4D-Var data assimilation with the Lorenz equations

Data Assimilation Research Centre (DARC)
University of Reading, UK

Contents

1 The Lorenz System **1**

2 Four-dimensional variational data assimilation (4D-Var) **2**

2.1 Introduction 2
2.2 Test routines: building a 4D-Var system 3
2.2.1 Test of tangent linear model correctness 3
2.2.2 Test of adjoint model 3
2.2.3 Gradient test 3
2.3 Assimilation set up 4
2.4 Suggested exercises 4
2.5 Advanced exercises 5

1 The Lorenz System

In this practical we explore the application of the full and incremental 4D-Var data assimilation algorithms to the Lorenz 1963 equations, a simple dynamical model with chaotic behaviour. The Lorenz equations are given by the nonlinear system

$$\frac{dx}{dt} = -\sigma(x-y), \quad (1)$$
$$\frac{dy}{dt} = \rho x - y - xz, \quad (2)$$
$$\frac{dz}{dt} = xy - \beta z, \quad (3)$$

where $x = x(t)$, $y = y(t)$, $z = z(t)$ and σ, ρ, β 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:

$$
\begin{aligned}
x_{k+1} &= x_k + \sigma\frac{\Delta t}{2}\left[2(y_k - x_k) + \Delta t(\rho x_k - y_k - x_k z_k)\right. \\
&- \left.\sigma\Delta t(y_k - x_k)\right], \qquad (4) \\
y_{k+1} &= y_k + \frac{\Delta t}{2}\left[\rho x_k - y_k - x_k z_k + \rho(x_k + \sigma\Delta t(y_k - x_k)) - y_k\right. \\
&- \Delta t(\rho x_k - y_k - x_k z_k) \\
&- \left.(x_k + \sigma\Delta t(y_k - x_k))(z_k + \Delta t(x_k y_k - \beta z_k))\right], \qquad (5) \\
z_{k+1} &= z_k + \frac{\Delta t}{2}\left[x_k y_k - \beta z_k\right. \\
&+ (x_k + \Delta t\sigma(y_k - x_k))(y_k + \Delta t(\rho x_k - y_k - x_k z_k)) \\
&- \left.\beta z_k - \Delta t(x_k y_k - \beta z_k)\right], \qquad (6)
\end{aligned}
$$

where Δt is the model time step and k is the time step index.

2 Four-dimensional variational data assimilation (4D-Var)

2.1 Introduction

The 4D-Var schemes in these programs minimize a cost function of the form

$$
\mathcal{J}(\mathbf{x}_0) = \frac{1}{2}(\mathbf{x}_0 - \mathbf{x}_0^b)^T\mathbf{B}^{-1}(\mathbf{x}_0 - \mathbf{x}_0^b) + \frac{1}{2}\sum_{i=0}^{n}(h_i(\mathbf{x}_i) - \mathbf{y}_i)^T\mathbf{R}^{-1}(\mathcal{H}_i(\mathbf{x}_i) - \mathbf{y}_i), \qquad (7)
$$

where we assume that $\mathbf{B} = \sigma_b^2\mathbf{I}$ and $\mathbf{R} = \sigma_o^2\mathbf{I}$. 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 Matlab routines for both types of schemes. The routines used are as follows:

lorenz4d.m	Top level routine for full 4D-Var
lorenz4d_inc.m	Top level routine for incremental 4D-Var
calcfg.m	Calculate cost function and its gradient for full 4D-Var
calcfg_inc.m	Calculate cost function and its gradient for incremental 4D-Var
modeuler.m	Nonlinear model for Lorenz system
modeuler_tl.m	Tangent linear model
modeuler_adj.m	Adjoint model
test_tl.m	Test tangent linear model
test_adj.m	Test adjoint model
test_grad.m	Test of *calcfg*
test_gradinc.m	Test of *calcfg_inc*
menu_asl	Used to provide menus

2.2 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.

