Lagrangian-Eulerian Multiscale Data Assimilation in Physical Domain based on Conditional Gaussian Nonlinear System

Hyeonggeun Yun[1] and Quanling Deng*[2]

[1]School of Computing, Australian National University, Canberra, ACT 2601, Australia
(Geun.Yun@anu.edu.au)
[2]School of Computing, Australian National University, Canberra, ACT 2601, Australia
(Quanling.Deng@anu.edu.au)

November 1, 2024

Contents

1 Introduction **2**

2 Background **4**
2.1 Two-layers Quasi-geostrophic (QG) model 4
2.2 Numerical schemes 5
2.3 Conditional Gaussian Nonlinear System (CGNS) 7

3 Numerical solution of the QG model **9**

4 Implementation of the QG DA via CGNS **12**
4.1 Estimating vorticity q given streamline function ψ for both layers 12
4.2 Estimating a second layer given ψ of first layer 14

5 Simulation Results **17**

6 Evaluation of the DA **19**

7 Conclusion **21**

8 Supporting Information **22**
8.1 4×4 grid discretisation for ψ update 22
8.2 Derivation of a_1 from $a_1\psi = -J(\psi, q)$ 23
8.3 Derivation of A_1 and a_1 CGNS matrices for recovering the second layer given the first layer . 23

*Supervisor

Keywords: Data assimilation, Lagrangian-Eulerian Multiscale Data Assimilation, Quasi geostrophic model, Conditional nonlinear gaussian system, Computational Science

Abstract

This research aims to further investigate the process of Lagrangian-Eulerian Multiscale Data Assimilation (LEMDA) by replacing the Fourier space with the physical domain. Such change in the perspective of domain introduces the advantages of being able to deal in non-periodic system and more intuitive representation of localised phenomena or time-dependent problems. The context of the domain for this paper was set as sea ice floe trajectories to recover the ocean eddies in the Arctic regions, which led the model to be derived from two-layer Quasi geostrophic (QG) model. The numerical solution to this model utilises the Conditional Gaussian Nonlinear System (CGNS) to accommodate the inherent non-linearity in analytical and continuous manner. The normalised root mean square error (RMSE) and pattern correlation (Corr) are used to evaluate the performance of the posterior mean of the model. The results corroborate the effectiveness of exploiting the two-layer QG model in physical domain. Nonetheless, the paper still discusses opportunities of improvement, such as deploying neural network (NN) to accelerate the recovery of local particle of Lagrangian DA for the fine scale.

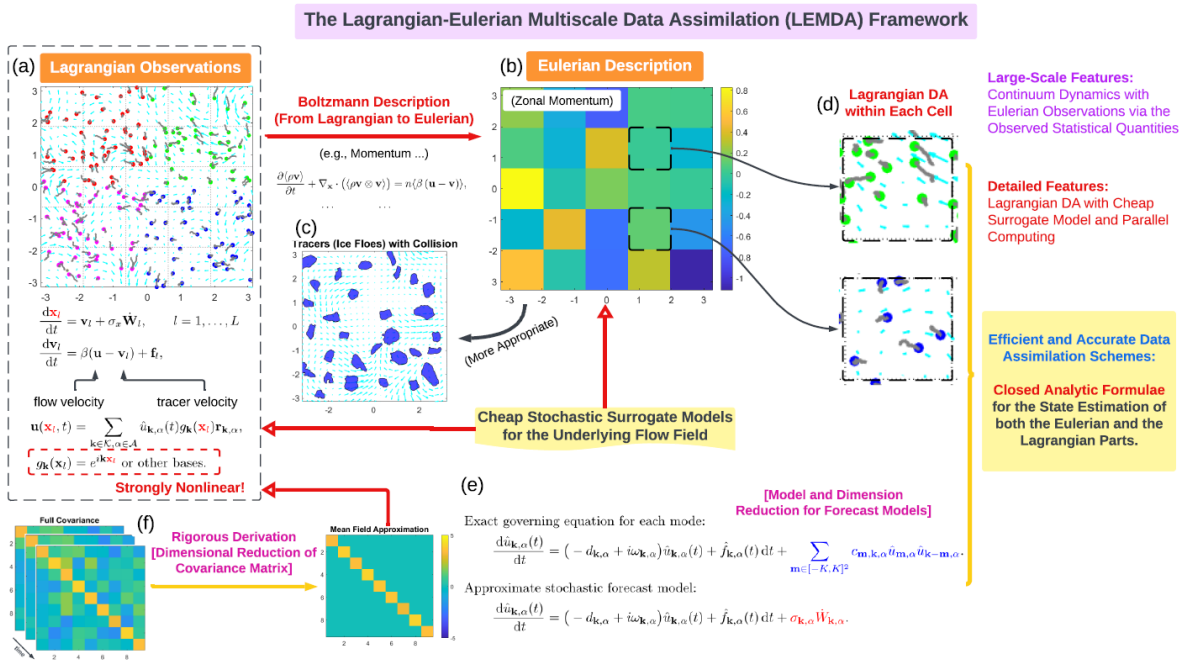
1 Introduction

Data assimilation (DA) has become an essential tool for improving predictions of complex physical systems, especially in fields like meteorology, oceanography and climate science, where observational data are sparse or noisy with high model uncertainties [1, 6]. The purpose of DA is to merge observational data with numerical model predictions to achieve an optimal estimate of the system's state [6]. By assimilating real-time observations into predictive models, DA can reduce forecast errors and enhance temporal consistency of simulations.

Traditional DA methods, however, face limitations when applied to systems characterised by multi-scale dynamics and significant nonlinearity. Lagrangian DA, for example, is well-suited to capturing small-scale structures through particle trajectories but is computationally demanding due to high dimensionality and nonlinearity in the Lagrangian observations [5]. On the other hand, Eulerian DA provides computational efficiency in representing large-scale flow fields on fixed grids, yet often lacks the resolution needed for smaller-scale features [9]. These challenges are exacerbated in environments with both large-scale and fine-scale structures, such as ocean eddies or atmospheric fronts, where accurate tracking of particle movements is essential for high-fidelity modelling.

The Lagrangian-Eulerian Multiscale Data Assimilation (LEMDA) framework, as introduced by Deng et al., aims to address these challenges by combining the strengths of both approaches [5]. LEMDA leverages the Eulerian DA for capturing large-scale dynamics (often by discretised grid cells) while using Lagrangian DA to resolve fine-scale features (often by individual particles), achieving a balanced and computationally efficient multiscale framework. It relies on the Boltzmann equation for deriving an Eulerian description of particle statistics, allowing for a continuum model that smooths out the noise and inherent in particle interactions and collisions. Moreover, its reliance on closed-form analytic solutions, as opposed to ensemble methods, offers significant computational advantages, which would be further amplified in high-dimensional, nonlinear applications. The outline of the framework is illustrated in Figure 1 below.

Figure 1: Overview of the LEMDA framework [5]. Panel (a): the Lagrangian observations and Lagrangian DA. Note the strong nonlinearity in the observational process of the Lagrangian DA. Panel (b): the Eulerian description of particle statistics, which can be derived from the Boltzmann description of the particle system. Panel (c): applying the Eulerian DA to the cases where the observed tracers are the ice floes with collisions. Panel (d): applying the Lagrangian DA to each grid cell to recover refined features that are missed by the large-scale Eulerian DA in the multiscale DA framework. Panel (e): the development of cheap stochastic surrogate models for the underlying flow field, which is a crucial part of allowing the analytic solvers of both the Eulerian and the Lagrangian DAs. Panel (f): rigorous derivation of reduced-order DA schemes for Lagrangian DA.



This research extends the LEMDA framework by exploring its application within the physical domain, moving beyond the current reliance on Fourier space. This shift to a physical domain enables the model to address localised, time-dependent phenomena more intuitively, which are critical in applications like sea ice floe tracking and ocean eddy recovery in Arctic regions. Specifically, the model is based on a two-layer quasi-geostrophic (QG) system, which provides a foundational structure to simulate large-scale geophysical flows with realistic assumptions. By implementing the Conditional Gaussian Nonlinear System (CGNS), this paper also addresses the inherent nonlinearity within the system, offering a solution to handle multiscale interactions in the physical domain.

The primary objective is to evaluate the effectiveness of applying the LEMDA framework, particularly the Eulerian part, to a two-layer QG model, which the DA will be aided by CGNS. The performance in recovering the posterior mean of the assimilated state was measured by commonly accepted metrics, including root mean square error (RMSE) and pattern correlation (Corr). Consequently, feasibility and insights of a physical domain-based LEMDA framework was validated through this work, setting the stage for future enhancements in multiscale data assimilation.

2 Background

2.1 Two-layers Quasi-geostrophic (QG) model

As the aim of the research focuses on the data assimilation framework for a specific landscape like Arctic, careful consideration is required in deciding the appropriate ocean or atmospheric model. Quasi-geostrophic (QG) model was deemed to be suitable with the following assumptions. The model is underpinned by a small Rossby number ($R_o << 1$) that makes the flow nearly geostrophic, which implies the inertia forces are significantly smaller than Coriolis forces [18, 4]. The Rossby number is given by

$$R_o = \frac{U}{fL}, \tag{1}$$

where U is a typical horizontal velocity, L is the horizontal length scale, and f is the Coriolis parameter. Such discrepancy suits a condition with predominantly horizontal flow. Another underlying assumption is the assertion of a hydrostatic balance in vertical motion [18, 4]. This means the vertical pressure gradient is balanced by the gravitational force, effectively allowing the model to ignore vertical accelerations. This is captured by the equation

$$\frac{\partial p}{\partial z} = -\rho g, \tag{2}$$

where p is the pressure, z is the vertical coordinate, ρ is the fluid density and g is the acceleration due to gravity. Furthermore, the flow is assumed to be incompressible in baroclinic models, which means the density does not change significantly with time or space in the horizontal directions [18, 4]. This assumption makes the use of a constant density possible. Altogether, it should be clear that the assumptions resembles of a large-scale, slowing-moving flows in the ocean and atmosphere like Arctic.

The main advantage of QG model in the given context is its extension to work for two layers. This two-layer QG can recover the important physical attributes at one depth, such as streamline function (ψ) and potential vorticity (q), by observing that of other depth. The equations below that establishes relationship between the layers are extremely useful, as they effectively set up a potential for inductive recovery.

$$\frac{\partial q_i}{\partial t} + J(\psi_i, q_i) = 0 \tag{3}$$

$$q_i = \nabla^2 \psi_i + \beta y + \frac{k_d^2}{2}(\psi_j - \psi_i), \tag{4}$$

where $i = \{1, 2\}$ denotes the level index, $j = 3 - i$ and the Jacobian is $J(\psi_i, q_i) = \frac{\partial \psi}{\partial x}\frac{\partial q}{\partial y} - \frac{\partial \psi}{\partial y}\frac{\partial q}{\partial x}$. For the sake of simplicity, the model is assumed to have no external force on the surface and depths of layers are even. Hence, the assumptions result in 0 for the RHS of Equation 3 and $\frac{k_d^2}{2}$ for Equation 4, respectively. Along with the assumptions, Table 1 illustrates the scale of the model, by providing the typical order of magnitude of relevant parameters. These were then adjusted and scaled accordingly throughout any computation involving the QG model (see Table 3). The Laplacian discretisation constant will be further discussed in Section 3, as it has been manually derived from the discretisation process.

