

# Solutions to the Project for Chapter 5

## Data Assimilation Experiments with the Lorenz-63 model

Last updated: 13:08, October 7, 2021

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Lorenz-63 model . . . . .	2
1.2	DA system L63-DAS for the Lorenz-63 model . . . . .	3
1.2.1	DA methods included in the L63-DAS . . . . .	3
1.2.2	General procedure for OSSE . . . . .	3
1.3	A short code guide to L63-DAS . . . . .	4
1.3.1	Install the L63-DAS on <i>deepthought2</i> . . . . .	4
1.3.2	General code structures . . . . .	4
1.3.3	Important output files . . . . .	4
1.3.4	Useful Utilities . . . . .	5
1.4	Description of this project . . . . .	5
<b>2</b>	<b>Part A: Warm-up</b>	<b>5</b>
2.1	Tasks . . . . .	5
<b>3</b>	<b>Part B: Variational method</b>	<b>6</b>
3.1	Related codes . . . . .	6
3.2	Tasks . . . . .	7
<b>4</b>	<b>Part C: Ensemble method</b>	<b>19</b>
4.1	Related codes . . . . .	19
4.2	Tasks . . . . .	19
<b>5</b>	<b>Part D: Hybrid method</b>	<b>30</b>
5.1	Tasks . . . . .	30

# 1 Introduction

In Chapter 5, we explored different data assimilation (DA) methods theoretically. This project is intended to give you hands-on experience about their code implementation and get you familiar with different DA methods' behaviors. We use the famous 3-variable Lorenz-63 model as our toy numerical weather model for its simplicity and its similar chaotic behaviors as the real NWP models.

## 1.1 Lorenz-63 model

Lorenz (1963) developed a simplified 3-variable model for the atmospheric convection. The equations for the Lorenz-63 model are

$$\begin{cases} \dot{x} = \sigma(y - x) \\ \dot{y} = x(\rho - z) - y \\ \dot{z} = xy - \beta z \end{cases} \quad (1)$$

where  $\sigma = 10$ ,  $\beta = 8/3$ ,  $\rho = 28$ . We use the Runge-Kutta 4-stage method to discretize the forward Lorenz-63 model. After getting the forward model, we developed its tangent linear and adjoint model following the instructions in Chapter 5. Figure 1 shows the trajectory generated by the Lorenz-63 model.

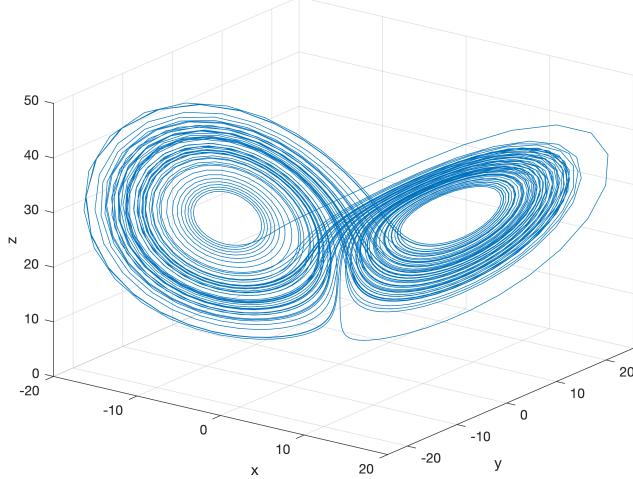


Figure 1: The trajectory of the Lorenz-63 model with parameters  $\sigma = 10$ ,  $\beta = 8/3$ ,  $\rho = 28$ .

## 1.2 DA system L63-DAS for the Lorenz-63 model

### 1.2.1 DA methods included in the L63-DAS

L63-DAS is a data assimilation package for the Lorenz-63 model. It includes the following DA methods:

- 3D-Var
- 4D-Var with multi-outerloops
- Perturbed Observation EnKF
- (Local) Ensemble Transform Kalman Filter (LETKF)

All the mentioned methods are already available in the package, and you can directly compare their performance through observing system simulation experiments (OSSE).

### 1.2.2 General procedure for OSSE

Here we briefly introduce the general procedure for OSSE. One advantage of the OSSE is that we have truth (i.e., the true trajectory of the  $x, y, z$  for our Lorenz-63 model), which are not available in our real daily applications. Access to the true trajectory (i.e.,  $x^t, y^t, z^t$ ) enables us to directly compare the accuracy of the analysis (i.e.,  $x^a, y^a, z^a$ ) generated by different DA methods.

In L63-DAS, an OSSE includes the following steps:

1. Generate the true trajectory  $\mathbf{x}^t$  from the Lorenz-63 model  $\mathcal{M}$ .
2. Create the observations  $\mathbf{y}^o$  by adding Gaussian noise  $\epsilon$  to the simulated quantity from the true trajectory:  $\mathbf{y}^o = \mathcal{H}(\mathbf{x}^t) + \epsilon$ , where  $\epsilon \sim \mathcal{N}(0, r^2)$ .
3. After setting an initial condition  $\mathbf{x}_0^b$ , integrate the Lorenz-63 model from  $\mathbf{x}_0^b$  to get the background  $\mathbf{x}_1^b = \mathcal{M}(\mathbf{x}_0^b)$  until the first DA step. Assimilate available observations  $\mathbf{y}_1^o$  at this time step and update the background to the analysis  $\mathbf{x}_1^a$ . This analysis will serve the initial condition to get the background for the next DA step:  $\mathbf{x}_2^b = \mathcal{M}(\mathbf{x}_1^a)$ .
4. Repeat Step 3 for the remaining steps.
5. Finally, you get the whole trajectory of background  $\mathbf{x}^b$  and analysis  $\mathbf{x}^a$ . You can compare their accuracy with the truth  $\mathbf{x}^t$  to assess each DA method's performance.

One metric to evaluate the performance of assimilation method is the root mean square error (RMSE). For each assimilation cycle, we have both truth (i.e.,  $x^t, y^t, z^t$ ) and analysis (i.e.,  $x^a, y^a, z^a$ ). The total RMSE of the analysis at this cycle is defined as

$$RMSE = \sqrt{\frac{1}{3}[(x^t - x^a)^2 + (y^t - y^a)^2 + (z^t - z^a)^2]} \quad (2)$$

## 1.3 A short code guide to L63-DAS

### 1.3.1 Install the L63-DAS on *deepthought2*

1. Log into *deepthought2* through the command

```
ssh -X -Y your_user_name@rhel8.deepthought2.umd.edu
```

2. Load environment by running the following commands

```
module load intel
```

```
module load matlab (if you want to use Matlab)
```

3. Go to your experiment directory:

```
cd /lustre/your_user_name
```

4. Copy the L63-DAS package to your current directory:

```
cp -r /lustre/aosc614-1vb7/L63-DAS .
```

5. After entering directory L63-DAS, generate executables:

```
bash -xe compile_all.bsh
```

After running this command, you will see `fwd.exe`, `3dvar.exe`, `inc4dvar.exe` and `enda.exe` in your current directory.

### 1.3.2 General code structures

The naming convention is:

- `test_*.f90`: main programs for generating executables `*.exe`.
- `mod_*.f90`: modules contain different DA methods and support libraries.
- `compile_*.bsh`: compiling scripts for different executables `*.exe`.

The numerical Lorenz-63 model is in `mod_lorenz63_fwd.f90` while each data method is built independently as a module in the file `mod_lorenz63_[DA_short_name].f90`.

### 1.3.3 Important output files

Each DA subsystem (`3dvar.exe`, `inc4dvar.exe`, `enda.exe`) will output its experiment results in several files:

- `fort.10020`: truth of  $x, y, z$  for each DA cycle (Each line represents results from one cycle).
- `fort.10030`: background of  $x, y, z$  for each DA cycle.
- `fort.10040`: analysis of  $x, y, z$  for each DA cycle.
- `fort.10010`: observation of  $x, y, z$  for each DA cycle.

Since all executables will generate their outputs with the same name, you might want to rename those files immediately after finishing each experiment. You can use the script `rename_output.bsh` for easy renaming. Its usage is introduced in the next section.

### 1.3.4 Useful Utilities

- `rename_output.bsh`: script for renaming experiment output.

After running each DA method (those executables ended with `.exe`), you may want to rename the output for easier post-processing. In this case, you can utilize the script `rename_output.bsh`. The command is:

```
bash rename_output.bsh [filename]
```

For example, you may want to run the command:

```
bash rename_output.bsh letkf_m3
```

after you get the results from 3-member LETKF. Now all your experiment results are in the files `letkf_m3.100*0`.

- `rmse_single_method.m`: MATLAB script for a quick check of the time series of the truth, background, analysis trajectory, and analysis errors.

## 1.4 Description of this project

This project has three goals. Using Lorenz-63 model as an example, we will explore:

- why we need to do DA for NWP applications (Part B)
- how variational methods work (Part B)
- how ensemble Kalman filters work (Part C)
- how hybrid methods work (Part D)

## 2 Part A: Warm-up

In this part, we will try to install L63-DAS on *deepthought2*, and then run the Lorenz-63 forward model and generate a trajectory similar to the one in Figure 1.

### 2.1 Tasks

1. Follow the instructions in Section 1.3 and install L63-DAS under your directory.
2. Now run the Lorenz-63 forward model to generate a random trajectory.  
(run the command `./fwd.exe`. It then generates outputs into the file `fort.10020`)
3. Rename the output as `fwd.10020` with the script `rename_output.bsh`.  
(run the command `bash rename_output.bsh fwd`)
4. Read the file `fwd.10020` and plot the trajectory. Are you able to get something similar to Figure 1?

## 3 Part B: Variational method

From this part, we will start to conduct assimilation experiments: We will run OSSEs for different assimilation methods with the L63-DAS. Let's first explore variational methods.

### 3.1 Related codes

You will need to use `3dvar.exe` and `inc4dvar.exe` in Part B. These two executables receive no interactive inputs, which means you need to modify the source code if you want to change parameters. After you made modification to the source code, **remember to re-compile the code to generate new executable for your modifications to take into effect**. You can compile the source codes through utility scripts named `compile_*.bsh`. For these two executables, their main programs, major modules, and stand-alone compiling utilities are:

- `3dvar.exe` (lean 3D-Var):

```
test_3dvar.f90 mod_lorenz63_3dvar.f90 compile_3dvar.bsh
```

- `inc4dvar.exe` (incremental 4D-Var with multi-outerloops):

```
test_inc4dvar.f90 mod_lorenz63_inc4dvar.f90 compile_inc4dvar.bsh
```

For all the methods in this part, they use a static background error covariance **B**. This matrix is stored in the file `fort.1040`.

You can also use the plotting script `rmse_single_method.m` to have a quick check of the time series of the true/background/analysis trajectory and analysis errors.

## 3.2 Tasks

1. (Necessity of DA) Like the real weather, the Lorenz-63 system is a chaotic system, which means even small discrepancy at the initial time will lead to completely different trajectories at the later time. We will first explore why we need DA. Now let's see what will happen if we stop doing assimilation (Though we are using 3D-Var as an example, it applies to all DA methods):
  - (a) Perform OSSE with 3D-Var for every cycle and then overlay the truth and analysis trajectory with different colors in a separate panel for each variable  $x, y, z$ . Finally, plot the total analysis RMSE for each cycle in another panel. From this figure, can the analysis trajectory always be near the truth?  
*(hints:* Without modifying anything in `test_3dvar.f90`, run `3dvar.exe` and its outputs are in the files `fort.100*0`. Rename them with `rename_output.bsh`. Then plot the truth and analysis trajectories. You can use `rmse_single_method.m` for a quick look.)

**Ans:**

Results are shown in Figure 2. Based on (a)-(c), the analysis are almost completely overlapped with the truth trajectory, so we know the analysis of all model states are near the truth. This indicates our data assimilation method works correctly.

