Computational Tools for Chemical Data Assimilation with CMAQ

Tianyi Gou

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Science and Applications

Adrian Sandu, Chair
Yang Cao
Linsey C. Marr

January 11, 2010
Blacksburg, Virginia

Keywords: Chemical Transport Models, Data Assimilation, Adjoint Sensitivity Analysis

Computational Tools for Chemical Data Assimilation with CMAQ

Tianyi Gou

(ABSTRACT)

The Community Multiscale Air Quality (CMAQ) system is the Environmental Protection Agency's main modeling tool for atmospheric pollution studies. CMAQ-ADJ, the adjoint model of CMAQ, offers new analysis capabilities such as receptor-oriented sensitivity analysis and chemical data assimilation.

This thesis presents the construction, validation, and properties of new adjoint modules in CMAQ, and illustrates their use in sensitivity analyses and data assimilation experiments. The new module of discrete adjoint of advection is implemented with the aid of automatic differentiation tool (TAMC) and is fully validated by comparing the adjoint sensitivities with finite difference values. In addition, adjoint sensitivity with respect to boundary conditions and boundary condition scaling factors are developed and validated in CMAQ.

To investigate numerically the impact of the continuous and discrete advection adjoints on data assimilation, various four dimensional variational (4D-Var) data assimilation experiments are carried out with the 1D advection PDE, and with CMAQ advection using synthetic and real observation data. The results show that optimization procedure gives better estimates of the reference initial condition and converges faster when using gradients computed by the continuous adjoint approach. This counter-intuitive result is explained using the nonlinearity properties of the piecewise parabolic method (the numerical discretization of advection in CMAQ).

Data assimilation experiments are carried out using real observation data. The simulation domain encompasses Texas and the simulation period is August 30 to September 1, 2006. Data assimilation is used to improve both initial and boundary conditions. These experiments further validate the tools developed in this thesis.

Acknowledgments

I would like to thank my advisor, Dr. Adrian Sandu, for his help and guidance at all stage of this work. His insight on my research topic, encouragement and support are of great importance for the work in this thesis. I would also like to thank my committee members Dr. Linsey Marr and Dr. Yang Cao for their suggestions on my research.

This work was supported by the award HARC H98 from the Texas Commission on Environmental Quality. I would like to thank Dr. Daewon Byun, Dr. Tianfeng Chai, Dr. Peter Percell, Dr. Bernhard Rappenglueck, and Mr. Kumaresh Singh for their input, feedback, and continuous collaboration.

I would also like to thank my colleagues: Mihai Alex for the discussion of adjoint theory, Kumaresh Singh for the explanation of the original code of CMAQ-ADJ, John Linford for workstation system support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview of the Research

The chemical composition of the atmosphere has important implications for urban and regional air quality, for human health, and for climate change. Complex atmospheric chemical transport models (CTMs) are designed to describe the fate and transport of atmospheric chemical constituents associated with the gas and aerosol phases.

The Community Multi-scale Air Quality (CMAQ) system is a powerful third generation modeling tool which is capable of addressing important air quality issues such as tropospheric ozone, acid deposition, fine particles and visibility degradation and is used to predict the state of the airborne pollutants [1, 2, 10].

A key prerequisite for good forecasting of atmospheric composition is the optimal estimation of the state of the atmosphere. Data assimilation optimally uses the observations (reality within errors) along with the model (imperfect representation of the processes and their connections) to produce a better estimate (in some optimal sense) of the state of the atmosphere and of key parameters such as emissions and boundary conditions.

In practice, three main methods of data assimilation are widely used: optimal interpolation, variational methods, and stochastic methods. Optimal interpolation distributes (interpolates) the observations onto grid points using weights that reflect the statistics of the observation errors (e.g., covariance matrices). Although simple to implement, this method produces retrieved fields that may not be consistent with the physical states as it completely ignores the governing equations of the atmosphere. In the variational approach data assimilation is formulated as a constrained optimization problem which minimizes the discrepancy between model predictions and the observations. The main drawback of this approach is the assumption of the static background error covariances. Stochastic methods are based on the Kalman filter (KF) theory and include several variations of the ensemble Kalman

filter (EnKF) and particle filters. The EnKF provides the estimation of background error covariances from a small ensemble of short term forecasts. However, the major shortcoming of the ensemble-based stochastic approach is the poor representation of the huge covariance matrices (in operational models) via small ensembles [3].

In this thesis we focus on the four dimensional variational (4D-Var) approach to data assimilation. 4D-Var provides an optimal control framework to integrating spatially and temporally distributed observations into models. Mathematically, 4D-Var is formulated as a constrained optimization problem that seeks the best parameter values to minimize the misfit between the model predictions and observations as well as the misfit between the optimal parameters and a background estimate of them. The constraints are the chemical transport model equations.

Sensitivity analysis is the approach to analyze how the model outputs change given a perturbation of the model inputs. In air quality modeling, this information is of great importance for air pollution control policy, data assimilation, inverse modeling, parameter estimation, etc. For example, to solve the 4D-Var optimization problem, one typically employs a gradient based numerical procedure. Sensitivity analysis provides the required gradients.

In general, there are two approaches to obtain the sensitivities/gradients. One is the direct method where each component of the gradient is calculated by performing a (tangent linear) model run; the direct method is not efficient when the number of model parameters is large. The second approach to obtaining the gradient is the adjoint method, which requires one forward model run followed by one backward adjoint model run. The adjoint method is effective when gradients are needed with respect to a large number of model parameters, e.g., when data assimilation seeks an optimal initial condition.

The construction of an adjoint model is a highly complex, time consuming, and error prone process. To construct the adjoint model, two approaches are possible. In the *continuous adjoint* approach one first differentiates the underlying physical equations at an abstract level, and then discretizes the resulting adjoint partial differential equations. In the *discrete adjoint* approach one first discretizes the physical equations (using suitable numerical methods), then differentiates the discrete algorithm. Discrete adjoints are popular since they can be obtained by automatic differentiation[8]. In general the two approaches lead to different adjoint models; the choice of the adjoint model has important implications on the performance of the data assimilation scheme.

1.2 Outline of the Thesis

The rest of thesis is organized as follows. Chapter 2 discusses the construction and validation of continuous and discrete adjoint models for the advection equation. The nonlinearity of the piecewise parabolic method (PPM) used to discretize advection and its effect on different types of adjoints are discussed. Chapter 3 presents our work on the adjoint sensitivity with

respect to boundary conditions. The implementation and results are shown for both a 1D advection PDE and for CMAQ. Chapter 4 reports data assimilation results using real data for both initial condition control and boundary condition control. Chapter 5 summarizes the accomplishments of this thesis and suggests directions of future research.

Chapter 2

Continuous versus Discrete Advection Adjoints

In this chapter, we present the construction and validation of the continuous and discrete adjoint model of advection. In addition, we also discuss the issues of nonlinearity of the piecewise parabolic method (PPM) and its impact on the properties of adjoint.

2.1 Construction of Continuous and Discrete Adjoints of Advection

The implementation of advection in CMAQ is done using a dimensionally split approach on the Cartesian grid. The one dimensional advection problems in the x, y, and z directions are solved in succession during one time step. Each one dimensional advection problem is discretized using the piecewise parabolic method (PPM.)

Following the directional split approach, a three-dimensional advection adjoint can be constructed by successively solving the adjoint advection in the z, y, and x directions (respectively) during each reverse time step. Both continuous and discrete approaches can be employed to construct the adjoint of the one-dimensional advection. We next discuss each of these approaches.

Continuous advection adjoint. Consider the one-dimensional advection partial differential equation

$$\frac{\partial(\rho\, c)}{\partial t} = -\frac{\partial(u \cdot \rho\, c)}{\partial x} \tag{2.1}$$

where $c(t, x)$ is the concentration vector (a function of time and space), $\rho(t, x)$ is the air density, and $u(t, x)$ is the x component of the wind field.

The corresponding adjoint partial differential equation [10] is

$$\frac{\partial(\lambda/\rho)}{\partial t} = -\frac{\partial(u \cdot \lambda/\rho)}{\partial x} \tag{2.2}$$

where λ is adjoint variable. Note that (2.2) is itself an advection equation of the form (2.1), applied to λ/ρ (instead of ρc) and solved backwards in time. Formally the equation (2.2) can be solved forward in time, but with the wind field changed from u to $-u$ to account for the different time direction. The continuous adjoint equation is discretized with the piecewise parabolic method. To be specific, the continuous adjoint is implemented by calling the forward advection subroutine with a reversed wind field ($-u$) and with properly scaled adjoint variables.

Discrete advection adjoint. Discrete adjoint of advection is implemented with the aid of the automatic differentiation tool TAMC (Tangent Linear and Adjoint Model Compiler) [8]. TAMC is applied to (one step of) the piecewise parabolic method to obtain (one step of) the corresponding discrete adjoint. Due to the nonlinearity of the PPM scheme, the discrete adjoint depends not only on the adjoint variable λ but also on the forward concentration c. During the forward run concentrations are checkpointed into files at the beginning of each dynamic time step. These concentrations are read during the corresponding adjoint step.

Modified discrete advection adjoint. PPM ensures monotonicity properties with the help of slope and curvature limiters. The reverse mode differentiation of these limiters may introduce discontinuities in the discrete adjoint solution [4]. In the modified discrete adjoint approach the monotonicity characteristics of the PPM are removed by commenting out the steepening procedure (i.e., the limiters). The resulting algorithm is smooth and has no points of non-differentiability. This smooth (but non-monotonic) code is then processed by automatic differentiation to obtain the modified discrete adjoint code.

2.2 Adjoint Validation

To validate the adjoints of vertical and horizontal advection, we perform the finite difference tests and present scatter plots of the sensitivities given by finite difference approach and adjoint approach.

2.2.1 1D Sensitivity Studies

We first consider a test case based on the one dimensional advection PDE (2.1). The spatial domain is $x \in [0, 2\pi]$ with periodic boundary conditions $c(t, 0) = c(t, 2\pi) = 1.0$. The wind

field is constant, $u = 1$. The initial conditions are $c(t_0, x) = 1.0 + \sin(3x)$. The computational grid has 200 points evenly distributed in the spatial domain. The initial time and final time are $t_0 = 0$ and $t_F = 1.17$ respectively.

The cost function (receptor) is defined as the sum of forward solutions at gridpoints [a, b] at the final time:

$$\psi = \sum_{i=a}^{b} c\left(t_F, x_i\right)$$

The adjoint advection PDE (2.2) has periodic boundary conditions, and the initial condition

$$\lambda\left(t_F, x_i\right) = \begin{cases} 1 & \text{for } \; a \le i \le b \; , \\ 0 & \text{otherwise} \; . \end{cases}$$

Solutions of the adjoint system (2.2) at the initial time represent the sensitivities of the cost function (receptor) with respect to the initial condition (source)

$$\lambda\left(t_0, x_i\right) = \frac{d\,\psi}{d\,c\left(t_0, x_i\right)} \; .$$

The continuous and discrete adjoint subroutines from CMAQ-ADJ are used to solve our 1D advection adjoint system. Figure 2.1 shows the solution of the forward system at initial and final time. Figure 2.2 shows solutions of the continuous and discrete adjoint systems at initial time for different receptors. Backward integration of continuous adjoint and discrete adjoint system starts with the same final time condition but solutions become different as the time evolves. From figure 2.2, we find that discrete adjoint solutions and modified discrete adjoint solutions show non-smoothness (wiggles) at corners, however the continuous adjoint solution is smooth.

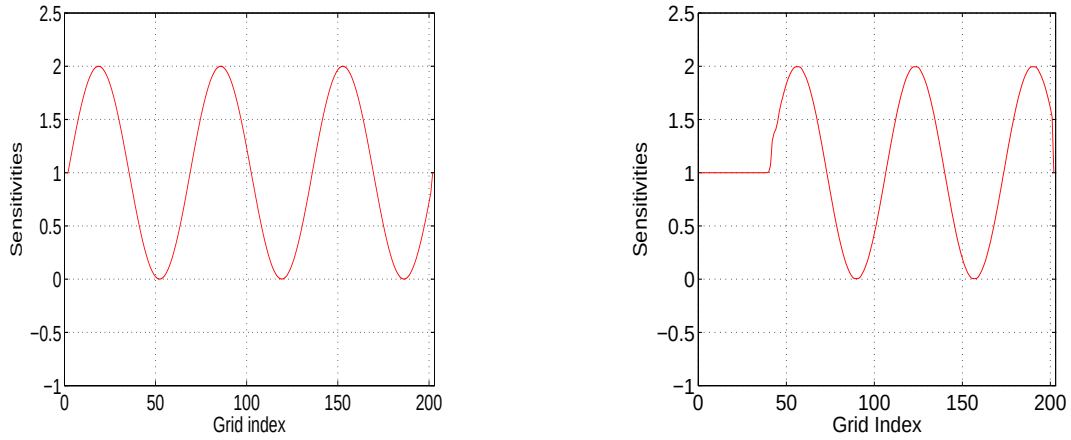
2.2.2 3D Sensitivity Studies

We now validate the advection adjoints in the full three-dimensional CMAQ code. The tests are essentially one dimensional, as all the processes except one-dimensional advection are turned off.

For each validation case we compute the R^2 and RMS correlation factor between the adjoint and the finite difference sensitivity values. The R^2 and RMS correlation factors of two series X and Y of length n are defined as

$$\mathrm{R}^2(X,Y) = \frac{\left(n\sum_{i=1}^{n} X_i\,Y_i - \sum_{i=1}^{n} X_i \sum_{i=1}^{n} Y_i\right)^2}{\left(n\sum_{i=1}^{n} X_i^2 - \left(\sum_{i=1}^{n} X_i\right)^2\right)\left(n\sum_{i=1}^{n} Y_i^2 - \left(\sum_{i=1}^{n} Y_i\right)^2\right)},$$

$$RMS(X,Y) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(X_i - Y_i\right)^2} \; .$$



(a) fwd solution at initial time (b) fwd solution at final time

Figure 2.1: Solutions of the forward system of 1D advection PDE. (a) initial condition of the forward system; (b) solution of the forward system at finial time.

Validation of Vertical Advection Adjoints

To validate the adjoints of vertical advection, we compare the adjoint sensitivities with central finite difference sensitivities. Specifically, we choose a source layer ($L_s = 3$), a receptor layer ($L_r = 2$), and a species ($S = O_3$, ozone). Sensitivities of the receptor with respect to source, $dc(L_r, S)/dc(L_s, S)$, are computed using both central finite difference approach and the adjoint sensitivity approach. Since only vertical advection is considered, each column for the source layer and the receptor is independent. As a result, we get NCOLS $\times$ NROWS independent sensitivities coefficients. Fig 2.3 (a) shows the scatter plot of continuous adjoint sensitivity vs. central finite difference values for a 20 hours run of CMAQ-ADJ with vertical advection only. Each point represents one sensitivity (one column), with the x coordinate the value obtained by the adjoint method and the y coordinate the value obtained by central differences. Fig. 2.3 (b) shows a similar scatter plot for discrete adjoint sensitivities. Table 2.1 lists the R^2 and RMS corresponding to the scatter plots results. From the scatter plots and the R^2, RMS metrics, we see that discrete adjoint sensitivities of vertical advection agree better with finite difference results than continuous adjoint sensitivities.

(X,Y)	$R^2(X,Y)$	RMS(X,Y)
(continuous adj. vs. finite difference)	0.43	0.49
(discrete adj. vs. finite difference)	0.99	0.03

Table 2.1: R^2 and RMS values for the vertical advection adjoint validation

Validation of Horizontal Advection Adjoints

Because x advection and y advection are equivalent (they are implemented by calling the same hppm routine, and provide similar test results), we only show validation of x advection results in this section. To generate point-to-point sensitivities test, we choose the source as cells at column 25 and all rows and layers for species O_3 at initial time. The receptor is defined to measure ozone in all cells at column 26 for the same rows and layers as source at final time. Since only x advection is involved, each component of sensitivity of the receptor with respect to the source is the same as the sensitivity of the single cell at column 26 for a certain row and layer at final time with respect to the single cell at column 25 for the same row and layer at initial time. Therefore, we have the NROWS x NLAYS independent pairs of sensitivities (finite difference sensitivity and adjoint sensitivity).

Figure 2.4 presents the scatter plots of adjoint versus finite difference sensitivities. Each point represents the sensitivity in a different column. The x-coordinates are the adjoint values and the y-coordinates are the central finite difference values of the same sensitivity coefficient. Figure 2.4 (a) use the continuous horizontal advection adjoint while Figure 2.4 (b) and (c) present the discrete horizontal advection and modified horizontal advection results, respectively. R^2 and RMS values for three adjoint approaches can be found in table 2.2. Again, a comparison of the plots shows that the discrete adjoint results and modified adjoint results are in better agreement with the finite difference results than the continuous adjoint values.

(X, Y)	$R^2(X,Y)$	RMS(X,Y)
continuous adj. vs. finite difference	0.77	0.17
discrete adj. vs. finite difference	0.97	0.05
modified discrete adj. vs. finite difference	0.86	0.11

Table 2.2: R^2 and RMS values for the horizontal advection adjoint validation

2.3 Nonlinearity of the PPM Numerical Scheme

From previous sections, we found that discrete adjoint sensitivities agree better with finite difference sensitivities than their continuous adjoint counterparts. One of the reasons is that both discrete adjoint and finite difference approaches compute the sensitivities of the numerical solutions with respect to changes in model inputs, while continuous adjoint give the approximation of the sensitivities of the model solutions with respect to changes in model inputs. In our case, the nonlinearity property of PPM is the main reason for the discrepancy between continuous adjoint sensitivities and the finite difference sensitivities.

In the following discussion, we at first show the PPM scheme is nonlinear and then explain in detail the link between nonlinearity and the discrepancy between continuous adjoint and

finite difference sensitivities.

To show the PPM method is not linear, we consider a 1D linear advection PDE (2.1) and transport a delta function, the profiles at initial time and final time are shown in Figure 2.5 (a)(b). Next, we consider a "base" smooth profile and advect it forward in time. We add a delta perturbation to the smooth profile, and transport the perturbed profile forward in time. The differences between the perturbed and the "base" profiles at both initial time and final time are shown in Figure 2.5 (c)(d). The transported delta function (Figure 2.5(b)) and the difference of the base and perturbed profiles (Figure 2.5(d)) are very different, which indicates the PPM scheme is strongly nonlinear. (If advection was solved exactly then these two profiles would have been the same due to linearity).

Our question is how this nonlinearity of the PPM scheme affects adjoint sensitivity and finite difference sensitivity (therefore the validation results). To answer this question, we perform another validation of the adjoint sensitivity with different receptors. The sensitivity analysis problem is defined as follows.

The cost function/receptor is:

$$\psi = \sum_{i=a}^{b} c(t_F, x_i) \tag{2.3}$$

where $c(t, x)$ is the solution of the 1D advection PDE: (2.1) with initial condition $c(0, x) = 1 + sin(x)$, and boundary condition $c(t, 0) = c(t, 2\pi) = 1$ as well as constant wind velocity $u = 1$. $x \in [0, 2\pi]$, i is the index of equally spaced computational grids.

The source (perturbation location) is $c(0, x_1)$, initial condition at x_1.

We want: $\partial\psi/\partial c(0, x_1)$, the sensitivity of the cost function (receptor) with respect to the initial condition at the x_1 (source).

Two approaches are used to get $\partial\psi/\partial c(0, x_1)$. One is finite difference approach and the other one is continuous adjoint approach. Table 2.3 shows the comparison between finite difference sensitivity and adjoint sensitivity. For finite difference test, the perturbation of the source is 0.1%.

Receptor	Δ Finite Difference	Δ Con. Adj.	Relative Error
a = 0 , b = 70	0.00	2.68E-08	0.00 %
a = 71 , b = 97	9.99E-04	9.99E-04	0.00 %
a = 98 , b =202	0.00	4.74E-25	0.00 %
a = 75 , b = 80	1.01E-03	9.56E-04	-5.59 %
a = 81 , b = 86	-1.27E-04	1.23E-08	1026388 %
a = 75, b = 85	9.43E-04	9.58E-04	1.59 %

Table 2.3: Finite difference validation of continuous adjoint gradients for different receptors

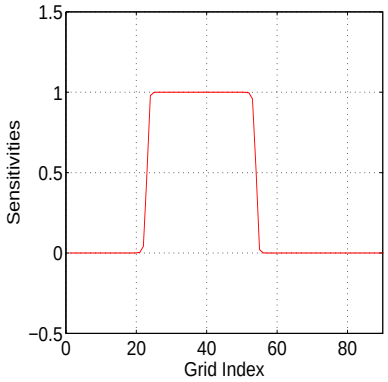
The results in Table 2.3 show that the agreement between the continuous adjoint and finite

difference sensitivities depends strongly on the choice of the receptor. This is caused by the effect of nonlinearity of PPM to the finite difference results. To explain this we revisit Figures 2.5. Figure 2.5(c)(d) show the difference of the perturbed and base solutions which corresponds to the finite difference sensitivity in Table 2.3. Figure 2.5(a)(b) show what the difference of two solutions would be if the PPM scheme was linear, and corresponds to the adjoint (linearized sensitivity) case of Table 2.3. In Figure 2.5(d), there are wiggles resulted from nonlinearity of PPM which do not appear in the linearized solution (Figure 2.5(b)). This implies that if receptors are chosen at grids where wiggles are present, the finite difference sensitivity is not accurate it leads to a big disagreement between finite difference sensitivity and continuous adjoint sensitivity. For example, when we choose the receptor as the sum of the solutions at grids 81 to 86, the relative difference is very large. However, this big discrepancy is mainly due to the inaccurate finite difference sensitivity. From figure 2.5(b) (the linearized case), we see that $\delta\psi$ should be very close to 0 - but in finite difference test the change in the cost function is large.

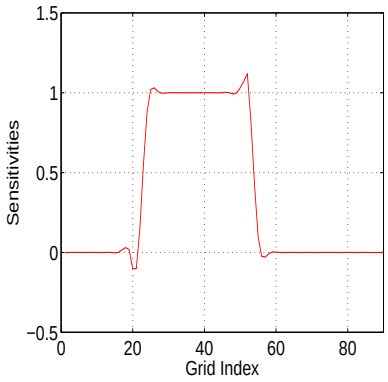
To sum up, the large discrepancy between continuous adjoint sensitivities and finite difference sensitivities is caused by nonlinearity of PPM scheme. Specifically, the finite difference sensitivity (the difference between a perturbed and a base solution) is a wiggly profile that depends not only on perturbation, but on the base solution as well. The continuous adjoint sensitivity (the PPM scheme applied to propagate the initial perturbation) is a smooth solution due to the monotonic properties of the scheme.

We expect the large differences between the profile difference and the linearized profiles to lead to differences between the continuous adjoint and the finite difference sensitivities. The magnitude of those differences greatly depends on the choice of the receptor. Good agreement is observed when the receptors include the real change in the solution profile, but they do not include the spurious wiggles resulting from the nonlinearity of the scheme. On the other hand when a pointwise receptor is located on a wiggle the finite difference result and the linearized (adjoint) result can be arbitrarily different; for example, the finite difference can give a negative result while the adjoint sensitivity is positive.

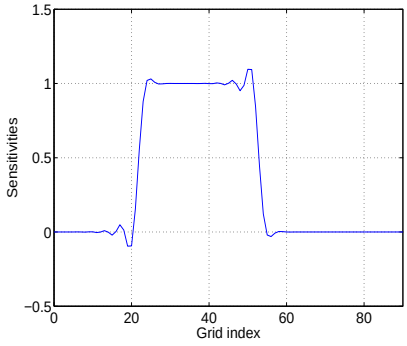
One practical way to overcome this problem is to always choose a large receptor area (e.g., an integral of concentrations over several grid cells). In this case one "integrates" over the wiggles and cancels out their spurious contribution, while capturing the real change in the concentration profile (and therefore, in the cost function).



