Project for Chapter 5
Data Assimilation Experiments with the Lorenz-63 model

Last updated: 13:16, November 11, 2020

Contents

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

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

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.

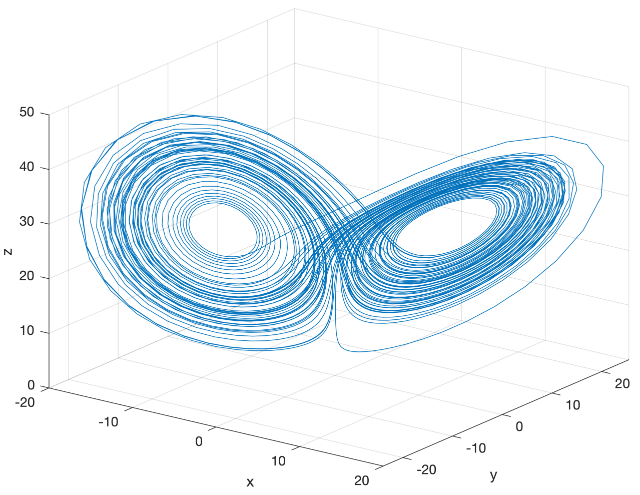

Figure 1: The trajectory of the Lorenz-63 model with parameters $\sigma = 10, \beta = 8/3, \rho = 28$.

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 ϵ 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 $\mathbf{x}^b_0$, integrate the Lorenz-63 model from $\mathbf{x}^b_0$ to get the background $\mathbf{x}^b_1 = \mathcal{M}(\mathbf{x}^b_0)$ until the first DA step. Assimilate available observations $\mathbf{y}^o_1$ at this time step and update the background to the analysis $\mathbf{x}^a_1$. This analysis will serve the initial condition to get the background for the next DA step: $\mathbf{x}^b_2 = \mathcal{M}(\mathbf{x}^a_1)$.
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

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

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 methods (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 *deepthought*2, 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.)

   (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`)

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

What you observed in this experiment is a universal phenomenon for chaotic systems. It illustrates why we need to do DA continuously.

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:

   (a) Generate 3D-Var analysis and plot the time series of analysis RMSE. Calculate the average RMSE for cycle `3000-4000`.

   (b) 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?
   (*Tips*: The time series of the analysis RMSE will be very noisy. You can further perform a moving average to your raw time series of RMSE. Specifically, for the RMSE σ_i at cycle i, we can generate a smoothed one by

$$\sigma_i^{smooth} = \frac{1}{2N+1} \sum_{-N}^{N} \sigma_{i+k} \tag{3}$$

   where N is the half window size. You can select, for example, $N = 20$, and you will see the new RMSE series is smoother than the original series.

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? Remember to recompile `inc4dvar.exe` for each experiment.
   (*Tips*:
   You can wrap all command together to get the result for each experiment. For example, assume you are now running the experiments with the window length of `3`. After you set `nsteps_per_da=3,nsteps_da_window=3`. you can run the command below in one line:

```
bash compile_inc4dvar.bsh; ./inc4dvar.exe; bash rename_output.bsh 4dvar_out3_obs3
```

Then your 4D-Var results for this case will be saved into files `4dvar_out3_obs3.100*0`.)

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 `6` (set `nsteps_per_da=6,nsteps_da_window=6`). Change the number of outerloop (`kitermax_out`) from `1` to `3` and generate 4D-Var analysis for each case.

   (a) Overlay their smoothed analysis RMSE series in one figure and calculate average analysis RMSE for cycle `3000-4000`.

   (b) For outerloop equals 1, overlay the truth and analysis trajectory for x, y, z.

   (c) repeat (b) for outerloop is 2 and 3.

   What did you find? Can you try to explain why this happens?
   (*hints*: Think about the approximation used in the formulation of incremental 4D-Var, and whether they are still valid for long assimilation window length).

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?
   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.

   (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?

