

Tangent linear and adjoint models for variational data assimilation

Angela Benedetti

with contributions from:

**Marta Janisková , Yannick Tremolet, Philippe Lopez,
Lars Isaksen, and Gabor Radnoti**

Introduction

- 4D-Var is based on minimization of a cost function which measures the distance between the model with respect to the observations and with respect to the background state
- The **cost function and its gradient** are needed in the minimization.
- The tangent linear model provides a computationally efficient (although approximate) way to calculate the model trajectory, and from it the cost function. The adjoint model is a very efficient tool to compute the gradient of the cost function.
- **Overview:**
 - Introduction to 4D-Var
 - General definitions of Tangent Linear and Adjoint models and why they are extremely useful in variational assimilation
 - Writing TL and AD models
 - Testing them
 - Automatic differentiation software (more on this in the afternoon)

4D-Var

In 4D-Var the cost function can be expressed as follows:

$$J = \underbrace{\frac{1}{2} (\mathbf{x}_v - \mathbf{x}_{v'})^T \mathbf{B} (\mathbf{x}_v - \mathbf{x}_{v'})}_{J_b} + \underbrace{\frac{1}{2} \sum_{i=0}^1 (\mathbf{x}_v - \mathbf{y}_i)^T \mathbf{R}_i \mathbf{H}_i (\mathbf{x}_v - \mathbf{y}_i)}_{J_o}$$

B background error covariance matrix,

R observation error covariance matrix (instrumental + interpolation + observation operator error),

H observation operator (model space \rightarrow observation space),

M forward nonlinear forecast model (time evolution of the model state).

$$\min J \Leftrightarrow \nabla_{\mathbf{x}_0} J = \mathbf{B}^{-1} (\mathbf{x}_v - \mathbf{x}_{v'}) + \sum_{i=0}^1 \mathbf{M}^T \mathbf{H}_i^T \mathbf{R}_i^{-1} (\mathbf{x}_v - \mathbf{y}_i) = 0$$

H^T = adjoint of observation operator and **M**^T = adjoint of forecast model.

Incremental 4D-Var at ECMWF

- In incremental 4D-Var, the cost function is minimized in terms of **increments**:

$$\delta_{t_0} = x_t - x_{t_0}$$

with the model state defined at any time t_i as: $x_i = M^i x_0 + \dots$
 x_t is the **trajectory** around which the linearization is performed ($x_t = x_0$ at $t=0$)

- 4D-Var cost function can then be approximated to the first order by:

$$J(\delta_{t_0}) = \frac{1}{2} \delta_{t_0}^T B^{-1} \delta_{t_0} + \frac{1}{2} \sum_{i=1}^n (M^i \delta_{t_0} - d_i)^T R^{-1} (M^i \delta_{t_0} - d_i)$$

where $d_i = x_i - x_{i-1}$ is the so-called **departure**.

- The gradient of the cost function to be minimized is:

$$\min J \Leftrightarrow \nabla_{\delta_{t_0}} J = B^{-1} \delta_{t_0} + \sum_{i=1}^n M_i^T (M_i \delta_{t_0} - d_i) = 0$$

M^i and H_i^i are the tangent linear models which are used in the computations of incremental updates during the minimization (iterative procedure).

M^i and H_i^i are the adjoint models which are used to obtain the gradient of the cost function with respect to the initial condition.

Details on linearisation

In the first order approximation, a perturbation δx of the control variable (initial condition) evolves according to the **tangent linear model**:

$$\delta x_i = \begin{bmatrix} \dots \\ \dots \\ \dots \end{bmatrix}$$

where i is the time-step.

The perturbation of the cost function around the initial state is:

$$\begin{aligned} \delta J &= \frac{1}{2} \delta x^T H_0 \delta x + \sum_{i=1}^N \delta x_i^T H_i \delta x_i + \sum_{i=1}^N d_i^T \delta x_i \\ &= \frac{1}{2} \delta x^T H_0 \delta x + \sum_{i=1}^N \delta x_i^T H_i \delta x_i + \sum_{i=1}^N d_i^T \delta x_i \end{aligned}$$

where H_i' is the linearised version of H_i about x_i and $d_i = y_i - \hat{y}_i$ are the departures from observations.

Details of the linearisation (cnt.)

