

# Data assimilation with the Lorenz equations

Amos S. Lawless, University of Reading

## Contents

<b>1</b>	<b>The Lorenz equations</b>	<b>1</b>
<b>2</b>	<b>Sequential data assimilation</b>	<b>2</b>
2.1	Data assimilation schemes . . . . .	2
2.1.1	Successive corrections . . . . .	2
2.1.2	Optimal interpolation . . . . .	2
2.1.3	Analysis correction . . . . .	2
2.1.4	Kalman filter . . . . .	3
2.2	The program . . . . .	3
2.3	Suggested exercises . . . . .	4
<b>3</b>	<b>Four-dimensional variational data assimilation (4D-Var)</b>	<b>5</b>
3.1	Introduction . . . . .	5
3.2	Test routines - Building a 4D-Var system . . . . .	6
3.2.1	Test of tangent linear model . . . . .	6
3.2.2	Test of adjoint model . . . . .	6
3.2.3	Gradient test . . . . .	7
3.3	Assimilation program . . . . .	7
3.4	Suggested exercises . . . . .	7

## 1 The Lorenz equations

We consider various data assimilation schemes applied to the Lorenz equations, a simple dynamical model with chaotic behaviour. The Lorenz equations are given by the nonlinear system

$$\frac{dx}{dt} = -\sigma(x - y), \tag{1}$$

$$\frac{dy}{dt} = \rho x - y - xz, \tag{2}$$

$$\frac{dz}{dt} = xy - \beta z, \tag{3}$$

where  $x = x(t)$ ,  $y = y(t)$ ,  $z = z(t)$  and  $\sigma, \rho, \beta$  are parameters, which in these experiments are chosen to have the values 10, 28 and 8/3 respectively.

The system is discretized using a second order Runge-Kutta method, which gives the following discrete equations:

$$\begin{aligned} x^{k+1} &= x^k - \sigma \Delta t / 2 [2(y^k - x^k) + \Delta t(\rho x^k - y^k - x^k y^k) \\ &\quad - \sigma \Delta t(y^k - x^k)], \end{aligned} \tag{4}$$

$$\begin{aligned} y^{k+1} &= y^k + \Delta t / 2 [\rho x^k - y^k - x^k z^k + \rho(x^k + \sigma \Delta t(y^k - x^k)) - y^k \\ &\quad - \Delta t(\rho x^k - y^k - x^k z^k) \\ &\quad - (x^k + \sigma \Delta t(y^k - x^k))(z^k + \Delta t(x^k y^k - \beta z^k))], \end{aligned} \tag{5}$$

$$\begin{aligned} z^{k+1} &= z^k + \Delta t / 2 [x^k y^k - \beta z^k \\ &\quad + (x^k + \Delta t \sigma(y^k - x^k))(y^k + \Delta t(\rho x^k - y^k - x^k z^k)) \\ &\quad - \beta z^k - \Delta t(x^k y^k - \beta z^k)], \end{aligned} \tag{6}$$

where  $\Delta t$  is the model time step and  $k$  is the time step index.

The exercises will investigate various sequential data assimilation schemes and four-dimensional variational assimilation applied to these equations.

## 2 Sequential data assimilation

You will investigate various data assimilation schemes using the program *lorenz\_menu*. The true state and the background state are defined in the program. You can compare different data assimilation schemes with different observation frequencies and accuracies.

### 2.1 Data assimilation schemes

Four sequential schemes are implemented. They are successive corrections, optimal interpolation, analysis correction and the Kalman filter. The schemes themselves can be applied exactly or approximately. We describe briefly the four schemes. The notation we use is as follows -  $\mathbf{x}_b(i)$  is the background state with error covariance matrix  $B^{-1}$ ,  $\mathbf{y}(i)$  is the vector of observations with covariance matrix  $R^{-1}$  and  $H$  is the (possibly nonlinear) observation operator which converts the model field  $\mathbf{x}(i)$  into an equivalent model observation value. The solution to the data assimilation problem, which we call the analysis, we write  $\mathbf{x}_a$ .

