Adjoint Model Basics

JCSDA Summer Colloquium
Tuesday 4 August 2015

Daniel Holdaway
NASA Global Modeling and Assimilation Office
dan.holdaway@nasa.gov

Outline

- Introduction
- Derivation of the linearized model
- How to write and adjoint model
- Validation of the linearized model
- Practical issues
- Sensitivity examples
- Optimal perturbations

Introduction

What is the adjoint?

- It's a version of our model that propagates 'information' backwards in time

Why do we need it?

- 4DVAR
- Observation impact calculations
- Sensitivity studies
- Singular vector calculations
- Helps us understand the model

Outline

- Introduction
- Derivation of the linearized model
- How to write and adjoint model
- Validation of the linearized model
- Practical issues
- Sensitivity examples
- Optimal perturbations

Derivation of the linearized model

We have a, most likely, nonlinear model,

$$y_i = m\left(x_1, x_2, ..., x_j\right),$$

where x and y are discrete model variables at an old and new time.

Derivation of the linearized model

We have a, most likely, nonlinear model,

$$y_i = m\left(x_1, x_2, ..., x_j\right),$$

where x and y are discrete model variables at an old and new time.

From a DA perspective we may ask a couple of questions:

1. How does y_i change with respect to x_j? If we make perturbations to x_j how do they affect y_i? Suppose x_j has some error. How does that error grow?

Derivation of the linearized model

We have a, most likely, nonlinear model,

$$y_i = m\left(x_1, x_2, ..., x_j\right),$$

where x and y are discrete model variables at an old and new time.

From a DA perspective we may ask a couple of questions:

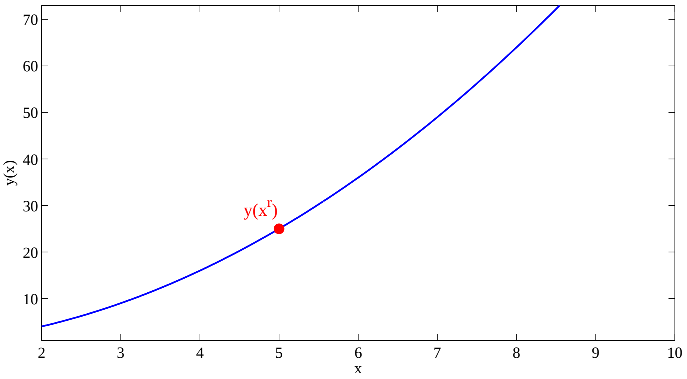
1. How does y_i change with respect to x_j? If we make perturbations to x_j how do they affect y_i? Suppose x_j has some error. How does that error grow?
2. We're analyzing something about y_i. This could be an error or just some interesting physical behavior. What specifically was it about x_j that led to this?

Derivation of the linearized model

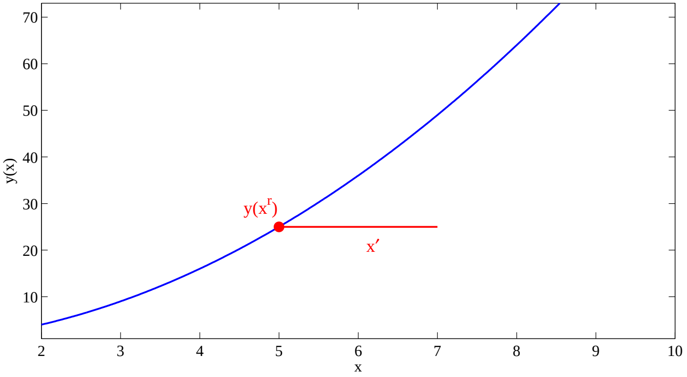
We're trying to get a handle on perturbations so we 'linearize' model variables by separating into reference and perturbation parts,

$$x_j = x_j^r + x_j'.$$

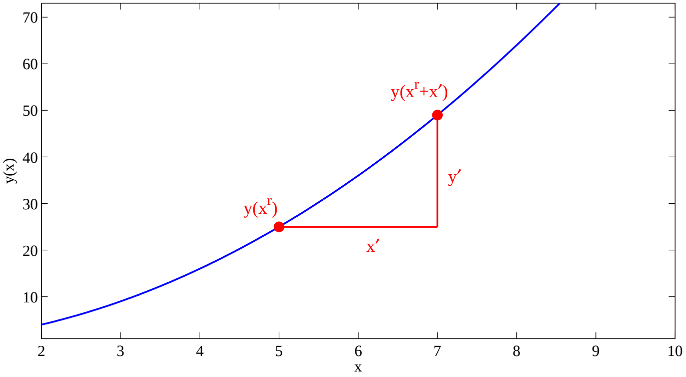
Derivation of the linearized model



Derivation of the linearized model



Derivation of the linearized model



Derivation of the linearized model

We're trying to get a handle on perturbations so we 'linearize' model variables by separating into reference and perturbation parts,