Name	Symbol	Typical order of magnitude
Streamline function	ψ	Heavily depends on the initial streamline function
Potential vorticity	q	Heavily depends on the initial streamline function
Coriolis parameter	f_0	$10^{-4}s^{-1}$
Beta parameter	β	$10^{-11}t^{-1}s^{-1}$
Rossby radius of deformation	L_d	$50 \sim 100km$
Layer thickness	H	$100 \sim 1000m$
Buoyancy frequency	N	$10^{-2}s^{-1}$
Square of the deformation wavenumber	k_d^2	$10^{-7}m^{-2}$
Laplacian Discretisation constant	Ω	Heavily depends on other parameters

Table 1: Table of parameters for the two-layer QG model.

Despite a clear existence of coupling between layers from Equation 4, such link is not all that strong as ψ_j only appears once from the perspective of layer i. Another notable insight is the absence of time respective term for ψ_i, insinuating its necessity to couple with q_i in order to exploit the parameter in temporal perspective. Such property is to be later utilised in constructing the CGNS framework (see Section 2.3).

2.2 Numerical schemes

Before getting further into the details of CGNS, various numerical schemes to solve the model should first be discussed to identify their pros and cons as well as any notable insights. For the numerical weather prediction and fluid dynamics particularly, the accurate and stable numerical solution of the QG model is paramount. The reviews below provides key numerical schemes relevant to the QG model with insights into their accuracy, stability and applicability in varying modelling contexts.

Chehab and Moalla provide a foundational perspective by exploring several numerical approaches for the QG model, with a focus on the Forward Euler scheme [2]. The scheme approximates time derivative by the finite difference:

$$\phi^{n+1} = \phi^n + \Delta t f(\phi^n), \tag{5}$$

where ϕ is the state variable, such as potential vorticity or streamline function, $f(\phi^n)$ is the evaluated rate of change at the current time step, and Δt represents the time-step size. This formula's simplicity enables ease of implementation but may have limited stability and accuracy when applied to nonlinear systems with high-frequency oscillations, common in QG models. They discuss how variations in key parameters, such as the Rossby number and Reynolds number, influence divergence and model stability. They highlight that improper tuning of these parameters, or selection of a large Δt, can induce numerical instability. This analysis reinforces the necessity of careful time-step and parameter selection to maintain computational robustness, providing a baseline understanding of simple, explicit schemes. The discussion sets the stage for introducing more advanced methods that build upon the Forward Euler scheme's insights.

To leverage the lessons from the introductory findings, Medjo presented the leapfrog scheme as a second-order accurate alternative, offering a greater precision than the Forward Euler [12] . The scheme calculates the state variable at the next time step (ϕ^{n+1}) based on the state two steps back (ϕ^{n-1}), improving precision for oscillatory solutions:

$$\phi^{n+1} = \phi^{n-1} + 2\Delta t f(\phi^n). \tag{6}$$

However, a well-known drawback of the scheme is its susceptibility to time-splitting errors due to oscillatory instabilities inherent in two-step methods. To address this issue, Medjo recommends the Asselin-Robert time filter, a technique that dampens the time-splitting oscillations but at the cost of reducing the scheme's accuracy to first-order. The filter is defined as:

$$\phi^{n+1} = \phi^{n+1} - \epsilon(\phi^{n+1} - 2\phi^n + \phi^{n-1}), \tag{7}$$

where ϵ is a small coefficient that controls the damping strength. Despite the reduced accuracy to first-order, it significantly enhances the stability of the solution, enabling its use in the QG model for long-term simulations. The study demonstrates that filtering is crucial for balancing stability and accuracy, particularly for high-frequency oscillatory systems. This scheme should be positioned after the Forward Euler to highlight the progression to second-order accuracy and the role of time-filtering in numerical stability.

In contrast to purely numerical methods, Jamal presents an analytical approach based on Lie symmetry reductions, transforming the QG model's PDEs into a system of ODEs [8]. Recall the equation 3 for QG vorticity. Jamal applies the symmetry principles to reduce the PDE to an ODE form:

$$\frac{d\phi}{dt} = g(\phi, t), \tag{8}$$

where $g(\phi, t)$ encapsulates simplified dynamics, making the equations computationally less intensive and thus more efficient to solve.This analytical technique is particularly effective when parameterised for specific QG model layers, allowing efficient model simplification. This provides a unique contrast to traditional numerical solutions by demonstrating a pathway to reducing computational cost through dimensional reduction, especially under certain parameter constraints.

Lastly, Oka advocates the fourth-order Runge-Kutta (RK4) method for handling nonlinear dynamics within the QG model [14]. This is an advanced scheme that effectively balances stability and computational cost. The approach improves precision by evaluating intermediate steps within each time-step, formulated as:

$$\phi^{n+1} = \phi^n + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4), \tag{9}$$

where the intermediate values k_1, k_2, k_3 and k_4 are calculated as follows:

$$k_1 = f(\phi^n), \tag{10}$$

$$k_2 = f\left(\phi^n + \frac{\Delta t}{2}k_1\right), \tag{11}$$

$$k_3 = f\left(\phi^n + \frac{\Delta t}{2}k_2\right), \tag{12}$$

$$k_4 = f(\phi^n + \Delta t k_3). \tag{13}$$

By averaging these intermediate slopes, RK4 captures the underlying dynamics with high accuracy, making it suitable for nonlinear systems with significant variability, such as the QG model. The research also implements adaptive time-stepping, dynamically adjusting Δt based on error estimates to enhance computational efficiency and ensure stability without excessive recalculations. This technique is instrumental for high-fidelity simulations, particularly in nonlinear advection-diffusion systems, namely fluid dynamics and meteorology.

The literature above for solving the QG model highlight a progression from basic first-order methods to more sophisticated schemes capable of handling complex, nonlinear dynamics. Nonetheless, this research used the Forward Euler as its temporal numerical scheme since it serves a great benchmark. In addition, central finite difference was use to discretise the state variables as its approximation assures a second order error $O(h^2)$ compared to first order in forward difference, where h is the interval length between grid cells. This allows other schemes to easily replace to accommodate different needs of accuracy and stability, if it is found to be suitable with CGNS (which it is, as discussed in Section 4). It also eases the process of simulating the model numerically to spare more work on the main focus, data assimilation with CGNS.

2.3 Conditional Gaussian Nonlinear System (CGNS)

The CGNS offers a powerful framework to model complex dynamical systems that exhibit both nonlinear and non-Gaussian characteristics. In these systems, while the overall dynamics may be highly nonlinear, the conditional distributions of specific variables retain a Gaussian structure [13, 3]. This property is especially beneficial for data assimilation, uncertainty quantification and prediction, as it enables closed-form solutions for conditional mean and covariance, streamlining the analysis and computational tasks with associated with high-dimensional, nonlinear systems [13].

The framework typically uses two state variables, $\mathbf{X}$ (observational) and $\mathbf{Y}$ (estimation). It should be noted that each state variable is defined in multi-dimension (i.e. $\mathbf{X} \in \mathbb{C}^{n_1}$ and $\mathbf{Y} \in \mathbb{C}^{n_2}$), as exemplified in Section 4.1. With such state variables, the general form of the CGNS is expressed as:

$$d\mathbf{X} = [\mathbf{A}_0(\mathbf{X}, t) + \mathbf{A}_1(\mathbf{X}, t)\mathbf{Y}]\, dt + \mathbf{B}_1(\mathbf{X}, t) d\mathbf{W}_1(t), \tag{14}$$

$$d\mathbf{Y} = [\mathbf{a}_0(\mathbf{X}, t) + \mathbf{a}_1(\mathbf{X}, t)\mathbf{Y}]\, dt + \mathbf{b}_2(\mathbf{X}, t) d\mathbf{W}_2(t), \tag{15}$$

where $\mathbf{A_0}, \mathbf{A_1}, \mathbf{a_0}, \mathbf{a_1}$ denote the system matrices that may vary with both $\mathbf{X}$ and time t, while diffusion matrices $\mathbf{B_1}$ and $\mathbf{b_2}$ incorporate stochasticity through independent Wiener processes $\mathbf{W_1}$ and $\mathbf{W_2}$ [13, 7]. As signposted by the use of Wiener processes to compute noise, one beneficial property of the system is its continuity in observing and recovering, unlike traditional data assimilation framework, such as Kalman filter and Ensemble Kalman filter [17, 10, 6]. This lead to smoother propagation of uncertainties over time, and ultimately more reliable assimilation outcomes compared to other discrete models with observational interval.

Another key advantage of the CGNS lies in conditional Gaussian structure, which allows $\mathbf{Y}$, conditioned on $\mathbf{X}$, to follow a linear Gaussian process. This relationship simplifies the computation of conditional means (μ_f) and conditional covariances (R_f), which can be derived using closed-form expressions:

$$d\mu_f = (a_0 + a_1\mu_f)\, dt + (R_f A_1^*)(B_1 B_1^*)^{-1} \left(dX - (A_0 + A_1\mu_f)dt\right), \tag{16}$$

$$dR_f = [a_1 R_f + R_f a_1^* + b_2 b_2^* - (R_f A_1^*)(B_1 B_1^*)^{-1}(A_1 R_f)]dt, \tag{17}$$

where $.^*$ is the complex conjugate transpose, but this is to be treated as the transpose ($.^T$) in the paper, as all values are real number. These equations form the basis for optimal filtering within CGNS and enable efficient, scalable computations in high-dimensional systems, such as oceanographic and atmospheric models.

Nevertheless, the most difficult and critical part when implementing the framework involves the derivation of system matrices, especially $\mathbf{A_1}$ and $\mathbf{A_0}$, which is responsible for the coupling between observed and predicted variables. The construction of matrices and their impact on the stability will be among the main discussion of Section 4.

In addition to the closed filtering equations, CGNS supports optimal smoothing as well with analytical formulae, which further refines state estimation by incorporating future observations, thus providing a more accurate estimation of past states. The smoothing extends the filtering process by adjusting the previously calculated estimates based on subsequent observations, offering improved accuracy in systems with sparse data or long temporal dependencies, as expressed below.

$$\overleftarrow{d\mu_s} = \Big(-a_0 - a_1\mu_s + (b_2 b_2^*)R_f^{-1}(\mu_f - \mu_s)\Big)\, dt, \tag{18}$$

$$\overleftarrow{dR_s} = \Big(-(a_1 + (b_2 b_2^*)R_f^{-1})R_s - R_s(a_1^* + (b_2 b_2^*)R_f^{-1}) + b_2 b_2^*\Big)\, dt. \tag{19}$$

Here, the subscript 's' in the conditional mean and conditional covariance abbreviates 'smoother'. Furthermore, the notation $\overleftarrow{\cdot}$ depicts the backward flow at which the system is to be solved as it adjusts the past prediction based on future observations. Having this opposite direction to the optimal filtering computationally introduces extra $O(N_t)$, where N_t is the time spanned by the system. Hence, the implementation of this additional feature should be considered carefully, rather than adopting it by default. In the case of two-layers QG model, this extra layer was thought to be unnecessary, since the data has low sparsity with no explicit sign of long temporal dependency.

3 Numerical solution of the QG model

Solving the QG model means computing the values of ψ and q for each layer at each time step given the initial streamline function. To accommodate this, recall the two equations, 3 and 4 below. They must interact with each other carefully as they both involve parameters with different inherent properties.

$$\frac{\partial q}{\partial t} + \frac{\partial \psi}{\partial x}\frac{\partial q}{\partial y} - \frac{\partial \psi}{\partial y}\frac{\partial q}{\partial x} = 0 \tag{3}$$

$$q_i = \nabla^2 \psi_i + \beta y + \frac{k_d^2}{2}(\psi_j - \psi_i) \tag{4}$$

Firstly, note that all terms of Equation 3 are partial derivatives in respect to time and both physical dimensions. As the simulation is time-dependent by the nature of data assimilation, terms involving ∂t must exist, which only appears in this first equation. Hence, this equation was deemed to be the main equation to find the matrices A_0, A_1, a_0 and a_1 to perform CGNS filtering. Meanwhile, Equation 4 primarily reflects the interaction between two layers as the presence of subscripts i and j of ψ naturally suggest. This results in the entry of the CGNS matrices for each layer to be represented by terms involving the other layer as section 4 describes in detail.