2.2.1 Test of tangent linear model correctness

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

$$\mathcal{M}(\mathbf{x}+\gamma\delta\mathbf{x}) - \mathcal{M}(\mathbf{x}) \approx \mathbf{M}(\mathbf{x})\gamma\delta\mathbf{x}\,. \tag{8}$$

Hence if we plot the *relative error*

$$E_R = 100\frac{||\mathcal{M}(\mathbf{x}+\gamma\delta\mathbf{x}) - \mathcal{M}(\mathbf{x}) - \mathbf{M}(\mathbf{x})\gamma\delta\mathbf{x}||}{||\mathbf{M}(\mathbf{x})\gamma\delta\mathbf{x}||}\,, \tag{9}$$

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

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 *(remember to save a copy first!)*.

2.2.2 Test of adjoint model

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

$$< \mathbf{M}\delta\mathbf{x}, \mathbf{M}\delta\mathbf{x} >=< \delta\mathbf{x}, \mathbf{M}^T\mathbf{M}\delta\mathbf{x} > \tag{10}$$

for any inner product $<,>$ 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.

2.2.3 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

$$\Phi(\alpha) = \frac{\mathcal{J}(\mathbf{x}+\alpha\mathbf{h}) - \mathcal{J}(\mathbf{x})}{\alpha\mathbf{h}^T\nabla\mathcal{J}(x)} = 1 + O(\alpha), \tag{11}$$

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 α (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 the coding of either the full or incremental 4D-Var cost function gradient. 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.

2.3 Assimilation set up

The routines used to run assimilation experiments are *lorenz4d* for the full 4D-Var and *lorenz4dinc* for the incremental 4D-Var algorithm. The menu options you must specify (with some suggested values given) 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 background	Variance = 0–4 (excluding zero)
Noise on observations	Variance = 0–4 (excluding zero)
Convergence criteria	Default values given
Number of outer loops	2 (*incremental version only*)

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.

The noise on the background and observations is produced randomly each time the program is run. In order to compare the effect of different settings you can choose to use the same realisation of random noise as in your previous experiment by answering 'Yes' to the question 'Read in noise from file?'. Note that in order for this to work the number of observations must remain the same.

2.4 Suggested exercises

Try starting with

truth = (1.0, 1.0, 1.0)
assimilation period = 2
forecast period = 3
time step = 0.05
frequency of observations = 2.

Consider the following questions:

1. Run the 4D-Var with different relative errors on the background and observations. How does the behaviour of the scheme change?
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 background error is decreased or increased so that the first guess is moved closer to or away from the truth?

5. Compare full 4D-Var and incremental 4D-Var for the same total number of iterations. Are there conditions for which one scheme is better than the other?

6. For the incremental 4D-Var investigate the effect of multiple outer-loops with the same total number of iterations. Compare against the full 4D-Var solution.

2.5 Advanced exercises

The following exercises require you to understand and change the code.

1. Investigate the performance of 4D-Var when the model state is only partially observed:

   Change the code so that only two components of the state vector are observed. How well is the other component retrieved by the assimilation? Compare the effect of using a diagonal and non-diagonal background error covariance matrix.

2. Investigate the effect of correlated observation errors in 4D-Var:

   Introduce correlations in your observation errors by changing the program so that the same random noise is used to create the observation error for x, y and z (or just two of these). How does this affect the assimilation results? Consider what happens when the observation error covariance matrix is assumed to be diagonal and not.

3. Investigate the effect of biased observations in 4D-Var:

   Try replacing the random observation error with a constant bias for one or more of the variables. What is the effect on the analysis?

4. Investigate the effect of model error in 4D-Var:

   Usually the numerical model we use to assimilate is not an exact representation of the true system, but will contain model errors. We can investigate the effect of this in a simple assimilation experiments by using one version of the model to produce the 'truth' trajectory and the observations, and using a different version of the model in the assimilation. For example, you could consider

   (a) random stochastic error - add random forcing to one of the equations in the assimilation model;
   (b) an error in the parameters - change one of the model parameters σ, β or ρ to be slightly different in the assimilation model;
   (c) a bias error - add a constant forcing to one of the equations in the assimilation model.

   How does this affect the analysis?