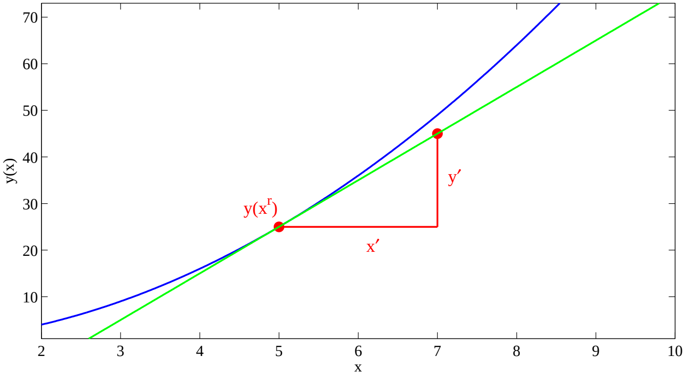
$$x_j = x_j^r + x_j'.$$

The output perturbation can be approximated by Taylor expansion,

$$\begin{aligned} y_i' &= m\left(x_1^r + x_1', x_2^r + x_2', \ldots\right) - m\left(x_1^r, x_2^r, \ldots\right), \\ &= \cancel{m\left(x_1^r, x_2^r, \ldots\right)} + \frac{\partial y_i^r}{\partial x_1} x_1' + \frac{\partial y_i^r}{\partial x_2} x_2' + \ldots - \cancel{m\left(x_1^r, x_2^r, \ldots\right)}, \\ y_i' &= \sum_j \left(\frac{\partial y_i}{\partial x_j}\right)^r x_j'. \end{aligned}$$

The above equation is the **tangent linear model (TLM)** and gives an approximation for the growth of perturbations x_j'.

Derivation of the linearized model



Derivation of the linearized model

The TLM is a useful tool but it doesn't help us with our second question! To get somewhere towards doing so we first simplify things by introducing a scalar measure of the outputs, $J = J\left[\mathbf{y}\left(\mathbf{x}\right)\right]$.

Using the Taylor series again,

$$J' = \sum_i \frac{\partial J^r}{\partial y_i} y'_i,$$

or,

$$J' = \sum_j \frac{\partial J^r}{\partial x_j} x'_j.$$

The two estimates are equivalent when both functions are linear.

Derivation of the linearized model

The latter equation is of greater interest since it depends on the perturbation inputs.

Since J is a function of a function we can use the chain rule to expand,

$$\frac{\partial J^r}{\partial x_j} = \sum_i \left(\frac{\partial y_i}{\partial x_j}\right)^r \frac{\partial J^r}{\partial y_i}.$$

Notice how powerful this equation is on its own. Before even using it to estimate J' we can integrate a forecast time gradient (sensitivity) back to beginning time.

TLM vs. adjoint

The TLM sums $\partial y_i/\partial x_j$ over j while this new model sums over i. Let's adopt a vector notation to compute all y_i' or $\partial J/\partial x_j$,

$$\mathbf{M}_{i,j} = \frac{\partial y_i}{\partial x_j}.$$

For the TLM we sum over the columns while for the new model we sum over the rows. In standard matrix multiplication we sum over columns. The TLM is fine,

$$\mathbf{y}' = \mathbf{M}\mathbf{x}',$$

but for the new model we have to transpose the matrix (also known as taking the adjoint!),

$$\frac{\partial J}{\partial \mathbf{x}} = \mathbf{M}^\top \frac{\partial J}{\partial \mathbf{y}}.$$

TLM vs. adjoint

The matrix $\mathbf{M}$ is known as the Resolvant of the TLM or Jacobian of the nonlinear model. The model which maps sensitivities backwards is known as the adjoint model because it uses the adjoint of the TLM.

We generally use a compact notation for the adjoint model,

$$\hat{\mathbf{x}} = \mathbf{M}^\top \hat{\mathbf{y}}.$$

Jointly the TLM and adjoint are often referred to as the linearized version of the model.

TLM vs. adjoint

Both the TLM and adjoint depend on the model gradient. They pick out the structures that evolve along the steepest gradients, either forwards or backwards.

1. The tangent linear model maps a specific perturbation onto all forecast aspects. This allows us to investigate the first question we posed.

2. The adjoint maps a specific sensitivity backwards with respect to all possible perturbations. This allows us to investigate the second question.

The adjoint is generally more useful but the TLM plays a crucial role in validating the linearized model.

Assumptions that we're making

All of this is based on the Taylor series expansion. So what assumptions have we actually made?

- The Taylor series is a good approximation when $|x'|$ is 'small'.
- The nonlinear model is continuous and is differentiable.

For most modern atmospheric models this is not the case:

- In actual applications x' can be large, e.g. the specific humidity analysis increment.
- Unresolved processes are not represented with continuous equations.

The adjoint is useful enough to try and overcome these issues!