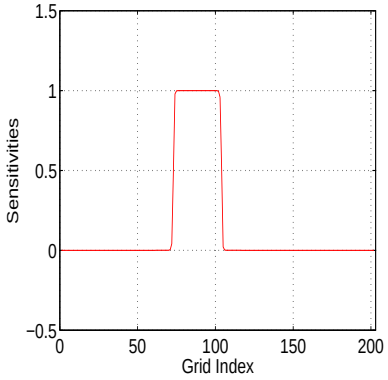
(a) continuous adjoint sensitivities (receptor a:60, b:90)



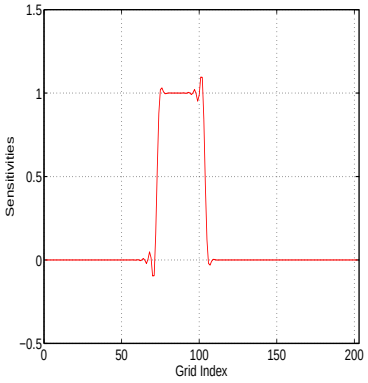
(b) discrete adjoint sensitivities (receptor a:60, b:90)



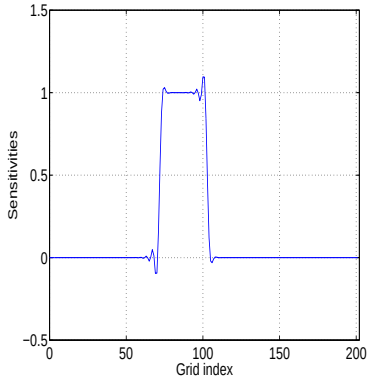
(c) modified discrete adjoint sensitivities (receptor a:60, b:90)



(d) continuous adjoint sensitivities (receptor a:110, b:140)

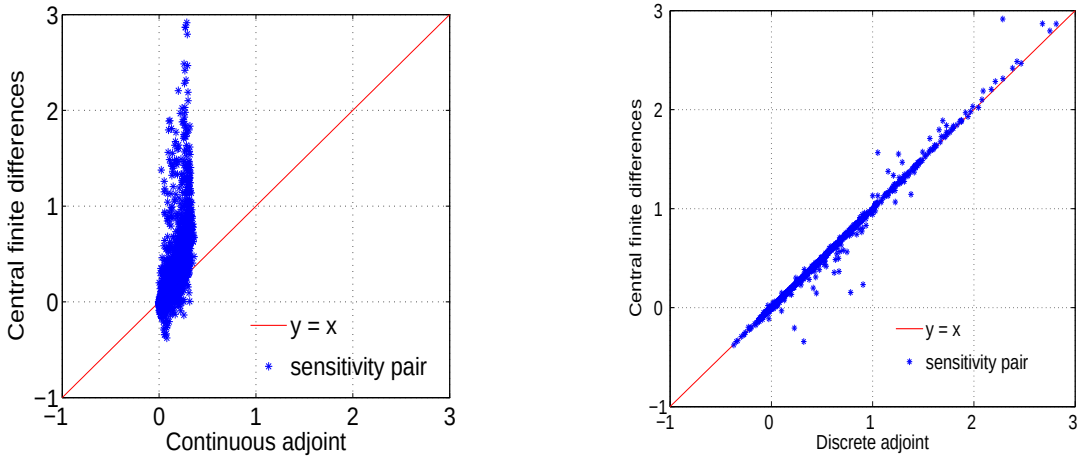


(e) discrete adjoint sensitivities (receptor a:110, b:140)



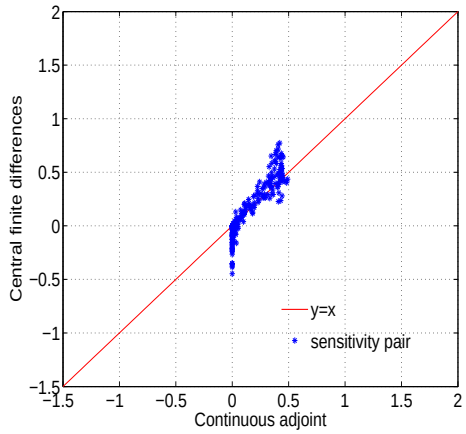
(f) modified discrete adjoint sensitivities (receptor a:110, b:140)

Figure 2.2: Solutions of the adjoint system of 1D advection PDE using continuous adjoint approach and discrete adjoint approach.(a)-(c): solutions of the continuous adjoint system, discrete adjoint system and modified discrete adjoint system at initial time for receptor $\sum_{i=60}^{90} c(i, t_F)$; (d)-(f): solutions of the continuous adjoint system, discrete adjoint system and modified discrete adjoint system at initial time for the receptor $\psi = \sum_{i=110}^{140} c(t_F, i)$.

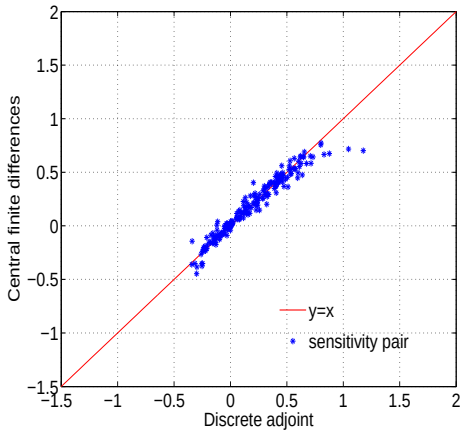


(a) continuous adjoint vs. finite difference (b) discrete adjoint vs. finite difference

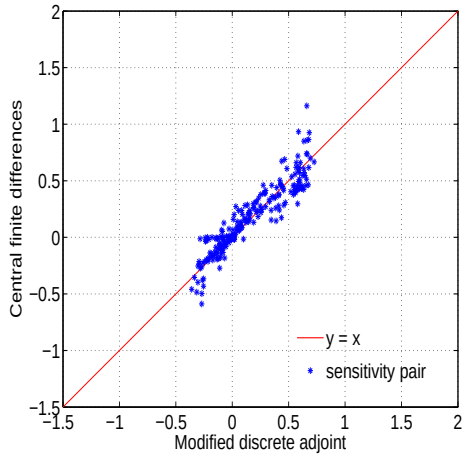
Figure 2.3: Scatter plot of (a) continuous adjoint sensitivity and (b) discrete adjoint sensitivity vs. finite difference values with source = (col=1:NCOLS, row=1:NROWS, lay=3, spc = O3) at time 1999183:000000 and receptor = (col=1:NCOLS, row=1:NROWS, lay= 2, spc = O3) at time 1999183:200000. Vertical advection only.



(a) continuous adjoint vs. finite difference

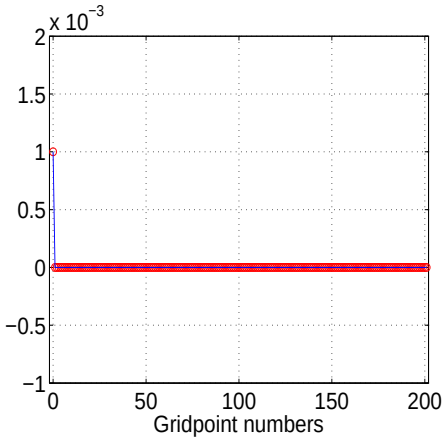


(b) discrete adjoint vs. finite difference

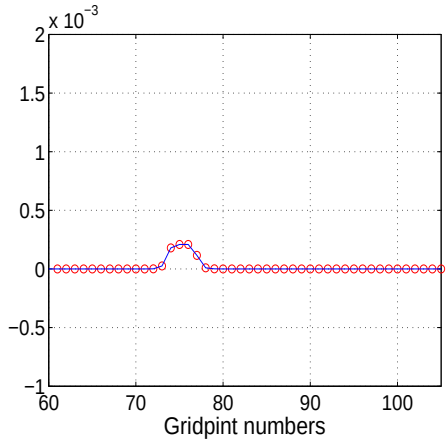


(c) modified discrete adjoint vs. finite difference

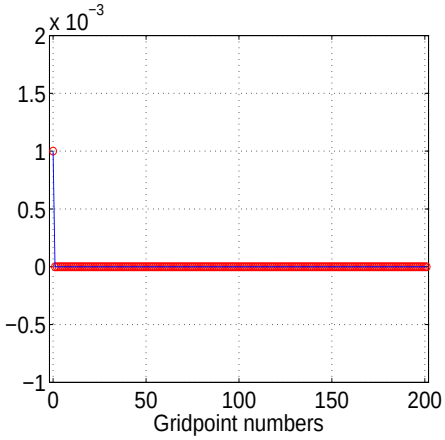
Figure 2.4: Scatter plot of (a) continuous adjoint sensitivities and (b) discrete adjoint sensitivities (c) modified discrete adjoint sensitivities vs. finite difference values with source = (col=25, row=1:NROWS, lay=1:NLAYS, spc = O3) at time 1999183:000000, receptor = (col=26, row=1:NROWS, lay=1:NLAYS, spc = O3) at time 1999183:020000. Horizontal (x) advection only



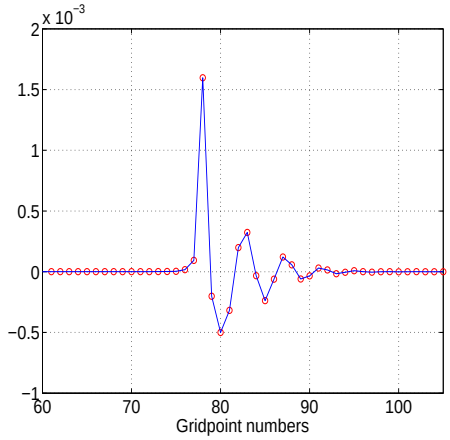
(a) Delta function at initial time



(b) Delta function at final time transported by PPM



(c) Delta perturbation at initial time (difference of perturbed and base profiles)



(d) Perturbation at final time (difference of perturbed and base profiles)

Figure 2.5: Perturbation of the initial condition (difference between perturbed solution and base solution) transported in time along 1D axis.

Chapter 3

Adjoint Sensitivity with respect to Boundary Conditions

In this chapter we discuss the work on developing adjoint sensitivities with respect to boundary conditions and boundary condition scaling factor for CMAQ. First, we give the formula used to compute boundary sensitivities. We then show validation results first in a simple 1D advection problem, then for the current implementation of boundary sensitivity in CMAQ.

3.1 Sensitivity to Boundary Conditions and Boundary Condition Scaling Factors for 1D Advection

3.1.1 Sensitivity to Boundary Conditions

Let $\psi(c(t,x_i))$ be a cost function defined on $\{c(t,x_i), i=1,...,N\}$. $c(t,x_i)$ is the concentration of the control variable at time t and gridpoint x_i. Let $c(t,x_0)$ and $c(t,x_{N+1})$ be the boundary conditions. The sensitivity of the cost function with respect to the boundary conditions depends on adjoint variables as follows:

$$\lambda_0 = \left(\frac{\partial\psi(c(t,x_i))}{\partial c(t,x_0)}\right)^T = \left(\sum_{i=1}^{N}\frac{\partial\psi}{\partial c(t,x_i)}\cdot\frac{\partial c(t,x_i)}{\partial c(t,x_0)}\right)^T = \sum_{i=1}^{N}\left(\frac{\partial c(t,x_i)}{\partial c(t,x_0)}\right)^T\cdot\left(\frac{\partial\psi}{\partial c(t,x_i)}\right)^T \tag{3.1}$$

since

$$\left(\frac{\partial \psi}{\partial c(t,x_i)}\right)^T = \lambda_i, \quad i = 1, ..., N \tag{3.2}$$

we have that:

$$\lambda_0 = \left(\frac{\partial \psi(c(t,x_i))}{\partial c(t,x_0)}\right)^T = \sum_{i=1}^{N} \left(\frac{\partial c(t,x_i)}{\partial c(t,x_0)}\right)^T \cdot \lambda_i \tag{3.3}$$

Therefore, in order to obtain λ_0, we need the derivatives $\frac{\partial c_i}{\partial c_0}$. These derivatives represent the dependency of different concentrations on the boundary condition at the current time step. They depend on the particular numerical discretization used for one time step of the advection equation.

Considering the advection process and assuming the numerical scheme is the first order upwind scheme, we have:

$$c(t^{new}, x_1) = c(t^{old}, x_1) - U_1 \Delta t \frac{c(t^{old}, x_1) - c(t^{old}, x_0)}{\Delta x} \tag{3.4}$$

So

$$\frac{\partial c(t,x_1)}{\partial c(t,x_0)} = \frac{U_1 \Delta t}{\Delta x} \quad and \quad \frac{\partial c(t,x_j)}{\partial c(t,x_0)} = 0, j = 2, .., N \tag{3.5}$$

Therefore,

$$\lambda_0 = \left(\frac{\partial c(t,x_1)}{\partial c(t,x_0)}\right)^T \cdot \lambda_1 = \frac{U_1 \Delta t}{\Delta x} \cdot \lambda_1 \tag{3.6}$$

Similarly, for λ_{N+1}, we have

$$\lambda_{N+1} = \left(\frac{\partial \psi(c(t,x_i))}{\partial c(t,x_{N+1})}\right)^T = \sum_{i=1}^{N} \left(\frac{\partial c(t,x_i)}{\partial c(t,x_{N+1})}\right)^T \cdot \lambda_i \tag{3.7}$$

Since

$$c(t^{new}, x_N) = c(t^{old}, x_N) - U_N \Delta t \frac{c(t^{old}, x_{N+1}) - c(t^{old}, x_N)}{\Delta x} \tag{3.8}$$

We have:

$$\frac{\partial c(t,x_N)}{\partial c(t,x_{N+1})} = -\frac{U_N \Delta t}{\Delta x} \quad and \quad \frac{\partial c(t,x_j)}{\partial c(t,x_{N+1})} = 0, j = 1, ..., N-1 \tag{3.9}$$

Therefore,

$$\lambda_{N+1} = \left(\frac{\partial c(t,x_N)}{\partial c(t,x_{N+1})}\right)^T \cdot \lambda_N = -\frac{U_N \Delta t}{\Delta x} \cdot \lambda_N \tag{3.10}$$

When the numerical discretization is not the first order upwind, these formulas represent the sensitivities with respect to boundary conditions within an approximation error.

3.1.2 Sensitivity to Boundary Condition Scaling Factors

Let B^{old} be the base boundary conditions and α be the scaling factor of the base boundary conditions. Consequently, $b = \alpha B^{old}$ is the new boundary conditions after the base boundary is scaled by α. The same (fixed) scaling factor is applied to the boundary values at all time steps. We are interested to find the sensitivity of a cost function with respect to such boundary scaling factors.

The sensitivity of the cost function (ψ) with respect to the scaling factor is derived as follows:

$$\frac{\partial \psi}{\partial \alpha} \mid_{t_n} = \frac{\partial \psi}{\partial b} \cdot \frac{\partial b}{\partial \alpha} = \frac{\partial \psi}{\partial b} \cdot \frac{\partial (\alpha B^{old})}{\partial \alpha} = B^{old} \cdot \frac{\partial \psi}{\partial b} \mid_{t_n} \tag{3.11}$$

$\frac{\partial \psi}{\partial b} \mid_{t_n}$ is the sensitivity of the cost function with respect to the boundary condition at time t_n. Therefore, we can easily get the sensitivity to scaling factor after we get sensitivity to the boundary conditions by just multiplying it with B^{old}.

Since α is constant in time, we have

$$\frac{\partial \psi}{\partial \alpha} \mid_{overall} = \sum_{t_n} \frac{\partial \psi}{\partial \alpha} \mid_{t_n} \tag{3.12}$$

3.1.3 Validation of Adjoint Sensitivity to Boundaries on the 1D Advection PDE

To validate the adjoint sensitivity with respect to the boundary conditions and the scaling factor, we first compare the adjoint sensitivity with finite difference values on 1D advection PDE (2.1) with $x \in [0, 2\pi]$, the initial condition is $c(0, x) = 1 + sin(x)$, and boundary condition is $c(t, 0) = c(t, 2\pi) = 1$ (only the inflow condition is actually used). We solve this 1-D advection using the PPM method implemented in CMAQ. The boundary sensitivities computed with the formulas discussed above are approximations of the discrete boundary sensitivities on PPM.

Validation of sensitivities with respect to boundary conditions. Consider the cost function:

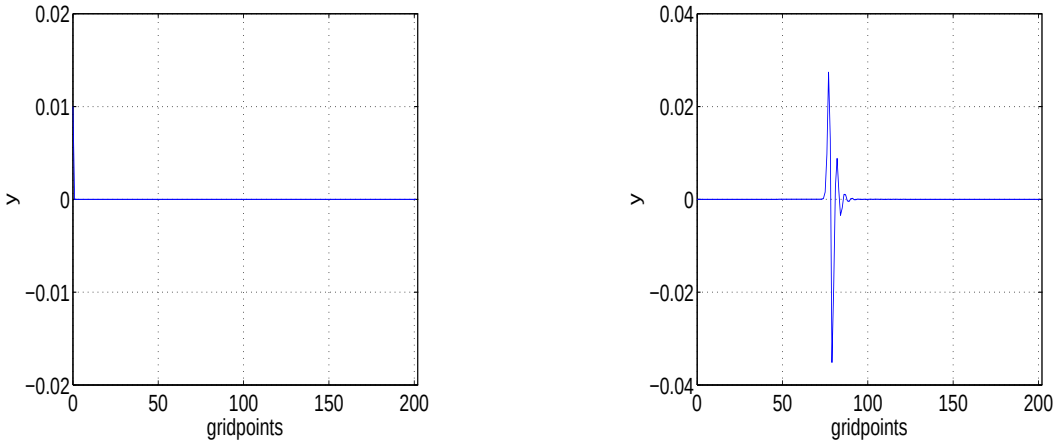
$$\psi = \sum_{i=a}^{b} c(t_F, x_i) \tag{3.13}$$

where a and b are grid indices from a total of 200 internal gridpoints. We want to validate the computation of the sensitivity: $\frac{\partial \psi}{\partial b} \mid_{t_0}$

To test continuous adjoint sensitivity we compare the finite difference result $\psi(b+\delta b)-\psi(b)$ against the product between the sensitivity coefficient and the perturbation in the boundary, $\frac{\partial \psi}{\partial b} \mid_{t_0} \cdot \delta b$. Here $\frac{\partial \psi}{\partial b} \mid_{t_0}$ is the adjoint sensitivity to boundary value. The results are shown in Table 3.1. We notice that for some receptor choices the results are very accurate (e.g., gridpoints 71-97) while for other choices the results are very inaccurate (e.g., gridpoints 75-80).

Receptor	$\Delta\psi$ Finite Difference	$\Delta\psi$ Continuous Adjoint	Relative Error
a = 0, b= 70	0.00	2.01E-7	0.00%
a = 71, b = 97	7.49E-3	7.50E-3	0.15%
a = 98, b = 202	0.00	3.56E-24	0.00%
a = 75, b = 80	-2.44E-3	7.17E-3	134.06%
a = 81, b = 86	9.29E-3	9.27E-8	-1.0E7 %
a = 75, b = 85	5.77E-3	7.19E-3	19.73 %

Table 3.1: Left boundary Sensitivity: Finite Difference vs. Adjoint.



(a)δb (left) at initial time (b) δc at final time

Figure 3.1: Evolution of a perturbation in the left boundary condition (inflow boundary). (a) Perturbation at left boundary at initial time. (b) Difference between the perturbed and base solutions at the receptor at final time.

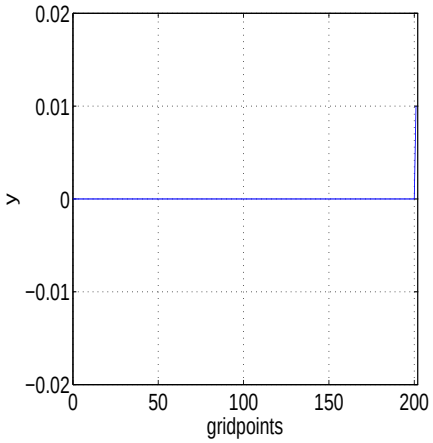
In order to better understand how a perturbation of the boundary condition at the initial time affects the 1-D advection PDE solution, we plot the difference between the solutions with perturbed and base boundary values at the final time. The results are shown in figure 3.1. For meaningful finite difference results the receptor needs to be chosen such as to include the grids whose values are changed by the change in the boundary condition. In our test case, we see that solutions at grids 71-97 are changed. When the receptor includes the entire area where the solution changes, a good match between finite difference and adjoint is

obtained. However, if we choose other receptors, we may get very large differences between finite differences and adjoint. This is again caused by the nonlinearity of the PPM method.

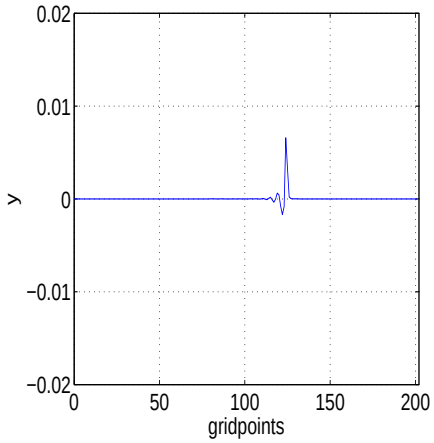
A similar analysis was carried out for the right boundary condition, the wind is negative in this experiment. Validation results for the right boundary sensitivity are shown in Table 3.2.

Receptor	$\Delta\psi$ Finite Difference	$\Delta\psi Continuous Adjoint$	Relative Error
a = 0, b= 104	0.00	1.74E-23	0.00%
a = 105, b = 130	7.50E-3	7.50E-3	0.00%
a = 131, b = 202	0.00	2.01E-7	0.00%
a = 105, b = 110	2.38E-7	1.30E-18	-1.83E13%
a = 110, b = 115	1.74E-4	6.55E-13	-2.66E10 %
a =118, b = 123	-2.29E-3	9.48E-5	2.51E3 %
a =118, b = 130	7.79E-3	7.50E-3	-3.88 %

Table 3.2: Right boundary Sensitivity: Finite Difference vs. Adjoint.



(a)δb (right) at initial time



(b) δc at final time

Figure 3.2: Evolution of a perturbation of the right boundary condition. (a) Perturbation at right boundary at initial time. (b) Difference between perturbed and base solutions at final time.

The conclusions of the right boundary perturbation test are similar to the ones obtained from the left boundary tests. The agreement between adjoint and finite differences depends on how the receptor is chosen. If we choose the receptor to include all the changes in the solution at the final time, then the adjoint result and the finite difference result are close to each other.

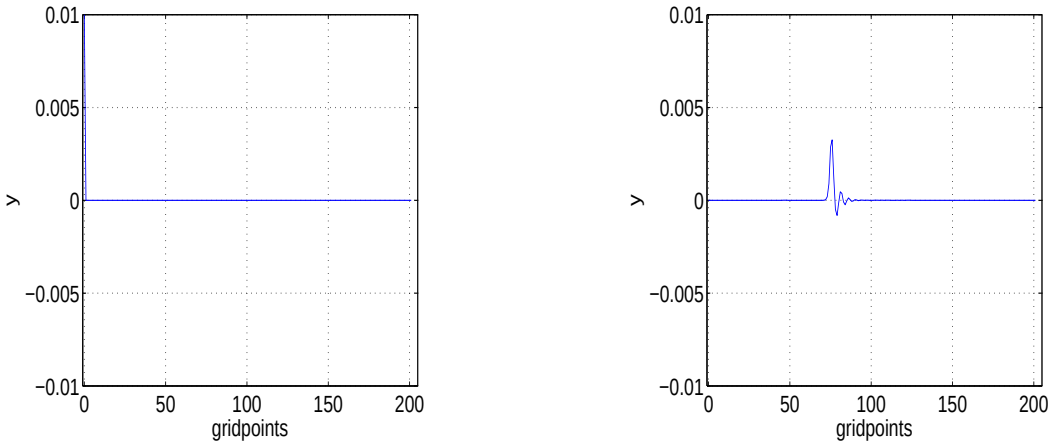
Validation of sensitivities with respect to boundary scaling factors. The cost function in this validation test takes the form of (3.13). We want to validate the sensitivities

$\frac{\partial \psi}{\partial \alpha} |_{t_n}$ and $\frac{\partial \psi}{\partial \alpha} |_{overall}$ where α is the scaling factor of the boundary conditions.

a) Validation results for $\frac{\partial \psi}{\partial \alpha} |_{t_n}$ are shown in Table 3.3. Like in the case of additive boundary perturbations, the agreement between the adjoint sensitivity and the finite difference sensitivity depends on the choice of the receptor (the choice of a,b in the definition of the cost function).

Receptor	$\Delta\psi$ Finite Difference	$\Delta\psi$ Continuous Adjoint	Relative Error
a = 0, b= 71	0.00	1.27E-6	0.00%
a = 72, b = 99	7.58E-3	7.57E-3	-5.04E-2%
a = 100, b = 202	0.00	5.73E-26	0.00%
a = 75, b = 80	5.88E-3	7.24E-3	18.82%
a = 80, b = 85	3.16E-4	9.48E-7	-3.32E4 %

Table 3.3: Sensitivity to West boundary Scaling Factor: Finite Difference vs. Adjoint.



(a)δb (left) at initial time (b) δc at final time

Figure 3.3: Evolution of the perturbation (with a scaling factor) at left boundary. (a) Perturbation at left boundary at initial time. (b) Solution difference in the receptor at final time.

b) Validation of $\frac{\partial \psi}{\partial \alpha} |_{overall}$. In this case, the boundary conditions are perturbed by scaling the boundary values at each time step. The scaling factor is the same for all time steps, and we seek to estimate the impact of this scaling to the cost function. For the base case the scaling factor is equal to 1.