To test such CGNS framework against a reliable and accurate benchmark, it was necessary to first adopt the numerical solution of the QG model since analytically solving convoluted trigonometry and derivative operations for a large matrix size (up to 10000×10000 for testing 100×100 mesh) with a great number of time steps ($\approx 10^4$) is beyond the available computation power. With computation times being differed by several orders of magnitude ($10 \sim 1000$), the analytical method in MATLAB would take several days to run the intensive simulations in the paper which was numerically done under 30 minutes. To further ensure reasonable computation time without sacrificing the validity of the results, both the simulation-specific and QG model parameters were scaled appropriately, as stated in the tables below.

Name	Symbol	Order of magnitude or constant value in simulation
Initial Time	T_0	$0s$
End time	T_f	$0.05 \sim 2s$
Timestep	dt	$10^{-4}s$
Mesh grid boundary	$\{(x_1, y_1), (x_2, y_2)\}$	$\{(0,0),(1,1)\}$
Mesh grid dimension	$N_x \times N_y$	50×50
Interval length between cells	$h = h_x = h_y$	0.02
Initial streamline function 1 (asymmetric sinusoidal)	$\psi_i^{t=0}$	$\psi_1^{t=0}(x,y) = -\sin(1.2\pi x)\sin(1.5\pi y) + 0.6\cos(2.3\pi x)\cos(2.8\pi y)$ $\psi_2^{t=0}(x,y) = \sin(3.1\pi x)\sin(0.8\pi y) + 0.7\cos(1.6\pi x)\cos(2.4\pi y)$
Initial streamline function 2 (symmetric Gaussian)	$\psi_i^{t=0}$	$\psi_1^{t=0}(x,y) = \exp(-(2(x-1/2)^2 + (y-1/2)^2)/(2(1/8)^2))$ $\psi_2^{t=0}(x,y) = \exp(-((x-1/2)^2 + 4(y-1/2)^2)/(3(1/8)^2))$
Smoothing kernel dimension	$n \times n$	5×5

Table 2: Table of temporal and spatial parameters of the simulation.

Note that the coefficients of initial streamline functions were randomly chosen. Furthermore, the asymmetric version was mainly used to generate all the findings to be discussed due to its ability to capture more realistic, chaotic flow of the particles, while the other was used purely to verify its sanity at the initial stage of modeling.

Name	Symbol	Order of magnitude or constant value in simulation
Streamline function	ψ	Heavily depends on the initial streamline function
Potential vorticity	q	Heavily depends on the initial streamline function
Coriolis parameter	f_0	$10^{-2}s^{-1}$
Beta parameter	β	$10^{-1}m^{-1}s^{-1}$
Rossby radius of deformation	L_d	$50 \sim 100km$
Layer thickness	H	$100 \sim 1000m$
Buoyancy frequency	N	$10^{-4} \sim 10^{-3}s^{-1}$
Square of the deformation wavenumber	k_d^2	$10m$

Table 3: Table of the QG model parameters used for the simulation

Another factor to consider was a boundary condition that guarantees the stability of cells near the boundary. For this, doubly periodic boundary condition was deployed in both x and y dimensions with all the edge cells being fixed to 0 for the sake of simplicity. This meant the independence of 196 (out of 2500) cells from the specific context of the model, which did not have significant impact on the overall simulation since the streamline function was intentionally initialised to generate stochastic behaviour in central regions. The main streamline function clearly outputs 0 as long as one of the dimensions is at 0.

With such parameters, computing q^{t+1} at the new time step given the data ψ^t, q^t at the previous time step can be done trivially by rearranging Equation 3.

$$\partial q = \left(\frac{\partial \psi^t}{\partial y} \frac{\partial q^t}{\partial x} - \frac{\partial \psi^t}{\partial x} \frac{\partial q^t}{\partial y} \right) \partial t \tag{20}$$

$$q^{t+1} = q^t + \partial q \tag{21}$$

It should be noted the PDE above is solved numerically by considering its equivalent ODE through central finite difference. To precisely describe the discretisation, the following indexing notations are introduced, which are to be kept constant throughout the paper, unless otherwise specified. The $\texttt{var}^t_{i_{k,l}}$ equates to a value of ith layer of the state variable, `var`, located in x and y coordinates of k and l of its discretised mesh, respectively, at time t. If notations on some parameters are skipped, it implies the expression is not dependent on them.

Hence, the discretisation of Equation 20 is:

$$\begin{aligned} q^{t+1}_{k,l} - q^t_{k,l} &= \left(\frac{\psi^t_{k,l+1} - \psi^t_{k,l-1}}{2h_y} \frac{\psi^t_{k+1,l} - \psi^t_{k-1,l}}{2h_x} - \frac{\psi^t_{k+1,l} - \psi^t_{k-1,l}}{2h_x} \frac{\psi^t_{k,l+1} - \psi^t_{k,l-1}}{2h_y} \right) dt \\ &= \left(\frac{(\psi^t_{k,l+1} - \psi^t_{k,l-1})(\psi^t_{k+1,l} - \psi^t_{k-1,l}) - (\psi^t_{k+1,l} - \psi^t_{k-1,l})(\psi^t_{k,l+1} - \psi^t_{k,l-1})}{4h} \right) dt \end{aligned} \tag{22}$$

Meanwhile, ψ^{n+1} is more challenging to update due to the absence of term $\frac{\partial \psi}{\partial t}$, which emphasises the importance of the first equation regarding time once again. Hence, Equation 4 was considered the most convenient to minimise the derivative operations. The solution begins with isolating the ψ_i terms before discretising them.

$$\nabla^2 \psi_i - \frac{k_d^2}{2} \psi_i = q_i - \beta y - \frac{k_d^2}{2} \psi_j$$

The central finite difference was used to numerically exploit $\nabla^2\psi_i$ as mentioned in Section 2.2. The discretisation at time n then becomes as follows.

$$\begin{aligned}\frac{\psi_{i_{k+1,l}} + \psi_{i_{k-1,l}} + \psi_{i_{k,l+1}} + \psi_{i_{k,l-1}} - 4\psi_{i_{k,l}}}{h^2} - \frac{k_d^2}{2}\psi_{i_{k,l}} &= C_{k,l} \\ \psi_{i_{k+1,l}} + \psi_{i_{k-1,l}} + \psi_{i_{k,l+1}} + \psi_{i_{k,l-1}} + \Omega\psi_{i_{k,l}} &= h^2 C_{k,l} = C'_{k,l} \\ A\vec{\psi} &= C', \end{aligned} \tag{23}$$

where $\Omega = -(4 + \frac{h^2 k_d^2}{2})$ and $C_{k,l} = q_{i_{k,l}} - \beta Y - \frac{k_d^2}{2}\psi_{j_{k,l}}$.

An example of the scheme with 4×4 grid is outlined in Section 8.1. In addition, Algorithm 1 below provides the pseudo code of numerical solution for the model. It should be highlighted how the matrix A is invariant since all the coefficients including Ω consist of constants. Such inherent property allows optimisation of the scheme since the matrix A needs to be computed only once before the program enters the main time marching loop.

Algorithm 1 Two-layers QG numerical solver ▷ All computations take place for both layers in parallel

Require: {All the parameters in Tables 2 and 3}
  init constants, time variables and mesh constraints
  init compute the invariant matrix A before time marching
  init ψ^1 and q^1 given the initial ψ
  for $t = 1 : N_t - 1$ **do**
    compute q^{t+1} by solving Equation 3 with 22
    compute ψ^{t+1} by solving Equation 4 with 23
  end for
return $\psi^{1:N_t}$ and $q^{1:N_t}$

4 Implementation of the QG DA via CGNS

4.1 Estimating vorticity q given streamline function ψ for both layers

As a stepping stone towards more realistic and complicated cases, the CGNS should first be utilised to recover one of the key state variables of q given its ψ for both layers. Consequently, the observational data and posterior data are defined as below.

$$X = \begin{bmatrix} \psi_1 \\ \psi_2 \end{bmatrix}, Y = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \tag{24}$$

It is worth noting that each state variable is described in 50×50 grid, which each grid can be transformed into matrix (see Equation 25).

$$X = \begin{bmatrix} \psi_{1_{1,1}} & \cdots & \psi_{1_{1,50}} \\ \vdots & \ddots & \vdots \\ \psi_{1_{50,1}} & \cdots & \psi_{1_{50,50}} \\ \psi_{2_{1,1}} & \cdots & \psi_{2_{1,50}} \\ \vdots & \ddots & \vdots \\ \psi_{2_{50,1}} & \cdots & \psi_{2_{50,50}} \end{bmatrix}, Y = \begin{bmatrix} q_{1_{1,1}} & \cdots & q_{1_{1,50}} \\ \vdots & \ddots & \vdots \\ q_{1_{50,1}} & \cdots & q_{1_{50,50}} \\ q_{2_{1,1}} & \cdots & q_{2_{1,50}} \\ \vdots & \ddots & \vdots \\ q_{2_{50,1}} & \cdots & q_{2_{50,50}} \end{bmatrix} \tag{25}$$

The indexing should be not be confused with what has been and will be used throughout the paper, since the above is based on the conventional matrix index, while the custom one that resembles of Cartesian plane coordinates. In this respect, the operations are mostly done element-wise, except when the entire state has to interact with the other state, namely $A_1 Y$ and $a_1 Y$. This means a careful consideration ins required when decomposing the derivative of each state variable in respect to time into (A_0 and $A_1 Y$) or (a_0 and $a_1 Y$). It is clearly more ideal to reduce complexity in the matrix multiplication, although some terms are inevitably part of it to capture the stochastic influence of Y.

Similar to the direct numerical solution (see Section 3), linearising q (Y) into its matrices is not hard. Moving the Jacobian term to the other side in Equation 3 yields:

$$dY = dq = -J(\psi, q)dt$$

Since the Jacobian only consists of terms that are multiplication of each state variable, there is no independent term from Y. Hence, the entire expression must be captured under $a_1 Y$, despite knowing it is not ideal. In fact, the discretisation of Jacobian term to separate a_1 from $a_1 Y$ is used repeatedly in this CGNS implementation (see Section 8.2 for its derivation). This results in:

$$dY = \begin{bmatrix} dq_1 \\ dq_2 \end{bmatrix} = \left[\quad \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad + \quad \begin{bmatrix} -J(\psi_1, q_1) \\ -J(\psi_2, q_2) \end{bmatrix} \right] dt = [a_0 + a_1 Y]\, dt$$

Meanwhile, the decomposition of ψ cannot be as easy as directly using 4 in Section 3, since CGNS forces both variables to be represented in respect to time to obtain a continuously analytical solution. Hence, Equation 4 must be substituted into Equation 3 as it is the only way to extract a term $\frac{d\psi}{dt}$.

$$\frac{d(\nabla^2\psi_i + \beta y + \frac{k_d^2}{2}(\psi_j - \psi_i))}{dt} = -J(\psi_i, q_i)$$

Note that values of y-coordinates y is independent of a time, leading to a removal of βy.

$$\frac{d(\nabla^2\psi_i + \frac{k_d^2}{2}(\psi_j - \psi_i))}{dt} = -J(\psi_i, q_i)$$

