

A Two-Dimensional Incompressible Navier-Stokes Model for Data Assimilation in DAPPER

Nabeel Chasmawala, Avon High School
with Dr. Chang Liu, University of Connecticut

October 2025

1 Introduction

The dynamics of turbulent fluid regimes have been challenging to fully understand and simulate over time due to their chaotic behaviors and computational complexity. A small difference in initial conditions will diverge quickly in chaotic systems, leading to qualitatively different behaviors. As a result, a better estimation of the initial condition through data assimilation is essential for understanding chaotic dynamics, e.g., in weather forecast applications. The general premise of modern data assimilation involves multiple simulations with slightly different initial conditions, and they will begin to diverge as a result of the chaotic property of the underlying equations. After a few time steps, the data assimilation algorithm will prune the collection of models around observed measurements of the system; those models that are closer to the observation stay, while those that have diverged are eliminated. The observed measurements are not blindly trusted, however; the measurement instruments in real situations introduce some noise and this is accounted for in the data assimilation method. This process continues across many time steps and allows a reasonable estimate of the chaotic system to persist over larger scales in time in a way that a traditional singular model would not [1]. The DAPPER [2] library is built for the purpose of researching data assimilation; however, it lacks a model for the Navier-Stokes equations. Therefore, this work adds two-dimensional Navier-Stokes equations (NSE) to DAPPER to test the effectiveness of data assimilation on fluid dynamics. While the initial test cases were laminar regimes with some artificial noise added (less chaotic), the accuracy of the data assimilation systems suggests potential success with more chaotic systems as well.

2 Methods

To implement the Navier-Stokes equations into DAPPER, the first step involves creating a model that conducts numerical simulations of the NSE. The variable of interest is the stream function ψ because the incompressibility condition

that enforces the conservation of mass ($\nabla \cdot \mathbf{u} = 0$) is guaranteed by using the stream function. To advance the stream function state in time, we use the Fourier spectral method in spatial directions and use the ETD-RK4 scheme [3] to advance the system over time.

2.1 2D incompressible flow in stream function formulation

The 2D stream function form of the Navier-Stokes equations is used in the model so as not to have to repeatedly calculate pressure. The stream function in 2D is described as:

$$u = -\frac{\partial \psi}{\partial y}, \quad (1)$$

and

$$v = \frac{\partial \psi}{\partial x}. \quad (2)$$

The incompressibility condition ($\nabla \cdot \mathbf{u} = 0$) is therefore inherently satisfied by using this form. The 2D stream function form is:

$$\frac{\partial}{\partial t}(\nabla^2 \psi) + \frac{\partial \psi}{\partial x} \frac{\partial}{\partial y}(\nabla^2 \psi) - \frac{\partial \psi}{\partial y} \frac{\partial}{\partial x}(\nabla^2 \psi) = \nu \nabla^4 \psi. \quad (3)$$

This can also be written more concisely using the Jacobian determinant J :

$$\frac{\partial}{\partial t}(\nabla^2 \psi) + J(\psi, \nabla^2 \psi) = \nu \nabla^4 \psi, \quad (4)$$

where

$$J(\psi, \nabla^2 \psi) := \det \begin{bmatrix} \frac{\partial \psi}{\partial x} & \frac{\partial \psi}{\partial y} \\ \frac{\partial(\nabla^2 \psi)}{\partial x} & \frac{\partial(\nabla^2 \psi)}{\partial y} \end{bmatrix} = \frac{\partial \psi}{\partial x} \frac{\partial}{\partial y}(\nabla^2 \psi) - \frac{\partial \psi}{\partial y} \frac{\partial}{\partial x}(\nabla^2 \psi). \quad (5)$$

2.2 Fourier Spectral Analysis

The numerical method used to discretize equation (3) is the same as the one used by the Kuramoto-Sivashinsky (KS) model within DAPPER [2]. By taking the Fourier transform of the system, higher-order partial derivatives become multiplications. The method in the Kuramoto-Sivashinsky model is extended into two spatial dimensions in this model. The Fourier transform in two dimensions for a function $f(x, y)$ is defined as

$$\hat{f}(k_x, k_y) = \mathcal{F}[f(x, y)] = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-i(k_x x + k_y y)} dx dy, \quad (6)$$

where $i = \sqrt{-1}$ is the imaginary unit.