Outline

- Introduction
- Derivation of the linearized model
- How to write and adjoint model
- Validation of the linearized model
- Practical issues
- Sensitivity examples
- Optimal perturbations

1D advection example

One dimensional advection of a tracer $q = q(x, u, t)$

$$\frac{\partial q}{\partial t} + u\frac{\partial q}{\partial x} = 0$$

The important variables are u and q. Linearize,

$$\frac{\partial q'}{\partial t} + u'\frac{\partial q^r}{\partial x} + u^r\frac{\partial q'}{\partial x} + \cancel{u'\frac{\partial q'}{\partial x}} = 0$$

Discretize

$$q'_{j,n+1} = q'_{j,n} - \Delta t\left(u'_{j,n}\frac{q^r_{j+1,n} - q^r_{j,n}}{\Delta x} + u^r_{j,n}\frac{q'_{j+1,n} - q'_{j,n}}{\Delta x}\right)$$

We don't derive the adjoint equation but rather build from code.

Steps for coding success

- Make a list of 'dependent' variables, here q and u.
- Write (re-write) your model so all calculations dependent on these variables are within a module.
- Only arguments should be dependent variables and constants.
- Do things in double precision within modules if possible.
- Minimize your use of intrinsic functions.
- Write the tangent linear model first.
- Don't write the adjoint until you've tested the TLM.

The nonlinear model

```
subroutine FirstOrderAdvection(qr,ur,dx,dt,Nx,Nt)

 implicit none

 integer, intent(in) ::  Nx, Nt
 real(8), intent(in) ::  dx, dt
 real(8), intent(inout) ::  qr(Nx), ur(Nx)

 integer ::  n,j
 real(8) ::  qnewr(Nx)

 qnewr = 0.0
 do n = 1,Nt
    do j = 1,Nx
       if (j .eq.  1) then
          qnewr(1) = qr(1) - ur(j)*dt/dx*(qr(1) - qr(Nx))
       else
          qnewr(j) = qr(j) - ur(j)*dt/dx*(qr(j) - qr(j-1))
       endif
    enddo
    qr = qnewr
 enddo

end subroutine
```

The Tangent Linear Model

```
subroutine FirstOrderAdvectionTLM(qr,ur,qp,up,dx,dt,Nx,Nt)

 ...declarations...

 do n = 1,Nt
    do j = 1,Nx
       if (j .eq.  1) then
          qnewr(1) = qr(1) - ur(j)*dt/dx*(qr(1) - qr(Nx))
          qnewp(1) = qp(1) - dt/dx*( ur(1)*(qp(1) - qp(Nx)) &
                                   + up(1)*(qr(1) - qr(Nx)) )
       else
          qnewr(j) = qr(j) - ur(j)*dt/dx*(qr(j) - qr(j-1))
          qnewp(j) = qp(j) - dt/dx*( ur(j)*(qp(j) - qp(j-1)) &
                                   + up(j)*(qr(j) - qr(j-1)) )
       endif
    enddo
    qp = qnewp
 enddo

end subroutine
```

The Tangent Linear Model

```
subroutine FirstOrderAdvectionTLM(qr,ur,qp,up,dx,dt,Nx,Nt)

 ...declarations...

 do n = 1,Nt
    do j = 1,Nx
       if (j .eq.  1) then
          qnewr(1) = qr(1) - ur(j)*dt/dx*(qr(1) - qr(Nx))
          qnewp(1) = qp(1) - dt/dx*( ur(1)*(qp(1) - qp(Nx)) &
                                  + up(1)*(qr(1) - qr(Nx)) )
       else
          qnewr(j) = qr(j) - ur(j)*dt/dx*(qr(j) - qr(j-1))
          qnewp(j) = qp(j) - dt/dx*( ur(j)*(qp(j) - qp(j-1)) &
                                  + up(j)*(qr(j) - qr(j-1)) )
       endif
    enddo
    qp = qnewp
 enddo

end subroutine
```

This isn't the best way to write TLM code...

The Tangent Linear Model

Order is often important!

```
qnewr(j) = qr(j) - ur(j)*dt/dx*(qr(j)-qr(j-1))
qnewp(j) = qp(j) - dt/dx*(ur(j)*(qp(j)-qp(j-1)) + up(j)*(qr(j)-qr(j-1)))
```

Get used to writing as ...

```
qnewp(j) = qp(j) - dt/dx*(ur(j)*(qp(j)-qp(j-1)) + up(j)*(qr(j)-qr(j-1)))
qnewr(j) = qr(j) - ur(j)*dt/dx*(qr(j)-qr(j-1))
```

Rules for coding the adjoint

- Create a copy of the TLM module.
- Strip out all non-perturbation lines.
- Reverse the order of everything (including reversing loops).
- For each line:
  - For every perturbation variable that occurs on the RHS make a new line with that variable on the LHS
  - Set it equal to itself with a plus, e.g. $qp(j) = qp(j)+$
  - Include the original LHS variable and multiply by the coefficient of the original RHS variable, e.g. $qp(j) = qp(j) + qnewp(j) * (-ur(j) * dx/dt)$
- If a variable is overwritten do not add to itself.
- Order is important!