The issue with the current state of decomposition is that $fracd\psi_i dt$ cannot easily be isolated due to the complexity of reversing $\nabla^2\psi_i$ term. At this point, it was realised that shifting the perspective of j introduces new paradigm as $\frac{k_d^2}{2}$ was the only variable linked with ψ_j.
Hence,

$$\frac{d\psi_j}{dt} = \frac{\partial\left(\psi_i - \frac{2}{k_d^2}\nabla^2\psi_i\right)}{\partial t} - \frac{2}{k_d^2}J(\psi_i, q_i)$$

This time, the first term is independent of q, and hence, can be represented as A_0. Now, all the system matrices required by CGNS framework is obtained, as represented as below.

$$dX = \begin{bmatrix} d\psi_1 \\ d\psi_2 \end{bmatrix} = \left[\begin{bmatrix} \frac{\partial\left(\psi_2 - \frac{2}{k_d^2}\nabla^2\psi_2\right)}{\partial t} \\ \frac{\partial\left(\psi_1 - \frac{2}{k_d^2}\nabla^2\psi_1\right)}{\partial t} \end{bmatrix} + \begin{bmatrix} -\frac{2}{k_d^2}J(\psi_2, q_2) \\ -\frac{2}{k_d^2}J(\psi_1, q_1) \end{bmatrix} \right] dt = [A_0 + A_1 Y]\, dt \tag{26}$$

$$dY = \begin{bmatrix} dq_1 \\ dq_2 \end{bmatrix} = \left[\begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -J(\psi_1, q_1) \\ -J(\psi_2, q_2) \end{bmatrix} \right] dt = [a_0 + a_1 Y]\, dt \tag{27}$$

$$A_0 = \begin{bmatrix} \frac{\partial\left(\psi_2 - \frac{2}{k_d^2}\nabla^2\psi_2\right)}{\partial t} \\ \frac{\partial\left(\psi_1 - \frac{2}{k_d^2}\nabla^2\psi_1\right)}{\partial t} \end{bmatrix}, \; A_1 = a_1 \begin{bmatrix} 0 & \frac{2}{k_d^2} \\ \frac{2}{k_d^2} & 0 \end{bmatrix} \tag{28}$$

$$a_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \; a_1 Y = \begin{bmatrix} -J(\psi_1, q_1) \\ -J(\psi_2, q_2) \end{bmatrix} \tag{29}$$

For the diffusion matrices, B_1 and b_2 were chosen as constant value with conventional Wiener processes being $W_1, W_2 = \texttt{Rand}(0,1)\sqrt{dt}$ for each cells of both layers. The last component to implement was the conditional covariance matrix R_f.

$$C_{Y|X} = \Sigma_{YY} - \Sigma_{YX}\Sigma_{XX}^{-1}\Sigma_{XY}$$

Since there was no known covariance for the arbitrarily created the initial streamline functions, 1% of the simulation time was dedicated to compute a covariance as well as values of both state variables via the numerical solution to truly reflect the trend. It is worth noting such technique is widely used at the initial stage of data assimilation to ensure stability. The Algorithm 2 below demonstrates the entire process.

Algorithm 2 Two-layers QG CGNS solver ▷ All computations take place for both layers in parallel

Require: {All the parameters in Tables 2 and 3}
  init constants, time variables and mesh constraints
  init compute the invariant matrix A before time marching
  init ψ^1, q^1 and μ_f for both layers given the initial ψ
  for $t = 1 : \frac{N_t}{100} - 1$ **do**
    compute q^{t+1} by solving Equation 3 with 22
    compute ψ^{t+1} by solving Equation 4 with 23
    update $\Sigma\psi$ and Σq for R_f
  end for
  init R_f based on $\psi^{1:\frac{N_t}{100}}$, $\Sigma\psi$, $q^{1:\frac{N_t}{100}}$ and Σq
  for $t = \frac{N_t}{100} : N_t$ **do**
    compute observational noise
    compute ψ^{t+1} with μ_f^t
    find A_0, A_1, a_0 and a_1 with Equations 28 and 29
    compute μ_f^{t+1} via CGNS framework with Equation 16
    compute R_f^{t+1} via CGNS framework with Equation 17
  end for
return $\psi^{1:N_t}$ and $\mu_f^{1:N_t}$

4.2 Estimating a second layer given ψ of first layer

Now that the use of CGNS to recover physical properties of two-layers QG model is found to be possible, its applicability in more realistic setting must be discussed. Only ψ_1 is observed now with aim to recover the attributes of second layer. Since attributes among the same layer can be numerically solved as discussed in Section 3, the Y of this DA was chosen to be ψ_2.

$$X = \left[\psi_1\right], Y = \left[\psi_2\right] \tag{30}$$

Same approach from 4.1 was naively taken to deconstruct equations into the CGNS framework as follows.

$$dX = \left[d\psi_1\right] = \left[\left[\frac{\partial\left(\psi_2 - \frac{2}{k_d^2}\nabla^2\psi_2\right)}{\partial t}\right] + \left[-\frac{2}{k_d^2}J(\psi_2, q_2)\right]\right]dt = \left[A_0 + A_1 Y\right]dt \tag{31}$$

$$dY = \left[d\psi_2\right] = \left[\left[\frac{\partial\left(\psi_1 - \frac{2}{k_d^2}\nabla^2\psi_1\right)}{\partial t}\right] + \left[-\frac{2}{k_d^2}J(\psi_1, q_1)\right]\right]dt = \left[a_0 + a_1 Y\right]dt \tag{32}$$

$$A_0 = \left[\frac{\partial\left(\psi_2 - \frac{2}{k_d^2}\nabla^2\psi_2\right)}{\partial t}\right],\ A_1\psi_2 = -\frac{2}{k_d^2}J(\psi_2, q_2) \tag{33}$$

$$a_0 = \left[\frac{\partial\left(\psi_1 - \frac{2}{k_d^2}\nabla^2\psi_1\right)}{\partial t}\right],\ a_1\psi_2 = -\frac{2}{k_d^2}J(\psi_1, q_1) \tag{34}$$

However, this derivation was found to be inappropriate in capturing the weak coupling between two layers (see 8.3 for more details). More specifically, it lacks stability when properly discretised (see 8.3), which the form appears to be extremely convoluted, suggesting the disparity between theory and practicality once again; The simulation blew up after a certain period of time, although it occurred later for longer time settings. Since the approach was arguably what was enforced by the physical equations, a different perspective had

to be taken that no longer directly revolves around equations 3 and 4.

Hence, the alternative approach was to exploit the numerical scheme as described in Section 3, because of its well defined discretisation from the analytical solution of the equations. This does not necessarily mean the compromise of accuracy to enhance stability, because the central finite difference will still generate approximations with an error term proportional to h^2 from the already discretised form, which happens to take identical value of h.

As stated in Section 3, ψ is updated via 23,

$$\psi_{i_{k+1,l}} + \psi_{i_{k-1,l}} + \psi_{i_{k,l+1}} + \psi_{i_{k,l-1}} + \Omega\psi_{i_{k,l}} = h^2 C_{k,l} = C'_{k,l}$$
$$A\vec{\psi} = C',$$

where $\Omega = -(4 + \frac{h^2 k_d^2}{2})$, $C_{k,l} = q_{i_{k,l}} - \beta Y - \frac{k_d^2}{2}\psi_{j_{k,l}}$. From this, the relevant matrices for the CGNS can be constructed as follows.
Let $d\psi_1 = \psi_1^{t+1} - \psi_1^t$ at time t.
Then, terms of ψ_1 at each timestep can be expressed as:

$$\psi_1^{t+1} = A^{-1}h^2 C \tag{35}$$
$$= h^2 A^{-1}\left(q_1^{t+1} - \beta Y - \frac{k_d^2}{2}\psi_2^t\right) \tag{36}$$

$$\psi_1^t = A^{-1}h^2 C \tag{37}$$
$$= h^2 A^{-1}\left(q_1^t - \beta Y - \frac{k_d^2}{2}\psi_2^{t-1}\right) \tag{38}$$

Substitute them into $d\psi = \psi^{t+1} - \psi^t$:

$$d\psi_1 = h^2 A^{-1}\left(q_1^{t+1} - \beta Y - \frac{k_d^2}{2}\psi_2^t\right) - h^2 A^{-1}\left(q_1^t - \beta Y - \frac{k_d^2}{2}\psi_2^{t-1}\right)$$
$$= h^2 A^{-1}\left(q_1^{t+1} - q_1^t - \frac{k_d^2}{2}(\psi_2^t - \psi_2^{t-1})\right)$$

This now captures the dynamics of ψ_1 in the alternative approach. For ψ_2, it was necessary to make $d\psi_2 = \psi_2^t - \psi_2^{t-1}$, because q_2^{t+1} is required to be able to use ψ_2^{t+1}, which is not computed at the start. Decrementing a timestep by one also intuitively makes sense since utilising q_2^{t+1} based on the assumption that ψ_2^{t+1} is found, but the goal of this CGNS is to recover ψ_2^{t+1}.

$$d\psi_2 = h^2 A^{-1}\left(q_2^t - \beta Y - \frac{k_d^2}{2}\psi_1^{t-1}\right) - h^2 A^{-1}\left(q_2^{t-1} - \beta Y - \frac{k_d^2}{2}\psi_1^{t-2}\right)$$
$$= h^2 A^{-1}\left(q_2^t - q_2^{t-1} - \frac{k_d^2}{2}(\psi_1^{t-1} - \psi_1^{t-2})\right)$$

Using the above definitions of $d\psi_1$ and $d\psi_2$, the system matrices can be computed.

$$dX = \left[d\psi_1\right] = \left[\left[\tfrac{1}{dt} \cdot h^2 A^{-1}\left(q_1^{t+1} - q_1^t + \tfrac{k_d^2}{2}\psi_2^{t-1}\right)\right] + \left[\tfrac{1}{dt} \cdot -\tfrac{h^2 k_d^2}{2} A^{-1}\psi_2^t\right]\right] dt = \left[A_0 + A_1 Y\right] dt \tag{39}$$

$$dY = \left[d\psi_2\right] = \left[\frac{1}{dt} \cdot h^2 A^{-1}\left(q_2^t - q_2^{t-1} - \frac{k_d^2}{2}(\psi_1^{t-1} - \psi_1^{t-2})\right)\right] dt = \left[a_0 + a_1 Y\right] dt \tag{40}$$

$$A_0 = \left[\tfrac{1}{dt} \cdot h^2 A^{-1}\left(q_1^{t+1} - q_1^t + \tfrac{k_d^2}{2}\psi_2^{t-1}\right)\right],\ A_1 = \frac{1}{dt} \cdot -\frac{h^2 k_d^2}{2} A^{-1} \tag{41}$$

$$a_0 = \left[\tfrac{1}{dt} \cdot h^2 A^{-1}\left(q_2^t - q_2^{t-1} - \tfrac{k_d^2}{2}(\psi_1^{t-1} - \psi_1^{t-2})\right)\right],\ a_1 = 0 \tag{42}$$

It should be noted that the ψ_2 terms that are computed at previous timesteps, and therefore considered to be known, is independent of ψ_2^t. This allows the previously predicted values to be a part of observation, which leads a_1 to be empty matrix. This interestingly depicts the recovery of posterior mean is completely independent of the observations of ψ_2 itself, which equates to the full dependence of other variables. Such property corroborates to the strong coupling between state variables, fulfilling the motivation to alter the native approach. The pseudo code for this algorithm is provided below.

