Development of a Multi-phase Tangent Linear Model for AURAMS

A thesis submitted to
the Faculty of Graduate and Postdoctoral Affairs
in Partial Fulfillment of the requirements for the degree

Master of Applied Sciences

by

Matthew G. Russell

Department of Civil and Environmental Engineering
Carleton University

Ottawa-Carleton Institute of Civil and Environmental Engineering

May 2011

Your file Votre référence
ISBN: 978-0-494-81715-5
Our file Notre référence
ISBN: 978-0-494-81715-5

Abstract

AIR QUALITY MODELS (AQMs) are traditionally used to provide predictions of concentrations but they can also be modified to produce complementary information such as source, parameter, or process contributions and influences. Models equipped with these "probing tools" are referred to (for the lack of a better term) as instrumented AQMs. TANGENT LINEAR MODELS (TLMs), also called DECOUPLED DIRECT METHOD (DDM) models, are among the most widely used instrumented versions of AQMs.

This thesis presents a comprehensive account of the development of a multi-phase TLM for ENVIRONMENT CANADA's AQM, AURAMS. Development of a TLM for an AQM requires differentiation of governing equations and/or numerical solutions. AQMs are not generally written with differentiation in mind and can contain many features that do not readily lend themselves to linearization (i.e. differentiation). This is particularly true for aerosol processes. Challenges of TLM development for AURAMS, as well as strategies and methodologies for manual or automatic code differentiation are discussed. TLM results are compared with finite difference (brute-force) approximation of derivatives. For aerosols that exhibit highly non-linear behaviour, comparisons are made with derivatives calculated by the COMPLEX VARIABLE METHOD. In general, AURAMS-TLM shows good agreement with brute-force methods for sensitivity estimation.

TLM sensitivities from AURAMS are compared against finite-difference approximations from CMAQ in order to investigate how sensitivities compare with respect to concentrations in inter-model comparisons. Preliminary results for a one day simulation are presented but they do not support the hypothesis that sensitivity fields are more consistent across different models.

Development of AURAMS-TLM in this work results in only the second active multi-phase AQM that is fully instrumented with forward sensitivity analysis capabilities in a TLM version. Furthermore, this implementation of AURAMS-TLM has resulted in unprecedented accuracy compared to all various AQMs that have fully or partially implemented formal sensitivity versions.

Acknowledgments

First, this work would have never been possible without the guidance and inspiration of my supervisor Dr. Amir Hakami. His advice, experience and encouragement over the last two and a half years lead me through the countless challenges experienced throughout this research. I would also like to extend my warmest thanks to Dr. Shunliu Zhao for the many sleepless nights he spent helping me with the TLM implementation in the CANADIAN AEROSOL MODULE (CAM). Without his suggestions and dedication, the TLM of the CAM would have surely missed its delivery date and be replete with numerical errors.

Next, I would like to thank the other members of the Air Quality Modelling group at Carleton, namely Maryam Mirzajani, Farid Amid, Nicole McDonald and Seyed Morteza Mesbah, for their help with everything from chemistry to technical support, and for their encouragement and camaraderie.

I would also like to thank Dr. Yongtao Hu at the Georgia Institute of Technology for his patient help with SMOKE and CMAQ.

I am also very grateful to the Air Quality Modelling section at ENVIRONMENT CANADA. Dr. Mike Moran, Dr. Craig Stroud, Dr. Jack Chen and Balbir Pabla provided unique support that really enabled most of this research to be performed. I am indebted to them for their support and the opportunities they exposed me to, and look forward to continuing to work with them in the future.

I would also like to extend my thanks to the Air Quality Modelling team that was housed at the National Research Council at the beginning of this project; notably Dr. Jack Chen and Dr. Weimin Jiang. They provided the help I needed to kick-start the A UNIFIED REGIONAL AIR QUALITY MODELLING SYSTEM (AURAMS) development by showing me how to build the software on our local machines.

Thanks to Gray O'Byrne, Alex Trudel, Patrick Stewart, Thalia Russell, and John Marianne Daly for their invaluable external insight on my work, patiently allowing me to present dry runs of presentations to them and in the case of John, the technical support I would have been lost without.

Special thanks to my parents for their support of my schooling throughout the last few decades. My career path - and most notably existence - would have likely been far less fruitful and tangible had it not been for you.

And lastly, my warmest gratitudes to Janet Clark for her kindness, encouragement and divine initials that has pushed me through even the most difficult stages of this research period.

Contents

Abstract ii

Acknowledgments iii

List of Tables vi

List of Figures vii

List of Acronyms ix

1 Introduction 1

2 Background 5

2.1 Chemical Transport Models 5

2.1.1 CTM Processes 7

2.1.1.1 Transport Processes 9

2.1.1.2 Emission Injection 12

2.1.1.3 Chemistry 13

2.1.1.4 Aerosols 18

2.1.2 AURAMS 23

2.1.2.1 Features and Capabilities 24

2.2 Sensitivity Analysis 29

2.2.1 Global vs. Local Sensitivity Analysis 32

2.2.2 Forward vs. Backward Sensitivity 33

2.2.3 Formal vs. Perturbation-Based Sensitivity Methods 34

2.2.3.1 The Brute-Force Method 35

2.2.3.2 The Complex Variable Method 36

2.2.3.3 Tangent Linear Model/Decoupled Direct Method 38

2.2.3.4 Backward Sensitivity 39

2.3 Sensitivity Analysis using the TLM 39

2.3.1 Different Initial Condition and Boundary Condition Cases . . 44

2.3.2 Some Considerations on DDM Sensitivities 45

2.3.3 Higher-Order Sensitivity Analysis 47

2.4 TLM Code Generation Tools 48
2.4.1 TAPENADE 49
2.4.2 KINETIC PRE-PROCESSOR 53

3 Methodology **55**
3.1 AURAMS Installation and Modification 55
3.1.1 Building AURAMS 56
3.1.2 New Modules 59
3.1.3 Sensitivity 59
3.1.4 Physics Constants 60
3.1.5 Common Derivatives 60
3.1.6 Polynomials 60
3.1.7 Statement Function Replacements 61
3.2 Process Differentiation 61
3.2.1 Differentiating Advection 61
3.2.2 Differentiating Diffusion 64
3.2.3 Differentiating Emissions 66
3.2.4 Differentiating the Chemistry Solver 66
3.2.5 Differentiating the CANADIAN AEROSOL MODULE 67
3.3 TLM Evaluation Methods 72
3.3.1 Visualization of the Results 73
3.3.2 Result Management Tools 77

4 Results **81**
4.1 Model Setup 83
4.2 Evaluation of Gas Phase TLM 83
4.3 Evaluation of TLM implementation in the CANADIAN AEROSOL MODULE 99
4.4 Comparison of AURAMS and CMAQ Sensitivity Fields 105

5 Conclusion **110**

List of References **113**

A Sensitivity Module Documentation **119**
A.1 User Documentation 120
A.2 Developer Documentation 123
A.3 Appendix: Description of *sensitivities.namelist* 126

B Example of sensitivity to chemistry **129**

List of Tables

2.1 Species in the ADOM-II model gas-phase chemistry mechanism (Stockwell & Lurmann, 1989) 26
2.2 Aerosol size bins use in the CAM (Moran et al., 1998; Gong et al., 2003) 27
4.1 AURAMS vertical layer heights. 82
4.2 Summary of gas phase processes in AURAMS. 98

List of Figures

1.1 Nelson's Column during the Great Smog of December 1952 2
2.1 Process of Air Quality Modelling 6
2.2 Typical ozone isopleth 16
2.3 NO_x and VOLATILE ORGANIC COMPOUND (VOC) limited isopleths for regions in California 17
2.4 Average urban aerosol size distribution 19
2.5 Examples of different domain projections 25
2.6 Flow chart of CAM processes 28
2.7 Illustration of (left) forward vs (right) backwards sensitivities. 34
2.8 Illustration of BFM and TLM calculated sensitivity coefficients. . . . 35
2.9 Brute force and DDM sensitivity analysis of ozone response to emissions. 48
2.10 Architecture of TAPENADE 51
2.11 Example of TAPENADE differentiation 52
3.1 Effect of solving numerical precision issue with altering the order of operation 69
3.2 Impact of using post-convergence Newton procedure 71
3.3 Sample Sensitivity difference plot 75
3.4 Sample Radar Plot 76
3.5 Example of Results Management Web Tool: Runs 78
3.6 Example of Results Management Web Tool: Compare Runs 79
3.7 Example of Results Management Web Tool: Select Comparisons . . . 80
4.1 Ground level sensitivity of ozone concentration to initial ozone with chemistry after four simulation hours. 85
4.2 Sensitivity of ozone to NO POINT SOURCE (PS) emissions without transport after 4 simulated hours. 88
4.3 Sensitivity of ozone concentration to initial ozone with diffusion after four simulation hours. 89
4.4 Sensitivity of NO to NO PS emissions with diffusion after 4 simulated hours. 90
4.5 Analysis of weaker agreement between BFM and TLM sensitivities for FS emissions 92
4.5 Ozone sensitivity to NO FLUX SOURCE (FS) emissions, k=4 93
4.6 Ozone sensitivity to all NO emissions without advection after 2 hours of simulation. 95

4.7 Ground level sensitivity of ozone concentration to initial ozone with advection after two simulation hours. 96
4.8 Ozone sensitivity to NOx emissions with all processes (except aerosols) after 4 hours of simulation. 97
4.9 Sensitivity of all species to a perturbation in all species in the CAM . 101
4.10 TLM/CVM sensitivity for CAM after 1 time step 102
4.11 TLM/CVM sensitivity for CAM after 1 time step 102
4.12 TLM/CVM sensitivity for CAM after 3 time step 103
4.13 TLM/CVM sensitivity for NO_3 after 1 time step 103
4.14 TLM/CVM sensitivity for SO_4 after 1 time step 104
4.15 TLM/CVM sensitivity for SO_4 after 3 time steps 104
4.16 Example of artifacts resulting from imposing AURAMS results on the CMAQ grid 107
4.17 Comparing AURAMS and CMAQ output concentration 108
4.18 Comparing AURAMS and CMAQ output sensitivity 109

List of Acronyms

ADE Advection Diffusion Equation 8
ADOM Acid Deposition and Oxidant Model 24
AD Automatic Differentiation 49
AQHI Air Quality Health Index 2
AQM Air Quality Model ii
AURAMS A Unified Regional Air Quality Modelling System iii
BAQS-MET Border Air Quality Study - Meteorology 83
BC Black Carbon 18
BF Brute-Force 83
BEIS Biogenic Emissions Inventory System 26
BELD3 Biogenic Emissions Landcover Database 26
BFM Brute-Force Method 29
CAC Criteria Air Contaminants 2
CAM Canadian Aerosol Module iii
CAMx Comprehensive Air Quality Model with Extensions 40
CEPS Canadian Emissions Processing System 24
CHRONOS Canadian Hemispherical Regional Ozone and NO_x System 23
CMAQ Community Multiscale Air Quality 24
CTM Chemical Transport Model 4
CVM Complex Variable Method 36
DDM Decoupled Direct Method ii
EC Environment Canada 56
FS Flux Source vii
GCM Global Circulation Models

HETV	Vectorized Inorganic Heterogeneous Chemistry	29
KPP	Kinetic Pre-Processor	53
HDDM	Higher Order Decoupled Direct Method	41
GEM	Global Environmental Multiscale	83
MSC	Meteorological Service of Canada	23
NAPS	National Air Pollution Surveillance	2
NRC	National Research Council	56
PM	Particulate Matter	9
PS	Point Source	vii
RPO	Regional Planning Organization	106
SOA	Secondary Organic Aerosol	29
SMOKE	Sparse Matrix Operator Kernel Emissions	26
TLM	Tangent Linear Model	ii
TSP	Total Suspended Particulate	2
VOC	Volatile Organic Compound	vii

Chapter 1

Introduction

Clean air is not only a basic requirement for human health and well being, but a fundamental human right. Ethics asides, the sheer magnitude of mortality rates make air pollution a very serious issue. Recent epidemiological studies provide evidence that hundreds of thousands of premature deaths can be attributed to polluted air (World Health Organization, 2000).

Complaints about negative air quality date back to at least 1272 when King Edward I outlawed the burning of coal in London at the penalty of execution. In those times, no tools for sophisticated emissions monitoring and air quality assessment were available or even required to attribute the thick smog in the city to coal burning. This law was later repealed of course; London saw the construction of taller stacks, and a London laced with smog grew to be normal.

In 1952, a period of cold weather combined with an anticyclone and windless conditions led to a thick layer of smog to form over the city in what is today known as *The Great Smog of '52*, or *The Big Smoke*. Photographs (e.g. figure 1.1) taken during the event depict a thick fog reducing visibility to within meters. Recent estimates place the death toll of this event at 12,000 people (Bell & Davis, 2001).

Extreme episodes such as this, as well as more sustained adverse air quality, such

Figure 1.1: Nelson's Column during the Great Smog of December 1952. Photo courtesy by N T Stobbs.

as smog over Los Angeles, led to greater research into air quality in the mid 20^{th} century. Research showing the negative effects poor air quality has on human health (Basrur, 2000; Sandhu, 1998), agricultural productivity (Reich, 1987; Mauzerall & Wang, 2003; U.S. EPA, 2006), and the economic consequences of these were published and resulted in establishment of air quality and emission regulations.

In Canada, the 1960's saw the first attention to CRITERIA AIR CONTAMINANTS (CAC) with SO_2. Studies linking SO_2 to acidification of the aquatic system in Scandinavia and North America demonstrated that the long-range transport of SO_2 was indeed taking place and causing acid rain.

The 1970's also saw the beginning of TOTAL SUSPENDED PARTICULATE (TSP) sampling – defined in Canada as particles with an aerodynamic diameter under 100 μm – as part of the NATIONAL AIR POLLUTION SURVEILLANCE (NAPS) program (Pryor & Barthelmie, 1996). Different standards, regulations, and guidelines appeared over the years including the most recent development of the AIR QUALITY HEALTH INDEX (AQHI) founded to provide an air quality metric with consideration

for potential health effects. This was the first air quality index of its kind in the world (Stieb et al., 2008).

These standards and regulations all face the difficulties of proper measurement and/or estimation of atmospheric concentrations of pollutants. It is practically impossible to measure concentrations of all pollutants at all locations and all times. This is one of the main problems that saw the birth of AIR QUALITY MODELS (AQMs) to aid in the estimation and even forecasting of species of interest in the atmosphere. Beyond prediction of atmospheric concentrations (either forecast or hindcast), AQMs have important applications in air pollution decision support and policy making. Today, AQMs play an integral role in any policy developed concerning air quality.

Air quality managers are expected to bring ambient pollutant concentrations to levels that are deemed healthy for populations, the environment, and the climate. This is often a daunting task and involves many "how-to" or "what-if" or "how-much" questions which do not have easy answers. Questions such as:

- What is the impact of decommissioning a particular power plant on smog in Greater Toronto Area (what-if)?
- What is the most cost-effective approach to comply with the newly promulgated and more stringent air quality standard (how-to)?
- How much various local, provincial, national, or international sources contribute to the nationwide air pollution induced mortality (how-much)?

Nowadays it is a commonly accepted view that answering these questions would require using an AQM. However, a further look at these questions (or any air quality policy question for that matter) reveals that all these questions are intertwined with the concept of sensitivity. All the questions listed above seek to find or use the change in a concentration or a concentration-based metric as the result of change

in a parameter (in these cases, emissions) – by definition a sensitivity coefficient or derivative. It is not an exaggeration to claim that any air pollution decision making event is an exercise in sensitivity analysis, as any decision implies a change or contemplation of a change. Therefore, an informed decision requires quantification of the impact resulting from the change, which in itself is sensitivity analysis.

This thesis aims to implement a sensitivity analysis method into ENVIRONMENT CANADA'S AQM A UNIFIED REGIONAL AIR QUALITY MODELLING SYSTEM (AURAMS), and to evaluate the accuracy of the implementation. The thesis contains the following chapters apart from this one:

Chapter 2 contains a detailed introduction to CHEMICAL TRANSPORT MODELS (CTMs) and AURAMS in particular, and introduces sensitivity analysis and its variety of types and introduces and derives the TANGENT LINEAR MODEL (TLM) equations. The chapter includes discussions of non-linearity in atmospheric chemistry and describes the workings of certain key tools that were used in this research.

Chapter 3 contains a comprehensive description of the methodology and code modifications done to AURAMS, how the TLM was implemented into each process and evaluation methods used to validate the TLM implementation for both the gas phase and aerosol processes.

Chapter 4 is composed of the results of systematic tests performed on AURAMS in order to validate the TLM implementation. This chapter also contains comparisons of sensitivity coefficients computed with AURAMS to those calculated with CMAQ along with a discussion of the potential significance of this comparison.

Chapter 5 includes general concluding remarks and a discussion of how this work can be applied, how results from this work should be interpreted, and our recommendations on how this work should be extended in future.

Chapter 2

Background

As mentioned before atmospheric models play an integral role in various aspects of air quality management and science. Models used for this purpose are referred to as CHEMICAL TRANSPORT MODELS (CTMs) as they represent transport and transformation of pollutants in the atmosphere. Regional versions of CTMs are commonly called AQMs as their main use is in air quality applications. This research aims to implement a formal method for sensitivity analysis, called TLM or DDM, in a Canadian model, AURAMS. This chapter provides a brief background for this thesis. Readers will find a general description of CTMs and AURAMS (the CTM used in this research) in particular, an overview of sensitivity analysis methods, and a discussion of the method of choice for this research as applicable to atmospheric modelling.

2.1 Chemical Transport Models

Pollution in the atmosphere is a complex system that is heavily impacted by two processes: atmospheric dynamics and physical-chemical transformations. The former is addressed by models of atmospheric dynamics (meteorological models), and the latter by CTMs.

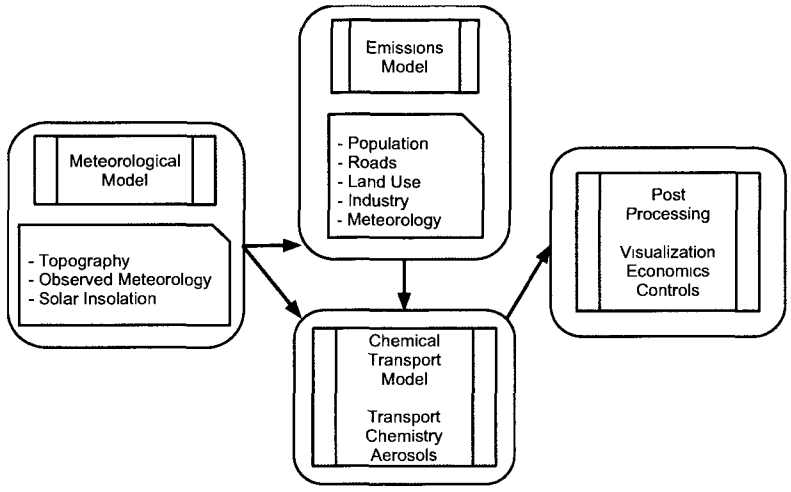


Figure 2.1: Conceptional diagram of an *Off-Line* AQM setup.

Meteorological models solve for variables such as wind speeds and direction, cloud cover, temperature, relative humidity and more. CTMs on the other hand investigate the evolution of chemical species in the atmosphere to provide concentration data that is relevant to human health, agricultural productivity or other endpoints. They accomplish this by using inputs from a variety of sources including those from meteorological and emission models to integrate mathematical representations of various processes in the atmosphere. These processes are described in section 2.1.1, and illustrated in figure 2.1.

Most CTMs have been developed using what is called an "*off-line*" approach that implements the assumption of one-way interaction of pollutants with meteorology. In the very common *off-line* approach, only meteorology has an effect on pollution formation, transformation, or transport while the impact from pollutant concentrations on meteorology is neglected. Having "decoupled" dynamics and chemistry models have provided modellers with a computational attractiveness for retrospective chemical transport simulations since many runs could be performed without re-simulating the meteorology. More importantly, coupled integration of dynamics and chemistry

at resolutions used in CTMs is a significantly more challenging task from a scientific and computational point of view. The limiting factors in the "off-line" approach, however, are that feedback responses between chemical and meteorological processes cannot be modelled, and spatial or temporal interpolation issues between dynamics and chemistry models can create mass inconsistencies in wind fields. Despite these drawbacks, the assumption of one-way interaction with meteorology is generally valid for episodic tropospheric regional-scale air quality modelling (Moran et al., 1998).