The adjoint model

```
subroutine FirstOrderAdvectionADM(qr,ur,qp,up,dx,dt,Nx,Nt)

 ...declarations...

 do n = Nt,1,-1
    qnewp = qp
    do j = Nx,1,-1
       if (j .ne.  1) then
          qp(j) = qp(j) + (1 - dt/dx*ur(j)) * qnewp(j)
          qp(j - 1) = qp(j - 1) + dt/dx*ur(j) * qnewp(j)
          up(j) = up(j) + dt/dx*(qr(j) - qr(j-1))*qnewp(j)
          qnewp(j) = 0
       else
          qp(1) = qp(1) + (1 - dt/dx*ur(1)) * qnewp(1)
          qp(Nx) = qp(Nx) + dt/dx*ur(1) * qnewp(1)
          up(1) = up(j) + dt/dx*(qr(1) - qr(Nx))*qnewp(1)
          qnewp(1) = 0
       endif
    enddo
 enddo
 end subroutine
```

Adjoint: check-pointing vs re-calculation

Check-point the trajectory (increases memory use):

First run though nonlinear code saving what you later need:

```
do n = 1,Nt
   qtraj(n,:)  = qr
   utraj(n,:)  = ur
   do j = 1,Nx
      ...
   enddo
enddo
```

As you integrate the adjoint read back in :

```
do n = Nt,1,-1
   qr = qtraj(n,:)
   ur = utraj(n,:)
   do j = Nx,1,-1
      ...
   enddo
enddo
```

Adjoint: check-pointing vs re-calculation

Re-compute the trajectory within adjoint loop (increases computational time):

```
subroutine FirstOrderAdvectionADM(qr,ur,qp,up,dx,dt,Nx,Nt)

 ...declarations...

 do n = Nt,1,-1
    do n1 = 1,n
       do j = 1,Nx
          ...
       enddo
    enddo
    do j = Nx,1,-1
       ...
    enddo
 enddo
 end subroutine
```

For complex models you need a combination of the two techniques.

Auto differentiation

Many great tools exist that can save a lot of pain when coding TLM and adjoints, e.g.

- Tapenade (http://tapenade.inria.fr:8080/tapenade/)
- Taf (http://www.fastopt.de/)
- OpenAD

Pass them a subroutine and they return working code! Some considerations though:

- Code not usually the most efficient it can be.
- They cannot tell you if you *should* differentiate.

Outline

- Introduction
- Derivation of the linearized model
- How to write and adjoint model
- Validation of the linearized model
- Practical issues
- Sensitivity examples
- Optimal perturbations

Validation of the TLM

It is crucial to validate the linearized model carefully when devloping code. This is done using the TLM and verifying that,

$$\lim_{\mathbf{x}' \to 0} \frac{\mathbf{m}(\mathbf{x}+\mathbf{x}') - \mathbf{m}(\mathbf{x})}{\mathbf{M}\mathbf{x}'} = 1.$$

But really what we want to look at is:

$$\mathbf{m}(\mathbf{x}+\mathbf{x}') - \mathbf{m}(\mathbf{x}) \quad \text{vs.} \quad \mathbf{M}\mathbf{x}',$$

for realistic $\mathbf{x}'$. Look at the correlations between the two, root mean squared error and the root mean square of each.

Validation of the TLM

Note that

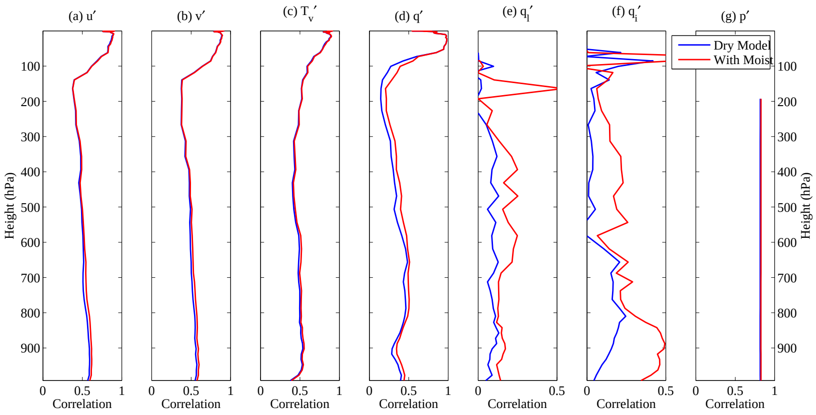
$$\lim_{\mathbf{x}' \to 0} \frac{\mathbf{m}(\mathbf{x} + \mathbf{x}') - \mathbf{m}(\mathbf{x})}{\mathbf{M}\mathbf{x}'} = 1.$$

will actually fail miserably for most modern atmospheric models that include physics!

This is because under switches, even though you make the denominator infinitesimal, the numerator will remain finite. So instead the result can tend to infinity.

Validation of the TLM

$$\operatorname{corr}\left(\mathbf{m}(\mathbf{x}+\mathbf{x}') - \mathbf{m}(\mathbf{x}), \mathbf{M}\mathbf{x}'\right).$$



Validation of the adjoint

Once you are happy with the TLM performance, the correct coding of the adjoint is confirmed using a dot product test.

$$\left\langle \mathbf{y}', \hat{\mathbf{y}} \right\rangle - \left\langle \mathbf{x}', \hat{\mathbf{x}} \right\rangle = 0$$

The result wont be zero but something on the order allowed by precision.