Algorithm 3 Two-layers QG CGNS advanced solver

Require: {All the parameters in Tables 2 and 3}
  init constants, time variables and mesh constraints
  init compute the invariant matrix A before time marching
  init ψ^1, q^1 and μ_f for both layers given the initial ψ
  for $t = 1 : \frac{N_t}{100} - 1$ **do**
    compute q^{t+1} by solving Equation 3 with 22
    compute ψ^{t+1} by solving Equation 4 with 23
    update $\Sigma\psi$ and Σq for R_f
  end for
  init R_f based on $\psi^{1:\frac{N_t}{100}}$, $\Sigma\psi$, $q^{1:\frac{N_t}{100}}$ and Σq
  for $t = \frac{N_t}{100} : N_t$ **do**
    compute observational noise
    compute ψ_2^{t+1} with μ_f^t
    compute q_1^{t+1} and q_2^{t+1} based on ψ^{t+1}
    find A_0, A_1, a_0 and a_1 with Equations 28 and 29
    compute μ_f^{t+1} via CGNS framework with Equation 41
    compute R_f^{t+1} via CGNS framework with Equation 42
  end for
return $\psi_2^{1:N_t}$ and $\mu_f^{1:N_t}$

5 Simulation Results

There were 3 type of simulations for each initial streamline functions (sinusoidal and Gaussian in Table 2. Firstly, CGNS framework that observes both layers of QG model in Section 4.1 was simulated along with the numerical solution in Section 3, by following Algorithms 2 and 1, respectively. Similarly, the DA for recovering the second layer given only the first layer (4.2) was simulated with Algorithm 3. Finally, the recovered attributes were used to interpolate the trajectories of particles.

The processor which ran the simulation was `Intel i3-8130U CPU` with 8GB RAM. All the artefacts, including the MATLAB code, simulation videos and pefromance evaluation, can be found on the Github repository [19]. Below is a snippet of the simulation.

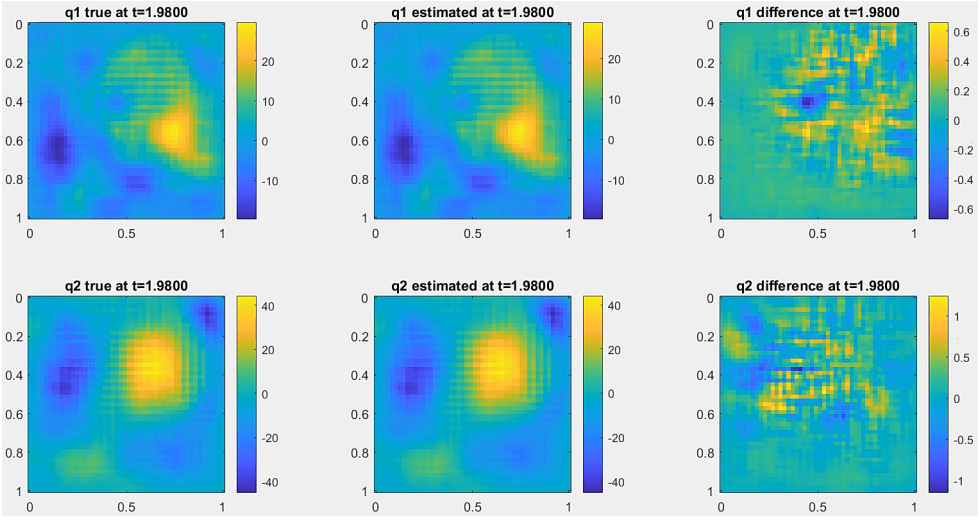


Figure 2: Simulation of true and recovered q at $t = 1.98$ with sinusoidal streamline function

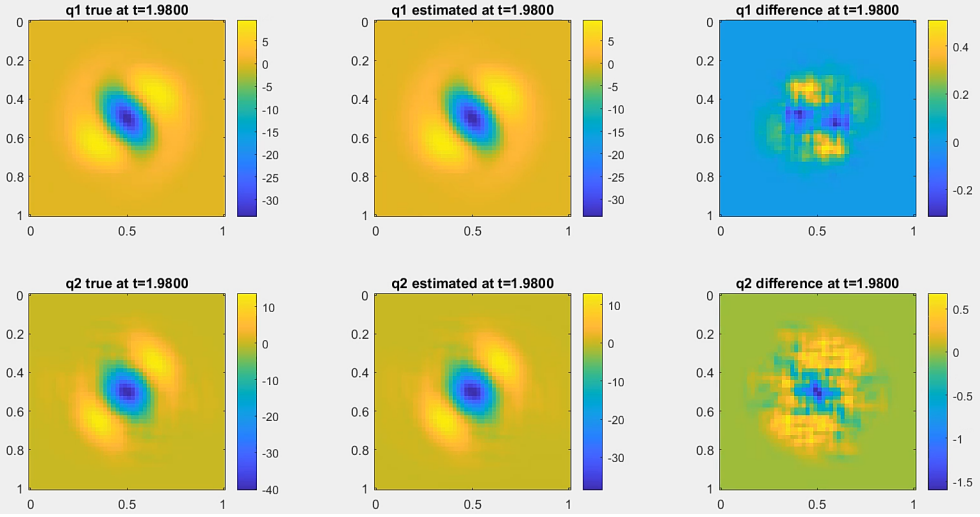


Figure 3: Simulation of true and recovered q at $t = 1.98$ with Gaussian streamline function

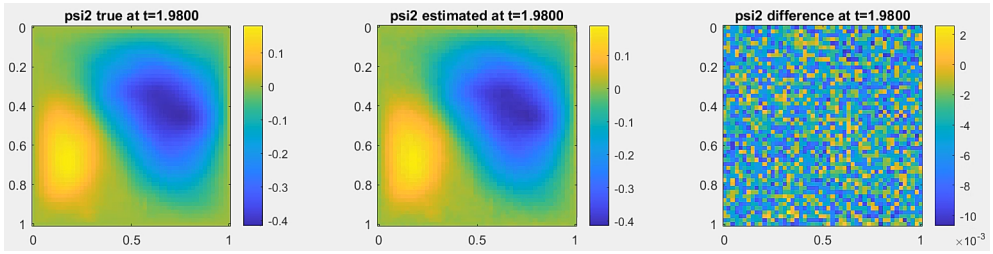


Figure 4: Simulation of true and recovered ψ_2 at $t = 1.98$ with sinusoidal streamline function

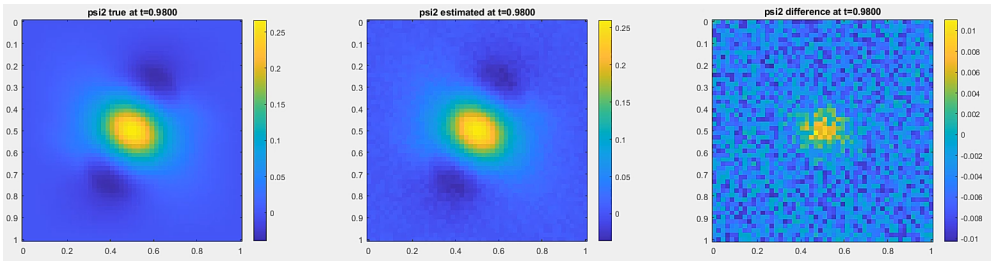


Figure 5: Simulation of true and recovered ψ_2 at $t = 0.98$ with Gaussian streamline function

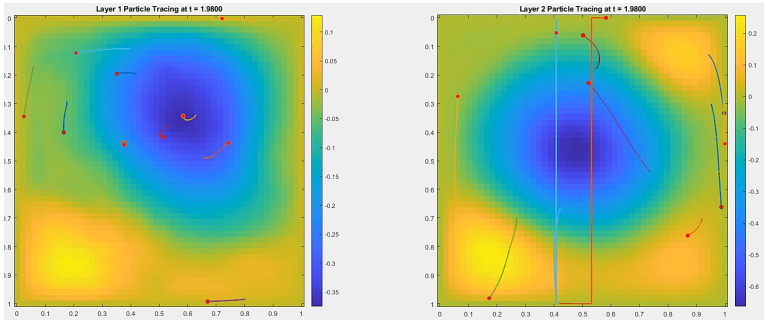


Figure 6: Simulation of particle tracing of two-layers QG model at $t = 1.98$ with sinusoidal streamline function

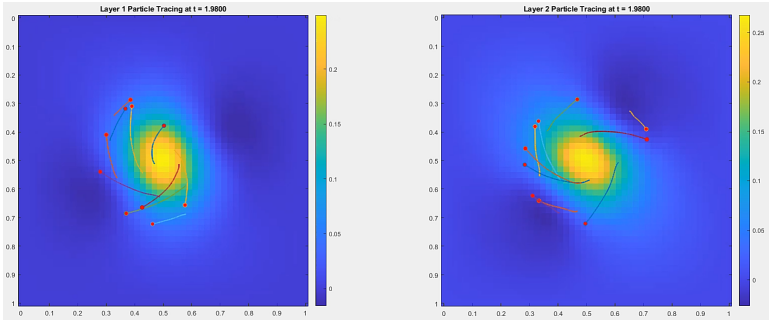


Figure 7: Simulation of particle tracing of two-layers QG model at $t = 1.98$ with Gaussian streamline function

Note that the absurd looking jump of particles in Figure 6 is a result of doubly periodic boundary condition fully trapping all particles inside the grid. It is also worth noting the sparsity of recovery via Algorithm 3 is much higher than that of Algorithm 2.

6 Evaluation of the DA

To evaluate the performance of the main algorithm (3), grid search method was used across the dimensions of mesh ($\{10\times10,\ 30\times30,\ 50\times50\}$) and number of timesteps ($\{500, 1000, 3000, 5000, 10000, 15000, 20000\}$).

Root mean squared error (RMSE) and pattern correlation (Corr) were used as two path-wise metrics. Both metrics are one of the most common ways to compare the posterior mean and truth as given below. RMSE indicates the average deviation between the simulated state and the true state. Meanwhile, Corr highlights how well the spatial pattern of the simulation aligns with the reference model.

Let μ and x be vectors for posterior mean and truth, respectively with a population of P. Then,

$$\mathbf{RMSE} = \left(\sqrt{\frac{\sum_{i=1}^{P}(\mu_i - x_i)^2}{P}}\right) / \mathbf{std}(x) \tag{43}$$

$$\mathbf{Corr} = \frac{\sum_{i=1}^{P}\left((\mu_i - \mathbf{mean}(\mu))(x_i - \mathbf{mean}(x))\right)}{\sqrt{\sum_{i=1}^{P}(\mu_i - \mathbf{mean}(\mu))^2}\sqrt{\sum_{i=1}^{P}(x_i - \mathbf{mean}(x))^2}} \tag{44}$$

The RMSE is always non-negative, with lower RMSE values indicating a more accurate posterior mean estimate. Since the RMSE in Equation 43 is normalised by the standard deviation of the true data, an RMSE above 1 implies that the error in the posterior mean estimate exceeds the mean variation in the true data. Pattern correlation ranges from -1 to 1, where higher values denote greater similarity between the two profiles. Typically, a reliable estimate achieves a pattern correlation above 0.5.

Figure 8 illustrates the results of RMSE, Corr and computation time as plots. The normalised computation time refers to actual computation time divided by $N_t \times N_x^2$ to depict the time complexity of the algorithm, which is often hard to be captured in the original time plot, especially with only 3 coordinates for each line. Therefore, the linearity in normalised graph apart from $N_t = 500$ indicates the time complexity of $O(N_t \times N_x^2)$, which follows from a $N_x \times N_y$ (where $N_x = N_y$) matrix performing operations iteratively over N_t times. This can be further corroborated by comparing the raw computation time across different N_t when N_x is fixed to 50. For example, $N_t = 5000$ and $N_t = 10000$ in such settings lasted about 400 and 800 seconds, respectively, which is proportional to the ratio between the N_ts. Similarly, $N_t = 20000$ indeed ran for 1800 seconds, which is roughly quadruple and double of the other two settings, respectively. Furthermore, it can be seen that the most involved simulation approximately took 30 minutes under an average or below-average computational resources, suggesting a low demand of such operations.