Various CTMs have been developed. These CTMs generally (with some exceptions) employ an Eulerian framework where the coordinates of the system is fixed (as opposed to a moving Lagrangian framework). CTMs can be further divided into global and regional categories. Global CTMs span the entire globe at a very coarse resolution and often cover, or are well extended into the stratosphere. Regional (or limited area) CTMs, on the other hand, cover a smaller region at higher resolution with a more limited vertical extent. A comprehensive - though dated - review of CTMs can be found in Russell & Dennis (2000). Here a brief overview of atmospheric processes modelled in CTMs is provided.

2.1.1 CTM Processes

CTMs have grown tremendously in their level of complexity over the last 3 decades. First CTMs mainly dealt with transport while disregarding atmospheric chemistry or considering a linear source-concentration relationship. Next generation of CTMs included atmospheric chemistry; however, inclusion of aerosols came years later. The third (current) generation of CTMs were developed with the concept of "One Atmosphere Modelling" in mind where all aspects of air pollution (gas-phase pollutants, aerosols, toxins, heavy metals, visibility, etc) would be considered within a single modelling framework. Next generation of CTMs appears to focus on on-line (or coupled)

dynamics-chemistry modelling. The general trend throughout the years, however, has been a growth in the level of sophistication of modelled processes and a tendency towards higher resolution in time, space, and species representation.

The evolution of pollutant concentrations in time and space in the atmosphere is, in general terms, governed by the ADVECTION DIFFUSION EQUATION (ADE):

$$\frac{\partial c_i}{\partial t} = -\nabla(\mathbf{u}c_i) + \nabla(\mathbf{K}\nabla c_i) + R_i\left(\mathbf{c}, T, t\right) + E_i(\tilde{x}, t) \tag{2.1}$$

where c_i is the grid cell average of concentration of species i, $\mathbf{u}$ is the three-dimensional wind field, $\mathbf{K}$ is the turbulent diffusivity tensor, E_i is the source/sink of species i at location $\tilde{x}$, and R_i is the chemical reaction rate of species i. R can also be a function of meteorological variables (e.g., temperature, T) (Russell & Dennis, 2000). For multi-phase (including aerosols) CTMs, R would also include aerosol processes such as nucleation, coagulation, condensation, and thermodynamics.

The ADE is solved subject to specific initial and boundary conditions. Lateral boundary conditions are taken from observations, chemical composition climatology, GLOBAL CIRCULATION MODELS (GCMs or global models of dynamics/chemistry), global CTMs, or coarse domain CTMs in nested configurations. Top boundary conditions are either climatologically prescribed or based on satellite observations or GCM simulations. Surface boundary conditions include dry deposition as well as the surface emission flux. The ADE is solved using the operator splitting method where the equation is integrated in pieces formed by splitting the ADE in processes and dimensions (when applicable.) Details of the operator splitting approach can be found elsewhere (McRae et al., 1982).

2.1.1.1 Transport Processes

CTMs model two transport mechanisms, *advection* and *diffusion*. Advection is defined as the transport of pollutants (and also air) due to bulk motion of the atmosphere along a force or pressure gradient, i.e., the wind. Diffusion is the natural tendency of a substance to move from an area (or volume) of high concentration to lower concentration. Diffusion in the atmosphere can happen due to random movement of molecules (molecular diffusion), shear forces between moving layers of air (shear dispersion), movement of air around physical obstacles (hydrodynamic dispersion), or buoyancy-induced production of eddies (atmospheric turbulence). Molecular diffusion is usually negligible (except at very high altitudes) compared to other forms of diffusion while other types are often called turbulence. Modelling of advection and diffusion in CTMs requires meteorological inputs.

Transport of the gaseous and PARTICULATE MATTER (PM) species in the atmospheric turbulent flow field is described by the first two terms in the ADE shown in (2.2). Considering the operator splitting scheme:

$$\frac{\partial c}{\partial t} = -u\frac{\partial c}{\partial t} + K\frac{\partial^2 c}{\partial x^2} \tag{2.2}$$

where u and K are now expressed as homogeneous and isotropic quantities in the 1-D equation for simplicity.

Different CTMs model transport processes differently. Advection and diffusion may be modelled together or separately. Horizontal transport may be performed simultaneously in both directions or transport may be split into three separate dimensional operators. Advective fluxes are typically orders of magnitude larger than diffusive fluxes in the horizontal directions while in the vertical direction the opposite is true. As such, some models may ignore horizontal diffusion (particularly in coarse

resolutions) or vertical advection in modelling of atmospheric transport. In short, various philosophical and numerical approaches are used in modelling of transport in CTMs with every implementation having its own benefits and pitfalls.

Integration of the advection process is a particularly challenging task from a numerical point of view. Criteria to heed when choosing an advection scheme include (Byun et al., 1999)

- Does the scheme conserve mass?
- How significant is the numerical diffusion in the algorithm?
- Does the scheme have a large phase error that will produce spurious oscillations?
- Given positive concentrations, will the scheme produce positive concentrations (positive-definiteness)?
- Is the scheme monotonic?
- Is the scheme linear or adequately representative of a linear process?[1]
- Is the scheme numerically stable?

Typically any scheme that is chosen satisfies some but not all of the attributes listed above. Tradeoffs of the attributes above stem in part from available computational efficiency, thus the question changes from "Which numerical scheme will produce the most accurate results?" to "Given the computational capabilities, development time, grid setup, and simulation period, which transport scheme can produce the most accurate results?" A number of specialized advection schemes have been developed for atmospheric transport modelling. Discussion and comparison of these algorithms is beyond the scope of this work but a summary can be found elsewhere (Odman, 1998).

[1]Because advection is a linear process, any implementation of an advection process must adequately represent that linearity. In this research, whether the implementation is linear heavily impacts the required effort for differentiation.

Vertical and (to a lesser extent) horizontal diffusion also play a key role in atmospheric transport. Due to the predominance of horizontal advection over horizontal diffusion, some models only implement vertical diffusion. In fact, in the early days of atmospheric modelling, horizontal diffusion was often ignored as the numerical diffusion associated with the advective algorithms was too large (Byun et al., 1999). Less diffusive numerical advection schemes and higher resolution grids are now available such that horizontal diffusion is now a meaningful process, but there is still a limited understanding of horizontal turbulence.

Because of the stochastic nature of turbulence, the equations used in CTMs to model atmospheric pollutant transport are averaged into a set of deterministic equations for average quantities (concentrations and wind) (Byun et al., 1999). This process (Reynolds decomposition and averaging) involves averaging the ADE that is constructed from decomposed velocities and concentrations into mean and random (turbulent) components. Averaging the turbulent terms produces Reynolds Flux terms, which in turn generates a new problem set in which there are more unknowns than equations. This is known as the "*closure problem*". Fundamentally, the closure problem is caused by trying to represent a non-linear, stochastic process with a linear decomposition such as Reynolds decomposition.

Most CTMs use a local approach to overcome the closure problem where turbulence is linked to known local quantities such as wind, temperature, humidity and other parameters. The most commonly used local closure approach in CTMs is *gradient transport theory* or *K-theory*. In this approach the non-linear component in Reynolds decomposition is parameterized as an idealized Fickian (turbulent) diffusion. *K-theory*, despite its wide spread use, has its own limitations as it does not account for counter-gradient turbulent transport and it could fail when large-scale eddies are present (particularly at higher resolution applications (Stull, 1988).)

In any transport process, mass consistency is a fundamental requirement. Some of the advection schemes used in CTMs do not conserve mass. Also other processes modelled in a CTM may cause mass inconsistencies. As a result, CTMs often use an external subroutine to impose mass conservation. One common method of ensuring mass conservation is for the CTM to operate on a passive primary trace species the mass (concentrations) of which is adjusted at each time step. Using this method species can undergo mass conservation algorithms on either a domain-wide or grid cell basis. Byun (1999) describes other methods, ranging from adjusting wind fields prior to advection or adjusting mass after advection.

2.1.1.2 Emission Injection

CTMs are mainly driven/forced by continuous or intermittent emissions. In other words, emissions are possibly the most influential inputs into CTMs. Various CTMs process emissions differently. While most models use pre-processed emission inputs (out of emission models), some other have an in-line (or partially in-line) emission processing system where emissions (or some emissions) or the location and quantity of their injection into the domain is calculated during the simulation time.

Emissions typically considered in CTMs include mobile sources (on-road, such as cars or off-road such as agricultural machines and lawn mowers), area sources (e.g. residential heating), point sources (e.g. power plants or refineries), and biogenic sources (VOCs from trees.) Some emission models consider a more comprehensive range of natural or anthropogenic emission sources such as wildfires and biomass burning (e.g. agricultural prescribed burning), lightning, see salt, shipping, aviation, etc. Various types of sources can be significant for specific pollutants, for example ozone producing NO_x emissions are mainly generated in combustion processes in power plants and mobile sources while VOCs can have strong regional dependence on

biogenic sources. CTMs usually receive emissions in two forms, as an injection into a cell above the surface layer (point sources, wildfires, lightning, etc), or as a boundary flux (typically in the diffusion subroutine) at the surface (i.e. area and mobile sources). The specific implementation of emission injection varies from model to model, but in general they are input from gridded emission files generated in an emission model, and injected overtime into the concentration field during simulations.

2.1.1.3 Chemistry

Thousands of reactive compounds exist in the atmosphere that participate in tens or hundreds of thousands of possible reactions. Representing a detailed chemistry in CTMs is a near impossible task. All CTMs use a simplified representation of atmospheric chemistry where only reactions that are deemed most important are included. The collection of atmospheric reactions that are considered representative for various purposes is called a chemical mechanism. Various chemical mechanisms are generally similar in representation of inorganic reactions but can differ significantly in their organic species and reactions.

The focus of gas-phase chemistry has historically been on modelling and accurate prediction of tropospheric ozone (and mainly ground-level ozone.) Ozone (O_3) is a secondary pollutant in the troposphere which indicates that it is not emitted into the atmosphere but is chemically formed from other species. Ozone is formed from oxides of nitrogen (NO_x) and organic compounds (i.e. VOCs). Ozone forming reactions between VOCs and NO_x requires the presence of radicals, the production of which is often initiated by sunlight. Therefore, chemical reactions that lead to ozone production are referred to as photochemistry.

Ozone and NO_x are generally at an equilibrium through the NO_x cycle:

$$NO_2 + O_2 \xrightarrow{h\nu} NO + O_3 \quad \text{(R1)}$$

$$O_3 + NO \longrightarrow NO_2 + O_2 \quad \text{(R2)}$$

These three reactions (the first reaction is a combination of two reactions) define a null cycle because there is no net production or consumption of ozone. This cycle operates by creating a temporary ozone molecule in (R1), and then consuming it in (R2) within approximately few minutes (Jacob, 1999). (R2) happens at all times but (R1) requires sunlight[1] and therefore only occurs during the daytime. Enhanced ozone production observed in many areas cannot be explained by the NO_x cycle. High levels of ozone concentrations observed require reactions with intermediary species such as hydroxyl (OH) and peroxy (HO_2 and RO_2) radicals. OH is also produced photochemically in the atmosphere by photolyzing ozone in the presence of water vapour (R3-R5):

$$O_3 \xrightarrow{h\nu} O(^1D) + O_2 \quad \text{(R3)}$$

$$O(^1D) + M \longrightarrow O + M \quad \text{(R4)}$$

$$O(^1D) + H_2O \longrightarrow 2\,OH \quad \text{(R5)}$$

In the presence of VOCs (shown as RH), OH reacts with VOCs to produce peroxy radical RO_2,

$$OH + RH \xrightarrow{O_2} RO_2 + H_2O \quad \text{(R6)}$$

[1]Specifically, in the presence of light at wavelengths between λ = [300 nm, 320 nm], shorter wavelengths are typically attenuated before they reach the troposphere.

When the RO_2 (or HO_2) radical is present, NO to NO_2 conversion in reaction (R2) takes place through an alternative path that does not consume ozone.

$$RO_2 + NO \longrightarrow NO_2 + RO \tag{R7}$$

Reaction (R7) (combined with NO_x cycle) represent the main source of daytime ozone generation in the troposphere. Another way to explain (R7) as a production reaction is to note that NO_2 goes on to produce O_3 through (R1).

At night, without the presence of light, reactions (R1) and (R3) do not take place and NO_x usually exists exclusively as NO_2 as a result of O_3 titration by NO. NO_2 can then react with O_3 to produce NO_3 (R8) which in turn will be removed from the gas-phase chemistry as HNO_3 (R9-R10).

$$NO_2 + O_3 \longrightarrow NO_3 + O_2 \tag{R8}$$

$$NO_3 + NO_2 \xrightarrow{M} N_2O_5 \tag{R9}$$

$$N_2O_5 + H_2O \xrightarrow{\text{aerosol}} 2\,HNO_3 \tag{R10}$$

Given a very large NO_x concentration, the impact of reaction (R7) is lessened due to OH scavenging by NO_2 in reaction (R11),

$$OH + NO_2 \xrightarrow{M} HNO_3 \tag{R11}$$

The impact of (R11) on ozone production is often illustrated by the ozone isopleth (figure 2.2).

The centre diagonal line in figure 2.2 is called the ozone ridge. Under this line is a NO_x-limited regime that is typical of rural areas due to high VOC emissions and distance from urban NO_x emissions. In this region production of ozone through

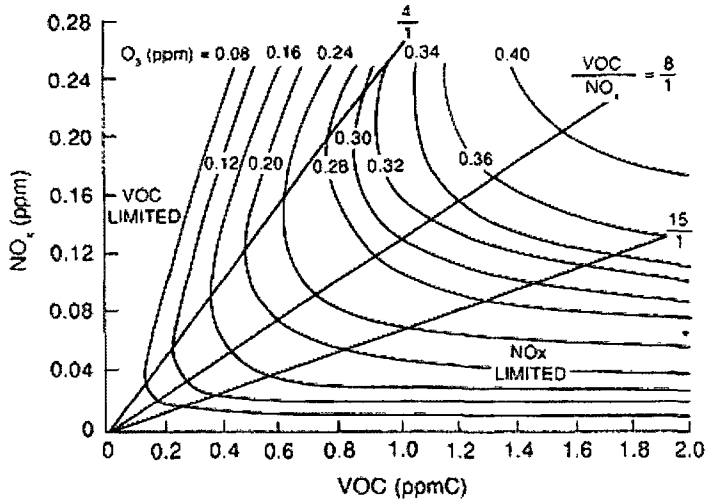
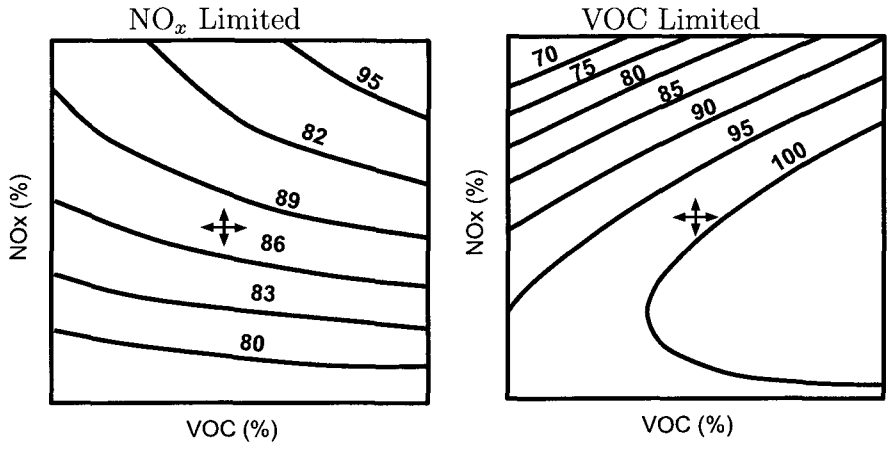


Figure 2.2: A typical ozone isopleth. The NO_x-limited region is typical of rural areas, whereas the VOC-limited region is typical of highly polluted urban areas (Seinfeld et al., 1991)

(R7) is controlled by availability of NO_x (i.e. NO_2.) The area above the ridge line is the VOC-limited, radical-limited, or NO_x-inhibited regime that is typical of urban areas. In these region, NO_x and VOC compete for OH through reactions (R11) and (R6), respectively. While reaction (R6) results in propagation of radicals and production of more ozone, reaction (R6) removes radicals from the system and inhibits ozone production. Therefore, in NO_x-rich regions ozone production through (R7) is controlled by availability of radicals (hence the name radical-limited) and is inhibited by NO_x (hence NO_x-inhibited). This non-linear behaviour of ozone with respect to NO_x is the prime example of chemical non-linearity in the atmosphere and can easily be seen in the isopleth. While NO_x reduction in NO_x-limited regime results in decreased ozone, the opposite is true in the NO_x-inhibited regime. In other words, sensitivity of ozone with respect to NO_x availability/emissions is positive in NO_x-limited region but negative in the NO_x-inhibited regime. Similarly, ozone sensitivity with respect to NO_x is close to zero around the ozone ridge which is typically the predominant regime in regions of maximum ozone concentrations. In these high



(a) upwind of Fresno

(b) downwind of the San Francisco Bay area

Figure 2.3: The crosshairs in the centre of the figures denotes 0% perturbation in both NO_x and VOC. (a) shows a NO_x-limited regime upwind of Fresno, California, (b) shows a VOC-limited region downwind of the San Francisco Bay area. Adapted from Hakami (2003)

concentration areas ozone exhibits the highest degree of non-linearity (Hakami et al., 2004).

The VOC to NO_x ratio that defines the ridge (border between the opposing regimes) is very dependent on the VOC composition and environmental factors, and as such, differs significantly from one location to another or in time. Therefore, values shown in figure 2.2 are only examples. While the isopleth for a location may appear different for different times, in general, they do help characterize whether a region is typically NO_x-limited or NO_x-inhibited. Figure 2.3 for example shows isopleths for Fresno and San Francisco, California, where one region is NO_x limited and one VOC limited, respectively. This information is valuable in developing effective policies for reducing ozone concentrations through control of VOC and/or NO_x emissions.

Integrating the system of chemical ODEs (third term in the ADE) is a challenging task and one of the main computational costs in CTMs. The challenge stems from the fact that various species in chemical mechanisms have characteristic times (an

indication of their lifetime) that differ by many orders of magnitude. This results in a very stiff system of ODEs that can only be solved efficiently by specialized solvers. The discussion of many specialized chemistry solvers can be found elsewhere (Sandu et al., 1997).

2.1.1.4 Aerosols

Aerosols play an important role in air quality with regard to human health. Increases in ambient aerosol concentrations have been associated with many adverse health effects including respiratory irritation, decrease in level of lung functionality, and short-term and long-term mortality (Brook et al., 2002; Dominici et al., 2006; Vedal, 1997).

Aerosols also have significant influence on short term weather and long term climate. For instance, in the presence of aerosols, clouds may more easily form. Moreover, the size of cloud droplets are also indirectly influenced by aerosols. This influence on cloud behaviour alters the cloud's reflectivity, probability of precipitation and lifetime which in turn has a significant impact on the Earth's albedo and regional or global weather (Pandis et al., 1991).

Other than the size and concentration of aerosol populations, the chemical composition is also an important factor to consider. Sulphate aerosol species, for example, reflect sunlight and cause a negative forcing on the Earth's radiative budget. BLACK CARBON (BC) aerosols on the other hand absorb sunlight and cause local heating in the troposphere by reducing cloud particle and snow albedo (Menon et al., 2002; Hansen et al., 2000) and absorption of the incoming solar radiation. Significant fraction of aerosols have natural origin. However, anthropogenically emitted or produced aerosols are of particular importance as they can be more harmful to health, and also because they are more concentrated in populated areas.

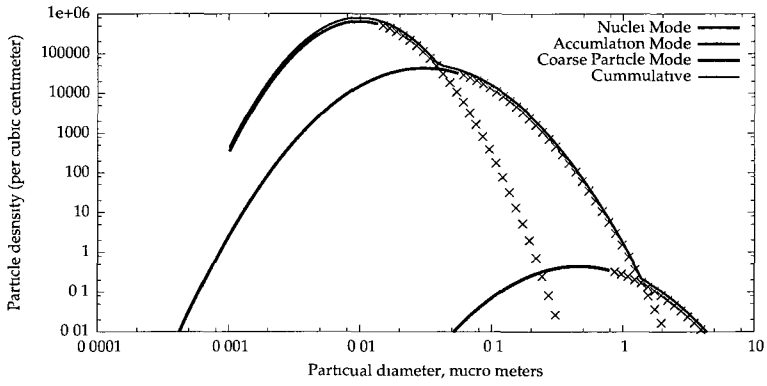


Figure 2.4: Average urban aerosol size distribution on a log-log graph. The combined distribution represented by the black line masks the contribution of the individual modes. (Hinds, 1999)

Because of the growth and removal processes intrinsic to aerosols, size distribution of urban aerosol is complex. A common way to represent aerosols is in terms of three modes, the *nucleation*, *accumulation*, and *coarse* particle modes. Figure 2.4 shows a typical urban aerosol size distribution.