#### 2.1.1 Successive corrections

The successive corrections method is an iterative algorithm, which can be written

$$\mathbf{x}^{j+1}(i) = \mathbf{x}^j(i) + W[\mathbf{y}(i) - H(\mathbf{x}^j(i))], \tag{7}$$

where  $\mathbf{x}^0(i) = \mathbf{x}_b(i)$  is the background state,  $j$  is the iteration or correction index and  $W$  is a weighting matrix. The algorithm is stopped after  $k$  corrections, after which the analysis is given by  $\mathbf{x}_a(i) = \mathbf{x}^k(i)$ . For the experiments in these programs the weighting matrix is given by  $W = 0.5 \times I$ .

### 2.1.2 Optimal interpolation

The optimal interpolation analysis is given by

$$\mathbf{x}_a(i) = \mathbf{x}_b(i) + K[\mathbf{y}(i) - H(\mathbf{x}_b(i))], \quad (8)$$

with

$$K = BH^T(HBH^T + R)^{-1}. \quad (9)$$

If  $H$  is a nonlinear operator then it should be linearized around a background state.

### 2.1.3 Analysis correction

The analysis correction algorithm is written

$$\mathbf{x}^{j+1}(i) = \mathbf{x}^j(i) + WQ[\mathbf{y}^j(i) - H(\mathbf{x}^j(i))], \quad (10)$$

$$\mathbf{y}^{j+1}(i) = \mathbf{y}^j(i) - Q[\mathbf{y}^j(i) - H(\mathbf{x}^j(i))], \quad (11)$$

where  $\mathbf{y}^0(i)$  is the vector of observations,  $W = BH^TR^{-1}$ ,  $Q = (HW + I)^{-1}$ . If  $H$  is a nonlinear operator then its linearization around a background state should be used. The algorithm is stopped after  $k$  corrections, after which the analysis is given by  $\mathbf{x}_a(i) = \mathbf{x}^k(i)$ .

### 2.1.4 Kalman filter

In the Kalman filter we explicitly evolve the background and analysis error covariance matrices, which we write  $P_f$  and  $P_a$  respectively. We also assume that the model error covariance  $Q(i)$  is known. The Kalman filter algorithm is then as follows:

$$K(i) = P_f(i)H^T(i)[H(i)P_f(i)H^T(i) + R(i)]^{-1}, \quad (12)$$

$$\mathbf{x}_a(i) = \mathbf{x}_f(i) + K(i)[\mathbf{y}(i) - H(i)\mathbf{x}_f(i)], \quad (13)$$

$$P_a(i) = [I - K(i)H(i)]P_f(i), \quad (14)$$

$$\mathbf{x}_f(i+1) = M(t_{i+1}, t_i)[\mathbf{x}_a(i)], \quad (15)$$

$$P_f(i+1) = M(t_{i+1}, t_i)P_a(i)M(t_{i+1}, t_i)^T + Q(i). \quad (16)$$

Here  $M(t_{i+1}, t_i)$  is the model operator which maps a model state at time  $t_i$  to a state at time  $t_{i+1}$ ,  $\mathbf{x}_f(0)$  is equal to the initial background field  $\mathbf{x}_b(0)$  and the analyses are given by the sequence of  $\mathbf{x}_a(i)$ .

For a nonlinear model this can be extended to give the *extended Kalman filter* by linearizing the model around a background state. This linear model is then used to evolve the error covariance matrices. As before we must also linearize the observation operator  $H(i)$  if it is nonlinear.

## 2.2 The program

The programs ask for inputs from the user in the following order:

- Please choose an assimilation scheme  
This determines the type of assimilation scheme used, the options being successive correction, optimal interpolation, analysis correction and Kalman filter.
- How many iterations?  
This question appears only if the successive correction or analysis correction scheme is chosen. The user chooses how many iterations of the scheme should be performed, between 1 and 5.
- Use correct weighting matrices?  
This option is available for all except the successive correction scheme. If the user chooses ‘Yes’ then the program will calculate the background and observation error covariance matrices using the truth trajectory. A response of ‘No’ will set these matrices to the identity.
- How many time steps between observations?  
This determines the time frequency of the observations, according to the number of model time steps between observations. The observations can be every 25, 50, 100 or 200 time steps. The total assimilation time is 2000 time steps.
- Noise on observations?  
If the user responds ‘No’ then perfect observations are taken from the ‘truth’ trajectory. A response of ‘Yes’ produces noisy observations equal to the truth with some random noise added.

