https://doi.org/10.5194/gmd-2023-38
Preprint. Discussion started: 21 March 2023

Geoscientific Model Development
Discussions
Open Access

ParticleDA.jl v.1.0: A real-time data assimilation software platform

Daniel Giles[1,2], Matthew M. Graham[2], Mosè Giordano[2], Tuomas Koskela[2], Alexandros Beskos[1,3], and Serge Guillas[1,2,3]

[1]Department of Statistical Sciences, University College London, London, UK
[2]Centre of Advanced Research Computing (ARC), University College London, London, UK
[3]The Alan Turing Institute, London, UK

Correspondence: Daniel Giles (d.giles@ucl.ac.uk)

Abstract. Digital twins of physical and human systems informed by real-time data, are becoming ubiquitous across weather forecasting, disaster preparedness, and urban planning, but researchers lack the tools to run these models effectively and efficiently, limiting progress. One of the current challenges is to assimilate observations in highly nonlinear dynamical systems, as the practical need is often to detect abrupt changes. We developed a software platform to improve the use of real-time data in highly nonlinear system representations where non-Gaussianity prevents the use of more standard Data Assimilation. Optimal Particle filtering data assimilation (DA) techniques have been implemented within an user-friendly open source software platform in Julia - ParticleDA.jl. To ensure the applicability of the developed platform in realistic scenarios, emphasis has been placed on numerical efficiency, scalability and optimisation for high performance computing frameworks. Furthermore, the platform has been developed to be forward model agnostic, ensuring that it is applicable to a wide range of modelling settings, for instance unstructured and non-uniform meshes in the spatial domain or even state spaces that are not spatially organised. Applications to tsunami and numerical weather prediction demonstrate the computational benefits in terms of lower errors, lower computational costs (due to ensemble size and the algorithm's overheads being minimised) and versatility thanks to flexible I/O in a high level language Julia.

1 Introduction

Data assimilation (DA) focuses on optimally combining observations with a dynamical model of a physical system to estimate how the system state evolves over time. The field of research has its origins within the *numerical weather prediction* (NWP) community, where DA techniques are applied iteratively to update current best estimates of the state of the atmosphere. Recently the methods and practices developed have been employed in diverse areas of geosciences, with Carrassi et al. (2018); Vetra-Carvalho et al. (2018) providing recent overviews. Further DA has seen a huge expansion into other scientific disciplines with applications in, for example, robotics (Berquin and Zell, 2022), economic modelling (Nadler et al., 2019) and plasma physics (Sanpei et al., 2021). In the era of digital twinning, which involves combining high-fidelity representations of reality with the optimal use of observations, real time data has become vital and DA frameworks have naturally been incorporated. The area of data learning has also emerged where DA approaches are integrated with machine learning techniques (Buizza et al., 2022).

There are various popular data assimilation techniques, with variational methods (3DVar and 4DVar) and *ensemble Kalman filter*s (EnKFs) (Evensen, 1994; Burgers et al., 1998) being extensively used in operational and research settings. However, these methods have difficulties with handling nonlinear problems and with representing uncertainties accurately (Lei et al., 2010; Bocquet et al., 2010). For instance Miyoshi et al. (2014); Kondo and Miyoshi (2019) updated an ensemble of 10 240 particles using the EnKF to demonstrate the bimodality of some distributions due to inherent nonlinearities. Furthermore, the continuing growth in compute hardware performance has allowed running increasingly complex and high resolution models which are able to resolve non-linear processes happening at a fine spatial scale (Vetra-Carvalho et al., 2018), creating an increasing demand for data assimilation methods which are able to accurately quantify uncertainty in such settings. *Particle filter*s (PFs) (Gordon et al., 1993) are an alternative approach which offer the promise of consistent DA for problems with non-linear dynamics and non-Gaussian noise distributions. Traditionally the main difficulty with particle filtering techniques has been the 'curse of dimensionality' (Bengtsson et al., 2008; Bickel et al., 2008; Snyder, 2011), where in high dimensional settings filtering leads to degeneracy of the importance weights associated with each particle and loss of diversity within an ensemble. To improve the applicability of PFs there have been many recent developments involving; localisation techniques (e.g., the reviews in Farchi and Bocquet (2018); Graham and Thiery (2019)), incorporation of tempering/mutation steps (e.g., Cotter et al. (2020); Ruzayqat et al. (2022)), hybrid approaches, improved computational implementations and combination of the above with improved proposal distributions. The above ongoing efforts have extended the applicability of PF methods within geoscientific domains. Leeuwen et al. (2019) provides an overview on the integration of particle filters in high-dimensional geoscience applications.

This paper presents PF algorithms that will often make use of the so-called 'optimal' proposal distribution. Though the latter is not amongst the latest contributions in the PF literature (e.g., Doucet et al. (2000)), it has not been extensively explored in the high-dimensional PDE-driven systems which we use as case studies in this work. As also noted in Snyder (2011), improved proposals used within PF can in practice dramatically effect the performance of the algorithm, thus providing working algorithms for wide classes of high-dimensional filtering applications, even if from a theoretical viewpoint computational costs for standard PF implementations (not incorporating some of the extra tools mentioned above, e.g. localisation) might still be required to scale exponentially fast with the number of observations.

The data assimilation paradigm of optimally combining observations and model has a wide range of applications. However, an existing hurdle impeding the incorporation of real-time data into pre-existing dynamical models is the lack of readily available software packages capable of bridging the two sources of information: model and observations. This is the motivation behind ParticleDA.jl, to provide a generic and user friendly framework to enable the incorporation of particle filtering techniques with pre-existing numerical models. ParticleDA.jl is an open-source package in Julia which provides efficient implementations of several particle filter algorithms, and also importantly offers an extensible framework to allow the simple addition of new filter implementations. It has been developed to be agnostic to the forward model to ensure applicability in a wide range of settings and emphasis has been placed on computational efficiency to be ready to enable, in a follow-up version, real-time applications. Initial efforts have been focused on the integration with spatially dependent numerical models, however

https://doi.org/10.5194/gmd-2023-38
Preprint. Discussion started: 21 March 2023

Package name	*Algorithms*	*Parallelism*
DataAssim.jl (Barth et al., 2016)	EnKF, 4DVar	
EnKF.jl (Le Provost, 2016)	EnKF	
Kalman.jl (Schauer et al., 2018)	KF	
KalmanFilters.jl (Schoenbrod, 2018)	KF	
LowLevelParticleFilters.jl (Carlson et al., 2018)	PF, KF	Multi-threading
ParticleFilters.jl (Sunberg et al., 2017)	PF	
SequentialMonteCarlo.jl (Lee and Piibeleht, 2017)	PF (SMC)	Multi-threading
ParticleDA.jl	PF	Multi-threading and multi-node (MPI)

Table 1. Summary of algorithms implemented and parallelism support in existing Julia data assimilation packages.

the implementation is applicable to a much more general class of state space models (see Section 2) allowing incorporation in a broad range of applications.

Within the Julia ecosystem, there are several existing packages which implement data assimilation algorithms. DataAssim.jl (Barth et al., 2016) provides implementations of a range of EnKF and extended *Kalman filter* (KF) methods and an incremental variant of 4DVar. EnKF.jl (Le Provost, 2016) implements stochastic and deterministic (square-root) variants of the EnKF which can be combined with various approaches (e.g. covariance inflation) to avoid ensemble collapse in models with deterministic dynamics. EnsembleKalmanProcesses.jl (Dunbar et al., 2022) implements several derivative-free optimization algorithms based on the EnKF mainly targetted at Bayesian inverse problem settings. Kalman.jl (Schauer et al., 2018) and KalmanFilters.jl (Schoenbrod, 2018) both provide implementations of the exact KF algorithm for linear Gaussian models, with KalmanFilters.jl additionally implementing unscented variants of the KF for use in models with non-linear dynamics or observation operators. ParticleFilters.jl (Sunberg et al., 2017) and LowLevelParticleFilters.jl (Carlson et al., 2018) both provide PF implementations, with LowLevelParticleFilters.jl additionally providing KF implementations. SequentialMonteCarlo.jl (Lee and Piibeleht, 2017) provides an interface for implementing (and example implementations of) the wider class of *sequential Monte Carlo* (SMC) methods, of which PFs can be considered a special case, with the ability to run particle ensembles in parallel on multiple threads. Table 1 summarizes the algorithm and parallelism support of existing Julia data assimilation packages.