As figure 2.4 shows, nucleation aerosols are the smallest mode. This mode primarily consists of particles emitted from combustion processes and from particles formed in the atmosphere from gas-to-particle conversion. Because of their high number concentration, especially close to their source, these aerosols are very short lived as they rapidly coagulate with each other and with particles in the accumulation mode (Hinds, 1999). It is particles in this size range that serve as sites for the formation of cloud droplets.

The accumulation mode is the next mode, and combined with the nuclei mode these modes comprise "fine" particles. Aerosols in this mode typically have a higher lifetime as the particles are too large to quickly coagulate into the next mode, but small enough that removal mechanisms such as settling and rain-out are weak. Included in the size range of accumulation particles is the wavelength of visible light and thus it is these particles that account for most of the visibility effects and climate

impacts of atmospheric aerosols (Hinds, 1999).

Coarse mode particles consist of wind blown dust, large sea salt, and mechanically generated anthropogenic particles (e.g. from agriculture, surface mining or construction.) Because of the large size of these particles, they rapidly settle out of the atmosphere or impact on surfaces within hours or days.

The distinction in these modes is important as each mode has its sources and is subject to certain removal processes. Furthermore, the chemical composition and health effects vary greatly from mode to mode. Because of the large size difference between fine and coarse mode particles, they exist together in the atmosphere as two somewhat chemically distinct aerosol populations. Fine particles are acidic and contain most of the sulphates, ammonium compounds, hydrocarbons, elemental carbon (soot), toxic metals and water in the atmosphere. The coarse particles on the other hand contain most of the crustal materials and their oxides, such as silicon, iron, calcium, and aluminum, as well as large sea salt particles and vegetation debris (Hinds, 1999).

Much work in the last few decades on AQMs has been done to better model atmospheric aerosol concentrations and processes. Different models have been produced using different assumptions, methods and implementations. The main differences are:

Modal vs Sectional Representation

There are two ways that CTMs commonly represent aerosols, these are known as the *modal* and *sectional* representations.

The modal approach considers several distinct sub-populations of aerosols called modes (figure 2.4). These modes are determined by the size of the aerosols. Each of these modes is represented by an analytical modal (typically lognormal) distribution

function. These distributions can be uniquely characterized by three parameters: total number concentration, geometric mean size of the number moment, and geometric standard deviation. Because it is often possible to realistically represent atmospheric aerosols with only a few modes, the modal approach is usually computationally fast to process (Moran et al., 1998). This approach also does not suffer form numerical diffusion across section boundaries as it does with the sectional approach discussed next.

The sectional method represents aerosol size distributions by a set of contiguous non-overlapping discrete size bins. The computational complexity of this method depends on the size and number of each bin and is often greater than that of the modal approach. Gong et al. (2003) found that 18 bins should be used in order to resolve major nucleation (gas-to-particle conversion) events. Though this method is typically more computationally costly than the modal approach, it provides a greater flexibility with fewer assumptions.

Internally vs Externally Mixed

Atmospheric aerosols species seldom consist exclusively of an individual component, instead they generally consist of a mixture of species. When modelling aerosol interactions, one can consider an aerosol population as *internally mixed*, *externally mixed*, or anywhere in between.

Internal mixtures assume that all particles of a given size contain a uniform mixture of components from each source (Seinfeld & Pandis, 2006; Moran et al., 1998). External mixtures represent the other extreme of this definition where each particle of a given size arises from only one source.

The mixing state of aerosols has a considerable impact on their behaviour with respect to radiative effects. This is not however to imply that the internally mixed

assumption is necessarily unrealistic. Winkler (1973) listed four atmospheric processes that promote higher degree of aerosol internal mixing:

- condensation/absorption of gases by aerosol particles;
- coagulation of aerosol particles;
- aqueous-phase chemistry within cloud droplets followed by droplet evaporation;
- coagulation of cloud droplets or coalescence of aerosol particles and cloud droplets followed by droplet evaporation.

This suggests that the degree of internal mixing of an aerosol should increase with time as the aerosol ages (Moran et al., 1998).

Nevertheless, there are a number of models that do not apply an internally mixed assumption in order to improve prediction of aerosol properties. Not using an internally mixed aerosol assumption raises the computational complexity. For instance, modelling the chemical composition of a size-distributed internally-mixed aerosol consisting of n_p chemical constituents requires n_p chemical fields which are functions of one independent variable, and particle size, whereas modelling an incompletely-mixed aerosol requires a separate field for each pure chemical species or combination of species in each range (Wexler et al., 1994; Kourti & Schatz, 1998). For example, Jacobson (1994) shows that coagulation of 11 distinct particle types can produce 2047 conglomerations.

Choice of aerosol species to model

Any given aerosol population or even individual particle may be composed of a wide range of different chemical components. To better work with the very diverse range of possible chemical composition, aerosol populations are generally considered to be a member of a broader class of aerosol species which best describes it. These classes

include but are not limited to $SO_4^=$, NO_3^-, NH_4^+, EC (elemental carbon), primary organic carbon PC and secondary organic carbon OC (this might be further divided into anthropogenic OC and biogenic OC), sea salt, crustal materials (e.g., soils and dust), H_2O (Moran et al., 1998), Na^+, Cl^- and *other*.

The choice of which aerosol species to consider in a model depends on the geography of the region (e.g. is the region near a large body of salt water), the goal of the model (e.g. aerosol mass prediction, radiative effects, species relevant to human health, etc), seasonal aspects (e.g. salting the roads in winter) and land use. CTMs often model a number of aerosol-related processes including, but not limited to, nucleation, coagulation, evaporation/condensation, thermodynamics (equilibrium between gas- and liquid-phase concentrations), cloud activation (formation of cloud droplets on aerosols), aqueous chemistry, heterogeneous chemistry, etc. Again, discussion of these processes and their detailed treatment in CTMs is beyond this work and can be found elsewhere (Zhang et al., 2002).

2.1.2 AURAMS

The METEOROLOGICAL SERVICE OF CANADA (MSC) developed a size and composition resolved, episodic, regional particulate matter modelling system named "A Unified Regional Air Quality Modelling System", or AURAMS. This model was branched out of their previously developed CANADIAN HEMISPHERICAL REGIONAL OZONE AND NO_x SYSTEM (CHRONOS) CTM, but despite sharing a number of model components, their development since has been largely independent.

The intent of the AURAMS model is to better understand PM and other regional pollutants in North America, and especially in Canada (Zhang et al., 2002). AURAMS is capable of assessing the impact of emission reduction scenarios separately or simultaneously for PM, ground-level ozone, acidic deposition, and air toxins

(Zhang et al., 2002). This model makes use of four existing MSC air-quality related models or modules as "building blocks." These existing models include:

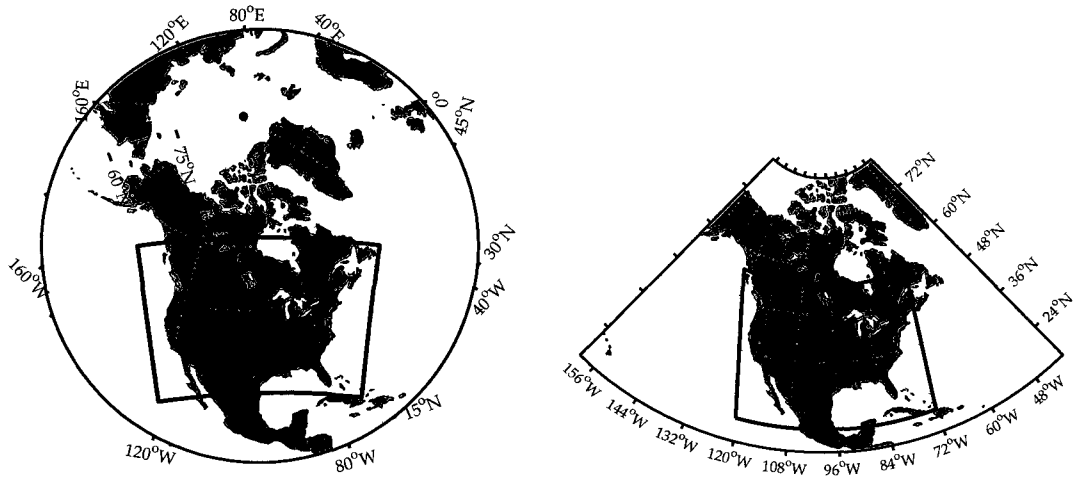
- the ACID DEPOSITION AND OXIDANT MODEL (ADOM)
- a comprehensive episodic regional acid deposition model (Venkatram et al., 1988)
- the CAM, a composition - and size-resolved aerosol module that has already been used within both a global and a regional climate model (Gong et al., 2003)
- the CANADIAN EMISSIONS PROCESSING SYSTEM (CEPS), an emissions modelling system for regional-scale air quality models that is needed to create emissions input files (Moran et al., 1998)

Typically AURAMS employs a secant polar stereographic grid projection that encompasses most of Canada and the US with a true north of 60° and various grid resolutions (42 km at continental scale). This projection is used because it provides good resolution in northern regions (see figure 2.5.) This is in contrast to the Lambert grid typically used in other models (e.g. US EPA's COMMUNITY MULTISCALE AIR QUALITY (CMAQ)) that provides better resolution at mid level latitudes.

2.1.2.1 Features and Capabilities

A conceptual design for AURAMS is laid out in (Moran et al., 1998), this section is not meant to duplicate that document, but to familiarize the reader with a general overview of AURAMS. As such, for details on assumptions and processes the reader is referred to Moran et al. (1998).

The transport processes contained in AURAMS include a 3D non-oscillatory positive definite semi-Lagrangian scheme (Pudykiewicz et al., 1997) and a vertical column



(a) Polar Secant Stereographic Projection of the domain used in AURAMS

(b) Lambert Projection of the RPO unified continental 36 km domain

Figure 2.5: Examples of different domains used in air quality modelling. a) Polar Stereographic domain common in AURAMS. b) Lambert domain commonly used in CMAQ. These domains are projections of a rectangular grid, i.e. the top and bottom boundaries are the same lengths, as are the left and right boundaries. Notice then how in (a) the northern boundary of the domain is stretched, and how the bottom boundary (b) is stretched out.

Table 2.1: Species in the ADOM-II model gas-phase chemistry mechanism (Stockwell & Lurmann, 1989)

1 **SO_2**	11 **ETHE**	21 **CRES**	31 $O^{(1D)}$	40 **CRG_1**
2 **SO_4**	12 **ALKE**	22 **HONO**	32 $O^{(3P)}$	41 **CRG_2**
3 **NO**	13 **TOLU**	23 **RNO_3**	33 **NO_3**	42 **CH_4**
4 **NO_2**	14 **AROM**	24 **ISOP**	34 **N_2O_5**	43 **C_2H_6**
5 **O_3**	15 **HCHO**	25 **HO_2**	35 **HNO_4**	44 **H_2O**
6 **H_2O_2**	16 **ALD2**	26 **RO_2**	36 **OH**	45 **O_2**
7 **HNO_3**	17 **MEK**	27 **MCO_3**	37 **RO_2R**	46 **M**
8 **PAN**	18 **MGLY**	28 **CO**	38 **R_2O_2**	
9 **C_3H_8**	19 **DIAL**	29 **NH_3**	39 **RO_2N**	
10 **ALKA**	20 **ROOH**	30 **DUST**	40 **BZO**	

The first 29 species (bolded) species are advected by the CTM, species 30 to 41 are stored but not advected, and the remaining 6 (42-46) are held constant.

Crank-Nicolson (1.5 order) diffusion operator. The chemistry mechanism is ADOM-II, see table 2.1 for a list of species used by the mechanism. The model is designed to run at various grid spacings, typically between 2.5 and 42 km.

Emissions are pre-processed in SPARSE MATRIX OPERATOR KERNEL EMISSIONS (SMOKE) (Houyoux et al., 2000). This model outputs gridded speciated hourly anthropogenic emission files including 18 model gas-phase species and two bulk PM species. Biogenic emissions of model gas-phase species are processed on-line using BIOGENIC EMISSIONS INVENTORY SYSTEM (BEIS) v3.09 and the BIOGENIC EMISSIONS LANDCOVER DATABASE (BELD3) vegetation database (Moran, 2010).

To process aerosols, AURAMS includes the CANADIAN AEROSOL MODULE (CAM). This module employs a sectional approach (discussed in section 2.1.1.4) for 9 species with 12 bins (see table 2.2) per species. The species are SO_4, NO_3, NH_4,

Table 2.2: Aerosol size bins use in the CAM (Moran et al., 1998; Gong et al., 2003)

	Diameter (μm)	
Bin	Lower Edge	Upper Edge
1	0.01	0.02
2	0.02	0.04
3	0.04	0.08
4	0.08	0.16
5	0.16	0.32
6	0.32	0.64
7	0.64	1.28
8	1.28	2.56
9	2.56	5.12
10	5.12	10.24
11	10.24	20.48
12	20.48	40.96

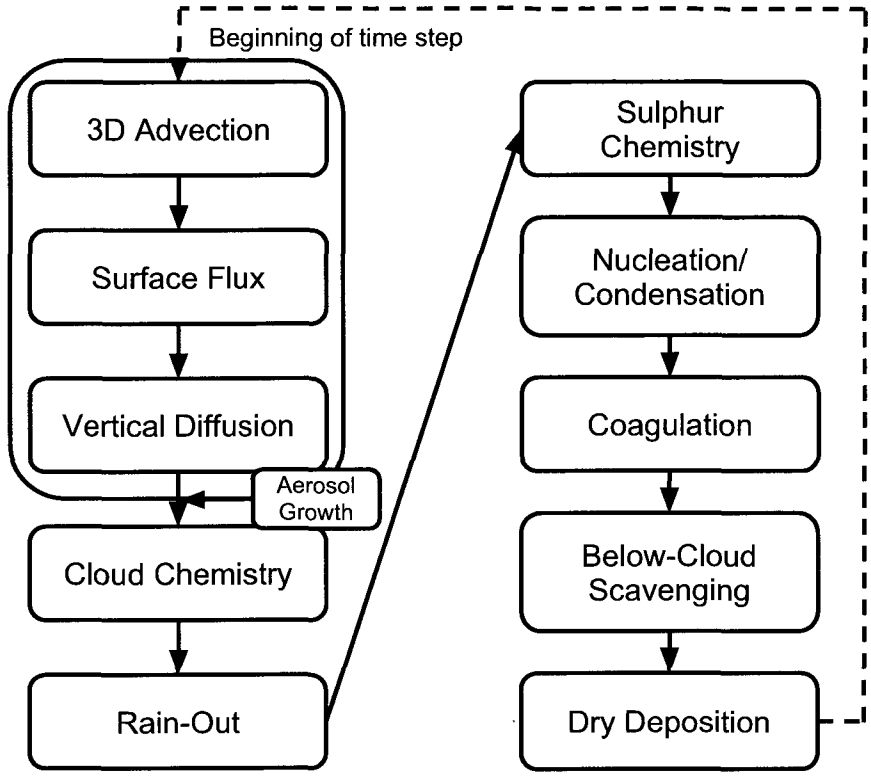


Figure 2.6: Flow chart of the processes that make up the CAM. The blue area denotes processes that are performed in the host model. Figure adapted from Gong et al. (2003)

elemental carbon, particulate organic carbon (primary and secondary), crustal material, dust and H_2O. Within each size bin, an internally mixed (discussed in section 2.1.1.4) aerosol is assumed for all types of aerosol except for freshly emitted insoluble components (i.e. BC and soil dust) which are assumed to be externally mixed for a fixed amount of time (e.g. one time step) (Gong et al., 2003).

Figure 2.6 illustrates the processes involved in the CAM. Transport and emission (surface flux) processes contained in the blue area in said figure are performed in the

host model (i.e. AURAMS.) The heterogeneous chemistry solver in AURAMS is VECTORIZED INORGANIC HETEROGENEOUS CHEMISTRY (HETV) based on the ISORROPIA solver (Makar et al., 2003). SECONDARY ORGANIC AEROSOL (SOA) parameterization is based on Jiang (2003).

2.2 Sensitivity Analysis

AQMs require numerous input parameters. It is often of scientific or policy importance to evaluate the impact these parameters have on model outputs. In particular the influence of emissions on predicted concentrations is of great interest to air quality decision makers. Evaluation of these influences (called sensitivities, derivatives, sensitivity coefficients, etc) is the subject of sensitivity analysis. Traditional approach to sensitivity analysis is the one-at-a-time perturbation to model parameters, i.e. the BRUTE-FORCE METHOD (BFM). The BFM becomes cumbersome when large numbers of sensitivity parameters are to be considered. The BFM may also produce inaccurate sensitivities due to numerical and round-off errors that are inherent to integration of AQMs. Finally, brute-force derivatives may not truly represent the local sensitivity (i.e. slope) of the response surface. Formal methods for sensitivity analysis in AQMs that attempt to solve sensitivity equations (rather than concentration equations) have proven useful over the past two decades. Sensitivity analysis in atmospheric modelling has been used extensively for air quality management and pollution control strategy design, uncertainty analysis, forecasting, inverse modelling of emissions, etc.

Though this thesis focuses on sensitivity analysis in AQMs, sensitivity analysis has been used in many fields of research; for example, Glarborg et al. (1986) used sensitivity analysis to better understand the chemical kinetics of nitrogen oxide in

well-stirred reactions; Leamer (1985) discusses the use of sensitivity analysis in economic studies; and Oh et al. (1993) used sensitivity analysis to evaluate the effect of converter volume in a vehicle for cold-start emissions.

In AQMs, sensitivity analysis may be used to answer questions along the following lines:

- how do emissions drive secondary pollutants?
- what are the temporal and spatial effects of emissions on a region?
- how do emissions from different jurisdictions affect one another?
- what is the effect of chimney stack height on local and non-local surface concentration of secondary pollutants?
- how does the choice of boundary conditions for the model domain affect concentrations; e.g. does a static or dynamic boundary condition for a domain that spans from the Atlantic to the Pacific have any effect on interior ozone concentrations?

Sensitivity results can have a significant impact on policy decisions with regard to emission controls strategies. For example, Sillman et al. (2003) identified species indicator ratios which signal whether ozone formed under NO_x or VOC-limited conditions in Paris. Based on their analysis, they concluded that Paris is often very close to the transition from NO_x-sensitive to VOC-sensitive chemistry. In another example, Hakami et al. (2004) showed that projection from linear response to emission reductions in California are most inaccurate for peak concentrations due to inherent non-linearity these locations experience.

Sensitivity analysis methods better equip scientists and policy makers to asses the effectiveness of control strategies, and allow them to study whether the issues at hand may be appropriately addressed as local or if long-range transport issues and secondary products must be considered. Beyond policy applications, sensitivity

analysis is also used in inverse modelling of emissions or other uncertain parameters, data assimilation, air quality forecasting, and uncertainty analysis.

Sensitivity Coefficients

The process of analyzing sensitivities involves calculating *sensitivity coefficients*. Sensitivity coefficients are the partial derivative of the system output to a parameter of interest. These coefficients can also be referred to as gradients if a scalar output is differentiated with respect to a parameter vector.

Sensitivity coefficients can be represented as,

$$S(t,p) = \frac{\partial c(t,p)}{\partial p} \tag{2.3}$$

where $c(t,p)$ is the model output (concentration), p is the parameter of interest (e.g. emission of a specific species), and $S(t,p)$ is the sensitivity of $c(t,p)$ with respect to p.

The sensitivity coefficient defined above is absolute or non-normalized. The magnitude of model parameters and inputs can vary significantly in time and space, and so can the sensitivity coefficients. Therefore it is often useful to define a *semi-normalized* sensitivity coefficient. Given a model parameter $p(x,t) = \epsilon P(x,t)$ where $P(x,t)$ is the unperturbed field and ϵ is a scaling variable with a nominal value of 1, one can calculate the sensitivity coefficient s as the partial derivative of the concentration c to the scaling variable ϵ (Yang et al., 1997).

$$s(t,p) = P \times \frac{\partial c(t,p)}{\partial p} = P \times \frac{\partial c(t,p)}{\partial (\epsilon P)} = \frac{\partial c(t,p)}{\partial \epsilon} \tag{2.4}$$

The semi-normalized sensitivity coefficient described in (2.4) has the same units

as the concentration $c(t,p)$. The semi-normalized sensitivity coefficient indicates sensitivity to fractional change (scaled to full fraction or 100%). For example, a 70 ppb semi-normalized sensitivity of ozone with respect to NO emissions indicates that 100% reduction in NO emissions would result in 70 ppb reduction in ozone concentrations. Since TLM sensitivities are local (i.e. linearized models or slopes), this value can be scaled down to smaller perturbations. In other words, the sensitivity coefficient mentioned before also indicates a 7 (or 0.7) ppb reduction in ozone concentration as a result of 10% (or 1%) reduction in NO emissions.