Fourier transforms have a property where a differentiation with respect to a variable in the domain of the transformed function (for example, x) is equivalent to a multiplication by $\text{i}k_x$:

$$\mathcal{F}\left[\frac{\partial f}{\partial x}\right] = \text{i}k_x \mathcal{F}[f]. \quad (7)$$

A similar relation also holds for the Fourier transform in y :

$$\mathcal{F}\left[\frac{\partial f}{\partial y}\right] = \text{i}k_y \mathcal{F}[f]. \quad (8)$$

We leverage this property to transform the PDE into an ODE by taking the Fourier transform (\mathcal{F}) in both spatial dimensions of both sides of equation (4):

$$-(k_x^2 + k_y^2) \frac{\partial \widehat{\psi}}{\partial t} + \widehat{J} = \nu(k_x^2 + k_y^2)^2 \widehat{\psi}, \quad (9)$$

where k_x and k_y represent the wavenumbers in the x and y dimensions, respectively, and $\widehat{J} := \mathcal{F}\left(\frac{\partial(\psi, \nabla^2 \psi)}{\partial(x, y)}\right)$. Rearranging for the time derivative gives:

$$\frac{\partial \widehat{\psi}}{\partial t} = \frac{\widehat{J}}{k_x^2 + k_y^2} - \nu(k_x^2 + k_y^2)^2 \widehat{\psi}. \quad (10)$$

Note that in the code, the system advances $\widehat{\nabla^2 \psi}$ in time instead of $\widehat{\psi}$ and the $-(k_x^2 + k_y^2)$ term is divided out at the end, but both are mathematically equivalent.

2.3 Time Integration Scheme

Time integration methods used in the KS model are again repurposed for the Navier-Stokes solver. I used the ETDRK4 scheme initially described by Cox and Matthews [4], then modified it by adding the contour integral method used by Kassam and Trefethen [3] in order to reduce numerical instability. As with all the ODEs that use ETDRK4, equation (10) can be split into two parts:

$$\widehat{\psi}_t = \mathbf{L}\widehat{\psi} + \mathbf{N}[\widehat{\psi}], \text{ where} \quad (11)$$

$$\mathbf{L} = -\nu(k_x^2 + k_y^2), \text{ and} \quad (12)$$

$$\mathbf{N}[\widehat{\psi}] = \frac{\widehat{J}}{k_x^2 + k_y^2}. \quad (13)$$

The linear part is exactly calculated and the non linear part is numerically approximated.

2.4 The Nonlinear Operator

Nonlinear operators tend to be less trivial to transform to frequency space because multiplications in the real space become convolutions (*) in frequency space:

$$\mathcal{F}[f(x, y)g(x, y)] = \hat{f}(k_x, k_y) * \hat{g}(k_x, k_y) \quad (14)$$

$$= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \hat{f}(k'_x, k'_y) \hat{g}(k_x - k'_x, k_y - k'_y) dk'_x dk'_y. \quad (15)$$

The computation of the nonlinear term $-u \frac{\partial u}{\partial x}$ in the KS equations bypasses the convolutions as it can be manipulated to remove the product:

$$-u \frac{\partial u}{\partial x} = -\frac{1}{2} \frac{\partial(u^2)}{\partial x}, \quad (16)$$

and taking the Fourier transform gives

$$\mathcal{F}\left[-\frac{1}{2} \frac{\partial(u^2)}{\partial x}\right] = -\frac{i k_x}{2} \mathcal{F}[u^2]. \quad (17)$$

However, the nonlinear operator of the NSE cannot be treated the same way. To avoid the convolutions, the derivatives are multiplied in real space. To prevent unresolvable higher frequencies in the factors aliasing into the resolved lower frequencies of the products, we only use the middle 2/3 of the frequencies of each factor, as this is a quadratic nonlinearity. The approach zeroes out frequencies outside of the middle 2/3 so that they do not alias themselves in the product [5].