We implemented ParticleDA.jl in the Julia programming language (Bezanson et al., 2017) because of its combination of performance and productivity, which enables rapid prototyping and development of high-performance numerical applications (Churavy et al., 2022; Giordano et al., 2022), with the possibility of using both shared and distributed memory parallelism strategies. In particular, Julia makes use of the multiple-dispatch programming paradigm, which is particularly well-suited for designing a program which combines different models with different filtering algorithms, keeping the two concerns separated. This allowed domain experts and software engineers to collaborate on the code using the same high-level language.

The rest of this manuscript is organized as follows. The mathematical set-up of the particle filtering algorithm is defined in section 2 along with the various filtering proposal distributions implemented. Section 3 outlines the code structure and parallelisation schemes. Sections 4 and 5 illustrate applications of the framework to simple low-dimensional state space models, namely a stochastically driven damped simple harmonic oscillator model and a stochastic variant of the Lorenz '63 chaotic attractor model. Section 6 introduces an application to a spatially extended state space model, specifically a tsunami modelling test case formulated as a linear Gaussian state space model, with validation of the filtering approaches and highlights the scaling performance. In section 7 the incorporation with a more complex non-linear atmospheric dynamical model is investigated along with some results. Finally in section 8 concluding remarks and future work are outlined.

2 Particle filtering

Let $\boldsymbol{x}_t \in \mathbb{R}^{d_x}$ represent the state of the model at an integer time index t and $\boldsymbol{y}_t \in \mathbb{R}^{d_y}$ the vector of observations of the system at this time index. We assume a state space model formulation, with the states following a Markov process and the observations depending only on the state at the corresponding time index, that is

$$\boldsymbol{x}_0 \sim p_0(\cdot); \quad \boldsymbol{x}_t \sim p_t(\cdot\,|\,\boldsymbol{x}_{t-1}), \quad \boldsymbol{y}_t \sim g_t(\cdot\,|\,\boldsymbol{x}_t), \quad t \geq 1; \tag{1}$$

where $p_0 : \mathbb{R}^{d_x} \to \mathbb{R}_{\geq 0}$ is the density of the initial state distribution, $p_t : \mathbb{R}^{d_x} \times \mathbb{R}^{d_x} \to \mathbb{R}_{\geq 0},\ t \geq 1$ are the densities of the state transition distributions and $g_t : \mathbb{R}^{d_y} \times \mathbb{R}^{d_x} \to \mathbb{R}_{\geq 0},\ t \geq 1$ are the densities of the conditional distribution on the observations given the current states.

For the most part, we will concentrate on a specialisation of this general state space model class, whereby the state and observations are both subject to additive Gaussian noise and the observations depend linearly on the state, which covers a wide range of modelling scenarios in practice. Concretely we consider a state update of the form

$$\boldsymbol{x}_t = F_t(\boldsymbol{x}_{t-1}) + \boldsymbol{u}_t, \quad \boldsymbol{u}_t \sim \mathcal{N}(0, Q), \quad t \geq 1, \tag{2}$$

where $F_t : \mathbb{R}^{d_x} \to \mathbb{R}^{d_x}$ is the forward operator at time index t, representing the deterministic dynamics of the system and $\boldsymbol{u}_t \in \mathbb{R}^{d_x}$ is the additive Gaussian state noise at time index t, representing stochastic aspects of the system dynamics. The observations are modelled as being generated according to

$$\boldsymbol{y}_t = H\boldsymbol{x}_t + \boldsymbol{v}_t, \quad \boldsymbol{v}_t \sim \mathcal{N}(0, R), \quad t \geq 1, \tag{3}$$

where $H \in R^{d_y \times d_x}$ is a linear observation operator and $\boldsymbol{v}_t \in \mathbb{R}^{d_y}$ is the additive Gaussian observation noise. The distributions of the state and observation noise are parameterized by positive-definite covariance matrices $Q \in \mathbb{R}^{d_y \times d_y}$ and $R \in \mathbb{R}^{d_x \times d_x}$ respectively.

The objective of the particle filter is to estimate the filtering distribution for each time index t which are the conditional probability distributions of the state $\boldsymbol{x}_t$ given observations $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t$ up to time index t, with the density of the filtering distribution at time index t denoted $\pi_t(\boldsymbol{x}_t|\boldsymbol{y}_{1:t})$.

The particle filtering algorithm builds on sequential importance sampling by introducing additional resampling steps. See Doucet et al. (2000) for an in-depth introduction but the key features are introduced in Algorithm 1. An ensemble of *particles* $\{\boldsymbol{x}_t^{(i)}\}_{i=1}^N$ represents an approximation to the filtering distribution at each time index $t \geq 1$ as $\pi_t(\mathrm{d}\boldsymbol{x}_t|\boldsymbol{y}_{1:t}) \approx \frac{1}{N}\sum_{i=1}^N \delta_{\boldsymbol{x}_t^{(i)}}(\mathrm{d}\boldsymbol{x}_t)$.

In each filtering step, new values for the particles are sampled from a proposal distribution (more details about the proposals implemented in ParticleDA.jl are given in section 2.1) and importance weights computed for each proposed particle value. At the end of the filtering step, the weighted proposed particle ensemble is resampled to produce a new uniformly weighted ensemble to use as the input to the next filtering step.

Algorithm 1 Particle filter

```
1: Initialise particles $\{\boldsymbol{x}_0^{(i)}\}_{i=1}^N$, with $\boldsymbol{x}_0^{(i)} \sim p_0(\cdot)$, for $1 \leq i \leq N$.
2: for time index $t = 1$ to $T$ do
3:     for particle index $i = 1$ to $N$ do
4:         Sample proposed particle $\tilde{\boldsymbol{x}}_t^{(i)} \sim q_t(\cdot \,|\, \boldsymbol{x}_{t-1}^{(i)}, \boldsymbol{y}_t)$.
5:         Compute (unnormalized) importance weight $w_t^{(i)} = W_t(\tilde{\boldsymbol{x}}_t^{(i)}, \boldsymbol{x}_{t-1}^{(i)}, \boldsymbol{y}_t) = \frac{p_t(\tilde{\boldsymbol{x}}_t^{(i)} \,|\, \boldsymbol{x}_{t-1}^{(i)}) g_t(\boldsymbol{y}_t \,|\, \tilde{\boldsymbol{x}}_t^{(i)})}{q_t(\tilde{\boldsymbol{x}}_t^{(i)} \,|\, \boldsymbol{x}_{t-1}^{(i)}, \boldsymbol{y}_t)}$.
6:     end for
7:     Generate new (equally weighted) particles $\{\boldsymbol{x}_t^{(i)}\}_{i=1}^N$ by resampling from the weighted empirical distribution
       $\boldsymbol{x}_t^{(i)} \sim \frac{\sum_{i=1}^N w_t^{(i)} \delta_{\tilde{\boldsymbol{x}}_t^{(i)}}(\cdot)}{\sum_{i=1}^N w_t^{(i)}}$
8: end for
```

2.1 Proposal distributions

Two forms of proposal distributions are implemented in ParticleDA.jl: the ‘naive’ bootstrap proposal, applicable to general state space models described by (1), and the ‘locally optimal’ proposal, applicable to the restricted class of state space models described by (2) and (3).

The bootstrap proposal ignores the observations with the particle proposals sampled from the state transition distributions,

$$q_t(\boldsymbol{x}_t \,|\, \boldsymbol{x}_{t-1}, \boldsymbol{y}_t) = p_t(\boldsymbol{x}_t \,|\, \boldsymbol{x}_{t-1}), \tag{4}$$

with the unnormalized importance weights at time index $t \geq 1$ then simplifying to

$$W_t(\boldsymbol{x}_t, \boldsymbol{x}_{t-1}, \boldsymbol{y}_t) = g_t(\boldsymbol{y}_t \,|\, \boldsymbol{x}_t) \;(= W_t(\boldsymbol{x}_t, \boldsymbol{y}_t)). \tag{5}$$

While appealingly simple and applicable to a wide class of models, the bootstrap particle filter performs poorly when observations are informative about the state due to the observations being ignored in the proposal. In such cases the proposed particles will typically be far away from the mass of the true filtering distribution, with the importance weights in this setting tending to have high variance leading to *weight degeneracy* whereby all the normalized importance weights but one are close to zero.

