19

1 Introduction

We present a proof-of-concept framework for performing fully nonlinear data assimilation of ocean drift trajectories into a shallow-water model. Forecasting drift trajectories in the ocean is an integral part of offshore preparedness services, and the forecasts are used in, e.g., search and rescue operations, oil spill tracking, and operations involving large floating structures [1]. Our approach is to use massive ensembles of simplified ocean models and assimilate observations using a particle filter based on the sequential importance resampling algorithm [2]. We first generate a massive ensemble of perturbed ocean states and simulate each ensemble member forward in time until we have an observation. Next, we use the particle filter to discard ensemble members that match poorly with the observation, and then reinitialize

Håvard H. Holm
SINTEF Digital, Mathematics and Cybernetics, Norway, e-mail: havard.heitlo.holm@sintef.no
Norwegian University of Science and Technology, Department of Mathematical Sciences, Norway,

Martin L. Sætra and André R. Brodtkorb
Norwegian Meteorological Institute, Norway, e-mail: {martinls, andreb}@met.no
Oslo Metropolitan University, Department of Computer Science, Norway,

Author accepted manuscript version of the publication by Håvard H. Holm, Martin L. Sætra & André R. Brodtkorb. In Finite Volumes for Complex Applications IX - Methods, Theoretical Aspects, Examples. 2020. Published version: https://doi.org/10.1007/978-3-030-43651-3_68

the discarded members based on the simulated states that have a good match. We continue the simulation until the next available observation and repeat the process.

Particle filters as used here are embarrassingly parallel and require synchronization only when resampling individual ensemble members. We therefore use MPI to distribute ensemble members to different nodes, and each member is simulated forward in time using a modern explicit finite-volume scheme. The scheme is implemented on the GPU in a massively data-parallel fashion and includes all the complex source terms required for oceanographic simulations of real-world domains [3].

Our experiments show significantly increased forecast skill compared to both deterministic and Monte Carlo simulations, and we are able to run experiments with 10 000 ensemble members on the Nvidia DGX-2 server, which has 16 GPUs [4]. The implementation exhibits good weak and strong scaling and is possible to extend with more complex perturbation methods with minimal effort.

2 Data assimilation of ocean drift observations

Sequential importance resampling is an example of a particle filter for fully nonlinear data assimilation (see the recent review paper by Vetra-Carvalho et al. [5] on ensemble-based data assimilation techniques). A benefit of the algorithm is that, contrary to e.g., the ensemble Kalman filter [6], it does not manipulate variables of individual simulations that match well with available observations. This means that the resulting ensemble contains ocean states that are consistent with respect to the physics of the model. Furthermore, particle filters do not make any assumptions on linearity in the physical model or Gaussian probability distributions. However, the algorithm requires a large number of ensemble members as the probability that an individual ensemble member matches an observation is small. In fact, the required number of members increases exponentially with the number of observations [7, 2]. This means that it is most suitable for nonlinear problems with few observations, which is the typical situation for our target application area.

General ocean circulation models, such as ROMS [8], conserves mass, three dimensional momentum, salinity, and temperature, and operational setups typically require large computational resources to run even a single simulation. Herein, we investigate using simplified ocean models through the two-dimensional shallow-water equations, which were used operationally in the early days of computational oceanography [9]. These simplified ocean models are suitable for short-term forecasts in which the ocean can be modeled as a barotropic fluid, and they can be efficiently simulated using GPUs. A further challenge is that even the operational models often have limited forecast abilities due to by uncertain initial conditions, model parameters and forcing. By using a simplified model instead of the full three dimensional equations, we can afford to run a much larger ensemble of perturbed ocean states that can give us a more detailed description of the uncertainties in ocean forecasts that can be used as a complement to the current operational methods [10].

We have developed a GPU-based simulation framework that uses operational ocean forecasts for initial and boundary conditions, bathymetry, and forcing [3].

The framework is an extension of the high-resolution, central-upwind finite-volume scheme proposed by Chertock et al. [11], which is well-balanced with respect to steady states in which the Coriolis force balances a non-zero momentum and water surface displacement (the geostrophic balance). The scheme uses H as the water depth, η as the deviation from mean sea level, and hu and hv as the momentum along the abscissa and ordinate, respectively. We can perturb this ocean state using the approach in [12], in which we first generate a smooth random field, $\Delta\eta$, for each ensemble member, representing deviations of the ocean surface elevation. We continue by computing the momentum required to balance this perturbation, namely

$$\Delta hu_{j,k} = -\frac{gH_{j,k}}{f_{j,k}} \frac{\Delta\eta_{j,k+1} - \Delta\eta_{j,k-1}}{2\Delta y}, \qquad \Delta hv_{j,k} = \frac{gH_{j,k}}{f_{j,k}} \frac{\Delta\eta_{j+1,k} - \Delta\eta_{j-1,k}}{2\Delta x}, \quad (1)$$

and finally add these perturbations to the state variables.

