Changes in CMAQ-ADJ

Overview

New features of CMAQ-ADJ include new module to control execution order of x, y advection; corrected time argument for diffusion and advection science process, discrete adjoint of advection module and sensitivity to boundary conditions.

Execution order of x, y advection. In CMAQ, x advection and y horizontal advection are performed alternatively. If x advection is performed first and then followed by y advection in current advection time step, the next advection time step will have y advection performed first and then x advection. In the adjoint run the advection adjoint sequence is in the reverse order of that in the forward run. Figure 1 illustrates the idea of x, y advection execution order in the forward run and the corresponding order of adjoint of x, y advection in the adjoint run.

In current implementation, we use a global variable XFIRST to decide the execution order of x and y advection current advection step. If XFIRST is true, then x advection will be executed first in the current time step and XFIRST needs to be set to false so that y advection will be started first in next time step. In addition, the value of XFIRST after the last step of the forward run is used to determine the adjoint of x, y advection execution order at the beginning step of the adjoint run. Figure 2 also shows the use of XFIRST to control the execution order of x and y advection in forward run and the order of the adjoint of x and y advection in adjoint run.

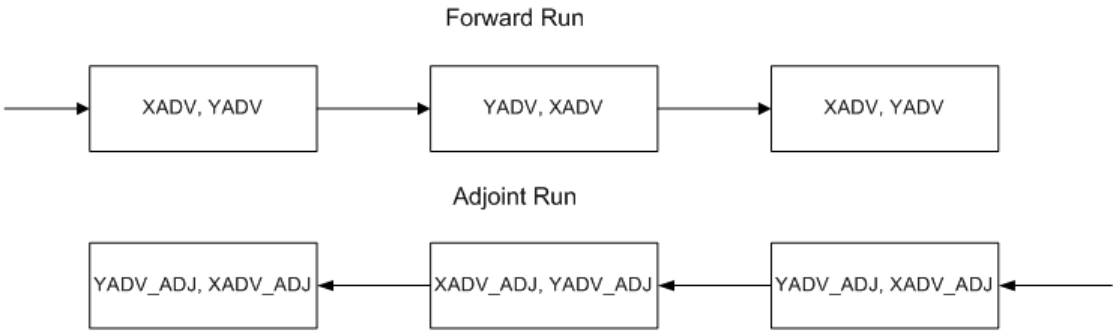

Figure 1. x and y advection execution order

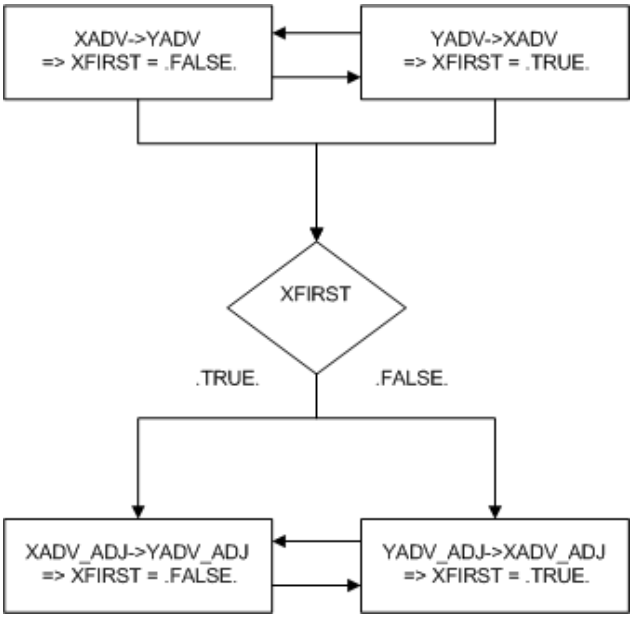

Figure 2. control of x and y advection execution in the forward and the adjoint run

Corrected time argument for the adjoint of diffusion and advection. In the adjoint run, the adjoint of advection and diffusion need correct wind velocity u and diffusion constant K respectively, for each time step. Therefore, the key issue is to provide the subroutine that read these inputs with correct time argument. In the forward run, from time step $t_n \rightarrow t_{n+1}$, the wind velocity is approximated by $u(t_{n+\frac{1}{2}})$ and the diffusion constant is approximated by $K(t_{n+\frac{1}{2}})$. Therefore, in the adjoint run, from time step $t_n \rightarrow t_{n-1}$, the wind velocity should be $u(t_{n-\frac{1}{2}})$ and the diffusion constant should be $K(t_{n-\frac{1}{2}})$.