Meanwhile, the RMSE metric, plotted on a logarithmic x-axis for clarity, effectively captures the algorithm's performance across timesteps N_t and grid sizes N_x. Notably, RMSE values decrease steadily as N_t increases, showing that the error between the posterior mean estimate and the true state reduces over time. This decline in RMSE appears to converge toward a value around 0.1, irrespective of the grid size. This convergence suggests that the DA algorithm achieves a consistent level of accuracy over time, largely independent of spatial resolution. In other words, as the time marching scheme progresses, the DA framework consistently refines its estimates, reinforcing its robustness across varying spatial scales. This outcome highlights one of DA's primary strengths: its adaptability and effectiveness in multiscale settings, where it can achieve reliable accuracy regardless of spatial granularity.

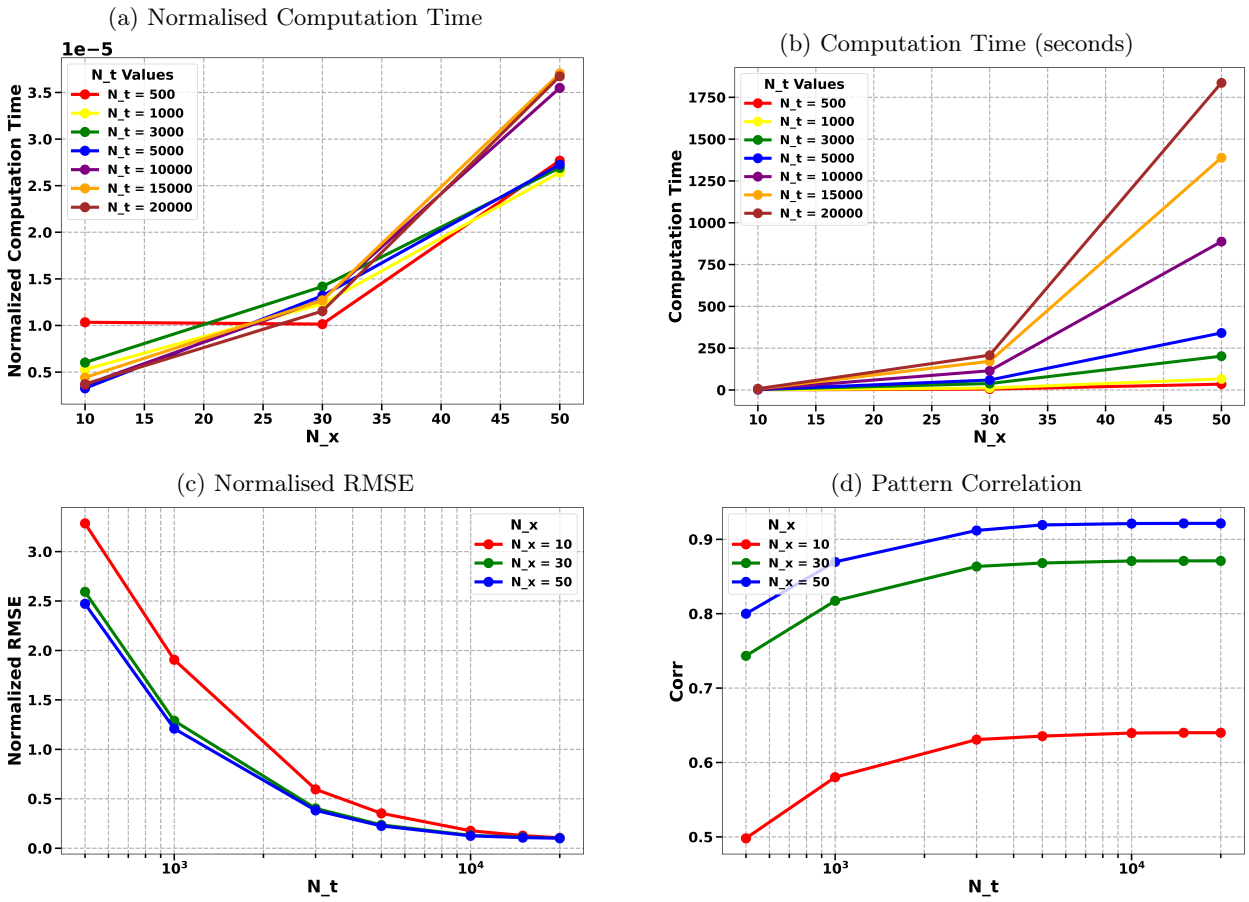


Figure 8: Performance evaluation of two-layer QG model simulation via CGNS

The Corr metric exhibits a general upward trend with respect to the number of timesteps across different grid sizes, indicating improved alignment between the model estimate and the true state as the assimilation process advances. However, the influence of N_x on Corr is significant. Larger grid sizes consistently yield higher Corr values, showing that higher spatial resolution enables the algorithm to better capture the pattern of the true state. For example, Corr is maximised at 0.92 for $N_x = 50$, while $N_x = 10$ can go as high as 0.64. Interestingly, Corr values show diminishing returns after $N_t = 3000$, where further increases in N_t contribute less to improving Corr across all N_x. This plateau suggests that while finer temporal resolution continues to provide marginal benefits, the spatial resolution ultimately becomes a limiting factor in the Corr metric's growth. This outcome implies that the data assimilation (DA) framework is particularly sensitive to spatial granularity, meaning that for more detailed alignment with the true state pattern, a sufficiently high grid resolution N_x is essential.

Overall, the evaluation indicates the DA of two-layers QG model via CGNS have performed well with decreasing and increasing trends of RMSE and Corr with N_t and N_x as a major influence, respectively.

7 Conclusion

This study has introduced an adaptation of the Lagrangian-Eulerian Multiscale Data Assimilation (LEMDA) framework to the physical domain, utilising a two-layer quasi-geostrophic (QG) model and Conditional Gaussian Nonlinear System (CGNS) to provide a robust approach for data assimilation. The results have illustrated that this framework, through CGNS, can successfully maintain low RMSE values over time, independent of grid scale, and achieve consistent alignment with the true state across various spatial resolutions. The pattern correlation analysis underscores the framework's adaptability in multiscale applications, although sensitivity to spatial resolution also reveals that further benefits depend heavily on grid granularity. This adaptability and the model's demonstrated scalability establish a solid foundation for further exploration of advanced capabilities within this framework. Nonetheless, several promising avenues for future research arise from the findings of this study as follows.

Leveraging neural networks (NN) on a cell-by-cell basis represents a key pathway toward parallelising the computational processes inherent in the LEMDA framework. This approach could allow each cell in the spatial grid to independently perform the required assimilation steps, which would not only expedite the data assimilation process but also maximise computational efficiency on parallel computing architectures [15]. Neural networks, due to their inherently parallel structure, are well-suited for such spatially distributed tasks.

An intriguing extension of this research involves exploring whether the fundamental states variables can be accurately recovered solely by observing particle displacements. Current methods primarily rely on direct observations of ψ and q, but a system where only particle trajectories are observed could offer a more feasible setup in real-world applications. This approach would require developing advanced interpolation and estimation techniques, possibly with support from neural network-based inference, to reconstruct field states from sparse data.

Establishing rigorous analytical conditions for the convergence of the two-layer QG CGNS solver is essential for ensuring stability across various parameter settings. Such conditions would enable precise control over model parameters and provide clear boundaries within which the solver maintains both accuracy and stability. It would serve as critical tools for designing and fine-tuning the DA process, allowing researchers to optimize computational resources without compromising the fidelity of the posterior mean estimate [11].

Extending the two-layer QG model to support more layers offers the potential for modelling more complex systems that may involve vertical stratification, such as those found in oceanographic and atmospheric flows [16]. Each additional layer would introduce unique challenges related to inter-layer coupling and the propagation of uncertainties through the model, thus necessitating a refined data assimilation framework. However, using the two-layer model as an inductive step offers a scalable approach to incrementally increasing model complexity.

To conclude, this paper has laid the groundwork for further innovations in data assimilation by adapting the LEMDA framework to the physical domain and establishing a scalable, multiscale approach using CGNS within a two-layer QG model. These prospective research directions are instrumental in addressing real-world computational and observational challenges, with applications extending to fields requiring high-fidelity multiscale modelling. With these advances, LEMDA could evolve into a more versatile and powerful framework, capable of integrating complex, multi-layer, and observation-constrained environments.

8 Supporting Information

8.1 4×4 grid discretisation for ψ update

Recall Equation 23, $A\vec{\psi} = C'$. The vector will be in a form of

$$\vec{\psi} = \begin{bmatrix} \psi_{0,0} \\ \psi_{0,1} \\ \vdots \\ \psi_{0,N_y} \\ \psi_{1,0} \\ \vdots \\ \psi_{1,N_y} \\ \vdots \\ \psi_{N_x,N_y} \end{bmatrix}$$

Col index $j = 0, \ldots, (N_x+1)(N_y+1)$ of A denotes the coefficient of the j^{th} corresponding ψ from $\vec{\psi}$.
Row index $i = 0, \ldots, (N_x+1)(N_y+1)$ of A represents the equation at $k = \lfloor \frac{i}{N_x+1} \rfloor$ and $l = i \mod (N_y+1)$.
As an example, let $N_x, N_y = 3$, meaning a 4 by 4 grid with both horizontal and vertical indexes from 0 to 3. Then,

$$A\vec{\psi} = \begin{bmatrix}
\Omega & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & \Omega & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & \Omega & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & \Omega & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & \Omega & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & \Omega & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & \Omega & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & \Omega & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \Omega & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & \Omega & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & \Omega & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & \Omega & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \Omega & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & \Omega & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & \Omega & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & \Omega
\end{bmatrix}
\begin{bmatrix} \psi_{0,0} \\ \psi_{0,1} \\ \psi_{0,2} \\ \psi_{0,3} \\ \psi_{1,0} \\ \psi_{1,1} \\ \psi_{1,2} \\ \psi_{1,3} \\ \psi_{2,0} \\ \psi_{2,1} \\ \psi_{2,2} \\ \psi_{2,3} \\ \psi_{3,0} \\ \psi_{3,1} \\ \psi_{3,2} \\ \psi_{3,3} \end{bmatrix} = C'$$

For the sake of readability, $\psi_{k-1,l}$ and $\psi_{k+1,l}$ were coloured as light red and red, respectively, while $\psi_{k,l-1}$ and $\psi_{k,l+1}$ were coloured as light blue and blue, respectively. The vector $\vec{\psi}$ would then be solved by computing $A^{-1}C'$.

8.2 Derivation of a_1 from $a_1\psi = -J(\psi, q)$

Same indexing scheme was used as 8.1, to express interaction of each cell against every other cells. This results in a_1 with a size of 2500×2500 matrix for a 50×50 mesh.
The discretisation uses central finite difference as follows.

$$\begin{aligned} a_1\psi &= \frac{d\psi}{dy}\frac{dq}{dx} - \frac{d\psi}{dx}\frac{dq}{y} \\ &= \frac{\psi_{k,l+1} - \psi_{k,l-1}}{2h_y}\frac{q_{k+1,l} - q_{k-1,l}}{2h_x} - \frac{\psi_{k+1,l} - \psi_{k-1,l}}{2h_x}\frac{q_{k,l+1} - q_{k,l-1}}{2h_y} \\ &= \frac{(\psi_{k,l+1} - \psi_{k,l-1})(q_{k+1,l} - q_{k-1,l}) - (\psi_{k+1,l} - \psi_{k-1,l})(q_{k,l+1} - q_{k,l-1})}{4h^2} \end{aligned}$$

Now consider a n-th cell of a grid, which would be in n-th row of a_1 when vectorised. The columns at the row will now be

$$\begin{aligned} &q_{k+1,l} - q_{k-1,l}, \\ &q_{k-1,l} - q_{k+1,l}, \\ &q_{k,l-1} - q_{k,l+1}, \\ &q_{k,l+1} - q_{k,l-1}, \end{aligned}$$

for positions of $\psi_{k,l+1}, \psi_{k,l-1}, \psi_{k+1,l}, \psi_{k-1,l}$, respectively. As a final note, the matrix should be divided by $4h^2$ element-wise. Such construction of a_1 ensures $a_1\psi = -J(\psi, q)$.