- $\mathcal{O}(10^{-15})$ for double precision is acceptable
- $\mathcal{O}(10^{-9})$ for single precision is acceptable

If your result is larger than this you likely have a bug somewhere.

Validation for sensitivity studies

For sensitivity studies you also have to check the linearity of J.

This is done by comparing the nonlinear J' with the linearized approximation.

$$\sum_j \frac{\partial J^r}{\partial x_j} x'_j \quad \text{vs} \quad J(y(x^r + x')) - J(y(x^r))$$

Outline

- Introduction
- Derivation of the linearized model
- How to write and adjoint model
- Validation of the linearized model
- Practical issues
- Sensitivity examples
- Optimal perturbations

Real World example

Consider the 1D advection problem again,

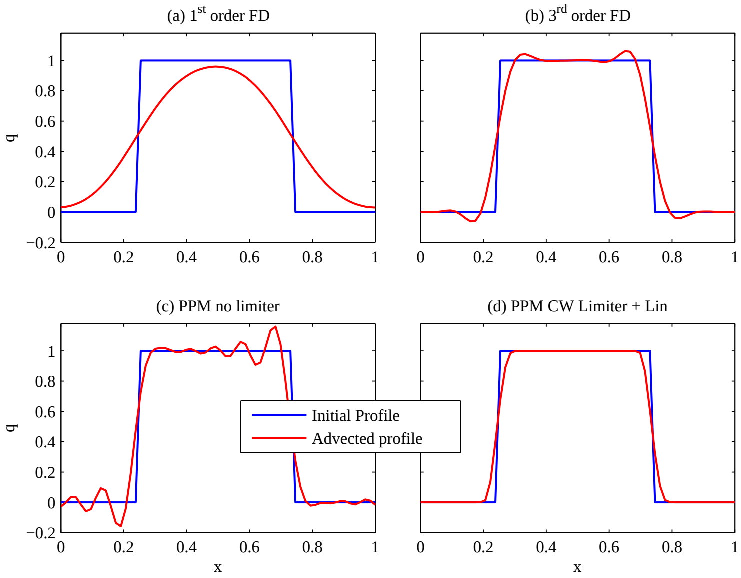
$$\frac{\partial q}{\partial t} + u\frac{\partial q}{\partial x} = 0.$$

But now with constant wind, $u' = 0$

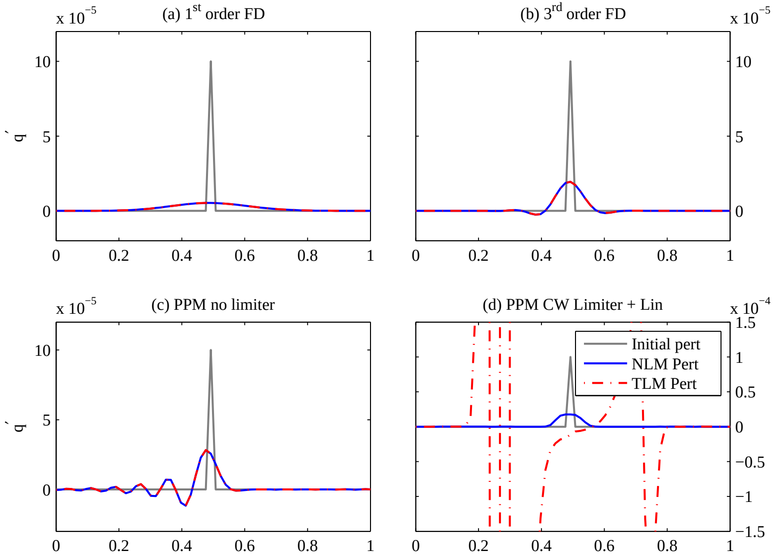
$$\frac{\partial q'}{\partial t} + u^r\frac{\partial q'}{\partial x} = 0.$$

The problem is exactly linear, so long as we choose a linear scheme then whatever perturbation we choose the linearization is exact.

Advection of the reference part



Advection of a perturbation



What causes all this growth?

When we're encountering problems one way to figure out what's going on is to compute the Jacobian. 2D example,

$$\begin{pmatrix} \dot{y_1}' \\ \dot{y_2}' \end{pmatrix} = \begin{pmatrix} M_1 & M_2 \\ M_3 & M_4 \end{pmatrix} \begin{pmatrix} x_1' \\ x_2' \end{pmatrix}$$

$$= \begin{matrix} M_1 x_1' + M_2 x_2' \\ M_3 x_1' + M_4 x_2' \end{matrix}$$

Set $x_1' = 1$ and $x_2' = 0$,

$$M_1 = \dot{y_1}'$$
$$M_3 = \dot{y_2}'$$

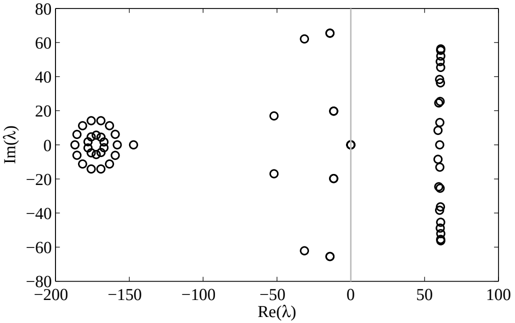
Repeat, setting each element of **x** to 1 and running the TLM to obtain corresponding column of the Jacobian.

