

PHY 2506, Fall 2015
Problem Set 3
Assigned Nov 21st; Due Dec 2nd.

1. Let us repeat the 1D advection problem using a Kalman filter instead of an OI scheme. Obtain kf.m for this MATLAB exercise.

(a) Compare kf.m and oi.m. What is the difference between these two algorithms?

(b) Run the KF by typing kf(0,1,0.95,0). The Courant number will be fixed at 0.95 and $T_{\text{final}} = 1$, as before. The same three questions will be asked. Hitting “return” will give the default. First enter observation frequency of 5 (obs every 5 timesteps), 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? Does the error estimate make sense?

(c) Now let’s make the problem a little harder. Again, type kf(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. What happened to the analysis and the error estimate?

(d) Now let’s see what happens when there are data gaps. Type kf(0,1,0.95,0), but answer “return” to all questions. This gives an observation every time step, over the left half of the domain with a standard deviation of 0.02. How does the analysis fare? Now decrease the observation frequency by typing first 2 then 5 and keeping the observation pattern and error standard deviation the same as before. Now what happens to the solution? Why?

(e) Compare your OI and KF solutions for parts (b)-(d). Discuss what you see.

2. Let us repeat the 1D advection problem using a 4Dvar scheme. To run 4Dvar you will need the additional MATLAB routines: testadj.m, cost.m, gradcost.m, upwindad.m, 4dvar.m

(a) Note that the forecast model for the 1D passive advection case 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:

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

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

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

where we assume the model is linear and given by

$$\begin{aligned} \mathbf{x}_k &= \mathbf{M}_{k-1} \mathbf{x}_{k-1} \\ &= \mathbf{M}_{k-1} \mathbf{M}_{k-2} \dots \mathbf{M}_0 \mathbf{x}_0 \end{aligned}$$

and the adjoint model is

$$\delta \mathbf{x}_{k-1} = \mathbf{M}_{k-1}^T \delta \mathbf{x}_k$$

or

$$\delta \mathbf{x}_0 = \mathbf{M}_0^T \mathbf{M}_1^T \dots \mathbf{M}_{k-1}^T \mathbf{M}_{k-1}^T \delta \mathbf{x}_k$$

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

4DVAR Minimization Note: Normally, we can use some packaged software (based on quasi-Newton or conjugate gradient methods, 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. As described in Question 2 on Problems Set 2, Newton's method requires knowledge of the Hessian of the cost function. To approximate the Hessian we simply perturbed the gradient. We approximate the j th column of the Hessian matrix by a finite difference:

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

where \mathbf{x}_g is the guess of background state and e_j is the j th column of the identity matrix. I used $\alpha = 10^{-3}$.

- (c) Gradient test. Note that the gradient test is already in the `var4d.m` code. Run `var4d(0,1,0.95,0)`. Try observations 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) For the case with an obs frequency of 5 in part (c), how does the analysis compare with the truth? Note, 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.
- (e) Now let's make the problem 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.
- (f) Now let's see what happened where 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 deviations of 0.02. How does the analysis fare? Now decrease the observation frequency by typing first 2 then 5 and keeping the obs pattern and error std deviation the same as before. Now what happened to the solution? Why? Again, compare your results to the KF solution and discuss what you see.