The propagation of the boundary perturbation to the solution is plotted in Figure 3.4. We see that the change affects a wider area (due to the fact that the perturbation in boundary conditions is applied at all time steps). The chances of a receptor to capture the change in solution are therefore better. The results in Table 3.4 show a better agreement between the

adjoint and the finite difference sensitivities. This agreement is less sensitive to the choice of receptor than in the case of perturbations applied to individual boundary conditions.

Receptor	$\Delta\psi$ Finite Difference	$\Delta\psi$ Continuous Adjoint	Relative Error
a = 0, b= 94	7.67E-1	7.57E-1	-1.33%
a = 95 b = 202	0.00	3.27E-22	0.00%
a = 70, b = 90	6.06E-2	6.82E-2	11.07%
a = 50, b = 70	2.12E-1	2.12E-1	3.60E-3%
a = 30, b = 33	4.04E-2	94.04E-2	1.18E-3 %

Table 3.4: Overall Boundary Scaling Factor Sensitivity: Finite Difference vs. Adjoint.

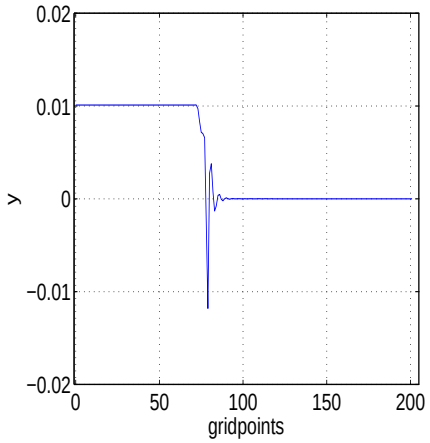


Figure 3.4: Evolution of perturbation of boundary at overall time: δc at final time

3.2 Sensitivity to Boundary Conditions and Boundary Condition Scaling Factors for CMAQ

3.2.1 Implementation

The adjoint boundary sensitivities in CMAQ have been implemented based on equations 3.6 and 3.10 discussed in the previous section. Boundary sensitivities at synchronization times are calculated in the adjoint run after the adjoint variables (LGRIDs) are updated at the current synchronization time.

3.2.2 Validation

We now show validation results for the sensitivity of a cost function with respect to western boundary condition perturbations at initial time. Both the cost function and western boundary condition are only considered for ozone. We switch off all other science processes except x advection. This test is similar to the 1D advection test. Specifically, each western boundary gridpoint BCON(ROW, LAY) is perturbed individually (acts as a single grid source). The receptor is chosen as $\psi = \sum_{col=1}^{ncols} CGRID(COL, ROW, LAY)$ the sum of the concentrations in the whole line of gridpoints having the same row and layer number as the source. Therefore, there are NROWS x NLAYS source-receptor pairs for which sensitivities are computed independently of each other. We calculate the sensitivities for each source and receptor pair using both finite difference and adjoint approaches, and we compare the results using scatter plots and relative differences.

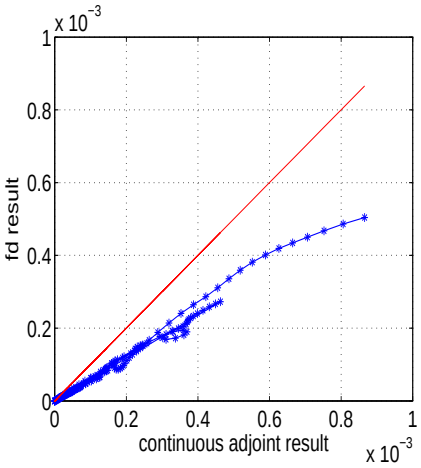
To obtain the sensitivity to boundary condition, we first perform the sensitivity run (choose the simulation mode in run.cctm as snst). The results, LGRID_B, are stored in a file named BSENST_WEST_ADJ_O3.txt.

To obtain the finite difference results, we perform the finite difference run (choose the simulation mode in run.cctm script as fd which use the driver fd_driver.F). The perturbed run uses boundary conditions changed by Delta_B; the boundary perturbation is 1% of the nominal boundary values. In this run, the difference of cost functions before and after perturbing the boundary conditions are calculated. The cost function difference is compared against the (LGRID_B, Delta_B) dot product. LGRID_B are read from BSENST_WEST_ADJ_O3.txt file.

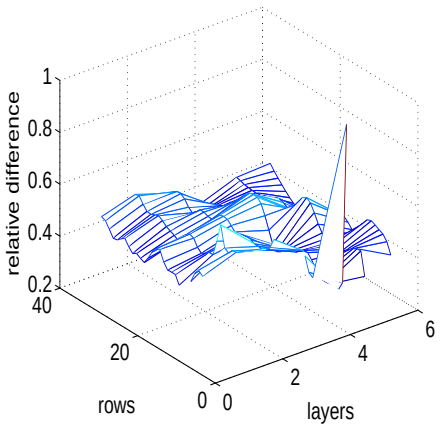
The simulation is performed for 4 hours. The perturbation for finite difference test is 1% of the original boundary concentrations. We repeat the tests using both the continuous and the discrete adjoints of x advection in the computation of adjoint boundary sensitivities. Continuous adjoint results are reported in Figure 3.5, and discrete adjoint results in Figure 3.6.

Figures 3.5(a) and 3.6 (a) show the scatter plot of the finite difference sensitivity and continuous adjoint sensitivity/discrete adjoint sensitivity respectively, for each grid cell at western boundary. Figure 3.5 (b) and 3.6 (b) shows the relative difference between finite difference sensitivities and adjoint sensitivities.

The results reported in Figures 3.5 and 3.6 , indicate that in our current implementation the sensitivity to boundary conditions is about 40% off the finite difference results. This difference seems to be systematic (points are aligned well in the scatter plot). Further work is needed to better understand these differences and to improve the results.

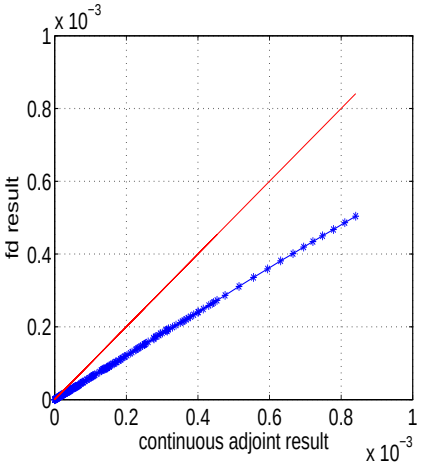


(a) scatter plot of west boundary sensitivities for continuous adjoint

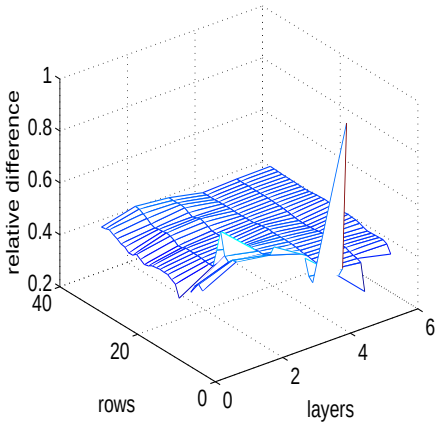


(b)relative difference for continuous adjoint

Figure 3.5: Western boundary continuous adjoint sensitivities validation. (a) Scatter plot of finite difference sensitivity (y-axis) and continuous adjoint sensitivity (x-axis). (b) Relative difference between continuous adjoint sensitivity and finite difference sensitivity.



(a) scatter plot of west boundary sensitivities for discrete adjoint



(b)relative difference for discrete adjoint

Figure 3.6: Western boundary discrete adjoint sensitivities validation. (a) Scatter plot of finite difference sensitivity and discrete adjoint sensitivity. (b) Relative difference between discrete adjoint sensitivity and finite difference sensitivity.

Chapter 4

Data Assimilation Results

4.1 Initial Conditions as Control Variables

Data assimilation is the process of fusing information from model predictions and observations in order to obtain better initial conditions, boundary conditions or emission estimates. 4D-Var data assimilation optimally combines three sources of information: a priori estimate of the state of the atmosphere; knowledge about the physical and chemical processes that govern the evolution of pollutant fields as captured in the model and observations of some of the state variables [16].

We apply 4D-Var data assimilation with CMAQ-ADJ to provide optimal estimates of the initial conditions. 4D-Var data assimilation is posed as an optimization problem where the best estimate of initial conditions minimizes the following cost function:

$$J(c^0) = \frac{1}{2}\left(c^0 - c^b\right)^T B^{-1}\left(c^0 - c^b\right) + \frac{1}{2}\sum_{k=1}^{N}\left(H_k c^k - c^{k,obs}\right)^T R_k^{-1}\left(H_k c^k - c^{k,obs}\right) \qquad (4.1)$$

where c^0 is the initial concentration, c^k is the model prediction at time step k, $c^{(k,obs)}$ is the observation at time step k, c^b is the background concentration, R_k is the observation error covariance matrix, B is the background error covariance matrix and H_k is the observation operator. The cost function measures the misfit between model predictions and observations as well as the misfit between initial conditions and background concentrations.

4.1.1 Implementation of the 4D-Var Methodology

To minimize the cost function (4.1), we apply L-BFGS, a limited memory quasi-Newton method for solving large scale optimization problems [13]. L-BFGS is an iterative method

which requires, at each iteration step, the cost function value and its gradient at that point, and returns another point which is a better approximation to the optimal solution. The whole process continues until the convergence criteria are satisfied.

In the implementation of CMAQ/4D-Var we interface L-BFGS with the CMAQ-ADJ. The cost function value and gradient value are obtained through a forward run followed by an adjoint run of CMAQ-ADJ. The general computational framework is as follows:

- Start the iteration with $x_0 = c_p^0 = c_0 + \delta c_0$
- For each iteration i = 0,..,k
  - i) Using x_i as initial condition, perform a forward run and adjoint run to get the cost function f_i and gradient of the cost function g_i at x_i.
  - ii) Feed (f_i, g_i) to L-BFGS to get the next iteration start point. L-BFGS $(f_i, g_i, x_i) \rightarrow x_{i+1}$. Stop the iteration while meet the terminating criteria e.g. convergence, max iteration number. Otherwise, continue.

4.1.2 4D-Var Results with 1D advection

To investigate how continuous and discrete adjoint gradients of advection impact the 4D-Var data assimilation, we first conduct an experiment of 4D-Var optimization with only the one-dimensional advection process.

4D-Var with 1D advection is used in formulating the optimization problem (4.1). The 4D-Var application is the same as in CMAQ-ADJ but the constraint is changed to the 1D advection PDE (2.1).

Four different test cases are designed in our study of 1D advection 4D-Var experiment. The detailed settings are as follows:

1. Reference initial condition. We choose two reference initial conditions to be retrieved through our adjoint gradient based optimization approach. Two reference initial conditions differ in their smoothness properties: one is smooth and the other is non-smooth.

   Case 1: $c_{ref}^0 = 1 + \sin(3x)$

   Case 2: $c_{ref}^0 = 1 + \text{signum}(1.0, \sin(3x))$

2. Observations. Two kinds of observation points are used. One is chosen as 21 evenly spaced grids out of 200 total computation grids ranging from grid number 1 to grid

number 200. Another set of observation locations are the whole 200 computation grids. Starting from reference initial condition, one forward advection PDE run is performed and the observations are saved for later use.

3. Observation error covariance matrix and background error covariance matrix. Both covariance matrices are diagonal. Background error covariance matrix is chosen as identity matrix. The diagonal entries of observation covariance matrix are constant with value 0.1.

4. Initial perturbation. The initial guess, i.e., the starting point for the iteration is $c_p^0 = 1.0$.

5. Simulation parameters are set as follows: maximum number of model runs: 150; science process: advection only.

Results. Four optimization experiments are carried out with two different reference initial conditions and two different observation grids. The corresponding results are analyzed in three aspects. First, the optimized solutions are compared against reference solution as well as the initial guess. Secondly, we show the decrease of the cost function as the model runs and also the value of the cost function at each iterate of the optimization procedure. Thirdly, root mean square (RMS) errors with the number of model runs and iterates are presented respectively. Figures (4.1-4.6) illustrate results of our experiments in these three aspects for both continuous adjoint approach and discrete adjoint approach.

Figure 4.1 and figure 4.2 present solutions for the case with a smooth reference solution. Figure 4.1 corresponds to the case where we only have 21 observation points out of 200 computation grids. However, figure 4.2 shows the results for the case that 200 observation points are used. In the first case, both continuous adjoint approach and discrete adjoint approach give us solutions that are in good agreement with reference solution in general. This can be seen in figure 4.1(a). However, continuous adjoint approach outperforms the discrete adjoint approach for the better match with reference solution as shown in Figure 4.1(b), which illustrates the non-smoothness of the solution given by discrete adjoint approach compared against the smooth reference solution. Figure 4.1(c) and figure 4.1(e) show that continuous adjoint results in faster convergence to the reference solution than discrete adjoint. In addition, we find that it takes fewer model runs for continuous adjoint to find more iterates than discrete adjoint. For example, discrete adjoint approach uses 139 model runs to get 55 iterates. However, for continuous adjoint approach, 75 iterates are generated by 79 model runs. This suggests that, for the discrete adjoint, LBFGS optimization does more line search to find an iterate leading to large enough decrease in cost function. Therefore, it is clear that gradient obtained by continuous adjoint provides a more suitable descent direction.

In the second case where 200 observation points are used, both retrieved initial conditions using continuous adjoint approach and discrete adjoint approach agree well with the reference initial condition, indicated by 4.2(a) and (b). In addition, different from the first case, both

continuous and discrete adjoint approach have similar convergence speed to the reference solution. An explanation about the difference between the first case and the second case is that the gradient of the second case has less discontinuity than that of the first case since adjoint forcing is continuous in the second case. As a result, the gradients obtained by two approaches are similar to each other. Consequently, the performance of 4D-Var optimization using the two approaches is similar.

For the third and fourth test cases, the reference solutions are not smooth. In addition, the third test case use 21 observation points while 200 observations points are used in the fourth test case. In both cases, the initial conditions retrieved by continuous adjoint approach match better with reference solutions, which can be seen in figure 4.3(a)-(c) and figure 4.5(a)-(c). Specifically, solution obtained by discrete adjoint approach oscillates while approaching disjoint point in the case where we have fewer observation grids. Similarly, the solutions using continuous adjoint approach at the disjoint points are closer to the reference solutions when we have 200 observation points. In addition, figure 4.4(a)-(d) and figure 4.6(a)-(d) illustrate that discrete adjoint approach leads to a slower convergence of optimization for both test cases.

To sum up, L-BFGS optimization gives more accurate retrieved initial condition using gradients computed by continuous adjoint of PPM advection. Results with discrete adjoint fail to accurately retrieve the solution especially at the positions where discontinuity exist or slope change sign. Also, the faster convergence of optimization for continuous adjoint case indicates that continuous adjoint provides the better decent direction.

4.1.3 CMAQ 4D-Var Results

To validate the implementation of 4D-Var data assimilation in CMAQ-ADJ, we conduct a test with the following settings:

1. Synthetic observations. The computation domain for our simulation has 38 columns, 38 rows and 6 layers in total. We choose $10 \times 10 \times 3$ grid cells as observation grids and choose O_3 as the species that we want to retrieve. A reference forward run is performed, starting from the reference initial condition. During this run the ozone concentrations at all the observation grid points are saved. They are the synthetic observation values.

Columns	Rows	Layers	Species
1,5,9,13,17,22,26,30,34,38	1,5,9,13,17,22,26,30,34,38	1,3,5	4(O_3)

Table 4.1: Observation grid consists of a subset of the model grid points

2. Observation error covariance matrix and background error covariance matrix. Both covariance matrices are diagonal. The entries are the error variances at the corre-

sponding grid points. These variances correspond to standard deviations equal to 10% of the reference concentration values for the background errors, and 1% of the reference concentration values for the observation errors.

3. Initial perturbation. The best guess, i.e., the starting point for the iteration is $c_p^0 = c_0(1.3 + 9 * eps)$, $eps = 0.001$.

4. Simulation parameters are set as follows:
   Assimilation time window: 12 hrs: July 2, 1999, 00:00:00 - July 2, 1999, 12:00:00
   Maximum number of model runs: 26
   Science Processes: advection, diffusion, chemistry.

Three different optimization experiments are carried out. The first uses gradients computed via a continuous advection adjoint, the second and the third use the gradients computed via the discrete advection adjoint and modified discrete advection adjoint. In all cases (the same) discrete adjoints are used for the diffusion and chemistry processes.

Figure 4.7(a) presents the difference between the best guess (perturbed initial condition) and reference initial condition before data assimilation. The initial field after data assimilation is compared against the reference initial field; if data assimilation process converges then the initial field after data assimilation should approximate well the reference initial condition. This is indeed the case with the solution obtained using a continuous advection adjoint, presented in Figure 4.7(b). However, the optimized initial field using the discrete advection adjoint shows large differences from the reference. This can be seen in Figure 4.7(c). Optimal of initial field given by modified discrete adjoint approach has even larger differences from the reference, which is illustrated in Figure 4.7(d).

Figure 4.8 presents the evolution of the optimization process. Specifically, Figure 4.8 (a) shows the decrease of the cost function with the number of iterations. Figure 4.8(b) shows the decrease of the root mean square error (difference between the initial conditions found by optimization and the reference solution) with the number of iterations. For both metrics the optimization converges faster when continuous advection adjoints are used. The solution obtained with discrete advection adjoints shows a much slower convergence and it actually converges to a local minimal instead of the global minimal.

4.1.4 CMAQ 4D-Var Using Discrete Advection Adjoints with Artificial Diffusion

Our previous experiments show that optimization using gradients via continuous adjoint converge fast to reference solution. However, optimization failed to progress towards the

reference solution while using the gradients via discrete adjoint. Figure 2.1 shows discrete adjoint introduces wiggles to the sensitivities(gradients) where discontinuity presents. To investigate how smoothness of the gradient affects optimization procedure, we add artificial diffusion to the adjoint fields after adjoint advection process to smooth out the gradient. The smoothed discrete adjoint gradients are then used in the optimization process.

Specifically, the artificial diffusion takes the form of:

$$\frac{\partial \lambda}{\partial t} = K(\frac{\partial^2 \lambda}{\partial x^2} + \frac{\partial^2 \lambda}{\partial y^2} + \frac{\partial^2 \lambda}{\partial z^2}) \tag{4.2}$$

where $\lambda(x, y, z, t)$ is the adjoint and K is constant.

Figure 4.9 illustrates the convergence of L-BFGS optimization with different levels of artificial diffusions applied to discrete adjoint of advection sensitivities. Firstly, we note that even if artificial diffusion is utilized for discrete adjoint gradients, optimization doesn't converge as fast as the case of continuous adjoint. In addition, optimized initial fields with artificial diffusion approach still have relatively large difference from the reference initial fields, as can be seen in figure 4.9(c). Moreover, there is a tradeoff between the amount of artificial diffusion added and the accuracy of gradients. As we can see from table 4.2, while we increase the amount of artificial diffusion, the gradients become more and more inaccurate although the gradients become more and more smooth. Specifically, if the artificial diffusion is small, the corresponding change in gradients is small too. However, this small change would not be enough to smooth out the wiggles in the gradients. Therefore, convergence acceleration is not obtained. This is indeed the case with K = 100 as shown in Figure 4.9(b)(d). If the amount of artificial diffusion is decent, the changed gradients would be both relatively more accurate and smooth. In this case, faster decrease of the RMS values can be obtained. This means the the retrieved solution gets closer to the reference solution. K = 1E3 and K = 1E4 are just such cases in our experiment as shown in fig. 4.9 (c)(d). Similarly, worse convergence is obtained if the artificial diffusion is too large to make the gradients accurate enough. This is illustrated by the case with K = 1E5 as shown in figure 4.9. Note that, in the case of K = 1E3 and K = 1E4, faster decrease of cost function value is not achieved due to different convergence series after the artificial diffusion diffusion is applied. However, new solutions converge to local minimals closer to the reference.

4.1.5 CMAQ 4D-Var with Real Observation Data

In this section, real observation data (AIRNOW) is used to further investigate the difference between continuous adjoint approach and discrete adjoint approach in the 4D-Var application.

In this numerical experiment, the assimilation period is 18:00 to 24:00 (UTC) on Sept. 1st, 2006. The AIRNOW observation data is used over Texas. To show the effect of assimilation, we use both scatter plots and time series plots to compare the ozone concentration values

Artificial Diffusion	ΔCost Function via Finite Difference	Δ Cost Function via Artif. Diffusion	Relative Error
K = 0	290.08	291.56	0.51 %
K = 1E2	290.08	291.32	0.43 %
K = 1E3	290.08	284.34	-2.02 %
K = 1E4	290.08	219.95	-31.88 %
K = 1E5	290.08	56.07	-417.33 %

Table 4.2: Finite difference validation of discrete adjoint gradients after using artificial diffusion

of observation, background and model analysis (for both continuous adjoint approach and discrete adjoint approach). Model analysis is the model prediction using optimized initial condition for the assimilation period.

For each (col, row, layer, hrs), the space and time vector, there are corresponding observation, background and analysis values. We show the scatter plots for every two of them to see how good the analysis is. For each observation location, we plot the ozone concentration values of observation, background and analysis as time evolves to see if the analysis is really an improvement compared to background. Six specific locations are chosen and results for these locations are reported. In addition, we plot the model analysis obtained using continuous adjoint approach and discrete adjoint approach to see their difference in the experiment.

From figure 4.10, we see that both continuous adjoint analysis and discrete adjoint analysis are more linearly correlated to the observations than the background, which indicates an improvement in the initial condition estimation. In addition, the R^2 correlation coefficient between analysis and observation are similar for both continuous adjoint approach and discrete adjoint approach. Figure 4.12 shows that the model analysis after data assimilation gets closer to the observations than the background and that the continuous adjoint analysis and the discrete adjoint analysis are quite similar to each other.

4.1.6 CMAQ 4D-Var for the Period Aug. 30, 2006 to Sept. 01, 2006

In this section, we continue to investigate the performance of 4D-Var with real observation data set in CMAQ-ADJ. Instead of comparing the performance of different adjoint models, only continuous adjoint approach is considered here. In addition, the assimilation period is 3 days in this new experiments.

Settings

1. Grid description and input files. The grid name is uh_dw04_01 with 82 x 64 grid points. Input data used corresponds to this grid definition.(Bcon: Bcon_dw04km; Icon: Cmaq_cb4)

2. Background error covariance matrix and Observation error covariance matrix. Both matrices are diagonal. The BG_WGT variable in GET_BG_FG subroutine is set to 1.0 and the UNC_OBS variable in GET_OBS_FG subroutine is set to 0.08.

3. Observation data. We were using original Airnow observation data for the 82 x 64 domain.

4. Assimilation period and assimilation windows. Assimilation period is from Aug. 30, 2006, 4am to Sept. 01, 2006, midnight. Note that the start time at the first day is 4am since the observation data for that day starts from 4am. All times are UTC. The typical assimilation window used in this study is 6 hours. The data assimilation runs are initialized with the "analysis" from the previous 6-hour run. Specifically, the initial condition (guess) for current 6-hour run is from the analysis for the previous 6-hour run. We perform cyclic 4D-Var data assimilation for the following windows:

   (a) 2006242:040000 2006242:100000;
   (b) 2006242:100000 2006242:160000;
   (c) 2006242:160000 2006242:240000 (8hrs);
   (d) 2006243:000000 2006243:060000;
   (e) 2006243:060000 2006243:120000;
   (f) 2006243:120000 2006243:180000;
   (g) 2006243:180000 2006243:240000;
   (h) 2006244:000000 2006244:060000;
   (i) 2006244:060000 2006244:120000;
   (j) 2006244:120000 2006244:180000;
   (k) 2006244:180000 2006244:240000;

5. Control variables. This experiment is for O3 initial conditions (at the beginning of each subinterval).

6. Optimization settings. The maximum number of model runs (forward and backward runs) is set to 15.

Results

We use both scatter plots and time series plots to compare the ozone concentrations from observations, background model run, and model analysis. Model analysis is the model prediction using optimized initial condition for each assimilation window.

For each space-time point (column, row, layer, hour) where an observation is available we read the corresponding model background and analysis predictions. Scatter plots for pairs of observation-background values and observation-analysis values are constructed. They illustrate the improvements in the model-observation agreement after the assimilation, and therefore illustrate the quality of the analysis. For each case we also compute the R-square mismatch factor between analysis and observations.

For each observation location, we plot the concentration values of observation, background and analysis as time evolves to see if the analysis is really an improvement compared to background. Six specific locations are chosen and results for these locations are reported.

The background value are obtained by running the forward model 3 days using the model input at the beginning of the first day as the initial condition. The results are presented next.

