

User guide for oscillating system and Lorenz models

Amos S. Lawless, University of Reading

1 Introduction

This document explains how to run the simple data assimilation programs for the damped oscillating system and the Lorenz model using *Matlab*. The two routines needed are *os_sys_menu.m* to run the oscillating system and *lorenz_menu.m* to run the Lorenz model. In order to run the routines, the user must start up *Matlab* and then *cd* into the directory in which the routines are stored. The programs are then started by typing *os_sys_menu* for the oscillating system or *lorenz_menu* for the Lorenz model. For each of these two models there are four different data assimilation schemes available. Before explaining the inputs needed for the programs, we outline some details of the models and data assimilation schemes implemented. In doing this we make heavy use of [1] and [2], written by the original author of these programs.

Matlab hint: The up arrow of the cursor keys can be used to recall previous commands. Once the program has been run once, this can be used as a quick way of running it again without typing the name of the program.

2 The models

2.1 Damped oscillating system

The damped oscillating system is given by the linear ordinary differential equation

$$\frac{d^2 x}{dt^2} = -l \frac{dx}{dt} - mx, \quad (1)$$

where l is the damping coefficient, m the square of the frequency and $x = x(t)$. This second order equation can be written as the first order system

$$\begin{pmatrix} dx_1/dt \\ dx_2/dt \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -m & -l \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \quad (2)$$

with $x_1 = x_1(t), x_2 = x_2(t)$. This first order system is then discretized using a second order Runge-Kutta method, which gives the following set of discrete equations:

$$x_1^{k+1} = (1 - m\Delta t^2/2)x_1^k + (\Delta t - l\Delta t^2/2)x_1^k, \quad (3)$$

$$\begin{aligned} x_2^{k+1} &= (ml\Delta t^2/2 - m\Delta t)x_2^k \\ &+ (1 - l\Delta t + l^2\Delta t^2/2 - m\Delta t^2/2)x_2^k, \end{aligned} \quad (4)$$

where Δt is the model time step and k is the time step index. The default values set in the program are $l = 0.1, m = 1.0$ and $\Delta t = 0.1$. The initial conditions are taken to be $x_1^0 = 1.0, x_2^0 = 0.0$. The background trajectory for the data assimilation schemes is generated using the same model parameters, but with initial conditions $x_1^0 = 1.5, x_2^0 = 0.5$.

2.2 Lorenz equations

The Lorenz equations are given by the nonlinear system

$$\frac{dx}{dt} = -\sigma(x - y), \quad (5)$$

$$\frac{dy}{dt} = \rho x - y - xz, \quad (6)$$

$$\frac{dz}{dt} = xy - \beta z, \quad (7)$$

where $x = x(t), y = y(t), z = z(t)$ and the parameters σ, ρ, β are chosen to have the values 10, 28 and 8/3 respectively. The system has three equilibrium points, one at the origin and two at the coordinates

$$(\pm\sqrt{\beta(\rho - 1)}, \pm\sqrt{\beta(\rho - 1)}, \rho - 1). \quad (8)$$

All three equilibrium points are unstable for this choice of parameters.

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

$$\begin{aligned} x^{k+1} &= x^k - \sigma\Delta t/2[2(y^k - x^k) + \Delta t(\rho x^k - y^k - x^k y^k) \\ &- \sigma\Delta t(y^k - x^k)], \end{aligned} \quad (9)$$

$$\begin{aligned} y^{k+1} &= y^k + \Delta t/2[\rho x^k - y^k - x^k z^k + \rho(x^k + \sigma\Delta t(y^k - x^k)) - y^k \\ &- \Delta t(\rho x^k - y^k - x^k z^k) \\ &- (x^k + \sigma\Delta t(y^k - x^k))(z^k + \Delta t(x^k y^k - \beta z^k))], \end{aligned} \quad (10)$$

$$\begin{aligned} z^{k+1} &= z^k + \Delta t/2[x^k y^k - \beta z^k \\ &+ (x^k + \Delta t\sigma(y^k - x^k))(y^k + \Delta t(\rho x^k - y^k - x^k z^k)) \\ &- \beta z^k - \Delta t(x^k y^k - \beta z^k)], \end{aligned} \quad (11)$$

where Δt is the model time step and k is the time step index. The default values set in the program are $\Delta t = 0.01$, with initial conditions $x^0 = 2.0$, $y^0 = 2.0$, $z^0 = 2.0$. The background trajectory for the data assimilation experiments is generated by using initial conditions $x^0 = 1.0$, $y^0 = 1.0$, $z^0 = 1.0$.

3 Data assimilation schemes

There are four sequential data assimilation schemes implemented for each problem. These are successive corrections, optimal interpolation, analysis correction and the Kalman filter. The general sequential data assimilation problem can be considered as the minimization of the cost function

$$\begin{aligned} \mathcal{J} = & (\mathbf{y}(i) - H(\mathbf{x}(i)))^T R^{-1} (\mathbf{y}(i) - H(\mathbf{x}(i))) \\ & + (\mathbf{x}(i) - \mathbf{x}_b(i))^T B^{-1} (\mathbf{x}(i) - \mathbf{x}_b(i)) \end{aligned} \quad (12)$$

with respect to $\mathbf{x}(i)$. Here $\mathbf{x}_b(i)$ is a background state with error covariance matrix B^{-1} , $\mathbf{y}(i)$ is a 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 . Each of the data assimilation methods implemented approximates the solution to this minimization problem. Here we do not aim to give a full explanation of the schemes, but just briefly outline how each is implemented. Further details can be found in [1].