To alleviate the tendency to weight degeneracy we can use alternative proposal distributions which decrease the variance of the importance weights. For proposals distributions $q_t(\boldsymbol{x}_t \,|\, \boldsymbol{x}_{t-1}, \boldsymbol{y}_t)$ which condition only on the previous state $\boldsymbol{x}_{t-1}$ and current observation $\boldsymbol{y}_t$, the optimal, in the sense of minimising the variance of the importance weights, proposal can be shown

(Doucet et al., 2000) to be

$$q_t(\boldsymbol{x}_t\,|\,\boldsymbol{x}_{t-1},\boldsymbol{y}_t) = \frac{p_t(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})g_t(\boldsymbol{y}_t\,|\,\boldsymbol{x}_t)}{\int p_t(\tilde{\boldsymbol{x}}_t|\boldsymbol{x}_{t-1})g_t(\boldsymbol{y}_t\,|\,\tilde{\boldsymbol{x}}_t)\mathrm{d}\tilde{\boldsymbol{x}}_t}, \tag{6}$$

with corresponding unnormalized importance weights

$$W_t(\boldsymbol{x}_t,\boldsymbol{x}_{t-1},\boldsymbol{y}_t) = \int p_t(\tilde{\boldsymbol{x}}_t|\boldsymbol{x}_{t-1})g_t(\boldsymbol{y}_t\,|\,\tilde{\boldsymbol{x}}_t)\mathrm{d}\tilde{\boldsymbol{x}}_t \;(= W_t(\boldsymbol{x}_{t-1},\boldsymbol{y}_t)). \tag{7}$$

Note that in this case the importance weights are independent of the sampled values of the particle proposals.

For general state space models, sampling from this *locally optimal proposal* and computing the importance weights can be infeasible due to the integral in (6) and (7) not having a closed form solution. However, for the specific case of a state space model of the form described by (2) and (3), the proposal distribution has the tractable form

$$q_t(\boldsymbol{x}_t\,|\,\boldsymbol{x}_{t-1},\boldsymbol{y}_t) = \mathcal{N}\left(\boldsymbol{x}_t\,|\,F_t(\boldsymbol{x}_{t-1}) + QH^\mathsf{T}(HQH^\mathsf{T}+R)^{-1}(\boldsymbol{y}_t - HF_t(\boldsymbol{x}_{t-1})), Q - QH^\mathsf{T}(HQH^\mathsf{T}+R)^{-1}HQ\right), \tag{8}$$

with corresponding importance weights

$$W_t(\boldsymbol{x}_t,\boldsymbol{x}_{t-1},\boldsymbol{y}_t) = \mathcal{N}\left(\boldsymbol{y}_t\,|\,HF_t(\boldsymbol{x}_{t-1}), HQH^\mathsf{T}+R\right). \tag{9}$$

2.2 Resampling

A resampling step similar to the one shown in Algorithm 1 is included in all implementations of PF methodology. Resampling multiplies particles found at good positions in space that agree with observations and removes unwanted particles. It is key for establishing analytical theory showing that Monte-Carlo errors in estimates of expectations under the filtering distribution are controlled *uniformly* in time, see, e.g., the standard reference Del Moral (2004). Such a result provides a critical justification for the powerful performance of PF-based algorithms in many applications.

In this paper, we make use of standard multinomial resampling, though more sophisticated options that lead to PFs of improved performance are available (Douc and Cappé, 2005). In what follows, we will often show the value of the Effective Sample Size (ESS), capturing the variability of the weights before resampling. ESS is defined as follows:

$$\mathrm{ESS}_t = \frac{\{\sum_{i=1}^N w_t^{(i)}\}^2}{\sum_{i=1}^N \{w_t^{(i)}\}^2}.$$

ESS is used as a proxy for the number of iid samples that would produce estimates of similar precision as the ones obtained by the available (correlated) particles.

3 Code Structure

As stated ParticleDA.jl is designed to be forward model agnostic (*i.e.* capable of running with arbitrary state space forward models). To enable this support the model and filter portions of ParticleDA.jl are carefully delineated. The main structure of the code is given as follows in Algorithm 2, where T is the total number of filtering steps. The steps marked with an asterisk $(*)$ are related to the parallelisation framework and more details will be given in section 3.3. Further details on the filter and model components are given as follows.

Algorithm 2 Code Structure of ParticleDA.jl

```
Initialize particles and model data which is defined by the user
Initialize filter data which is dependent on the choice of filter type
for time index $t = 1$ to $T$ do
    Update the observations of the truth
    Update the particle dynamics
    Update particle proposals
    Get particle observations
    Calculate particle weights
    Gather particle weights* and re-sample
    Broadcast new sampling indices* and copy particle states
    calculate population statistics*
end for
```

3.1 Filter

The default parameters for the filter can be altered by passing values within a configurable YAML file. Parameters to be set include the number of particles, number of assimilation time steps, I/O options and output file names. When running the particle filter the key setting is the choice of filtering proposal (Bootstrap or Optimal).

3.2 Model

By construction the model set up will be defined by the user. The current implementation provides a template for model integration. However, one of the key commonalities across forward model integrations will be the definition of model noise. At present the model noise is generated from realisations of Gaussian Random Fields, ParticleDA.jl makes use of the GaussianRandomFileds.jl package for this. The user is required to define the choice of covariance kernel and the associated parameters. For a Matérn covariance kernel this will include: the length scale (λ), the smoothness (μ) and the standard deviation (σ). The number of observations and the indices of the observed variables within a multiple dimensional state space model are needed. The locations of the observation stations can be passed within a simple .txt file. The observations can come from an online integration of the state space model or read in from file/sensor.

3.3 Parallelisation scheme

As each particle update and (unnormalized) weight calculation can be computed independently, many of the steps in Algorithm 2 can be performed in parallel. Both shared and distributed memory parallelisation approaches are leveraged within ParticleDA.jl. Particle and weight updates are parallelised across multiple threads on shared memory systems using the native @threads macro in Julia. In distributed memory environments ParticleDA.jl uses the *message passing interface* (MPI) library through the MPI.jl Julia library (Byrne et al., 2021), which has been found by Giordano et al. (2022) to have little to no overhead in applications with thousands of MPI ranks. Input and outputs (I/O) in ParticleDA.jl are supported through the HDF5 library and are carried out on the master rank.

The distributed parallelisation scheme is sketched out in Figure 1. The principle is simply to keep the large particle state vectors distributed to MPI ranks, only copying them point-to-point when required by particle duplication, and use quantities derived from the local particle ensemble in global communications. The parallel algorithm requires three global steps: a Gather of particle weights, a Scatter of particle indices (one integer per particle) and a Reduction of population statistics, highlighted by asterisks in Algorithm 2. The gather and scatter communicate one floating point and integer number per particle, respectively. The Reduction is performed on the state vectors, the output being the size of a single state vector per rank. The mean of states is a simple sum reduction, implemented in the MPI library.

We have implemented several reductions for the variance of states. By default, the variance is reduced as a custom reduction that only requires a single global communication step and computes the sum and sum of squares within the reduction. For a discussion on the implementation in MPI.jl, see Byrne et al. (2021). Custom reductions are currently not supported by MPI.jl on all CPU architectures,[1] therefore we also provide a variance reduction using native sum reductions and a single communication step. However, this algorithm can be numerically unstable for large ensembles or state components with large values. The reduction becomes the main bottleneck when scaling up, we discuss the performance impacts in Section 6.2.

4 Stochastically driven damped simple harmonic oscillator

As a tractable first test case we consider a two-dimensional state space model corresponding to the time discretisation of a stochastic differential equation

$$\mathrm{d}\boldsymbol{x}(t) = \begin{pmatrix} 0 & 1 \\ -\omega_0^2 & -\omega_0/Q \end{pmatrix} \boldsymbol{x}(t)\,\mathrm{d}t + \begin{pmatrix} 0 \\ 1 \end{pmatrix} \mathrm{d}W(t),$$

representing a damped simple harmonic oscillator driven by a Wiener noise process $W(t)$. This process has been proposed as a model for astronomical time series data (Foreman-Mackey et al., 2017), with details of its formulation as a state space model given in Jordán et al. (2021). Importantly, the state space model is linear-Gaussian and so we can use a Kalman filter to exactly compute the true Gaussian filtering distributions.