The scatter plots are shown in Figure 4.13. The model-observation agreement, as quantified by the R-square factor, increases from the background value of 0.54 to the analysis value of 0.91.

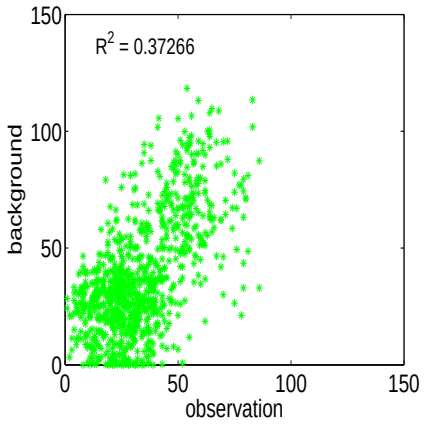
Time series of observations, background model prediction, and analysis model predictions at selected stations during the 3 day are shown in Figure 4.14. For each station the analysis is consistently closer to the observations. Recall that each analysis trajectory is obtained from four 6-hour assimilation windows, and the analysis profiles may not be smooth at the beginning of a new assimilation window.

4.1.7 Comparison of Assimilation Results against CAMS Data

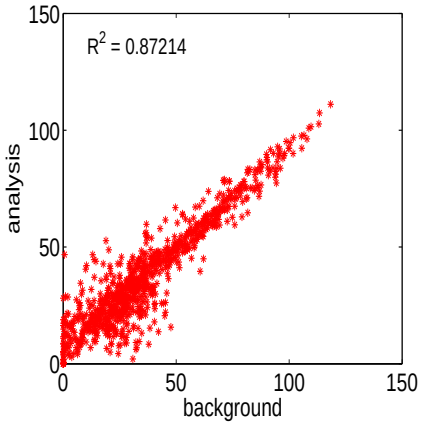
The CAMS observations were not used directly in data assimilation. After assimilation of AirNow data the agreement of model predictions and CAMS data is improved. Note that the CAMS and the AirNow data are not fully independent as many of the locations overlap. Nevertheless, the test provides a measure of the beneficial effects of assimilation.

(1) First day: from Aug. 30, 2006, 4am to Aug. 31, 2006, 0am.

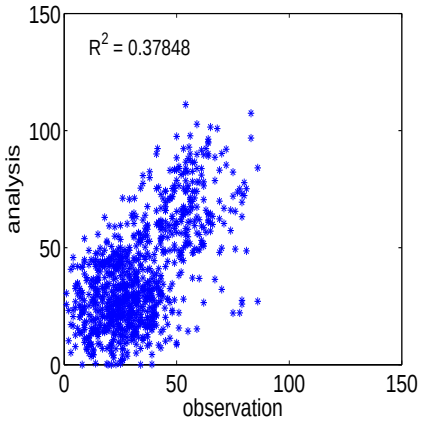
The scatter plots for the first day are shown in Figure 4.15. The agreement between model predictions and CAMS observations, as quantified by the R-square factor, remains about the same with a background value of 0.373 and an analysis value of 0.378.



(a) observation vs. background



(b) analysis vs. background

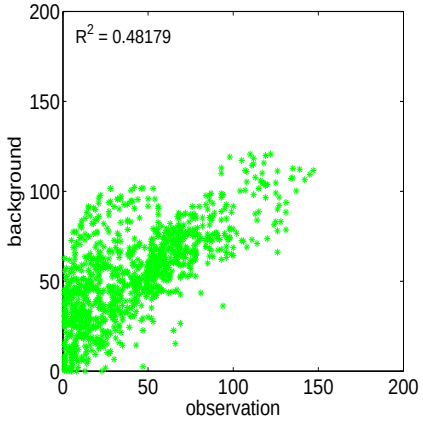


(c) analysis vs. observation

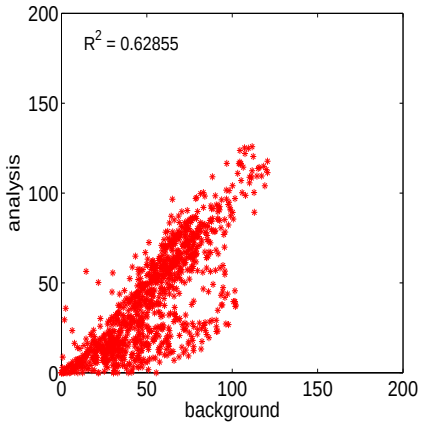
Figure 4.15: Scatter plots of CAMS observations and the corresponding model predictions for Aug. 30, 2006.

(2) Second day: from Aug. 31, 2006, 0am to Sept. 1, 2006, 0am.

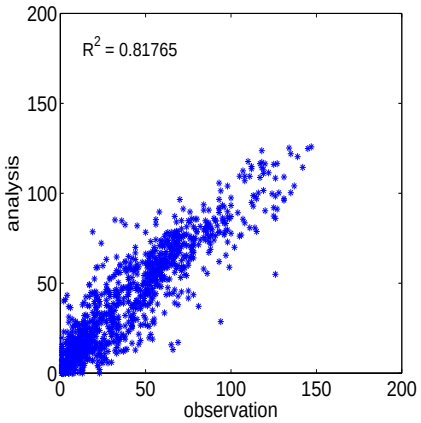
The scatter plots for the second day are shown in Figure 4.16. The agreement between model predictions and CAMS observations, as quantified by the R-square factor, increases from the background value of 0.48 to the analysis value of 0.82.



(a) observation vs. background



(b) analysis vs. background

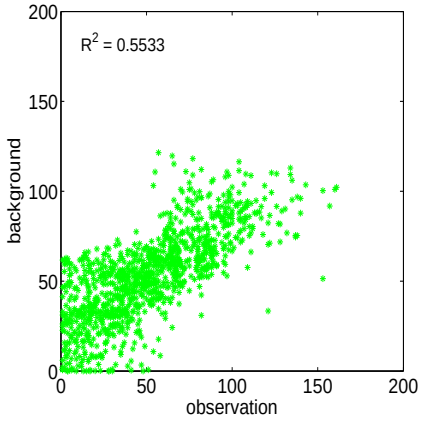


(c) analysis vs. observation

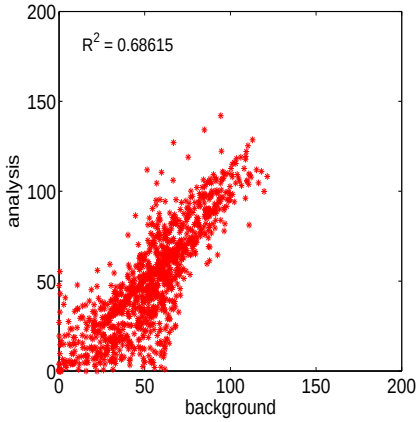
Figure 4.16: Scatter plots of CAMS observations and the corresponding model predictions for Aug. 31, 2006.

(3) Third day: from Sept. 1, 2006, 0 am to Sept. 2, 2006, 0am.

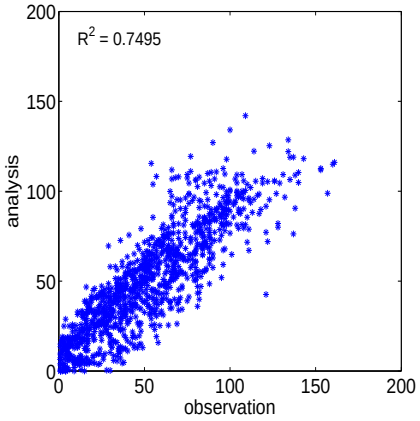
The scatter plots for the third day are shown in Figure 4.17. The agreement between model predictions and CAMS observations, as quantified by the R-square factor, increases from the background value of 0.55 to the analysis value of 0.75.



(a) observation vs. background



(b) analysis vs. background



(c) analysis vs. observation

Figure 4.17: Scatter plots of CAMS observations and the corresponding model predictions for Sept. 1, 2006.

4.2 Boundary Condition Scaling Factors as Control Variables

In this section, we discuss the 4D-Var analysis in CMAQ with boundary condition scaling factors as control variables.

4.2.1 Implementation

Implementation of 4D-Var with boundary condition scaling factor is similar to that of 4D-Var with initial condition. Specifically, instead of minimizing the misfit between initial condition

and background, we want to minimize the boundary scaling factor and the best guess of the scaling factor (the prior scaling factor). The optimization problem is formulated as follows:

$$J(\alpha) = \frac{1}{2}(\alpha - \alpha^b)^T B_\alpha^{-1}(\alpha - \alpha^b) + \frac{1}{2}\sum_{k=1}^{N}(H_k c^k - c^{k,obs})^T R_k^{-1}(H_k c^k - c^{k,obs}) \tag{4.3}$$

where α is the vector of boundary condition scaling factor with size of the dimension of the boundary conditions, , α^b is the prior scaling factor vector, c^k is the model prediction at time step k, $c^{(k,obs)}$ is the observation at time step k, R_k is the observation error covariance matrix, B_α is the background error covariance matrix and H_k is the observation operator. The cost function measures the misfit between model predictions and observations as well as the misfit between boundary scaling factor and prior boundary scaling factor.

4.2.2 Results

To validate the implementation of 4D-Var with boundary scaling factor, we conduct an experiment where we want to further improve the model prediction based on the optimized initial condition after 4D-Var using initial condition as control variable.

Settings:

1. Grid description and input files. The grid name is uh_dw04_01 with 82 x 64 grid points. Input data used corresponds to this grid definition.(Bcon: Bcon_dw04km; Icon: Cmaq_cb4)

2. Background error covariance matrix and Observation error covariance matrix. Both matrices are diagonal. The BG_WGT variable in GET_BG_FG subroutine is set to 1.0 and the UNC_OBS variable in GET_OBS_FG subroutine is set to 0.08.

3. Observation data and background. We were using the Airnow observation data for the 82 x 64 domain. The background is the optimized initial concentration after 4DVar using the initial condition as control variable.

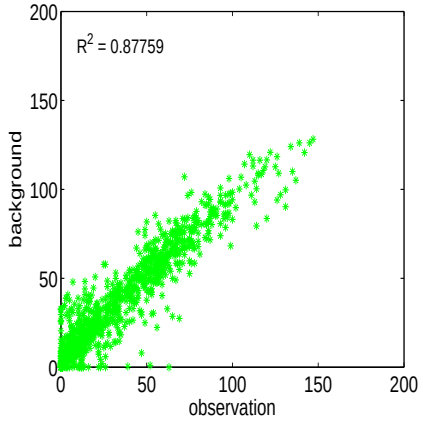
4. Assimilation period and assimilation window. Assimilation period is from 08/31/2006, 0am to 09/01/2006, midnight . All times are UTC. The assimilation window is 48 hrs. Note that the reasons for choosing this period and assimilation window is that we want the boundary conditions take more impact on the cost function so a long assimilation window would be helpful.

5. CMAQ-ADJ codes: Parallel version of CMAQ-ADJ is used for this experiment.

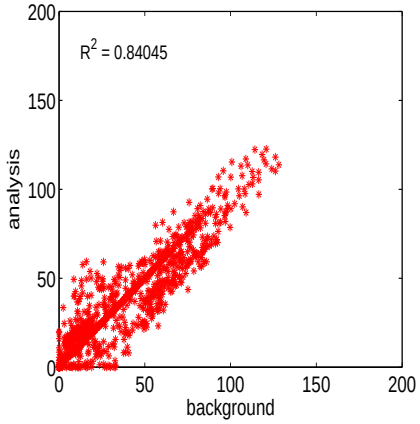
6. Control variable. This experiment is for 4DVar with boundary scaling factor as control variable.

7. Optimization settings. Max number of model runs (forward and backward runs) is set to 15.

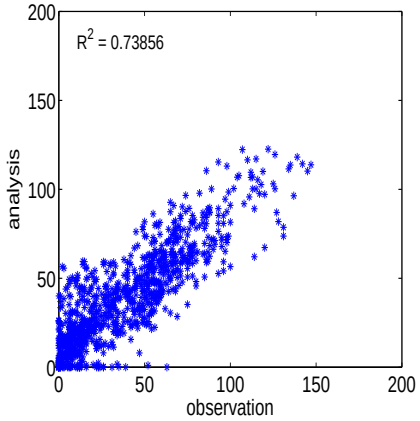
Results.
(a) Aug. 31, 2006: Figure 4.18 shows the agreement between model predictions and observations, as quantified by the R-square factor, doesn't increase from the agreement between background and observation. In addition, figure 4.19 indicates that the model predictions after assimilation fail to get closer to the observation than the background.



(a) observation vs. background

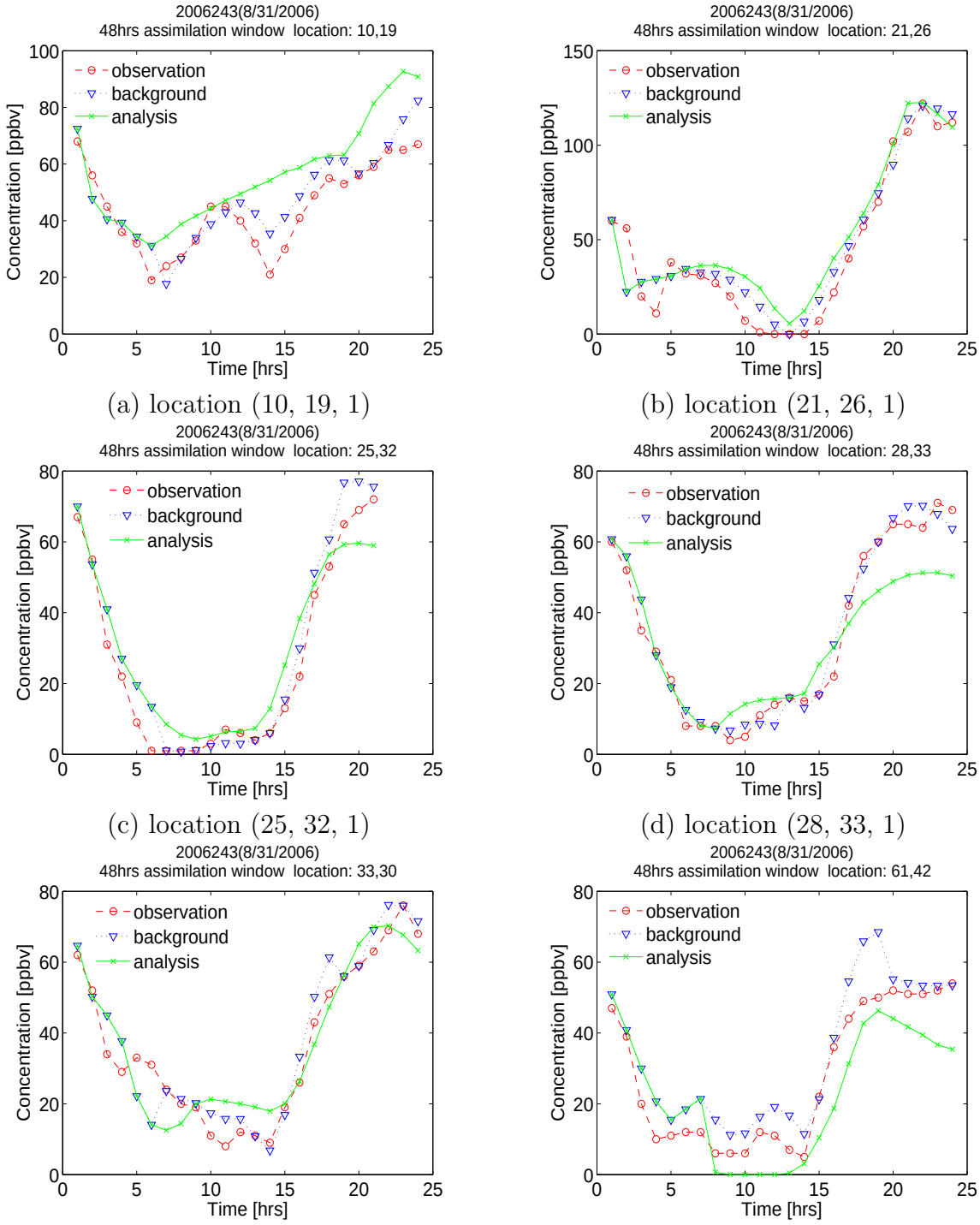


(b) analysis vs. background



(c) analysis vs. observation

Figure 4.18: Scatter plots for Boundary Scaling factor 4D-Var on Aug. 31, 2006



(a) location (10, 19, 1)

(b) location (21, 26, 1)

(c) location (25, 32, 1)

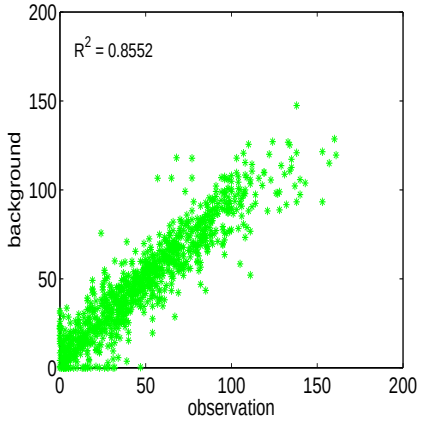
(d) location (28, 33, 1)

(e) location (33, 30, 1)

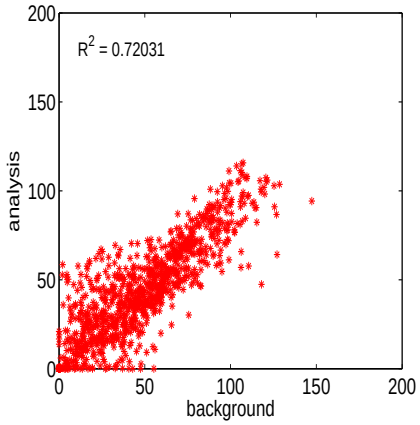
(f) location (61, 42, 1)

Figure 4.19: Time series of observations, background model prediction, and analysis model predictions at selected stations on Aug. 31, 2006. CMAQ with 4D-Var assimilation for boundary scaling factors.

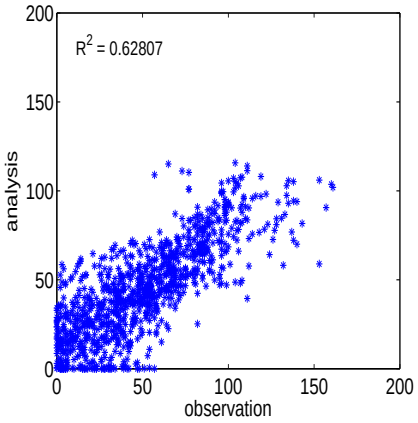
(b) Sept., 1, 2006. For this date, figure 4.20 and figure 4.21 also indicates that the model predictions after assimilation are not an improvement over the background.



(a) observation vs. background

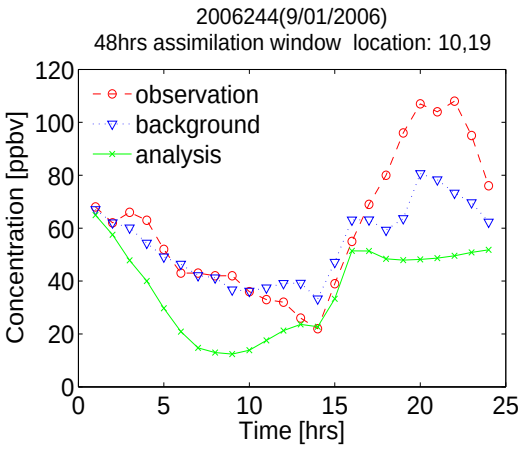


(b) analysis vs. background

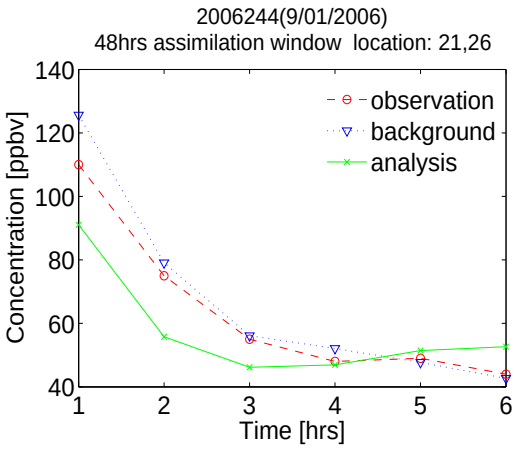


(c) analysis vs. observation

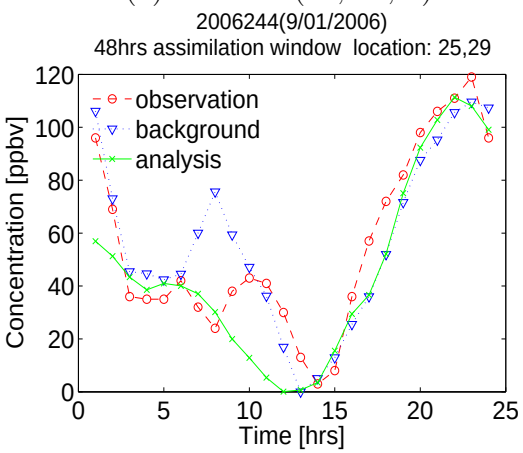
Figure 4.20: Scatter plots for Boundary Scaling factor 4D-Var on 9/1/2006



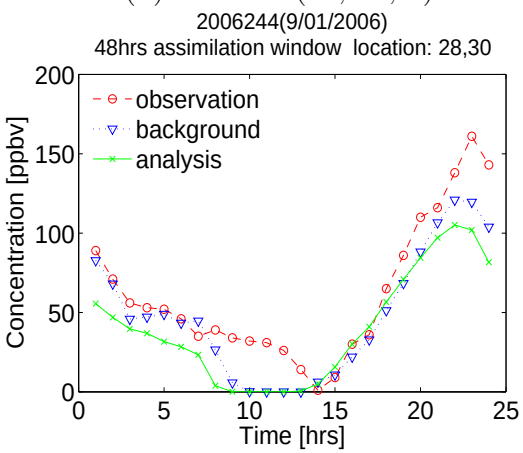
(a) location (10, 19, 1)



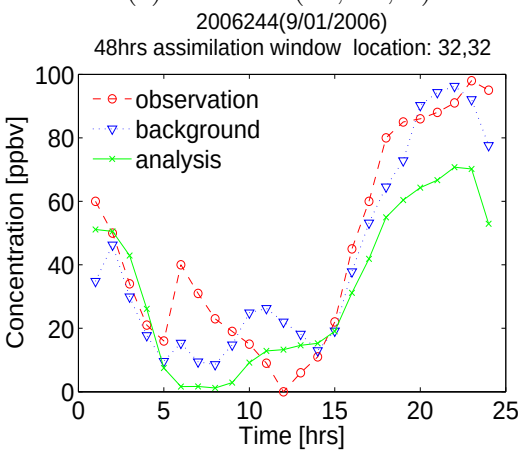
(b) location (21, 26, 1)



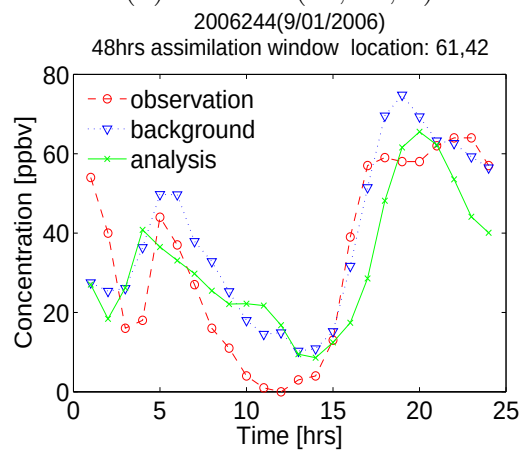
(c) location (25, 29, 1)



(d) location (28, 30, 1)



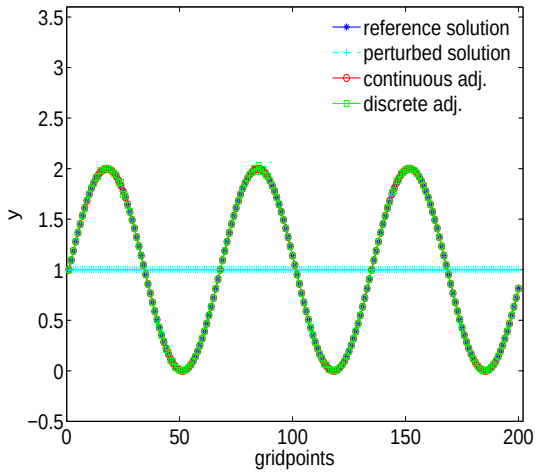
(e) location (32, 32, 1)



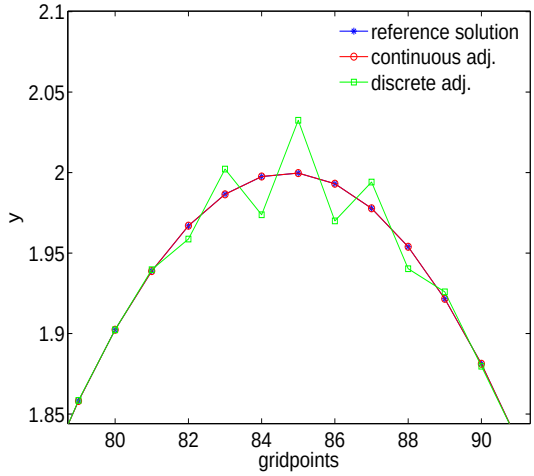
(f) location (61, 42, 1)

Figure 4.21: Time series of observations, background model prediction, and analysis model predictions at selected stations on Sept. 1, 2006.

