Hands-on derivation of tangent linear and adjoint codes (supplementary material)

Angela Benedetti and Marta Janisková

Road map

- Simple exercise of adjoint derivation & test of adjoint

- "Manual" derivation of tangent linear for the Lorenz three-variable model (note that the first two equations were derived this morning during the lecture and can be found in the handouts).

- Derivation of the adjoint code (if time is an issue we will only start this task)

- Derivation of tangent linear and adjoint codes using an automatic differentiation software online (TAPENADE).
http://www-sop.inria.fr/tropics/ for general info
http://tapenade.inria.fr:8080/tapenade/index.jsp for the actual deal

- Questions

Details on the Lorenz code and its use in variational data assimilation can be found in Huang and Yang (1996)

Find the adjoint of this nonlinear statement using the matrix method:

$$x = Ay + Bz^2$$

Note: A and B are constants

Naming conventions

•In the test of the exercises all tangent linear variables will have a suffix “d” and all adjoint variables will have the suffix “b”, in order to be able to compare more directly with codes derived using the automatic differentiation code TAPENADE. The variables without any suffix are the nonlinear model variables (the trajectory).

Test of adjoint - example

Starting from the example of the non-linear statement: $x = Ay + Bz^2$

Input variable: y,z Output variable: x Constants: A, B

1. Perform the test of adjoint using the code simple_example.f90 which you can find at the following ftp:

ftp ftp.ecmwf.int
Login: benedetti
Password: ang31a
cd Training_Course/Simple_Example
mget *

2. Look the structure of the source code (simple_example.f90)

3. Compile the code using the script compile_simple_example and create an executable (you will need to change the permissions of the compilation script using the command chmod 777)

4. Run the executable – what can you see from the output?

5. Repeat the test by changing the value of the adjoint initialization…what happens?

Lorenz model code (FORTRAN)

```
SUBROUTINE model(x,dt,nstep)
REAL,INTENT(INOUT) :: x(3)
REAL,INTENT(IN)    :: dt     ! Constant
REAL               :: y(3)
INTEGER, INTENT(IN):: nstep
  DO i = 1,nstep
    CALL lorenz(x,y)
    CALL step  (x,y,dt)
  ENDDO
! ----------------------------
SUBROUTINE lorenz(x,y)
REAL,INTENT(IN) :: x(3)
REAL,INTENT(OUT):: y(3)
REAL :: p, r, b            ! constants
  y(1) = -p*x(1)+p*x(2)
  y(2) = x(1)*(r-x(3))-x(2)
  y(3) = x(1)*x(2)-b*x(3)
END SUBROUTINE lorenz
! -------------------------
SUBROUTINE step(x,y,dt)
REAL,INTENT(INOUT):: x(3)
REAL,INTENT(IN)   :: y(3),dt
  DO i = 1,3
    x(i) = x(i)+dt*y(i)
  ENDDO
END SUBROUTINE step
```

Remember:
in this exercise the timestep dt is simply a constant, exactly like p, r and b, as far as the adjoint derivation is concerned.
You are deriving the tangent linear and adjoint of y with respect to x! Forget the fact that y is a time derivative.

Steps to use the automatic differentiation software

1. Work in groups of 2-3 people
2. Make sure you have access to the internet from your desktops
3. Open a terminal window
4. Go to ftp ftp.ecmwf.int
5. Login: benedetti
6. Password: ang31a
7. cd Training_Course/Lorenz
8. mget * (you will be acquiring three files: model.f95 lorenz.f95 and step.f95)
9. Open an internet browser and go to:
   http://tapenade.inria.fr:8080/tapenade/index.jsp

10. Upload files model.f95, lorenz.f95, step.f95 as “source”
11. Enter “model” as top routine, “y x” as dependent variable and “x” as independent variable.
12. Differentiate in: “Tangent mode” for tangent linear and “Adjoint” for adjoint code
13. You can look at the resulting codes directly on the screen by clicking on the specific routine or download them onto your desktop.
14. There will be slight differences with respect to your manually-derived codes: ask if you are confused.
15. Have fun!