We use an instance of the model with parameters $\omega_0 = 1$, $Q = 2.$ and time discretisation step $\delta t = 0.2$. We assume an observation model $\boldsymbol{y}_t \sim \mathcal{N}(\boldsymbol{x}_t, \sigma^2 I)$ with $\sigma = 0.5$ and simulate observations from the model for $T = 200$ time steps with initial state distribution $\boldsymbol{x}_0 \sim \mathcal{N}(\boldsymbol{0}, I)$. Figure 2 shows the *root mean squared error*s (RMSEs) in particle filter estimates of the means and log-variances of the Gaussian filtering distributions (compared to ground truth values computed using a Kalman filter), as a function of the number of particles used in the ensemble, for filters using both the bootstrap and locally optimal proposal. We see that the locally optimal proposal gives a small but consistent improvement in RMSE for a given ensemble size, reflecting the lower variance in the empirical estimates to the filtering distributions. As expected the errors in the filter

[1]https://github.com/JuliaParallel/MPI.jl/issues/404.

https://doi.org/10.5194/gmd-2023-38
Preprint. Discussion started: 21 March 2023

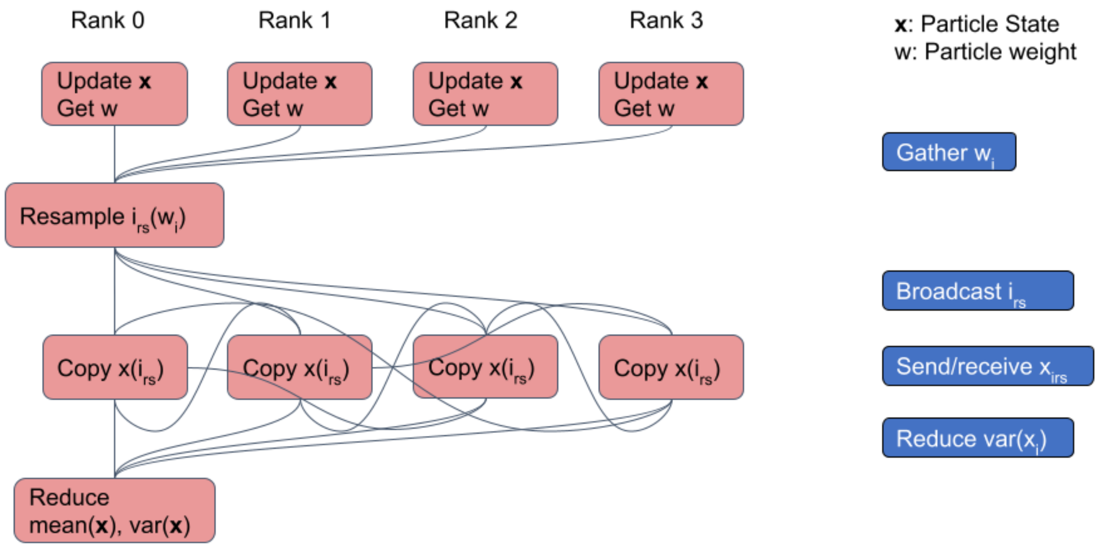

Figure 1. A sketch of the parallelisation framework in ParticleDA.jl. Serial operations are represented by red boxes, and the communication operations by blue boxes. Each rank updates particle states and weights independently. The communication between parallel processes requires three global communication steps and a point-to-point communication step. The impact on performance is discussed in Section 6.2.

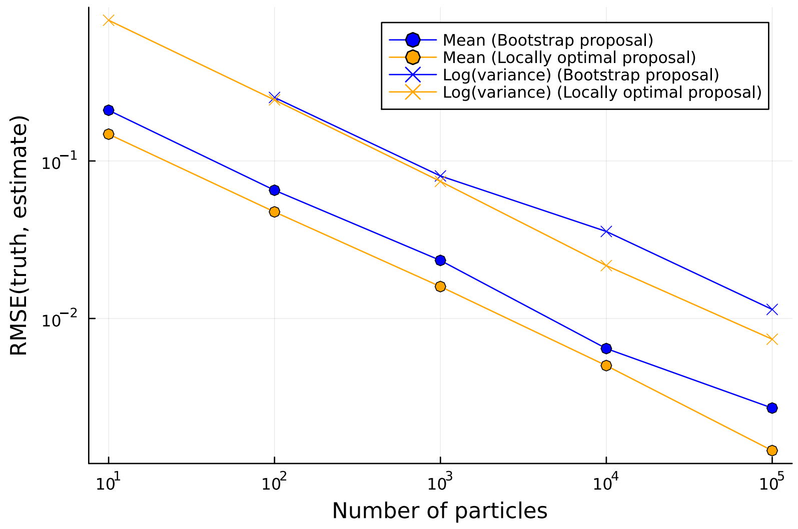

Figure 2. RMSE in particle filter estimates of filtering distribution means and (log) variances against number of particles for the damped simple harmonic oscillator model. The RMSEs compute the mean of the squared errors across all state components and time steps.

https://doi.org/10.5194/gmd-2023-38
Preprint. Discussion started: 21 March 2023

estimates appear to be asymptotically tending to zero at a polynomial rate in the ensemble size, providing some assurance of the correctness of the ParticleDA.jl filter implementations.

5 Lorenz '63

The Lorenz '63 model was introduced by Lorenz (1963) and is a nonlinear dynamical model which has been used to test data assimilation methods. The system consists of three nonlinear differential equations which capture a simplified representation of thermal convection. The differential equations are as follows:

$$\begin{aligned}\frac{dx_1}{dt} &= \sigma(x_2 - x_1),\\ \frac{dx_2}{dt} &= \rho x_1 - x_2 - x_1 x_3,\\ \frac{dx_3}{dt} &= x_1 x_2 - \beta x_3,\end{aligned} \tag{10}$$

where $x_1(t), x_2(t)$ and $x_3(t)$ are the prognostic variables of the model and σ, ρ and β are the free parameters. As outlined by Lorenz (1963) we have set the free parameters to $\sigma = 10$, $\rho = 28$ and $\beta = \frac{8}{3}$ as this set up will lead to chaotic behaviour.

The system is simulated for $T = 500$ time steps and with a time step size of 0.1 s. The state variables are assimilated at each time step. The observation noise standard deviation is set to 0.1 and the state noise standard deviation is 0.5. The initial conditions of both the observations run and particles are drawn from an initial state distribution $\boldsymbol{x}_0 \sim \mathcal{N}(\mathbf{0}, \sigma^2 I)$ with $\sigma = 0.5$. Fig. 3 showcases the performance of an ensemble of $N = 20$ particles using the locally optimal proposal. The left subplot of Fig. 3 shows the observations and the mean of the particles at each of the time steps, note the appearance of the Lorenz attractor. The right subplot of Fig. 3 is the estimated *effective sample size* (ESS) at each time step of the simulation, where the ESS at time index t is estimated as $\left(\sum_{i=1}^{N} w_t^{(i)}\right)^2 / \sum_{i=1}^{N} \left(w_t^{(i)}\right)^2$ with $\{w_t^{(i)}\}_{i=1}^{N}$ the *unnormalized* particle importance weights. The estimated ESS remains close to the number of particles ($N = 20$) for the duration of the simulation therefore particle degeneracy has been avoided. The distributions of the particle states were observed to showcase some non-Gaussian characteristics with absolute skewness values of 1.1.