In (d), we observe:

- i. Generally the analysis RMSE is smaller than the background RMSE;
- ii. Since we have direct observations for model states, the analysis error is generally smaller than the observation error.

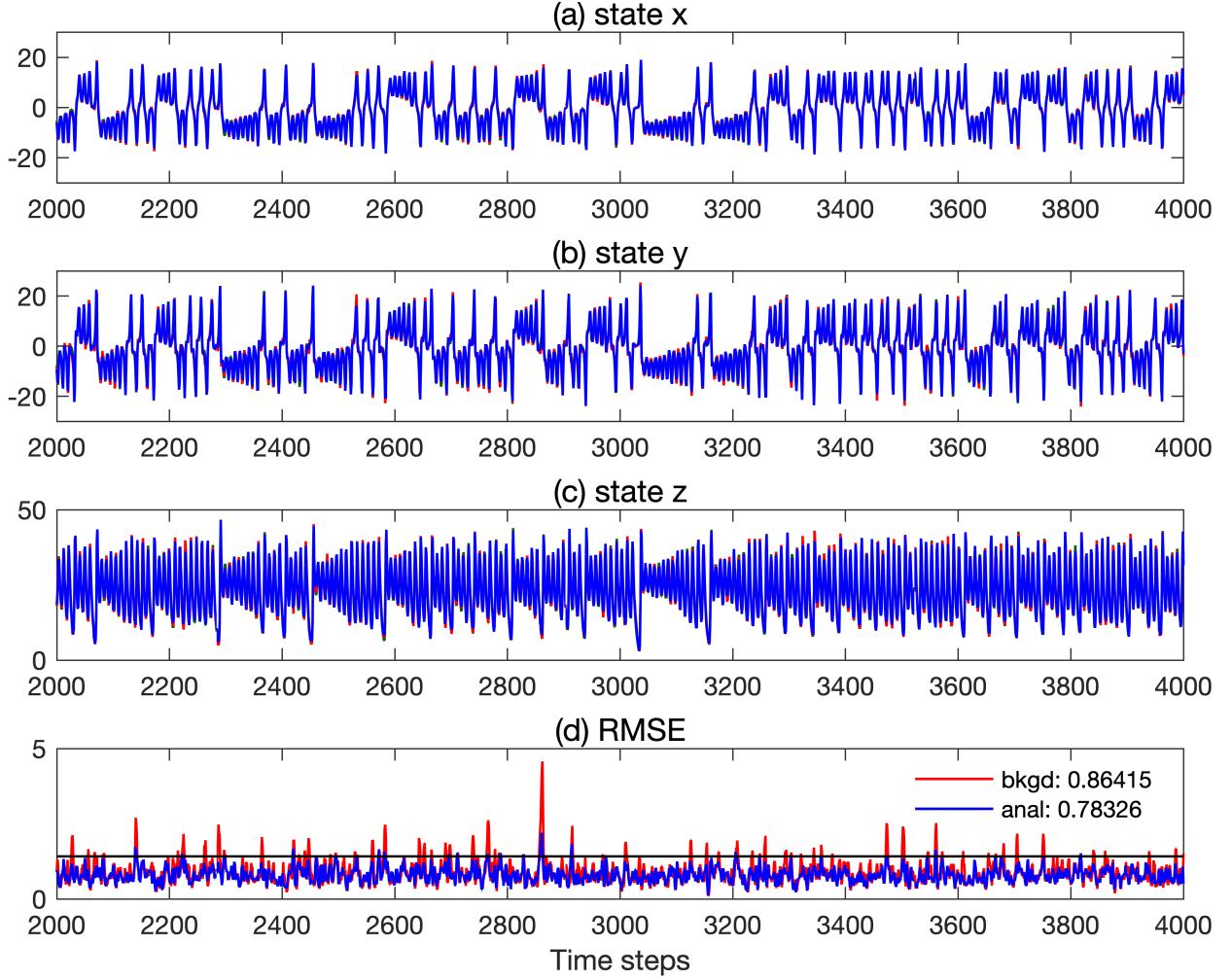


Figure 2: Time series of the background (red), analysis (blue), and truth (black) for model state (a) x, (b) y, and (c) z. The smoothed background and analysis RMSE are shown in (d). The averaged RMSE for cycle 3000-4000 is shown in the figure. The observation error is indicated by the black line.

- (b) Now let's stop doing DA after cycle 3000. Plot the same figure required in (a). Then  
(1) What do you observe after cycle 3000? (2) How does the RMSE grow after the  
cycle 3000? (3) Will the RMSE grow to infinity, or its maximum value is bounded?  
(*hints*: Modify the condition statement above the line "`Call lorenz_3dvar(...)`"  
in the source code `test_3dvar.f90`. After modification, recompile `3dvar.exe` with  
the command `bash compile_3dvar.sh`. Then run `3dvar.exe`)

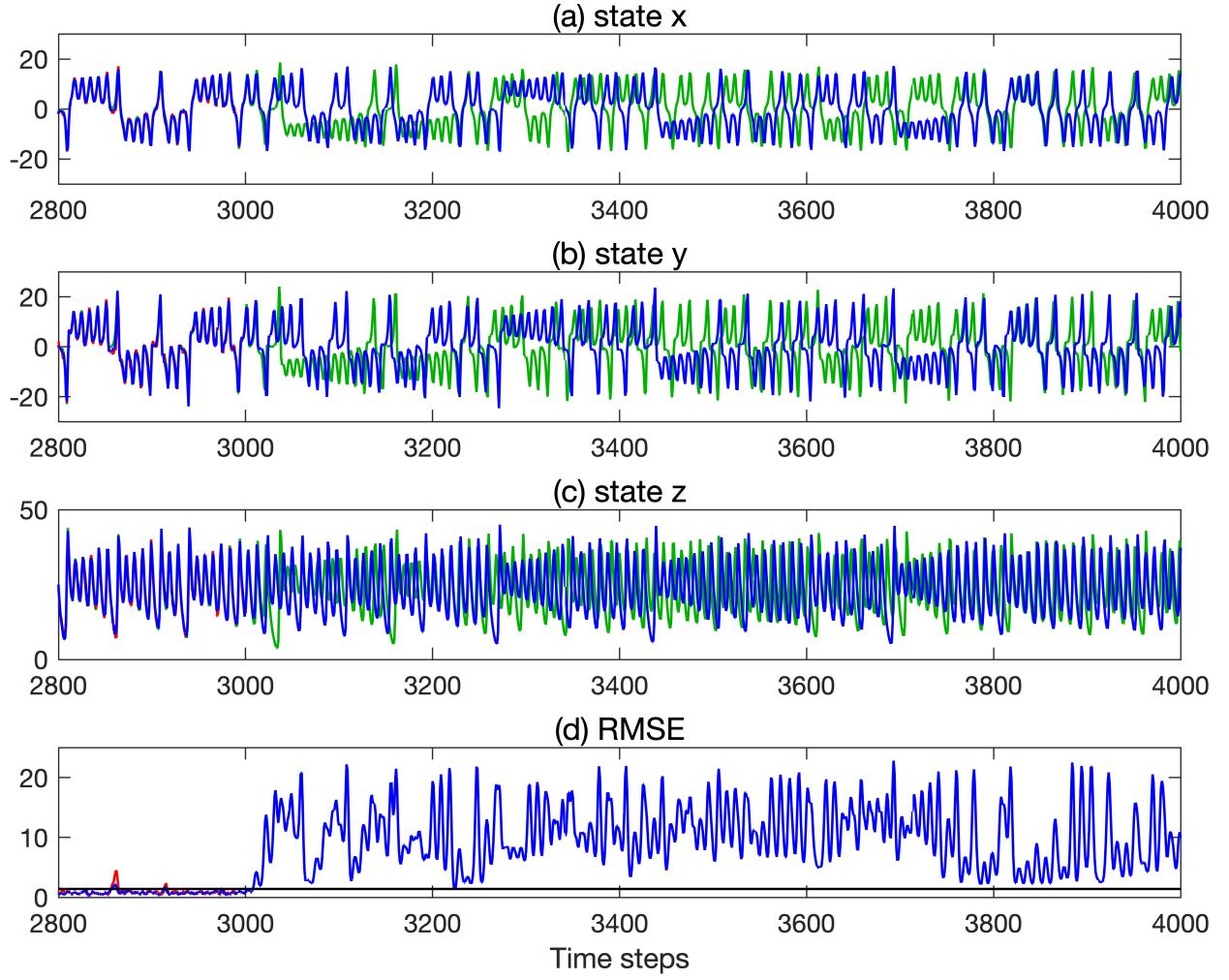


Figure 3: Time series of the background (red), analysis (blue), and truth (black) for model state (a) x, (b) y, and (c) z. The background and analysis RMSE are shown in (d). The observation error is indicated by the black line.

**Ans:**

Results are shown in Figure 3.

After we stop performing DA after cycle 3000, the background and analysis no longer track the truth trajectory. In addition, the analysis RMSE starts to grow quickly, but it won't grow to infinity and becomes saturated.

This shows why we need to **continuously** perform the data assimilation for the chaotic systems. Even the initial error is small, the analysis and truth will get completely trajectories at a later time. However, the discrepancy between the analysis and the truth won't grow to infinity. The error will get saturated after a while.

Many applications use similar estimation methods as data assimilation (e.g., the optimal estimation used by the retrieval community, 1D-Var). However, they don't pay much attention to the time evolution of the error. For NWP community, error evolution is an important topic, because the weather system is chaotic!

How do we know if a system is chaotic? How does errors evolve for weather systems? What methods can we utilize to describe these errors? We will find answers to these questions in Chapter 6!

(c) Following (b), if we resume DA after cycle 3500, what will happen?

**Ans:**

Results are shown in Figure 4. After we resume DA from cycle 3500, the analysis trajectory gradually tracks the truth trajectory again! It's amazing!