One can also calculate the (fully) normalized sensitivity coefficient,

$$\hat{s}(t,p) = \frac{p}{c} \times \frac{\partial c(t,p)}{\partial p} \tag{2.5}$$

The normalized sensitivity coefficients are dimensionless and provide a fractional sensitivity of concentrations with respect to fractional changes in parameters (emissions). For example, a normalized sensitivity of ozone to NO emissions of 0.4 implies 40% change in ozone concentration as a result of 100% change in NO emissions. Again, these sensitivities are local and can be scaled down to smaller changes.

2.2.1 Global vs. Local Sensitivity Analysis

In general, sensitivity analysis methods can be categorized into *local* and *global* approaches. Rabitz et al. (1983) provides the following example for explaining the differences in local and global sensitivity methods. Let us define the temporal chemistry system,

$$\frac{\mathrm{d}c}{\mathrm{d}t} = f\left(c,p,t\right) \qquad c(0) = C^o \tag{2.6}$$

where c is the species concentration, p is a parameter, and t (time) is the variable of integration. Local methods of calculating sensitivity coefficients evaluate the partial

derivatives, $\partial c(t)/\partial p$, $\partial^2 c(t)/\partial p^2$, etc, about the nominal parameter value solution of (2.6). These sensitivities can be thought of as the gradients and curvatures about the nominal solution in a multidimensional parameter space.

Global methods, on the other hand, consider *a priori* the range and/or statistical properties of the parameter uncertainty, and calculate certain averaged sensitivities, e.g. $\langle \partial c(t)/\partial p \rangle$, over the region of parameter uncertainty (Rabitz et al., 1983). These coefficients quantify the effect of large simultaneous variation in p, and are thus qualitatively different than the local sensitivities.

Local methods typically provide a higher level of detail, while global methods are typically best for handling large variations in the system parameters (Rabitz et al., 1983). The choice of sensitivity methodology is often dictated by the characteristics of the particular system, the parameters under consideration (and their number), and computational considerations.

2.2.2 Forward vs. Backward Sensitivity

Local methods for sensitivity analysis can also be categorized into forward or backward sensitivity (*adjoint* sensitivity) methods.

For clarity let,

$$\tilde{\mathbf{y}} = L(\tilde{\mathbf{x}}) \tag{2.7}$$

where $\tilde{\mathbf{y}}$ is the output vector, $\tilde{\mathbf{x}}$ is the input vector, and L is a non-linear operator. Forward sensitivity can be thought of as calculating the sensitivity of all outputs with respect to a single input, or $S_i = d\tilde{\mathbf{y}}/dx_i$ where i is a parameter index. On the other hand, backward or adjoint sensitivity may be thought of as calculating the effect of all inputs on one output, or $S_i = dy_j/d\tilde{\mathbf{x}}$. In other words, forward sensitivities calculate how one source affects all receptors while backward sensitivity calculates the influence

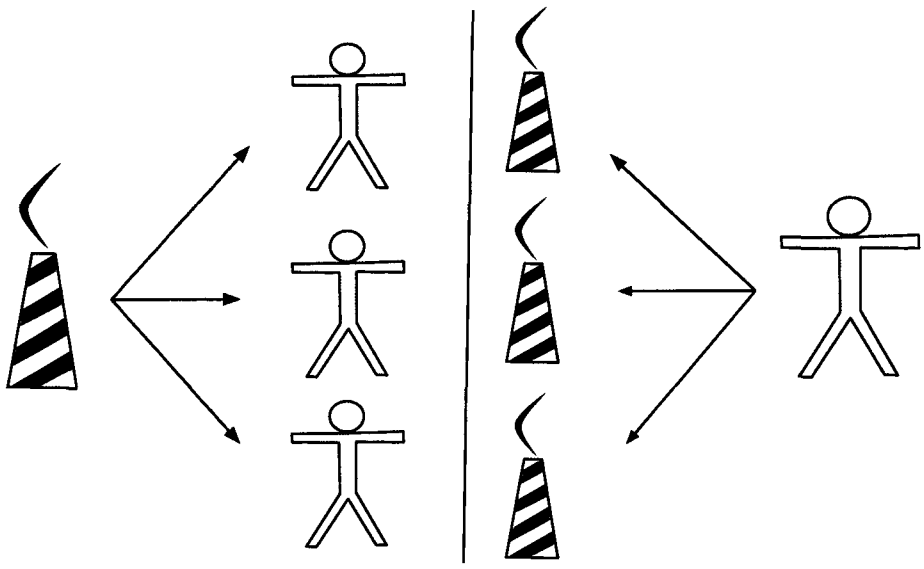


Figure 2.7: Illustration of (left) forward vs (right) backwards sensitivities.

of all sources on one receptor (or one receptor-based metric/function). This point is illustrated in figure 2.7.

There is no established way of comparing results of forward and backward sensitivities. This is due in part to the fact that each approach answers the question of sensitivities from a different perspective. Using the setup above (inputs $\tilde{\mathbf{x}}$ and outputs $\tilde{\mathbf{y}}$), if one were to calculate one set of forward and backward sensitivity each, the set of results would have only one intersection, in the pair of source i and receptor j.

2.2.3 Formal vs. Perturbation-Based Sensitivity Methods

In formal sensitivity analysis, equations that govern evolution of sensitivities in time and space are derived from model (concentration) equations. Sensitivity calculations are carried out by solving these auxiliary equations. In perturbation-based methods, sensitivity approximations are made by running the original model in a perturbed state (multiple times, if necessary). Since the original model is run forward in time, perturbation-based methods produce fields of "approximate" forward sensitivities.

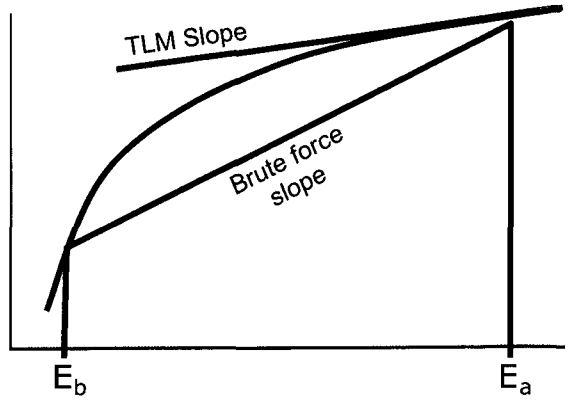


Figure 2.8: Illustration of brute force and TLM sensitivity of ozone to emission reduction. In this figure, E_a represents initial emissions of say NO_x, E_b represents and reduced level. One can see that for large perturbations in non-linear chemical regimes, the TLM and the BFM may provide very different results. In this figure, the first order TLM under predicts the response while the BFM over predicts the response.

2.2.3.1 The Brute-Force Method (BFM)

The BFM can be considered a finite difference approach, or most simply, *rise-over-run* (figure 2.8). The BFM is a rather straightforward method of calculating forward sensitivities, and thus can often be used as a comparison to evaluate the accuracy of formal methods of calculating derivatives.

The BFM involves re-running a simulation while making (slight) perturbations to the inputs and observing the change in the output in order to construct a sensitivity surface in the parameter space. This can be systematically done using factoral design techniques and response surface methods (Box et al., 1978; Rabitz et al., 1983). This method, in general, involves the investigator selecting a fixed number of "versions" or a number of variables (parameters) then running an experiment for all combinations. In the context of air quality modelling where numerous input parameters are used and simulations can take a long time, BFM could become impractical.

In general, the BFM can be expressed as a finite (in this case central) difference

equation,

$$F'(x) \approx \frac{F(x+\Delta x) - F(x-\Delta x)}{2\Delta x} \tag{2.8}$$

where F is a real valued function. This approach, therefore, entails running the model for each perturbation (value of Δx in this example) as well as choosing an appropriate magnitude for perturbation.

It is important to note that BFM is, by definition, a global method as it does not truly calculate the point (i.e. local) derivative. However, for perturbations small enough (or for rather linear models) BFM sensitivities should provide good approximations for local derivatives. On the other hand, for a very small perturbation, errors (numerical or round-off) in simulations may become comparable to the actual difference signal resulting in loss of accuracy.

2.2.3.2 The Complex Variable Method (CVM)

Squire and Trapp (1998) introduced a method of approximating derivatives of real functions using complex variables to avoid subtractive cancellation errors inherent in typical BFM approximations. Given the Taylor Series in Larson et al. (1999),

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(c)}{n!}(x-c)^n \tag{2.9}$$

setting $c = x$, and applying the Taylor Series to a perturbed function,

$$f(x+\Delta x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(x)}{n!}\Delta x^n \tag{2.10}$$

Using (2.10), a complex function $g(x)$ where $x \in \mathbb{C}$ can be expressed as:

$$g(x_0 + ih) = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!}(ih)^n \tag{2.11}$$

$$= g(x_0) + (ih)g^{(1)}(x_0) - \frac{h^2}{2}g^{(2)}(x_0) - \frac{ih^3}{3!}g^{(3)}(x_0) + ... \tag{2.12}$$

Isolating the imaginary component of (2.12), denoted by the $\Im()$ operator, yields:

$$\Im\{g(x_0 + ih)\} = hg^{(1)}(x_0) - \frac{h^3}{3!}g^{(3)}(x_0) + ... \tag{2.13}$$

Re-arranging (2.13), and dividing out h, one can isolate $g^{(1)}(x_o)$

$$g^{(1)}(x_o) = \frac{\Im\{g(x_0 + ih)\}}{h} + \frac{h^2}{3!}g^{(3)}(x_0) + ... \tag{2.14}$$

Therefore, provided the right hand side of (2.14), one can approximate the derivative,

$$\begin{aligned} g^{(1)}_{\text{approx}}(x_o) &\approx \frac{\Im\{g_{\text{calc}}(x_0 + ih)\}}{h} + \frac{h^2}{3!}g^{(3)}(x_0) + ... \\ &\approx g^{(1)}_{\text{actual}}(x_o) - \cancel{\frac{h^2}{3!}g^{(3)}(x_0)} + \cancel{\frac{h^2}{3!}g^{(3)}(x_0)} \\ g^{(1)}_{\text{approx}}(x_o) &\approx g^{(1)}_{\text{actual}}(x_o) \approx \frac{\Im\{g_{\text{calc}}(x_0 + ih)\}}{h} \end{aligned} \tag{2.15}$$

The result of (2.15) is not subject to cancellation errors and is accurate to $\mathbf{O}(h^2)$.

Using the CVM, the perturbation size h may be considerably smaller than that used in the BFM. Giles et al. (2008) found typically results remain accurate at a perturbation size on the order of $\mathbf{O}(10^{-4})$ for single precision calculations, and upwards of $\mathbf{O}(10^{-14})$ for double precision.

One constraint on the perturbation size is the magnitude of the truncation terms.

As (2.15) shows, the largest truncation term is,

$$T = \frac{h^2}{3!} g^{(3)}(x_0) \tag{2.16}$$

where T is the truncated terms. In certain circumstances the truncation error may grow large enough to affect the derivative approximation and even influence the real component of the function. For example, consider $f = 1/x$, the truncated term T would be,

$$T = \frac{h^2}{3!} g^{(3)}(x_0) = \frac{h^2}{x^4} \tag{2.17}$$

If the perturbation size in (2.17) is not much smaller than x, the truncated terms will degrade the accuracy of the approximated derivative.

2.2.3.3 Tangent Linear Model (TLM)/Decoupled Direct Method (DDM)

The TANGENT LINEAR MODEL (TLM) (aka the DECOUPLED DIRECT METHOD (DDM)) is a method of calculating the partial derivatives of a model (i.e. linearizing the model) by differentiating its governing equations. For example, consider the model described by (2.7). The TLM can be described as,

$$\delta\tilde{\mathbf{y}} = L'\delta\tilde{\mathbf{x}}, \qquad \text{or } L' = \frac{\delta\tilde{\mathbf{y}}}{\delta\tilde{\mathbf{x}}} \tag{2.18}$$

where $\delta\tilde{\mathbf{y}}$ represents perturbations in the output resulting from perturbation in the input, $\delta\tilde{\mathbf{x}}$, and L' is the linearized operator (the Jacobian) of L (Talagrand & Courtier, 1987). This linearization is done with respect to the tangents of the model trajectory in the model space. In other words, the derivative is based on the state variable rather than the variable of integration (typically space or time.) Details of TLM equations derivation will be discussed in the next section.

2.2.3.4 Backward Sensitivity

Backward sensitivity, also known as adjoint sensitivity, is in some sense an extension of TLM. Adjoint sensitivity is calculated backwards in time, and relates an output y_i to all inputs $\tilde{\mathbf{x}}$.

The adjoint operator can be defined as,

$$\delta J = \langle \nabla_{\tilde{\mathbf{y}}} J, \delta \tilde{\mathbf{y}} \rangle = \langle \nabla_{\tilde{\mathbf{y}}} J, L' \delta \tilde{\mathbf{x}} \rangle = \langle L'^{*} \nabla_{\tilde{\mathbf{y}}} J, \tilde{\mathbf{x}} \rangle$$
$$\Rightarrow \nabla_{\tilde{\mathbf{x}}} J = L'^{*} \nabla_{\tilde{\mathbf{y}}} J \tag{2.19}$$

where J is a scalar target (cost) functional of output, $\tilde{\mathbf{y}}$, and L'^{*} is the adjoint operator. Note that the gradient of J with respect to $\tilde{\mathbf{x}}$ is related to the gradient of J with respect to $\tilde{\mathbf{y}}$. For more information on the adjoint method, refer to Wang et al. (2001).

2.3 Sensitivity Analysis using the TLM

As discussed earlier, the TLM is a forward method of local sensitivity analysis that calculates the tangent to the response surface at the base case. This method may be applied to linear and non-linear systems alike. TLM is also referred to as the Decoupled Direct Method (DDM) (Dunker, 1981).

TLM/DDM stems from the direct method for sensitivity analysis (Dickinson & Gelinas, 1976). The specifics of the direct method are outside the scope of this thesis, but in short the method solves for coupled system of concentrations and sensitivities in the same step. This method has been shown to be unnecessarily expensive and prone to instability in certain circumstances (Dunker, 1981).

Dunker (1981) proposes DDM as applied to atmospheric models containing non-linear chemical kinetics. In contrast to the coupled direct method, DDM is faster,

simpler and more stable (Dunker, 1984). As discussed above, in the direct method a state matrix of concentrations and sensitivities is compiled into a matrix of size $2N$, where N is the number of species being solved. The decoupled method on the other hand solves the concentrations in two steps, both with a matrix of size N. In a proper implementation of DDM methodology, the effort required in solving for the sensitivity coefficients in each time step is approximately the same as that spent in solving for concentrations (Rabitz et al., 1983). DDM implementation entails linearization (i.e. differentiation) of the AQM and in that it is equivalent to TLM development. In this work TLM and DDM are used interchangeably.

DDM was first introduced in the context of atmospheric chemistry modelling by Dunker (1981, 1984). In 1981 Dunker implemented DDM in an Eulerian photochemical atmospheric model by Reynolds et al. (1973) and Roth et al. (1974). The host model described the transport, dispersion, and chemical reactions of gas-phase pollutants.

Pandis (1991) developed a limited DDM version for an aerosol box model. Yang et al. (1997) implemented a variation of the DDM called DECOUPLED DIRECT METHOD-3D (DDM-3D) in the California/Carnegie Institute of Technology three-dimensional photochemical AQM to show how the model responded to various emissions and a number of other model parameters. This version only included gas-phase processes. Dunker et al. (2002) also implemented DDM in COMPREHENSIVE AIR QUALITY MODEL WITH EXTENSIONS (CAMx) model for gas-phased species. Odman et al. (2002) extended DDM-3D in the urban-to-rural multiscale model (URM) (Boylan et al., 2002) to treat aerosol processes. This model was later extended to include secondary particulate matter formation in addition to gas-phase species (Napelenok et al., 2006). Hakami et al. (2003) and Cohan et al. (2005) implemented DDM-3D in gas-phase versions of MAQSIP and CMAQ, respectively.

Napelenok et al. (2006) extended DDM to include PM formation and transformation, called DDM-3D/PM in CMAQ version 4.1. To date, this implementation (CMAQ) is the only other active multi-phase TLM of an air quality model[1]. Further developments included extension of DDM to HIGHER ORDER DECOUPLED DIRECT METHOD (HDDM) for calculation of higher-order derivatives and cross-derivatives (Hakami et al., 2003).

The TLM equations are derived by differentiating concentration equations with respect to the parameter of interest. Integration of these equations results in what is referred to as the continuous DDM. Alternatively, one may differentiate numerical solutions to concentrations equations, i.e. discrete DDM. Both continuous and discrete DDM are expected to provide similar results for smooth functions. For linear subsystems/subroutines, the differentiated code closely resembles that of the original model, but for non-linear systems, such as those representing transformation of reactive pollutants in the atmosphere, TLM development requires additional work. In all cases, a key feature of the TLM is that the same spatial grid and time steps are used. This ensures maximum compatibility between the sensitivities and the concentrations (Dunker et al., 2002).

To derive the TLM, we begin with the ADE (Dunker et al., 2002):

$$\frac{\partial c}{\partial t} = [\text{advection} + \text{diffusion} + \text{chemistry} + \text{emissions}] \tag{2.20}$$

where c is the species concentration, and *process* is the presentation of the process by the numerical algorithm. For consistent nomenclature, equation (2.20) can be more precisely described by

[1]Koo et al. (2007) implemented DDM in portions of the CAMx model

$$\frac{\partial c_i}{\partial t} = -\nabla(\mathbf{u}c_i) + \nabla(\mathbf{K}\nabla c_i) + R_i + E_i \tag{2.1}$$

In a typical CTM the initial and boundary conditions could be given by

$$c_i\,(t_o) = c_i^o \tag{2.21}$$

and boundary conditions

1. Horizontal inflow: $\mathbf{u}c_i - \mathbf{K}\nabla c_i = \mathbf{u}c_i^b$
2. Horizontal outflow: $-\nabla c_i = 0$
3. Vertical inflow from surface ($z = 0$): $v_g^i c_i - K_{zz}\frac{\partial c_i}{\partial z} = E_i^b$
4. Vertical inflow from top ($z = H$): $-\frac{\partial c_i}{\partial z} = 0$

where $\mathbf{u}$ is the wind field, c_i^b is the concentration of compound i at the boundary, v_g^i is the dry deposition velocity, and E_i^b is the ground-level emission rate.

The local sensitivity can be calculated as the partial derivative of the output with respect to the parameter ϵ, i.e.,

$$s_{ij}(t) = \frac{\partial c_i(t)}{\partial \epsilon_j} \tag{2.22}$$

Conceptually, this can be represented as,

$$\frac{\partial s_i}{\partial t} = \left[\frac{\partial}{\partial \epsilon_i}\text{advection} + \frac{\partial}{\partial \epsilon_i}\text{diffusion} + \frac{\partial}{\partial \epsilon_i}\text{chemistry} + \frac{\partial}{\partial \epsilon_i}\text{emissions}\right]$$

Note that the semi-normalized sensitivity coefficients as described by (2.4) in section 2.2 are used.

Using a semi-normalized version of (2.22), the atmospheric advection diffusion equation may be differentiated with respect to a parameter ϵ_j.

$$\begin{aligned}\frac{\partial}{\partial \epsilon_j}\left(\frac{\partial c_i}{\partial t}\right) &= \frac{\partial}{\partial \epsilon_j}\left(-\nabla(\mathbf{u}c_i)\right) + \frac{\partial}{\partial \epsilon_j}\left(\nabla(\mathbf{K}\nabla c_i)\right) + \frac{\partial}{\partial \epsilon_j}\left(R_i\right) + \frac{\partial}{\partial \epsilon_j}\left(E_i\right) \\ \frac{\partial s_{ij}}{\partial t} &= -\nabla\left(\mathbf{u}s_{ij}\right) + \nabla\left(\mathbf{K}\nabla s_{ij}\right) + \frac{\partial R_i}{\partial \epsilon_j} + \delta_l + J_{ik}s_{ij} + E_i\delta_{ij} - \nabla(\mathbf{u}c_i)\delta_{ij} + \nabla(\mathbf{K}\nabla c_i)\delta_{ij}\end{aligned} \tag{2.23}$$

where J is the Jacobian matrix defined by $J_{ik} = \partial R_i/\partial c_k$, δ_{ij} is the Kronecker delta function – a binary variable whose value depends on whether i and j are equal ($\delta = 1$) or not ($\delta = 0$), and δ_l is a Kronecker delta function for if the sensitivity parameter is a chemical reaction rate.