Specifically, the subroutines that read u and K take t_n as the time input, then advance t_n half time step to get $t_{n+\frac{1}{2}}$ which is used to get the u and K. To get the $u(t_{n-\frac{1}{2}})$ and $K(t_{n-\frac{1}{2}})$, in the adjoint run time step $t_n \rightarrow t_{n-1}$, one approach is to provide the subroutines with time t_{n-1}. The new changes to the source codes have been made to make sure we follow this approach.

Discrete adjoint of advection. The discrete adjoint of advection is implemented in the current CMAQ-ADJ. This implementation includes the discrete adjoint of PPM, the new checkpoint files, the new drivers and sub-drivers for discrete adjoint of both vertical and horizontal advection.

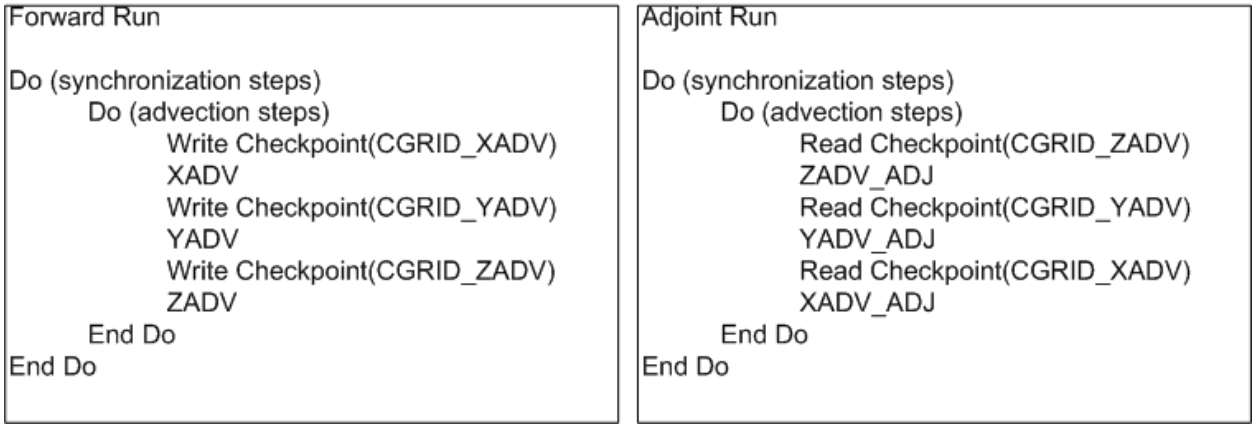

Figure 3. The sequence of function calls in the forward run and adjoint run for discrete adjoint of advection

Note: There is one issue in our current implementation of discrete adjoint of advection regarding the case that multiple advection time steps are within a synchronization step. In the forward run, the checkpoints are only performed at the synchronization steps due to the fixed checkpoint time step mechanism in CMAQ. If there are multiple advection time steps within a synchronization step, in the adjoint run, we need to perform the forward run again for that synchronization step to lay down the checkpoints for each advection time step and then use them in the adjoint run. However, our current implementation does not include the forward run checkpoint for the multiple advection sub-steps within a synchronization time step for a couple of reason: firstly, in our test experiments, the synchronization time step and advection step are the same; Secondly, our collaborator, Dr. Peter Percell has already implemented a version which fixed this issue and we decide to use the codes of his version in our later studies. Figure 3 shows the correct execution flow of the forward run and the adjoint run for discrete adjoint of advection.