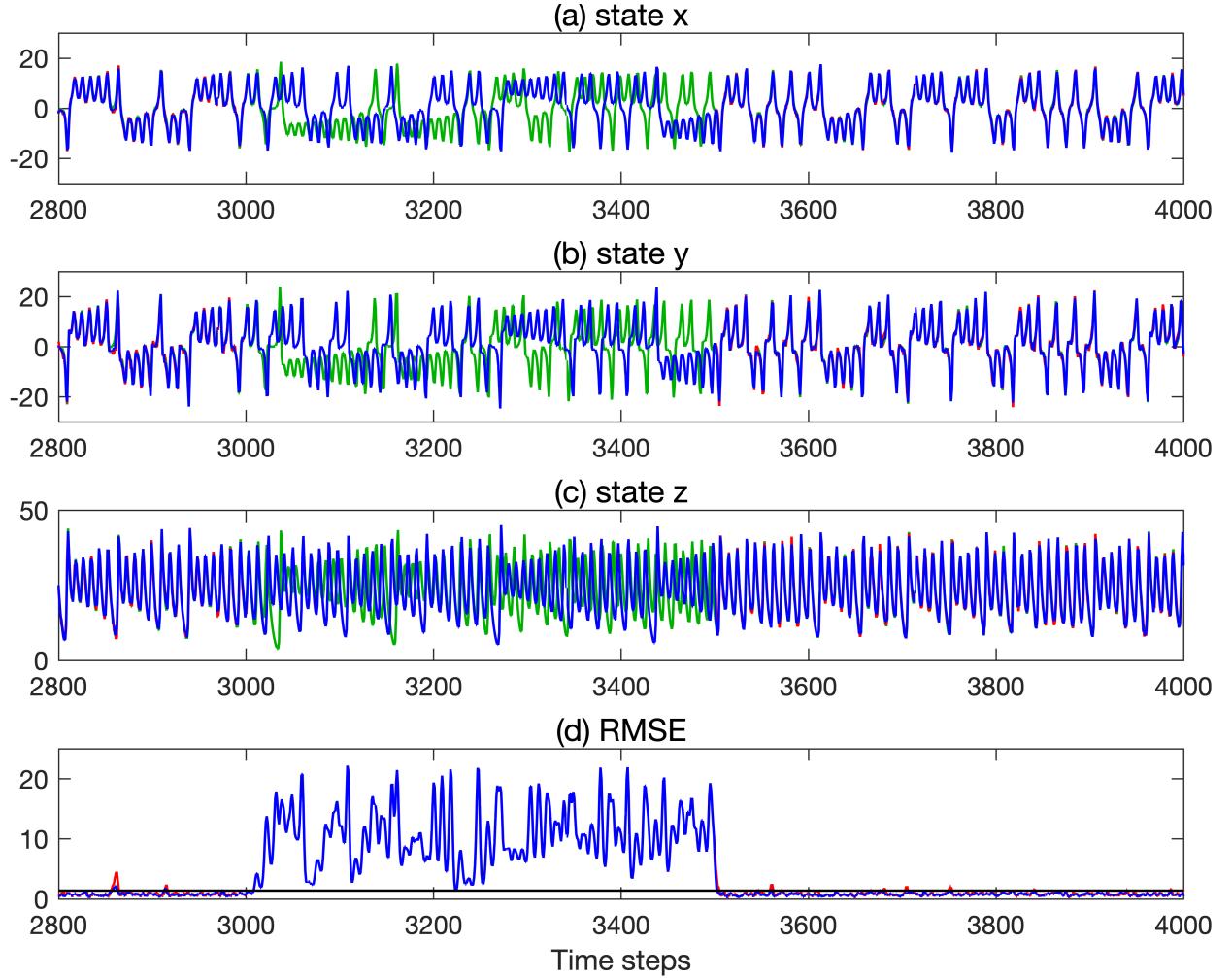


Figure 4: Time series of the background (red), analysis (blue), and truth (black) for model state (a) x, (b) y, and (c) z. The background and analysis RMSE are shown in (d). The observation error is indicated by the black line. In this case we assimilate observations for x, y, and z

Someone as smart as you might think: this is not a big deal because you have observations for every model state (i.e., x, y, z). Though each observation contains some error, it approximates the truth. But in reality, you don't have this luxury because we don't have observations for every variables everywhere. How do we deal with this situation?

This is a very good question!

So let's set another experiment: To mimic the reality, we now **only assimilate observations for  $x$  after cycle 3500**, and redo the experiment. The results are shown in Figure 5.

Even with observation of  $x$  only, the analysis trajectory still gradually tracks the truth trajectory! Look at the trajectory for  $y$  and  $z$ . Though the fluctuation of analysis around truth become larger now (also larger RMSE), we have no phase errors.

In this case, the background error matrix  $\mathbf{B}$  acts as a messenger: through error correlation between  $x$  and  $y$ , and between  $x$  and  $z$ , the uncertainties of  $y$  and  $z$  are reduced by observations of  $x$ . More amazingly, the background error matrix  $\mathbf{B}$  doesn't contain direct correlation among the variables  $x, y, z$ , but relation among their errors. Utilizing their error relationships among different variables, we can calculate a more accurate estimation for the variable itself. This is amazing.

A question when you lose your sleep at night: Besides  $\mathbf{B}$ , what other components can work as messengers?

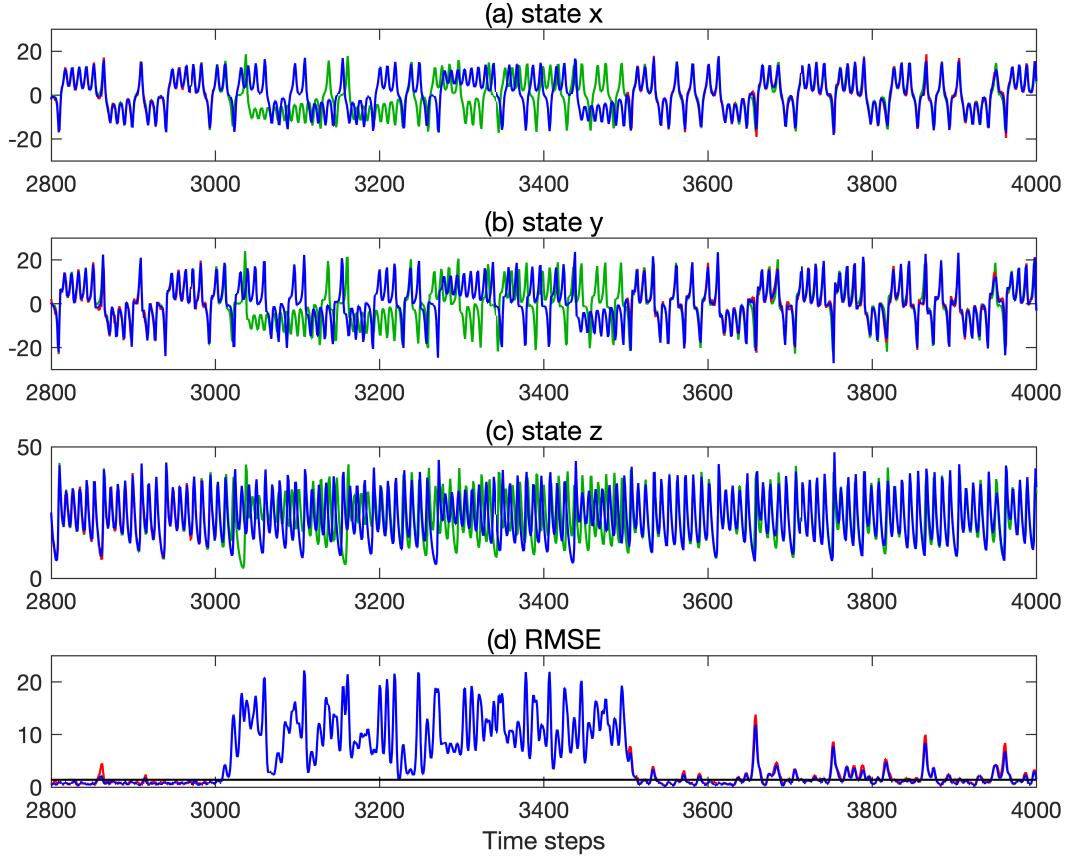


Figure 5: Time series of the background (red), analysis (blue), and truth (black) for model state (a)  $x$ , (b)  $y$ , and (c)  $z$ . The background and analysis RMSE are shown in (d). The observation error is indicated by the black line. In this case we assimilate observations for  $x$ ,  $y$ , and  $z$

2. (3D-Var versus 4D-Var) It's a tremendous advance from 3D-Var to 4D-Var since 4D-Var accounts for the "error of the day". In 4D-Var, we assimilate observations from several time slots in each DA cycle, tracing back their impact through the adjoint to the initial background. We will compare the performance of these two methods:
- Generate 3D-Var analysis and plot the time series of analysis RMSE. Calculate the average RMSE for cycle 3000–4000.
  - Genearte 4D-Var analysis, and then overlay its analysis RMSE over 3D-Var results. Calculate the average RMSE for cycle 3000–4000.  
(Make sure in `test_inc4dvar.f90`:  
`kitermax_out=3,`  
`nsteps_per_da=2,`  
`nsteps_da_window=2`  
then recompile `inc4dvar.exe` with “`bash compile_inc4dvar.sh`”, and run `inc4dvar.exe`)

What did you find from this experiment?

**Ans:**

The results are shown in Figure 6. Clearly, the analysis error of 4D-Var is smaller than that of 3D-Var, because 4D-Var considers the "errors of the day". Technically, this is achieved by utilizing the tangent linear and adjoint model to pass the impact of observations at different time to the initial conditions.

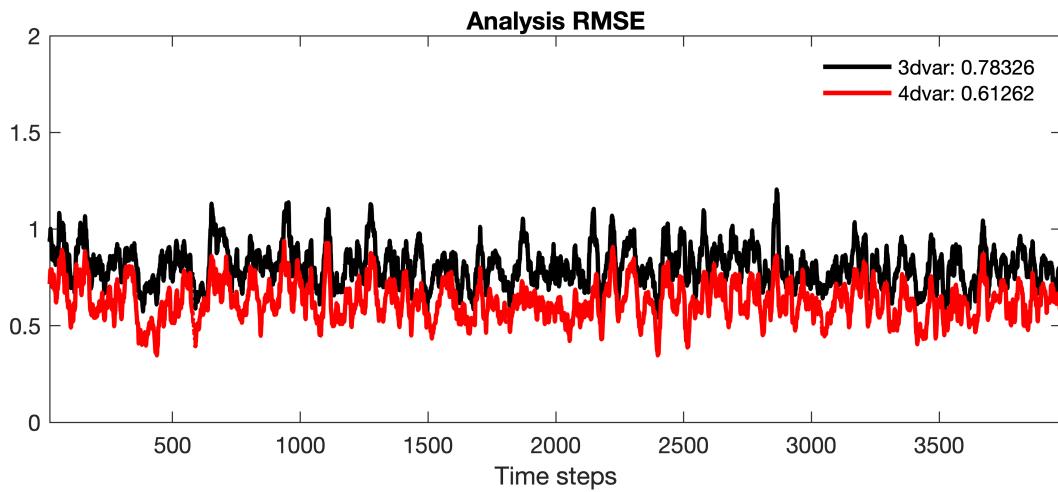


Figure 6: Time series of the smoothed analysis RMSE for 3D-Var (black) and 4D-Var (red). The average RMSE for cycle 3000–4000 is shown in the figure. The smoothing parameter N is 10.

3. (4D-Var: impact of the assimilation window length): What if we increase the assimilation window length? How does it influence the analysis? Let's find it out:

Let's fix the outerloop as 3 (set `kitermax_out=3` in `test_inc4dvar.f90`). Then vary the window length from 2 to 4 (i.e., `nsteps_per_da=2, nsteps_da_window=2` for window length of 2) and generate 4D-Var analysis for each scenario. Overlay their smoothed analysis RMSE series in one figure, and calculate average analysis RMSE for cycle 3000–4000.

What can you conclude from the results?

**Ans:**

The results are shown in Figure 7.

4D-Var analysis can achieve smaller RMSE by utilizing longer assimilation window length. On the contrary, EnKF prefers short assimilation window length.