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 $\Delta\mathbf{X}^a = [\delta\mathbf{x}^{a,1}|\delta\mathbf{x}^{a,2}|...|\delta\mathbf{x}^{a,m}]$. We create the final analysis perturbation by merging this analysis perturbation and background perturbation, which are

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

   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?

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?

5 Part D: Hybrid method

Penny [2014] developed a hyrbid 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 $\mathbf{x}^a_{HG}$ as

$$\begin{aligned} min\ J(\mathbf{x}^a_{3DVar}) =& (\mathbf{x}^a_{3DVar} - \bar{\mathbf{x}}^a_{LETKF})^T \mathbf{B}^{-1}_{3DVar} (\mathbf{x}^a_{3DVar} - \bar{\mathbf{x}}^a_{LETKF}) + \\ & (\mathbf{y}^o - \mathcal{H}(\mathbf{x}^a_{3DVar}))^T \mathbf{R}^{-1} (\mathbf{y}^o - \mathcal{H}(\mathbf{x}^a_{3DVar})) \end{aligned} \tag{4}$$

$$\mathbf{x}^a_{HG} = \alpha \bar{\mathbf{x}}^a_{LETKF} + (1-\alpha)\mathbf{x}^a_{3DVar} \tag{5}$$

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

1. Assume we already got the m-member ensemble background $\mathbf{X}^b = [\mathbf{x}^{b,1}|\mathbf{x}^{b,2}|...|\mathbf{x}^{b,m}]$, where each column of $\mathbf{X}^b$ represent one member.

2. Use LETKF to get the LETKF analysis mean $\bar{\mathbf{x}}^a_{LETKF}$ and analysis perturbations

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

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

3. Set $\bar{\mathbf{x}}^a_{LETKF}$ as the background for the 3D-Var: $\mathbf{x}^b_{3DVar} = \bar{\mathbf{x}}^a_{LETKF}$, and 3D-Var will then generate its analysis $\mathbf{x}^a_{3DVar}$.
4. Generate the hybrid-gain analysis $\mathbf{x}^a_{HG} = \alpha\bar{\mathbf{x}}^a_{LETKF} + (1-\alpha)\mathbf{x}^a_{3DVar}$.
5. Generate the final m-member ensemble HG analysis members $\mathbf{X}^a_{HG} = [\mathbf{x}^{a,1}_{HG}|\mathbf{x}^{a,2}_{HG}|...|\mathbf{x}^{a,m}_{HG}]$, where $\mathbf{x}^{a,k}_{HG} = \mathbf{x}^a_{HG} + \delta\mathbf{x}^{a,k}_{LETKF}, k = 1, 2, ..., m$
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 with 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 $\alpha = 0.5$.
   (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`.

   What did you find from 2(a) and 2(b)?

5.2 Coding tips

1. To implement hybrid-gain, you can make a copy of `test_enda.f90` and name it as `test_hg.f90` and then try to insert 3D-Var subroutine `lorenz63_3dvar` into `test_hg.f90`.
2. 3D-Var subroutine `lorenz63_3dvar` is in the code `mod_lorenz63_3dvar.f90`. LETKF subroutine `lorenz63_letkf` is in the code `mod_lorenz63_letkf.f90`.
3. After finishing writing the main program `test_hg.f90`, you will need to compile your code with the script `compile_hg.bsh`. If your code has no coding error, you will see an executable named `hg.exe` under the directory. And you will use this executable to run the hybrid-gain method. Remember to select filter `1` for the LETKF code section when running `hg.exe`.

6 (Optional) Part E: Information transport in a DA system

In this part, we try to figure out how observations adjust background model state through different components (i.e., $\mathbf{B}, \mathbf{H}, \mathbf{L}$) in a DA system. Knowing how each component modifies the background in a DA system enables us to trace back the components that failed to work realistically when we get an inferior analysis. For example, in carbon data assimilation, the variable localization method [Kang et al., 2011] significantly improve carbon flux estimation by removing unrealistic ensemble inter-variable covariance due to small ensemble size. As the first step, we will try to understand how those DA components transport information in different scenarios.