3.1 Successive corrections

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

$$\mathbf{x}^{j+1}(i) = \mathbf{x}^j(i) + W[\mathbf{y}(i) - H(\mathbf{x}^j(i))], \quad (13)$$

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 $W = 0.5 \times I$.

3.2 Optimal interpolation

The optimal interpolation analysis is given by

$$\mathbf{x}_a(i) = \mathbf{x}_b(i) + K[\mathbf{y}(i) - H(\mathbf{x}_b(i))], \quad (14)$$

with

$$K = BH^T(HBH^T + R)^{-1}. \quad (15)$$

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

3.3 Analysis correction

The analysis correction algorithm is written

$$\mathbf{x}^{j+1}(i) = \mathbf{x}^j(i) + WQ[\mathbf{y}^j(i) - H(\mathbf{x}^j(i))], \quad (16)$$

$$\mathbf{y}^{j+1}(i) = \mathbf{y}^j(i) - Q[\mathbf{y}^j(i) - H(\mathbf{x}^j(i))], \quad (17)$$

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 $\mathbf{x}_a(i) = \mathbf{x}^k(i)$.

3.4 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:

$$K(i) = P_f(i)H^T(i)[H(i)P_f(i)H^T(i) + R(i)]^{-1}, \quad (18)$$

$$\mathbf{x}_a(i) = \mathbf{x}_f(i) + K(i)[\mathbf{y}(i) - H(i)\mathbf{x}_f(i)], \quad (19)$$

$$P_a(i) = [I - K(i)H(i)]P_f(i), \quad (20)$$

$$\mathbf{x}_f(i+1) = M(t_{i+1}, t_i)[\mathbf{x}_a(i)], \quad (21)$$

$$P_f(i+1) = M(t_{i+1}, t_i)P_a(i)M(t_{i+1}, t_i)^T + Q(i). \quad (22)$$

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. In the model experiments set up the model error covariance $Q(i)$ is assumed to be zero for the oscillating system problem but is non-zero for the Lorenz model.

4 User inputs

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 exact 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. For the oscillating system the user can choose observations every 1, 2, 5, 10 or 25 time steps. For the Lorenz model the observations can be every 25, 50, 100 or 200 time steps. The total assimilation time is 250 time steps for the oscillating system and 2000 time steps for the Lorenz model.

- 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` and `sc_y_noise.mat` (and `sc_z_noise.mat` for the Lorenz model). ‘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.

5 Output

The output from the programs is graphical, with figures as follows.

5.1 Oscillating system

The output from the oscillating system is two figures. Figure 1 shows the solution for the x_1 variable. The truth trajectory is shown by a black dashed line, the background trajectory used is shown by a blue dashed-dotted line and the observations used are shown by magenta circles. The final analysis and forecast from the analysis is shown by a red line.

Figure 2 shows a plot of the error in the x_1 variable with time, which is the difference of the analysis/ forecast trajectory from the truth.

5.2 Lorenz system

There are three figures output from the program *lorenz_menu*. Figures 1 and 2 show the output for the x and z variables respectively. As for the oscillating system 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.

6 Advanced use

In this section we explain some of the parameters which are set in the code itself and which the user may want to experiment with. Before changing any of the code it is important to *take a copy of the original program* to keep as a backup. We discuss possible changes to the code referring to sections as numbered within the programs.

6.1 Section 2: True solution

The true solution can be changed by changing either the parameters of the problem or the initial conditions. The parameters for the oscillating system problem are clearly marked in this section of the code as l and m , the damping coefficient and square of the frequency of the system, which correspond to the coefficients in (1), and h , the time step for the Runge-Kutta scheme. The initial conditions for x_1 and x_2 are set by the statements 'x(1,1)= ' and 'x(2,2)= ' respectively.

For the Lorenz model the time step of the scheme is again denoted h . We also have three parameters in the code, 's', 'r' and 'b', corresponding to σ , ρ and β in (5), (6) and (7). The initial conditions for x, y and z are set by 'x(1)', 'y(1)' and

‘z(1)’. The user should be aware that the Lorenz model will be much more sensitive to small changes in the parameters than the oscillating system.

6.2 Section 3: Background solution

The background solution for both models is obtained by running the model from slightly different initial conditions. These are set in Section 3 of the code and can be changed by the user. An alternative method for generating the background would be to start the model using the true initial conditions with some random noise added.

6.3 Section 5: Error covariance matrices

Section 5 of the code sets up the observation and background error covariance matrices, R and B . These are used in all schemes except the successive correction scheme. If the correct weighting matrices are requested then the true covariances are calculated using the known solution, otherwise these matrices are set to the identity. Alternatives could be investigated by the user.

6.4 Section 6: Successive corrections weighting matrix

The weighting matrix W for the successive corrections scheme is set in the code to be 0.5 times the identity matrix. the performance of the scheme using other weighting matrices may be investigated.

References

- [1] M.J. Martin, N.K. Nichols and M.J. Bell, *Treatment of Systematic Errors in Sequential Data Assimilation*. Met Office Ocean Applications Technical Note, No. 21, July 1999.
- [2] M.J. Martin, *Data Assimilation in Ocean Circulation Models with Systematic Errors*. PhD Thesis, Department of Mathematics, The University of Reading, 2001.