A nice paper that compares 4D-Var and EnKF:

Eugenia Kalnay, Hong Li, Takemasa Miyoshi, Shu-Chih Yang & Joaquim Ballabrera-Poy (2007) 4-D-Var or ensemble Kalman filter?, *Tellus A: Dynamic Meteorology and Oceanography*, 59:5, 758-773, DOI: 10.1111/j.1600-0870.2007.00261.x

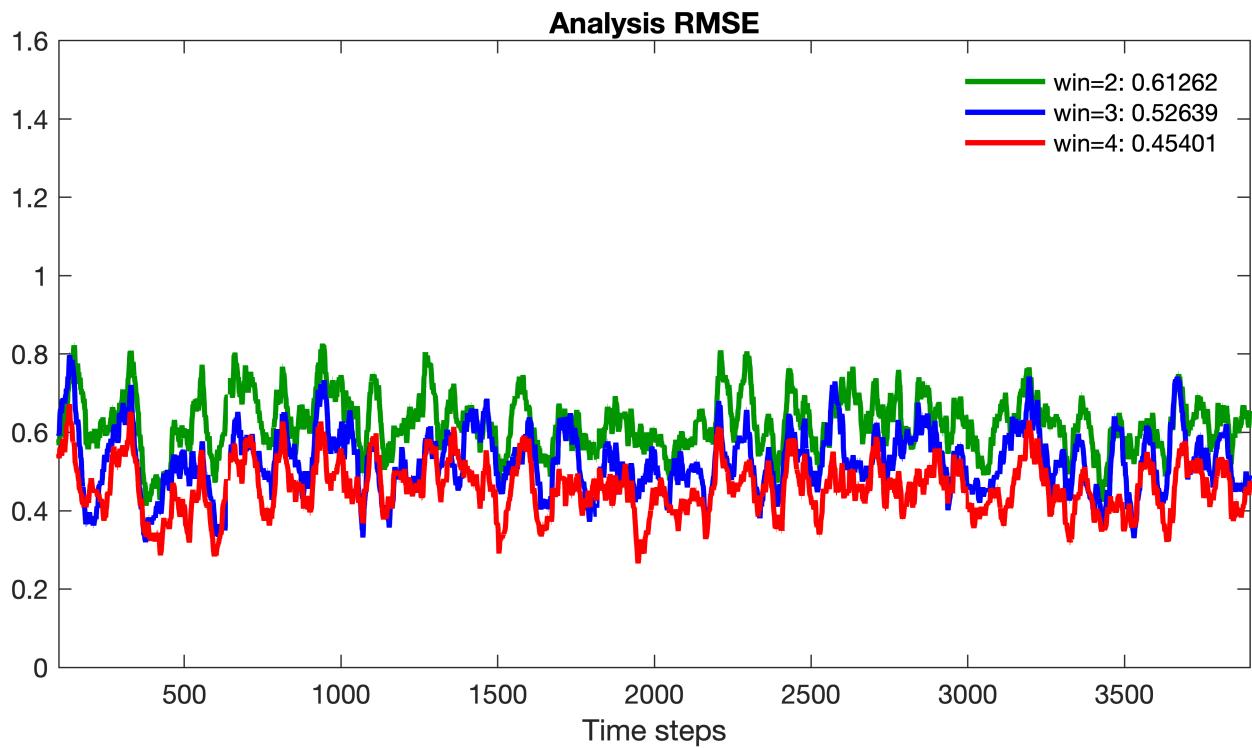


Figure 7: Time series of the smoothed 4D-Var analysis RMSE with the assimilation window length of 2 (green), 3 (blue) and 4 (red). 3 outerloops are used for all experiments. The average RMSE for cycle 3000–4000 is shown in the figure. The smoothing parameter N is 20.

4. (4D-Var: impact of the outer loop) In all previous 4D-Var experiments, we use three outerloops (`kitermax_out=3`). But how does the outerloop influence the results? First let's set a long assimilation window length of 5 (set `nsteps_per_da=5,nsteps_da_window=5`). Change the number of outerloop (`kitermax_out`) from 1 to 3 and generate 4D-Var analysis for each case.

What did you find? Can you try to explain why this happens?

**Ans:**

For the experiments, we adopts a slightly different parameters in Figure 8. All experiments utilize assimilation window length of 5, and we try outerloops from 1 to 4. The averaged analysis RMSE for each experiment is shown in the figure. We notice as the number of outerloop increases, the RMSE decreases.

Besides, we notice when outerloop is less than 4, there are occasional spikes for the analysis RMSE, especially when 1 outerloop is utilized. What happened?

Let's zoom in cycle 2000-2500 (Figure 9 and 10).

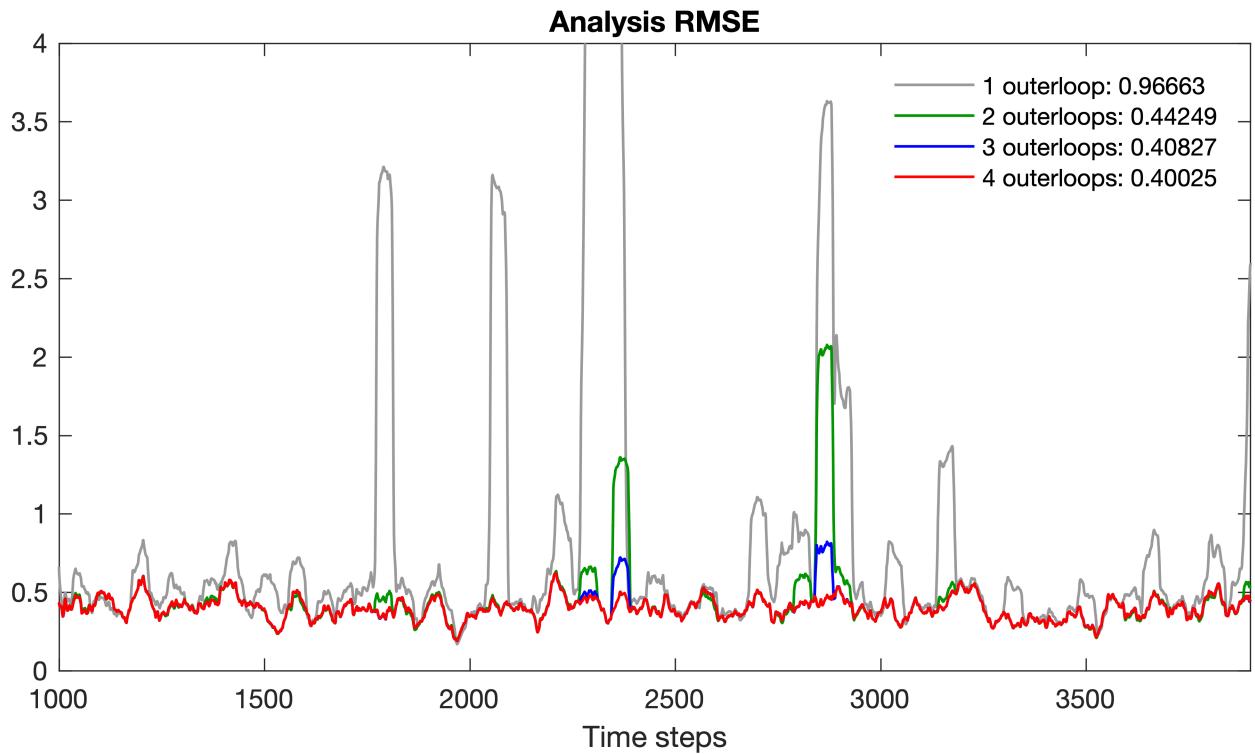


Figure 8: Time series of the smoothed 4D-Var analysis RMSE with 1 (gray), 2 (green), 3 (blue) and 4 (red) outerloops. The assimilation window lenght is set as 5. The average RMSE for cycle 1000-4000 is shown in the figure. The smoothing parameter N is 20.

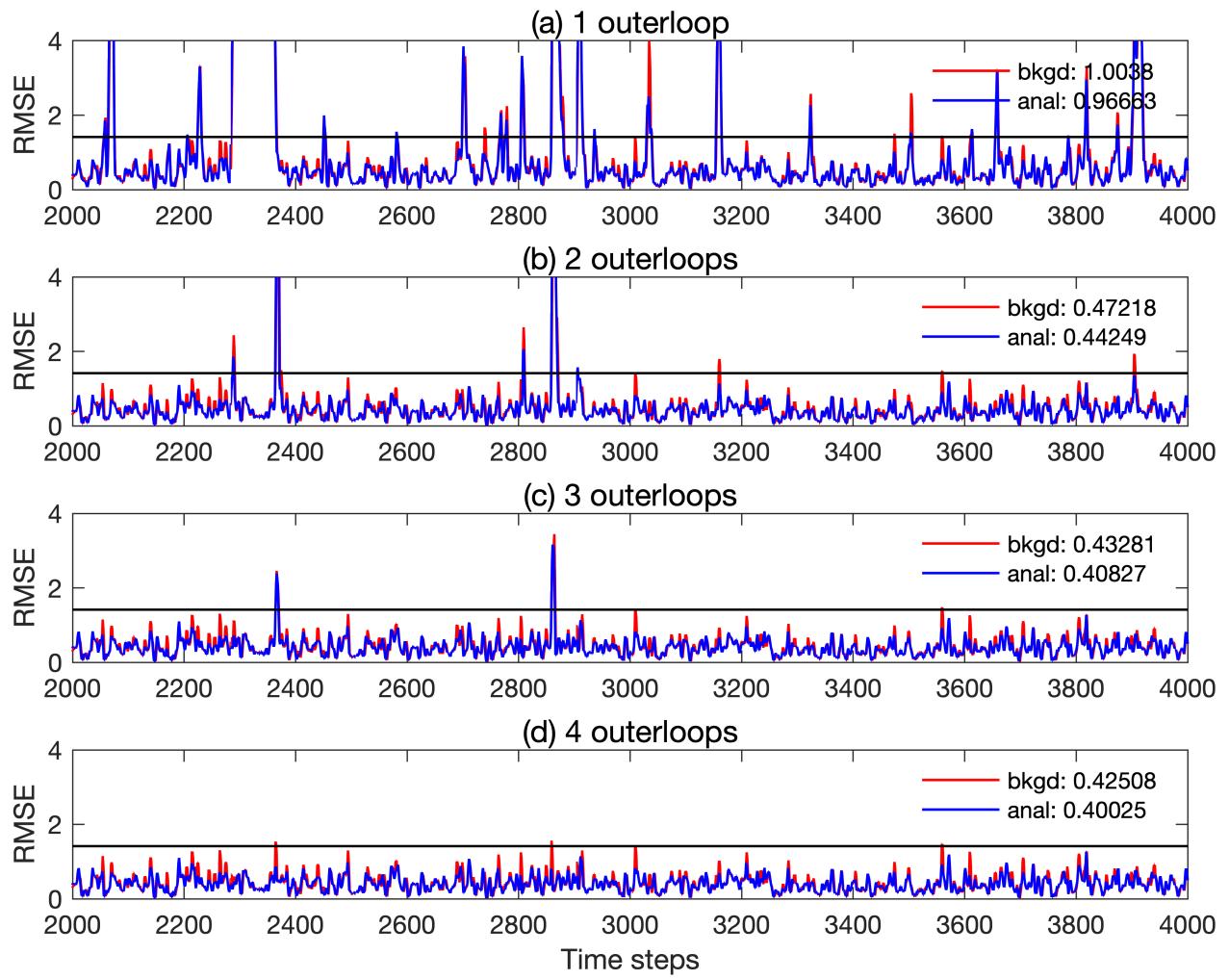


Figure 9: Time series of the smoothed 4D-Var analysis RMSE with 1-4 outerloops. The assimilation window lenght is set as 5. The average RMSE for cycle 1000-4000 is shown in the figure. The smoothing parameter  $N$  is 20.