Using the perturbations from (1), we generate an ensemble of ocean states and use the ensemble $\psi^n = \{\psi_0^n, \psi_1^n, ..., \psi_N^n\}$ at time t_n as an approximation to the probability density function (pdf) $p(\psi^n)$ of our ocean state. If we have an observation y^n of part of the state (e.g., one drift trajectory), we can improve the probabilistic forecast by using the conditional pdf $p(\psi^n|y^n)$. Using Bayes theorem, $p(\psi^n|y^n) = p(y^n|\psi^n)p(\psi^n)/p(y^n)$, we can write $p(\psi^n|y^n)$ as a weighted ensemble[1]:

$$p(\psi^n|y^n) \propto \sum_{i=1}^{N} \frac{p(y^n|\psi_i^n)}{\sum_{j=1}^{N} p(y^n|\psi_j^n)} \delta(\psi^n - \psi_i^n) = \sum_{i=1}^{N} w_i^n \delta(\psi^n - \psi_i^n), \quad (2)$$

in which δ is the Dirac's delta function. The weights, w_i^n, reflect how well ensemble member i matches the observation, and members with very low weights have a negligible contribution to $p(\psi^n|y^n)$. Sequential importance resampling therefore discards members with low weights and duplicates members with high weights to maintain a higher sample density in the high-probability areas.

A challenge with sequential importance resampling is the so-called curse of dimensionality, as we are operating in a very high-dimensional space. The particle filter is prone to ensemble collapse, in which the ensemble quickly reduces into only a very few significant states and thereby only has marginally better predictive skill than a purely deterministic simulation [7, 2]. This means that we need a much larger number of ensemble members compared to the number of observations. A major benefit, however, is that the particle filter makes no changes to the states of individual ensemble members during the data assimilation phase. This means that the perturbation strategy is not limited by the data assimilation method.

Fig. 1 gives an overview of the ensemble prediction system used in this paper. We use the sea-surface elevation and vertically integrated ocean currents from the operational ocean forecast provided by the ROMS-based NorKyst-800 model system [13] as initial and boundary conditions, and give an independent perturbation to each ensemble member to represent the uncertainty in the initial condition. Furthermore, we also use the same bathymetry and wind forcing as NorKyst-800. The

[1] We have ignored the marginal probability as it only serves as a normalization constant.

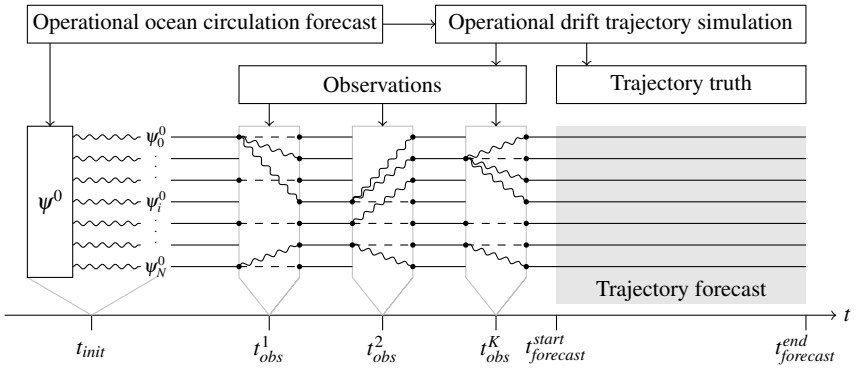

Fig. 1: Algorithmic overview of the simulation, data assimilation and drift trajectory forecast. Straight lines are deterministic simulation, wiggly lines are perturbations, and dashed lines show ensemble members that are kept during the resampling phase.

true drift trajectories are generated by OpenDrift [14] using the vertically integrated ocean currents from the hourly NorKyst-800 data, which means that the underlying physical model for the simulated truth is significantly different from our much simpler shallow-water model. From these drift trajectories, we estimate the underlying direction and velocity of the ocean currents and use this as an observation in the particle filter. We assume that the observations contain a Gaussian error with standard deviation σ and that they are independent from each other. The weight of each ensemble member is then computed as

$$w_i = \alpha \cdot \exp\left(-0.5\left(\frac{\|\text{obs} - \text{sim}\|}{\sigma}\right)^2\right), \tag{3}$$

in which we use the Euclidean norm to compute the distance between the observed and simulated momentum. Furthermore, α is the normalization constant such that $\sum_{i=1}^{N_e} w_i = 1$. There are several strategies for choosing which ensemble members to discard and duplicate (see [2] and references therein), and we use the residual resampling scheme [15]. Between observation times, each ensemble member runs independently and deterministically, which means that we need to perturb the duplicated ensemble states during the resampling stage.