8.3 Derivation of A_1 and a_1 CGNS matrices for recovering the second layer given the first layer

Recall the setting in Section 4.2

$$\frac{\partial\left(\psi_2 - \frac{2}{k_d^2}\nabla^2\psi_2\right)}{\partial t} - \frac{2}{k_d^2}J(\psi_2, q_2)$$

$$X = \left[\psi_1\right], Y = \left[\psi_2\right]$$

$$\begin{aligned} dX = \left[d\psi_1\right] &= \left[\left[\frac{\partial\left(\psi_2 - \frac{2}{k_d^2}\nabla^2\psi_2\right)}{\partial t}\right] + \left[-\frac{2}{k_d^2}J(\psi_2, q_2)\right]\right]dt = \left[A_0 + A_1 Y\right]dt \\ dY = \left[d\psi_2\right] &= \left[\left[\frac{\partial\left(\psi_1 - \frac{2}{k_d^2}\nabla^2\psi_1\right)}{\partial t}\right] + \left[-\frac{2}{k_d^2}J(\psi_1, q_1)\right]\right]dt = \left[a_0 + a_1 Y\right]dt \end{aligned}$$

$$\begin{aligned} A_0 &= \left[\frac{\partial\left(\psi_2 - \frac{2}{k_d^2}\nabla^2\psi_2\right)}{\partial t}\right],\ A_1\psi_2 = -\frac{2}{k_d^2}J(\psi_2, q_2) \\ a_0 &= \left[\frac{\partial\left(\psi_1 - \frac{2}{k_d^2}\nabla^2\psi_1\right)}{\partial t}\right],\ a_1\psi_2 = -\frac{2}{k_d^2}J(\psi_1, q_1) \end{aligned}$$

For the sake of readability, let u, d, r, l represents immediately above, below, right, left coordinates of a current coordinate (i, j), denoting $(i, j+1), (i, j-1), (i+1, j), (i-1, j)$, respectively. Assuming equal spacing in x and y dimensions, where $h = dx = dy$,

$$\begin{aligned} J(\psi_2, q_2) &= \frac{\partial \psi_2}{\partial x}\frac{\partial q_2}{y} - \frac{\partial \psi_2}{\partial y}\frac{\partial q_2}{\partial x} \\ &= \frac{\psi_{2_r} - \psi_{2_l}}{2dx}\frac{q_{2_u} - q_{2_d}}{2dy} - \frac{\psi_{2_u} - \psi_{2_d}}{2dy}\frac{q_{2_r} - q_{2_l}}{2dx} \\ &= \frac{1}{4h^2}\left((q_{2_u} - q_{2_d})\psi_{2_r} + (q_{2_d} - q_{2_u})\psi_{2_l} + (q_{2_l} - q_{2_r})\psi_{2_u} + (q_{2_r} - q_{2_l})\psi_{2_d}\right) \end{aligned}$$

This discretisation computes A_1. However, a careful approach must be taken for a_1 since the vector $\vec{\psi_2}$ does not directly appear on the RHS of Jacobian with ψ_1 and q_1.
Hence, Equation 4 was rearranged in terms of ψ_2 and substituted for $a_1\psi_2 = -\frac{2}{k_d^2}J(\psi_1, q_1)$ to be valid.

$$\begin{aligned} q_2 &= \nabla^2\psi_2 + \beta y + \frac{k_d^2}{2}(\psi_1 - \psi_2) \\ \psi_1 &= \frac{2}{k_d^2}(q_2 - \nabla^2\psi_2 - \beta y) + \psi_2 \end{aligned}$$

More coordinates notations are used, where $uu, ur, ul, dr, dl, dd, rr, ll$ represents $(i, j+2), (i+1, j+1), (i-1, j+1), (i+1, j-1), (i-1, j-1), (i, j-2), (i+2, j), (i-2, j)$ given a current coordinate of (i, j).

$$\begin{aligned} J(\psi_1, q_1) &= J\left(\frac{2}{k_d^2}(q_2 - \nabla^2\psi_2 - \beta y) + \psi_2, q_1\right) \\ &= \frac{\partial\left(\frac{2}{k_d^2}(q_2 - \nabla^2\psi_2 - \beta y) + \psi_2\right)}{\partial x}\frac{\partial q_1}{\partial y} - \frac{\partial\left(\frac{2}{k_d^2}(q_2 - \nabla^2\psi_2 - \beta y) + \psi_2\right)}{\partial y}\frac{\partial q_1}{\partial x} \end{aligned}$$

Discretise the first term and generalise it to the second term.

$$\begin{aligned} &\frac{\partial\left(\frac{2}{k_d^2}(q_2 - \nabla^2\psi_2 - \beta y) + \psi_2\right)}{\partial x}\frac{\partial q_1}{\partial y} \\ &= \frac{\frac{2}{k_d^2}(q_{2_r} - q_{2_l} - (\nabla^2\psi_{2_r} - \nabla^2\psi_{2_l})) + \psi_{2_r} - \psi_{2_l}}{2dx}\frac{q_{1_u} - q_{1_d}}{2dy} \\ &= \frac{\frac{2}{k_d^2}\left(q_{2_r} - q_{2_l} - \left(\frac{\psi_{2_{ur}} + \psi_{2_{dr}} + \psi_{2_{rr}} + \psi_{2_{(i,j)}} - 4\psi_{2_r}}{h^2} - \frac{\psi_{2_{ul}} + \psi_{2_{dl}} + \psi_{2_{(i,j)}} + \psi_{2_{ll}} - 4\psi_{2_l}}{h^2}\right)\right) + \psi_{2_r} - \psi_{2_l}}{2dx}\frac{q_{1_u} - q_{1_d}}{2dy} \\ &= \frac{\frac{2}{k_d^2}\left(q_{2_r} - q_{2_l} - \left(\frac{\psi_{2_{ur}} + \psi_{2_{dr}} + \psi_{2_{rr}} - 4\psi_{2_r} - \psi_{2_{ul}} - \psi_{2_{dl}} - \psi_{2_{ll}} + 4\psi_{2_l}}{h^2}\right)\right) + \psi_{2_r} - \psi_{2_l}}{2dx}\frac{q_{1_u} - q_{1_d}}{2dy} \\ &= \left(\frac{\frac{2}{k_d^2}(q_{2_r} - q_{2_l})}{2dx} - \frac{\frac{2}{k_d^2h^2}(\psi_{2_{ur}} + \psi_{2_{dr}} + \psi_{2_{rr}} - 4\psi_{2_r} - \psi_{2_{ul}} - \psi_{2_{dl}} - \psi_{2_{ll}} + 4\psi_{2_l}) + \psi_{2_r} - \psi_{2_l}}{2dx}\right)\frac{q_{1_u} - q_{1_d}}{2dy} \\ &= \frac{\frac{2}{k_d^2}(q_{2_r} - q_{2_l})(q_{1_u} - q_{1_d})}{4h^2} - \\ &\frac{2(q_{1_u} - q_{1_d})}{4k_d^2h^4}\left(\psi_{2_{ur}} + \psi_{2_{dr}} + \psi_{2_{rr}} + ((\frac{k_d^2h^2}{2} - 4)\psi_{2_r} - \psi_{2_{ul}} - \psi_{2_{dl}} - \psi_{2_{ll}} + (((4 - \frac{k_d^2h^2}{2})\psi_{2_l}\right) \end{aligned}$$

Hence, the second term can be computed as follows.

$$
\begin{aligned}
&\frac{\partial\left(\frac{2}{k_d^2}(q_2-\nabla^2\psi_2-\beta y)+\psi_2\right)}{\partial y}\frac{\partial q_1}{\partial x}\\
&=\frac{\frac{2}{k_d^2}(q_{2_u}-q_{2_d}-(\nabla^2\psi_{2_u}-\nabla^2\psi_{2_d})-2\beta dy)+\psi_{2_u}-\psi_{2_d}}{2dy}\frac{q_{1_r}-q_{1_l}}{2dx}\\
&=\frac{\frac{2}{k_d^2}\left(q_{2_u}-q_{2_d}-\left(\frac{\psi_{2_{uu}}+\psi_{2_{(i,j)}}+\psi_{2_{ur}}+\psi_{2_{ul}}-4\psi_{2_u}}{h^2}-\frac{\psi_{2_{(i,j)}}+\psi_{2_{dd}}+\psi_{2_{dr}}+\psi_{2_{dl}}-4\psi_{2_d}}{h^2}\right)-2\beta dy\right)+\psi_{2_u}-\psi_{2_d}}{2dy}\frac{q_{1_r}-q_{1_l}}{2dx}\\
&=\frac{\frac{2}{k_d^2}\left(q_{2_u}-q_{2_d}-\left(\frac{\psi_{2_{uu}}+\psi_{2_{ur}}+\psi_{2_{ul}}-4\psi_{2_u}-\psi_{2_{dd}}-\psi_{2_{dr}}-\psi_{2_{dl}}+4\psi_{2_d}}{h^2}\right)-2\beta dy\right)+\psi_{2_u}-\psi_{2_d}}{2dy}\frac{q_{1_r}-q_{1_l}}{2dx}\\
&=\left(\frac{\frac{2}{k_d^2}(q_{2_u}-q_{2_d}-2\beta dy)}{2dy}-\frac{\frac{2}{k_d^2h^2}\left(\psi_{2_{uu}}+\psi_{2_{ur}}+\psi_{2_{ul}}-4\psi_{2_u}-\psi_{2_{dd}}-\psi_{2_{dr}}-\psi_{2_{dl}}+4\psi_{2_d}h^2\right)+\psi_{2_u}-\psi_{2_d}}{2dy}\right)\frac{q_{1_r}-q_{1_l}}{2dx}\\
&=\frac{\frac{2}{k_d^2}(q_{2_u}-q_{2_d}-2\beta h)(q_{1_r}-q_{1_l})}{4h^2}-\\
&\frac{(q_{1_r}-q_{1_l})\frac{2}{k_d^2h^2}\left(\psi_{2_{uu}}+\psi_{2_{ur}}+\psi_{2_{ul}}+((\frac{k_d^2h^2}{2}-4)\psi_{2_u}-\psi_{2_{dd}}-\psi_{2_{dr}}-\psi_{2_{dl}}+(((4-\frac{k_d^2h^2}{2})\psi_{2_d}\right)}{4h^2}\\
&=\frac{\frac{2}{k_d^2}(q_{2_u}-q_{2_d}-2\beta h)(q_{1_r}-q_{1_l})}{4h^2}-\\
&\frac{2(q_{1_r}-q_{1_l})}{4k_d^2h^4}\left(\psi_{2_{uu}}+\psi_{2_{ur}}+\psi_{2_{ul}}+((\frac{k_d^2h^2}{2}-4)\psi_{2_u}-\psi_{2_{dd}}-\psi_{2_{dr}}-\psi_{2_{dl}}+(((4-\frac{k_d^2h^2}{2})\psi_{2_d}\right)
\end{aligned}
$$

This finally produces the entire Jacobian term.