The gradient of the cost function with respect to $\delta_{\mathbf{u}}$ is given by:

$$\begin{aligned} \nabla_{\delta_{\mathbf{u}}} J &= \nabla_{\mathbf{u}} J + \sum_{i=1}^n \lambda_i \mathbf{A}_i^T \mathbf{e}_i \\ &= \nabla_{\mathbf{u}} J + \sum_{i=1}^n \lambda_i \mathbf{A}_i^T \mathbf{e}_i \\ &= \nabla_{\mathbf{u}} J + \sum_{i=1}^n \lambda_i \mathbf{A}_i^T \mathbf{e}_i \end{aligned}$$

remembering that $(AB)^T = B^T A^T$

The optimal initial perturbation is obtained by finding the value of $\delta_{\mathbf{u}}$ for which:

$$\nabla_{\delta_{\mathbf{u}}} J = \mathbf{0}$$

The gradient of the cost function with respect to the initial condition is provided by the **adjoint solution** at time $t=0$.

Definition of adjoint operator

For any linear operator M' there exist an *adjoint* operator M^* such as:

$$\langle x, M' y \rangle = \langle M^* x, y \rangle$$

where $\langle \cdot, \cdot \rangle$ is an inner scalar product and x, y are vectors (or functions) of the space where this product is defined.

It can be shown that for the inner product defined in the Euclidean space :

$$M^* = (M')^T$$

We will now show that the gradient of the cost function at time $t=0$ is provided by the solution of the adjoint equations at the same time:

$$\nabla_{u_0} J = - \lambda_0$$

Adjoint solution

Usually the initial guess x_0 is chosen to be equal to the background x_b so that the initial perturbation $\delta_{x_0} = 0$.
The gradient of the cost function is hence simplified as:

$$\nabla_{x_0} J = - \sum_{i=1}^n H_i^T R^{-1} a_i$$

We choose the solution of the adjoint system as follows:

$$x_{n+1}^* = 0$$

$$x_0^* = M_1^* x_1^*$$

$$x_i^* = M_{i+1}^* x_{i+1}^* + H_i^* R^{-1} a_i, i = 1, \dots, n$$

We then substitute progressively the solution x_i^* into the expression for x_0^*

Adjoint solution (cnt.)

$$\begin{aligned}
 x_0^* &= \dots + \dots = \\
 &= \dots + \dots + \dots = \\
 &= \dots
 \end{aligned}$$

Finally, regrouping and remembering that $x_{n+1}^* = 0$ and that $M^* = I$ and $H^* = 0$ we obtain the following equality:

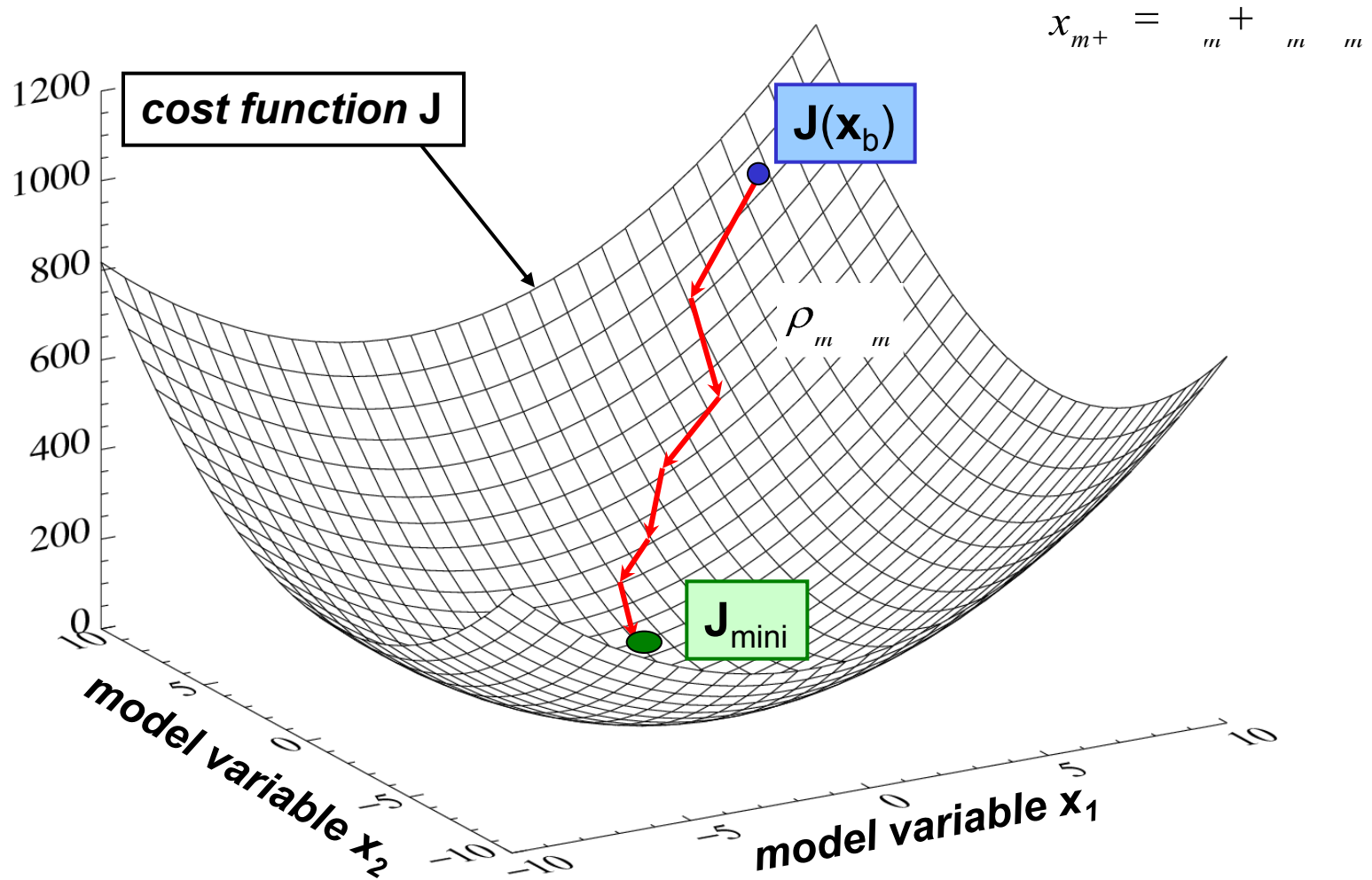
$$x_0^* = \sum_{i=0}^n \lambda_i^T R^{-1} a_i \Rightarrow \lambda_i = -\nabla_{u_i} J$$

The **gradient** of **the cost function** with respect to the control variable (initial condition) is obtained by a **backward** integration of the **adjoint model**.