The general initial conditions can be described by

$$s_{ij}(t_o) = c_j^o\delta_{ij} \tag{2.24}$$

and boundary conditions:

1. Horizontal inflow: $\mathbf{u}s_{ij} - \mathbf{K}\nabla s_{ij} = \mathbf{u}c_i^b\delta_{ij} - \mathbf{u}c_i\delta_{ij} + \mathbf{K}\nabla c_i\delta_{ij}$
2. Horizontal outflow: $-\nabla s_{ij} = 0$
3. Vertical inflow from surface ($z = 0$): $v_g^i s_{ij} - K_{zz}\frac{\partial s_{ij}}{\partial z} = -v_g^i c_i\delta_{ij} + \mathbf{K}_{zz}\frac{\partial c_i}{\partial z}\delta_{ij} + E_i^b\delta_{ij}$
4. Vertical inflow from top ($z = H$): $-\frac{\partial s_{ij}}{\partial z} = 0$

Gas-phase chemistry is described by the following non-linear system of ODEs:

$$\frac{\partial c(t,p)}{\partial t} = f(c,k,t) \tag{2.25}$$

as subject to specific initial conditions. Differentiating the above equations with

respect to a parameter p leads to a set of TLM ODEs:

$$\frac{\partial s(t,p)}{\partial t} = J(c,k,t)S + \frac{\partial f(c,k,t)}{\partial \epsilon} \tag{2.26}$$

where J and $\partial f/\partial \epsilon$ are the Jacobian of the reaction rates and the local derivative vector of the reaction rates, respectively. An example of using the Jacobian to calculate sensitivity is shown in appendix B.

2.3.1 Different Initial Condition and Boundary Condition Cases

The initial conditions and boundary conditions used in TLM equations can vary depending on the sensitivity parameter in question. For example:

- To calculate the sensitivity of species (i) with respect to the initial concentrations of species (j):

  Initial conditions would be:

  $$s_{ij}(t_o) = c_j^o \delta_{ij} \tag{2.27}$$

  Therefore, sensitivity field is initialized to concentrations of species j for sensitivities of j and is set to zero for all other species.

- To calculate sensitivity of ozone (i) to surface NO_x emissions (j),

  Initial conditions would be:

  $$\begin{aligned} \delta_{ij} &= 0 \\ s_{ij}(t_o) = c_j^o(0) &= 0 \end{aligned} \tag{2.28}$$

With the vertical influx boundary condition (BC #3) becoming:

$$v_g^i s_{ij} - K_{zz}\frac{\partial s_{ij}}{\partial z} = E_i^b \delta_{ij} \tag{2.29}$$

Therefore, emissions of the two NO_x species are injected to the sensitivity fields for those species while all other emissions are set to zero for sensitivity calculations. Sensitivity to non-ground source emissions, i.e., E_i from (2.1) rather than E^b are injected directly into the vertical layer rather than via the boundary condition. In order to calculate sensitivity to these emissions, one would use the initial conditions,

$$\begin{aligned} \delta_{ij} &= 0 \\ s_{ij}(t_o) = c_j^o(0) &= 0 \end{aligned} \tag{2.30}$$

with the vertical influx boundary condition (BC #3) becoming:

$$\begin{aligned} \delta_{ij} &= 0 \\ v_g^i s_{ij} - K_{zz}\frac{\partial s_{ij}}{\partial z} &= 0 \end{aligned} \tag{2.31}$$

2.3.2 Some Considerations on DDM Sensitivities

Equations (2.1) and (2.23) have very similar structures, and therefore, for the most part can be integrated by the same numerical routines (Yang et al., 1997). Note that in equations (2.1) and (2.23) sensitivities are dependant on concentrations, not the other way around; hence the decoupled nature of DDM. Therefore, DDM can be implemented with reduced effort and without significant architectural changes to AQMs. Due to its computational efficiency and relative ease of implementation, DDM remains the main local sensitivity analysis technique that has been extensively used

in three-dimensions AQMs (Hakami, 2003; Yang et al., 1997) although in recent years adjoint sensitivity analysis has received significant attention.

Above it was mentioned that TLM/DDM uses the same time steps for integration as for the concentration field. Many implementations of TLM in CTMs use an extension of DDM called DDM-3D (Yang et al., 1997). DDM-3D is conceptually the same as DDM but uses a different algorithm and time steps to integrate sensitivities through the chemistry step than those for the concentrations.

Sensitivity ODEs in DDM-3D are discretized (Hakami, 2003) for time step $n+1$ in terms of solution at time step n using the following semi-implicit scheme:

$$\frac{s^{n+1}-s^n}{\Delta t} = \mathbf{J}_{\bar{c}}\frac{s^{n+1}-s^n}{2} + \frac{\partial \mathbf{R}_{\bar{c}}}{\partial \epsilon} \tag{2.32}$$

where $\bar{c} = (c^{n+1}+c^n)/2$ results in the semi-implicit scheme. Solving this linear system for s^{n+1} yields:

$$s^{n+1} = \left(\mathbf{I} - \frac{\Delta t}{2}\mathbf{J}_{\bar{c}}\right)^{-1}\left\{\left(\mathbf{I} + \frac{\Delta t}{2}\mathbf{J}_{\bar{c}}\right)s^n + \Delta t\frac{\partial \mathbf{R}_{\bar{c}}}{\partial \epsilon}\right\} \tag{2.33}$$

The Jacobian matrix is independent of sensitivities, and the required matrix inversion - the main computational cost - is carried out only once for all sensitivity parameters at each time step and grid cell. Performing the inversion only once per time step and grid enables DDM-3D to calculate sensitivity coefficients for a large number of sensitivity parameters in an efficient manner (Hakami, 2003). Computational gain aside, this approach can reduce accuracy of results compared to the original DDM implementation if time steps are too large (Dunker et al., 2002).

2.3.3 Higher-Order Sensitivity Analysis

As mentioned before, TLM/DDM implementation amounts to linearization of the model. Therefore, TLM cannot account for non-linearity in response. For proper characterization of response surface under highly non-linear regimes, knowledge of higher-order derivatives or a global approach to sensitivity analysis may be necessary. Calculating higher order sensitivity coefficients allows one to better estimate the response of non-linear processes at larger perturbations away from the base case. These sensitivity coefficients are derived by further differentiating (2.23). Given (as above) a concentration c, a parameter P and perturbation ϵ, and perturbed parameter $p = \epsilon P$, the concentration for any fractional perturbation in a sensitivity parameter can be approximated as,

$$\begin{aligned} p &= \epsilon P = (1 + \Delta\epsilon)\, P \\ c(\Delta\epsilon) &\approx c(P) + (\Delta\epsilon)\, s^{(1)}(P) + \frac{\Delta\epsilon^2}{2} s^{(2)}(P) + \text{higher order terms...} \end{aligned} \tag{2.34}$$

where $s^{(n)}(P)$ is the n^{th}-order sensitivity of c to parameter P. Note that the second-order term scales with $\Delta\epsilon^2$, and thus its relative importance increases with the size of the perturbation (Cohan et al., 2005). This point is illustrated in figure 2.9.

First order derivatives provide good information on local response (not far away from the base case) to inputs while second order derivatives provide information about the curvature of the response, i.e. how the slope changes as one moves away from the base case. If the change is large, third and even fourth order derivatives may become necessary for an accurate projection of the response.

Hakami et al. (2004) extended DDM to include calculation of higher-order sensitivity coefficients (second, third, and fourth-order). By paying attention to the specific structure of atmospheric chemical rates and their Jacobian, their method,

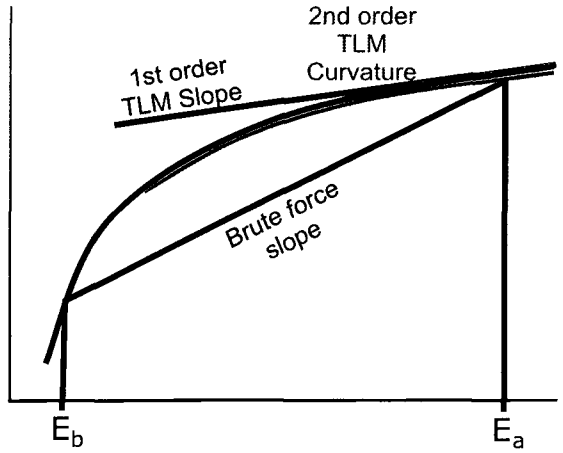


Figure 2.9: Brute force and DDM sensitivity analysis of ozone response to emissions. Given the typical concave down response, the brute force response to a large emission reduction is steeper than the local sensitivity at point E_A computed by first order DDM. The first and second order DDM enable a better approximation of response

called Higher-order DDM (HDDM), were able to eliminate the need for construction of a Hessian[1] and showed that the computational efficiency of calculating higher order derivatives (including cross-sensitivities) is comparable to that expended calculating first order derivatives. Hakami et al. (2004) used HDDM to create time and location specific ozone isopleths in one single simulations. (Cohan et al., 2005) used HDDM to investigate non-linearity in ozone response to control strategies. Both studies suggested that up to a change of about 30 percent, assumption of linear response is generally acceptable.

2.4 TLM Code Generation Tools

In developing the TLM for AURAMS we used two software for automated generation of differentiated code. This section introduces these two tools, TAPENADE and KPP.

[1]Matrix of second order derivatives

2.4.1 Tapenade

Given source code to a computer program that computes a differentiable mathematical function, say $f(x)$, TAPENADE will build a new source program that computes both the function $f(x)$ and the derivative, $f'(x)$. This practice is known as an AUTOMATIC DIFFERENTIATION (AD) tool.

AD functions by symbolically differentiating source code. The result is that the calculated derivative is exact within the limits of numerical precision and does not suffer from round-off error that would be introduced by numerical differentiation and finite difference approximations. Because the symbolic differentiation occurs in the source code (rather than being performed at run-time), no complex symbolic computation is done at run-time that would drastically slow down a program.

There are a number of AD tools available - both commercial and free. Some apply overloading techniques, some implement the derivative of a signal at compile time, and others use source regeneration techniques. TAPENADE uses the latter. This is done by creating an interpreter that can input and understand the mathematical operations within the source code, apply differentiation rules and output new source code.

The differentiation engine of TAPENADE works by making a number of assumptions about the program, P, being differentiated. First, AD assumes that P represents all its possible run-time sequences of instructions, i.e. the series of instructions for each path the program may take, and it will in fact differentiate these sequences (Hascoet & Pascual, 2004). In other words, the program P is broken down into a series of *controls*, and P is assumed to be piece-wise differentiable. This assumption has shown to be appropriate in the majority of situations.

Once P is broken up into a control of a series of instructions I_i, i.e.,

$$P_i = \{I_1, I_2, \ldots, I_p\}$$

one can interpret P as a function F expressed as,

$$F = f_p \circ f_{p-1} \circ \ldots \circ f_1$$

where each f_i is the elementary function implemented by instruction I_i. Once F is defined, AD applies the chain rule to obtain derivatives of F. Therefore, given $X_0 = X$ and $X_k = f_k(X_{k-1})$, the chain rule yields the Jacobian F' of F,

$$F'(X) = f'_p(X_{p-1}) \times f'_2(X_{p-2}) \times \ldots \times f'_1(X_0) \tag{2.35}$$

Finally, equation (2.35) may be mechanically translated back in to a sequence of instructions I'_k and inserted back into a copy of the control of P, yielding program P'.

In general, arithmetic operations defined by the programming language are indeed differentiable, however there are exceptions. For instance, the square root operator is defined on a null argument, but its derivative is not. This process is also subject to numerical precision. In the event of a quotient for example where exponents are introduced ($y = (f'g - fg')/g^2$), the resulting differentiated code retains the precision of the source which may be insufficient to store the value of the derivative. This is an issue encountered in this research and discussed in detail in section 3.2.5.

TAPENADE was built to produce the following derivative modes, i) tangent ("Jacobian times vector"), ii) adjoint ("Transposed Jacobian times vector") and iii) multi-directional tangent where it produces tangent code for several directions at a time

(Hascoet & Pascual, 2004). It is constructed in Java such that it can run on virtually any platform (even a cell phone.) TAPENADE is also available via a web form for users who do not wish to install it.

One advantage of TAPENADE is that its design includes a language interpreter and language output module for FORTRAN77, FORTRAN95 and C. As the user's program is completely re-written, a user can input source code of one language and output it in another. This is useful when working with code that is a mix of the three languages above - a situation that is not uncommon. The architecture permitting this is shown in figure 2.10.

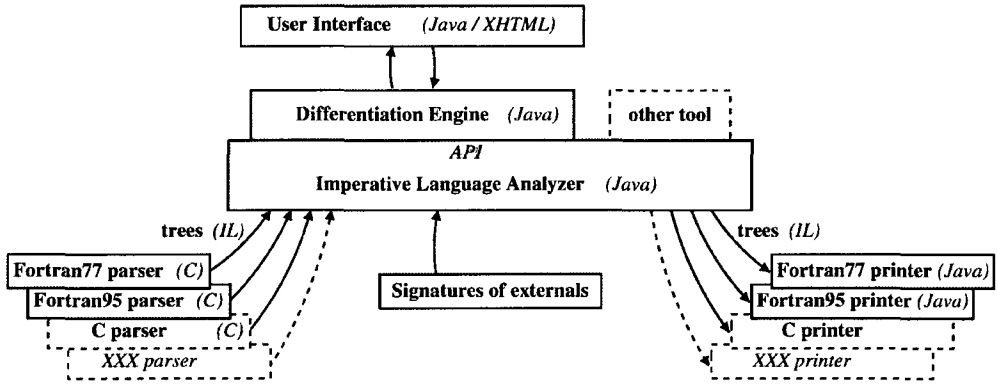
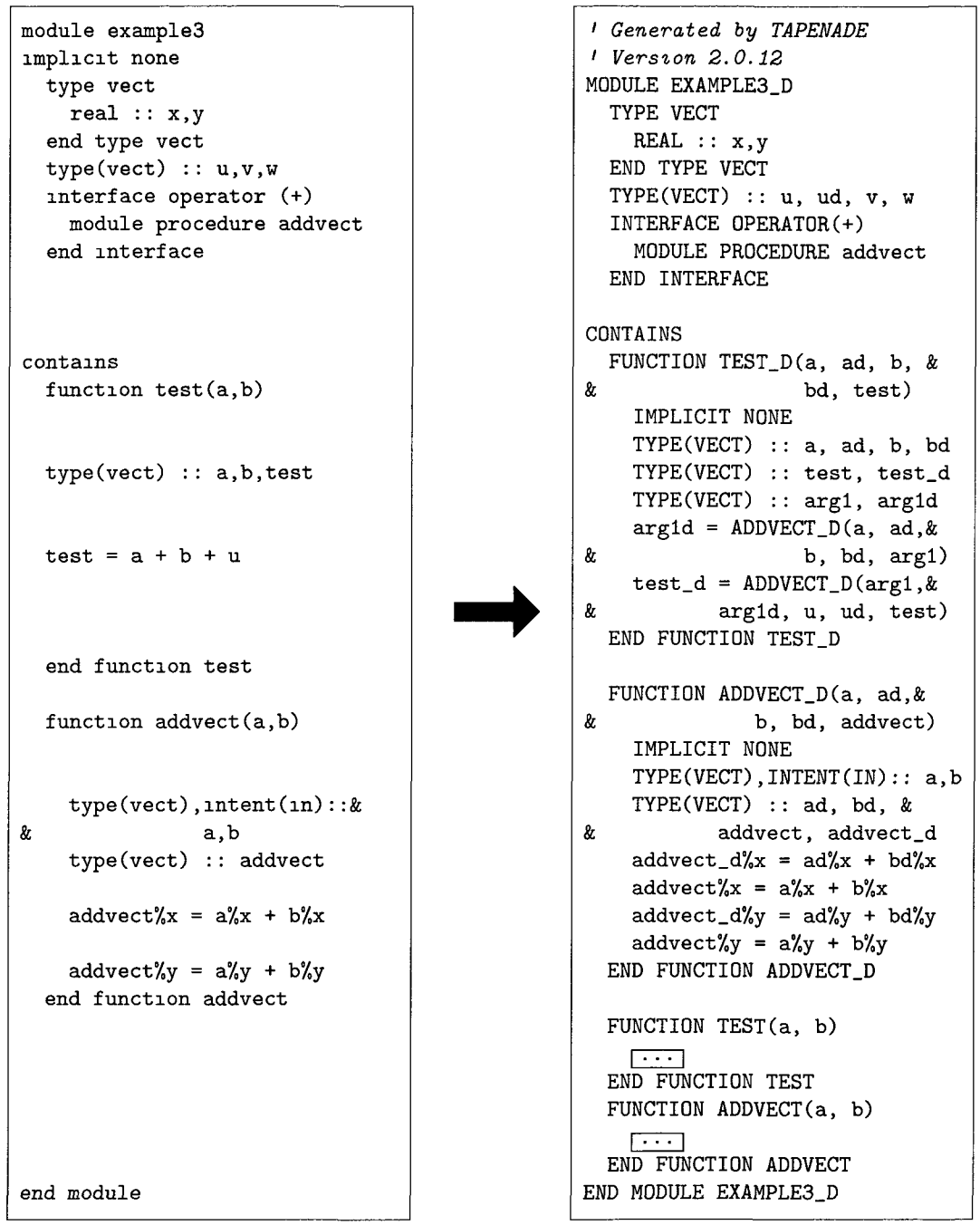


Figure 2.10: Architecture of TAPENADE. Figure adapted from Pascual and Hascoet (2006)

An example of the translation TAPENADE performs for FORTRAN95 code is shown in figure 2.11.



```
module example3
implicit none
  type vect
    real :: x,y
  end type vect
  type(vect) :: u,v,w
  interface operator (+)
    module procedure addvect
  end interface



contains
  function test(a,b)


  type(vect) :: a,b,test


  test = a + b + u



  end function test

  function addvect(a,b)


    type(vect),intent(in)::&
&               a,b
    type(vect) :: addvect

    addvect%x = a%x + b%x

    addvect%y = a%y + b%y
  end function addvect









end module
```

```
! Generated by TAPENADE
! Version 2.0.12
MODULE EXAMPLE3_D
  TYPE VECT
    REAL :: x,y
  END TYPE VECT
  TYPE(VECT) :: u, ud, v, w
  INTERFACE OPERATOR(+)
    MODULE PROCEDURE addvect
  END INTERFACE

CONTAINS
  FUNCTION TEST_D(a, ad, b, &
&                 bd, test)
    IMPLICIT NONE
    TYPE(VECT) :: a, ad, b, bd
    TYPE(VECT) :: test, test_d
    TYPE(VECT) :: arg1, arg1d
    arg1d = ADDVECT_D(a, ad,&
&                 b, bd, arg1)
    test_d = ADDVECT_D(arg1,&
&          arg1d, u, ud, test)
  END FUNCTION TEST_D

  FUNCTION ADDVECT_D(a, ad,&
&             b, bd, addvect)
    IMPLICIT NONE
    TYPE(VECT),INTENT(IN):: a,b
    TYPE(VECT) :: ad, bd, &
&          addvect, addvect_d
    addvect_d%x = ad%x + bd%x
    addvect%x = a%x + b%x
    addvect_d%y = ad%y + bd%y
    addvect%y = a%y + b%y
  END FUNCTION ADDVECT_D

  FUNCTION TEST(a, b)
    [...]
  END FUNCTION TEST
  FUNCTION ADDVECT(a, b)
    [...]
  END FUNCTION ADDVECT
END MODULE EXAMPLE3_D
```

Figure 2.11: Example of differentiating a FORTRAN95 module including operator overloading and derived types. Figure adapted from Pascual and Hascoet (2006)

2.4.2 Kinetic Pre-Processor

KINETIC PRE-PROCESSOR (KPP) (Damian et al., 2002; Sandu et al., 2005) is a tool designed to transform a chemical mechanism into computer code (FORTRAN77, FORTRAN95, C, or MATLAB.) It does so by inputting the chemical mechanism specified by the user and transforming it into a collection of differential equations. The user then has the option of which solver they wish to use. The choice of solver (from a suite of Rosenbrock and Runge-Katta solvers) grants the user to ability to optimize the run time or the precision of the solver.