6 Tsunami Modelling

One of the built-in test cases in ParticleDA.jl focuses on tsunami modelling. Tsunamis are rare events which have the capacity of causing severe loss of life and damages. At present, tsunami warning centres rely on crude decision matrices, pre-computed databases of high resolution simulations or 'on-the-fly' real time simulations to rapidly deduce the hazard associated with an event (Gailler et al., 2013). These existing approaches have been developed with seismically generated tsunamis in mind and the alternative tsunamigenic sources (landslide and volcanic eruptions) are less well constrained. The ongoing efforts of incorporating data assimilation techniques within tsunami modelling could augment a warning centres capability in this regard (Maeda et al., 2015; Gusman et al., 2016). It should be noted that the tsunami model built into ParticleDA.jl is a drastic

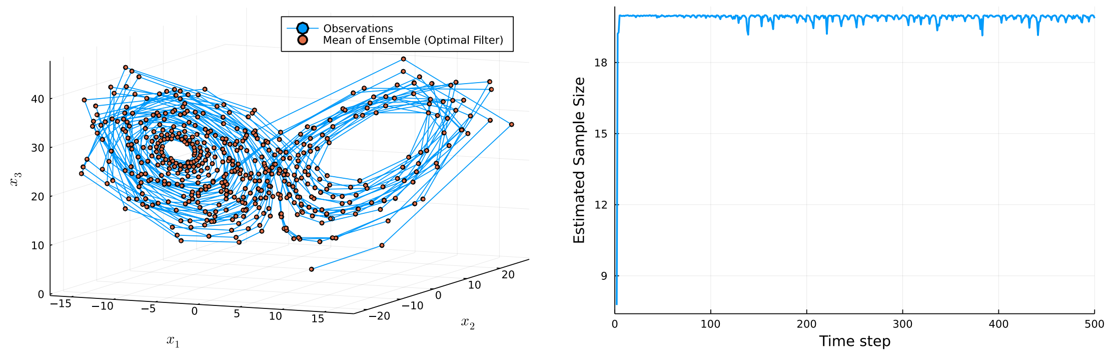

Figure 3. Left: Mean of the particles and the observations at each time step in state space. Right: The estimated ESS at each time step.

simplification to industry used tsunami models but provides a useful test case for users and showcases the potential of particle filters within tsunami modelling efforts. Further, it allows for direct validation with a Kalman filter as the model is linear and Gaussian.

To a first order approximation the linear shallow water equations are used to capture the tsunami dynamics. The forward model $F_t(\boldsymbol{x}_t)$ for the test case is therefore an inbuilt solver of the linear shallow water equations:

$$\frac{\partial \eta}{\partial t} + \frac{\partial (hu)}{\partial s_1} + \frac{\partial (hv)}{\partial s_2} = -\frac{\partial h}{\partial t},$$
$$\frac{\partial (hu)}{\partial t} = -gh\frac{\partial \eta}{\partial s_1},$$
$$\frac{\partial (hv)}{\partial t} = -gh\frac{\partial \eta}{\partial s_2}, \tag{11}$$

where $\eta(s_1,s_2)$ is the free surface elevation (wave height), $h(s_1,s_2)$ is the water depth, g is the acceleration due to gravity and $\boldsymbol{u} = (u,v)$ are the depth averaged horizontal velocities. Eqs. 11 are solved using a first order finite difference scheme which is based on TDAC (Maeda et al., 2015).

6.1 Validation

The experimental set-up consists of a square domain $\Omega = [0, 200 \text{ km}] \times [0, 200 \text{ km}]$ with 15 observation locations (Fig. 4) and a uniform grid of 51×51 in the spatial domain. The initial condition in the observations run is a Gaussian shaped wave centred on $(1 \text{ km}, 1 \text{ km})$. The observation run is integrated forward in time for 1280s with the observations at the gauge locations extracted and stored. The observation error standard deviation for both the observation run and particles is set to 0.1. The particles are initialised with a randomly perturbed free surface elevation and velocities (realisations of a Gaussian random field with zero mean and standard deviation of 0.01). The particle states are integrated for 1280s, with the assimilation of the surface elevation occurring every $dt = 2$s. Figure 4 showcases snapshots of the experiment with the locally optimal proposal and 50 particles ($N = 50$). The surface elevation coming from the observations run is plotted on the left while the mean of the particles is on the right.

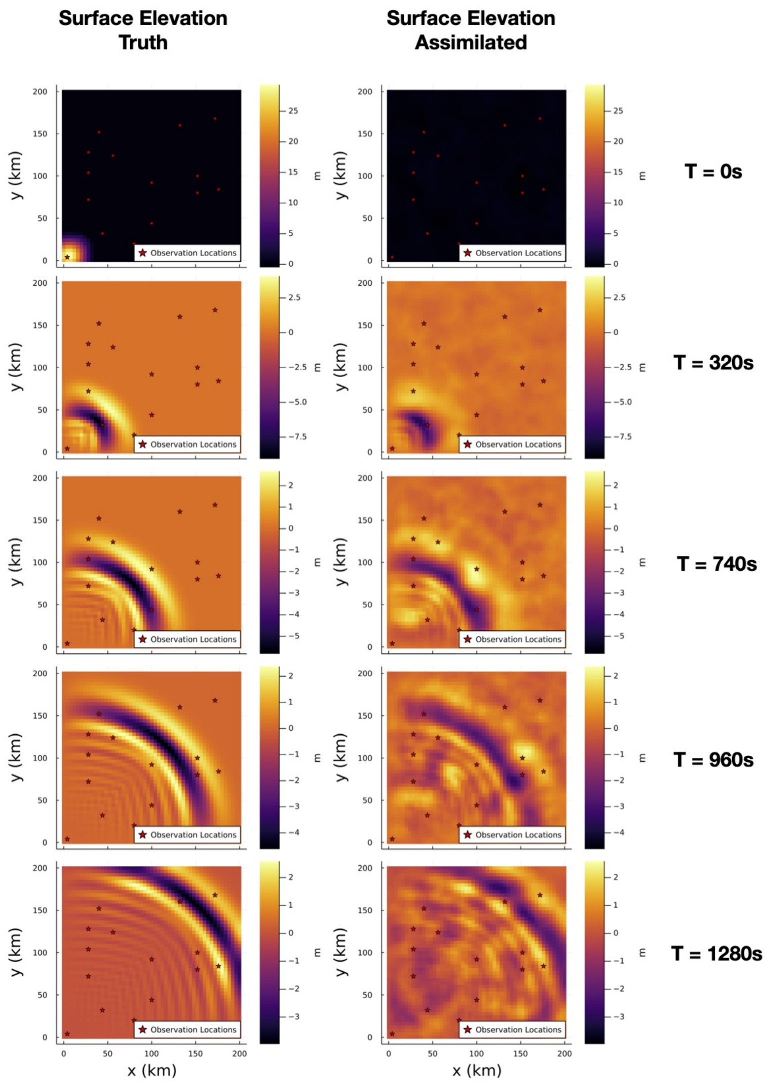

Figure 4. Snapshots of the surface elevation as simulated by the observations run (left) and the mean of the particles ($N = 50$) (right).

As the tsunami model implemented here is linear and Gaussian one can compare the obtained distributions with a ground truth Kalman filter. Fig. 5 (left) highlights the RMSE error through time of both the locally optimal and bootstrap proposals with the same number of particles ($N = 50$), it can be clearly seen that the locally optimal proposal performs better through time. Fig. 5 (right) showcases the error at 200s with increasing number of particles for both proposals. The asymptotic behaviour of the two approaches again highlights the benefits of the locally optimal over the bootstrap proposal.

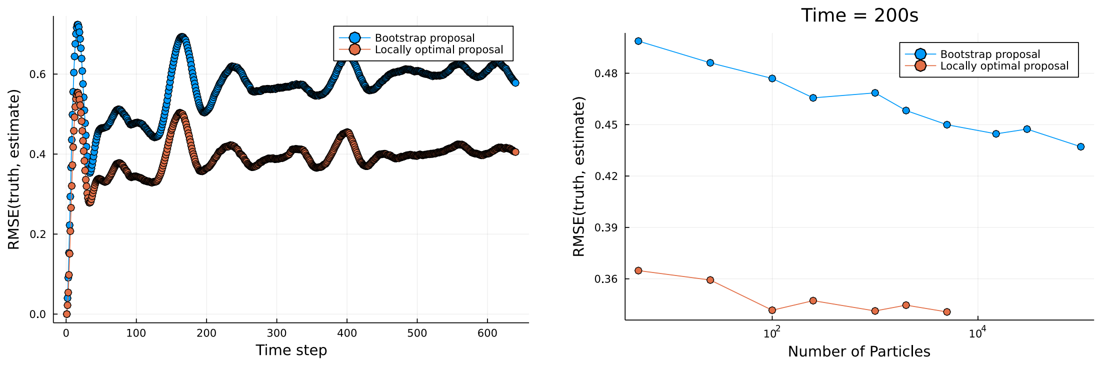

Figure 5. Left: RMSE (50 particles) for both the bootstrap and locally optimal proposal at each time step, tsunami modelling. Right: RMSE for both the bootstrap and locally optimal proposal at t = 200s for increasing number of particles.