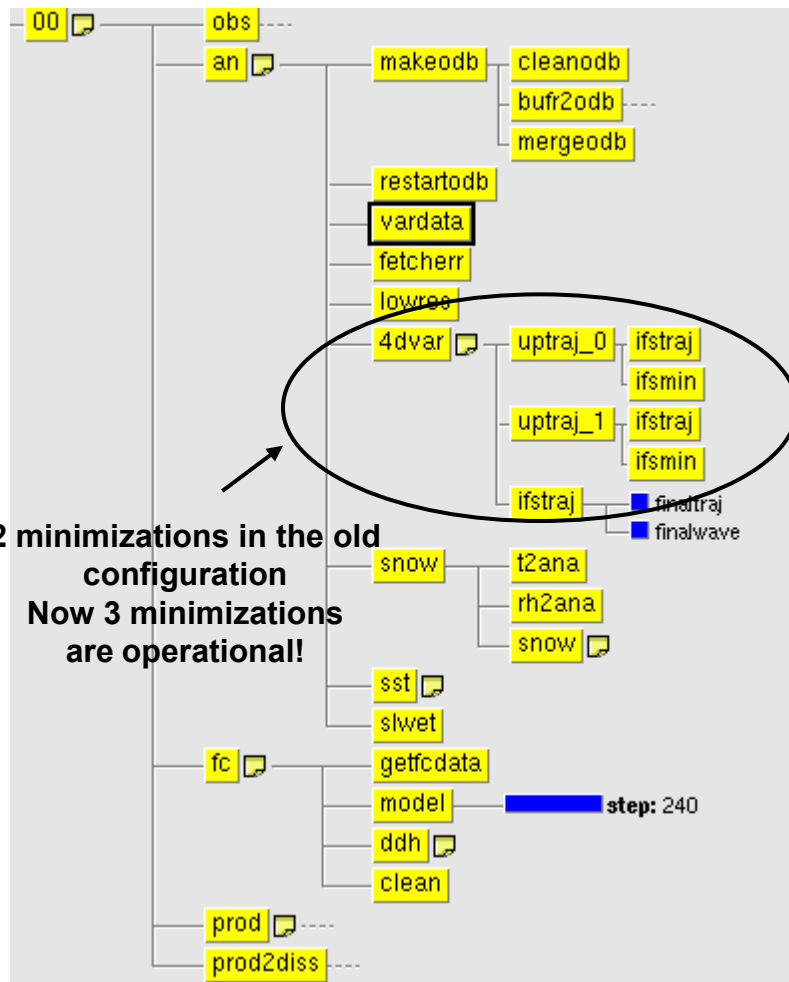
Iterative steps in the 4D-Var Algorithm

1. Integrate forward model gives J .
2. Integrate adjoint model backwards gives ∇J .
3. If $\|\nabla J(x_m)\| \leq \varepsilon$ then stop.
4. Compute descent direction D_m (Newton, CG, ...).
5. Compute step size ρ_m : $J(x_m + \rho_m D_m) = \min_{\rho} J(x_m + \rho D_m)$.
6. Update initial condition: $x_{m+1} = x_m + \rho_m D_m$.

Finding the minimum of cost function $J \rightarrow$ iterative minimization procedure



An analysis cycle in 4D-Var



1st ifstraj:

- Non-linear model is used to compute the high-res trajectory (T1279 operational, 12-h forecast)
- High-res **departures** are computed at exact obs time and location
- Trajectory is interpolated at low res (T159)

1st ifsmin (70 iterations):

- Iterative minimization at T159 resolution
- Tangent linear with simplified physics to calculate the increments δx_i
- The Adjoint is used to compute the gradient of the cost function with respect to the departure in initial condition $\nabla_{\delta x_0}$
- Analysis increment at initial time δx_0 is interpolated back linearly from low-res to high-res and it provides a new initial state for the 2nd trajectory run

2nd ifstraj:

- repeat 1st ifstraj and interpolates at T255 resolution

2nd ifsmin (30 iterations):

- repeat 1st ifsmin at T255

Last ifstraj:

- Uses updated initial condition to run another 12-h forecast and stores analysis departures in the Observational Data Base (ODB)

Brief summary on TL and AD models

- **TANGENT LINEAR MODEL**

If M is a model such as:

$$\mathbf{x}(t_{i+1}) = M[\mathbf{x}(t_i)]$$

then the tangent linear model of M , called M' , is:

$$\delta\mathbf{x}(t_{i+1}) = M'[\mathbf{x}(t_i)]\delta\mathbf{x}(t_i) = \frac{\partial M[\mathbf{x}(t_i)]}{\partial \mathbf{x}}\delta\mathbf{x}(t_i)$$

- **ADJOINT MODEL**

The adjoint of a linear operator M' is the linear operator M^* such that, for the inner product \langle, \rangle ,

$$\forall \mathbf{x}, \forall \mathbf{y} \quad \langle M'\mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{x}, M^*\mathbf{y} \rangle$$

Remarks:

- with the euclidian inner product, $M^* = M'^T$.
- in variational assimilation, $\nabla_x \mathcal{J} = M^* \nabla_y \mathcal{J}$, where \mathcal{J} is the cost function.

Simple example of adjoint writing

- non-linear statement

$$x = y + z^2$$

$$z = z$$

$$y = y$$

$$x = y + z^2$$

- tangent linear statement

$$\delta z = \delta z$$

$$\delta y = \delta y$$

$$\delta x = \delta y + 2z\delta z$$

or in a matrix form:

$$\begin{pmatrix} \delta z \\ \delta y \\ \delta x \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 2z & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \delta z \\ \delta y \\ \delta x \end{pmatrix}$$

Simple example of adjoint writing (cnt.)