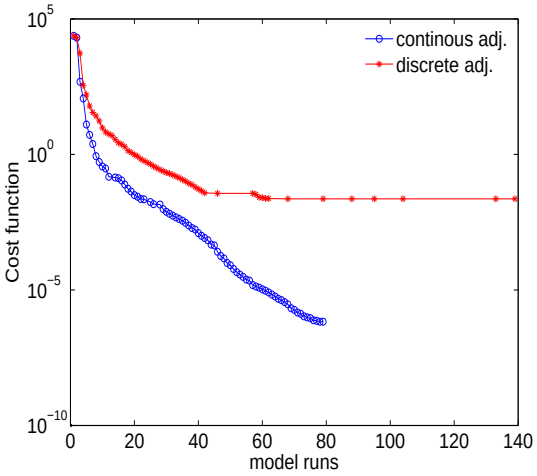
The results are presented in figures 4.18 - 4.21 which indicate that the new model prediction after this 4DVar experiment doesn't get closer to the observations than the background. The reason needs to be investigated in the future. One possible reason is that the scaling factors are considered as constant at this time so scaling factors as a function of time may need to be used in the future.



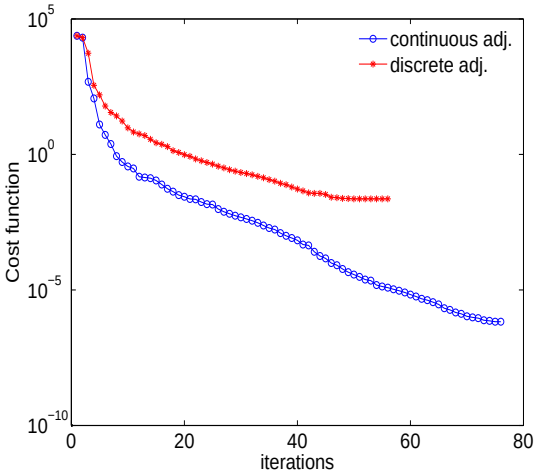
(a) reference, perturbed and retrieved initial condition



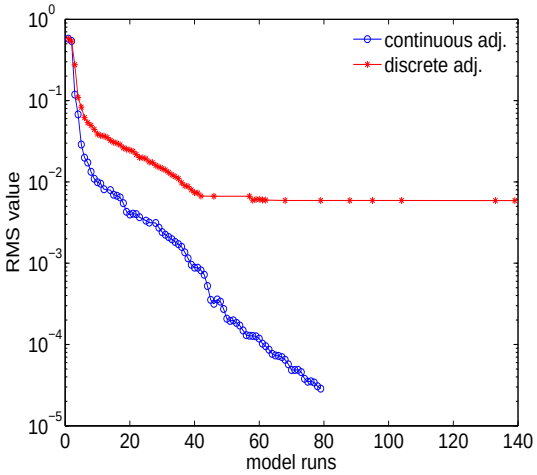
(b) reference, perturbed and retrieved initial condition from grids 80 to 90



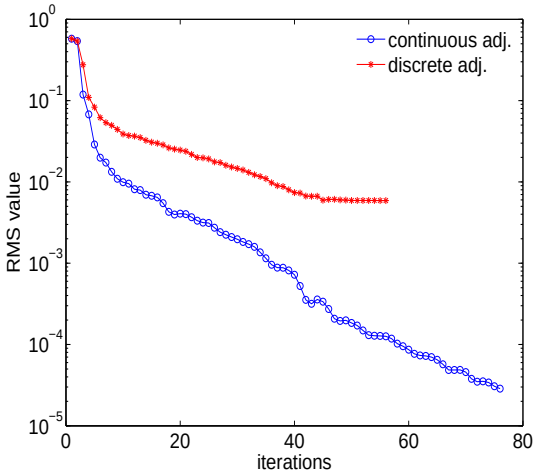
(c) cost function vs. model runs



(d) cost function vs. iterate

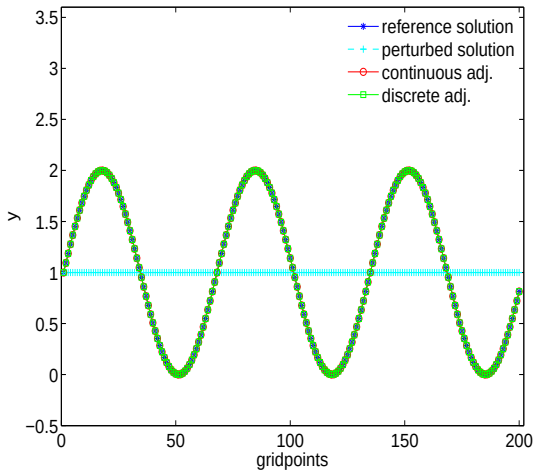


(e) RMS vs. model runs

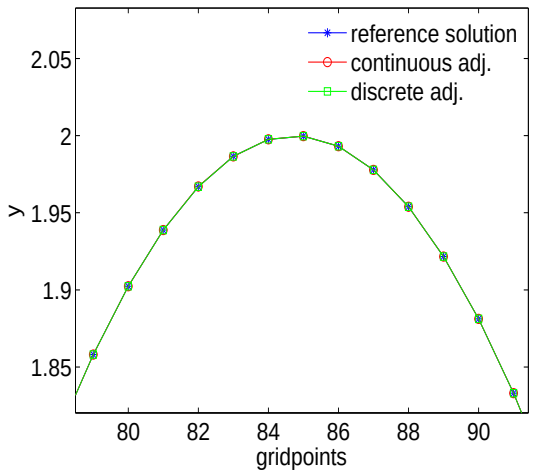


(f) RMS vs. iteration

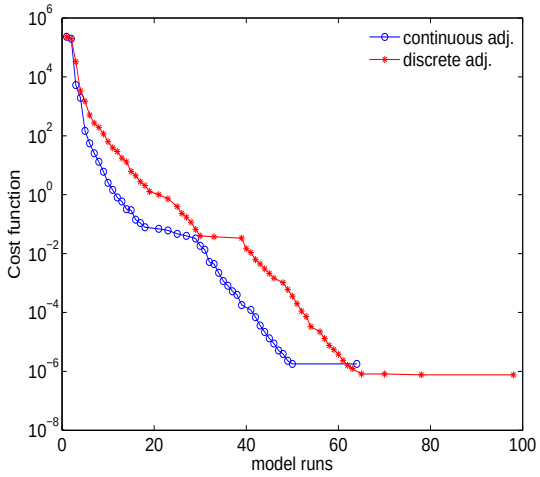
Figure 4.1: Solution and convergence of 4D-Var optimization 1D test for case where reference: $c^0_{ref} = 1 + \sin(3x)$, initial guess: $c^0_P = 1.0$ and 21 observation points. (a) plots of reference, perturbed and retrieved solution; (b) reference, perturbed and retrieved solution from grids 80 to 90; (c) cost function value as a function of model runs; (d) cost function value at each iterate; (e) RMS value as a function of model runs; (f) RMS value at each iterate.



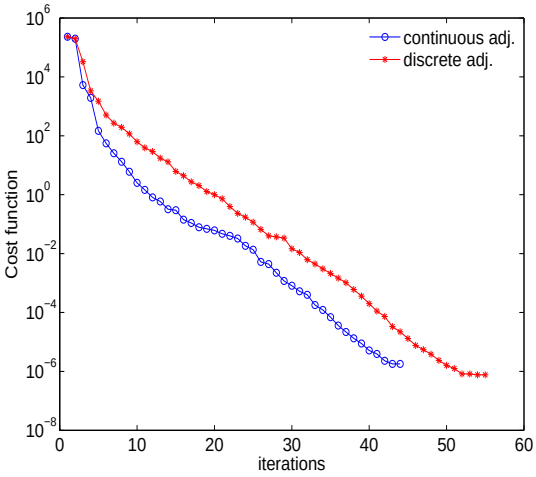
(a) reference, perturbed and retrieved initial condition



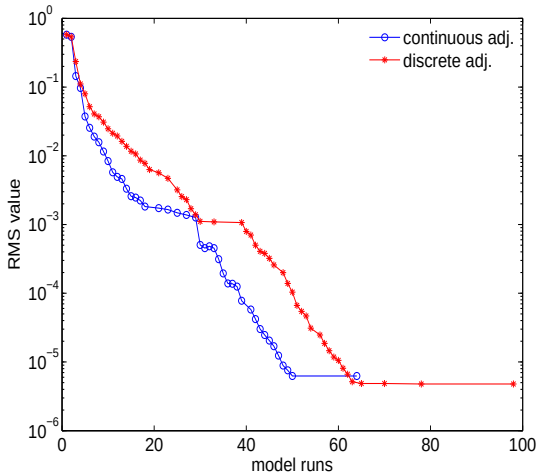
(b) reference, perturbed and retrieved initial condition from grids 80 to 90



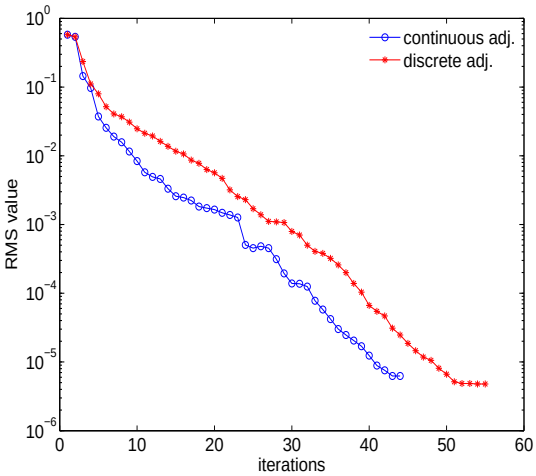
((c) cost function vs. model runs



(d) cost function vs. iteration

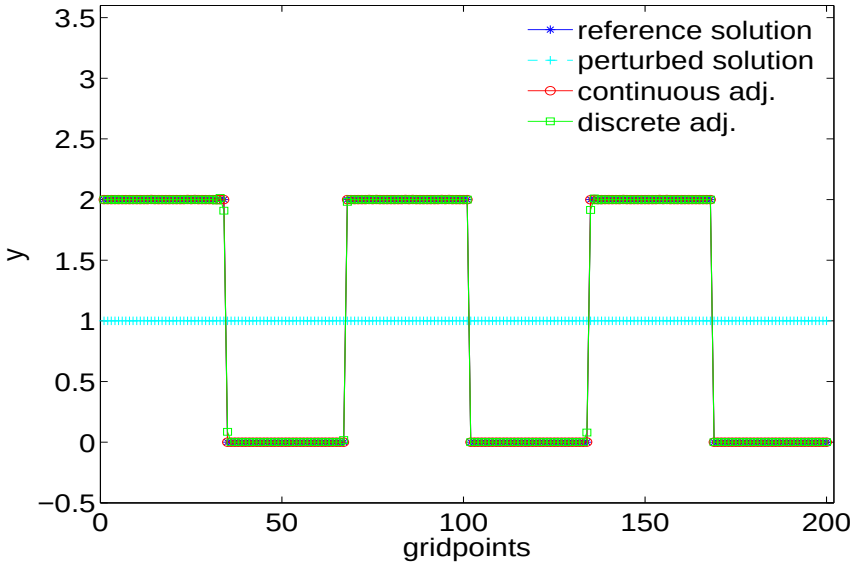


(e) RMS vs. model runs

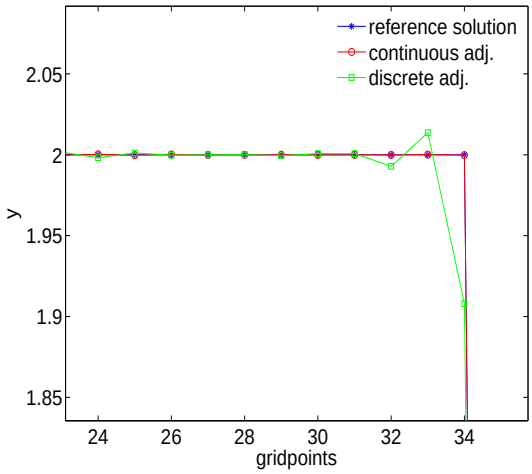


(f) RMS vs. iteration

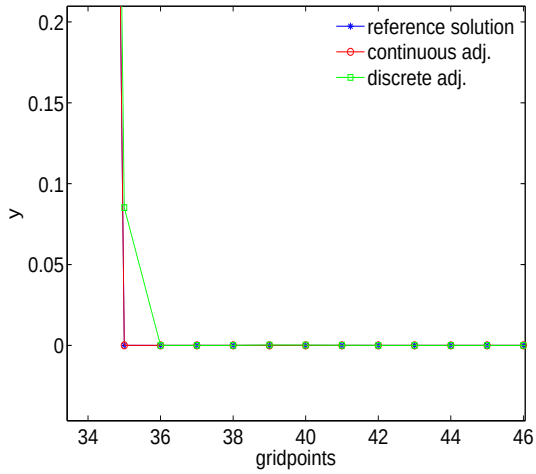
Figure 4.2: Solution and convergence of 4D-Var optimization 1D test for case where reference: $c^0_{ref} = 1 + \sin(3x)$, initial guess: $c^0_P = 1.0$ and 200 observation points. (a) plots of reference, perturbed and retrieved solution; (b) reference, perturbed and retrieved solution from grids 80 to 90; (c) cost function value as a function of model runs; (d) cost function value at each iteration; (e) RMS value as a function of model runs; (f) RMS value at each iteration.



(a) reference, perturbed and retrieved initial condition

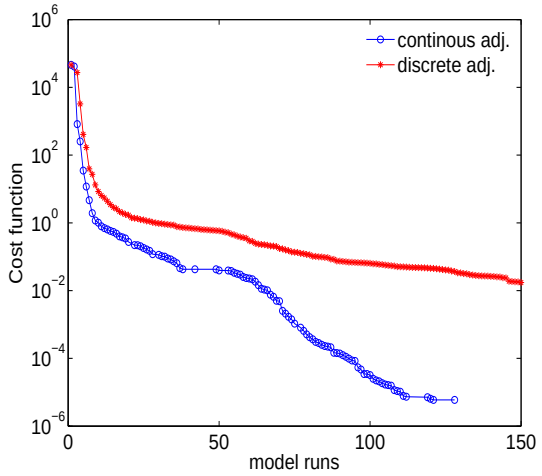


(b) reference and retrieved initial condition (grids 24 to 34)

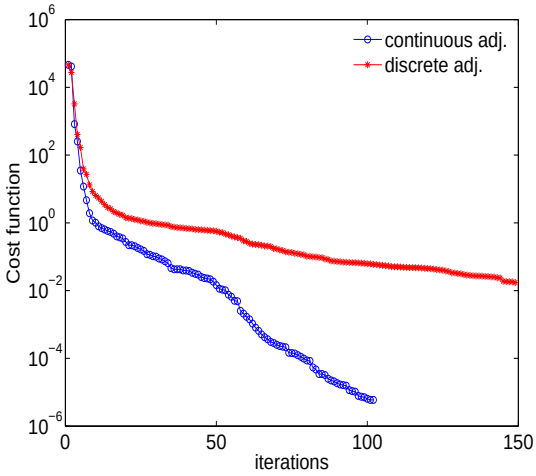


(c) reference and retrieved initial condition (grids 35 to 45)

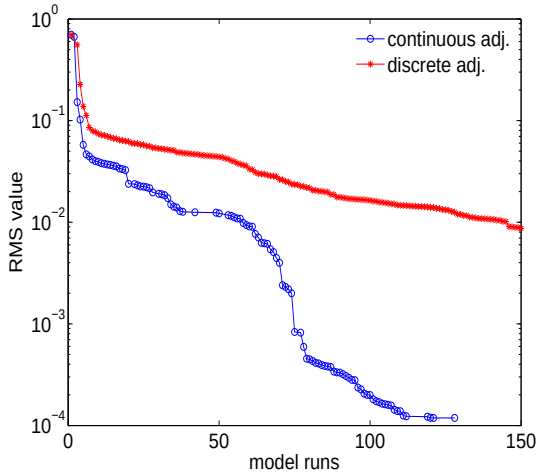
Figure 4.3: Solution of 4D-Var optimization 1D test for case where reference: $c^0_{ref} = 1 + signum(1.0, \sin(3x))$, initial guess: $c^0_P = 1.0$ and 21 observation points. (a) reference, perturbed and retrieved initial condition. (b) reference and retrieved initial condition grids 24 to 34. (c) reference and retrieved initial condition grids 35 to 45.



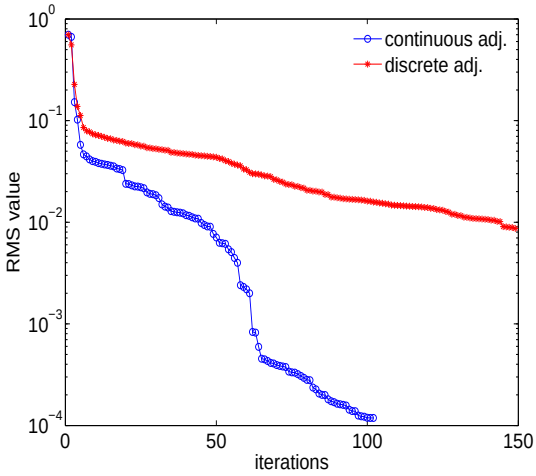
(a) cost function vs. model runs



(b) cost function vs. iteration

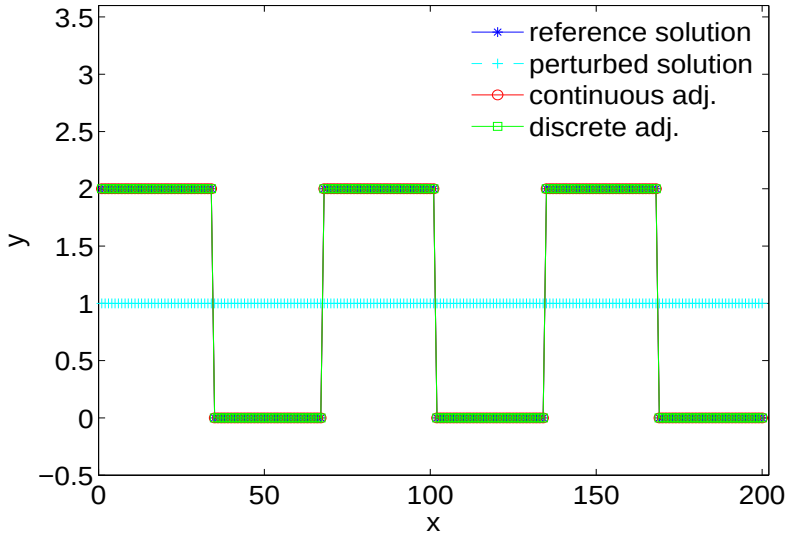


(c) rms vs. model runs

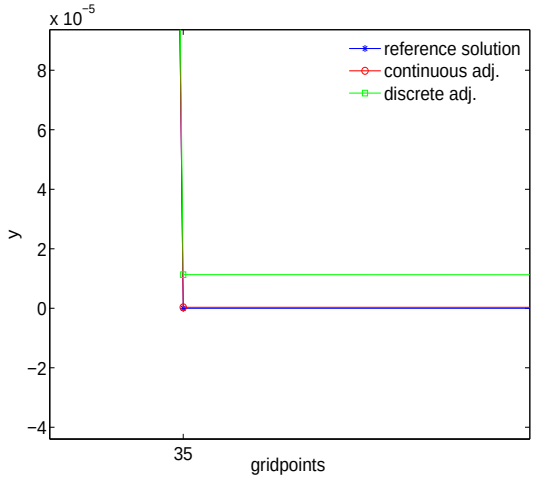


(d) rms vs. iterations

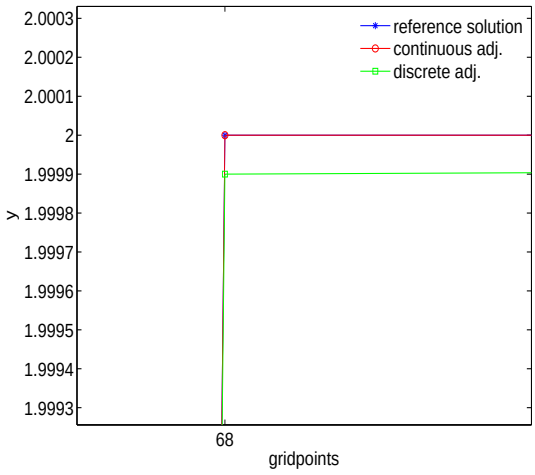
Figure 4.4: L-BFGS convergence of 4D-Var optimization 1D test for case where reference: $c^0_{ref} = 1 + signum(1.0, \sin(3x))$, initial guess: $c^0_P = 1.0$ and 21 observation points. (a) cost function value as a function of model runs. (b) cost function value as a function of iteration. (c) rms value as a function of model runs. (d) rms value as a function of iteration.



(a) reference, perturbed and retrieved initial condition

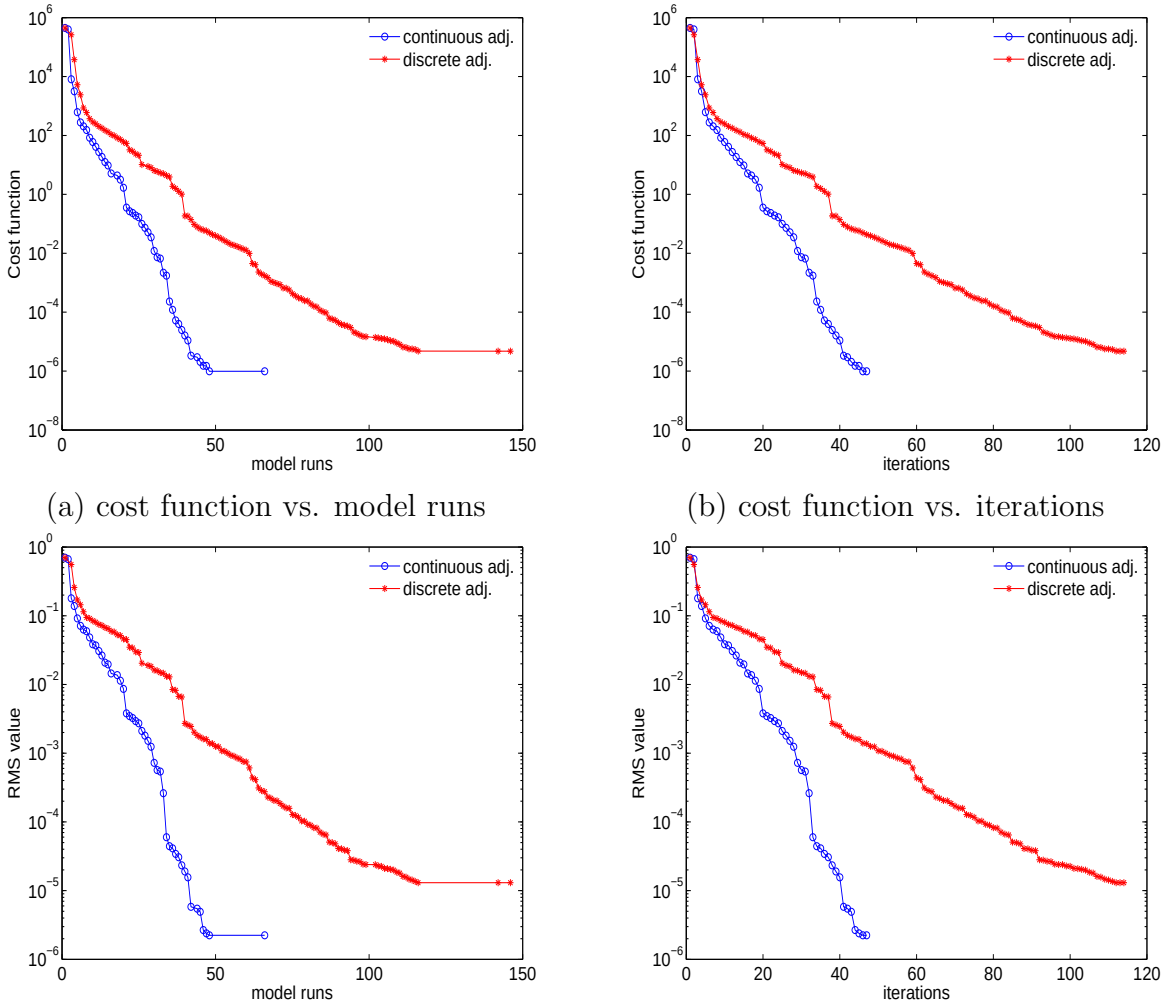


(b) reference and retrieved initial condition (grids 35)



(c) reference and retrieved initial condition (grids 68)

Figure 4.5: Solution of 4D-Var optimization 1D test for case where reference: $c^0_{ref} = 1 + signum(1.0, \sin(3x))$, initial guess: $c^0_P = 1.0$ and 200 observation points. (a) reference, perturbed and retrieved initial condition. (b) reference and retrieved initial condition grid 35. (c) reference and retrieved initial condition grid 68.