3 Results

We test our ensemble prediction system using a domain along the coast of Northern Norway. The domain consists of 315×630 grid cells with 800 m horizontal resolution, which is the same horizontal resolution as the operational ocean circulation model that we use for initial and boundary conditions. We run 48 hours of data assimilation and use the final observed positions for the drifters as initial positions for

24 hour trajectory forecasts. To avoid that the ensemble collapses during resampling, we need to balance the number of drifters we observe with the ensemble size. Here we run experiments with 1000 and 10000 ensemble members and limit ourselves to four drifters. All experiments are run on an Nvidia DGX-2 server, equipped with 16 Tesla V100 GPUs and two CPUs, each with 24 cores.

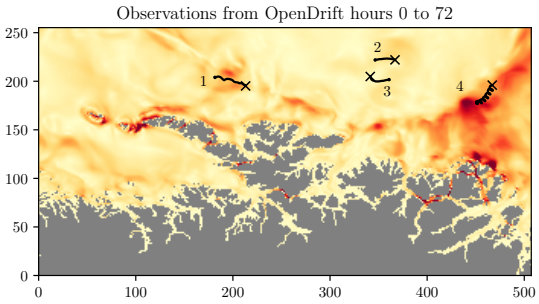

Fig. 2 Drift trajectories of four drifters over a three day period shown in the computational domain used in all our experiments. Red and yellow colors indicates strong and weak currents, respectively. Dots mark start positions and crosses end positions, and the values along the axes are in km.

Ensemble forecasts of drift trajectories We run three different experiments to illustrate the effect of data assimilation on forecasting of drift trajectories. The first is a Monte-Carlo experiment, meaning that we do not assimilate any observations during the first 48 hours. The second and third experiments are with data assimilation, and we use observations to run a particle filter every 30 minutes and 5 minutes, respectively. Fig. 2 shows the domain and the drift trajectories used as the truth.

Fig. 3 shows the forecasted drift trajectories for the four drifters in each of the three ensemble experiments with 1000 members, compared to a single deterministic forecast in green (dashed) and the truth in red (dash-dotted). The results show variable impact from data assimilation between the drifters. We see most positive effect for drifters three and four, and marginal improvement for drifter two, whereas the forecast for drifter one seems to be worse with data assimilation. The initial forecast for the first hour for drifter one, however, is significantly improved by the data assimilation, but our model is unable to capture the downward turn shortly into the forecast. This makes the ensemble perform worse in the long run when compared to the deterministic forecast. The results for drifters three and four show improvements when using observations in intervals of five minutes compared to 30 minutes. Finally, Fig. 4 shows how increasing the ensemble size to 10000 members significantly improves the forecast for drifter 2. Increasing the ensemble size increases the sampling of the pdf, and thereby also the chance that the true state is better represented by the ensemble.

Weak and strong performance scaling We evaluate the ensemble-level parallel performance by measuring the time spent in the data assimilation and forecasting parts of the code, running one hour of data assimilation with observations every five minutes and one hour of drift forecast. Note that the forecast contains no communication and should therefore show close to perfect scaling, whereas the data assimilation includes serial resampling and communication. For the weak scaling