6.2 Parallelisation Performance

As discussed in section 3.3 ParticleDA.jl is capable of leveraging both shared and distributed parallelism. Scaling runs on ARCHER2, which is the UK's Tier-1 supercomputer, have been carried out to highlight the performance in practice. A weak scaling study, using the same experimental set up as described in section 6.1 is run with the Bootstrap proposal keeping the number of particles per compute node constant while increasing the number of nodes. The compute nodes on ARCHER2 consist of 2 $\times$ AMD EPYC 7742, 2.25 GHz, 64-core, with 8 NUMA regions per node (16 cores per NUMA region, 8 cores per core complex die (CCD) and 4 cores per core complex (CCX) (shared L3 cache)). The weak scaling runs try to optimise for this hardware architecture with various runs targeting a MPI rank per NUMA/CCD/CCX region and an appropriate number of threads per MPI rank (Fig. 6 left subplot). There are 2,048 particles per MPI rank so at the maximum number of ranks tested here (128) there are 262,144 particles. Based on these findings the set up of 32 ranks per node and 4 threads per rank (targeting the CCX) scales best for higher number of ranks.

To investigate the scalings further, the right subplot in Fig. 6 highlights the breakdown per function call for the 32 ranks per node and 4 threads per rank results. The proposals and weights, green line in Fig. 6 (right) accounts for the following steps in Algorithm 2: Update particle dynamics, Update Particle Proposals, Get particle observations and Calculate particle weights. As one would expect from the description of Algorithm 2, the proposals and weights component of the run time scales extremely well due to particles updating their proposals independently. The main bottleneck becomes the update of the global statistics over all particles, because this requires a reduction of the mean and optionally the variance at every grid point over all MPI ranks. We can partially remedy this loss of performance by collecting the global statistics less frequently, but the required frequency will depend on the scientific use case of the simulation. The gather of particle weights and resampling of particle indices is similarly a global communication step that deteriorates the scaling, but they have less impact on total performance because only one number per particle is communicated. After the indices have been broadcast in Resample, the particle states are copied via point-to-point communications, which also contributes to the loss in performance.

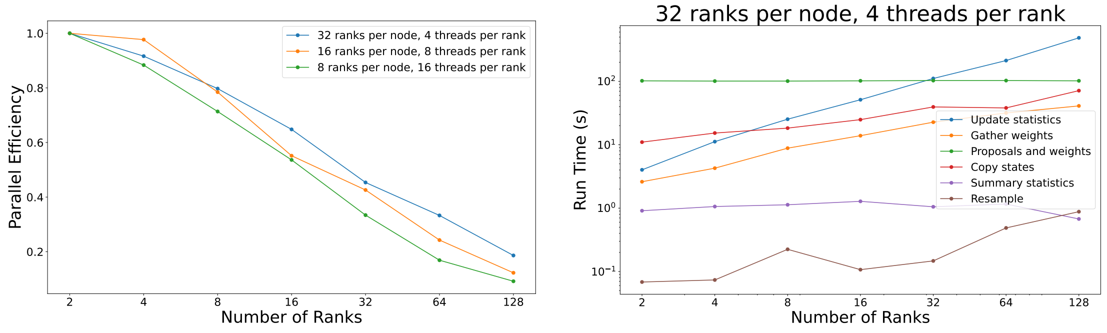

Figure 6. Left: Weak scaling parallel efficiency for different set ups of ranks per node and threads per rank. Right: A breakdown of run time spent in different function calls for the 32 ranks per node and 4 threads per rank set up. The functions shown here correspond to steps of Algorithm 2, with Proposals and Weights encompassing steps 1-5 of the time loop.

7 Atmospheric General Circulation Model (AGCM)

An integration of ParticleDA.jl with an atmospheric dynamical model *simplified parameterizations primitive equation dynamics* (SPEEDY) showcases the efforts involved in coupling the software with pre-existing model implementations. SPEEDY is an *atmospheric general circulation model* (AGCM) which was developed by Molteni (2003) and it consists of a spectral primitive-equation dynamic core along with a set of simplified physical parameterization schemes. The SPEEDY model retains the core characteristics of the current state-of-the-art AGCMs but requires drastically less (order of magnitude) computational resources (Molteni, 2003). This computational efficiency allows one to utilize the model to carry out large ensemble and/or data assimilation experiments. According to Molteni (2003) the SPEEDY model accurately simulates the general structure of global atmospheric circulation and exhibits similar systematic errors to the state-of-the-art AGCM, albeit with larger error amplitudes obtained. The model version (Hatfield, 2018) used here is written in Fortran and provides an interesting example of the integration steps required to interface with ParticleDA.jl, the coupling between the two relies on SPEEDY being set up to output its data fields at set intervals.

As stated SPEEDY is a simplified AGCM model. The prognostic variables consist of the zonal and meridional wind velocity components (u, v), temperature (T), specific humidity (q) and surface pressure (p). A T30 resolution of the model is used here which corresponds to a grid size of $96 \times 48 \times 8$. The vertical layers are defined by sigma levels, where the pressure is normalized by the surface pressure (p/p_s).

The atmospheric modelling undertaken here can be considered within the same framework as defined in Eq. 2 and Eq. 3 but with the dynamical operator/forward model (F_t) now defined to be the SPEEDY model. The SPEEDY model is set up to output its state vectors at 6 hour intervals. This I/O allows for the two codes to be coupled together. As the dynamics are simulated on the globe a bespoke covariance function dependent on the geodesic distance is used to generated the model and observation errors. The authors recognise that this approach has certain limitations but nevertheless for the purpose of showcasing the integration with ParticleDA.jl is sufficient.

https://doi.org/10.5194/gmd-2023-38
Preprint. Discussion started: 21 March 2023

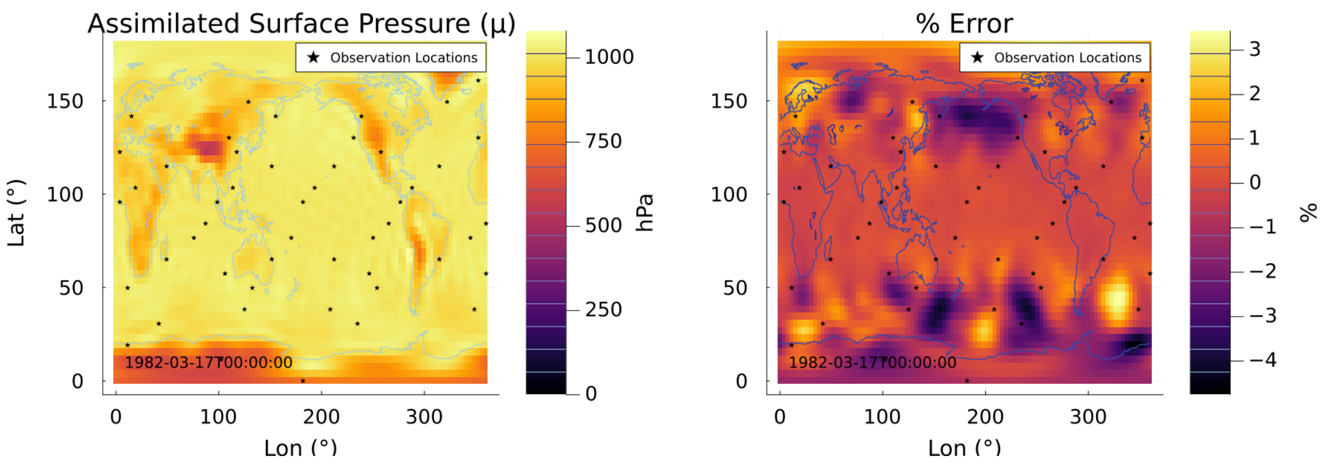

Figure 7. Snapshot of the mean assimilated surface pressure (left) and the corresponding percentage error when compared to the nature run (right) ($n = 256$). The 50 observation locations are highlighted by the black stars.

7.1 Results

The data assimilation experiments carried out here were introduced by Miyoshi (2005). An observation (nature) run is generated after an initial one year spin-up. The nature run is launched on the date of January 1, 1981 with the atmosphere at rest ($u = v = 0$). The data assimilation experiments start on January 1, 1982.