Sensitivity to boundary conditions. Assuming first order upwind scheme is used to solve advection PDE, the sensitivity to boundary condition is computed by multiplying sensitivity to initial condition by a factor. The implementation in CMAQ is illustrated in figure 4.

Figure 4. Logical function call for sensitivity to boundary condition. (x advection)

Note:

(1) Units issues in current implementation. For the sensitivity $\delta b(kg) \rightarrow \delta\psi(ppbm)$, the units of boundary conditions and cost function are different. Boundary conditions uses mass-based unit. However, the cost function (defined on the concentration) uses mix-ratio unit. Therefore, while computing the sensitivity respect to boundary conditions, we need first convert units of the sensitivity to initial condition to $\delta c(kg) \rightarrow \delta\psi(ppbm)$ and then use the formula to get boundary sensitivities. This can be achieved by decoupling the adjoint variable before passing it to the subroutine which calculates the boundary

sensitivities. However, after the calculation, we need to couple the adjoint variable back. This decouple and couple method is only required for continuous adjoint approach since the discrete adjoint has already decoupled the adjoint variable before calculation.

(2) In current implementation, we calculate the sensitivities at synchronization time. We can derive to get that the sensitivity at synchronization time is the sum of the sensitivity at the advection time within the synchronization time. However, if the wind velocity and LGRID at different advection time are close, we can have the approximation formula below for sensitivity at synchronization time.

$$\frac{\partial \psi}{\partial b} = \sum \frac{U_{1_{advection}} \Delta t_{advection}}{\Delta x} \cdot \lambda_1(\text{advection time}) \approx \frac{U_{1_{syn}} \Delta t_{syn}}{\Delta x} \cdot \lambda_1(\text{syn time})$$

(3) There is no checkpoint file for the sensitivity to boundary condition at this time. One option is to write the sensitivity to a data file e.g. txt file.

(4) 4D-Var with boundary scaling factors and boundary sensitivities checkpoints are implemented in Dr. Peter Percell's code.

Source Changes in the CMAQ-ADJ.
File name changes:

Original CMAQ-ADJ	**New CMAQ-ADJ**
KPP_Data_mod.F	hrdata_mod.F
KPP_Pars.F	KPP_Parameters.F
KPP_HessianTE.F	KPP_Hessian.F
KPP_JacobianTE.F	KPP_Jacobian.F
KPP_LinAlg.F	KPP_LinearAlgebra.F
KPP_Glob.F	KPP_Global.F
kppIntegrator.F	KPP_Integrator.F
kppIntegrator_adj.F	KPP_Integrator_adj.F
KPP_Init.F	hrinit.F
kppcalcks.F	hrcalcks.F
kppdriver.F	hrdriver.F
kppdriver_adj.F	hrdriver_adj.F
kppModel.F	KPP_Model.F

New files and modified files for Continuous Adjoint of Advection

(1) Drivers and subdrivers:

File Name	**Comments**
sensitivity_driver.F	– open/close new checkpoints files: ADJFAC_CHK, CONC_XADVCHK, CONC_YADVCHK

senstdriver_cbwd.F	– subroutine to perform one forward and one backward run to generate adjoint trajectory for the use of continuous adjoint of advection
modsciproc.F	– controls all of the physical and chemical processes for a grid – used in the forward run for continuous adjoint of advection – use module XFIRSTM which contains XFIRST and initialize xfirst – calls ADJADV
modsciproc_cadj.F	– carry out science process adjoint calculations for sensitivity analysis with continuous adjoint for advection (include sensitivity w.r.t emission) – added ADJADV_ADJ – used new module XFIRSTM defining variable XFIRST – added deallocate DEBUFF
sciproc_cadj.F	– carry out science process adjoint calculations for sensitivity analysis with continuous adjoint for advection (not include sensitivity w.r.t emission) – use new module XFIRSTM
fd_driver.F	– open/close new checkpoints files ADJFAC_CHK, CONC_XADVCHK, CONC_YADVCHK
subdriver_fwd.F	– using new module XFIRSTM – initialize XFIRST to TRUE
fddriver_cbwd.F	– subroutine to perform one forward and one backward run to calculate the cost function and update LGRID for observation misfit only – using new module XFIRSTM – initialize XFIRST to TRUE – used for continuous adjoint of advection

(2) Module for x advection and y advection control

File Name	Comments
xfirstm.F	– Contains the variable xfirst used to control the order of executing x advection and y advection. – used in subdrivers which call xadv and yadv – Xfirst is initialized to TRUE in drivers which call sciproc.