6.1 Tasks

As shown in the Introduction, Lorenz-63 system includes three model variables (i.e., x, y, z), so their background error covariance matrix $\mathbf{B}$ is a 3×3 matrix. For the following scenarios, please tell if the background will be updated due to the assimilation of observations. **Specifically, you need to determine: (1) if $x^a \neq x^b$, (2) if $y^a \neq y^b$, and (3) if $z^a \neq z^b$. Please also briefly explain your answer.** For all the scenarios, please assume that the background value never equals observation value(i.e., $x^b \neq x^o, y^b \neq y^o, z^b \neq z^o$), and the observation errors (i.e., $\mathbf{R}$) are nonzero.
(*hints*: Each scenario includes at least one special DA component so that you can make quick judgements without calculation. If you cannot intuitively answer them, you can refer to KF/OI/3D-Var equations.)

- (Scenario 1) Assuming you are using a OI/3D-Var system for the Lorenz-63 equations for one DA cycle. For this cycle, you have direct observations x^o, y^o, z^o, and your $\mathbf{B}$ has the structure $\begin{bmatrix} e_{xx} & e_{xy} & e_{xz} \\ e_{xy} & e_{yy} & e_{yz} \\ e_{xz} & e_{yz} & e_{zz} \end{bmatrix}$ where elements started with e are nonzero.
- (Scenario 2) Assuming you are using a OI/3D-Var system for the Lorenz-63 equations for one DA cycle. For this cycle, you only have direct observations x^o, and your $\mathbf{B}$ has the structure $\begin{bmatrix} e_{xx} & e_{xy} & e_{xz} \\ e_{xy} & e_{yy} & e_{yz} \\ e_{xz} & e_{yz} & e_{zz} \end{bmatrix}$ where elements started with e are nonzero.
- (Scenario 3) Assuming you are using a OI/3D-Var system for the Lorenz-63 equations for one DA cycle. For this cycle, you only have direct observations x^o, and your $\mathbf{B}$ has the structure $\begin{bmatrix} e_{xx} & e_{xy} & 0 \\ e_{xy} & e_{yy} & e_{yz} \\ 0 & e_{yz} & e_{zz} \end{bmatrix}$ where elements started with e are nonzero.
- (Scenario 4) Assuming you are using a OI/3D-Var system for the Lorenz-63 equations for one DA cycle. For this cycle, you only have direct observations x^o, and your $\mathbf{B}$ has the

structure $\begin{bmatrix} e_{xx} & 0 & 0 \\ 0 & e_{yy} & 0 \\ 0 & 0 & e_{zz} \end{bmatrix}$ where elements started with e are nonzero.

- (Scenario 5) Assuming you are using a OI/3D-Var system for the Lorenz-63 equations for one DA cycle. For this cycle, your $\mathbf{B}$ has the structure $\begin{bmatrix} e_{xx} & 0 & 0 \\ 0 & e_{yy} & 0 \\ 0 & 0 & e_{zz} \end{bmatrix}$ where elements started with e are nonzero. However, in this case, you don't have any direct observations. Instead, you have one observation $q^o = h(x, y) = x^t + y^t + \epsilon$, where x^t, y^t are true value of model state x, y, and ϵ a Gaussian noise.

- (Scenario 6) Assuming you are now using a 4D-Var system for the Lorenz-63 equations for one DA cycle. For this cycle, your $\mathbf{B_0}$ has the structure $\begin{bmatrix} e_{xx} & 0 & 0 \\ 0 & e_{yy} & 0 \\ 0 & 0 & e_{zz} \end{bmatrix}$ where elements started with e are nonzero. Still, you only have direct observations for model state x. But since you are using 4D-Var, the assimilation window length is set as 6, which means now you actually have x observations from 6 time slots (i.e.,$x_1^o, x_2^o, x_3^o, x_4^o, x_5^o, x_6^o$).