3 Results and Next Steps

Once a prototype was created, the next step was to verify its numerical accuracy before implementing it into DAPPER. I tested the solver by comparing our numerical solutions to 2D Navier-Stokes equations with existing analytical solutions.

3.1 Validating Solver Accuracy

The Taylor-Green vortex [6] is used as the primary test case, because an analytical solution describing its stream function has already been derived for a domain size $x \in [0, 2\pi]$ and $y \in [0, 2\pi]$:

$$\psi(x, y, t) = e^{-2\nu t} \sin x \sin y. \quad (18)$$

In the validation example, we set kinematic viscosity $\nu = 0.01$.

The normalized root mean square error (NRMSE) is computed as the quadratic mean of the residuals between the analytical and numerical solutions for each

grid point in space divided by the root mean square of the analytical solution: (let $\psi_a(x, y, t)$ and $\psi_n(x, y, t)$ equal the analytically and numerically computed values of stream function, respectively):

$$NRMSE[\psi_n(t) - \psi_a(t)] = \sqrt{\frac{\sum_{i=1}^{N_x} \sum_{j=1}^{N_y} [\psi_n(i\Delta x, j\Delta y, t) - \psi_a(i\Delta x, j\Delta y, t)]^2}{\sum_{i=1}^{N_x} \sum_{j=1}^{N_y} [\psi_a(i\Delta x, j\Delta y, t)]^2}}. \quad (19)$$

The computed NRMSE between the numerically derived solution and the analytical solution is plotted over time in Figure 1, demonstrating that NRMSE will decrease to 10^{-12} at $t = 100$.

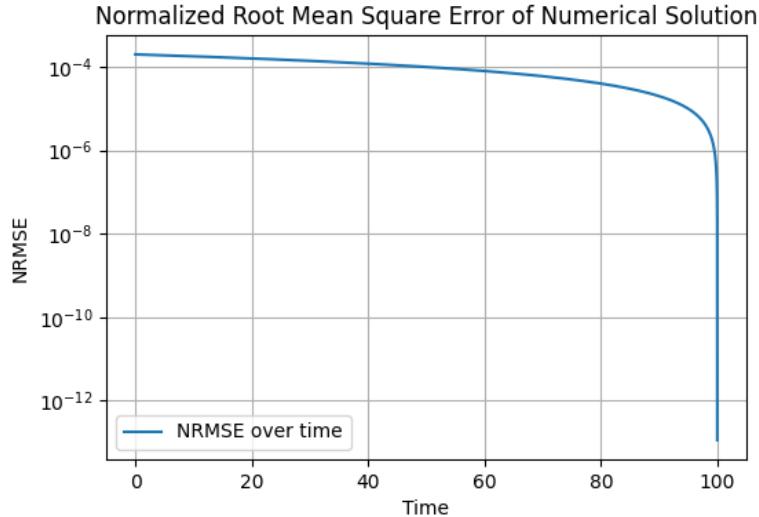


Figure 1: The $NRMSE_\psi(t)$ between our numerical solver and analytical solution of the Taylor-Green vortex.

Some side-by-side results of the numerical versus analytical solutions at $t = 50$ and $t = 100$ are shown in Figures 2-3. Note that the viscosity dissipates stream function differences over time. A sinusoidal spatial variation is noticeable at $t = 50$ in Figure 2, but these spatial variations are not visible at $t = 100$ shown in Figure 3.

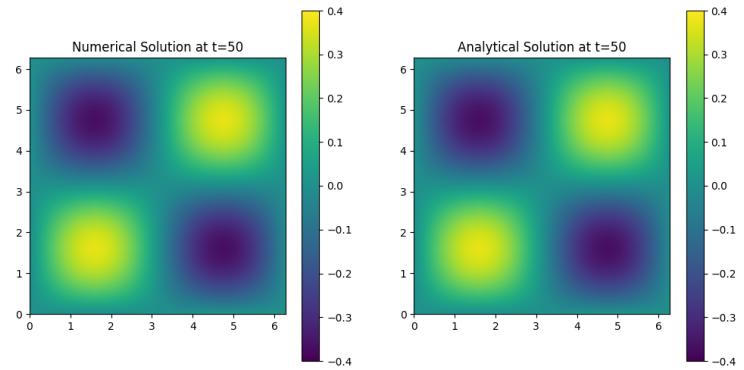


Figure 2: A comparison between $\psi(x, y)$ obtained from the numerical solver and analytical solutions of the Taylor-Green vortex at $t = 50$.