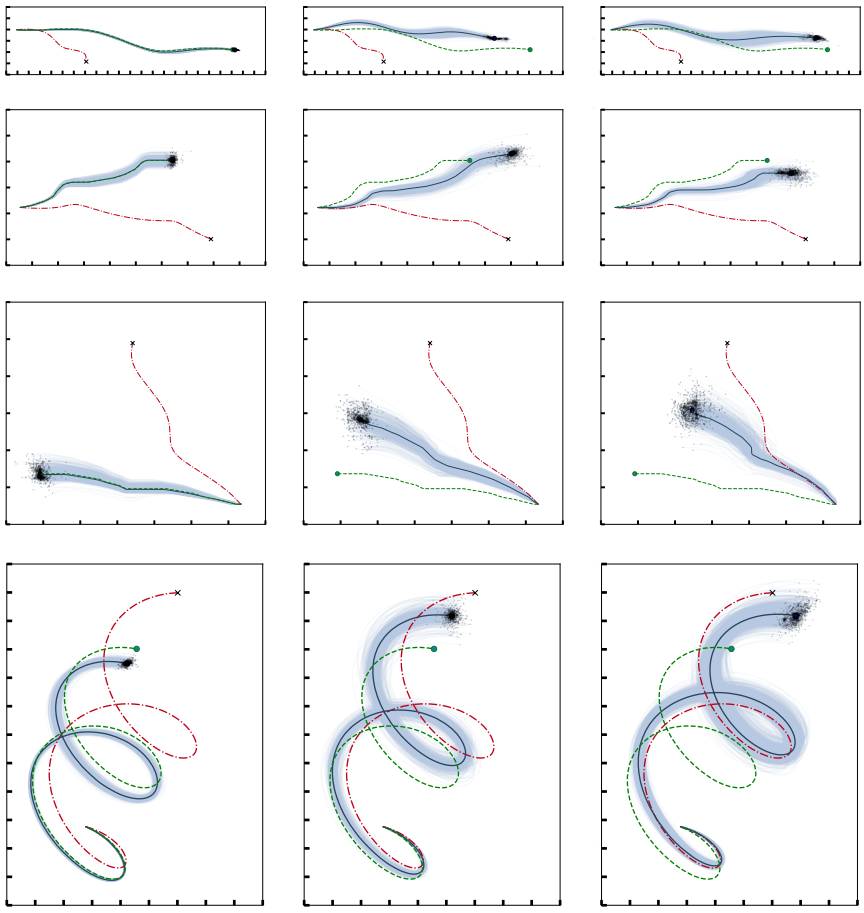

Fig. 3: Ensemble forecasts of drift trajectories with 1000 members in light blue, with the red (dash-dotted) line representing the truth, the green (dashed) line as the deterministic forecast, and the dark blue line as the ensemble mean. The four drifters are shown in separate rows, with the columns representing the three different experiments. From left to right: Monte Carlo without data assimilation, assimilation of observations every 30 minutes, and assimilation of observations every 5 minutes. The distance between the markers along the axis are one km.

experiment, we fix the per-process ensemble size at 20 members and increase the number of processes from one to 16. In the strong scaling experiment, the global ensemble size is fixed at 960 members and we vary the number of processes on which they are distributed. Each process utilizes one GPU. The results are shown in Fig. 5, with both experiments showing a 14x speedup by using 16 GPUs for the forecast. The speedup for data assimilation is nearly as good as the forecast, which means that the data assimilation does a good job preserving the parallel performance.

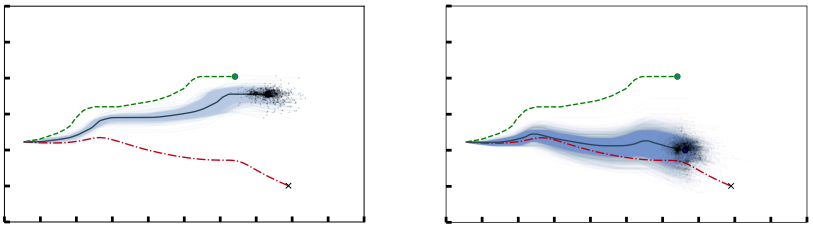

Fig. 4: Ensemble forcasting with 1000 ensemble members (left) and 10000 members (right). When using 10000 ensemble members, the forecast is significantly improved for drifter 2, while the effect is smaller for the other three drifters.

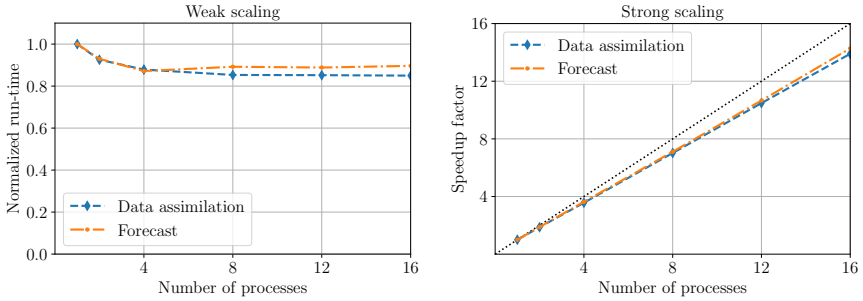

Fig. 5: The graphs show weak and strong scaling, using 1–16 GPUs on an Nvidia DGX-2 server. The top dotted line illustrates perfect strong scaling.

