Brief tutorial on tangent linear and adjoint models

Angela Benedetti (ECMWF)

with contributions from:

Marta Janisková, Philippe Lopez, Lars Isaksen, Gabor Radnoti and Yannick Tremolet (ECMWF)

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:
  - 4D-VAR cost function
  - Definition of TL and AD operators
  - Writing TL and AD examples and testing them
  - Brief mention of automatic differentiation software
  - Practical exercises

4D-Var

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

$$J = \underbrace{\frac{1}{2}(\mathbf{x}_0 - \mathbf{x}_b)^T \mathbf{B}^{-1}(\mathbf{x}_0 - \mathbf{x}_b)}_{J_b} + \underbrace{\frac{1}{2}\sum_{i=0}^{n}\left(H_i M_i\left[\mathbf{x}_0\right] - \mathbf{y}_i\right)^T \mathbf{R}_i^{-1}\left(H_i M_i\left[\mathbf{x}_0\right] - \mathbf{y}_i\right)}_{J_o}$$

B background error covariance matrix,
R observation error covariance matrix (instrumental + interpolation + observation operator error),
y_i observations
x_o initial model state
x_b background state
M forward nonlinear forecast model (time evolution of the model state, index i),
H observation operator (model space → observation space).

$$\min J \Leftrightarrow \nabla_{\mathrm{x}_0} J = \mathbf{B}^{-1}(\mathbf{x}_0 - \mathbf{x}_b) + \sum_{i=0}^{n} \mathbf{M}'^T[t_i, t_0]\mathbf{H}_i'^T \mathbf{R}_i^{-1}\left(H_i M_i\left[\mathbf{x}_0\right] - \mathbf{y}_i\right) = 0$$

H'^{T} = adjoint of observation operator and **M'**^{T} = adjoint of forecast model.

Definition of adjoint operator

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

$$\left\langle x, M^{'} y \right\rangle = \left\langle M^{*} x, y \right\rangle$$

where $\langle , \rangle$ 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 :

$$M^{*} = M^{'T}$$

It can be shown that the gradient of the cost function at time t=0 is provided by the solution of the adjoint equations at the same time:

$$\nabla_{\delta x_0} J = -x_0^{*}$$

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:

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

- **ADJOINT MODEL**

  The adjoint of a linear operator M' is the linear operator $\mathbf{M}^*$ such that, for the inner product $<,>$,

$$\boxed{\forall\mathbf{x},\forall\mathbf{y} \qquad < M'\mathbf{x},\mathbf{y} > = < \mathbf{x},\mathbf{M}^*\mathbf{y} >}$$

Remarks:

– with the euclidian inner product, $\mathbf{M}^* = M'^T$.

– in variational assimilation, $\nabla_x \mathcal{J} = \mathbf{M}^*\nabla_y\mathcal{J}$, where $\mathcal{J}$ is the cost function.

Simple example of adjoint writing

- **non-linear statement**

$$x = y + z^2$$

$$\begin{aligned} z &= z \\ y &= y \\ x &= y + z^2 \end{aligned}$$

- **tangent linear statement**

$$\begin{aligned} \delta z &= \delta z \\ \delta y &= \delta y \\ \delta x &= \delta y + 2z\delta z \end{aligned}$$

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.)

Often the adjoint variables in mathematical formulations are indicated with an asterisk

- **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:

$$\begin{aligned} \delta z^* &= \delta z^* + 2z\delta x^* \\ \delta y^* &= \delta y^* + \delta x^* \\ \delta x^* &= 0 \end{aligned}$$

Do not forget the last equation!!! That too is part of the adjoint!

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

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
Order of operations is important when variable is updated!
	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*

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 (again it depends on the complexity of the model).

Tangent linear code	Trajectory and adjoint code

if ($x > x0$) then
   $\delta x = A\ \delta x\ /\ x$
   $x = A\ \mathrm{Log}(x)$
end if
	------------- Trajectory ----------------
	$x_{store} = x$ (storage for use in adjoint)
	if ($x > x0$) then
	   $x = A\ \mathrm{Log}(x)$
	end if
	--------------- Adjoint ------------------
	if ($x_{store} > x0$) then
	   $\delta x^* = A\ \delta x^*\ /\ x_{store}$
	end if

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

Test for tangent linear model

- **Taylor formula:**

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

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	0.9999953179193711E+00
0.1E-02	0.9999724488345042E+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:**

$$\boxed{\forall \mathbf{x}, \forall \mathbf{y} \quad < M'.\mathbf{x}, \mathbf{y} > = < \mathbf{x}, \mathbf{M}^*.\mathbf{y} >}$$

$$< \mathrm{F(X)} \ , \ \mathrm{Y} > \ = \text{-.13765102625251640000E-01}$$
$$< \mathrm{X} \ , \ \mathrm{F^*(Y)} > \ = \text{-.13765102625251680000E-01}$$

ratio of norms = 1.00000000000000005

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. If properly tested, the adjoint is the only piece of code on Earth to be entirely bug-free (although you may still have an imperfect tangent linear)!

Automatic differentiation

- Because of the strict rules of tangent linear and adjoint coding, automatic differentiation is possible.

- Existing tools: TAF (TAMC), TAPENADE (Odyssée), ...
  - 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. Available on ftp site (see notes for practical session).
  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 PlanetaryScience,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.

Practical exercises

1. Find the adjoint of this linear statement

$$\frac{dy}{dt} = -\lambda y$$

2. Find the adjoint of this nonlinear statement

$$x = Ax + y + z^2$$

1. Solution

Set: $z = dy/dt$

Non-linear statement:

$$y = y$$

$$z = -\lambda y$$

Tangent linear statement:

$$\delta y = \delta y$$

$$\delta z = -\lambda \delta y$$

Tangent linear statement in matrix form:

$$\begin{pmatrix} \delta y \\ \delta z \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -\lambda & 0 \end{pmatrix} \begin{pmatrix} \delta y \\ \delta z \end{pmatrix}$$

Adjoint statement in matrix form (matrix transposition):

$$\begin{pmatrix} \delta y^* \\ \delta z^* \end{pmatrix} = \begin{pmatrix} 1 & -\lambda \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \delta y^* \\ \delta z^* \end{pmatrix}$$

Adjoint statement:

$$\delta y^* = \delta y^* - \lambda \delta z^*$$
$$\delta z^* = 0$$

2. Solution

Non-linear statement:

$$\begin{aligned} z &= z \\ y &= y \\ x &= Ax + y + z^2 \end{aligned}$$

Tangent linear statement:

$$\begin{aligned} \delta z &= \delta z \\ \delta y &= \delta y \\ \delta x &= A\delta x + \delta y + 2z\delta z \end{aligned}$$

Tangent linear statement in matrix form:

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

Adjoint statement in matrix form (note the matrix transposition):

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

Adjoint statement:

$$\begin{aligned} \delta z^* &= \delta z^* + 2z\delta x^* \\ \delta y^* &= \delta y^* + \delta x^* \\ \delta x^* &= A\delta x^* \end{aligned}$$