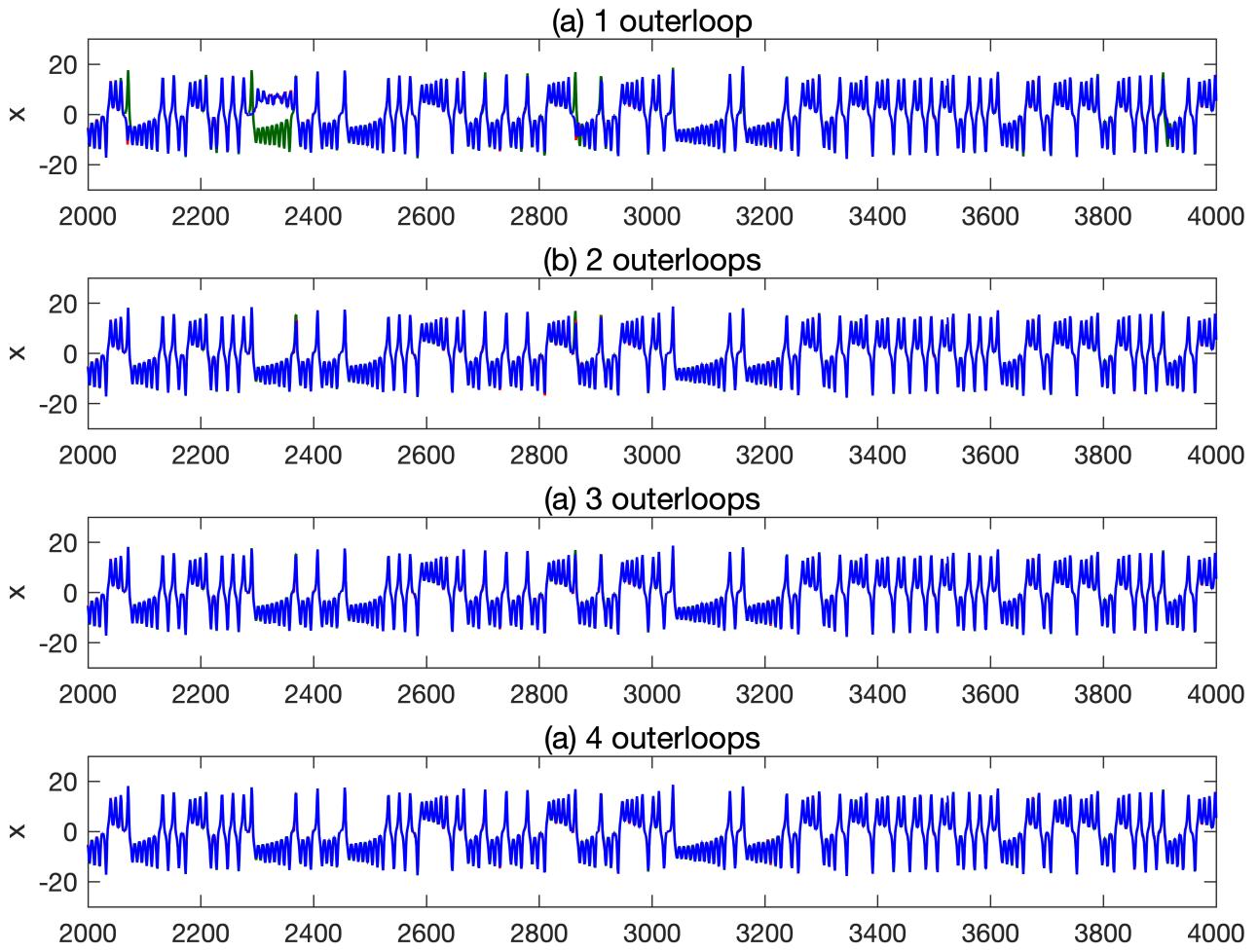


Figure 10: Time series of the analysis trajectory for state  $x$  in each experiment. The assimilation window length is set as 5.

In the previous example, we notice by utilizing longer assimilation window length, we can further reduce the analysis RMSE. However, the benefit with long window length is not free: we need to increase the number of outerloops. For longer window length, the model evolution is no longer approximately linear, and the assumption of the tangent linear model breaks. We need to add additional outerloops to iteratively improve the initial condition.

For ensemble DA, Kalnay et al. developed similar methods for the LETKF (i.e., Running in Place and Quasi-outerloop).

## 4 Part C: Ensemble method

We will now focus on the ensemble methods, which also takes the "error of the day" into account while whose developments are much more simple than 4D-Var. In addition to the technical simplicity, ensemble methods also provide us the explicit uncertainty estimates of our analysis as byproducts, which are not available in 4D-Var.

### 4.1 Related codes

For the experiments in Part C, you only need to use `enda.exe`, which works with you interactively. After typing `./enda.exe` in the command line, the program will:

1. ask you to input the observation error, which should be a float number. For all the experiments in this project, input the same value `1.414d0` (where `d0` indicates this is a double-precision float number).
2. Then the program will ask you to input the ensemble size. You will need to input an integer from the keyboard (for example, 3).
3. Then the program will ask you to input the inflation parameter, which should be a real number between `0.d0` and `1.d0`.
4. Finally, the program will print out a list of all ensemble methods available, and ask you to select one. In the experiments in Part B, you will input 1 for LETKF, 2 for perturbed observation EnKF.

After these four inputs, the program will run OSSE, and the results are written into files, as indicated in Section 1.3.3.

### 4.2 Tasks

For all the following experiments, the observation errors are all set with a value of `1.414d0`.

1. (Compare 3D-Var, 4D-Var and LETKF)
  - (a) Use 10-member LETKF (filter 1 in `enda.exe`) with no inflation (value `0.d0` for inflation parameter). Then in the figure overlay its analysis RMSE with 3D-Var and 4D-Var. Compare their analysis RMSE for cycle 3000–4000  
How is its performance compared with the variational methods?

**Ans:**

Results are shown in Figure 11.

Generally LETKF and 4D-Var have comparable error, which are both smaller than 3D-Var.

Note that since we have ensemble members now, we can **explicitly calculate the covariance between any pair of model states** (e.g.,  $x$  and  $y$ ) **for any time step**. By contrast, though the background error covariance in 4D-Var will evolve with time, we cannot write out the varying covariance that contains the “error of the day” at later time steps.

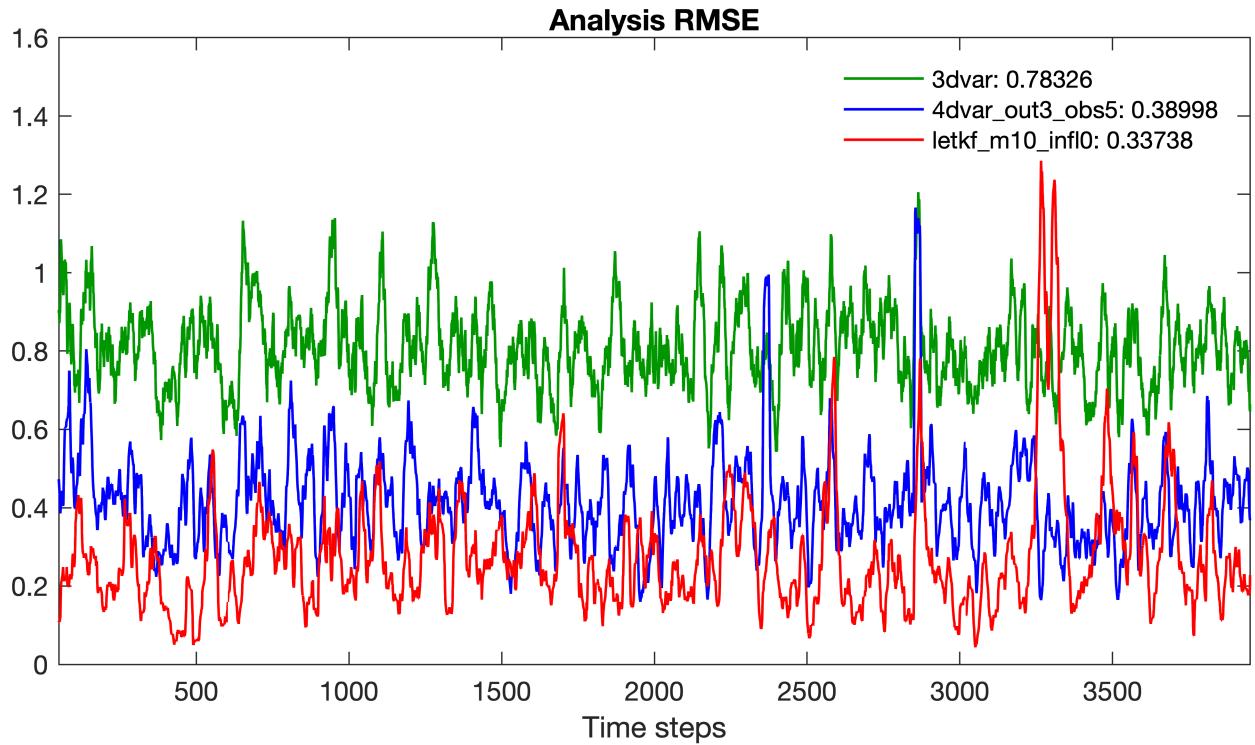


Figure 11: Time series of the analysis RMSE for 3D-Var (green), 4D-Var (blue) and 10-member LETKF (red). The average RMSE for cycle 3000-4000 are shown in the figure. The smoothing parameter N is 10.

- (b) Now use 3-member LETKF (filter 1 in `enda.exe`) with no inflation (value 0.d0 for inflation parameter). Then plot the same figure in (a). How is its performance compared with the variational methods now? If comparing its analysis trajectory with truth, what did you find?

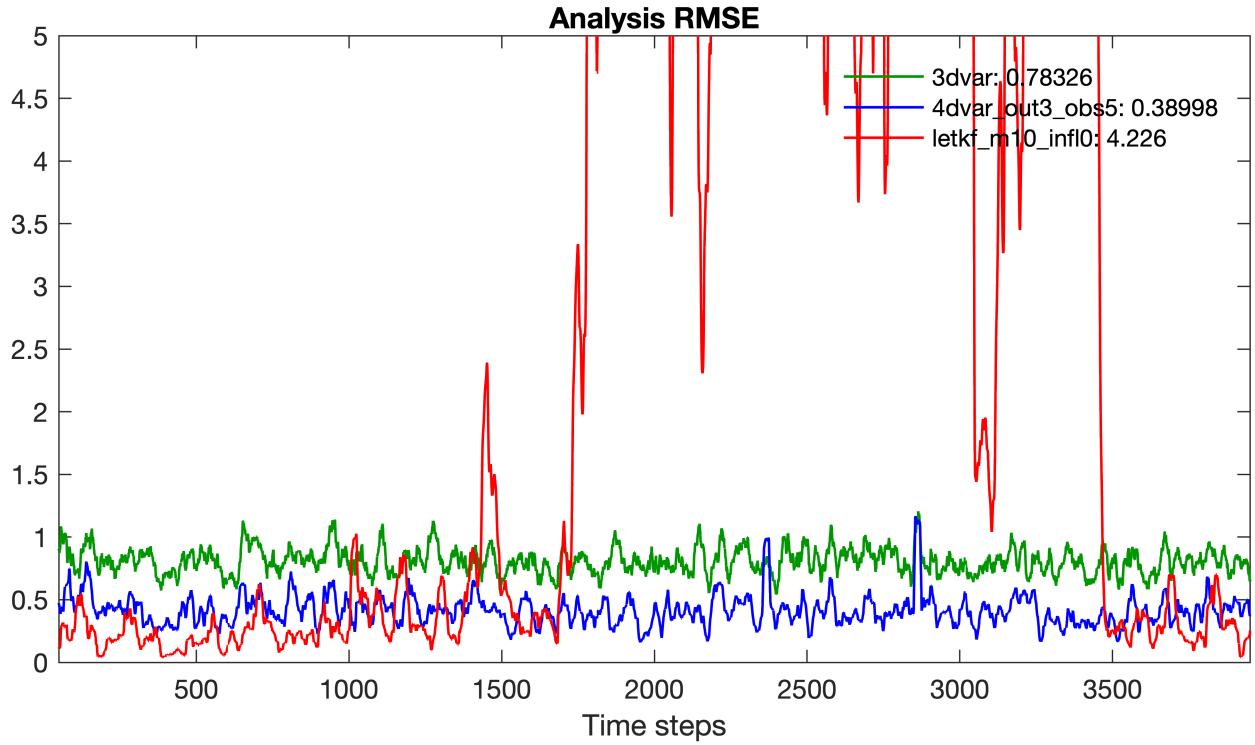


Figure 12: Time series of the analysis RMSE for 3D-Var (green), 4D-Var (blue) and 3-member LETKF (red). The average RMSE for cycle 3000-4000 are shown in the figure. The smoothing parameter N is 10.