KPP was created to address the need of modellers to quickly and effectively implement chemical mechanisms into their models. Without KPP, a modeller may,

- Hard-code their solver into their model. This approach is error-prone and does not lend itself well to modification, however the efficiency of the implementation is bounded only by the modellers programming talents.

- Creating a suit of tools that allows input of new chemical mechanism to be read at run time and stored in memory. This approach may add extra computational intensity. A similar approach was used by Nowak (1982) in LARKIN.

- Implementing a system which pre-processes a chemical mechanism, the approach used by KPP.

The strength of KPP compared to the methods above is that the routines generated by KPP have been optimized and provide a routine virtually as efficient as any hard-coded method could offer. The generated routines are also designed to be easily integrated into any [FORTRAN77, FORTRAN95, C, or MATLAB based] host model. Moreover, KPP generated code has been tested and is a proven technology, therefore KPP generated code is considered virtually guaranteed to be error free. (Damian

et al., 2002). Finally, because the solvers are automatically generated, the chemical mechanisms lend themselves easily to changes.

Sandu et al. (2003) extended KPP to be capable of generating direct and adjoint sensitivity analysis of chemical systems. It is this extension that is used in this research (discussed in section 3.2.4.)

Chapter 3

Methodology

The predominant philosophy employed in developing the TLM in AURAMS was to divide the model in to separate processes. This allowed us to narrow in on one process, apply the appropriate changes, test it and then re-integrate the process into the model. The division of the processes follows fairly well the terms in equation (2.20). The CAM was also broken up in a similar manner for separate development and testing. Throughout, our focus has been on accuracy (particularly for aerosols) before computational efficiency. Improvements of computational efficiency will be addressed in future work.

This chapter discusses the details of methodologies used in TLM implementation in each process, challenges faced and the solutions to many of the problems that occurred during the implementation.

3.1 AURAMS Installation and Modification

AURAMS is the product of many years of development by Environment Canada and contains within it new and legacy code. In order to build AURAMS, one must become familiar with core Environment Canada software libraries and other resources. For

this reason Environment Canada has set up a site to ease collaboration with in-house and external parties, namely the RPN.COMM: COmmunity for Mesoscale Modeling (http://collaboration.cmc.ec.gc.ca/science/rpn.comm/weblog/)

The primary tool this site provides that is pertinent to AURAMS is *librmn.a*. This library handles all aspects of dealing with FST files - EC's custom grided data file format. By downloading the source for this library along with a myriad of build tools we were able to generate a 64 bit version of this library for both Linux and OS X systems. This task alone was a significant achievement as ENVIRONMENT CANADA (EC) no longer funds support for external developers nor makes its code available to an open community that would make this procedure more common and better documented. Indeed, this was the first time *librmn.a* was ever built on a Mac (Moran, personal communication), and sets precedent for external developers to be able to build and run AURAMS independent of EC's computing resources.

3.1.1 Building AURAMS

Our first successful build of AURAMS was aided by the Air Quality Modelling team at National Research Council (NRC). This build included a pre-compiled version of *librmn.a* that was built for 32 bit architecture. Using this version of the library limited the entire model to a 32 bit architecture. Using this build, the FST functions provided by *librmn.a* could only handle data file sizes up to 2.1 GiBs (personal communication). We developed work-arounds for this issue but it gradually became clear that we would have to build our own *librmn.a* library that ran on our native architecture (Mac OS X 86x64) to avoid the limitations imposed by the 32 bit architecture.

Using the resources on the *rpn.comm* website, we were able to build our own *librmn.a* (version 12) with the PGI FORTRAN95 compiler (version 10.9.) This task included many challenges that are omitted here, but are available in the AURAMS TLM

development report (Russell et al., 2011). A Linux 86x64 version of the library was also built using the same steps.

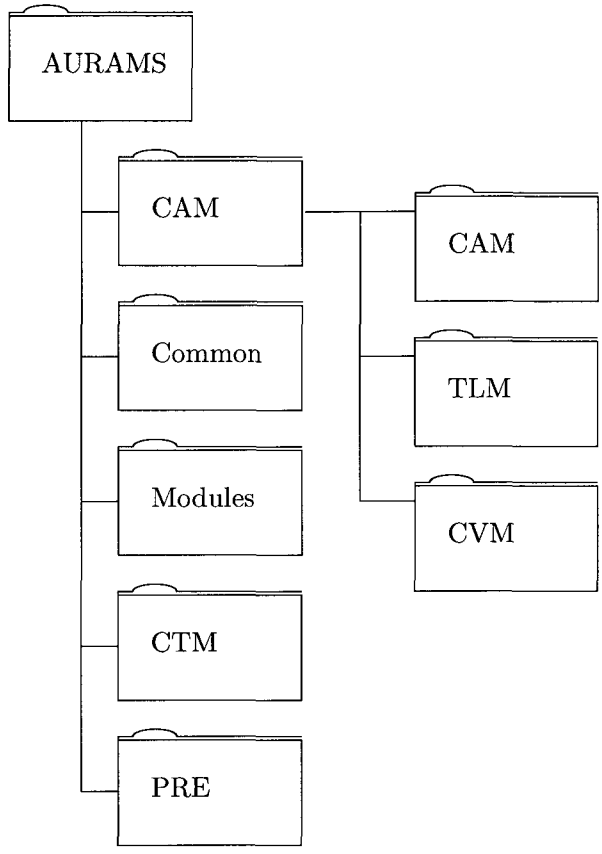
Once the 86x64 library was built, we observed a 15% speed increase on a Ubuntu OS with 4 core Xeon Kentsfield Q6600 processor, and a 23% speed increase on a Ubuntu OS with a 4x4 core Xeon Tigerton X7350 processor. Running on OS X with a hyper threaded 2x8 core Gulftown E5620 processors saw another speed increase of 50% relative to the Tigerton processor.

We were also now capable of using 12 hour gem files without splitting them into smaller files that could be handled by the 32 bit architecture. The results of the run with the 64 bit *libmn.a* (version 12) were virtually identical to the previous results attained with a 32 bit (version 8) *libmn.a* (not shown here; differences in the order of 10^{-3} ppbv for ozone.) From this point forward, we primarily developed AURAMS on a multi-core Mac OS X computer as it performed the fastest among available desktops.

During development, the AURAMS file structure was also re-organized to better accommodate modules (section 3.1.2) and differentiated versions of the code. For example, the CAM consists of over 40 FORTRAN95 files that required differentiated versions to be created, and upwards of 10 common files between the two. Because the CVM method was used to evaluate the TLM results, another 40 files for the CVM version were required. Though versions of the files had different file names, having this many files in one directory would inevitably lead to developer confusion and raise the probability of simple mistakes. Furthermore, module names were not changed from version to version, and because of the way modules are implemented in FORTRAN95[1] two versions of the same module cannot co-exist in the same directory.

[1]When a FORTRAN95 module is compiled, say *routine.f90* that contains the module `Routine`, two files are created: the object file *routine.o* and the module file *routine.mod*.

Therefore, the directory structure was separated into:



A recursive *Makefile* pattern was created to build all AURAMS source files.

Once we began work on the CAM, we were able to work without *librmn.a*, this made it feasible to build the CAM with different compilers to exploit the unique advantages provided by each compiler. Using different compilers also proved effective in identifying more bugs in the code. For example,

- Intel's ifort has a feature to generate subroutine interfaces which allowed us to check that each subroutine was getting the variables and variable shapes it expected. This led to the discovery of several bugs that we reported to EC.
- The GNU FORTRAN95 compiler provides many informative warnings that aided us in removing un-used variables, and performs better array bound checking at

compile time than PGI does by default.

- The PGI FORTRAN95 compiler warns the user when any intrinsic keyword has been overwritten.

Testing the CAM across multiple compilers imposed a stricter standard than any one of the above compilers and led to more compatible code.

3.1.2 New Modules

Several new modules were created and added to AURAMS during the development of the TLM in order to reduce the footprint on the original AURAMS code via module containment.

3.1.3 Sensitivity

The purpose of this module is to render the creation of sensitivity fields and effective management of the memory consumed by these fields simple so as not to be a distraction to the user. The sensitivity module has been primarily developed to provide first order sensitivity capabilities, however the base functionality of this module was developed to also store second order sensitivities; extending the capability to fully support second order sensitivities could be achieved in relatively short time.

This module is documented in detail in appendix A, but in brief, sensitivities fields are allocated at load time. The number of sensitivity fields is configured in a user input file thus eliminating the need to re-compile the model but simply to alter sensitivity parameters. These sensitivity fields are addressed via pointers implemented in the CTM. These pointers were prefixed with "`p_`".

3.1.4 Physics Constants

The second module is called *physics_constants*. Throughout the code, many of the same constants are repeatedly defined (e.g. π) with slightly differing values, or carried as non-parameter values in deprecated[1] *COMMON* blocks. Although organizing the commonly used constants into a module was outside the scope of the project, doing so greatly eased the automatic differentiation and the legibility of the code. Moreover, this move appears to be consistent with the direction later versions of AURAMS have taken. For example, revision 567 of AURAMS also appears to have a module for commonly used constants.

Moving these constants to a module allowed us to remove the majority of the *COMMON* blocks used in the CAM. This was seen as favourable by EC as they have also invested a good deal of effort removing *COMMON* blocks when they migrate code from AURAMS to their other [still in development] air quality model, GEM-MACH (Moran, 2011).

3.1.5 Common Derivatives

For the large amounts of code in the TLM implementation in AURAMS that were programmed manually, a module was created to store many common derivative functions often used. Examples of functions include a multiplicative and quotient rule for derivatives, and differentiated `min()` and `max()` functions. Examples on using these are given in section 3.2.1.

3.1.6 Polynomials

The CAM module involves many polynomial equations that when differentiated led to incredibly long lines. Worse, in routines where a Newton's method was used these lines were differentiated a second time resulting in even longer lines. These lines (some ranging to thousands of characters) are unfeasible to maintain, and cannot be compiled by some

[1] In the context of software, a deprecated practice or method is one that is discouraged by newer standards and will likely not be supported in future versions

compilers (e.g. PGI.)

These inline polynomial calculations were replaced with a call to a subroutine in the *calcPoly* module. Using this routine greatly increases the legibility of the code while not increasing computational demands.

3.1.7 Statement Function Replacements

Several other modules were introduced as replacements for deprecated statement functions. These modules are used exclusively by the subroutine in which they replace the statement functions. The driving force behind creating these modules was that the statement functions raised issues with automatic differentiation and OpenMP. Example routines where this replacement was done are `sulfate()`, and `sfss()` in the CAM.

3.2 Process Differentiation

The main philosophy of the TLM/DDM is for sensitivity operations to mimic concentration integrations as closely as possible. Throughout the code, all operations on the *mass* (concentration) array `MA`, have been modified to also include (or be repeated for) the sensitivity fields/arrays. Our implementation resulted in very few subroutine interfaces being modified to accommodate the TLM. A separate version of the CTM was then modified to permit brute force calculation of sensitivities with respect to certain parameters (described in section 3.3.) In this section specific aspects of differentiating various processes (if applicable) are discussed.

3.2.1 Differentiating Advection

Mathematically advection is a linear operator defined by eq (3.1).

$$\frac{\partial C}{\partial t} = -\frac{\partial (uC)}{\partial x} \tag{3.1}$$

Due to various reasons, most specialized advection algorithms are in fact a non-linear representation of this linear process. These inherent non-linearities (and in many cases discontinuities) make differentiation of advection schemes more complicated. In AURAMS the advection operator is an implementation of a positive definite non-oscillatory Semi-Lagrangian scheme (Smolarkiewicz & Pudykiewicz, 1992). This method is unconditionally stable and generally less diffusive compared to most Eulerian advection schemes. Since this method does not conserve mass, AURAMS contains a subroutine that corrects any discrepancy in mass (discussed below.)

Implementing the TLM in the advection subroutine required line-by-line differentiation of the algorithm, for example:

```
X(I,J, 1 - IS) = x(i,j, 1)

X(I,J,N3 + IS) = x(i,j,n3)
```

```
S_X(I, J, 1 - IS) = s_x(i,j, 1)
X(I, J, 1 - IS) = x(i,j, 1)

S_X(I, J,N3 + IS) = s_x(i,j,n3)
X(I, J,N3 + IS) = x(i,j,n3)
```

where S_X is the differentiated value of X.

The following is a more complicated example requiring differentiation of a concentration-dependent ratio:

```
Y00 = (mx0 - w0) /
   (-PN(f10) + PP(f00) + ep)
```

```
TMP_F = mx0 - w0;
TMP_G = PP(f10) - PN(f00);
TMP_FP = s_mx0 - s_w0;
TMP_GP = -PN_d(f10, s_f10) +
   PP_d(f00, s_f00);

S_YP10=quotient(TMP_F,TMP_FP,
                TMP_G,TMP_GP)

YP10 = (mx0 - w0) /
   (-PN(f10) - PP(f00) + ep)
```

The `quotient()` and `mult()` are two functions that were developed to perform the quotient and multiplicative law for derivatives in an attempt to reduce developer error and reduce the difficulty of testing the code.

Functions such as `min()` and `max()` were differentiated by creating separate versions

of the functions, namely `min_d()` and `max_d()`. These functions were developed with an interface in a separate module such that their usage is simple, seamless, extendable and not dependent on the number of passed arguments. These functions are contained in the *common_derivatives* module (section 3.1.5).

```
PN(XI)= min(0.0,XI)

YM10  = min(1.0, ym10)

MX0   = max(w0,w1,w2,f01)
```

```
PN(XI) = min(0.0,XI)
PN_D(XI, S_XI) = min_d(0., 0., XI, S_XI);

YM10 = min(1.0, ym10)
S_YM10 = min_d(1.0, 0.0, ym10, s_ym10);

MX0 = max(ym10,y00,yp10,w0);
S_MX0 = max_d(w0,s_w0, w1,s_w1, w2,s_w2,
   f01,s_f01)
```

The above code shows examples from the code of the `min_d()` function. The `min_d()` function finds the minimum of all the odd arguments (the *conditions*), and returns the corresponding even value (the *results.*) For example, in the second line, if the minimum value is `1.0`, then `0.0` is returned, if `ym10`, then `s_ym10` is returned.

This approach, though reducing development and testing time, significantly increased runtime to unacceptable levels using the Portland Compiler (results highly varied with different compilers.) The bottle neck was overwhelmingly due to how the `min_d()` and `max_d()` functions were implemented. To efficiently deal with this without losing the advantages of the approach, a transformation script was written that replaced the easy-to-read calls with inline code. Therefore, the code could be developed and maintained with intuitive function calls, then automatically translated to an optimized version. This avoided the need, and ran faster than, using advanced compiler inlining optimization options.

The TLM is developed such that sensitivity fields with respect to multiple parameters can be processed simultaneously. The first attempt to create such capability in advection was done outside of the routine by looping over it while providing a different sensitivity field for each iteration. The result was that concentrations had to be re-initialized and

re-integrated once for every sensitivity parameter being calculated. This too proved unacceptably slow and inefficient. To combat this the multiple sensitivity parameters were moved inside the routine. Every temporary variable was converted to an array of size n, where n is an input into to the routine defining the number of sensitivity parameters being considered, then all operations were looped over these arrays. For example,

```
S_X = s_x0
X = x0
```

```
IF (runTlm()) THEN
        do sens=1,n
        S_X(sens) = s_x(sens)
        ENDDO
ENDIF
X = x0
```

This architecture requires that all sensitivity fields also be provided to the routine. An efficient method for this was implemented in the `ctm()` but is omitted here as it is of a technical nature and does not contribute to the conceptual description provided.

Loop constructs were chosen over vector operations in the interest of run time (vector operations – even of a vector of dimension 1 – are considerably slower than scalar operations.)

The mass conservation routine, though a separate routine, is run immediately following advection and thus will also be discussed here. This routine scaled post-concentration values based on a tracer field. Implementing the TLM into this routine meant that every value in the sensitivity fields had to be scaled by the same factor as concentration field for that location and time. This was implemented and extended to support multiple sensitivity parameters in much the same fashion as it was in the advection routine.

3.2.2 Differentiating Diffusion

Mathematically, the diffusion operator can be described as

$$\frac{\partial c}{\partial t} = \mathbf{K}\frac{\partial^2 c}{\partial x^2} \tag{3.2}$$

This is a simplified form of the diffusion operator as in reality the diffusivity tensor can be variable in space. The diffusion operator in AURAMS is an unconditionally stable linear Crank-Nicholson diffusion scheme. The solver itself is wrapped in a high pass filter that "zero's out" negative concentration values that result from spurious oscillations in the algorithm. In order to differentiate this routine, `difvv()`, a separate version was created, `difvv_d()` to operate on the sensitivity fields.

With the exception of the high pass filter (sensitivities, unlike concentrations, can be negative), the two routines are virtually identical. This is possible because of the linear implementation of the diffusion routine. To accommodate the high pass filter the original diffusion operator now receives a masking field that it writes to whenever it zeros a negative concentration. The differentiated version, `difvv_d()`, then inputs this zero mask such that it can properly differentiate the concentration signal despite the non-linear high pass filter applied in the original version.

This implementation allowed us to extend the differentiated routine to accommodate multiple sensitivity parameters by calling it in an external loop over multiple sensitivity parameters. Because the differentiated process remains linear, the routine can also be used to calculate higher order sensitivities without any additional modifications.

This process is also responsible for the injection of flux emissions into the concentration field. As the processes that inject these fluxes are linear, no alterations were necessary. The differentiated version of diffusion, `diff_d()` now inputs (if applicable) the flux fields to be injected into sensitivities.

For example,

```
! Run diffusion on the concentration work field.
CALL difvv(WKFLD3D, [...] , flxt,flxb, [...] , ZEROD_MASK)

! Diffuse first order sensitivities
CALL difvv_d(P_WKFLD3D, [...] , p_flxt,p_flxb, [...] , zerod_mask)
```

where `flxt` and `flxb` are the flux fields to be injected into the concentration field, and `p_flxt` and `p_flxb` are flux fields to be injected into the sensitivity field. The character

case of the variables was set to indicate whether a variable is simply read (lowercase) or uppercase (written to), here it shows that `zerod_mask` is written in `difvv()` and only read by `difvv_d`. Note that our TLM implementation calculates semi-normalized sensitivity coefficients, and therefore, appropriate emission rates/fluxes are injected into the sensitivity field. For calculation of absolute sensitivities a flux/rate of unity should be injected instead – a modification that is straightforward to make.

3.2.3 Differentiating Emissions

Flux Emissions

The flux emissions in AURAMS include all emissions except major point source emissions. These emissions are compiled into 2D flux fields (first and top layers) and injected into the grid as a boundary condition in the diffusion operator (section 3.2.2.) In order to calculate sensitivity to flux emissions, parallel flux fields were implemented for each sensitivity parameter.

Major Point Source Emissions

AURAMS treats major point source emissions differently than all other emission sources. Because major point source emissions are often injected higher into the atmosphere than the first modelled layer, these need to be injected into the grid separately. To do this, AURAMS generates a 3D field of major point source emissions that gets injected into the concentration field. To calculate the forward sensitivity to a species that has a major emission, these emissions are also injected into the sensitivity field when applicable.

3.2.4 Differentiating the Chemistry Solver

We began the differentiation on a version of AURAMS that had a KPP-generated solver (*Rodas-3*) implemented for integration of gas-phase chemistry for the ADOM-II mechanism. Because the KPP implementation in AURAMS was not standard, we followed the

structure that existed in the original implementation but added additional KPP-generated subroutines that are required for TLM integration in the chemistry process (`chemf`). These additional subroutines were generated using original KPP input files (provided by EC.)

Details of continuous and discrete TLMs of Rosenbrock and Runge-Katta solvers are presented in detail in Sandu et al. (2003) and is not repeated here. As explained therein, in theory, for Rosenbrock solvers numerical solutions to the continuous sensitivity equations (continuous TLM) are the same as the sensitivities of the numerical solutions (discrete TLM). However, due to computational simplifications, continuous and discrete TLMs of equation (2.26) are in practice different. Here, we implement discrete TLM while noting that differences between continuous and discrete TLM will be negligible for all practical applications of atmospheric chemistry.

3.2.5 Differentiating the Canadian Aerosol Module (CAM)[1]

The aerosol processes were approached very similarly as the gas phase processes, i.e. flags were implemented to be able to individually test and run the sub-processes separately. These processes are `sulfate()`, `condsoa()`, `coagd()`, `aerocld_new()`, `scavenge()` and `drypar()`. `aerocld_new()` includes several other processes that were also individually tested.