$$
\begin{aligned}
J(\psi_1,q_1)&=\frac{\partial\left(\frac{2}{k_d^2}(q_2-\nabla^2\psi_2-\beta y)+\psi_2\right)}{\partial x}\frac{\partial q_1}{\partial y}-\frac{\partial\left(\frac{2}{k_d^2}(q_2-\nabla^2\psi_2-\beta y)+\psi_2\right)}{\partial y}\frac{\partial q_1}{\partial x}\\
&=\frac{\frac{2}{k_d^2}(q_{2_r}-q_{2_l})(q_{1_u}-q_{1_d})}{4h^2}-\frac{\frac{2}{k_d^2}(q_{2_u}-q_{2_d}-2\beta h)(q_{1_r}-q_{1_l})}{4h^2}-\\
&\frac{2(q_{1_u}-q_{1_d})}{4k_d^2h^4}\left(\psi_{2_{ur}}+\psi_{2_{dr}}+\psi_{2_{rr}}+((\frac{k_d^2h^2}{2}-4)\psi_{2_r}-\psi_{2_{ul}}-\psi_{2_{dl}}-\psi_{2_{ll}}+(((4-\frac{k_d^2h^2}{2})\psi_{2_l}\right)+\\
&\frac{2(q_{1_r}-q_{1_l})}{4k_d^2h^4}\left(\psi_{2_{uu}}+\psi_{2_{ur}}+\psi_{2_{ul}}+((\frac{k_d^2h^2}{2}-4)\psi_{2_u}-\psi_{2_{dd}}-\psi_{2_{dr}}-\psi_{2_{dl}}+(((4-\frac{k_d^2h^2}{2})\psi_{2_d}\right)\\
&=\frac{1}{2k_d^2h^2}\Bigg[(q_{2_r}-q_{2_l})(q_{1_u}-q_{1_d})-(q_{2_u}-q_{2_d}-2\beta h)(q_{1_r}-q_{1_l})-\\
&\frac{(q_{1_u}-q_{1_d})}{h^2}\left(\psi_{2_{ur}}+\psi_{2_{dr}}+\psi_{2_{rr}}+((\frac{k_d^2h^2}{2}-4)\psi_{2_r}-\psi_{2_{ul}}-\psi_{2_{dl}}-\psi_{2_{ll}}+(((4-\frac{k_d^2h^2}{2})\psi_{2_l}\right)+\\
&\frac{(q_{1_r}-q_{1_l})}{h^2}\left(\psi_{2_{uu}}+\psi_{2_{ur}}+\psi_{2_{ul}}+((\frac{k_d^2h^2}{2}-4)\psi_{2_u}-\psi_{2_{dd}}-\psi_{2_{dr}}-\psi_{2_{dl}}+(((4-\frac{k_d^2h^2}{2})\psi_{2_d}\right)\Bigg]
\end{aligned}
$$

Similarly, the below working discretises by replacing q_1 of the Jacobian with its corresponding equation.

$$
\begin{aligned}
J(\psi_1, q_1) &= J\left(\psi_1, \nabla^2\psi_1 + \beta y + \frac{k_d^2}{2}(\psi_2 - \psi_1)\right) \\
&= \frac{\partial\psi_1}{\partial x}\frac{\partial\left(\nabla^2\psi_1 + \beta y + \frac{k_d^2}{2}(\psi_2 - \psi_1)\right)}{\partial y} - \frac{\partial\psi_1}{\partial y}\frac{\partial\left(\nabla^2\psi_1 + \beta y + \frac{k_d^2}{2}(\psi_2 - \psi_1)\right)}{\partial x}
\end{aligned}
$$

The first term:

$$
\begin{aligned}
&\frac{\partial\psi_1}{\partial x}\frac{\partial\left(\nabla^2\psi_1 + \beta y + \frac{k_d^2}{2}(\psi_2 - \psi_1)\right)}{\partial y} \\
&= \frac{\psi_{1_r} - \psi_{1_l}}{2dx}\frac{\frac{\psi_{1_{uu}} + \psi_{1_{ur}} + \psi_{1_{ul}} - 4\psi_{1_u} - \psi_{1_{dd}} - \psi_{1_{dr}} - \psi_{1_{dl}} + 4\psi_{1_d}}{h^2} + 2\beta dy + \frac{k_d^2}{2}(\psi_{2_u} - \psi_{2_d} - \psi_{1_u} + \psi_{1_d})}{2dy} \\
&= \frac{\psi_{1_r} - \psi_{1_l}}{2dx}\frac{\frac{\psi_{1_{uu}} + \psi_{1_{ur}} + \psi_{1_{ul}} - 4\psi_{1_u} - \psi_{1_{dd}} - \psi_{1_{dr}} - \psi_{1_{dl}} + 4\psi_{1_d}}{h^2} + 2\beta dy + \frac{k_d^2}{2}(-\psi_{1_u} + \psi_{1_d})}{2dy} + \\
&\frac{k_d^2(\psi_{1_r} - \psi_{1_l})}{8dxdy}(\psi_{2_u} - \psi_{2_d})
\end{aligned}
$$

The second term:

$$
\begin{aligned}
&\frac{\partial\psi_1}{\partial y}\frac{\partial\left(\nabla^2\psi_1 + \beta y + \frac{k_d^2}{2}(\psi_2 - \psi_1)\right)}{\partial x} \\
&= \frac{\psi_{1_u} - \psi_{1_d}}{2dy}\frac{\frac{\psi_{1_{ur}} + \psi_{1_{dr}} + \psi_{1_{rr}} - 4\psi_{1_r} - \psi_{1_{ul}} - \psi_{1_{dl}} - \psi_{1_{ll}} + 4\psi_{1_l}}{h^2} + \frac{k_d^2}{2}(\psi_{2_r} - \psi_{2_l} - \psi_{1_r} + \psi_{1_l})}{2dx} \\
&= \frac{\psi_{1_u} - \psi_{1_d}}{2dy}\frac{\frac{\psi_{1_{ur}} + \psi_{1_{dr}} + \psi_{1_{rr}} - 4\psi_{1_r} - \psi_{1_{ul}} - \psi_{1_{dl}} - \psi_{1_{ll}} + 4\psi_{1_l}}{h^2} + \frac{k_d^2}{2}(-\psi_{1_r} + \psi_{1_l})}{2dx} + \\
&\frac{k_d^2(\psi_{1_u} - \psi_{1_d})}{8dxdy}(\psi_{2_r} - \psi_{2_l})
\end{aligned}
$$

Finally, the Jacobian term:

$$
\begin{aligned}
J(\psi_1, q_1) &= \frac{\partial\psi_1}{\partial x}\frac{\partial\left(\nabla^2\psi_1 + \beta y + \frac{k_d^2}{2}(\psi_2 - \psi_1)\right)}{\partial y} - \frac{\partial\psi_1}{\partial y}\frac{\partial\left(\nabla^2\psi_1 + \beta y + \frac{k_d^2}{2}(\psi_2 - \psi_1)\right)}{\partial x} \\
&= \frac{\psi_{1_r} - \psi_{1_l}}{2dx}\frac{\frac{\psi_{1_{uu}} + \psi_{1_{ur}} + \psi_{1_{ul}} - 4\psi_{1_u} - \psi_{1_{dd}} - \psi_{1_{dr}} - \psi_{1_{dl}} + 4\psi_{1_d}}{h^2} + 2\beta dy + \frac{k_d^2}{2}(-\psi_{1_u} + \psi_{1_d})}{2dy} - \\
&\frac{\psi_{1_u} - \psi_{1_d}}{2dy}\frac{\frac{\psi_{1_{ur}} + \psi_{1_{dr}} + \psi_{1_{rr}} - 4\psi_{1_r} - \psi_{1_{ul}} - \psi_{1_{dl}} - \psi_{1_{ll}} + 4\psi_{1_l}}{h^2} + \frac{k_d^2}{2}(-\psi_{1_r} + \psi_{1_l})}{2dx} + \\
&\frac{k_d^2(\psi_{1_r} - \psi_{1_l})}{8dxdy}(\psi_{2_u} - \psi_{2_d}) - \frac{k_d^2(\psi_{1_u} - \psi_{1_d})}{8dxdy}(\psi_{2_r} - \psi_{2_l})
\end{aligned}
$$

For both approaches, the constant term was taken out to be a part of corresponding A_0 or a_0 as they were independent of $Y = \psi_2$.

References

[1] Peter Bauer, Alan Thorpe, and Gilbert Brunet. The quiet revolution of numerical weather prediction. *Nature*, 525(7567):47–55, 2015.

[2] Jean-Paul Chehab and Maithem Trabelsi Moalla. Numerical simulations of a 2d quasi geostrophic equation, 2010.

[3] Nan Chen, Yingda Li, and Honghu Liu. Conditional gaussian nonlinear system: A fast preconditioner and a cheap surrogate model for complex nonlinear systems. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 32(5):053122, May 2022. Part of the Focus Issue on Theory-informed and Data-driven Approaches to Advance Climate Sciences.

[4] H.C. Davies and H. Wernli. Quasi-geostrophic theory. In James R. Holton, editor, *Encyclopedia of Atmospheric Sciences*, pages 1787–1794. Academic Press, Oxford, 2003.

[5] Quanling Deng, Nan Chen, Samuel Stechmann, and Jiuhua Hu. Lemda: A lagrangian-eulerian multiscale data assimilation framework, 2024.

[6] Geir Evensen. Data assimilation: The ensemble kalman filter. *Springer Science Business Media*, 2009.

[7] Daniel T. Gillespie. 3 - continuous markov processes. In Daniel T. Gillespie, editor, *Markov Processes*, pages 111–219. Academic Press, San Diego, 1992.

[8] S. Jamal. Solutions of quasi-geostrophic turbulence in multi-layered configurations. pages 207–216, Wits 2001, South Africa, 2010. axiv.

[9] Heiner Kornich, S Wahl, and S Basu. Dynamic downscaling of atmospheric fields for mesoscales and sub-mesoscales. *Journal of Geophysical Research: Atmospheres*, 113:D07108, 2008.

[10] J. N. Kutz. Amath 563 inferring structure of complex systems: Data assimilation. https://faculty.washington.edu/kutz/am563/page1/page6/am563.html, 2016.

[11] Andrew J Majda and John Harlim. A new perspective on multiscale geophysical data assimilation. *SIAM Review*, 54(2):377–418, 2012.

[12] HT. Tachim Medjo. Numerical simulations of a two-layer quasi-geostrophic equation of the ocean. In *SIAM Journal on Numerical Analysis*, 2000, vol 37, pages 2005–2022. Society for Industrial and Applied Mathematics, 2000.

[13] Chen Nan. *Stochastic Methods for Modeling and Predicting Complex Dynamical Systems*. Springer, 2023.

[14] Eiichi Oka. Nonlinear solutions to a two-layer quasi-geostrophic model of the gulf streamt, December 1989.

[15] Jaideep Pathak, Zhixin Lu, Brian Hunt, Michelle Girvan, and Edward Ott. Using machine learning to augment coarse-grid computational fluid dynamics simulations. *Journal of Computational Physics*, 375:973–991, 2018.

[16] Joseph Pedlosky. Geophysical fluid dynamics. *Springer Science Business Media*, 1987.

[17] M Roth, G Hendeby, and C et al Fritsche. The ensemble kalman filter: a signal processing perspective. EURASIP Journal on Advances in Signal Processing, 2017.

[18] Geoffrey K. Vallis. *Atmospheric and Oceanic Fluid Dynamics*. Second Edition. Cambridge University Press, 2017.

[19] Geun Yun. Quasi_geostrophic_model_for_lemda. https://github.com/geun-yun/Quasi_geostrophic_model_for_LEMDA/tree/main, 2024.