**Ans:**

Results are shown in Figure 12.

Now LETKF has far greater RMSE than both 3D-Var and 4D-Var after cycle 1600. What happened? Let's check analysis trajectory for the LETKF (Figure 13).

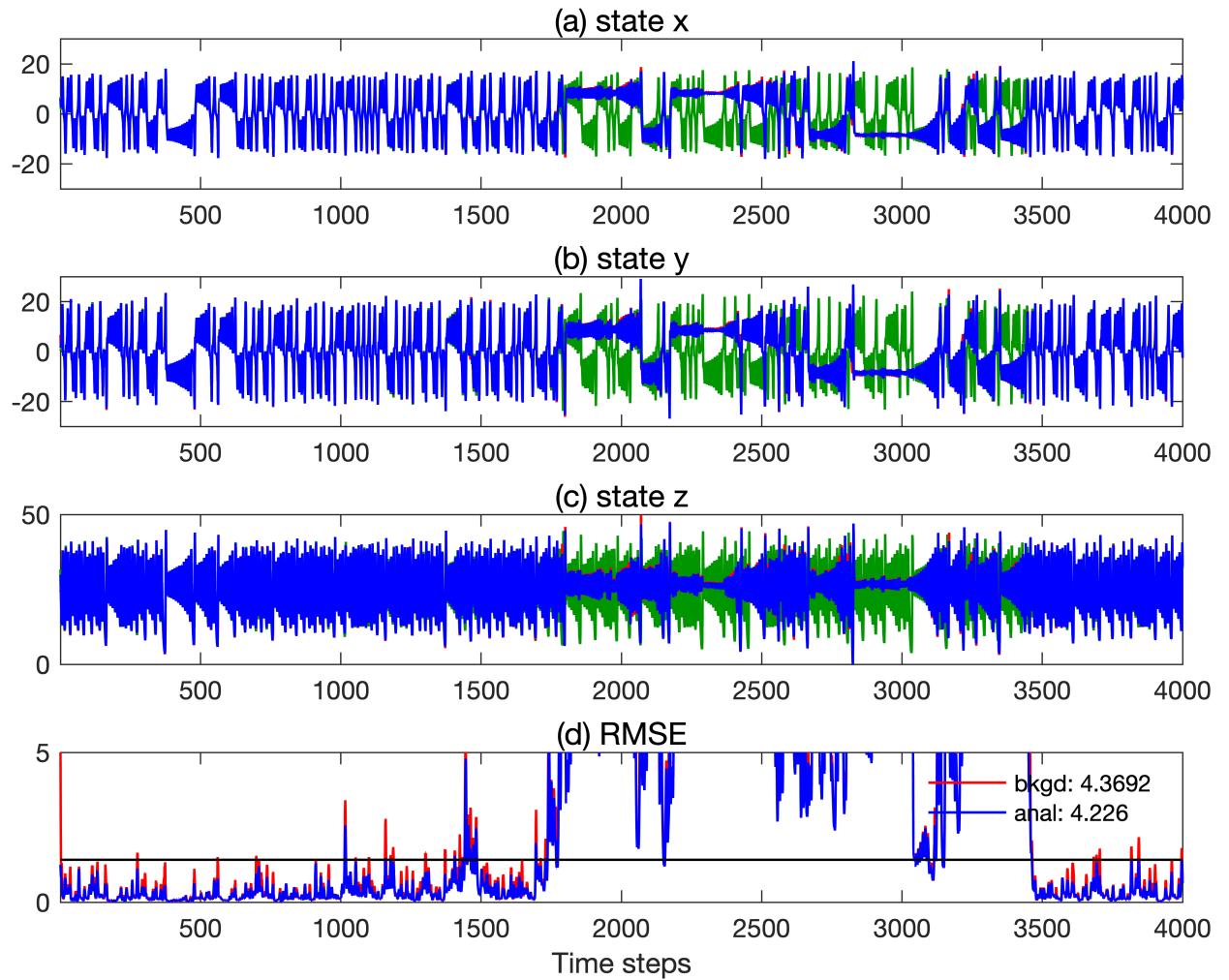


Figure 13: Time series of the background (red), analysis (blue), and truth (black) for model state (a) x, (b) y, and (c) z. The smoothed background and analysis RMSE are shown in (d). The averaged RMSE for cycle 3000-4000 is shown in the figure. The observation error is indicated by the black line.

We notice that around cycle 1600, the analysis generated by LETKF no longer track the error. This is a common problem for the EnKF with small ensemble size. Due to small size, the background error estimated from ensembles will be underestimated (i.e., overconfident about the background error). Under this circumstance, observations cannot be effectively assimilated. Their errors are too large compared with the background error so that observations cannot drag the background towards the truth.

2. (Remedy to small ensemble size) In 1(b), we see the ensemble Kalman filter encounters problems when the ensemble size is small compared to the dimension of the model state (i.e., 3). In real applications, the dimension of model state (i.e., model state of the GFS) is far greater than the ensemble size we use ( $N_{ensemble} \sim$  hundreds). How do we fix this problem?

The ensemble DA community developed several simple but effective tools to deal with this problem. Among them, one is called the Relaxation to Prior Perturbation (RTPP, Zhang et al., 2004). The scheme itself is extremely simple, after we get the  $m$ -member analysis perturbations  $\Delta\mathbf{X}^a = [\delta\mathbf{x}^{a,1} | \delta\mathbf{x}^{a,2} | \dots | \delta\mathbf{x}^{a,m}]$ . We create the final analysis perturbation by merging this analysis perturbation and background perturbation, which are

$$\Delta\mathbf{X}_{final}^a = \alpha\Delta\mathbf{X}^b + (1 - \alpha)\Delta\mathbf{X}^a$$

Then we get the final analysis members by adding these final analysis perturbations back to the ensemble analysis mean.

Now let's see if this simple method works. We still use 3-member LETKF, but this time with an inflation parameter of 0.5d0. Check the truth and analysis trajectory, does the LETKF work correctly now?

Ans:

Results are shown in Figure 14.

With relaxation, LETKF can now work correctly: its analysis always track the truth trajectory now. Let's overlay its results with 3D-Var and 4D-Var again (Figure 15): with RTPP, even with small ensemble size, LETKF still has similar performance as 4D-Var!

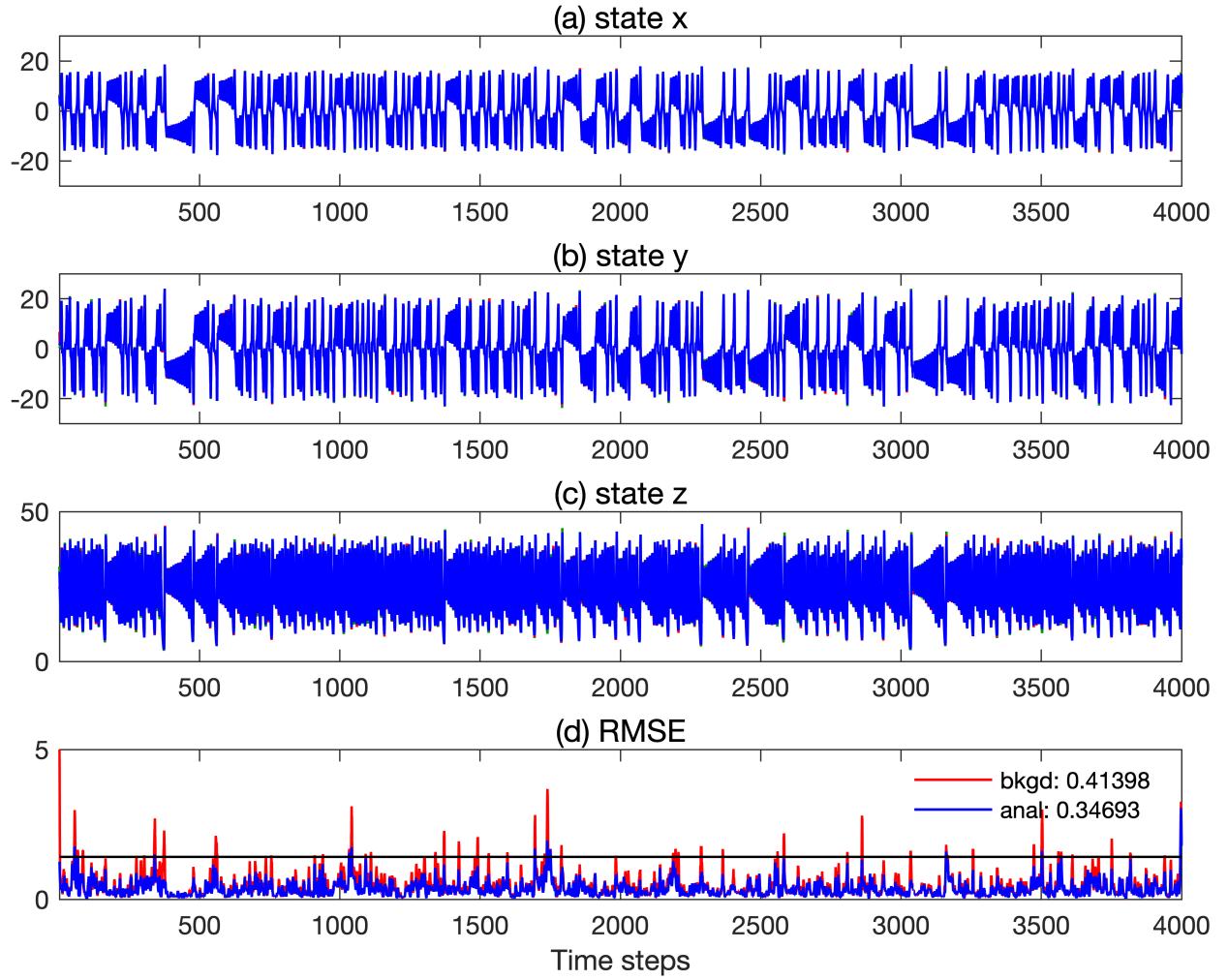


Figure 14: Time series of the analysis RMSE for 3D-Var (green), 4D-Var (blue) and 3-member LETKF (red). The average RMSE for cycle 3000-4000 are shown in the figure. The smoothing parameter  $N$  is 10.