The aerosol code was first differentiated using TAPENADE 3.5 (discussed in section 2.4.1.) For the vast majority of the aerosol code, TAPENADE provided quality code that functioned flawlessly. In a small number of cases however, the TAPENADE generated TLM code caused various problems such as numerical precision errors in single precision computations. For this reason, we thoroughly examined the TAPENADE generated TLM throughout the development process.

For example, assume a line of code to be differentiated with respect to `x` is,

[1]Considerable contribution to the development of the CAM was provided by Dr. Shunliu Zhao. His contributions include implementation of the Newton method, and solving issues related to numerical precision with regard to the TLM signal.

```
CALL someRoutine(x, f_x);
CALL someOtherRoutine(x, g_x);
y_x = f_x/g_x;
```

where all the variables are defined as `REAL*4`s

TAPENADE would generate code similar to,

```
CALL someRoutine_d(x, f_x, f_x_d);
CALL someOtherRoutine_d(x, g_x, g_x_d);
y_x_d = (f_x_d*g_x - f_x*g_x_d)/g_x**2;
y_x   = f_x/g_x;
```

`REAL*4`s in FORTRAN95 can only represent values ranging from approximately 1×10^{-34} to positive 1×10^{34}. When `g_x` is smaller than 1×10^{-17} (or larger than 1×10^{17}), because of order of operation `g_x**2` is evaluated first to a value outside the range of a `REAL*4` type and results in a `NaN` value.

A naïve fix of changing the order of operation, e.g.,

```
y_x_d = ((f_x_d*g_x - f_x*g_x_d)/g_x)/g_x);
```

might prevent `NaN`'s (depending on the magnitude of derivatives), but would not give good TLM results (see figure 3.1.)

As long as the value of `y_x_d` is still within the range of single precision numbers, a simple fix can be done,

```
double precision :: onedp = 1.d0
! ...
y_x_d = (f_x_d*g_x - f_x*g_x_d)/(onedp*g_x**2);
```

The insertion of `1.d0` converts the computation to double precision and results in an extremely accurate TLM signal. This has been done throughout the single precision routines in the TLM code of CAM. In future work, it may be prudent to alter differentiated routines from using single precision to double if one expects differentiated computations to often extend outside the range of single precision variables.

Once we were satisfied that the code in the CAM was properly differentiated, we turned our attention to the known problem of the bisection method used in CAM. The bisection

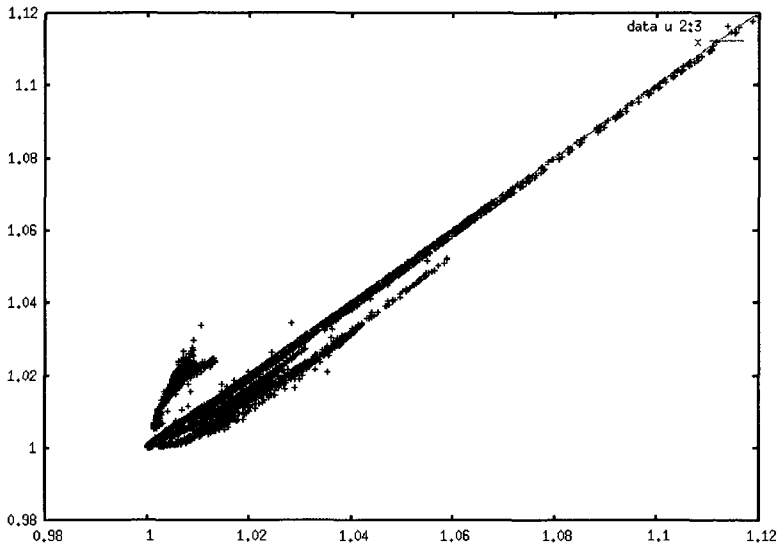


Figure 3.1: The comparison between CVM and TLM with a fix of changing the order of operation in TLM. Changing to double precision results in perfect accuracy. Only processes sulfate and condensation are included.

method is an effective root-searching algorithm. As far as differentiation is concerned, the bisection method removes perturbations from passing through the corresponding tangent linear procedure resulting in the computed derivative being zero. This interruption of perturbation propagation is inherent with the bisection procedure and has been extensively discussed by Gruntz (1997).

One solution is to replace the bisection procedure with a Newton-type approach which has been shown to work well with differentiation (Griewank, 2003). For a Newton-type approach to work properly with the aerosol processes in AURAMS for example, it must be globally convergent due to the high non-linearity of the processes. In terms of computational cost and ease of implementation, this is very expensive. An alternative approach is to perform post-differentiation (Bartholomew-Biggs, 1998), in which an extra step of post-convergence calculation of derivatives is added at the end of an iterative procedure (e.g. bisection.) The post-differentiation technique is accurate and efficient in derivative computations and was described in detail in Bartholomew-Biggs (1998). In this development work, the post-differentiation approach is adopted.

There are six such routines that implement bisection search methods which all reside in heterogeneous chemistry subroutine. Their routine names are `case1()`, `case5()`, `case8()`, `case11()`, `case12()` and `poly3v_p()`. They are all called by routine `het_v()` and call routines `water()` and `activity()`, except for `poly3v_p()` which did not require modification. Therefore, in total 8 routines were modified to facilitate post-differentiation.

The Newton method was used for the post-convergence derivative evaluation. Without loss of generality, the equation solved by the bisection method in CAM can be written as

$$f(x) = 0.0$$

The Newton method applied to the above equation can be written as

$$x = x + dx$$

$$dx = -f/f'$$

Thus, the implementation procedure of the Newton method for each case is as follows,

- Create a new routine to calculate function, f, based on the original code
- Generate a TLM routine to calculate f'
- Write a Newton routine and incorporate it into the code

After the routines of f were created, a wrapper program containing all the routines to be differentiated was written. The differentiation of routines of f were not carried out case by case. The reason is that there are shared routines, e.g. `activity()` and `water()`, among all the cases. If treated case-by-case, a shared routine would have various versions of TLM code generated by TAPENADE. Since different variables from a shared routine are needed by a different calling routine, TAPENADE would have treated different sets of variables as active. This treatment would have led to variations in the interface and the computational code.

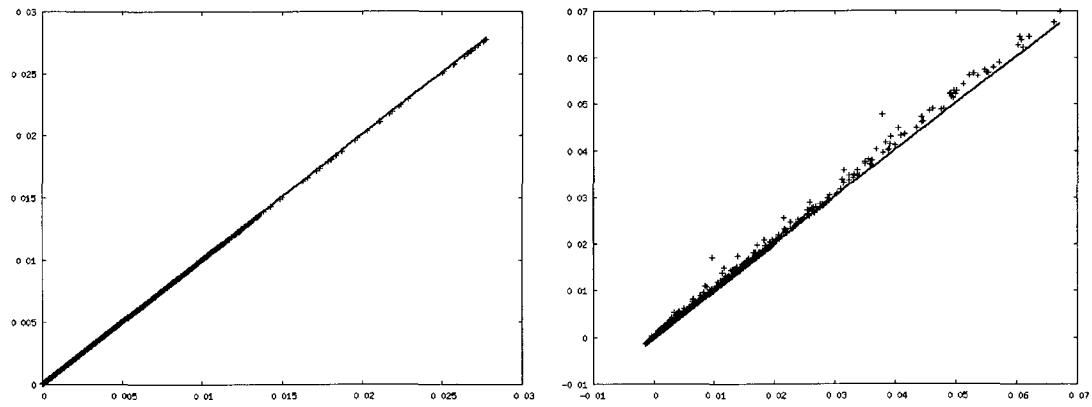
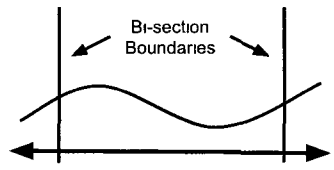


Figure 3.2: Impact of using post-convergence Newton procedure. The left figure shows no impact on concentrations, the right figure shows a significant impact on sensitivity. The range on the right figure is [0.0, 0.7] for both axes. Figures prepared by Dr. Shunliu Zhao.



The Newton procedure is called when a root is found by the bisection procedure. Under some circumstances however, when the value of f is small at the boundary (right figure), the bisection procedure gives a root when there is actually no root. An extra conditional, $f' > small$, was added to prevent calling the Newton's routine when such a situation arises.

Once the bisection searches were modified to include the Newton's Method, our testing turned to running the code and comparing TLM and BFM results. Immediately it was observed that brute force sensitivities were very incompatible with TLM sensitivities. Therefore we decided to create a version of the CAM that used the COMPLEX VARIABLE METHOD (CVM) (discussed in section 2.2.3.2.)

Similar to generating differentiated code using TAPENADE, we were able to create a program that automatically generated a CVM version of the code. This was made possible by tweaking the original code (e.g., changing 0 to 0.0 in comparison operators, defining variables that should be parameters as parameters, etc) and creating a *cvm_ops* module that provides or overloads all FORTRAN95 boolean comparison operators and intrinsic mathematics functions. The comparison operators were written to only compare the real portions

of a number such that the CVM version of the CAM would take the same "path" as the original. The mathematics functions were written simply to compensate for the few intrinsic functions for which the compiler did not accept a complex variable input (e.g. `sign`, `log10`, etc.)

For simplicity, the program we wrote to generate CVM files from the original files converted all variables (but not parameters) to complex variables (unless the file contained custom tags to skip the conversion.) This was done in order to trade development time (i.e., identifying only the variables that required conversion) for computation time (i.e., every operation is now duplicated to operate on both the real component and the imaginary component.)

Again similar to the generation of differentiated code with TAPENADE, the bisections in the CVM code had to be adjusted in a similar manner as in the TLM version.

Using CVM allowed us to run the CAM with much smaller perturbations (typically 1×10^{-9}) and provided a very good base to evaluate TLM sensitivities.

3.3 TLM Evaluation Methods

Testing the implementation of the TLM was a paramount challenge. Similar to the methods used in implementation, testing involved dividing the processes up and testing each individually, but the main efforts were in implementing different testing methods, in creating test data sets to use, creating systems to keep track of results to help narrow down bugs in the code, and developing post-processing visualizers that would allow the user to quickly see a broad view of the results, or narrow in on a specific species in a particular grid cell.

In general, testing was done in two phases, gas phase chemistry and aerosol dynamics/thermodynamics. For simplicity, run flags were introduced throughout the CTM that allowed processes to be skipped via a configuration file. Furthermore, because AURAMS is very CPU intensive and can take several hours to simulate just a few hours, some subroutines were also modified to only process a subsection of the grid. This substantially reduced

run-time while still providing a realistic test case.

This chapter discusses all of the above, namely, the testing methods used, the post-processing tools developed to visualize the outputs and the systems used to track our results.

Phase 1: Testing Gas Phase Processes

The predominant method of testing gas phase processes was using the BFM as discussed in section 2.2.3.1. The typical strategy would be to run two versions of the code, one with one sensitivity parameter slightly perturbed, and one TLM version. The perturbation would be applied to a species initial concentration, or emission. Species to perturb were chosen by their underlying chemistry, e.g., perturbing NO_x emissions and observing the response in ozone concentrations.

The first test consisted of a perturbation of 10% to calculate a forward difference approximation. For gas phase processes, a perturbation of 10% provides sensitivities often very compatible with TLM sensitivities (compatibility on the order of 10^{-2} ppbv.) If the test yields favourable results, a smaller perturbation is performed and the results are re-compared.

Phase 2: Testing Aerosol Processes

As discussed in section 3.2.5, traditional methods of perturbing inputs with the BFM were found inadequate. Therefore, the primary method of validating TLM results was to compare them with CVM results.

3.3.1 Visualization of the Results

Originally all AURAMS visualizations were performed with EC's SPI application. SPI is a powerful visualizer written in TCL with a large feature set. SPI met the majority of our visualization requirements, but was slow to use and difficult to run on new platforms.

Though SPI can run user scripts that allow it to be automatic, it was thought to be easier to develop custom visualizers. The first generation of custom visualizers was developed in Java. This too however was too cumbersome and inflexible. The following generation of visualizers were PHP applications that used gnuplot to visualize results. The advantage of this was that results could be queried with one command, integrated into the online results management tool (discussed later in 3.3.2) and quickly updated to support new visualization schemes.

Several main plot types were developed; these are discussed here in detail and used in section 4. The first (figure 3.3) is a tile plot showing a brute force sensitivity, TLM sensitivity, the difference between the two, and a scatter plot.

The axis in figure 3.3 represents the grid cells being displayed in the figure. In this research, visualizations of the entire domain would contain the majority of North America (figure 2.5a) with 150×106 grid cells. For sensitivity figures, the top left figure will generally have a title such as "BF-10"; this indicates that the sensitivity shown was calculated using the BFM technique for a perturbation of 10%.

This figure is ideal for zooming in on model output and comparing two different methods of sensitivity calculation. Sometimes this figure is presented with the colourbar range limited to a z-score range of ± 2. Recall a z-score score is defined as the number of standard deviations a datum is from the mean. This is done to eliminate outlier results when appropriate. Any colourbar modification on a figure is clearly stated in the figures caption.

The plot in 3.3 is ideal for close inspection, but it does not give a broad view of the compatibility of the results. For this, a *radar* plot was developed that could inspect all species in a given layer. A sample of this plot is shown in figure 3.4. This plot is generated by calculating the coefficient of correlation between one method, say brute force, and another, say TLM, for each species at a specific time and layer. This plot type was a great first step for quickly finding problems in how sensitivity coefficients were calculated.

Figure 3.4 shows the correlation coefficients plotted radially from the centre. A point is plotted at a radial distance of 1 indicates perfect correlation. Using this plot, one can quickly

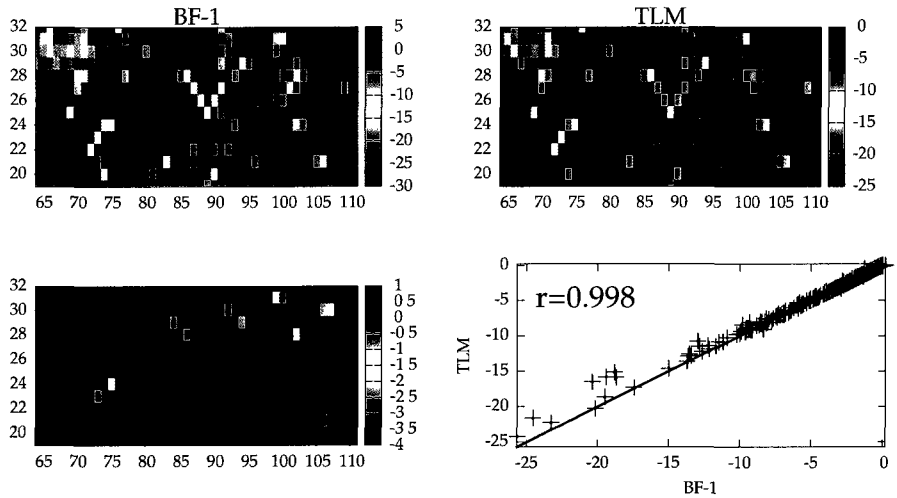


Figure 3.3: Sample sensitivity difference plot. This plot shows the sensitivity of ammonia (NH_3) to NO_x emissions in the surface layer after two simulation hours. The top left plot shows the brute force sensitivity, the top right shows the TLM sensitivity, the lower left shows the difference between the two, and the lower right shows the correlation between the two sensitivities.

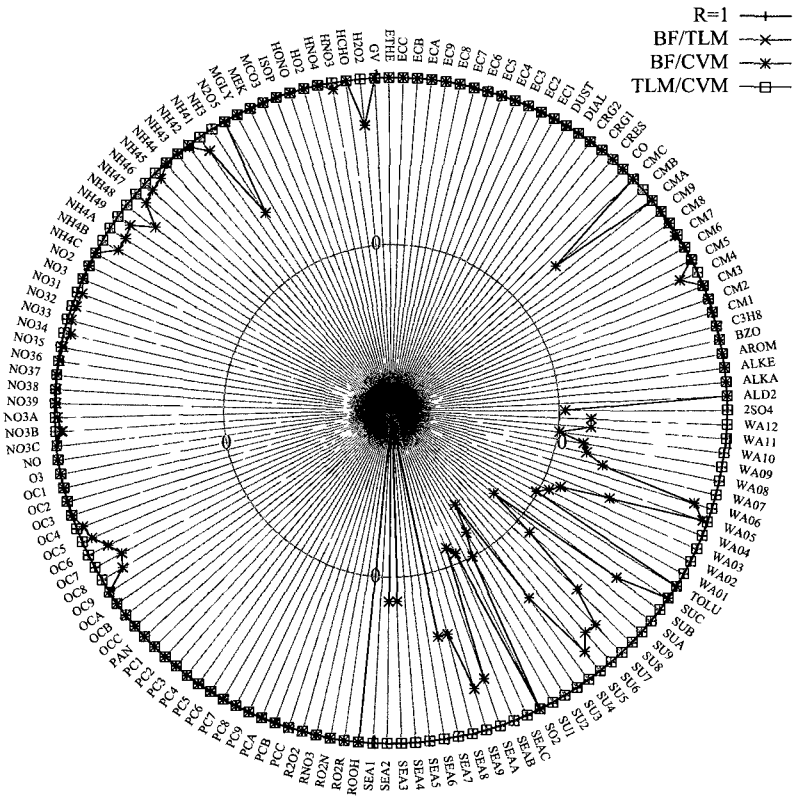


Figure 3.4: Sample sensitivity radar plot. This plot contains three data sets. Each data set is the coefficients of correlation of two methods of calculating sensitivity coefficients for each species at a given layer and time. i.e. The dark orange line shows how well BFM and TLM sensitivity coefficients correlate, the read line shows BFM and CVM and the blue line shows the correlation between TLM and CVM sensitivity coefficients. In this figure, one can see that brute force sensitivities are very incompatible with TLM and CVM sensitivities in the CAM.

see which species show good correlation for multiple methods of sensitivity calculation.

3.3.2 Result Management Tools

Testing AURAMS is a special challenge due to the size and run-time of the model. Efforts have been made to simplify the runs, either by creating branched versions of the code with fewer components (e.g., removing certain processes), and modifying the processes to only operate on interesting subsections of the grid (both vertically and horizontally.) These methods were very successful in shortening run times and isolating processes for testing.

These methods did not however address management of the results. For example, when testing the TLM implementation of process X and comparing it to sensitivities calculated with the BFM, for this several runs would be processed: a TLM run, a 10% perturbed run, perhaps a 1% perturbed run, and a -1% perturbed run. Some of these runs might fail due to user setup errors and would have to be re-run. One can see that testing of one process alone can generate a large amount of data. Furthermore, finding a result 6 months later proved almost impossible.

For this, an online web-based system was developed to manage all the results generated by AURAMS. The run script we developed for AURAMS now uploads all relevant run parameters into a PostgreSQL database via a web-service developed in PHP. This database is then accessible online, where a user can view all the runs in the database (figure 3.5), operate on the data (figure 3.6), and even visualize the results (figure 3.7); all in the comfort of an AJAX[1] enabled interface. All visualizations generated by the system or even off the system (e.g., in SPI, or in-house developed Java or GNUPLOT based tools) can be easily saved into and referenced from the database online.

This web-based system has contributed to this project by reducing redundant tests, eliminating data-download times for examining results remotely (e.g., when soliciting a second opinion, or working remotely), and provides a type of overview perspective of all our

[1] Asynchronous JavaScript and XML, this technology enables websites interfaces to be more akin to those found on Facebook than older static form based interfaces.