Stability analysis

Having the Jacobian lets us examine the gradients of the model, which is very useful. It also allows for stability analysis,

$$\lambda \mathbf{q} = \mathbf{M}\mathbf{q}$$

where λ are the eigenvalues of $\mathbf{M}$.



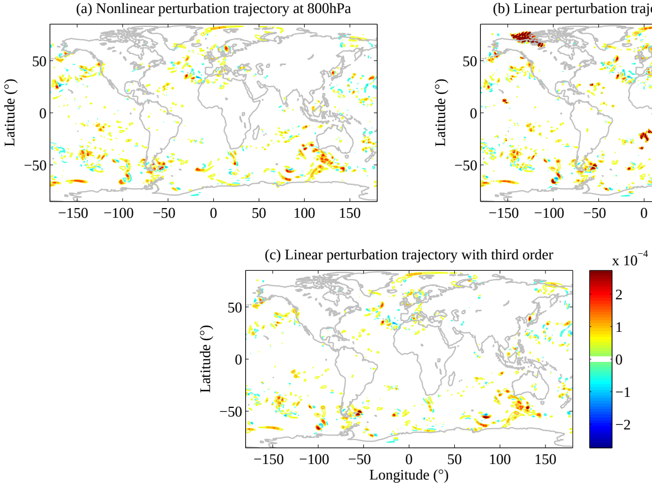
Stability analysis

When we see an eigenvalue spectrum like this it immediately tells us that this scheme should not be linearized, and has saved us the bother of writing the adjoint.

Fortunately, the reasons that linear schemes are not used for advection are not so important when it comes to transporting perturbations and sensitivities.

In this instance the nonlinear model and linearized model will be based on different equations.

Cloud liquid water perturbation in GEOS-5



(a) Nonlinear perturbation trajectory at 800hPa
(b) Linear perturbation trajectory with CWL limiter
(c) Linear perturbation trajectory with third order
Latitude (°)
Longitude (°)
50
0
−50
−150
−100
−50
0
50
100
150
x 10^−4
2
1
0
−1
−2

Problems with physics

These issues with advection aren't due to the equations but the scheme that is chosen.

For the physics the issues are often due to the equations themselves.

Below shows the equation for cloud fraction in GEOS-5,

$$C_{\mathrm{LS}} = \begin{cases} 0, & \text{if } q_s > q_T + \sigma \\ \frac{q_T + \sigma - q_s}{2\sigma}, & \text{if } q_T - \sigma < q_s \leq q_T + \sigma \\ 1, & \text{if } q_s \leq q_T - \sigma. \end{cases}$$

When linearized this produces cloud fraction perturbations $\mathcal{O}(10^3)$!

Problems with physics

Now we have to do some analysis and make a decision on how to proceed.

1. Do we try to find a new way of computing cloud fraction that is more linear and does not support rapid perturbation growth?

or,

2. Do we try to figure out, based on the trajectory, when the equation will cause a problem.

1. A new scheme (a.k.a. the ECMWF method)

Advantages:

- The new scheme can be fast and so suited to 4DVAR.
- Designed from the ground up means reliable results.

Disadvantages:

- Requires lots of expert help.
- Tuning new physics schemes is very tedious.
- The nonlinear model will have different behavior from the linearized model.

2. Scheme analysis (a.k.a. the GMAO method)

Advantages:

- Keeps the linearized model close to the nonlinear model.
- Avoids the need for too much expert help.
- Much faster to develop.
- We learn much as we go and find bugs.

Disadvantages:

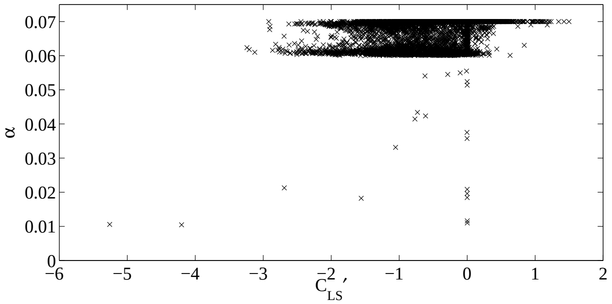
- Slower to run.
- Always the chance that you miss something.

3. Do both

Depending on application the ideal situation could be to analyze the behavior but also have an alternative scheme to use when we notice a problem.

Correlation with trajectory

Start by making plots of the problem perturbation against aspects of the trajectory.



Sometimes this can work incredibly well and give you a way to control problems. It can even speed the code up.