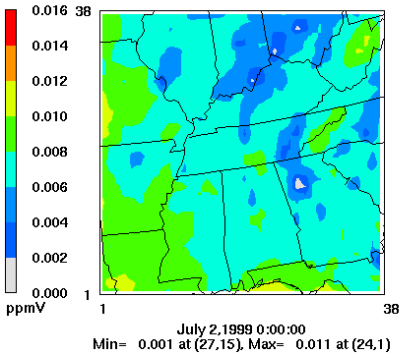
(a) cost function vs. model runs

(b) cost function vs. iterations

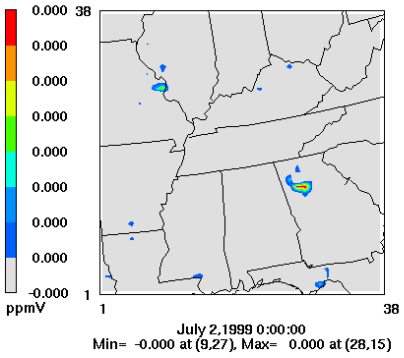
(c) rms vs. model runs

(d) rms vs. iterations

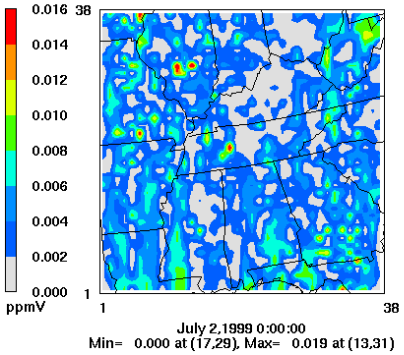
Figure 4.6: L-BFGS convergence of 4D-Var optimization 1D test for case where reference: $c^0_{ref} = 1 + signum(1.0, \sin(3x))$, initial guess: $c^0_P = 1.0$ and 200 observation points. (a) cost function value as a function of model runs. (b) cost function value as a function of iteration. (c) rms value as a function of model runs. (d) rms value as a function of iteration.



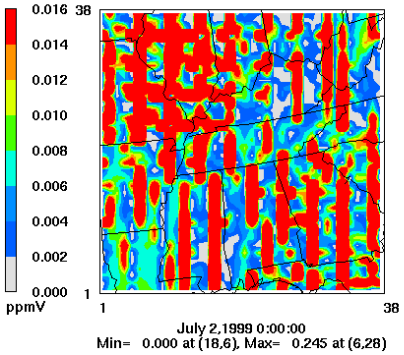
(a) Perturbation in initial conditions



(b) Optimum error: Continuous adj.

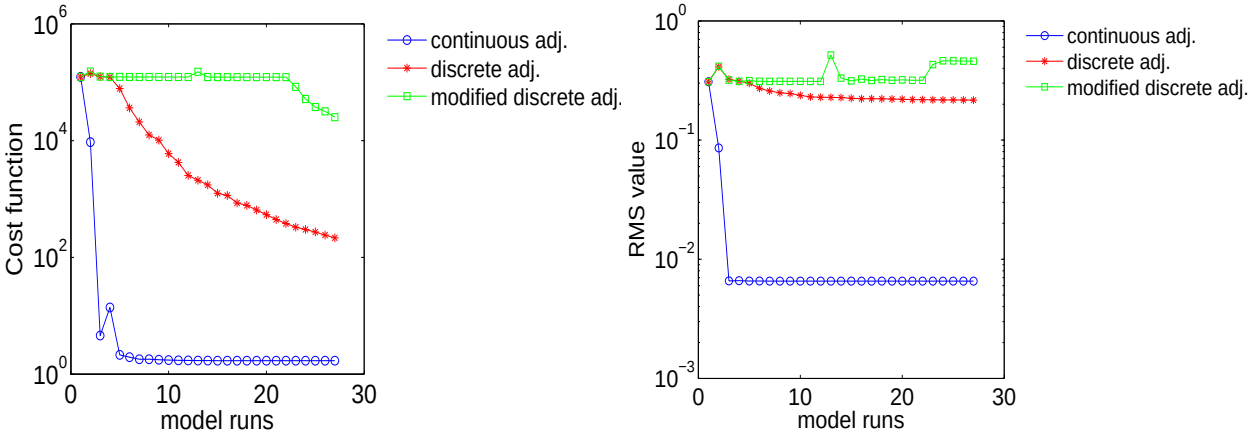


(c) Optimum error: Discrete adj.



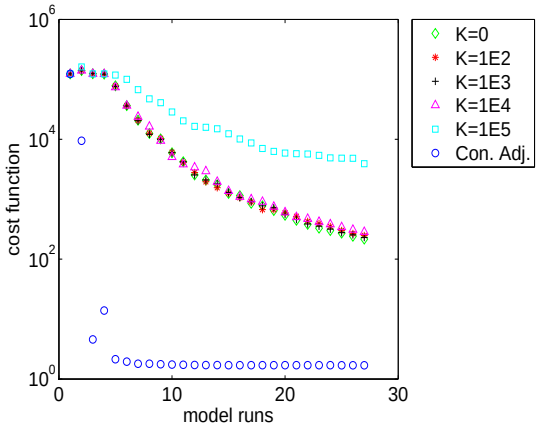
(d) Optimum error: Modified Discrete adj.

Figure 4.7: 4D-Var data assimilation carried out for 12 hours with species O3. (a) difference between perturbed and reference concentration at initial time; (b) difference between optimized and reference concentration at initial time using continuous adjoint; (c) difference between optimized and reference concentration at initial time using discrete adjoint; (d) difference between optimized and reference concentration at initial time using modified discrete adjoint.

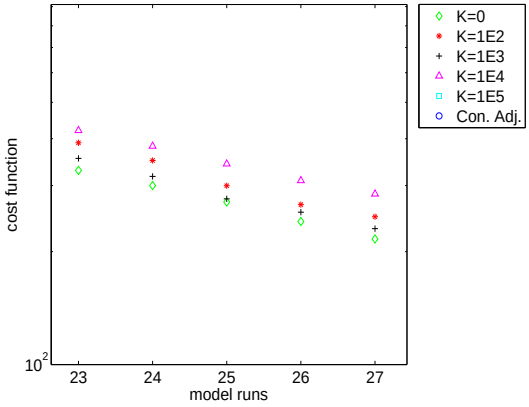


(a) log of cost function vs. model runs (b) rms vs. model runs

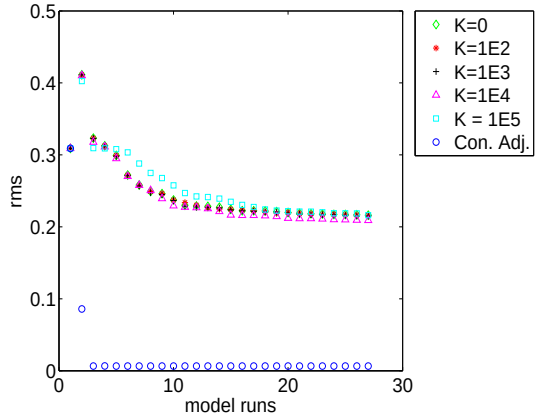
Figure 4.8: CMAQ 4D-Var with synthetic data. Convergence of L-BFGS optimization. (a) Log of the cost function values vs. model runs; (b) RMS values vs. model runs.



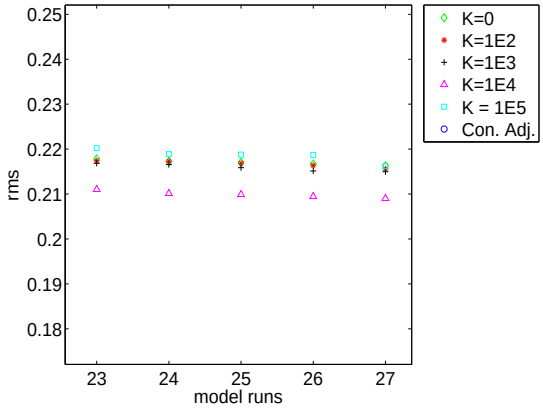
(a) cost function values vs. model (global view)



(b) cost function values vs. model (local view)

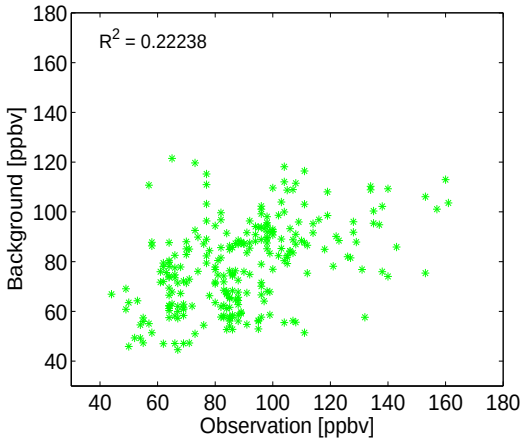


(c) RMS values vs. model (global view)

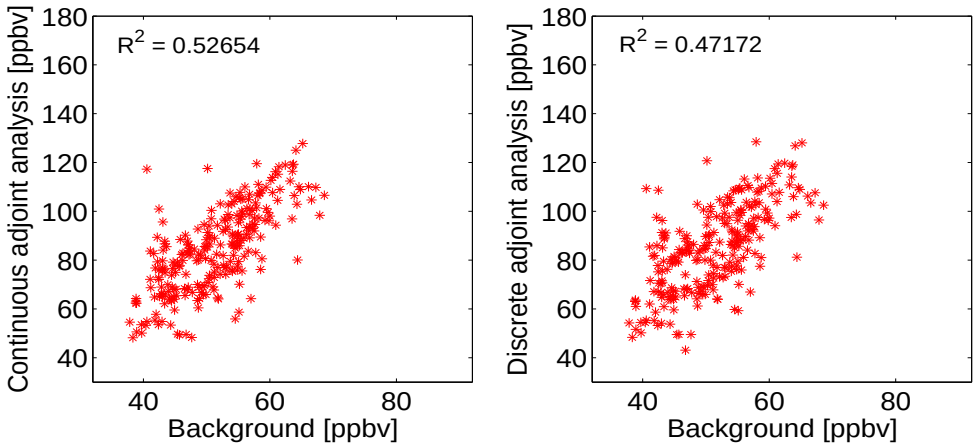


(d) RMS values vs. model (local view)

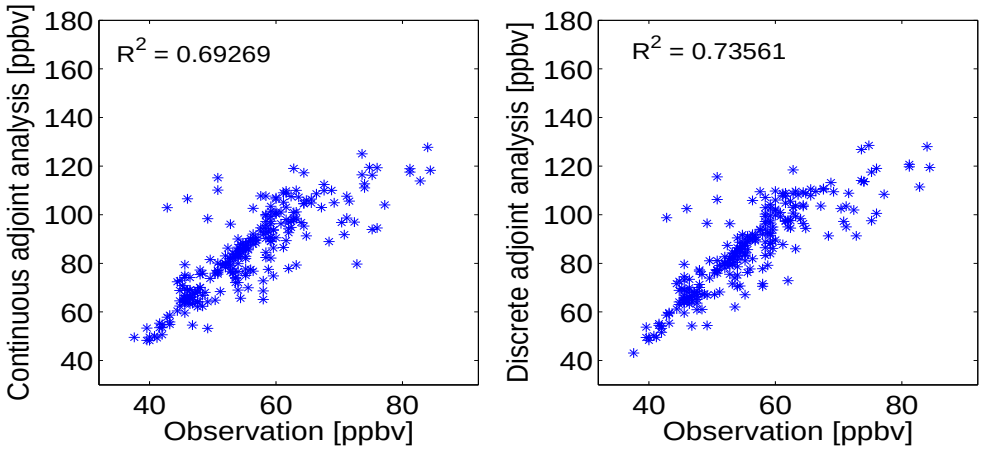
Figure 4.9: CMAQ with synthetic observations. Convergence of L-BFGS optimization for artificial diffusion. (a) cost function values vs. model runs global view); (b) cost function values vs. model runs (local view); (c) RMS values vs. model runs (global view); (d) RMS values vs. model runs (local view).



(a) observation vs. background

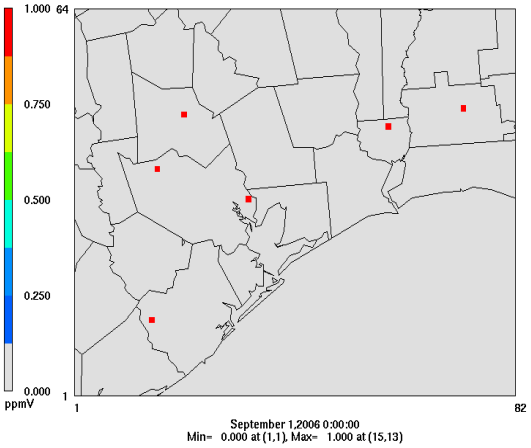


(b) analysis vs. background



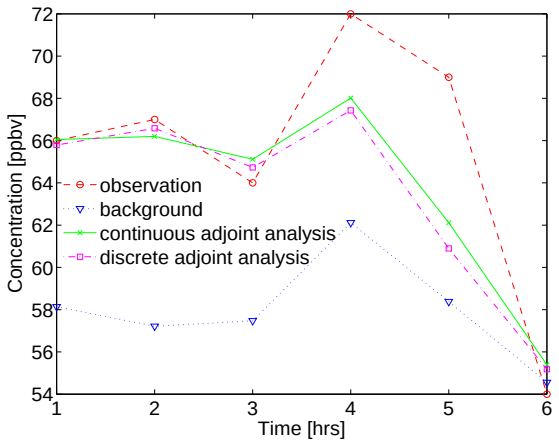
(c) analysis vs. observation

Figure 4.10: 4D-Var with AIRNOW observations. Scatter plots for both continuous adjoint and discrete adjoint. (a) observation vs. background; (b) analysis vs. background with left column continuous adjoint analysis and right column discrete adjoint analysis; (c) analysis vs. observation with left column continuous adjoint analysis and right column discrete adjoint analysis.

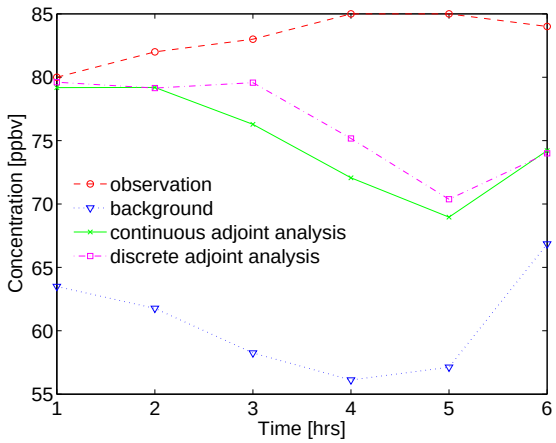


plot of observation locations

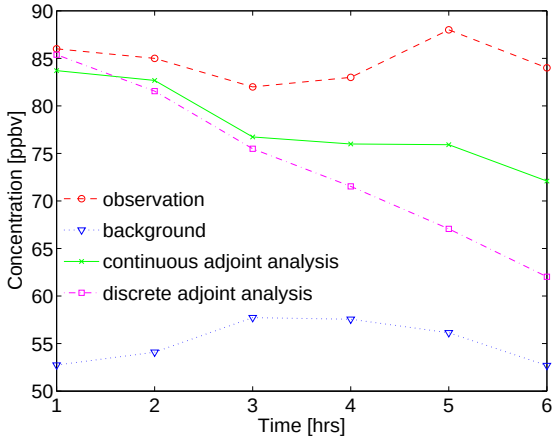
Figure 4.11: Plot of observation locations over Texas. Each red dot represents a observation location.



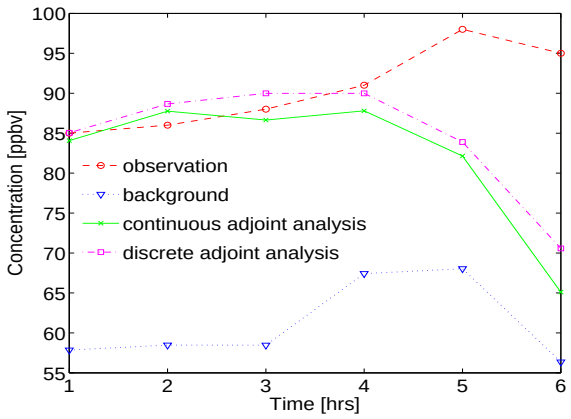
(a) location (15, 13, 1)



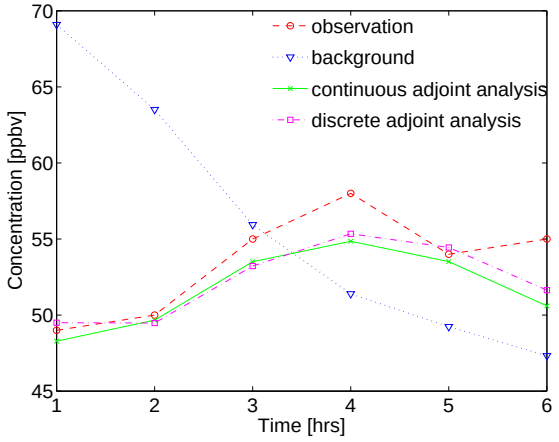
(b) location (16, 38, 1)



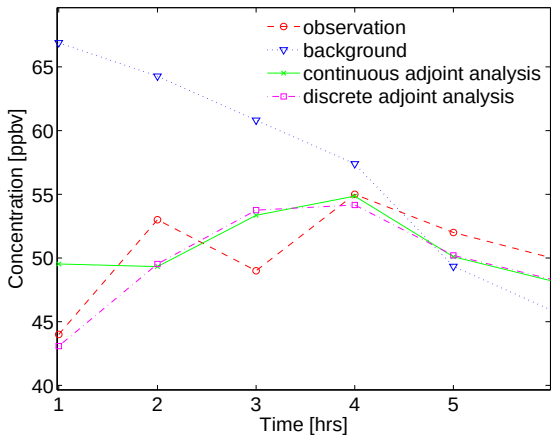
(c) location (21, 47, 1)



(d) location (33, 33, 1)

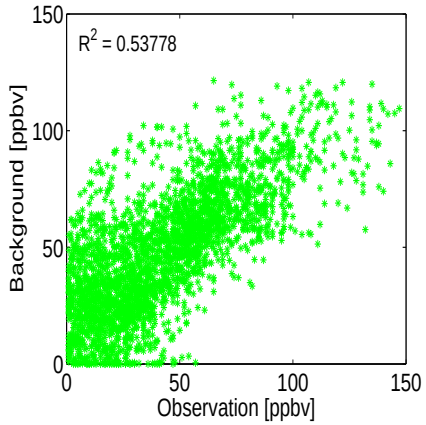


(e) location (59, 45, 1)

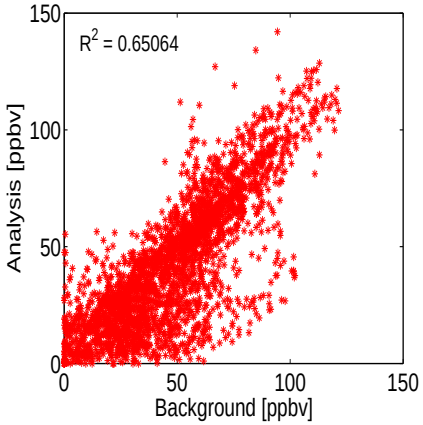


(f) location (73, 48, 1)

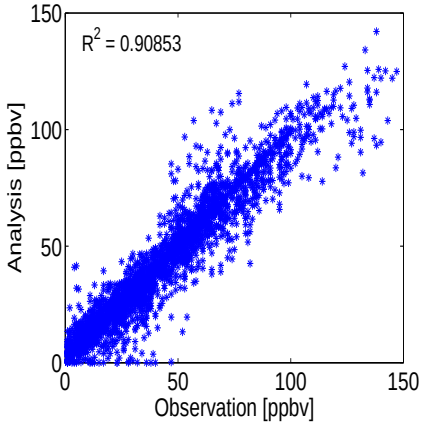
Figure 4.12: CMAQ 4D-Var with real observation data for date Sept. 1, 2006 from 18:00 to 24:00. Time series plots of ozone concentration at six observation locations represented by (col, row, layer) vector. (a) location (15,13,1); (b) location (16, 38, 1); (c) location (21, 47, 1); (d) location (33, 33, 1); (e) location (59, 45, 1); (f) location (73, 48, 1)



(a) observation vs. background

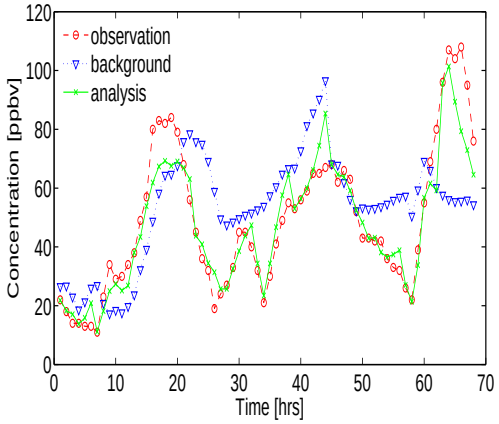


(b) analysis vs. background

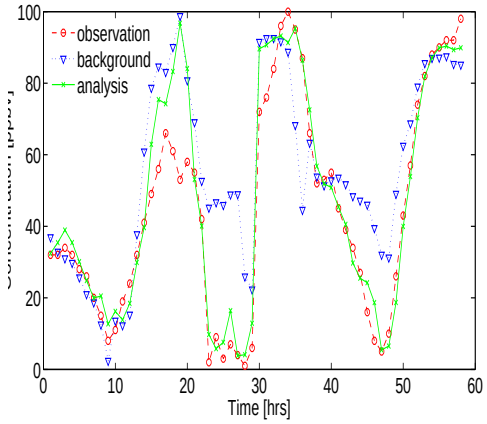


(c) analysis vs. observation

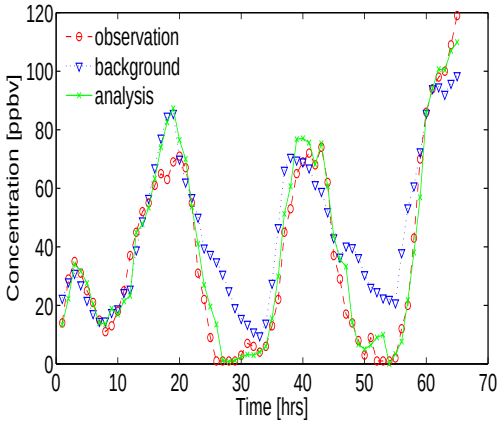
Figure 4.13: CMAQ with real data. Scatter plots of observations and the corresponding model predictions for Aug. 30, 2006 to Sept. 1, 2006.



