Problem Set 5

1 Problem Set 5

1. (a) Consider the 3-component Lorenz equations:

$$\begin{aligned} \frac{dx}{dt} &= \sigma(y-x), \\ \frac{dy}{dt} &= \rho x - y - xz, \\ \frac{dz}{dt} &= xy - \beta z, \end{aligned} \tag{1}$$

where σ, ρ and β are real constants. The cost function which defines the misfit between model state and observations is given by

$$J = \frac{1}{2}[(x-x^{obs})^2 + (y-y^{obs})^2 + (z-z^{obs})^2.]$$

What is the adjoint system of equations? First write the Tangent Linear system, then take the transpose of the transition matrix and add forcing by the gradient of the cost function, J.

(b) Consider the nonlinear advection-diffusion in one-dimension:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \nu\frac{\partial^2 u}{\partial x^2} = 0 \tag{2}$$

over the domain $[0, L] \times [0, T]$. The cost function is given by:

$$J = \frac{1}{2}\int_0^T \int_0^L (u - u^{obs})^2 dx dt.$$

We are now going to derive the adjoint equation following the control theory method of section 7.2.2.

(i) Write the TLM equation.
(ii) Write the expression for δJ.
(ii) Multiply the TLM equation by an arbitrary variable p(x,t) and integrate over the domain in x and t. For the time derivative term, integrate by parts first in time. For the x derivative term, integrate by parts in x first. For the term involving a second derivative, you will need to integrate by parts twice. Eliminate some terms using the following boundary conditions on p:

$$p(x,T) = 0, \qquad p(0,t) = p(L,t), \qquad \frac{\partial p}{\partial x}(0,t) = \frac{\partial p}{\partial x}(L,t)$$

Define the adjoint equation including forcing by the gradient of J.

2. Find the adjoint of **dum.m** by reversing the code. Test the adjoint routine and verify that it is correct to at least 14 digits. You will receive **dum.m** by email, but here are the contents of this file:

```
function y=dum(x,n)

y=zeros(1,n);
z=zeros(1,2*n);

for i=1:n-1
   y(i)=x(i+1);
end

for i=1:n
   z(i)=x(i);
end

for i=1:n
   z(i+n)=x(i);
end

for i=1:n
   y(i)=y(i) + 11*z(i) + 6*z(n+i);
end
```

3. Scalar 4DVAR. Consider the following scalar system:

$$\mathrm{x}_{k+1} = m\mathrm{x}_k + \mathrm{w}_k \tag{3}$$

$$\mathrm{z}_k = \mathrm{H}\mathrm{x}_k + \mathrm{v}_k \tag{4}$$

Let P_0 be the initial state error variance and R be the obs error variance. Then the cost function is given by:

$$J(\mathrm{x}_0) = \frac{1}{2\mathrm{P}_0}(\mathrm{x}_0 - \mathrm{x}_0^f)^2 + \frac{1}{2\mathrm{R}}\sum_{k=1}^{K}(\mathrm{z}_k - \mathrm{H}\mathrm{x}_k)^2.$$

The gradient is given by

$$\nabla_{\mathrm{x}_0} J = \frac{1}{\mathrm{P}_0}(\mathrm{x}_0 - \mathrm{x}_0^f) - \sum_{k=1}^{K} m_0 m_1 \ldots m_{k-1}\mathrm{H}\mathrm{R}^{-1}(\mathrm{z}_k - \mathrm{H}\mathrm{x}_k)$$

where we used the fact that the transpose of a scalar is the scalar itself.

In this problem, we will write a MATLAB script to run the 4DVAR scheme. The assimilation period will be timesteps.

1. What is the Hessian of J? You may use the fact that m is independent of timestep k.

2. Show by hand that the adjoint test is passed. In other words, let x be the model input and y be the model output after k time steps, while $\hat{x}$ is the adjoint model output and $\hat{y}$ is the adjoint model input. Then show that

$$< x, \hat{x} >=< y, \hat{y} > .$$

3. Here are some steps to help you write the 4DVAR script. Note that the first 2 steps (initialization statements) are the same as in the scalar KF problem of the last chapter. Note also that the instructions may seem laborious (given that simplifications are possible due to the specific model used) but this will give you the basic 4DVAR coding recipe for any model.

(a) Define parameters and constants.

(b) Define initial conditions for the background and truth.

(c) Loop in time for nk timesteps. First generate the obs by perturbing the truth. Then propagate the truth in time

(d) Compute the cost function. To do this, define a separate function to evaluate the cost given an initial state. This will involve integrating the background state in time.