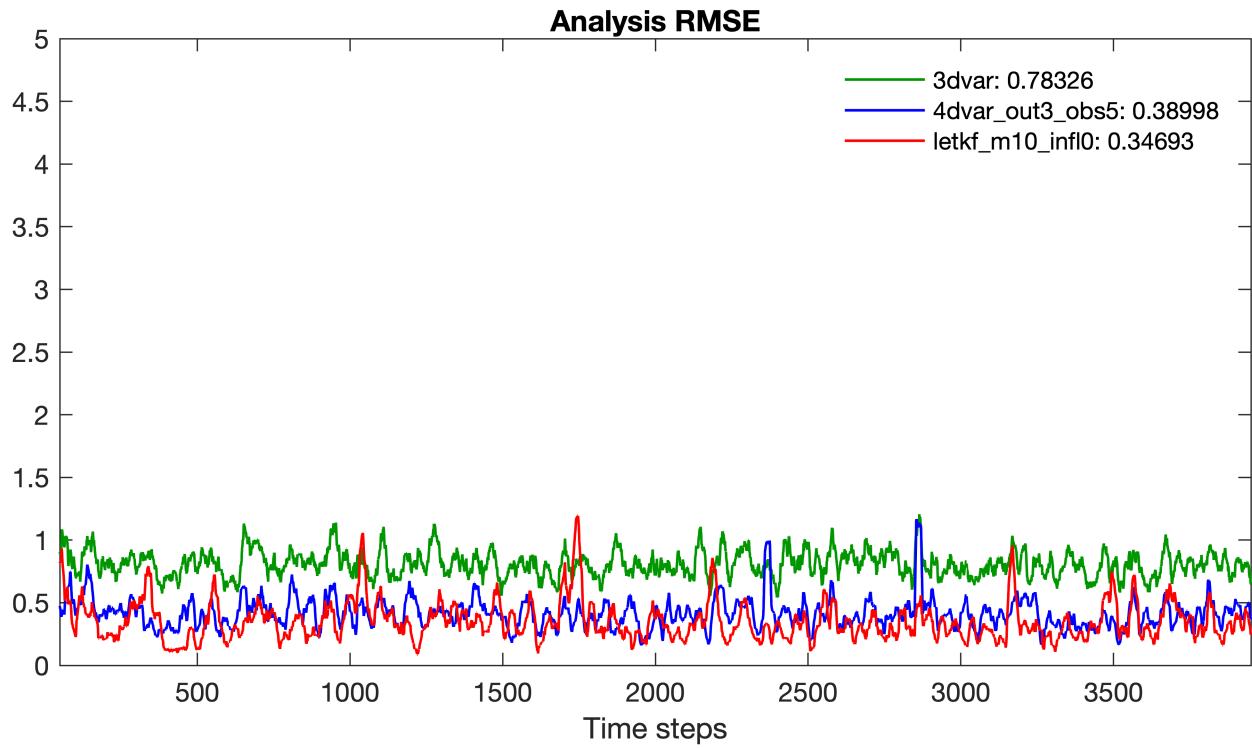


Figure 15: Time series of the background (red), analysis (blue), and truth (black) for model state (a) x, (b) y, and (c) z. The smoothed background and analysis RMSE are shown in (d). The averaged RMSE for cycle 3000-4000 is shown in the figure. The observation error is indicated by the black line.

3. (Partial observation coverage) In DA, the background error covariance plays an important role in sharing information between different variables. Even some variables are not directly observed, through the background error covariance, they can still be updated through observations for other variables. We will explore how LETKF works with partial observation coverage:
  - (a) Start with 5-member LETKF with no inflation, remove observation for variable  $z$ , and run the experiment. Plot the truth and analysis trajectory for each variable. (In `test_enda.f90`, change the last `.true.` to `.false.` in the call “`Call lorenz63_letkf(...)`”. After modification, recompile `enda.exe` and then `./enda.exe`)  
Is the LETKF able to work correctly? How is its average analysis RMSE compared with full observation coverage for cycle 3000–4000?
  - (b) Still start with 5-member LETKF with no inflation, but this time remove observation for variable  $y$  and rerun the LETKF experiments.  
Can the LETKF work correctly now? If not, can you think of a way to make it work without adding new observations?

4. (stochastic EnKF vs. deterministic EnKF) There are two types of ensemble Kalman filters. The perturbed observation EnKF belongs to the stochastic EnKF, while LETKF is formulated in a deterministic manner. Their different formulations lead to their different performances.

In this experiment, we will use no inflation. For both deterministic filter (1 for LETKF, ) and stochastic filter (2 for perturbed observation EnKF):

- (a) calculate their average RMSE for cycle 3000–4000 for three ensemble sizes (3, 10, 15).
- (b) plot the analysis trajectory over the truth trajectory for these two filters for the ensemble size 10.

From the results of 4(a) and 4(b), what did you find?

**Ans:**

The averaged RMSE for different ensemble size is shown in the table below.

Filter	ensemble size		
	3	10	15
PO-EnKF	10.6000	9.8041	0.3128
LETKF	4.226	0.3374	0.2818

From Figure 16, we notice the EnKF with a deterministic formulation requires smaller ensemble size to work correctly than stochastic EnKF. This is because in stochastic EnKF we need to add random observation error so that the posterior analysis error will not be underestimated. However, this random error introduce additional sampling error, which further degrades the performance of the PO-EnKF.

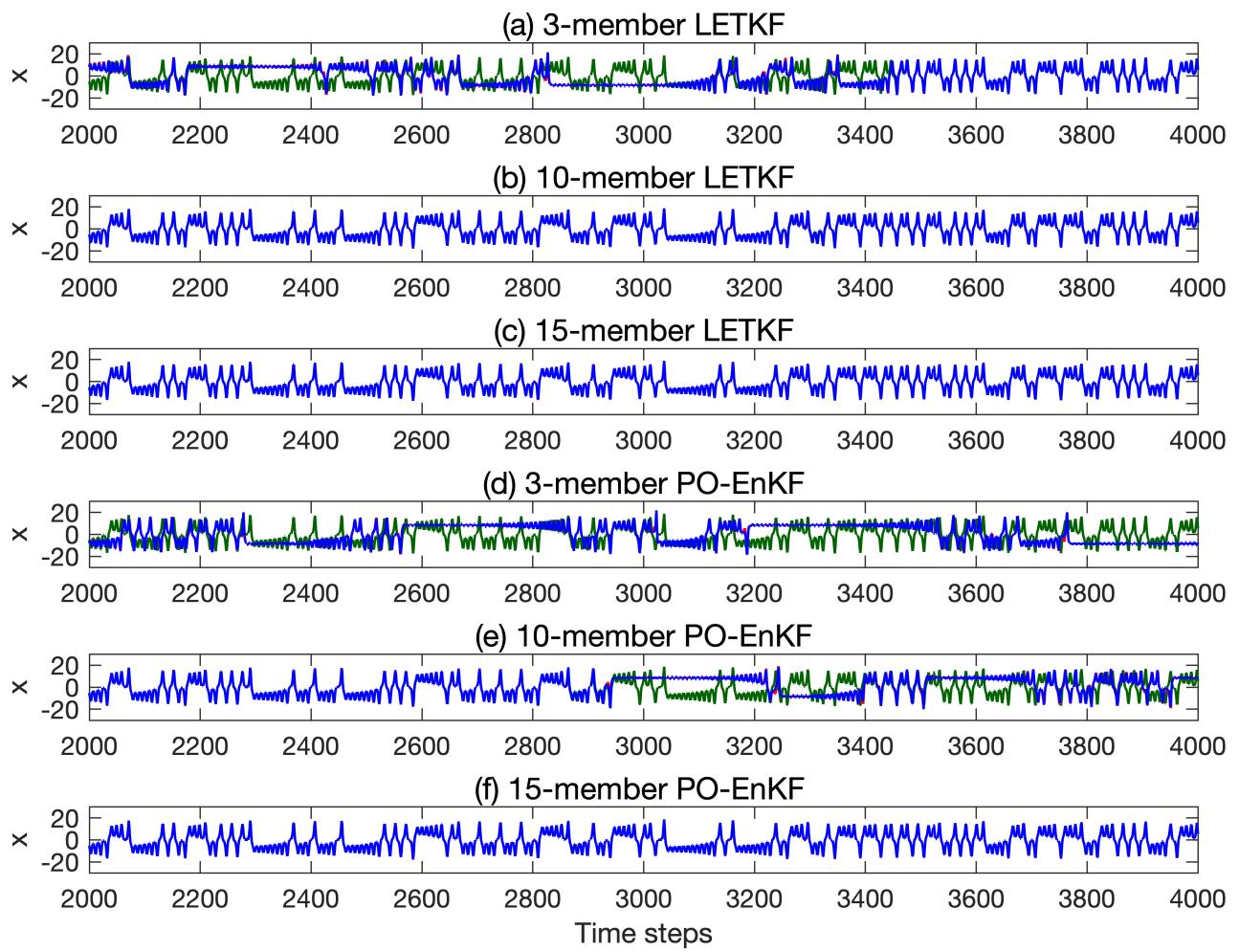


Figure 16: Time series of the analysis trajectory for state  $x$  in each experiment.

## 5 Part D: Hybrid method

Penny [2014] developed a hybrid method (hybrid-gain) that combines both the variational method and ensemble method. Unlike hybrid-covariance method, hybrid-gain method requires minimal code modifications if users already have a variational and ensemble system. In the hybrid-gain method that combines 3D-Var and LETKF, we try to seek the final analysis  $\mathbf{x}_{HG}^a$  as

$$\begin{aligned} \min J(\mathbf{x}_{3DVar}^a) = & (\mathbf{x}_{3DVar}^a - \bar{\mathbf{x}}_{LETKF}^a)^T \mathbf{B}_{3DVar}^{-1} (\mathbf{x}_{3DVar}^a - \bar{\mathbf{x}}_{LETKF}^a) + \\ & (\mathbf{y}^o - \mathcal{H}(\mathbf{x}_{3DVar}^a))^T \mathbf{R}^{-1} (\mathbf{y}^o - \mathcal{H}(\mathbf{x}_{3DVar}^a)) \end{aligned} \quad (3)$$

$$\mathbf{x}_{HG}^a = \alpha \bar{\mathbf{x}}_{LETKF}^a + (1 - \alpha) \mathbf{x}_{3DVar}^a \quad (4)$$

The equations above can be summarized as the following procedure for each DA cycle:

1. Assume we already got the  $m$ -member ensemble background  $\mathbf{X}^b = [\mathbf{x}^{b,1} | \mathbf{x}^{b,2} | \dots | \mathbf{x}^{b,m}]$ , where each column of  $\mathbf{X}^b$  represent one member.
2. Use LETKF to get the LETKF analysis mean  $\bar{\mathbf{x}}_{LETKF}^a$  and analysis perturbations

$$\Delta \mathbf{X}_{LETKF}^a = [\delta \mathbf{x}_{LETKF}^{a,1} | \delta \mathbf{x}_{LETKF}^{a,2} | \dots | \delta \mathbf{x}_{LETKF}^{a,m}]$$

where  $\delta \mathbf{x}_{LETKF}^{a,k} = \mathbf{x}_{LETKF}^{a,k} - \bar{\mathbf{x}}_{LETKF}^a$ ,  $k=1,2,\dots,m$ .

3. Set  $\bar{\mathbf{x}}_{LETKF}^a$  as the background for the 3D-Var:  $\mathbf{x}_{3DVar}^b = \bar{\mathbf{x}}_{LETKF}^a$ , and 3D-Var will then generate its analysis  $\mathbf{x}_{3DVar}^a$ .
4. Generate the hybrid-gain analysis  $\mathbf{x}_{HG}^a = \alpha \bar{\mathbf{x}}_{LETKF}^a + (1 - \alpha) \mathbf{x}_{3DVar}^a$ .
5. Generate the final  $m$ -member ensemble HG analysis members  $\mathbf{X}_{HG}^a = [\mathbf{x}_{HG}^{a,1} | \mathbf{x}_{HG}^{a,2} | \dots | \mathbf{x}_{HG}^{a,m}]$ , where  $\mathbf{x}_{HG}^{a,k} = \mathbf{x}_{HG}^a + \delta \mathbf{x}_{LETKF}^{a,k}$ ,  $k = 1, 2, \dots, m$
6. Evolve each HG analysis member through NWP model to get the background at the next DA step.

### 5.1 Tasks

In this part, we will build a hybrid-gain system based on the existing 3D-Var system (`test_3dvar.f90`) and LETKF (`test_enda.f90`) and performing two experiments:

1. Build a hybrid-gain system that combines 3D-Var and LETKF.
2. Now let's play the hybrid-gain method:
  - (a) (Hybrid-gain vs. EnKF) With full observation coverage, use only 2 or 3 members and no inflation, respectively run LETKF and your hybrid-gain system. For each method, overlay their analysis trajectory over the truth trajectory. For the hybrid-gain method, use  $\alpha = 0.5$ .

Ans:

The Results for LETKF and Hybrid-gain methods are shown in Figure 17.

From (a)-(c) we know 2-member LETKF cannot work correctly with no inflation. However, after we update the LETKF analysis mean with 3D-Var, 2-member LETKF can work correctly now! This shows the hybrid-gain method can stabilize the LETKF with small ensemble size.