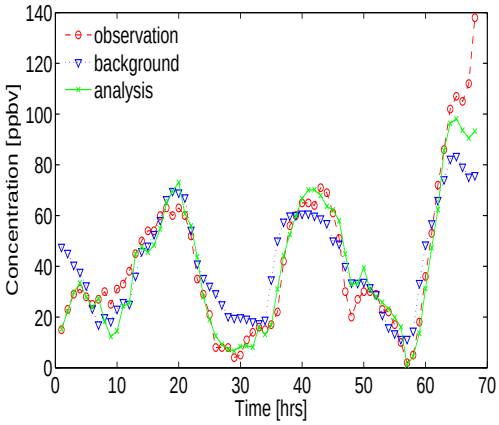
(a) location (10, 20, 1)



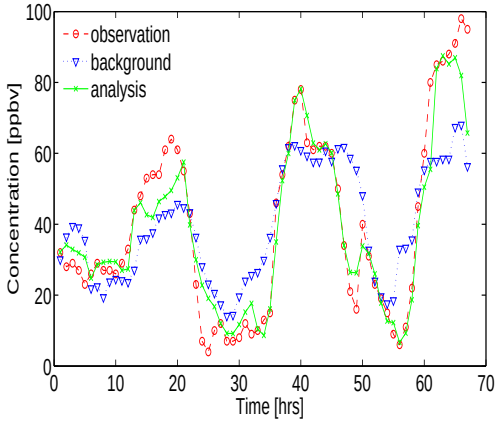
(b) location (21, 32, 1)



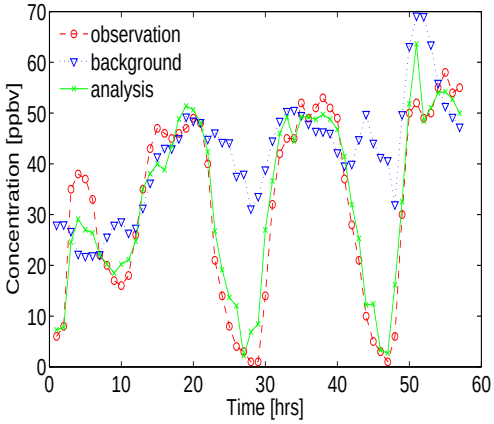
(c) location (25, 32, 1)



(d) location (28, 33, 1)



(e) location (33, 33, 1)



(f) location (59, 45, 1)

Figure 4.14: Time series of observations, background model prediction, and analysis model predictions at selected stations on Aug. 30, 2006 to Sept. 1, 2006.

Chapter 5

Conclusions and Future Work

In this thesis we have constructed, implemented, validated, and tested new adjoint modules for CMAQ-ADJ. They include a discrete adjoint of the advection process and adjoint sensitivities with respect to boundary conditions and boundary scaling factors.

The results show that discrete adjoint sensitivities agree better with finite difference values than continuous adjoint sensitivities. The reason of the discrepancy between continuous adjoint and finite difference values has been investigated. This discrepancy is partly due to the nonlinearity of PPM scheme. Consequently, the location of the receptor is important. If the receptor is placed such as to capture all the important perturbations of the solution, the finite difference results are accurate. However, if the receptor happens to capture only the spurious wiggles caused by the nonlinearity of PPM, the finite difference values are inaccurate.

A comparison of the continuous and discrete adjoint sensitivities in the context of 1D linear advection PDE indicates that the discrete adjoint may introduce wiggles in the gradients at particular grid points. On the other hand, the good agreement between finite differences and discrete adjoint sensitivities reveals that discrete adjoint sensitivities accurately account for the nonlinearity of PPM. In other words, the wiggles in the discrete adjoint gradients result from the nonlinearity of PPM as well.

4D-Var optimization with synthetic observations in CMAQ-ADJ converges much faster when using the continuous adjoint gradients than with discrete adjoint gradients. The optimization results with real AIRNOW observations, however, are similar for both the continuous adjoint and discrete adjoint approaches.

An idealized 4D-Var optimization experiment with only 1D advection constraint is conducted. This experiment shows that the optimal solution found by the continuous adjoint approach agrees better with the reference than the optimal solution given by the discrete adjoint approach. In addition, the continuous adjoint approach uses fewer model runs per iteration, and consequently achieves faster convergence. This suggests that the continuous

adjoint provides suitable descent directions.

Adjoint sensitivities with respect to boundary conditions and boundary condition scaling factors are implemented in CMAQ. The validation is performed on both a 1D advection model and on CMAQ. The results indicate that in our current implementation the sensitivity to boundary conditions is about 40% off the finite difference results. This difference seems to be systematic (points are aligned well in the scatter plot). Further work is needed to better understand these differences and to improve the results.

This work is the first attempt to improve lateral boundary conditions in regional chemical transport modeling via data assimilation. Specifically, we first perform 4D-Var assimilation with CMAQ using initial conditions as control variables; next, starting with the optimized initial conditions, we perform 4D-Var assimilation using boundary condition scaling factors as control variables. The results show that the optimization for boundary conditions leads to a further cost function decrease of 40%. However, the new analysis does not offer a considerable an improvement over the initial analysis results (4D-Var with initial conditions as control variables). The reason needs to be investigated in the future. In this regard, a joint optimization for correcting both the initial and the boundary conditions may prove to be a better approach. Moreover, the optimization of hourly boundary conditions is expected to give better results than the optimization of boundary scaling factors.

Bibliography

[1] Byun, D. W., Ching, J. K. S.: Science Algorithms of the EPA Models-3 Community Multiscale Air Quality (CMAQ) Modeling System; U.S. EPA/600/R-99/030; U.S. Environmental Protection, Agency: Research Triangle Park, NC, (1999).

[2] Byun, D.W., Schere, K.: Review of the Governing Equations, Computational Algorithms, and Other Components of the Models-3 Community Multiscale Air Quality (CMAQ) Modeling System, Applied Mechanics Reviews, vol 59, p. 51-77 (2006).

[3] Blum, J, Dimet, FX. L. and Navon, I.M.: Data Assimilation for Geophysical Fluids,Computational Methods for the Atmosphere and the Oceans, Volume 14: Special Volume (Handbook of Numerical Analysis). R. Temam and J. Tribbia, eds. Elsevier Science Ltd, New York (Philippe G. Ciarlet, Editor), 2008.

[4] Colella, P., and Woodward, P. R: The Piecewise Parabolic Method (PPM) for Gas-Dynamical Simulations, J. Comput. Phys., 54, pp.174-201 (1984).

[5] CMAQ website: http://www.cmaq-model.org/.

[6] Errico, R. M.: What is an adjoint model? Bull. Amer. Meteor. Soc., 78, 2577-2591 (1997).

[7] Errico, R. M., and K. D. Raeder: An examination of the accuracy of the linearization a mesoscale model with moist physics. Quart. J. Roy. Meteor. Soc., 125, 169195, (1999).

[8] Giering, R. and Kaminski, T: Recipes for Adjoint code Construction. ACM Trans. Math. Softw., 24: 437-474 (1998).

[9] Godunov, S. K.: A finite difference scheme for numerical computation of discontinuous solutions of equations in fluid dynamics.Math. Sb., 47, 357-393, (1959).

[10] Hakami, A., Henze, D.K., Seinfeld, J.H., Singh, K., Sandu, A.,Kim, S., Byun, D., and Li, Q.: The Adjoint of CMAQ. Environ. Sci. Technol., 41, no.22: 7807-7817 (2007).

[11] Janiskova, M., J.-N. Thepaut, and J.-F. Geleyn: Simplified and regular physical parameterizations for incremental four-dimensional varaitional assimilation. Mon. Wea. Rev., 127, 2645, (1999).

[12] Z. Liu and A. Sandu: "Analysis of Discrete Adjoints of Numerical Methods for the Advection Equation". *International Journal for Numerical Methods in Fluids*, Vol. 56, Issue 7, p. 769–803, doi:10.1002/fld.1550, 2008.

[13] Liu, D.C. and Nocedal, J.: On the limited memory BFGS method for large-scale optimization. Math. Programming 45: 503-528, (1989).

[14] Mahfouf, J.-F.: Influence of physical processes on the tangentlinear approximation. Tellus, 51A, 147166, (1999).

[15] Protas, B.: Adjoint-based optimization of PDE systems with alternative gradients. J. Comput. Phys. 227, 6490-6510 (2008)

[16] Sandu, A., Daescu, D. N., Carmichael, G. R., Chai, T. F.: Adjoint sensitivity analysis of regional air quality models. J. Comput. Phys., 204, 222-252 (2005).

[17] Singh, K., Sandu, A., Hakami, A. and Seinfeld: CMAQ-ADJv4.5. http://people.cs.vt.edu/ asandu/Software/CMAQ_ADJ/CMAQ_ADJ.html (2007)

[18] Singh, K., Sandu, A., Hakami, A. and Seinfeld, J.: CMAQ v4.5 Adjoint Users's Manual. (2007)

[19] Thuburn J. and Hainey T.: Adjoints of Nonoscillatory Advection Schemes. J. Comput. Phys., 171, pp.616-631 (2001).

[20] Vukicevic, T., and P. Hess: Analysis of tropospheric transport in the Pacific basin using the adjoint technique. J. Geophys. Res., 105, 72137230, (2000).

[21] Vukicevic , T., and J.-W. Bao: The effect of linearization errors on 4DVAR data assimilation. Mon. Wea. Rev., 126, 16951706, (1998).

[22] Vukicevic T., Steyskal M. and Hecht M.: Properties of Advection Algorithms in the Context of Variational Data Assimilation. Monthly Weather Review, Vol 129, Iss 5, pp 1221-1231 (2001).

[23] Zupanski, D and F. Mesinger: Four-dimensional variational assimilation of precipitation data. Mon. Wea. Rev., 123, 11121127, (1995).

Appendix A

Program Source

A.1 Discrete Adjoint Advection Code

The discrete adjoint of advection is developed with the aid of automatic differentiation tool, TAMC. TAMC receives the source code of the forward model as input and returns the corresponding adjoint code. Specifically, each statement of the code of the forward model is viewed as a function. Therefore, the differentiation of the code of the forward model can be achieved by the chain rule. The differentiated code (Jacobian of the forward model) is then transposed to get the corresponding adjoint code.

To obtain the discrete adjoint of advection, the forward subroutine HPPM is provided to TAMC and its adjoint ADHPPM is returned by TAMC. The complete code of ADHPPM is shown as follows.