4 Discussion and summary

We have presented a framework for fully nonlinear data assimilation of ocean drift trajectories into a shallow-water model. The framework is implemented using the GPU for the shallow-water simulation and MPI for distribution of, and communication between, ensemble members. Our experiments show significantly increased forecast skill for simulations with data assimilation, and we are able to run over 10000 ensemble members on the Nvidia DGX-2 with 16 GPUs. The results indicate, as expected, that data assimilation with 10000 members based on 5 minute sampling of the observed drifter positions yields a better forecast than 5 minute sampling with 1000 members.

The results presented herein show that the data assimilation increases the forecast skill for three of four drifters. The forecast skill for drifter one, however, appears to be unaffected by the data assimilation, and we believe this is caused by a local predominant baroclinic ocean dynamic, which is not captured by our current simplified model. It will be an important future development to see what criteria are significant for the data assimilation to be most effective, and perhaps include a multi-layer or reduced-gravity model which can represent such dynamics better.

The simple perturbation strategy presented in this paper adds a smooth perturbation to the sea-surface level and computes the momentum required to keep the perturbation in geostrophic balance. An important extension will be to conduct more experiments with more sophisticated perturbation methods and stochastic placement of the ocean eddies, as well as perturbation of the tidal wave phase.

Acknowledgements This work is supported by the Research Council of Norway (RCN) through grant number 250935 (GPU Ocean). The computations in this paper were performed on equipment provided by the Experimental Infrastructure for Exploration of Exascale Computing (eX^3), which is financially supported by the RCN under contract 270053. The source code for the methods and experiments described in this paper is available under an GNU free and open source license released under the DOI 10.5281/zenodo.3591850.

References

1. K. Christensen, Ø. Breivik, K.-F. Dagestad, J. Röhrs, and B. Ward. Short-term predictions of oceanic drift. Oceanography, 31(3):59–67, 2018.
2. P. van Leeuwen. Particle filtering in geophysical systems. Monthly Weather Review, 137(12):4089–4114, 2009.
3. A. Brodtkorb and H. Holm. Real-world oceanographic simulations on the GPU using a two-dimensional finite volume scheme. preprint: arXiv:1912.02457, [in review], 2019.
4. NVIDIA. DGX-2/2H System user guide. Technical report, 2019.
5. S. Vetra-Carvalho, P. van Leeuwen, L. Nerger, A. Barth, M. Altaf, P. Brasseur, P. Kirchgessner, and J.-M. Beckers. State-of-the-art stochastic data assimilation methods for high-dimensional non-Gaussian problems. Tellus A: Dynamic Meteorology and Oceanography, 70(1):1–43, 2018.
6. G. Evensen. Data Assimilation: The Ensemble Kalman Filter. Springer Berlin Heidelberg, 2006.
7. C. Snyder, T. Bengtsson, P. Bickel, and J. Anderson. Obstacles to high-dimensional particle filtering. Monthly Weather Review, 136(12):4629–4640, 2008.
8. A. Shchepetkin and J. McWilliams. The regional oceanic modeling system (ROMS): A split-explicit, free-surface, topography-following-coordinate oceanic model. Ocean Modelling, 9(4):347–404, 2005.
9. E. Martinsen, B. Gjevik, and L. Røed. A numerical model for long barotropic waves and storm surges along the western coast of Norway. Journal of Physical Oceanography, 9:1126–1138, 1979.
10. L. Røed. Documentation of simple ocean models for use in ensemble predictions. Part I: Theory. Technical report, Norwegian Meteorological Institute, 2012.
11. A. Chertock, M. Dudzinski, A. Kurganov, and M. Lukácová-Medvidová. Well-balanced schemes for the shallow water equations with Coriolis forces. Numerische Mathematik, 2017.
12. H. Holm, M. Sætra, and P. van Leeuwen. Massively parallel implicit equal-weights particle filter for ocean drift trajectory forecasting. preprint: arXiv:1910.01031, [in review], 2019.
13. J. Albretsen, A. Sperrevik, A. Staalstrøm, A. Sandvik, and F. Vikebø. NorKyst-800 report no. 1: User manual and technical descriptions. Technical Report 2, Institute of Marine Research, 2011.
14. K.-F. Dagestad, J. Röhrs, Ø. Breivik, and B. Ådlandsvik. OpenDrift v1.0: a generic framework for trajectory modelling. Geoscientific Model Development, 11(4):1405–1420, 2018.
15. J. Liu and R. Chen. Sequential Monte Carlo methods for dynamic systems. Journal of the American Statistical Association, 93:1032–1044, 1998.