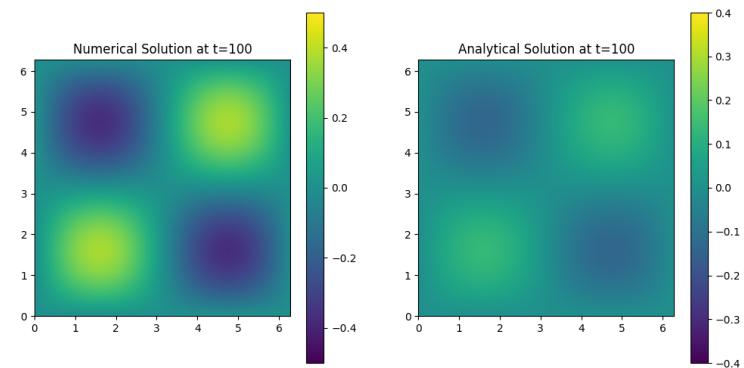


Figure 3: A comparison between $\psi(x, y)$ obtained from the numerical solver and analytical solutions of the Taylor-Green vortex at $t = 100$.

3.2 Implementation in DAPPER + Sample Test Results

Once I had the prototype NSE model operational, I then implemented it into the DAPPER library. I did this via making a model file (`dapper/mods/NS2D/__init__.py`) that contained the code detailing how to advance a fluid state in time, as well as an experiment file (`dapper/mods/NS2D/example_paper.py`) that set parameters regarding the twin-experiment; i.e. parameters regarding the generation of the dynamic truth and the observations of the experiment. Per DAPPER convention, experiment configurations should be made to reconstruct literature results, but as I was unable to find any cases of intersection between 2D Taylor-Green vortices or decaying Kolmogorov flows and data assimilation, I simply reused most of the configuration of Ref. [7] as found in the KS model [2, 7].

The model is tested with the EnKF with varying parameters. 10 ensemble members are run in parallel and use the numerical solver to simulate varying initial conditions. They then resample the true ψ value and the ensemble members are pruned to keep them from diverging too far away from this value. Two different double-periodic initial conditions are also used. The first is the Taylor-Green vortex, which is the same initial condition that was used in the validation process in equation (18):

$$\psi(x, y) = \sin x \sin y. \quad (20)$$

The second example uses the initial condition of Kolmogorov flow. This regime is turbulent with an external force, but without it decays laminarly. Its initial condition is simply

$$\psi(x, y) = -\frac{1}{2} \cos 2y. \quad (21)$$

That stream function is derived from defining the velocity in the x direction as $u = \sin 2y$.

The kinematic viscosity of the fluid ν controls how fast the fluid's energy decays, which is set as $\nu = 0.01$. I use a domain size $L_x \times L_y$ of $2\pi \times 2\pi$ for all experiments. The grid points $N_x \times N_y$ represent the spatial resolution of the solver. I use a 64×64 grid setup for all experiments. In figures 4 -7, mean ψ (μ_ψ) is the ensemble average of streamfunction ψ , and spread ψ (σ_ψ) refers to the standard deviation of streamfunction across the ensemble. True ψ is the computed value of the system using the numerical model, and err ψ is the difference between the ensemble mean and the computed truth ($\mu_\psi - \psi$). The following figures 4-7 show the Taylor-Green vortex and the decaying Kolmogorov flow with 0.5% noise as estimated by the ENKF (Sqrt, $N = 5$, infl = 1.02, rot = true) compared to the observed values.