- **adjoint statement**

- transpose matrix

$$\begin{pmatrix} \delta z^* \\ \delta y^* \\ \delta x^* \end{pmatrix} = \begin{pmatrix} 1 & 0 & 2z \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} \delta z^* \\ \delta y^* \\ \delta x^* \end{pmatrix}$$

or in the form of equation set:

$$\delta z^* = \delta z^* + 2z\delta x^*$$

$$\delta y^* = \delta y^* + \delta x^*$$

$$\delta x^* = 0$$

(often the adjoint variables are indicated in literature with an asterisk)

As an alternative to the matrix method, adjoint coding can be carried out using a **line-by-line** approach.

More practical examples on adjoint coding: the Lorenz model

$$\frac{dx_1}{dt} = -px_1 + px_2 \quad (1)$$

$$\frac{dx_2}{dt} = (r - x_3)x_1 - x_2 \quad (2)$$

$$\frac{dx_3}{dt} = x_1x_2 - bx_3 \quad (3)$$

where t is the time, p the Prandtl number, r the Rayleigh number, b the aspect ratio, x_1 the intensity of convection, x_2 the maximum temperature difference and x_3 the stratification change due to convection (see references).

The linear code in Fortran

Linearize each line of the code one by one:

$$\begin{aligned} dxdt(1) &= -p * x(1) + p * x(2) && : \text{Nonlinear statement} \\ (1) dxdt_tl(1) &= -p * x_tl(1) + p * x_tl(2) && : \text{Tangent linear} \end{aligned}$$

$$\begin{aligned} dxdt(2) &= x(1) * (r - x(3)) - x(2) \\ (2) dxdt_tl(2) &= x_tl(1) * (r - x(3)) \\ &\quad - x(1) * x_tl(3) - x_tl(2) \end{aligned}$$

...etc

If we drop the `_tl` subscripts and replace the trajectory `x`, with `x5` (as it is per convention in the ECMWF code), the tangent linear equations become:

$$\begin{aligned} (1) dxdt(1) &= -p * x(1) + p * x(2) \\ (2) dxdt(2) &= x(1) * (r - x5(3)) - x5(1) * x(3) - x(2) \end{aligned}$$

...

Similarly, the adjoint variables in the IFS are indicated without any subscripts (it saves time when writing tangent linear and adjoint codes).

Trajectory

The trajectory has to be available. It can be:

- saved which costs memory,
- recomputed which costs CPU time.

Intermediate options exist using check-pointing methods.

Adjoint of one instruction

From the tangent linear code: $\mathbf{dxdt}(1) = -p \cdot \mathbf{x}(1) + p \cdot \mathbf{x}(2)$

In matrix form, it can be written as:

$$\begin{pmatrix} x_1 \\ x_2 \\ dx_1 dt \end{pmatrix} == \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -p & p & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ dx_1 dt \end{pmatrix}$$

which can easily be transposed:

$$\begin{pmatrix} x_1^* \\ x_2^* \\ dx_1 dt^* \end{pmatrix} == \begin{pmatrix} 1 & 0 & -p \\ 0 & 1 & p \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1^* \\ x_2^* \\ dx_1 dt^* \end{pmatrix}$$

The corresponding code is:

$$\mathbf{x}(1) = \mathbf{x}(1) - p \cdot \mathbf{dxdt}(1)$$

$$\mathbf{x}(2) = \mathbf{x}(2) + p \cdot \mathbf{dxdt}(1)$$

$$\mathbf{dxdt}(1) = 0$$

The Adjoint Code

Property of adjoints (transposition):

$$(L_n L_{n-1} \dots L_2 L_1)^* = L_1^* L_2^* \dots L_{n-1}^* L_n^*$$

Application: $M_i = L_n L_{n-1} \dots L_2 L_1$ where L_i represents the line i of the tangent linear model.

The **adjoint code** is made of the **transpose** of each line of the tangent linear code in **reverse order**.

Adjoint of loops

In the TL code for the Lorenz model we have:

```
DO i=1,3
  x(i)=x(i)+dt*dxdt(i)
ENDDO
```

which is equivalent to

```
x(1)=x(1)+dt*dxdt(1)
x(2)=x(2)+dt*dxdt(2)
x(3)=x(3)+dt*dxdt(3)
```

We can transpose and reverse the lines:

```
dxdt(3)=dxdt(3)+dt*x(3)
dxdt(2)=dxdt(2)+dt*x(2)
dxdt(1)=dxdt(1)+dt*x(1)
```

which is equivalent to

```
DO i=3,1,-1
  dxdt(i)=dxdt(i)+dt*x(i)
ENDDO
```

Conditional statements

- What we want is the adjoint of the statements which were actually executed in the direct model.
- We need to know which branch was executed
- The result of the conditional statement has to be stored:
it is part of the trajectory !!!