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, in the adjoint run, for the multiple advection sub-steps within a synchronization time step for a couple of reasons: 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.

```
C                               DISCLAIMER
C
```

```
C    THIS FILE WAS GENERATED BY TAMC VERSION 5.3.2
C
C    THE AUTHOR DOES NOT MAKE ANY WARRANTY, EXPRESS OR IMPLIED,    OR ASSUMES
C    ANY LEGAL LIABILITY OR RESPONSIBILITY FOR THE ACCURACY, COMPLETENESS,
C    OR USEFULNESS OF ANY INFORMATION OR PROCESS DISCLOSED, OR REPRESENTS
C    THAT ITS USE WOULD NOT INFRINGE PRIVATELY OWNED RIGHTS.
C
C    THIS CODE IS FOR NON-PROFIT-ORIENTED ACADEMIC RESEARCH AND EDUCATION
C    ONLY.  ANY COMMERCIAL OR OTHER PROFIT-ORIENTED USE OR EVALUATION IS
C    STRICTLY FORBIDDEN.  PASSING THIS CODE TO ANY THIRD PARTY IS NOT
C    ALLOWED.
C
C    FOR COMMERCIAL OR OTHER PROFIT-ORIENTED APPLICATIONS PLEASE CONTACT
C    INFO@FASTOPT.COM
C
      SUBROUTINE ADHPPM( NI, CONI, VEL, DT, DSI, ORI, ADCONI )
C*********************************************************************
C*********************************************************************
C** THIS ROUTINE WAS GENERATED BY THE                                **
C** TANGENT LINEAR AND ADJOINT MODEL COMPILER,  TAMC 5.3.2     **
C*********************************************************************
C*********************************************************************
C=====================================================
C ALL ENTRIES ARE DEFINED EXPLICITLY
C=====================================================
      USE HGRD_DEFN

      USE SUBST_MODULES                 ! STENEX

      IMPLICIT NONE


C INCLUDES:
      INCLUDE SUBST_IOPARMS       ! I/O PARAMETERS DEFINITIONS
      INCLUDE SUBST_IODECL        ! I/O DEFINITIONS AND DECLARATIONS
      INCLUDE SUBST_PE_COMM       ! PE COMMUNICATION DISPLACEMENT AND DIRECTION

C=====================================================
C DEFINE PARAMETERS
C=====================================================
      REAL EPS
      PARAMETER ( EPS = 0.01 )
      REAL ETA1
      PARAMETER ( ETA1 = 20. )
      REAL ETA2
      PARAMETER ( ETA2 = 0.05 )
!     LOGICAL STEEPEN
!     PARAMETER ( STEEPEN =  .FALSE.  )

C FLAG FOR DISCONTINUTY CAPTURING (STEEPENING)
#ifdef STEEPEN_FLAG
```

```
      LOGICAL, PARAMETER :: STEEPEN = .TRUE.
#else
      LOGICAL, PARAMETER :: STEEPEN = .FALSE.
#endif

      REAL TWO3RDS
      PARAMETER ( TWO3RDS = 2./3. )

      REAL, ALLOCATABLE, SAVE :: ALPHA ( : )
      REAL, ALLOCATABLE, SAVE :: BETA  ( : )
!     *END CHANGE BY SNL
      REAL, ALLOCATABLE, SAVE :: MU    ( : ) ! LATTICE VAR. FOR CM
      REAL, ALLOCATABLE, SAVE :: NU    ( : ) ! LATTICE VAR. FOR CM
      REAL, ALLOCATABLE, SAVE :: LAMBDA( : ) ! LATTICE VAR. FOR CM
      REAL, ALLOCATABLE, SAVE :: CHI   ( : ) ! LATTICE VAR. FOR DC
      REAL, ALLOCATABLE, SAVE :: PSI   ( : ) ! LATTICE VAR. FOR DC
      REAL, ALLOCATABLE, SAVE :: ZETA  ( : ) ! LATTICE VAR. FOR ETABAR
      REAL, ALLOCATABLE, SAVE :: SIGMA ( : ) ! LATTICE VAR. FOR D2C
      REAL, ALLOCATABLE, SAVE :: TAU   ( : ) ! LATTICE VAR. FOR D2C

      INTEGER, SAVE :: MY_NI
      INTEGER, SAVE :: START1, END1
      INTEGER, SAVE :: START2, END2
      INTEGER, SAVE :: START3, END3
      INTEGER, SAVE :: NSPCS
      INTEGER, SAVE :: MNR
      LOGICAL, SAVE :: BNDY_LO_PE, BNDY_HI_PE
      CHARACTER( 96 ) :: XMSG = ' '

      CHARACTER, SAVE :: FIRSTORI = ' '
      LOGICAL, SAVE :: FIRSTIME = .TRUE.

C=======================================================
C DEFINE ARGUMENTS
C=======================================================
      REAL ADCONI(:,:)
      REAL CONI(:,:)
      REAL DSI(:)
      REAL DT
      INTEGER NI
      CHARACTER ORI
      REAL VEL(:)

C=======================================================
C DEFINE LOCAL VARIABLES
C=======================================================
      REAL A
      REAL ADC6(NI,SIZE(CONI,2))
      REAL ADCL(NI,SIZE(CONI,2))
      REAL ADCLD(NI,SIZE(CONI,2))
      REAL ADCM(NI+1,SIZE(CONI,2))
```

```
      REAL ADCON(0:NI+1,SIZE(CONI,2))
      REAL ADCR(NI,SIZE(CONI,2))
      REAL ADCRD(NI,SIZE(CONI,2))
      REAL ADD2C(0:NI+1,SIZE(CONI,2))
      REAL ADDC(0:NI+1,SIZE(CONI,2))
      REAL ADETA(NI,SIZE(CONI,2))
      REAL ADETABAR(NI,SIZE(CONI,2))
      REAL ADFM(NI+1,SIZE(CONI,2))
      REAL ADFP(0:NI,SIZE(CONI,2))
      REAL B
      REAL C
      REAL C6(NI,SIZE(CONI,2))
      REAL CL(NI,SIZE(CONI,2))
      REAL CLD(NI,SIZE(CONI,2))
      REAL CM(NI+1,SIZE(CONI,2))
      REAL CON(0:NI+1,SIZE(CONI,2))
      REAL CONIH(0:NI+1,SIZE(CONI,2))
      REAL CONII(0:NI+1,SIZE(CONI,2))
      REAL CR(NI,SIZE(CONI,2))
      REAL CRD(NI,SIZE(CONI,2))
      REAL D
      REAL D2C(0:NI+1,SIZE(CONI,2))
      REAL DC(0:NI+1,SIZE(CONI,2))
      REAL DS(-1:NI+1)
      REAL ETA(NI,SIZE(CONI,2))
      REAL ETABAR(NI,SIZE(CONI,2))
      REAL FM(NI+1,SIZE(CONI,2))
      REAL FP(0:NI,SIZE(CONI,2))
      REAL GAMMA
      INTEGER I
      INTEGER S
      REAL X
      REAL Y
      INTEGER :: ALLOCSTAT

C-----------------------------------------------------------------
C SAVE ARGUMENTS
C-----------------------------------------------------------------
      CONIH(:,:) = CONI(:,:)

C-----------------------------------------------------------------
C RESET LOCAL ADJOINT VARIABLES
C-----------------------------------------------------------------
      ADC6(:,:) = 0.
      ADCL(:,:) = 0.
      ADCLD(:,:) = 0.
      ADCM(:,:) = 0.
      ADCON(:,:) = 0.
      ADCR(:,:) = 0.
      ADCRD(:,:) = 0.
      ADD2C(:,:) = 0.
```

```
      ADDC(:,:) = 0.
      ADETA(:,:) = 0.
      ADETABAR(:,:) = 0.
      ADFM(:,:) = 0.
      ADFP(:,:) = 0.

C-----------------------------------------------------------------
C ROUTINE BODY
C-----------------------------------------------------------------
C-----------------------------------------------------------------
C FUNCTION AND TAPE COMPUTATIONS
C-----------------------------------------------------------------
      DO I = -1, NI+1
        DS(I) = DSI(I+2)
      END DO


      IF (ORI .NE. FIRSTORI) THEN

        FIRSTORI = ORI
        NSPCS = SIZE(CONI,2)
        MNR   = MAX( NCOLS, NROWS )

        CALL SUBST_LOOP_INDEX ( ORI, 2, NI, -1, MY_NI, START1, END1 )
        CALL SUBST_LOOP_INDEX ( ORI, 3, NI, -1, MY_NI, START2, END2 )
        CALL SUBST_LOOP_INDEX ( ORI, 3, NI, -2, MY_NI, START3, END3 )

        CALL SUBST_HI_LO_BND_PE ( ORI, BNDY_LO_PE, BNDY_HI_PE )

        IF ( FIRSTIME ) THEN
          FIRSTIME = .FALSE.

!*BEGIN CHANGE BY SNL
          ALLOCATE ( ALPHA ( MNR ), STAT = ALLOCSTAT )
          ALLOCATE ( BETA  ( MNR ), STAT = ALLOCSTAT )
!*END CHANGE BY SNL
          ALLOCATE ( MU    ( MNR ), STAT = ALLOCSTAT )
          ALLOCATE ( NU    ( MNR ), STAT = ALLOCSTAT )
          ALLOCATE ( LAMBDA( MNR ), STAT = ALLOCSTAT )
          ALLOCATE ( CHI   ( MNR ), STAT = ALLOCSTAT )
          ALLOCATE ( PSI   ( MNR ), STAT = ALLOCSTAT )
          ALLOCATE ( ZETA  ( MNR ), STAT = ALLOCSTAT )
          ALLOCATE ( SIGMA ( MNR ), STAT = ALLOCSTAT )
          ALLOCATE ( TAU   ( MNR ), STAT = ALLOCSTAT )

          IF ( ALLOCSTAT .NE. 0 ) THEN
            XMSG = 'FAILURE ALLOCATING LATTICE VARIABLE(S)'
            CALL M3EXIT ( 'HPPM', 0, 0, XMSG, XSTAT1 )
          END IF
        END IF
```

```
   DO I = START1, END1
      ALPHA(I) = DS(I)+DS(I+1)
      BETA(I) = DS(I-1)+DS(I)
      GAMMA = DS(I-2)+DS(I-1)
      D = DS(I)/(BETA(I)+DS(I+1))
      CHI(I) = D*(DS(I-1)+BETA(I))/ALPHA(I)
      PSI(I) = D*(ALPHA(I)+DS(I+1))/BETA(I)
      A = DS(I-1)/BETA(I)
      B = 2.*DS(I)/BETA(I)
      C = 1./(ALPHA(I)+GAMMA)
      MU(I) = C*DS(I-1)*GAMMA/(DS(I-1)+BETA(I))
      NU(I) = C*DS(I)*ALPHA(I)/(DS(I)+BETA(I))
      LAMBDA(I) = A+MU(I)*B-2.*NU(I)*A
   END DO
   IF (STEEPEN) THEN
      DO I = START1, END1
         C = 1./(BETA(I)+DS(I+1))
         SIGMA(I) = C/ALPHA(I)
         TAU(I) = C/BETA(I)
         ZETA(I) = 0.25*(ALPHA(I)*ALPHA(I)-ALPHA(I)*BETA(I)+BETA(I)*
$                  BETA(I))
      END DO
   ENDIF
ENDIF                              !FIRSTORI
DO I = 0, NI+1
   CON(I,:) = CONI(I+1,:)
END DO
DO S = 1, NSPCS
   FP(0,S) = 0.
   DO I = 1, NI
      FM(I,S) = 0.
      FP(I,S) = 0.
   END DO
   FM(NI+1,S) = 0.
END DO
IF (BNDY_LO_PE) THEN
   DO S = 1, NSPCS
      CM(1,S) = CON(1,S)
      CM(2,S) = (DS(1)*CON(2,S)+DS(2)*CON(1,S))/(DS(1)+DS(2))
   END DO
ENDIF
IF (BNDY_HI_PE) THEN
   DO S = 1, NSPCS
      CM(MY_NI+1,S) = CON(MY_NI,S)
      CM(MY_NI,S) = (DS(MY_NI-1)*CON(MY_NI,S)+DS(MY_NI)*CON(MY_NI-1,
$                    S))/(DS(MY_NI-1)+DS(MY_NI))
   END DO
ENDIF
IF (ORI .EQ. 'R') THEN
   CALL SUBST_COMM( CON,DSPL_N1_E0_S1_W0,DRCN_N_S,1,'1  0' )
ELSE
```

```
      CALL SUBST_COMM( CON,DSPL_N0_E1_S0_W1,DRCN_E_W,1,'1  0' )
    ENDIF

    DO S = 1, NSPCS
      DO I = START1, END1
        DC(I,S) = CHI(I)*(CON(I+1,S)-CON(I,S))+PSI(I)*(CON(I,S)-CON(I-
     $                1,S))
        IF ((CON(I+1,S)-CON(I,S))*(CON(I,S)-CON(I-1,S)) .GT. 0.) THEN
          DC(I,S) = SIGN(1.,DC(I,S))*MIN(ABS(DC(I,S)),2.*ABS(CON(I+1,
     $                S)-CON(I,S)),2.*ABS(CON(I,S)-CON(I-1,S)))
        ELSE
          DC(I,S) = 0.
        ENDIF
      END DO                    ! I
    END DO                      ! S
    IF (ORI .EQ. 'R') THEN
      CALL SUBST_COMM( DC,DSPL_N1_E0_S1_W0,DRCN_N_S,1,'1  0' )
    ELSE
      CALL SUBST_COMM( DC,DSPL_N0_E1_S0_W1,DRCN_E_W,1,'1  0' )
    ENDIF

    DO S = 1, NSPCS
      DO I = START2, END2
        CM(I,S) = CON(I-1,S)+LAMBDA(I)*(CON(I,S)-CON(I-1,S))-MU(I)*
     $              DC(I,S)+NU(I)*DC(I-1,S)
      END DO
      DO I = 1, NI
        ETA(I,S) = 0.
        CLD(I,S) = CON(I,S)
        CRD(I,S) = CON(I,S)
      END DO
    END DO
    IF (ORI .EQ. 'R') THEN
      CALL SUBST_COMM( CM,DSPL_N1_E0_S0_W0,DRCN_N,1 )
    ELSE
      CALL SUBST_COMM( CM,DSPL_N0_E1_S0_W0,DRCN_E,1 )
    ENDIF

    IF (STEEPEN) THEN
      DO S = 1, NSPCS
        DO I = START1, END1
          D2C(I,S) = SIGMA(I)*(CON(I+1,S)-CON(I,S))-TAU(I)*(CON(I,S)-
     $                CON(I-1,S))
        END DO
      END DO
      IF (ORI .EQ. 'R') THEN
        CALL SUBST_COMM( D2C,DSPL_N1_E0_S1_W0,DRCN_N_S,1,'1  0' )
      ELSE
        CALL SUBST_COMM( D2C,DSPL_N0_E1_S0_W1,DRCN_E_W,1,'1  0' )
      ENDIF
      DO S = 1, NSPCS
```

```
        DO I = START3, END3
            IF ((-(D2C(I+1,S)*D2C(I-1,S))) .GT. 0. .AND. ABS(CON(I+1,S)-
$               CON(I-1,S))-EPS*MIN(ABS(CON(I+1,S)),ABS(CON(I-1,S))) .GT.
    0.) THEN
                ETABAR(I,S) = -(ZETA(I)*(D2C(I+1,S)-D2C(I-1,S))/(CON(I+1,
$               S)-CON(I-1,S)))
            ELSE
                ETABAR(I,S) = 0.
            ENDIF
            ETA(I,S) = MAX(0.,MIN(ETA1*(ETABAR(I,S)-ETA2),1.))
            CRD(I,S) = CON(I+1,S)-0.5*DC(I+1,S)
            CLD(I,S) = CON(I-1,S)+0.5*DC(I-1,S)
        END DO
    END DO
ENDIF

IF (ORI .EQ. 'R') THEN
    CALL SUBST_COMM( VEL,DSPL_N1_E0_S0_W0,DRCN_N )
ELSE
    CALL SUBST_COMM( VEL,DSPL_N0_E1_S0_W0,DRCN_E )
ENDIF

DO S = 1, NSPCS
    DO I = 1, MY_NI
        CR(I,S) = CM(I+1,S)+ETA(I,S)*(CRD(I,S)-CM(I+1,S))
        CL(I,S) = CM(I,S)+ETA(I,S)*(CLD(I,S)-CM(I,S))
        IF ((CR(I,S)-CON(I,S))*(CON(I,S)-CL(I,S)) .GT. 0.) THEN
            DC(I,S) = CR(I,S)-CL(I,S)
            C6(I,S) = 6.*(CON(I,S)-0.5*(CL(I,S)+CR(I,S)))
            IF (DC(I,S)*C6(I,S) .GT. DC(I,S)*DC(I,S)) THEN
                CL(I,S) = 3.*CON(I,S)-2.*CR(I,S)
            ELSE IF ((-(DC(I,S)*DC(I,S))) .GT. DC(I,S)*C6(I,S)) THEN
                CR(I,S) = 3.*CON(I,S)-2.*CL(I,S)
            ENDIF
        ELSE
            CL(I,S) = CON(I,S)
            CR(I,S) = CON(I,S)
        ENDIF
        DC(I,S) = CR(I,S)-CL(I,S)
        C6(I,S) = 6.*(CON(I,S)-0.5*(CL(I,S)+CR(I,S)))
    END DO
    DO I = 1, MY_NI
        IF (VEL(I) .LT. 0.) THEN
            Y = -(VEL(I)*DT)
            X = Y/DS(I)
            FM(I,S) = Y*(CL(I,S)+0.5*X*(DC(I,S)+C6(I,S)*(1.-TWO3RDS*X)))
        ENDIF
        IF (VEL(I+1) .GT. 0.) THEN
            Y = VEL(I+1)*DT
            X = Y/DS(I)
            FP(I,S) = Y*(CR(I,S)-0.5*X*(DC(I,S)-C6(I,S)*(1.-TWO3RDS*X)))
```

```
            ENDIF
         END DO
      END DO
      IF (BNDY_LO_PE) THEN
         IF (VEL(1) .GT. 0.) THEN
            DO S = 1, NSPCS
               Y = VEL(1)*DT
               FP(0,S) = Y*CON(0,S)
            END DO
         ENDIF
      ENDIF
      IF (BNDY_HI_PE) THEN
         IF (VEL(NI+1) .LT. 0.) THEN
            DO S = 1, NSPCS
               Y = -(VEL(NI+1)*DT)
               FM(NI+1,S) = Y*CON(NI+1,S)
            END DO
         ENDIF
      ENDIF

      DO S = 1, NSPCS
         DO I = 1, MY_NI
            CON(I,S) = CON(I,S)+(FP(I-1,S)-FP(I,S)+FM(I+1,S)-FM(I,S))/
     $                  DS(I)
         END DO
      END DO
      DO I = 0, NI+1
         CONI(I+1,:) = CON(I,:)
      END DO

C-----------------------------------------------------------------
C SAVE DEPENDEND VARIABLES
C-----------------------------------------------------------------
      CONII(:,:) = CONI(:,:)
C-----------------------------------------------------------------
C ADJOINT COMPUTATIONS
C-----------------------------------------------------------------
      DO I = 0, NI+1
         CON(I,:) = CONI(I+1,:)
      END DO

      DO I = 0, NI+1
         ADCON(I,:) = ADCON(I,:)+ADCONI(I+1,:)
         ADCONI(I+1,:) = 0.
      END DO

      DO S = 1, NSPCS
         DO I = 1, MY_NI
            ADFM(I+1,S) = ADFM(I+1,S)+ADCON(I,S)/DS(I)
            ADFM(I,S) = ADFM(I,S)-ADCON(I,S)/DS(I)
            ADFP(I-1,S) = ADFP(I-1,S)+ADCON(I,S)/DS(I)
```

```
        ADFP(I,S) = ADFP(I,S)-ADCON(I,S)/DS(I)
    END DO
END DO

IF (BNDY_HI_PE) THEN
    IF (VEL(NI+1) .LT. 0.) THEN
        DO S = 1, NSPCS
            Y = -(VEL(NI+1)*DT)
            ADCON(NI+1,S) = ADCON(NI+1,S)+ADFM(NI+1,S)*Y
            ADFM(NI+1,S) = 0.
        END DO
    ENDIF
ENDIF

IF (BNDY_LO_PE) THEN
    IF (VEL(1) .GT. 0.) THEN
        DO S = 1, NSPCS
            Y = VEL(1)*DT
            ADCON(0,S) = ADCON(0,S)+ADFP(0,S)*Y
            ADFP(0,S) = 0.
        END DO
    ENDIF
ENDIF

CONI(:,:) = CONIH(:,:)
DO I = 0, NI+1
    CON(I,:) = CONI(I+1,:)
END DO

DO S = NSPCS, 1, -1
    DO I = MY_NI, 1, -1
        IF (VEL(I+1) .GT. 0.) THEN
            Y = VEL(I+1)*DT
            X = Y/DS(I)
            ADC6(I,S) = ADC6(I,S)+0.5*ADFP(I,S)*Y*X*(1.-TWO3RDS*X)
            ADCR(I,S) = ADCR(I,S)+ADFP(I,S)*Y
            ADDC(I,S) = ADDC(I,S)-0.5*ADFP(I,S)*Y*X
            ADFP(I,S) = 0.
        ENDIF
        IF (VEL(I) .LT. 0.) THEN
            Y = -(VEL(I)*DT)
            X = Y/DS(I)
            ADC6(I,S) = ADC6(I,S)+0.5*ADFM(I,S)*Y*X*(1.-TWO3RDS*X)
            ADCL(I,S) = ADCL(I,S)+ADFM(I,S)*Y
            ADDC(I,S) = ADDC(I,S)+0.5*ADFM(I,S)*Y*X
            ADFM(I,S) = 0.
        ENDIF
    END DO
    DO I = 1, MY_NI
        CR(I,S) = CM(I+1,S)+ETA(I,S)*(CRD(I,S)-CM(I+1,S))
        CL(I,S) = CM(I,S)+ETA(I,S)*(CLD(I,S)-CM(I,S))
```

```
          ADCL(I,S) = ADCL(I,S)-3*ADC6(I,S)
          ADCON(I,S) = ADCON(I,S)+6*ADC6(I,S)
          ADCR(I,S) = ADCR(I,S)-3*ADC6(I,S)
          ADC6(I,S) = 0.
          ADCL(I,S) = ADCL(I,S)-ADDC(I,S)
          ADCR(I,S) = ADCR(I,S)+ADDC(I,S)
          ADDC(I,S) = 0.
          IF ((CR(I,S)-CON(I,S))*(CON(I,S)-CL(I,S)) .GT. 0.) THEN
             DC(I,S) = CR(I,S)-CL(I,S)
             C6(I,S) = 6.*(CON(I,S)-0.5*(CL(I,S)+CR(I,S)))
             IF (DC(I,S)*C6(I,S) .GT. DC(I,S)*DC(I,S)) THEN
                ADCON(I,S) = ADCON(I,S)+3*ADCL(I,S)
                ADCR(I,S) = ADCR(I,S)-2*ADCL(I,S)
                ADCL(I,S) = 0.
             ELSE IF ((-(DC(I,S)*DC(I,S))) .GT. DC(I,S)*C6(I,S)) THEN
                ADCL(I,S) = ADCL(I,S)-2*ADCR(I,S)
                ADCON(I,S) = ADCON(I,S)+3*ADCR(I,S)
                ADCR(I,S) = 0.
             ENDIF
             ADCL(I,S) = ADCL(I,S)-3*ADC6(I,S)
             ADCON(I,S) = ADCON(I,S)+6*ADC6(I,S)
             ADCR(I,S) = ADCR(I,S)-3*ADC6(I,S)
             ADC6(I,S) = 0.
             ADCL(I,S) = ADCL(I,S)-ADDC(I,S)
             ADCR(I,S) = ADCR(I,S)+ADDC(I,S)
             ADDC(I,S) = 0.
          ELSE
             ADCON(I,S) = ADCON(I,S)+ADCR(I,S)
             ADCR(I,S) = 0.
             ADCON(I,S) = ADCON(I,S)+ADCL(I,S)
             ADCL(I,S) = 0.
          ENDIF
          ADCLD(I,S) = ADCLD(I,S)+ADCL(I,S)*ETA(I,S)
          ADCM(I,S) = ADCM(I,S)+ADCL(I,S)*(1-ETA(I,S))
          ADETA(I,S) = ADETA(I,S)+ADCL(I,S)*(CLD(I,S)-CM(I,S))
          ADCL(I,S) = 0.
          ADCM(I+1,S) = ADCM(I+1,S)+ADCR(I,S)*(1-ETA(I,S))
          ADCRD(I,S) = ADCRD(I,S)+ADCR(I,S)*ETA(I,S)
          ADETA(I,S) = ADETA(I,S)+ADCR(I,S)*(CRD(I,S)-CM(I+1,S))
          ADCR(I,S) = 0.
       END DO
    END DO

    IF (STEEPEN) THEN
       DO S = 1, NSPCS
          DO I = START3, END3
             IF ((-(D2C(I+1,S)*D2C(I-1,S))) .GT. 0.
$               .AND. ABS(CON(I+1,S)-
$             CON(I-1,S))-EPS*MIN(ABS(CON(I+1,S)),
$             ABS(CON(I-1,S))) .GT. 0.) THEN
             ETABAR(I,S) = -(ZETA(I)*(D2C(I+1,S)-D2C(I-1,S))
```

```
$                                   /(CON(I+1,S)-CON(I-1,S)))
            ELSE
               ETABAR(I,S) = 0.
            ENDIF
            ADCON(I-1,S) = ADCON(I-1,S)+ADCLD(I,S)
            ADDC(I-1,S) = ADDC(I-1,S)+0.5*ADCLD(I,S)
            ADCLD(I,S) = 0.
            ADCON(I+1,S) = ADCON(I+1,S)+ADCRD(I,S)
            ADDC(I+1,S) = ADDC(I+1,S)-0.5*ADCRD(I,S)
            ADCRD(I,S) = 0.
            ADETABAR(I,S) = ADETABAR(I,S)+ADETA(I,S)*
$                           (0.5-SIGN(0.5,0.-
$                           MIN(ETA1*(ETABAR(I,S)-ETA2),1.)))
$                           *(0.5+SIGN(0.5,1.-ETA1*(ETABAR(I,
$                           S)-ETA2)))*ETA1
            ADETA(I,S) = 0.
            IF ((-(D2C(I+1,S)*D2C(I-1,S))) .GT. 0.
$               .AND. ABS(CON(I+1,S)-
$               CON(I-1,S))-EPS*MIN(ABS(CON(I+1,S)),
$               ABS(CON(I-1,S))) .GT. 0.) THEN
$               ADCON(I-1,S) = ADCON(I-1,S)-ADETABAR(I,S)*(ZETA(I)
$                              *(D2C(I+1,S)-D2C(I-1,S))/((CON(I+1,S)-CON(I
    -1,S))*
$                              (CON(I+1,S)-CON(I-1,S))))
               ADCON(I+1,S) = ADCON(I+1,S)+ADETABAR(I,S)*(ZETA(I)*(D2C(I+
$                              1,S)-D2C(I-1,S))/((CON(I+1,S)-CON(I-1,S))*
$                              (CON(I+1,S)-CON(I-1,S))))
               ADD2C(I-1,S) = ADD2C(I-1,S)+ADETABAR(I,S)*(ZETA(I)/(CON(I+
$                              1,S)-CON(I-1,S)))
               ADD2C(I+1,S) = ADD2C(I+1,S)-ADETABAR(I,S)*(ZETA(I)/(CON(I+
$                              1,S)-CON(I-1,S)))
               ADETABAR(I,S) = 0.
            ELSE
               ADETABAR(I,S) = 0.
            ENDIF
         END DO
      END DO

      DO S = 1, NSPCS
         DO I = START1, END1
            ADCON(I-1,S) = ADCON(I-1,S)+ADD2C(I,S)*TAU(I)
            ADCON(I+1,S) = ADCON(I+1,S)+ADD2C(I,S)*SIGMA(I)
            ADCON(I,S) = ADCON(I,S)-ADD2C(I,S)*(SIGMA(I)+TAU(I))
            ADD2C(I,S) = 0.
         END DO
      END DO
   ENDIF

   DO S = 1, NSPCS
      DO I = 1, NI
         ADCON(I,S) = ADCON(I,S)+ADCRD(I,S)
```

```
        ADCRD(I,S) = 0.
        ADCON(I,S) = ADCON(I,S)+ADCLD(I,S)
        ADCLD(I,S) = 0.
    END DO
    DO I = START2, END2
        ADCON(I-1,S) = ADCON(I-1,S)+ADCM(I,S)*(1-LAMBDA(I))
        ADCON(I,S) = ADCON(I,S)+ADCM(I,S)*LAMBDA(I)
        ADDC(I-1,S) = ADDC(I-1,S)+ADCM(I,S)*NU(I)
        ADDC(I,S) = ADDC(I,S)-ADCM(I,S)*MU(I)
        ADCM(I,S) = 0.
    END DO
END DO

DO S = 1, NSPCS
    DO I = START1, END1
        DC(I,S) = CHI(I)*(CON(I+1,S)-CON(I,S))+PSI(I)*(CON(I,S)-CON(I-
$                   1,S))
        IF ((CON(I+1,S)-CON(I,S))*(CON(I,S)-CON(I-1,S)) .GT. 0.) THEN
            ADCON(I+1,S) = ADCON(I+1,S)+2.*ADDC(I,S)*(0.5-SIGN(0.5,2.*
$                           ABS(CON(I+1,S)-CON(I,S))-ABS(DC(I,S))))
$                           *SIGN(1.,CON(I+1,S)-CON(I,S))*SIGN(1.,DC(I,S))
            ADCON(I,S) = ADCON(I,S)-2.*ADDC(I,S)*(0.5-SIGN(0.5,2.*
$                         ABS(CON(I+1,S)-CON(I,S))-ABS(DC(I,S))))
$                         *SIGN(1.,CON(I+1,S)-CON(I,S))*SIGN(1.,DC(I,S))
            ADDC(I,S) = ADDC(I,S)*(0.5+SIGN(0.5,2.*ABS(CON(I+1,S)-CON(I,
$                        S))-ABS(DC(I,S))))*SIGN(1.,DC(I,S))*SIGN(1.,DC(I,S)
    )
        ELSE
            ADDC(I,S) = 0.
        ENDIF
        ADCON(I-1,S) = ADCON(I-1,S)-ADDC(I,S)*PSI(I)
        ADCON(I+1,S) = ADCON(I+1,S)+ADDC(I,S)*CHI(I)
        ADCON(I,S) = ADCON(I,S)+ADDC(I,S)*((-CHI(I))+PSI(I))
        ADDC(I,S) = 0.
    END DO
END DO

IF (BNDY_HI_PE) THEN
    DO S = 1, NSPCS
        ADCON(MY_NI-1,S) = ADCON(MY_NI-1,S)+ADCM(MY_NI,S)*(DS(MY_NI)/
$                           (DS(MY_NI-1)+DS(MY_NI)))
        ADCON(MY_NI,S) = ADCON(MY_NI,S)+ADCM(MY_NI,S)*(DS(MY_NI-1)/
$                         (DS(MY_NI-1)+DS(MY_NI)))
        ADCM(MY_NI,S) = 0.
        ADCON(MY_NI,S) = ADCON(MY_NI,S)+ADCM(MY_NI+1,S)
        ADCM(MY_NI+1,S) = 0.
    END DO
ENDIF
IF (BNDY_LO_PE) THEN
    DO S = 1, NSPCS
        ADCON(2,S) = ADCON(2,S)+ADCM(2,S)*(DS(1)/(DS(1)+DS(2)))
```

```
            ADCON(1,S) = ADCON(1,S)+ADCM(2,S)*(DS(2)/(DS(1)+DS(2)))
            ADCM(2,S) = 0.
            ADCON(1,S) = ADCON(1,S)+ADCM(1,S)
            ADCM(1,S) = 0.
          END DO
        ENDIF
        DO I = 0, NI+1
          ADCONI(I+1,:) = ADCONI(I+1,:)+ADCON(I,:)
          ADCON(I,:) = 0.
        END DO
C-------------------------------------------------------------
C       GET DEPENDEND VARIABLES
C-------------------------------------------------------------
        CONI(:,:) = CONII(:,:)

        END SUBROUTINE ADHPPM
```

A.2 Sensitivity to Boundary Condition

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 as we seen in equations 3.6 and 3.10. The implementation in CMAQ is illustrated in figure A.1. The code is also listed as below.

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.

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

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.

Figure A.1: Logical function call for sensitivity to boundary condition. (x advection)

```
SUBROUTINE STOB (LGRID, LGRID_B, JDATE, JTIME, DT, DX, VEL, ORI)
USE GRID_CONF              ! horizontal & vertical domain specifications
USE CGRID_SPCS             ! CGRID species number and offsets

IMPLICIT NONE
INCLUDE SUBST_IOPARMS      ! I/O parameters definitions
INCLUDE SUBST_FILES_ID     ! I/O definitions and declarations
INCLUDE SUBST_IODECL       ! I/O definitions and declarations

REAL, POINTER         :: LGRID( :,:,:,: )
REAL, POINTER         :: LGRID_B( :,:,:,: )
REAL                  :: BCON_OLD(NROWS,NLAYS)
INTEGER   ::     JDATE       ! current model date, coded YYYYDDD
INTEGER   ::     JTIME       ! current model time, coded HHMMSS
INTEGER   ::     DT
REAL      ::     DX
REAL      ::     VEL(:,:,:)
CHARACTER :: ORI
CHARACTER( 16 ) :: PNAME = 'STOB'
CHARACTER( 96 ) :: XMSG = ' '
INTEGER   ::    C, R, L, S, IOS, IOS1, IOS2

INTEGER, SAVE :: EFX     ! fixed parameter for eastern boundary
INTEGER, SAVE :: WFX     ! fixed parameter for western boundary
INTEGER, SAVE :: SFX     ! fixed parameter for southern boundary
INTEGER, SAVE :: NFX     ! fixed parameter for northern boundary
INTEGER, SAVE :: FIRSTTIME = .TRUE.

IF (FIRSTTIME) THEN
   FIRSTTIME = .FALSE.
```

```
          EFX = MY_NCOLS + 1
          WFX = 2 * MY_NCOLS + MY_NROWS + 4
          SFX = 0
          NFX = MY_NCOLS + MY_NROWS + 3

        END IF

C Caculate Sensivity to Boundary conditions
        IF (ORI .EQ. 'X') THEN      ! xadv to get west and east sensivity
           DO S = 1, NSPCSD
              DO R = 1, NROWS
                 DO L = 1, NLAYS
                    ! east
                    IF (VEL(R,L,1) .LT. 0) THEN ! east boundary inflow
                       LGRID_B(EFX+R,L,S) = -1 * VEL(R,L,1) * DT
     &                                       / DX * LGRID(NCOLS,R,L,S) ! EAST
                    ELSE
                       LGRID_B(EFX+R,L,S) = 0.0
                    END IF
                    ! west
                    IF (VEL(R,L,2) .GT. 0) THEN ! west boundary inflow
                       LGRID_B(WFX+R,L,S)  = VEL(R,L,2) * DT
     &                                        / DX * LGRID(1,R,L,S) ! WEST
                    ELSE
                       LGRID_B(WFX+R,L,S)  = 0.0
                    END IF
                 END DO
              END DO
           END DO
        ELSE IF (ORI .EQ. 'Y') THEN  ! yadv to get north and south sensitivity
           DO S = 1, NSPCSD
              DO C = 1, NCOLS
                 DO L = 1, NLAYS
                    ! north
                    IF (VEL(C,L,1) .LT. 0) THEN ! north boundary inflow
                       LGRID_B(NFX+C,L,S) = -1 * VEL(C,L,1) * DT
     &                                       / DX * LGRID(C,NROWS,L,S) ! NORTH
                    ELSE
                       LGRID_B(NFX+C,L,S) = 0.0
                    END IF
                    ! south
                    IF (VEL(C,L,2) .GT. 0) THEN ! south boundary inflow
                       LGRID_B(SFX+C,L,S)  = VEL(C,L,2) * DT
     &                                        / DX * LGRID(C,1,L,S) ! SOUTH
                    ELSE
                       LGRID_B(SFX+C,L,S)  = 0.0
                    END IF
                 END DO            ! L
              END DO               ! C
           END DO                  !S
```

```
END IF

END SUBROUTINE STOB
```

A.3 Changes in Source Files

In this section, we summarize the changes in the source files of CMAQ-ADJ corresponding to the work in this thesis. Specifically, table A.1 shows the changes of a few source file names. No codes changes are made to these files. In table A.2, we show the new drivers and subdrivers for continuous adjoint of advection. The new module of x and y advection control is presented in table A.3. In addition, table A.4 shows the new source files for the adjoint of concentration adjustment subroutine. The new drivers and subdrivers for discrete adjoint of advection are given in table A.5. New source files for discrete adjoint of advection can be found in table A.6. What's more, table A.7 shows the new files for sensitivity to boundary conditions.

Original CMAQ-ADJ	**New CMAQ-ADJ**
KPP_Data_mod.F	hrdat_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

Table A.1: File name changes without modification to codes

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

Table A.2: Drivers and subdrivers for continuous adjoint of advection

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.

Table A.3: Module for x advection and y advection control

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

Table A.4: Adjoint of adjustment factor subroutine (adjadv.F)

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

Table A.5: Drivers and subdrivers for 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.

Table A.6: New files for discrete adjoint of advection

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.
xadvppm_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.

Table A.7: New files for sensitivity to boundary conditions

Appendix B

New Implementation

B.1 Horizontal Advection Sequence Control

In CMAQ x advection and y advection are performed alternatively. If x advection is performed followed by y advection in current advection time step, y advection will be performed first in the next advection time step. In the adjoint run the advection adjoint sequence is in the reverse order of that in the forward run. Therefore, we need to record x and y advection execution order in current advection step so that we will have the correct order of execution of x and y advection in the next time step. In addition, the x, y advection execution order in the last time step of the forward run can be used to determine the execution sequence of x, y advection adjoint at the beginning step of the adjoint run.

To realize this idea, a global variable XFIRST in a module is used to record the order of x advection (adjoint) in next advection time step. If in the current step, x advection (adjoint) is executed first, then XFIRST is set to .False. and vice versa. Figure B.1 explains the sequence control of x and y advection in both forward run and adjoint run.

B.2 Time Argument for Adjoint of Advection and Diffusion

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 $K(t_{n-\frac{1}{2}})$.

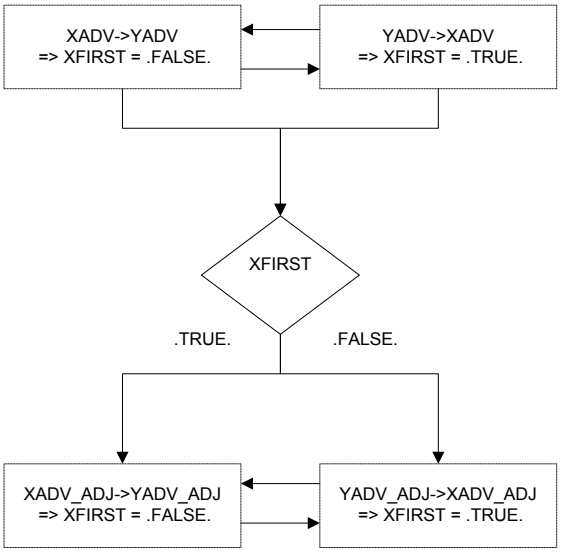


Figure B.1: Flow control of x and y advection execution in forward and adjoint run

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.

B.3 Parallelization

Parallelization of CMAQ is implemented by Dr. Peter Percell, a research professor in the department of geosciences, university of Houston (UH). In current implementation, the Message Passing Interface (MPI) is used as the parallelization tool and the approach is based on horizontal domain decomposition.

L-BFGS optimization requires cost function and its gradient at the current values of control variables. Each processor is responsible for computing these values for a subset of the whole computational domain. The results on each processor are then gathered into the root processor for L-BFGS.

For I/O manipulation, each processor gets the data for its local domain by extracting the data from the full domain. However, only the root processor is responsible for writing the data to files. Therefore, gathering local data into global arrays is required before wring.