If the user has requested noise on the observations, then the following questions will follow:

- How read noise?  
If the user chooses ‘Generate in program’, then the program produces random noise to add to the observations and stores it in files called `sc_x_noise.mat`, `sc_y_noise.mat` and `sc_z_noise.mat`. ‘Read from file’ reads in the noise from previously generated files. If this option is chosen, then it is necessary to have the same number of observations as the file was generated with.

- Variance of observation error

This allows the user to set the variance of the random noise to be added to the observations.

The user must then click ‘OK’ on a dialogue box to perform the analysis.

There are three figures output from the program. Figures 1 and 2 show the output for the  $x$  and  $z$  variables respectively. The truth trajectory is shown by a black dashed line, the background trajectory by a blue dashed-dotted line, the observations by magenta circles and the final analysis and forecast by a red line. Figure 3 shows the error (truth-forecast) in the  $x$  and  $z$  variables with time. In a fourth window you will see a list of the options that you have chosen.

## 2.3 Suggested exercises

The initial conditions have been set up such that a change of regime appears in the true solution during the assimilation period, which appears too late in the background forecast. You should investigate how the different assimilation schemes can correct for this and how well the subsequent forecast matches the truth. Things you may like to investigate are:

1. The variance of the background error is approximately 1. Compare the behaviour of the schemes with observation errors of variance 0.5, 1 and 2.
2. Is it better to have a few very accurate observations or many observations which are less accurate? Does this depend on which assimilation scheme you are using?
3. How important is it to have the correct error covariance matrices (i.e. correct weighting matrices) in the schemes?
4. Change the covariance matrices in Section 5 of the code so that  $\mathbf{B} = \alpha_1 \mathbf{I}$  and  $\mathbf{R} = \alpha_2 \mathbf{I}$ , where  $\alpha_1$  and  $\alpha_2$  are some chosen (positive) constants. Investigate the effect of the relative size of these constants on the analysis and forecast.
5. For the analysis correction scheme the iteration of the observations is often omitted. Investigate the effect of this for different observation frequencies and accuracies.

## 3 Four-dimensional variational data assimilation (4D-Var)

### 3.1 Introduction

The 4D-Var schemes in these programs have only an observation term, so minimize a function of the form

$$\mathcal{J} = \frac{1}{2} \sum_{i=0}^n (\mathbf{y}(i) - H_i(\mathbf{x}_i))^T \mathbf{R}^{-1} (\mathbf{y}(i) - H_i(\mathbf{x}_i)). \quad (17)$$

A full 4D-Var scheme minimizes this cost function by use of the nonlinear model and its adjoint, whereas an incremental 4D-Var scheme minimizes a series of simplified cost functions in different ‘outer loops’. You are provided with routines for both types of schemes. The routines used are as follows:

---

<i>lorenz4d.m</i>	Top level routine for full 4D-Var
<i>lorenz4d_inc.m</i>	Top level routine for incremental 4D-Var
<i>calcfg.m</i>	Calculate cost function and its gradient for full 4D-Var
<i>calcfg_inc.m</i>	Calculate cost function and its gradient for incremental 4D-Var

---

<i>modeuler.m</i>	Nonlinear model for Lorenz system
<i>modeuler_tl.m</i>	Tangent linear model
<i>modeuler_adj.m</i>	Adjoint model

---

<i>test_tl.m</i>	Test tangent linear model
<i>test_adj.m</i>	Test adjoint model
<i>test_grad.m</i>	Test of <i>calcfg</i>
<i>test_gradinc.m</i>	Test of <i>calcfg_inc</i>

---

<i>menu_asl</i>	Used to provide menus
-----------------	-----------------------

---

### 3.2 Test routines - Building a 4D-Var system

When building a 4D-Var system, there are standard ways of testing the various components before it is used for assimilation. You can experiment with these tests.

#### 3.2.1 Test of tangent linear model

Suppose that  $M$  is a nonlinear model and  $\mathbf{M}$  is the tangent linear model. Then for small perturbations  $\gamma\delta\mathbf{x}$  we have

$$M(\mathbf{x} + \gamma\delta\mathbf{x}) - M(\mathbf{x}) \approx \mathbf{M}(\mathbf{x})\gamma\delta\mathbf{x}. \quad (18)$$

Hence if we plot the *relative error*

$$E_R = \frac{M(\mathbf{x} + \gamma\delta\mathbf{x}) - M(\mathbf{x})}{\mathbf{M}(\mathbf{x})\gamma\delta\mathbf{x}} - 1 \quad (19)$$

as  $\gamma \rightarrow 0$  we should find that  $E_R \rightarrow 0$ .