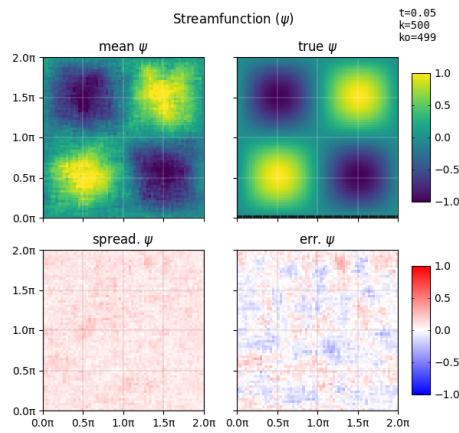


Figure 4: $t=0.05$ for the TG vortex. Top left is the EnKF estimate mean (μ_ψ); top right is the system's true value (ψ); bottom left is the EnKF estimate spread (σ_ψ); bottom right is the error between EnKF estimate and truth ($\mu_\psi - \psi$). k and k_o are indices for truth and observation values (top right and top left); $k = T/dt$ and $k_o = k - 1$ by DAPPER convention.

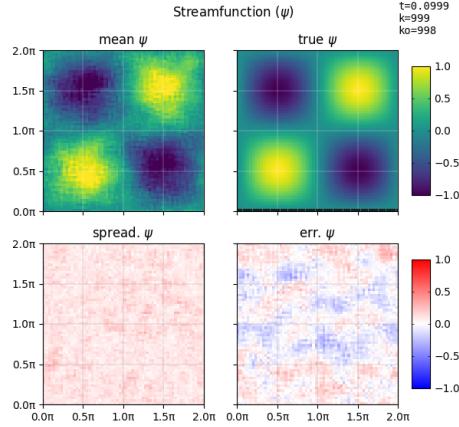


Figure 5: $t = 0.1$ for the TG vortex. Top left is the EnKF estimate mean (μ_ψ); top right is the system's true value (ψ); bottom left is the EnKF estimate spread (σ_ψ); bottom right is the error between EnKF estimate and truth ($\mu_\psi - \psi$). k and k_o are indices for truth and observation values (top right and top left); $k = T/dt$ and $k_o = k - 1$ by DAPPER convention.

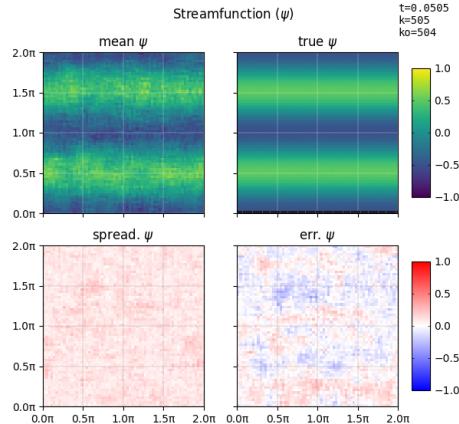


Figure 6: $t = 0.05$ for the decaying Kolmogorov system. Top left is the EnKF estimate mean (μ_ψ); top right is the system's true value (ψ); bottom left is the EnKF estimate spread (σ_ψ); bottom right is the error between EnKF estimate and truth ($\mu_\psi - \psi$). k and k_o are indices for truth and observation values (top right and top left); $k = T/dt$ and $k_o = k - 1$ by DAPPER convention.

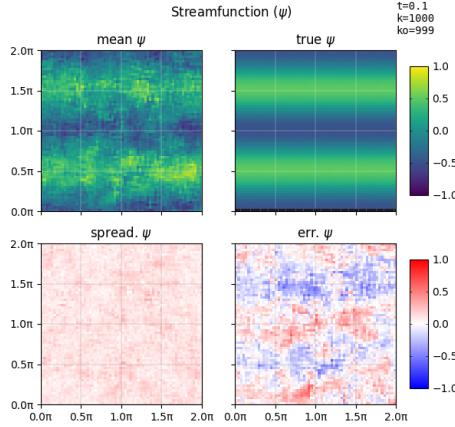


Figure 7: $t = 0.1$ for the decaying Kolmogorov system. Top left is the EnKF estimate mean (μ_ψ); top right is the system’s true value (ψ); bottom left is the EnKF estimate spread (σ_ψ); bottom right is the error between EnKF estimate and truth ($\mu_\psi - \psi$). k and k_o are indices for truth and observation values (top right and top left); $k = T/dt$ and $k_o = k - 1$ by DAPPER convention.

