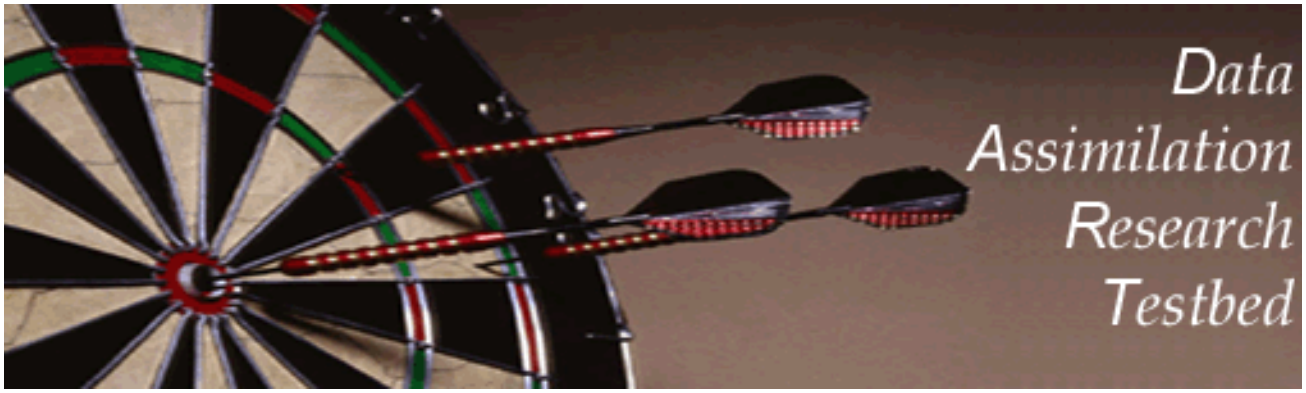
Data Assimilation Research Testbed Tutorial



Section Carbon: A Simple 1D Advection Model

Version 2.0: June, 2007

Overview:

1. One dimensional periodic domain.

2. Chaotic/stochastic model for wind.

3. Single passive tracer advected by wind.

4. Stochastic model for tracer source can be added.

5. Diurnal cycle for tracer source can be added.

6. Model for phase offset of diurnal cycle can be added.

Model domain, timestepping, and namelist controls (in italics):

1. One-dimensional, periodic grid on domain [0, 1].

2. Equally-spaced gridpoints, number controlled by *num_grid_points*.

3. Location of gridpoints is 0, 1/N, 2/N,..., N-1/N; N=*num_grid_points*.

4. Distance between gridpoints is *grid_spacing_meters*.

5. Time-stepping controlled by *time_step_days*, *time_step_seconds*.

The wind model: Burger's Equation

1. Wind equation is: $\frac{\partial u}{\partial t} = u\frac{\partial u}{\partial x}$.

2. Continuous solution forms shocks when u varies with x.

3. Use upstream Lagrangian differencing:
   a. Let u_i be velocity at gridpoint i located at x_i; Δt is timestep.
   b. Source location for wind is: $x_s = x_i - u_i \Delta t$.
   c. Linearly interpolate u to x_s; this is new wind at gridpoint i.

4. This is heavily damping.

5. Generally won't form shocks, even when analytic solution should.

The wind model (cont.) and namelist controls (in italics)

1. Without something additional, damps to spatially constant.

2. Randomly perturb each gridpoint value of u every timestep:

$$u_i = u_i + N(0, W\Delta t), \quad i = 1,..., N$$

W = *wind_random_amp*

3. Damp the spatial mean wind field to M = *mean_wind:*

$$\bar{u}(t + \Delta t) = \bar{u}(t) - W_D \Delta t[\bar{u}(t) - M]$$

W_D is *wind_damping_rate*

Trying out the wind model:

1. Run *workshop_setup.csh* in models/simple_advection/work.

2. Run *perfect_model_obs* and use ncview to check out the winds
   (ignore the other fields for now).

3. Make sure you understand the spatial/temporal behavior of winds.

4. Play with *mean_wind*, *wind_damping_rate*, *wind_random_amp*
   to get different types of flows.

5. Having winds reverse will be interesting for some tracer cases.

The passive tracer (CO_2) field and namelist control (in italics):

1. A single passive (doesn't affect wind) tracer is advected by the wind.

2. Have tracer concentration, C_i, at each gridpoint.

3. Semi-lagrangian upstream time differencing (same as for wind).

4. There is a tracer source/sink rate, S_i, at each gridpoint.

5. Also a background *D=destruction_rate* (percent / second).

$$C_i = C_i + S_i \Delta t - D \Delta t C_i$$

An assimilation example:

1. Observations (20 altogether):
   Ten equally-space co-located observations of u and C.
   Located halfway between gridpoints (0.05, 0.15,..., 0.95).
   Forward observation operator is linear interpolation (mean).
   u observations have error variance of 16.01.
   C observations have error variance of 1000.2.
   (Apparently insignificant decimals make changing these easier)