Summary of basic rules for line-by-line adjoint coding (1)

Adjoint statements are derived from tangent linear ones in a reversed order

Tangent linear code	Adjoint code
$\delta x = 0$	$\delta x^* = 0$
$\delta x = A \delta y + B \delta z$	$\delta y^* = \delta y^* + A \delta x^*$ $\delta z^* = \delta z^* + B \delta x^*$ $\delta x^* = 0$
$\delta x = A \delta x + B \delta z$	$\delta z^* = \delta z^* + B \delta x^*$ $\delta x^* = A \delta x^*$
do k = 1, N $\delta x(k) = A \delta x(k-1) + B \delta y(k)$ end do	do k = N, 1, -1 (Reverse the loop!) $\delta x^*(k-1) = \delta x^*(k-1) + A \delta x^*(k)$ $\delta y^*(k) = \delta y^*(k) + B \delta x^*(k)$ $\delta x^*(k) = 0$ end do
if (condition) tangent linear code	if (condition) adjoint code

Order of operations is important when variable is updated!

And do not forget to initialize local adjoint variables to zero !

Summary of basic rules for line-by-line adjoint coding (2)

To save memory, the trajectory can be recomputed just before the adjoint calculations.

Tangent linear code	Trajectory and adjoint code
<pre>if (x > x0) then $\delta x = A \delta x / x$ $x = A \text{Log}(x)$ end if</pre>	<pre>----- Trajectory ----- $x_{\text{store}} = x$ (storage for use in adjoint) if (x > x0) then $x = A \text{Log}(x)$ end if ----- Adjoint ----- if ($x_{\text{store}} > x0$) then $\delta x^* = A \delta x^* / x_{\text{store}}$ end if</pre>

The most common sources of error in adjoint coding are:

- 1) Pure coding errors
- 2) Forgotten initialization of local adjoint variables to zero
- 3) Mismatching trajectories in tangent linear and adjoint (even slightly)
- 4) Bad identification of trajectory updates

More remarks about adjoints

- The adjoint always exists and it is unique, assuming spaces of finite dimension. Hence, coding the adjoint does not raise questions about its existence, only questions of technical implementation.
- In the meteorological literature, the term *adjoint* is often improperly used to denote the adjoint of the tangent linear of a non-linear operator. In reality, the adjoint can be defined for any linear operator. One must be aware that discussions about the *existence of the adjoint* usually address the existence of the tangent linear model.
- Without re-computation, the cost of the TL is usually about 1.5 times that of the non-linear code, the cost of the adjoint between 2 and 3 times.
- The tangent linear model is not strictly necessary to run 4D-Var (but it is in the incremental 4D-Var formulation in use operationally at ECMWF). It is also needed as an intermediate step to write the adjoint.

Test for tangent linear model

- Taylor formula:

$$\lim_{\lambda \rightarrow 0} \frac{M(\mathbf{x} + \lambda \delta \mathbf{x}) - M(\mathbf{x})}{M'(\lambda \delta \mathbf{x})} = 1$$

Perturbation scaling factor

λ	RATIO
0.1E-09	0.9994875881543574E+00
0.1E-08	0.9999477148855701E+00
0.1E-07	0.9999949234236705E+00
0.1E-06	0.9999993501022509E+00
0.1E-05	0.9999999496119013E+00
0.1E-04	0.9999996111338369E+00
0.1E-03	0.9999993179193711E+00
0.1E-02	0.99999724488345042E+00
0.1E-01	0.9998727842790062E+00
0.1E+00	0.9978007454264978E+00
0.1E+01	0.9683066504549524E+00

} machine precision reached

Test for adjoint model

- adjoint identity:

$$\forall \mathbf{x}, \forall \mathbf{y} \quad \langle M' \cdot \mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{x}, M^* \cdot \mathbf{y} \rangle$$

$$\langle F(\mathbf{X}), \mathbf{Y} \rangle = -.13765102625251640000\text{E-}01$$

$$\langle \mathbf{X}, F^*(\mathbf{Y}) \rangle = -.13765102625251680000\text{E-}01$$

$$\text{ratio of norms} = 1.000000000000000005$$

THE DIFFERENCE IS 11.351 TIMES THE ZERO OF THE MACHINE

The adjoint test is truly unforgiving. If you do not have a ratio of the norm close to 1 within the precision of the machine, you know there is a bug in your adjoint. At the end of your debugging you will have a **perfect** adjoint (although you may still have an imperfect tangent linear!)