3.3 Overall DA Results

The preliminary tests I ran tested the effect of the analysis update method and the ensemble size N . The analysis update methods represent the different methods used by the EnKF to factor in observed values to the ensemble. The two tested were the typical perturbed observation scheme, in which noise is added to the observations when updating the ensemble, and the square root scheme, in which the ensemble is updated deterministically through a square-root operation.

One other constraint of the experiments was the total run time. Compared to other models, the experiments on the NSE were minuscule. For example, the time scale of the configuration in Ref. [7] was on the order of 10^3 compared to the 10^{-1} of the NSE setup. The model, when used by the assimilation methods, becomes unstable for timescales longer than this, so as of now it is not possible to run longer experiments. Additionally, the timestep must be extremely low to keep the model stable at all; the following tests use $dt = 10^{-4}$ (significantly smaller than $dt = 0.5$ in [7]). The results of the experiment are below:

| Taylor Green Vortex | | | | | |
|----------------------------|--------|-------------------|-------------------|----------------|---------------|
| Analysis Method | Update | N | RMSE $\pm\sigma$ | SD $\pm\sigma$ | runtime (sec) |
| Sqrt | 2 | 0.060 \pm 0.006 | 0.07 \pm 0.02 | 14 | |
| Sqrt | 5 | 0.125 \pm 0.004 | 0.121 \pm 0.002 | 18 | |
| Sqrt | 10 | 0.297 \pm 0.007 | 0.19 \pm 0.01 | 27 | |
| Sqrt | 20 | 0.52 \pm 0.02 | 0.41 \pm 0.03 | 53 | |
| Sqrt | 50 | 3 \pm 1 | 1.8 \pm 0.3 | 162 | |
| PertObs | 2 | 0.034 \pm 0.005 | 0.024 \pm 0.002 | 17 | |
| PertObs | 5 | 0.073 \pm 0.003 | 0.075 \pm 0.002 | 19 | |
| PertObs | 10 | 0.141 \pm 0.005 | 0.129 \pm 0.002 | 29 | |
| PertObs | 20 | 0.38 \pm 0.02 | 0.238 \pm 0.004 | 54 | |
| PertObs | 50 | 1.7 \pm 0.4 | 1.0 \pm 0.2 | 160 | |
| Non-forced Kolmogorov Flow | | | | | |
| Analysis Method | Update | N | RMSE $\pm\sigma$ | SD $\pm\sigma$ | runtime (sec) |
| Sqrt | 2 | 0.053 \pm 0.004 | 0.062 \pm 0.004 | 18 | |
| Sqrt | 5 | 0.132 \pm 0.005 | 0.115 \pm 0.003 | 23 | |
| Sqrt | 10 | 0.216 \pm 0.005 | 0.197 \pm 0.002 | 33 | |
| Sqrt | 20 | 0.6 \pm 0.02 | 0.38 \pm 0.02 | 55 | |
| Sqrt | 50 | 3.6 \pm 0.7 | 1.7 \pm 0.2 | 148 | |
| PertObs | 2 | 0.06 \pm 0.03 | 0.022 \pm 0.004 | 13 | |
| PertObs | 5 | 0.076 \pm 0.003 | 0.071 \pm 0.004 | 18 | |
| PertObs | 10 | 0.234 \pm 0.004 | 0.138 \pm 0.004 | 27 | |
| PertObs | 20 | 0.36 \pm 0.02 | 0.242 \pm 0.004 | 52 | |
| PertObs | 50 | 1.5 \pm 0.4 | 1.0 \pm 0.2 | 156 | |

Table 1: The results of the experiment from $t = 0$ to $t = 0.1$ with N as the ensemble size. The RMSE and SD are the mean and standard deviation of the error of the estimated stream function through space and time, and their respective $\pm\sigma$ represent the spread of error across the ensemble. That is, the RMSE value shows the average stream function error in space and time of the system, but that average in time could vary by $\pm\sigma_{RMSE}$ across the different ensemble members. Likewise, the SD shows how much the error varied through space and time, but that variance also varied across different ensemble members by $\pm\sigma_{SD}$.