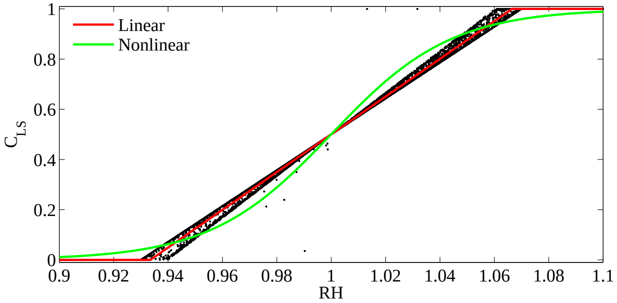
Correlation with trajectory

Other times you can exhaust all aspects of the trajectory and various combinations of trajectory values without a satisfactory solution.

But remember that there is always the Jacobian. You can isolate the part of the scheme that causes you a problem and put a Jacobian ‘wrapper’ around the code. You can try correlating elements of the Jacobian against the perturbations or even compute eigenvalues. If you turn off the code when large eigenvlaues occur you'll avoid large growth.

A new model (the simple way)

As complex as the model sometimes seems it is not always the case.



A new model (the simple way)

In this situation we can replace,

$$C_{\mathrm{LS}} = \begin{cases} 0, & \text{if } q_s > q_T + \sigma \\ \frac{q_T + \sigma - q_s}{2\sigma}, & \text{if } q_T - \sigma < q_s \leq q_T + \sigma \\ 1, & \text{if } q_s \leq q_T - \sigma. \end{cases}$$

with

$$C_{\mathrm{LS}} = \begin{cases} 0, & \text{if } \mathrm{RH} < c_1 \\ \frac{\mathrm{RH}}{c_2 - c1} - \frac{c_1}{c_2 - c1}, & \text{if } c_1 \leq \mathrm{RH} \leq c_2 \\ 1, & \text{if } \mathrm{RH} > c_2. \end{cases}$$

and get almost the same result. This function is piecewise linear and we always know the gradient. We tune one value until results are satisfactory.

Problems with physics

Most likely no perfect solution is possible so pick the lesser of the multiple evils!

- Unrealistically large perturbation growth has to be avoided.
- Piecewise linear functions have problems at the switches.
- But smoothing introduces steeper gradients.

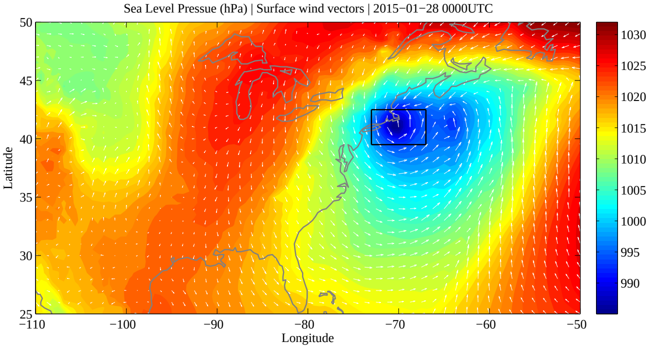
It is interesting to look at physics schemes in this way. Use of steep gradients and switches implies a certain confidence in the scheme's ability to 'determine' the cloud fraction. But can it fundamentally achieve that? And should that really be the aim?

Outline

- Introduction
- Derivation of the linearized model
- How to write and adjoint model
- Validation of the linearized model
- Practical issues
- Sensitivity examples
- Optimal perturbations

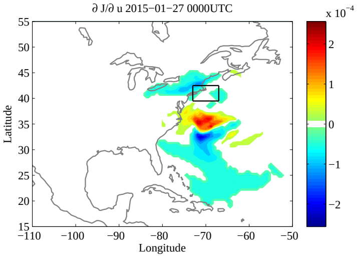
North East blizzard of January 2015

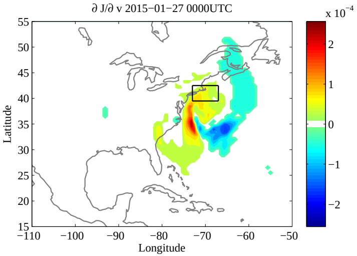
This case presented some interesting forecasting challenges.

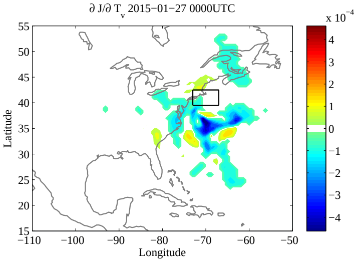


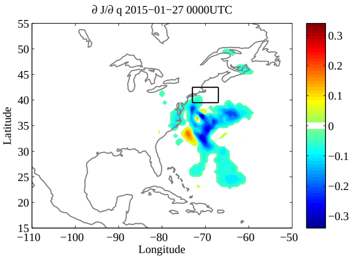
Initialize the adjoint with energy, surface pressure or circulation in the box.