Differing to the experimental set-up introduced by Miyoshi (2005), only the surface pressure (p_s) at 50 observation locations (see Fig. 7) is assimilated. The observational data is obtained by adding random noise to the nature run at the observation locations. The observational errors are generated from independent Gaussian numbers and the observational error standard deviations is fixed to 10 hPa.The particles ($n = 256$) are initialised from randomly selected dates within the month of December from a long-term (10-year) nature run. The standard deviation of the model and observation errors are set to 1 and 10 hPa respectively. In Fig. 7 (left) a snapshot of the mean assimilated surface pressure is shown. The mean surface pressure is compared to the nature run and a percentage error is plotted in Fig. 7 (right). It can be seen that the areas of greatest percentage error coincide with areas that lack observation stations.

As stated in the introduction, one of the key benefits of particle filters is to provide the promise of non-linear and non-Gaussian DA. To highlight this sample distributions of the surface pressure at various observation locations at different time points are shown in Fig. 8. The distributions across the $n = 256$ particles exhibit heavy tails towards the true surface pressure at the given locations.

8 Conclusions

Our particle filter software in Julia shows versatility by being able to interact with any model structure with an inherent ability to represent and sample nonlinearities and thus non-Gaussianity. By being written in Julia it allows users to understand and

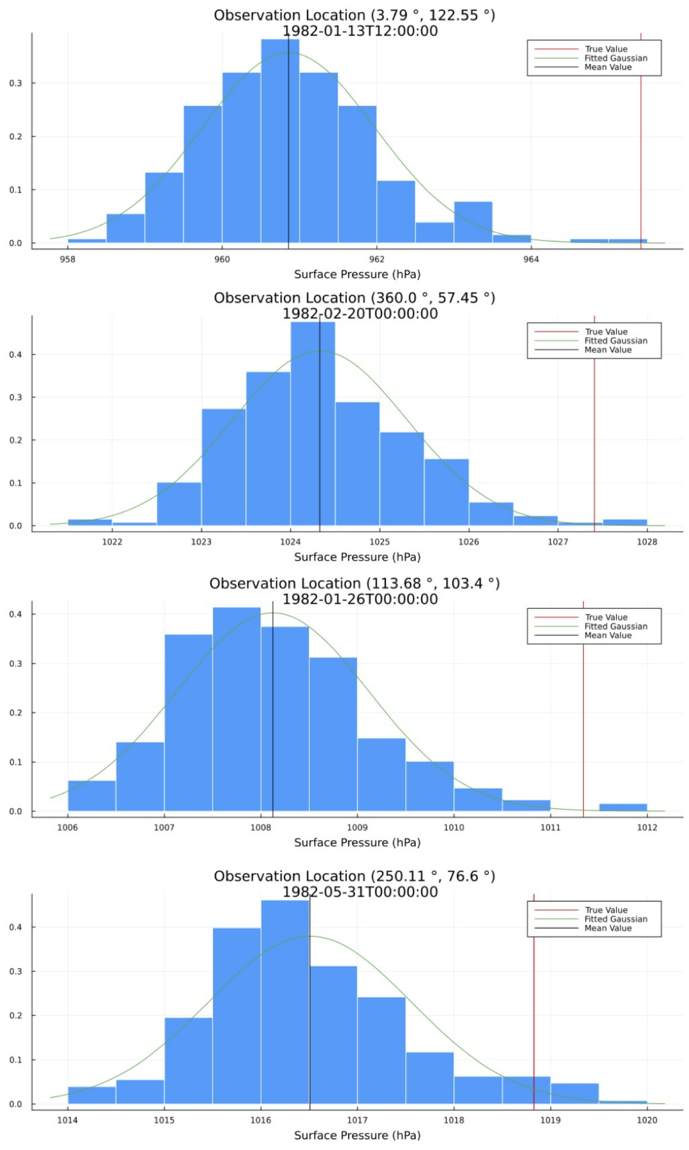

Figure 8. Normalised histograms of the surface pressure at various observation locations and at different points in time ($n = 256$). The true surface pressures are highlighted by the vertical red line and the mean surface pressure of the particles are highlighted by the black vertical line. The green line represents a fitted Gaussian distribution.

manipulate the code more easily than traditional packages in C++ or Fortran at similar speeds. The package offers both shared and distributed parallelism which are necessary when large ensembles are needed to combat particle degeneracy in some cases.

The various test case models shown here validate the implementation and highlight the strengths of ParticleDA. Errors remain small and the method is as computationally efficient as EnKF, but with the ability to tackle nonlinear processes better, thereby allowing much more accurate DA, especially.

Challenges remain to deploy this framework at scale in terms of model dimensions and speed, say towards exascale. In particular, the scalability of the method when the state space becomes large. Opportunities exists in terms of smart and efficient approaches to design and enrich the ensemble in these situations where state space dimension may create a difficult degeneracy issue.

Overall, the aim of our platform is to enable easily accessible and accurate, fast DA for a wide range of users. We hope that various scientific communities will adopt ParticleDA, possibly leading to fast step-changes in some geoscientific investigations and beyond.

Code availability. The code is freely available at https://github.com/Team-RADDISH/ParticleDA.jl

Author contributions. DG lead the computations and applications. TK, MMG and MG created the Julia platform, its I/O and computational acceleration. AB provided the optimal algorithm and its challenges. SG directed the overall research.

Competing interests. There are no competing interests at present

Acknowledgements. The RADDISH (Real-time Advanced Data assimilation for Digital Simulation of numerical twins on HPC) project supported this research. RADDISH was part of The Tools, Practice and Systems programme of the AI for Science and Government (ASG), UKRI's Strategic Priorities Fund awarded to the Alan Turing Institute, UK (EP/T001569/1). We also acknowledge funding for this research from UKAEA (T/AW085/21) for the project Advanced Quantification of Uncertainties In Fusion modelling at the Exascale with model order Reduction (AQUIFER).

References

Barth, A., Saba, E., Carlsson, K., and Kelman, T.: DataAssim.jl: Implementation of various ensemble Kalman Filter data assimilation methods in Julia, https://github.com/Alexander-Barth/DataAssim.jl, 2016.

Bengtsson, T., Bickel, P., and Li, B.: Curse-of-dimensionality revisited: Collapse of the particle filter in very large scale systems, in: Probability and statistics: Essays in honor of David A. Freedman, pp. 316–334, Institute of Mathematical Statistics, 2008.

Berquin, Y. and Zell, A.: A physics perspective on lidar data assimilation for mobile robots, Robotica, 40, 862–887, https://doi.org/10.1017/S0263574721000850, 2022.

Bezanson, J., Edelman, A., Karpinski, S., and Shah, V. B.: Julia: A Fresh Approach to Numerical Computing, SIAM Review, 59, 65–98, https://doi.org/10.1137/141000671, 2017.

Bickel, P., Li, B., and Bengtsson, T.: Sharp failure rates for the bootstrap particle filter in high dimensions, in: Pushing the limits of contemporary statistics: Contributions in honor of Jayanta K. Ghosh, pp. 318–329, Institute of Mathematical Statistics, 2008.

Bocquet, M., Pires, C. A., and Wu, L.: Beyond Gaussian statistical modeling in geophysical data assimilation, Monthly Weather Review, 138, 2997–3023, 2010.

Buizza, C., Quilodrán Casas, C., Nadler, P., Mack, J., Marrone, S., Titus, Z., Le Cornec, C., Heylen, E., Dur, T., Baca Ruiz, L., Heaney, C., Díaz Lopez, J. A., Kumar, K. S., and Arcucci, R.: Data Learning: Integrating Data Assimilation and Machine Learning, Journal of Computational Science, 58, https://doi.org/10.1016/j.jocs.2021.101525, 2022.

Burgers, G., van Leeuwen, P. J., and Evensen, G.: Analysis scheme in the ensemble Kalman filter, Monthly weather review, 126, 1719–1724, 1998.

Byrne, S., Wilcox, L. C., and Churavy, V.: MPI.jl: Julia bindings for the Message Passing Interface, Proceedings of the JuliaCon Conferences, 1, 68, https://doi.org/10.21105/jcon.00068, 2021.

Carlson, F. B., Roy, P., and Lu, Y.: LowLevelParticleFilters.jl: State estimation, smoothing and parameter estimation using Kalman and particle filters, https://github.com/baggepinnen/LowLevelParticleFilters.jl, 2018.

Carrassi, A., Bocquet, M., Bertino, L., and Evensen, G.: Data assimilation in the geosciences: An overview of methods, issues, and perspectives, Wiley Interdisciplinary Reviews: Climate Change, 9, e535, https://doi.org/10.1002/wcc.535, 2018.