Test of adjoint in practice...

(example from the aerosol assimilation)

- Compute perturbed variable (for example optical depth, τ) using perturbation in input variables (for example, mixing ratio, r , and humidity, q) with the tangent linear code

$$d\tau = \frac{\partial \tau}{\partial r} dr + \frac{\partial \tau}{\partial q} dq$$

$$NORM_TL = \sqrt{\tau^2}$$

- Call adjoint routine to obtain gradients in q and r with respect to initial condition (q_0 and r_0) from perturbation in τ .

$$dq = \frac{\partial q}{\partial \tau} d\tau + \left(\frac{\partial q}{\partial r_0} \right) dr_0$$

$$dr = \frac{\partial r}{\partial \tau} d\tau + \left(\frac{\partial r}{\partial r_0} \right) dr_0$$

- Compute the norm from the adjoint calculation, using unperturbed state and gradients:

$$NORM_AD = \sqrt{q^2 + r^2}$$

- According to the test of adjoint **NORM_TL must be equal to NORM_AD** to the machine precision!

Automatic differentiation

- Because of the strict rules of tangent linear and adjoint coding, automatic differentiation is possible.
- Existing tools: TAF (TAMC), TAPENADE (Odyssee), ...
 - Reverse the order of instructions,
 - Transpose instructions instantly without typos !!!
 - Especially good in deriving tangent linear codes!
- There are still unresolved issues:
 - It is **NOT** a black box tool,
 - Cannot handle non-differentiable instructions (TL is wrong),
 - Can create huge arrays to store the trajectory,
 - The codes often need to be cleaned-up and optimised.

Useful References

- **Variational data assimilation:**

Lorenc, A., 1986, *Quarterly Journal of the Royal Meteorological Society*, **112**, 1177-1194.
Courtier, P. *et al.*, 1994, *Quarterly Journal of the Royal Meteorological Society*, **120**, 1367-1387.
Rabier, F. *et al.*, 2000, *Quarterly Journal of the Royal Meteorological Society*, **126**, 1143-1170.

- **The adjoint technique:**

Errico, R.M., 1997, *Bulletin of the American Meteorological Society*, **78**, 2577-2591.

- **Tangent-linear approximation:**

Errico, R.M. *et al.*, 1993, *Tellus*, **45A**, 462-477.
Errico, R.M., and K. Reader, 1999, *Quarterly Journal of the Royal Meteorological Society*, **125**, 169-195.
Janisková, M. *et al.*, 1999, *Monthly Weather Review*, **127**, 26-45.
Mahfouf, J.-F., 1999, *Tellus*, **51A**, 147-166.

- **Lorenz model:**

X. Y. Huang and X. Yang. Variational data assimilation with the Lorenz model. Technical Report 26, HIRLAM, April 1996.
E. Lorenz. Deterministic nonperiodic flow. *J. Atmos. Sci.*, 20:130-141, 1963.

- **Automatic differentiation:**

Giering R., *Tangent Linear and Adjoint Model Compiler, Users Manual* Center for Global Change Sciences, Department of Earth, Atmospheric, and Planetary Science, MIT, 1997
Giering R. and T. Kaminski, *Recipes for Adjoint Code Construction*, *ACM Transactions on Mathematical Software*, 1998
TAMC: <http://www.autodiff.org/>
TAPENADE: <http://www-sop.inria.fr/tropics/tapenade.html>

- **Sensitivity studies using the adjoint technique**

Janiskova, M. and J.-J. Morcrette., 2005. Investigation of the sensitivity of the ECMWF radiation scheme to input parameters using adjoint technique. *Quart. J. Roy. Meteor. Soc.*, **131**, 1975-1996.