(e) Compute the gradient.

   i. First copy your script to evaluate the cost function, renaming it to the gradient function.

   ii. Take the derivative of the background term.

   iii. Set the adjoint variable at nk+1 to zero. Set the gradient equal to the adjoint variable at nk+1.

   iv. run the model to get the background trajectory.

   v. loop backwards in time from nk to 1. Within the loop (i) add the contribution to the gradient for time step k, then (ii) run the adjoint model backward one timestep using the current gradient as the initial condition. Overwrite the gradient with the output.

(f) Obtain the solution using Newton's method. Use the Hessian obtained in (a). Because the cost function is purely quadratic, Newton's method will obtain the minimum in 1 iteration.

(g) Integrate the solution to get the final trajectory (the analysis).

(h) Plot background, true and obs as a function time step. Plot the actual error ($|\mathrm{x}_k^a - \mathrm{x}_k^t|$) as a function of time.

(i) Be prepared to hand-in a printout of your code, or email it to Lisa.

4. Run the 4DVAR script for R=H=P_0=m=1. Let Q=0 and nk=50. Explain the results. Now try Q=1. Explain what is happening in the plots and how the solution differs from the scalar KF example for the same parameters. Note that the model is assumed to be perfect.

5. Now see what happens when the model is not perfect. When you generate the obs, use $m^t = m + 0.05 = 1.05$. Explain the results. Keep the parameter values used in part 4 with Q=1.

6. Finally, imagine what would happen with a different assimilation period, say np=10. Then there would be 5 cycles during the 50 time steps. (You may assume the model is perfect.)

Without actually coding anything (it would be too much work I think), draw a schematic of what you think the plot of analysis, truth and obs would look like. After each cycle, the initial background would be taken from the analysis at the last time step of the previous cycle.

4. MATLAB: Passive advection in 1D using 4DVAR. Let us return to the advection problem of Ch. 3, prob. 4. and Ch. 5 prob. 3. This time we will run 4DVAR instead of a KF or OI algorithm. To run 4DVAR you will need the additional MATLAB scripts: testadj.m, cost.m, gradcost.m, upwindad.m, 4dvar.m.

   (a) Note that the forecast model (for the 1D passive advection) is linear. Thus, there is no need to linearize the model. However, we need an adjoint model for 4DVAR. This is provided in upwindad.m . Once an adjoint is developed, we need to test it to see that it is correct to machine precision. To test the adjoint you will need to run testadj.m. However, you must first complete this code. Two lines near the bottom have been commented out. Uncomment these and complete the equations. Run testadj.m. Provide a hardcopy of your results when you have verified that the adjoint is correct.

   (b) Cost function and gradient. Because our model is linear, the cost function is purely quadratic:

$$J(\mathbf{x}_0) = \frac{1}{2}\sum_{k=0}^{N}(\mathbf{z}_k - \mathbf{H}\mathbf{x}_k)^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{z}_k - \mathbf{H}\mathbf{x}_k) + (\mathbf{x}_0 - \hat{\mathbf{x}}_0^f)^{\mathrm{T}}(\mathbf{P}_0^f)^{-1}(\mathbf{x}_0 - \hat{\mathbf{x}}_0^f). \quad (5)$$

   The cost function is coded in cost.m. Read through this code and then complete the missing two lines. The gradient of this cost function is

$$\nabla J(\mathbf{x}_0) = -\sum_{k=0}^{N}\mathbf{M}_0^{\mathrm{T}}\mathbf{M}_1^{\mathrm{T}}\ldots\mathbf{M}_{k-1}^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{z}_k - \mathbf{H}\mathbf{x}_k) + (\mathbf{P}_0^f)^{-1}(\mathbf{x}_0 - \hat{\mathbf{x}}_0^f). \quad (6)$$

   where we assumed the model was linear and given by:

$$\begin{aligned}\mathbf{x}_k &= \mathbf{M}_{k-1}\mathbf{x}_{k-1} \\ &= \mathbf{M}_{k-1}\mathbf{M}_{k-2}\ldots\mathbf{M}_0\mathbf{x}_0 \end{aligned} \quad (7)$$

   and the adjoint model is

$$\delta\mathbf{x}_{k-1} = \mathbf{M}_{k-1}^{\mathrm{T}}\delta\mathbf{x}_k \quad (8)$$

   or

$$\delta\mathbf{x}_0 = \mathbf{M}_0^{\mathrm{T}}\mathbf{M}_1^{\mathrm{T}}\ldots\mathbf{M}_{k-2}^{\mathrm{T}}\mathbf{M}_{k-1}^{\mathrm{T}}\delta\mathbf{x}_k \quad (9)$$

   for a Euclidean norm (adjoint = transpose). Read through gradcost.m and complete the missing two lines of code.