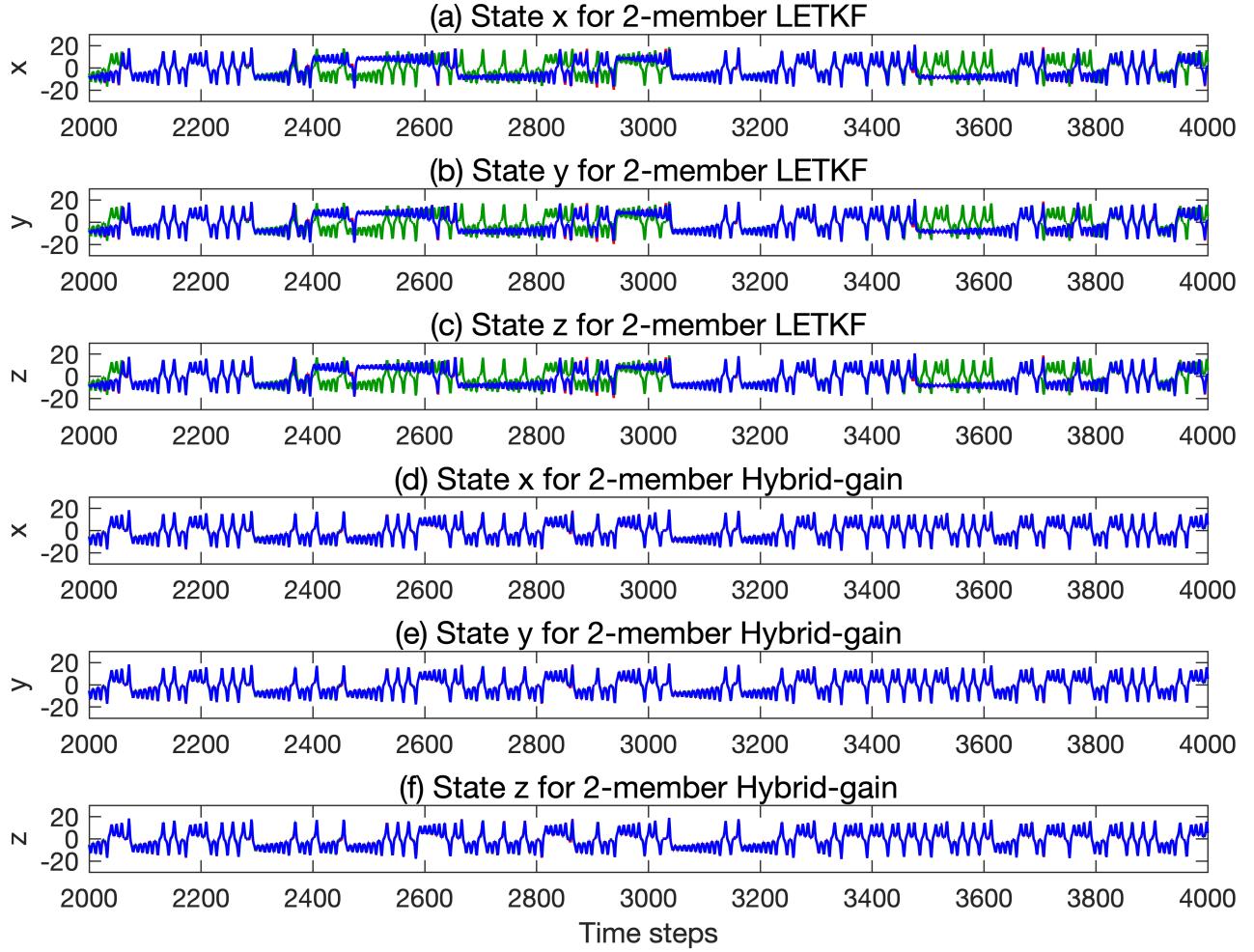


Figure 17: Time series of the analysis (blue) and truth trajectory (green) for states  $x$ ,  $y$ ,  $z$  in 2-member LETKF (a-c) and 2-member hybrid-gain (d-f).

We know from Part C that Perturbed observation EnKF (CMC's system) requires much more members to work than LETKF. What will happen if we combine PO-

EnKF and 3D-Var? The results are shown in Figure 18. Amazingly, the hybrid-gain method is also able to stabilize PO-EnKF with only 2 members!

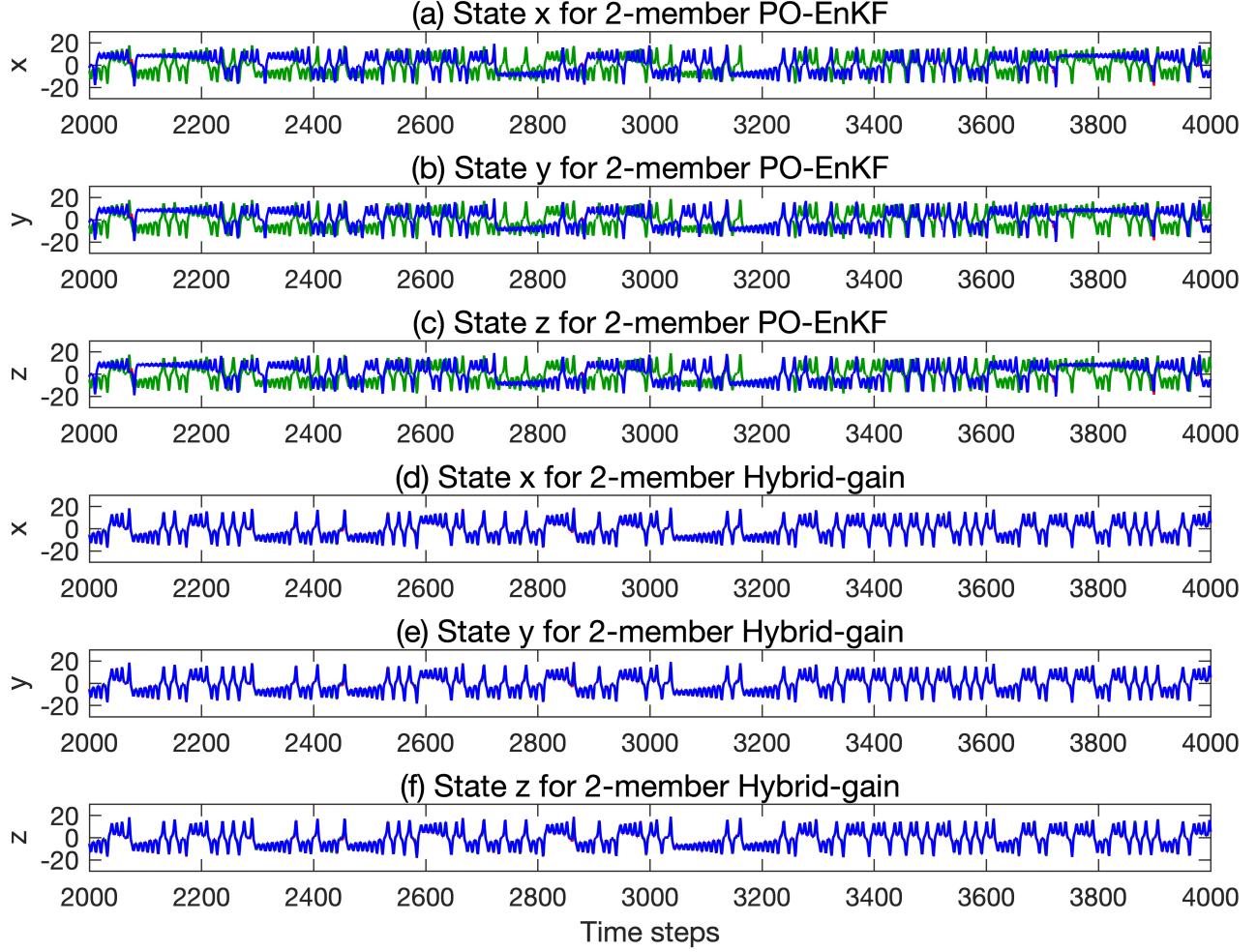


Figure 18: Time series of the analysis (blue) and truth trajectory (green) for states  $x$ ,  $y$ ,  $z$  in PO-EnKF (a-c) and 2-member hybrid-gain (d-f).

- (b) (Hybrid-gain vs. 3D-Var) following (2), now compare your smoothed Hybrid-gain analysis RMSE with 3D-Var RMSE in the same figure. Compare their average RMSE for cycle 3000–4000.

**Ans:**

The results are shown in Figure 19. Compared with 3D-Var, hybrid-gain method generates smaller analysis RMSE by introducing 2 ensemble members.

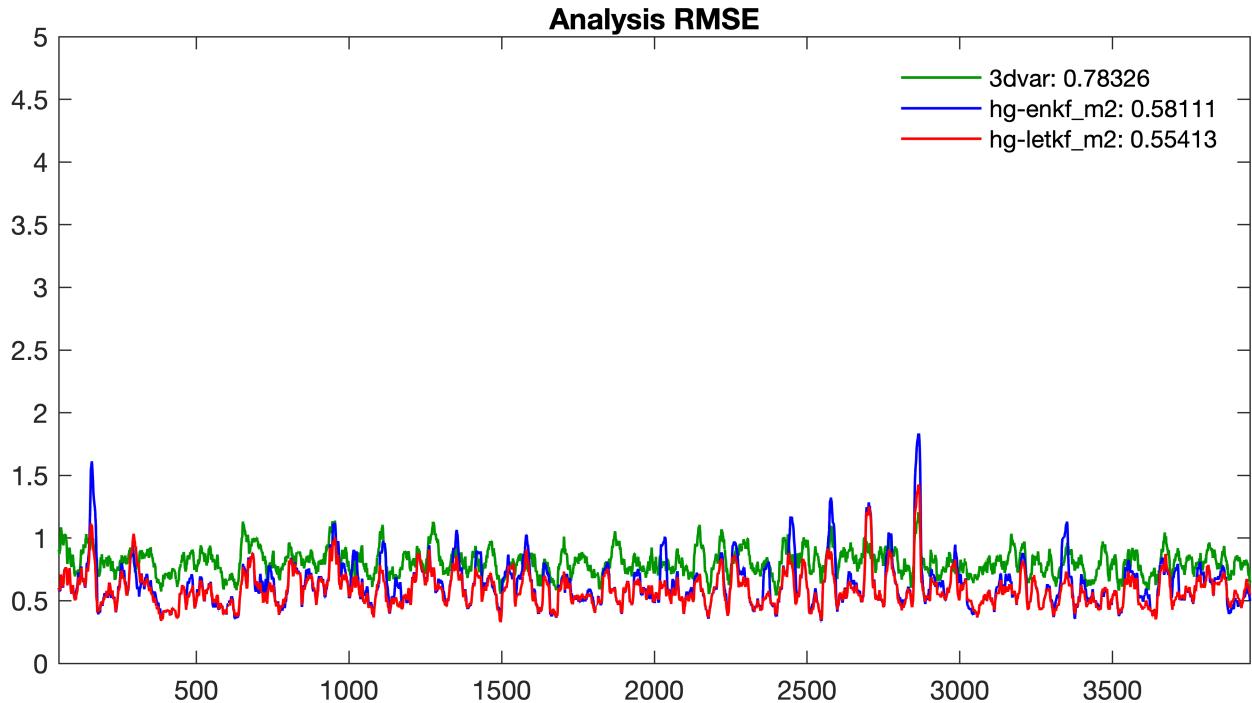


Figure 19: Time series of the analysis RMSE for 3D-Var (green), hybrid-gain with PO-EnKF (blue) and hybrid-gain with PO-EnKF (red). The averaged analysis RMSE for cycle 3000-4000 are shown in the legends.