Interestingly, increasing N for the Ensemble Kalman filter has the opposite of the expected effect: accuracy worsens rather than improves.

3.4 Using the Model

To see the demo of the numerical solver as in Figures 2 and 3, run `demo.py` after setting the desired initial condition in `__init__.py`:
`(__init__.py lines 55-56):`

```
#initial conditions for a Taylor-Green vortex
psi = np.sin(X) * np.sin(Y)
```

Command line:

```
python3 path/to/DAPPER/dapper/mods/NS2D/demo.py
```

To run data assimilation with the model, use the model to create a HMM object which can then be passed into DA methods like the EnKF. The simplest way to do this is using the `basic_1.py` example; replace

```
from dapper.mods.Lorenz63.sakov2012 import HMM
```

with

```
from dapper.mods.NS2D.example_paper import HMM
```

Make sure to keep the experiment time no greater than 0.1 to avoid instability; delete/comment this line as it would ironically lengthen the experiment instead:

```
HMM.tseq.T = 30 # shorten experiment
```

For the specific EnKF configuration I used to generate Figures 4, 5, 6, and 7, set `xp` to:

```
xp = da.EnKF("Sqrt", N=5, infl=1.02, rot=True)
```

Running `basic_1.py` will generate the same experiment as in the figures.

3.5 Next Steps

From here, adding a body force into the solver would allow us to conduct data assimilation in turbulent regimes rather than only slightly noisy laminar regimes. In addition, exploring ways to enhance the numerical stability of the solver would be beneficial in order to decrease the kinematic viscosity ν and increase the timestep dt . Lastly, testing this system with other DA methods to see if there are ways to increase its compatibility would also be an improvement. As of now, the LETKF typically experiences infinite value errors for some of the same parameters that work with the EnKF; allowing for more DA methods to work would increase the possibilities to conduct data assimilation of turbulent regimes.

4 Personal Reflection and Acknowledgments

I came across the DAPPER library while doing research with Dr. Liu in the summer of 2025. We were interested in comparing how different data assimilation methods compared with each other for different models in terms of their accuracy and performance, and this library seemed to be the right tool to do so. I wanted to extend this research further and noticed that a fluid dynamics model for non-atmospheric cases was missing. I therefore decided to implement the incompressible Navier-Stokes in 2D to see if we could try data assimilation with it.

I would like to thank Dr. Liu for all of the support and guidance he has given me throughout this project, from resources to explanations and more. Without his insightful suggestions, advice, and contributions, this project would not have taken place.

References

- [1] Detlef Stammer et al. “Ocean data assimilation in support of climate applications: status and perspectives”. In: *Annual review of marine science* 8.1 (2016), pp. 491–518.
- [2] Patrick N Raanes, Yumeng Chen, and Colin Grudzien. “DAPPER: data assimilation with Python: a package for experimental research”. In: *Journal of Open Source Software* 9.94 (2024), p. 5150.
- [3] Aly-Khan Kassam and Lloyd N Trefethen. “Fourth-order time-stepping for stiff PDEs”. In: *SIAM Journal on Scientific Computing* 26.4 (2005), pp. 1214–1233.
- [4] Steven M Cox and Paul C Matthews. “Exponential time differencing for stiff systems”. In: *Journal of Computational Physics* 176.2 (2002), pp. 430–455.
- [5] Steven A Orszag. “On the elimination of aliasing in finite-difference schemes by filtering high-wavenumber components”. In: *Journal of Atmospheric Sciences* 28.6 (1971), pp. 1074–1074.
- [6] Geoffrey Ingram Taylor and Albert Edward Green. “Mechanism of the production of small eddies from large ones”. In: *Proceedings of the Royal Society of London. Series A-Mathematical and Physical Sciences* 158.895 (1937), pp. 499–521.
- [7] Marc Bocquet and Alban Farchi. “On the consistency of the local ensemble square root Kalman filter perturbation update”. In: *Tellus A: Dynamic Meteorology and Oceanography* 71.1 (2019), p. 1613142.