   (c) Gradient test. Note that the gradient test is already in the var4d.m code. Run `var4d(0,1,0.95,0)`. Try various observation frequencies:20, 10, 5, 2, 1. Choose an observation pattern of 1, (obs every grid point), and an obs error of 0.02. Look for the output of the gradient test. Mathematically explain what this test is doing. (Hint: Consider a Taylor expansion of $J(\mathbf{x}+\alpha\delta\mathbf{x}) - J(\mathbf{x})$.

Then choose $\delta\mathbf{x} = \nabla J(\mathbf{x})$.) Why do we need this test, if we already tested the adjoint for accuracy?

(d) 4DVAR minimization. Normally, we can use some packaged software (based on a quasi-Newton or conjugate gradient method for example) for the minimization. MATLAB's basic package does not have an optimization routine that also uses the gradient information. However, because our cost function is purely quadratic, we can write our own minimization algorithm based on Newton's method. Read the section below on Newton's method. Newton's method requires knowledge of the Hessian of the cost function. To approximate the Hessian, we simply perturbed the gradient. We approximate the jth column of the Hessian matrix by a finite difference:

$$(J'')e_j = \frac{1}{\alpha}[\nabla J(\mathbf{x}_g + \alpha e_j) - \nabla J(\mathbf{x}_g)] \tag{10}$$

where $\mathbf{x}_g$ is the guess or background state. e_j is the jth column of the identity matrix. I used α=10^{-3}.

(e) Run the 4DVAR by typing `var4d(0,1,0.95,0)`. The Courant number will now be fixed at 0.95 and T_{final}=1 as in the OI problem. The same three questions will be asked. Hitting "return" will give the default. First enter observation frequency of 5 (obs every 5 time steps), an observation sparsity of 1 (obs at every gridpoint), and "return" for the obs error standard deviation. This will give the default value of 0.02. How does the analysis compare with the truth? Note that there is no error estimate. A bit more work is required to come up with an error estimate and this was not done. Unlike the KF, the analysis error covariance matrix is not part of the algorithm for 4DVAR.

(f) Now let's make the problem a little harder. Again type `var4d(0,1,0.95,0)`, but provide an obs error of 0.2. Keep the obs frequency of 5 and the obs sparsity of 1. Compare your results with those of the Kalman filter.

(g) Now let's see what happens when there are data gaps. Type `var4d(0,1,0.95,0)`, but answer "return" to all questions. This gives an obs every time step, over the left half of the domain with a std deviation of 0.02. How the analysis fare? Now decrease the observation frequency by typing first 2 then 5 and keeping the obs pattern and error std dev the same as before. Now what happens to the solution? Why? Again compare your results to the KF solution and discuss what you see.

Newton method solution for a quadratic cost function

Consider a quadratic cost function:

$$J(\mathbf{x}) = \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x} + \mathbf{b}^{\mathrm{T}}\mathbf{x} + c. \tag{11}$$

The gradient is

$$\nabla J(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b}. \tag{12}$$

The Hessian or second derivative is

$$J''(\mathbf{x}) = \mathbf{A}. \tag{13}$$

Since we want to minimize (11), we want to solve for $\nabla J(\mathbf{x}) = \mathbf{0}$. Now for an initial guess $\mathbf{x}_g$, we have that

$$\nabla J(\mathbf{x}_g) = \mathbf{A}\mathbf{x}_g + \mathbf{b} \tag{14}$$

while at the solution, $\hat{\mathbf{x}}$, the gradient is zero, i.e.

$$\nabla J(\hat{\mathbf{x}}) = 0 = \mathbf{A}\hat{\mathbf{x}} + \mathbf{b}. \tag{15}$$

Subtracting (14) - (15) yields:

$$\nabla J(\mathbf{x}_g) = \mathbf{A}(\mathbf{x}_g - \hat{\mathbf{x}}). \tag{16}$$

Solving for $\hat{\mathbf{x}}$, we obtain

$$\hat{\mathbf{x}} = \mathbf{x}_g - \mathbf{A}^{-1}\nabla J(\mathbf{x}_g) \tag{17}$$

or on substituting the Hessian for $\mathbf{A}$:

$$\hat{\mathbf{x}} = \mathbf{x}_g - (J'')^{-1}\nabla J(\mathbf{x}_g). \tag{18}$$