Churavy, V., Godoy, W. F., Bauer, C., Ranocha, H., Schlottke-Lakemper, M., Räss, L., Blaschke, J., Giordano, M., Schnetter, E., Omlin, S., Vetter, J. S., and Edelman, A.: Bridging HPC Communities through the Julia Programming Language, arXiv e-prints, https://doi.org/10.48550/arXiv.2211.02740, 2022.

Cotter, C., Crisan, D., Holm, D., Pan, W., and Shevchenko, I.: Data assimilation for a quasi-geostrophic model with circulation-preserving stochastic transport noise, Journal of Statistical Physics, 179, 1186–1221, 2020.

Del Moral, P.: Feynman-Kac formulae, Springer, 2004.

Douc, R. and Cappé, O.: Comparison of resampling schemes for particle filtering, in: Ispa 2005. proceedings of the 4th international symposium on image and signal processing and analysis, 2005., pp. 64–69, IEEE, 2005.

Doucet, A., Godsill, S., and Andrieu, C.: On sequential Monte Carlo sampling methods for Bayesian filtering, Statistics and Computing, 10, 197–208, https://doi.org/10.1023/A:1008935410038, 2000.

Dunbar, O. R. A., Lopez-Gomez, I., Garbuno-Iñigo, A., Huang, D. Z., Bach, E., and long Wu, J.: EnsembleKalmanProcesses.jl: Derivative-free ensemble-based model calibration, Journal of Open Source Software, 7, 4869, https://doi.org/10.21105/joss.04869, 2022.

Evensen, G.: Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics, Journal of Geophysical Research: Oceans, 99, 10 143–10 162, 1994.

Farchi, A. and Bocquet, M.: Comparison of local particle filters and new implementations., Nonlinear Processes in Geophysics, 25, 2018.

Foreman-Mackey, D., Agol, E., Ambikasaran, S., and Angus, R.: Fast and Scalable Gaussian Process Modeling with Applications to Astronomical Time Series, The Astronomical Journal, 154, 220, https://doi.org/10.3847/1538-3881/aa9332, 2017.

Gailler, A., Hébert, H., Loevenbruck, A., and Hernandez, B.: Simulation systems for tsunami wave propagation forecasting within the French tsunami warning center, Natural Hazards and Earth System Sciences, 13, 2465–2482, https://doi.org/10.5194/nhess-13-2465-2013, 2013.

Giordano, M., Klöwer, M., and Churavy, V.: Productivity meets Performance: Julia on A64FX, in: 2022 IEEE International Conference on Cluster Computing (CLUSTER), pp. 549–555, https://doi.org/10.1109/CLUSTER51413.2022.00072, 2022.

Gordon, N. J., Salmond, D. J., and Smith, A. F.: Novel approach to nonlinear/non-Gaussian Bayesian state estimation, in: IEE Proceedings F (Radar and Signal Processing), vol. 140, pp. 107–113, IET, 1993.

Graham, M. M. and Thiery, A. H.: A scalable optimal-transport based local particle filter, arXiv preprint arXiv:1906.00507, 2019.

Gusman, A. R., Sheehan, A. F., Satake, K., Heidarzadeh, M., Mulia, I. E., and Maeda, T.: Tsunami data assimilation of Cascadia seafloor pressure gauge records from the 2012 Haida Gwaii earthquake, Geophysical Research Letters, 43, 4189–4196, https://doi.org/10.1002/2016GL068368, 2016.

Hatfield, S.: samhatfield/letkf-speedy: Publication, https://doi.org/10.5281/zenodo.1198432, 2018.

Jordán, A., Eyheramendy, S., and Buchner, J.: State-space Representation of Matérn and Damped Simple Harmonic Oscillator Gaussian Processes, Research Notes of the AAS, 5, 107, https://doi.org/10.3847/2515-5172/abfe68, 2021.

Kondo, K. and Miyoshi, T.: Non-Gaussian statistics in global atmospheric dynamics: a study with a 10 240-member ensemble Kalman filter using an intermediate atmospheric general circulation model, Nonlinear Processes in Geophysics, 26, 211–225, https://doi.org/10.5194/npg-26-211-2019, 2019.

Le Provost, M.: EnKF.jl: A framework for data assimilation with ensemble Kalman filter, https://github.com/mleprovost/EnKF.jl, 2016.

Lee, A. and Piibeleht, M.: SequentialMonteCarlo.jl: A light interface to serial and multi-threaded Sequential Monte Carlo, https://github.com/awllee/SequentialMonteCarlo.jl, 2017.

Leeuwen, P. J., Künsch, H. R., Nerger, L., Potthast, R., and Reich, S.: Particle filters for high-dimensional geoscience applications: A review, Quarterly Journal of the Royal Meteorological Society, 145, 2335–2365, https://doi.org/10.1002/qj.3551, 2019.

Lei, J., Bickel, P., and Snyder, C.: Comparison of ensemble Kalman filters under non-Gaussianity, Monthly Weather Review, 138, 1293–1306, 2010.

Lorenz, E. N.: Deterministic Nonperiodic Flow, Journal of Atmospheric Sciences, 20, 130 – 141, https://doi.org/10.1175/1520-0469(1963)020<0130:DNF>2.0.CO;2, 1963.

Maeda, T., Obara, K., Shinohara, M., Kanazawa, T., and Uehira, K.: Successive estimation of a tsunami wavefield without earthquake source data: A data assimilation approach toward real-time tsunami forecasting, Geophysical Research Letters, 42, 7923–7932, https://doi.org/10.1002/2015GL065588, 2015.

Miyoshi, T.: Ensemble Kalman Filter Experiments with a Primitive-equation Global Model, Ph.D. thesis, University of Maryland, College Park, 2005.

Miyoshi, T., Kondo, K., and Imamura, T.: The 10,240-member ensemble Kalman filtering with an intermediate AGCM, Geophysical Research Letters, 41, 5264–5271, 2014.

https://doi.org/10.5194/gmd-2023-38
Preprint. Discussion started: 21 March 2023

Molteni, F.: Atmospheric simulations using a GCM with simplified physical parameterizations. I: model climatology and variability in multi-decadal experiments, Climate Dynamics, 20, 175–191, 2003.

Nadler, P., Arcucci, R., and Guo, Y. K.: Data assimilation for parameter estimation in economic modelling, Proceedings - 15th International Conference on Signal Image Technology and Internet Based Systems, SISITS 2019, pp. 649–656, https://doi.org/10.1109/SITIS.2019.00106, 2019.

Ruzayqat, H., Er-Raiy, A., Beskos, A., Crisan, D., Jasra, A., and Kantas, N.: A lagged particle filter for stable filtering of certain high-dimensional state-space models, SIAM/ASA Journal on Uncertainty Quantification, 10, 1130–1161, 2022.

Sanpei, A., Okamoto, T., Masamune, S., and Kuroe, Y.: A data-assimilation based method for equilibrium reconstruction of magnetic fusion plasma and its application to reversed field pinch, IEEE Access, 9, 74 739–74 751, 2021.

Schauer, M., Gagnon, Y. L., St-Jean, C., and Cook, J.: Kalman.jl: Flexible filtering and smoothing in Julia, https://github.com/mschauer/Kalman.jl, 2018.

Schoenbrod, S.: KalmanFilters.jl, https://github.com/JuliaGNSS/KalmanFilters.jl, 2018.

Snyder, C.: Particle filters, the “optimal” proposal and high-dimensional systems, Proceedings of the ECMWF Seminar on Data Assimilation for Atmosphere and Ocean, pp. 6–9, http://www2.mmm.ucar.edu/people/snyder/papers/Snyder_ECMWFSem2011.pdf, 2011.

Sunberg, Z., Lasse, P., Bouton, M., Fischer, J., Becker, T., Saba, E., Moss, R., Gupta, J. K., Dressel, L., Kelman, T., Wu, C., and Thibaut, L.: ParticleFilters.jl: Simple particle filter implementation in Julia, https://github.com/JuliaPOMDP/ParticleFilters.jl, 2017.

Vetra-Carvalho, S., van Leeuwen, P. J., Nerger, L., Barth, A., Altaf, M. U., Brasseur, P., Kirchgessner, P., and Beckers, J. M.: State-of-the-art stochastic data assimilation methods for high-dimensional non-Gaussian problems, Tellus, Series A: Dynamic Meteorology and Oceanography, 70, 1–38, https://doi.org/10.1080/16000870.2018.1445364, 2018.