**Exercise:** Use the routine *test\_tl* to plot the relative error. Try introducing an error into the tangent linear code *modeuler\_tl* and see what effect it has on the test.

### 3.2.2 Test of adjoint model

For a linear model  $\mathbf{M}$  and its adjoint  $\mathbf{M}^*$  we have the identity

$$\langle \mathbf{M}\delta\mathbf{x}, \mathbf{M}\delta\mathbf{x} \rangle = \langle \delta\mathbf{x}, \mathbf{M}^*\mathbf{M}\delta\mathbf{x} \rangle \quad (20)$$

for any inner product  $\langle, \rangle$  and perturbation  $\delta\mathbf{x}$ . This can be used to test that the adjoint is coded correctly.

**Exercise:** Use the routine *test\_adj* to test the adjoint code. Try introducing an error into the adjoint code *modeuler\_adj* and see what effect it has.

### 3.2.3 Gradient test

Let  $\mathcal{J}$  be a cost function and  $\nabla\mathcal{J}$  be its gradient. Then we can check that the exact gradient of the cost function has been coded by using the identity

$$\Phi(\alpha) = \frac{\mathcal{J}(\mathbf{x} + \alpha\mathbf{h}) - \mathcal{J}(\mathbf{x})}{\alpha\mathbf{h}^T\nabla\mathcal{J}(x)} = 1 + O(\alpha), \quad (21)$$

where  $\mathbf{h}$  is a vector of unit length, which we can take to be  $\nabla\mathcal{J}(x)/\|\nabla\mathcal{J}(x)\|^{-1}$ . For small values of  $\alpha$  not too close to machine zero we expect  $\Phi(\alpha)$  to be close to 1.

**Exercise:** Use the program *test\_grad* or *test\_gradinc* to test either of the two cost functions. The output of these routines is a plot of  $\Phi(\alpha)$  and a plot of  $|\Phi(\alpha) - 1|$ . Try introducing an error into the gradient calculation to see how this affects the test results.

## 3.3 Assimilation program

The routines used to run assimilation experiments are *lorenz4d* for the full 4D-Var and *lorenz4dinc* for incremental 4D-Var. The menu options you must specify, with some suggested values, are

Initial values of $x, y, z$	0.0–5.0
Assimilation period (in seconds)	0–10
Forecast period (in seconds)	Any
Time step (in seconds)	0.0–0.05
Frequency of observations (in time steps)	Any
Noise on observations	Variance = 0–2
Convergence criteria	Default values given
Number of outer loops	2 ( <i>Incremental version only</i> )

Note that the time step must be a divisor of your total time, so values such as 0.02, 0.025, 0.05 work well. The output of the program is the fields of  $x$  and  $z$ , the errors in  $x$  and  $z$  and the convergence of the cost function and its gradient. The final norm of the gradient is also output in the Matlab command window.

### 3.4 Suggested exercises

Start with the conditions

truth=(1.0,1.0,1.0)

first guess=(1.2,1.2,1.2)

Assimilation period = 2

Forecast period = 3

Time step = 0.05

Frequency of observations = 2

1. Compare full 4D-Var and incremental 4D-Var for the same total number of iterations, using perfect observations and observations with error. Are there conditions for which one scheme is better than the other?
2. Is it better to have very few accurate observations or more observations which are less accurate? Consider both the accuracy of the analysis and the rate of convergence.
3. Is it better to have a long assimilation window with few observations or a short assimilation window with more observations? Does this depend on how much error there is on the observations? Consider both the accuracy of the analysis and the rate of convergence.
4. How does the rate of convergence change if the first guess is moved closer to or away from the truth?
5. For the incremental 4D-Var investigate the effect of different outer loops with the same total number of iterations. Compare against the full 4D-Var solution.
6. The convergence is stopped when the change in the gradient is less than a given tolerance. Investigate other criteria such as relative change in gradient,

change in cost function, relative change in cost function. What is the best way of stopping the iteration so that a good solution can be achieved with the minimum amount of work?