2. Base namelist assimilates both types of observations.

3. Run *./perfect_model_obs* followed by *./filter*.
   (Make sure to use base *input.nml*, etc.).

4. Use matlab to examine behavior of the assimilation.
5. Keep track of errors for concentration and wind fields.

An Assimilation Example (continued)

1. Look at impact of using subsets of the observations.

2. Observations being used are controlled in namelist:
   In *obs_kind_nml*.
   Can list a sequence of observation types after

```
assimilate_these_obs_types = 'TRACER_CONCENTRATION',
                             'VELOCITY',
evaluate_these_obs_types = 'TRACER_SOURCE'/
```

   Change to assimilating only wind observations:

```
assimilate_these_obs_types = 'VELOCITY',
evaluate_these_obs_types = 'TRACER_CONCENTRATION',
                           'TRACER_SOURCE'/
```

3. Try assimilating only wind, only concentration and not assimilating anything. Record results and understand what's going on.

Tracer source model and namelist control:

1. So far, tracer source has been constant in time (not in space).

2. Make the source model a damped random walk at each gridpoint.
   Add random noise to each gridpoint at each timestep.
   Also damp to a time mean value.

$$S_i = S_i + N(0, S_R) - S_D[S_i - \bar{S}_i]$$

where S_R = *source_damping_rate*

and $\bar{S}_i$ is time mean value for source at this gridpoint.

$\bar{S}_i$ is actually a 4th set of state variables (defined at each gridpoint).
We will NOT work with this set of variables at this workshop.

Assimilations with a time-varying tracer source:

1. Set *source_random_amp_frac* = 0.000002 in *model_nml*.

2. Copy *perfect_ics_source_noise* and *filter_ics_source_noise* to *perfect_ics* and *filter_ics*.

3. Repeat the assimilations with different classes of observations.

4. This time, also keep track of errors for the source.

5. What happens if *source_random_amp_frac* = 0.00001?

Can you find ways to improve filter performance in these cases?

Diurnal component for tracer source model:

1. *source_diurnal_rel_amp* controls a diurnal component of source.

2. Controls amplitude of diurnal component as fraction of source value.

3. Turn this on and try assimilation experiments again.

   Use *perfect_ics_diurnal* and *filter_ics_diurnal*.
   (Copy them to *perfect_ics* and *filter_ics*)

   Set *source_diurnal_rel_amp* to 0.05.

   Keep *source_random_amp_frac* at 0.00001.

Impact of observation density.

1. Replace *set_def.out* and *obs_seq.in* with
   set_def.out.half and *obs_seq.in.half*

2. These have every other observation from the original set.

3. Do the different assimilation cases.

4. What happens if observations get even sparser? (You'll have to do your own observation sequence design for this).

Using assimilation to acquire information about the source model.

Change back to the denser *obs_seq.in* and *set_def.out* files

svn revert obs_seq.in set_def.out

Set *source_diurnal_rel_amp* = 0.05
Get *filter_ics*, *perfect_ics* from original (*revert filter_ics perfect_ics*)

Run *./perfect_model_obs*

Add some system noise to phase (*source_phase_noise* = 0.00001)
Run *./filter*

Assimilating for sources with lots of structure

Set *source_phase_noise* = 0.0

Copy *perfect_ics.saw* and *filter_ics.saw* to *perfect_ics* and *filter_ics*

The source has large mean value at points 1, 3 and 5, small elsewhere.

Can one resolve this spatial structure?

Problems to explore (or come up with your own):

1. Pick one of the configurations above. Suppose that wind observations cost $10 and concentration observations cost $20. If you have $200 to spend on observations (that MUST be located halfway between the gridpoints), what is the best observing/assimilation system you can design? The error variance of the observations is fixed as in the original cases. You can put multiple observations of any kind at an observing location. The metric of success is the smallest total error for the concentration. What if the metric is the smallest total error for the source?

2. Suppose you can only take observations once a day but the source has a fixed diurnal component. Is there any way to learn anything about the diurnal component? You can use as many observations as you want for this. Ask how to change the observation frequency.

3. For the saw tooth source pattern, what happens as the number of observing locations goes down in this case?

4. What happens to the quality of the assimilations in some of the cases above if the *destruction_rate* is made smaller or bigger (it's already pretty big in some sense)?

5. We might be happier if tracer were not created or destroyed by the assimilation. Is the assimilation creating and destroying tracer? Is there any systematic component to this? Can you find ways to reduce creation/destruction by the assimilation scheme?

6. What happens if we use a fixed-lag smoother in this problem? How should the smoother be tuned? What happens to tracer destruction/creation in the smoother estimates?

7. Can you think of good ways to add model error into this system? If you do, what happens to the quality of the assimilations for various types of experiments? Simulating model error in a meaningful way is essential to predicting what will happen when one switches to real observations.