North East blizzard of January 2015



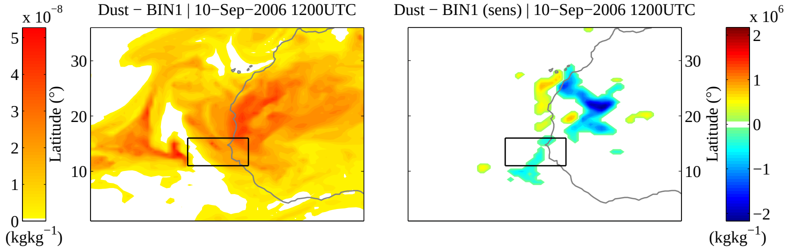






Sensitivity to dust example

Using the adjoint metric centered over the forming hurricane Helene we examine sensitivity to dust. Figure shows dust and sensitivity at 800hPa.



Dust − BIN1 | 10−Sep−2006 1200UTC
Dust − BIN1 (sens) | 10−Sep−2006 1200UTC
x 10−8
5
4
3
2
1
0
(kgkg−1)
Latitude (°)
30
20
10
x 106
2
1
0
−1
−2
(kgkg−1)

In regions of negative sensitivity a positive change to dust would reduce surface pressure in the box (increase the strength of the storm).

Outline

- Introduction
- Derivation of the linearized model
- How to write and adjoint model
- Validation of the linearized model
- Practical issues
- Sensitivity examples
- Optimal perturbations

Optimal perturbations

The general idea is that we want to maximize,

$$J' = \sum_j \frac{\partial J^r}{\partial x_j} x'_j$$

subject to a constraint, chosen as,

$$C = \frac{1}{2} \sum_j w_j x_j'^2$$

Could also minimize C subject to J'. This is Lagrange multiplier problem, which has the solution,

$$I = \sum_j \frac{\partial J^r}{\partial x_j} x'_j + \lambda \left(\frac{1}{2} \sum_j w_j x_j'^2 - C \right)$$

Optimal perturbations

Effectively we look for the location where gradient vectors of the function and the condition are parallel by differentiating I and setting equal to 0.

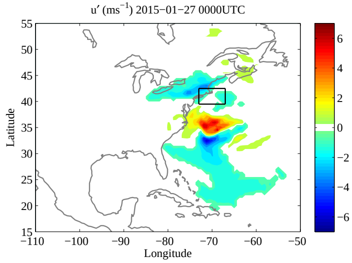
$$\frac{\partial J^r}{\partial x_j} - \lambda w_j x_j' = 0$$

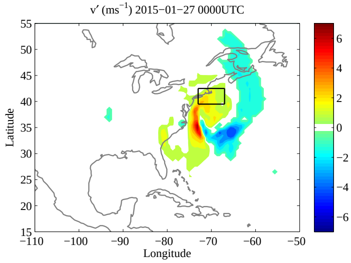
The optimal perturbation is,

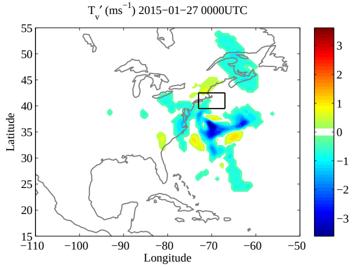
$$x_j' = \frac{\lambda}{w_j} \frac{\partial J^r}{\partial x_j}$$

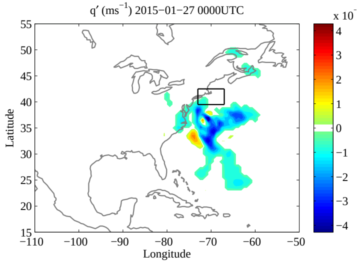
The constant λ results from the two parallel gradients generally having different magnitudes. It is calculated by plugging x_j' back into the constraint. The weights w_j are required to account for uneven grid spacing and mass changes with height.

North East blizzard optimal perturbation

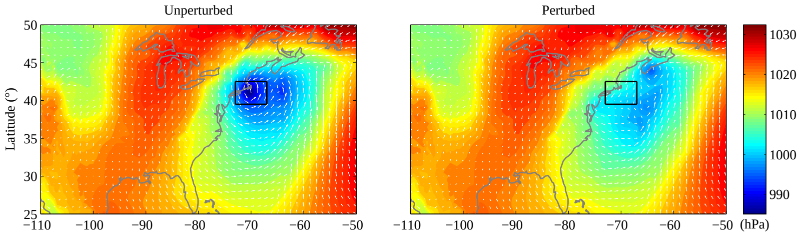


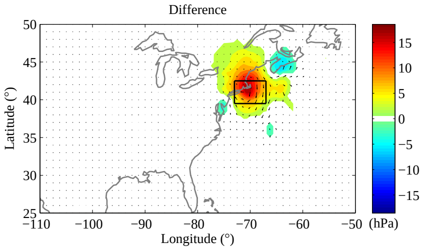






SLP difference after optimal perturbation





Conclusions

- The adjoint and TLM are very useful tools, used in a wide variety of applications (e.g. see next lecture).
- The derivation uses relatively straightforward principles.
- Development can be tedious but isn't difficult and automatic tools can help significantly.
- Care is required to understand the underlying behavior of the model but this in itself is very useful and we learn a lot.
- Looking at adjoint sensitivity can provide basic understanding of atmospheric behavior as well as informing us how to improve data assimilation.