(3) Adjoint of adjadv.F

File Name	Comments
wr_afchk.F	– subroutine to write the adjustment factor in the forward run to the checkpoint file: ADJFAC_CHK

rd_afchk.F	– subroutine to read adjustment factor in the backward run from checkpoint file: ADJFAC_CHK
adjadv.F	– adjusts concentration fields for mass balance inconsistencies in the forward model run – calls wr_afchk to write ADJFAC to checkpoint file
adjadv_adj.F	– adjoint of adjadv.F – adjusts concentration fields for mass balance inconsistencies in adjoint run – calls rd_afchk to read ADJFAC from checkpoint file

New files for discrete adjoint of advection

(1) Drivers and subdrivers:

File Name	Comments
senstdriver_dbwd.F	– build for the discrete adjoint – perform one forward and one backward run to generate adjoint trajectory – use XFIRSTM module and initialize FIRST
modsciproc_dfwd.F	– controls all of the physical and chemical processes for a grid – used in the forward run for discrete adjoint of advection – write the checkpoints for x, y, z advection – use module XFIRSTM which contains XFIRST and initialize xfirst – calls ADJADV
modsciproc_dadj.F	– carry out science process adjoint calculations for sensitivity analysis with discrete adjoint for advection (include sensitivity w.r.t emission) – use new module XFIRSTM defining variable XFIRST – added ADJADV_ADJ
sciproc_dadj.F	– carry out science process adjoint calculations for sensitivity analysis with discrete adjoint for advection (not include sensitivity w.r.t emission) – use new module XFIRSTM
fddriver_dbwd.F	– subroutine to perform one forward and one backward run to calculate the cost function and update LGRID for observation misfit only – used for discrete adjoint of advection

(2) Discrete adjoint of advection:

File Name	Comments
advppm.F	– discrete adjoint code of vppm.F used to integrate advection equation
adhppm.F	– discrete adjoint code of hppm.F used to integrate adjoint of advection equation

xadvppm_dad.F	– Advection DISCRETE ADJOINT in the horizontal plane; x1-direction.
yadvppm_dad.F	– Advection DISCRETE ADJOINT in the horizontal plane; x2-direction.
zadvppm_dad.F	– Advection DISCRETE ADJOINT in the vertical, x3-direction.

New files for Sensitivity to boundary conditions:

File Name	**Comments**
modsciproc_cadj_sb.F	– carry out science process adjoint calculations for sensitivity analysis with continuous adjoint for advection – LGRID_B are created to store sensitivities to boundary conditions and passed to horizontal advection process.
xadvppm_cad_sb.F	– Advection Continuous Adjoint in the horizontal plan; x1-direction. – Call stob subroutine to compute sensitivity to west or east boundary conditions.
yadvppm_cad_sb.F	– Advection Continuous Adjoint in the horizontal plan; x2-direction. – Call stob subroutine to compute sensitivity to north or south boundary conditions.
stob.F	– subroutine which implements the formula to compute the sensitivity to boundary condition.
modsciproc_dadj_sb.F	– carry out science process adjoint calculations for sensitivity analysis with discrete adjoint for advection – LGRID_B are created to store sensitivities to boundary conditions and passed to horizontal advection process.
xadvppm_dad_sb.F	– Advection Discrete Adjoint in the horizontal plan; x1-direction. – Call stob subroutine to compute sensitivity to west or east boundary conditions.
yadvppm_dad_sb.F	– Advection Discrete Adjoint in the horizontal plan; x2-direction. – Call stob subroutine to compute sensitivity to north or south boundary conditions.