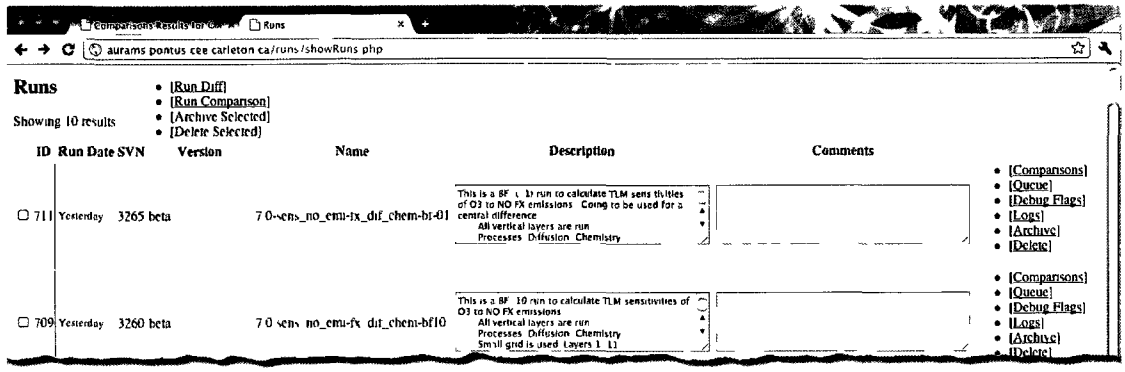


Figure 3.5: Example of the Results Management Tool. From this page, all runs that have been saved to the database (saving to the database is done automatically by the AURAMS run scripts) can be seen. Run options can be seen on the left, these include "Comparisons" to view all runs this run has been compared to (figure 3.7), "Queue" to view how many time steps the run was queued to run, "Debug Flags" to view the debug configuration file used in the run, "Logs" to view the run logs, and "Archive" and "Delete" to archive (no longer show by default on this page) or delete the run. The links at the top of the page operate on the data, using "Run Diff" a user can calculate brute force sensitivities between two runs, using "Run Comparison" a user was taken to a page to visualize and compare two runs (shown in figure 3.6

tests performed on AURAMS.

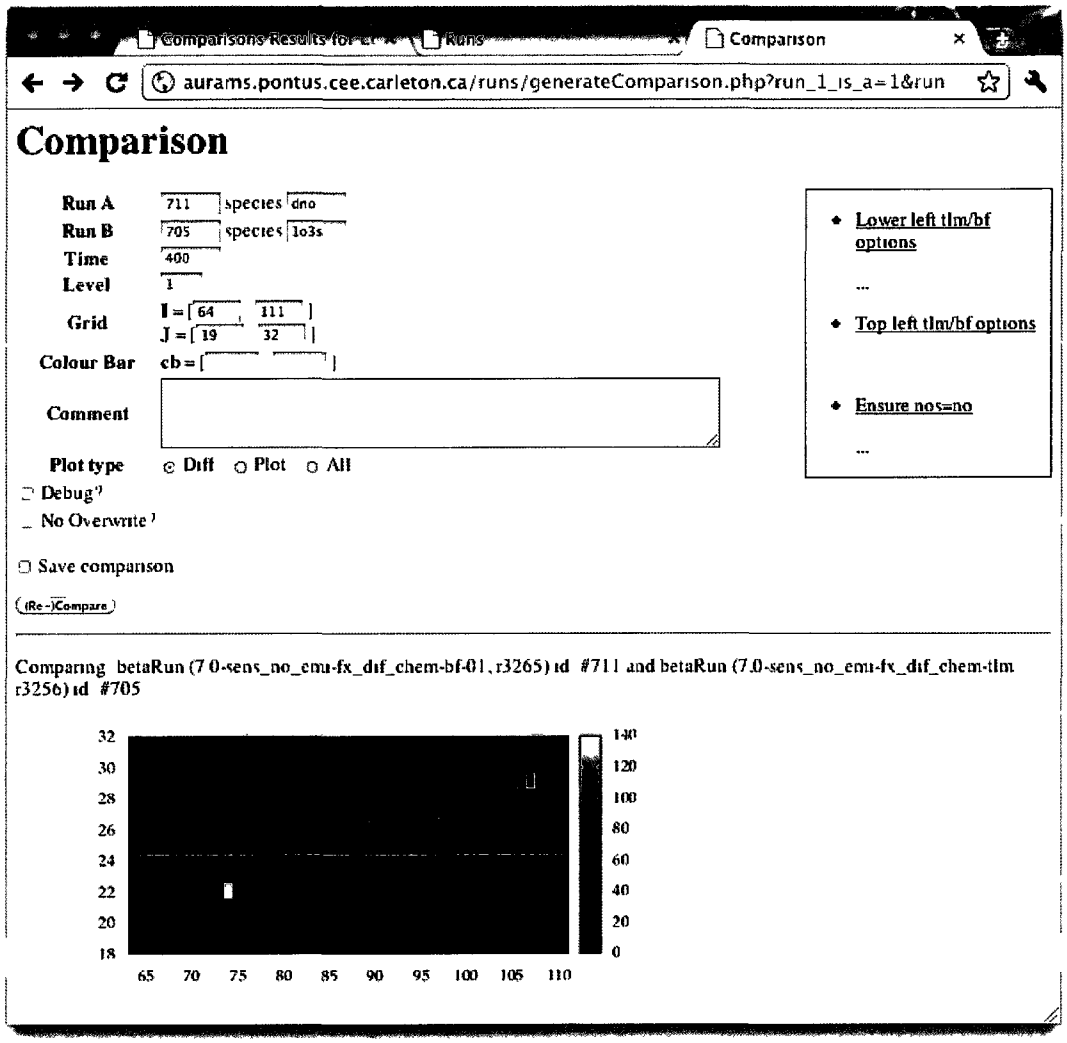


Figure 3.6: Result management tool visualizer. This tool allows a user to visually compare two runs. This site interfaces with the visualization tools previously developed, thus virtually any visualization that we could generate by other means may be generated using this site. The form provides intuitive options to zoom in on a sub grid at a certain layer, select different comparison types, and even save the visualization back to the database.

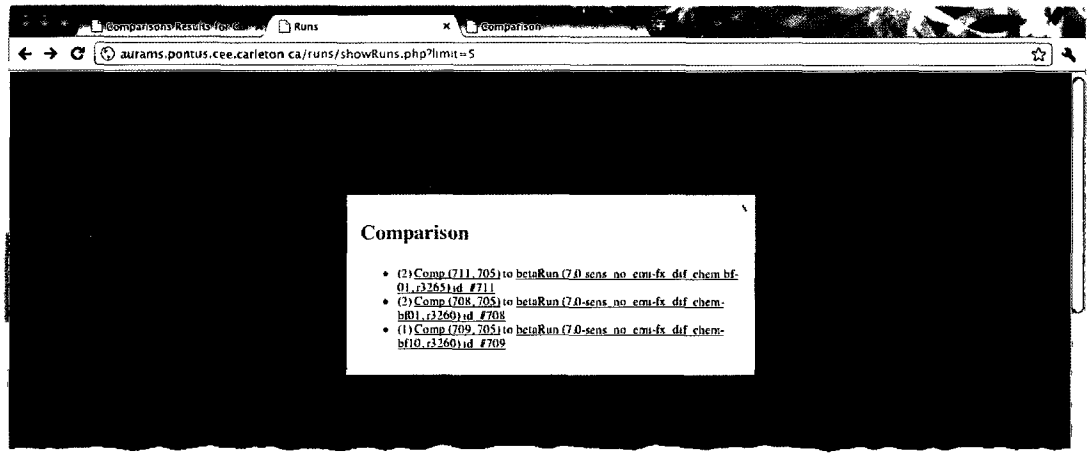


Figure 3.7: This figure shows how the user can select two-run comparisons. These links take the user to a site listing all the saved comparisons for the selected runs. This page is by nature too long to easily lend itself to a screen shot and is thus not showed here.

Chapter 4

Results

As discussed in chapter 3 the guiding philosophy for implementing and testing the implementation of the TLM was to divide and conquer. This chapter shows and discusses our results of this implementation for gas phase processes, aerosol processes and ends by comparing AURAMS TLM results to sensitivities calculated in CMAQ.

As mentioned in the previous chapter, we concluded that it was necessary to employ the CVM for evaluation of aerosol TLM results. However, brute force method appeared adequate (but maybe not ideal) for evaluation of gas phase processes. Because of this discrepancy that is discussed in sections 3.2.5, we have not devised a method for testing our TLM implementation with both gas phase and aerosol processes. Therefore all the results below are presented either for gas phase processes or aerosol processes. In this chapter first an overview of simulations is given and then the results of DDM evaluation are presented. We finish this chapter with a comparison between sensitivity fields calculated using two different CTMs in AURAMS and CMAQ. Furthermore, all figures in this chapter show semi-normalized sensitivity coefficients (section 2.2.)

Table 4.1: AURAMS vertical layer heights.

Layer (k)	Height (masl)
1	0
2	15
3	55
4	120
5[a]	195
6	285
7	390
8	510
9	650
10	815
11	1,010
12	1,235
13	1,500
14	1,810
15	2,170
16	2,590
17	3,085
18	3,660
19	4,335
20[b]	5,125

[a] Typically shows the highest response to PS emissions

[b] Vertical layers after layer 20 are omitted as no results for content in layers higher than 20 are presented.

4.1 Model Setup

AURAMS in this work was configured to use a secant polar stereographic grid projection of 42x42 km grid cell and 28 vertical layers. The true latitude of this grid configuration is 60° north. All AURAMS runs shown here were performed with a zero gradient boundary condition (*zero_grad* in configuration file) set. Furthermore, all figures in this chapter show semi-normalized sensitivity coefficients (section 2.2.)

The inputs used in the modelling scenarios were provided by Environment Canada. The meteorological inputs were generated using GLOBAL ENVIRONMENTAL MULTISCALE (GEM) for the duration of the BORDER AIR QUALITY STUDY - METEOROLOGY (BAQS-MET) campaign in summer 2007. The emission data were generated in SMOKE using the 2000/2001 Canadian and US emission inventories. A zero-gradient boundary condition was used for simplicity. AURAMS was run for July 1st and 2nd, 2007. For brevity, often the figure titles contain vertical layer numbers, i.e. "k=4" for vertical layer 4. Actual layer heights are referenced in table 4.1.

4.2 Evaluation of Gas Phase TLM

The predominant method of testing the gas phase processes in AURAMS was to compare the TLM sensitivities to BFM sensitivities, as is the common procedure (Dunker, 1984; Yang et al., 1997; Hakami et al., 2003). As various plots and results in this chapter indicate, in general, TLM and BFM provide results that are very compatible.

In the following plots and tables, BF-x refers to BRUTE-FORCE (BF) sensitivities that are calculated from x% increase in the sensitivity parameter p. For gas phase processes, unless otherwise mentioned, a one-sided forward difference approximation was used. One may also use centred approximations from negative and positive perturbations to produce more accurate BF estimations of local derivatives.

Results in the figures below are sometimes for the entire domain, and sometimes for a

subset of the domain. The domain chosen in each figure is dependant on two factors. i) run time ii) whether the entire domain was required. The latter is subject to the individual processes. To elaborate, the following processes were modified in the following ways in order to reduce run times.

Diffusion in AURAMS is a vertical column operator. We modified it to operate on a subset of the horizontal grid.

Chemistry in AURAMS operates on each cell independently of any other cell. We modified chemistry to operate on a sub-volume (cubic) of the domain in order to reduce run times.

Advection was not modified to operate on a smaller domain as it requires the complete extent of the domain.

Aerosol processes were initially modified to accept fewer vertical "slices", however no result in this thesis that involves aerosol processes was performed on a sub-domain.

Therefore, many of the results in the figures below do not include advection, or in other words, advection was either evaluated separately or was added last in a series of evaluation runs.

The small grid is the same in every figure. The specific sub-domain was chosen as it contains emissions of all sources supported by AURAMS - therefore it provided as a good domain for the majority of species and processes.

The first evaluation of the TLM implementation in AURAMS was performed on chemistry. Figure 4.1 compares TLM sensitivities to BF-10 sensitivities for ozone to initial ozone concentration without any transport, emission or aerosol processes. This was done by perturbing the initial value of ozone. Ozone was chosen as it has a fast evolution through time and thus shows a strong signal after a short simulation period. It is also the most important gas-phase species of concern. The figure shows only the surface layer response. This was chosen because the harmful effects of ozone exposure are highest at the surface layer, and

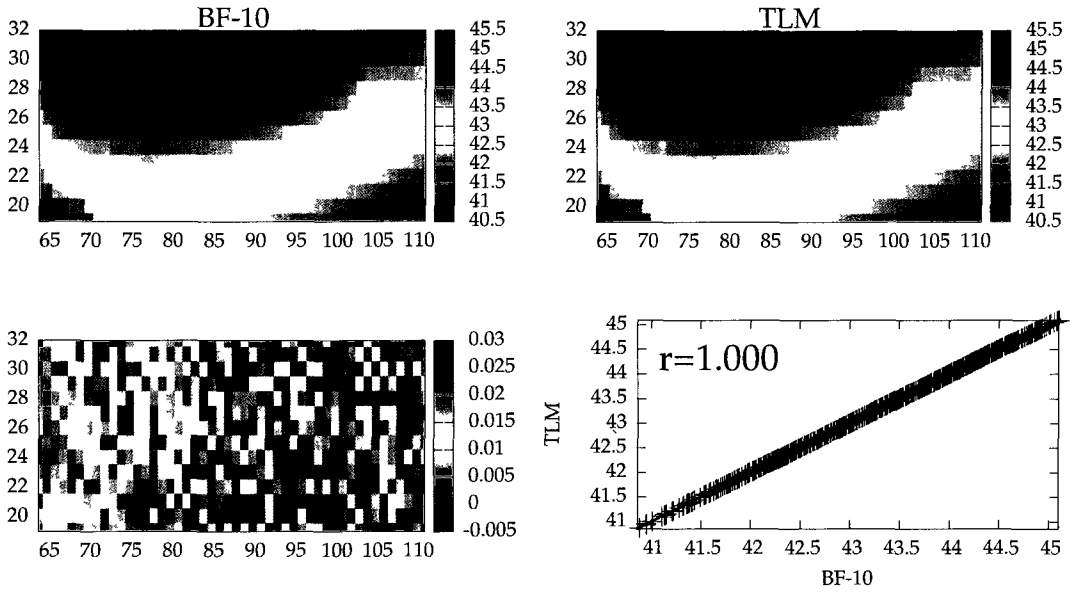


Figure 4.1: Ground level sensitivity of ozone concentration to initial ozone with chemistry after four simulation hours.

because no emission or transport processes are involved, ozone sensitivities at each point are not being affected by adjacent cells or boundary conditions. The two methods of calculating sensitivities shown in figure 4.1 are very compatible. The agreement compares favourably with reported TLM versions of other 3-D CTMs (Dunker et al., 2002; Hakami et al., 2004).

The result shown in figure 4.1 shows that the TLM implementation of chemistry is properly implemented.

Confident that the TLM implementation of chemistry was working, the next test added NO major point source emissions to the simulation. PS emissions have their own separate process for being injected into the domain. Figure 4.2 shows the sensitivity of NO to NO emissions over different layers after 4 simulated hours. Several layers were chosen because NO major PS emissions represent emissions such as factory smoke stacks, and are thus injected into higher layers than the surface. Note that major PS are likely to create a NO_x rich environment resulting in negative sensitivities.

The results in figure 4.2 show that the TLM implementation on the major PS emission processes are properly implemented.

Figure 4.2 only shows the response to major POINT SOURCE (PS) emissions. Other sources, such as biogenic, mobile, and minor PS emissions (hereinafter referred to as FLUX SOURCE (FS) emissions) are injected into the concentration field in AURAMS as surface fluxes via boundary condition in the vertical diffusion subroutine. Therefore, before simulating sensitivity to all emissions, first diffusion was tested.

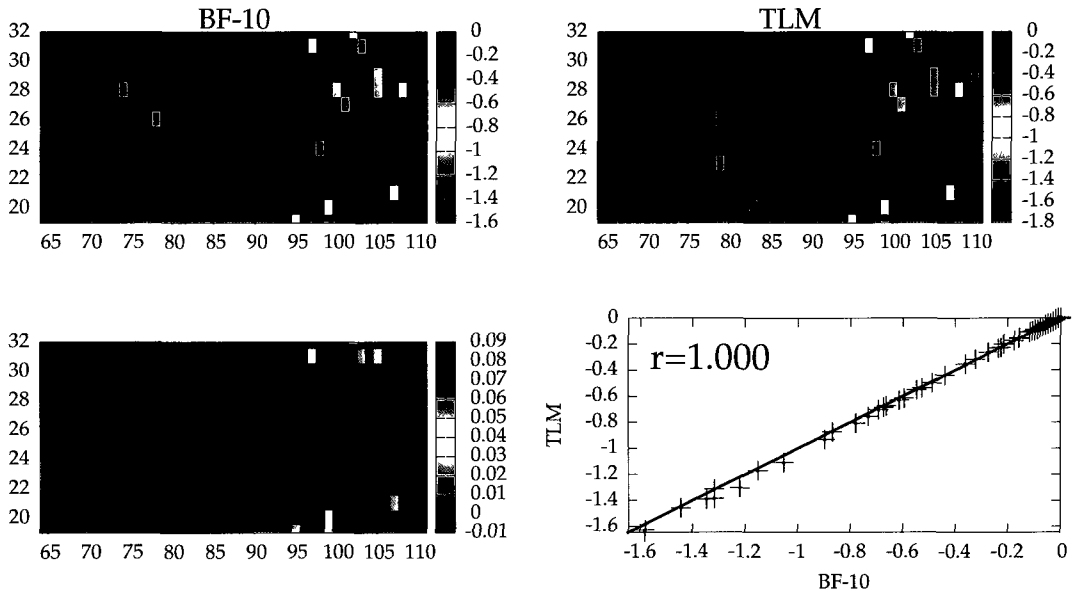
Isolation of the effects of the diffusion was accomplished by shutting off all other processes (advection, chemistry, emissions, aerosols) and running diffusion on ozone. Figure 4.3 shows sensitivity of ozone to initial ozone after four simulation hours with only vertical diffusion as the active process. The error shown in figure 4.3 remains within the same range after 18 simulated hours. Longer simulations for this specific process were not performed. As expected, all sensitivities are positive as more initial ozone results in more final ozone after vertical diffusion.

The results in figure 4.3 show that the TLM implementation on the diffusion process functions properly.

Figures 4.2 shows that sensitivity to major PS functions properly, figure 4.3 shows that the diffusion process functions properly. The combination of the two processes, i.e. diffusion and major PS emissions is shown in figure 4.4. Figure 4.4 shows the sensitivity of NO to major PS emissions of NO with only the diffusion process operating. This figure shows the that the combination of diffusion and major PS emissions was properly implemented for the TLM.

Adding chemistry to the simulation in figure 4.4 yields the results in figure 4.5. The BF-10 and TLM comparisons are weaker in this case, therefore further investigation was done comparing the TLM to BF-01, and then to BF-01 performed with a central difference. Figure 4.5 shows these three tests in reverse order (best to worse.) It can be seen that - as expected - agreement decreases as the perturbation is increased, and the accuracy of the derivative (central to forward) is decreased. We suspect that the reduced agreement

Ozone sensitivity to NO PS emissions with (k=1)



Ozone sensitivity to NO PS emissions with (k=5)

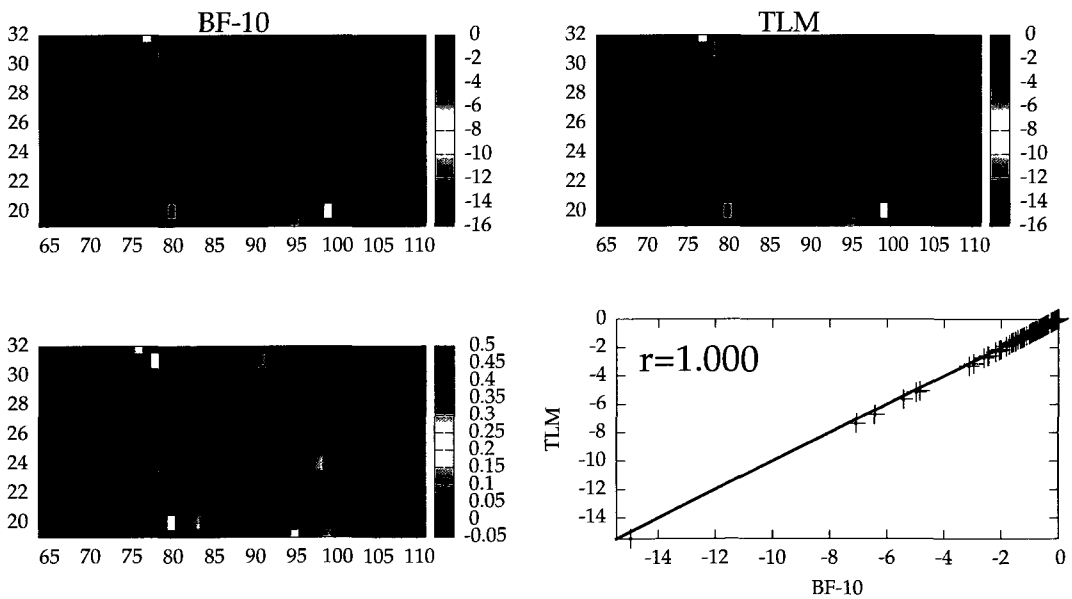


Figure 4.2

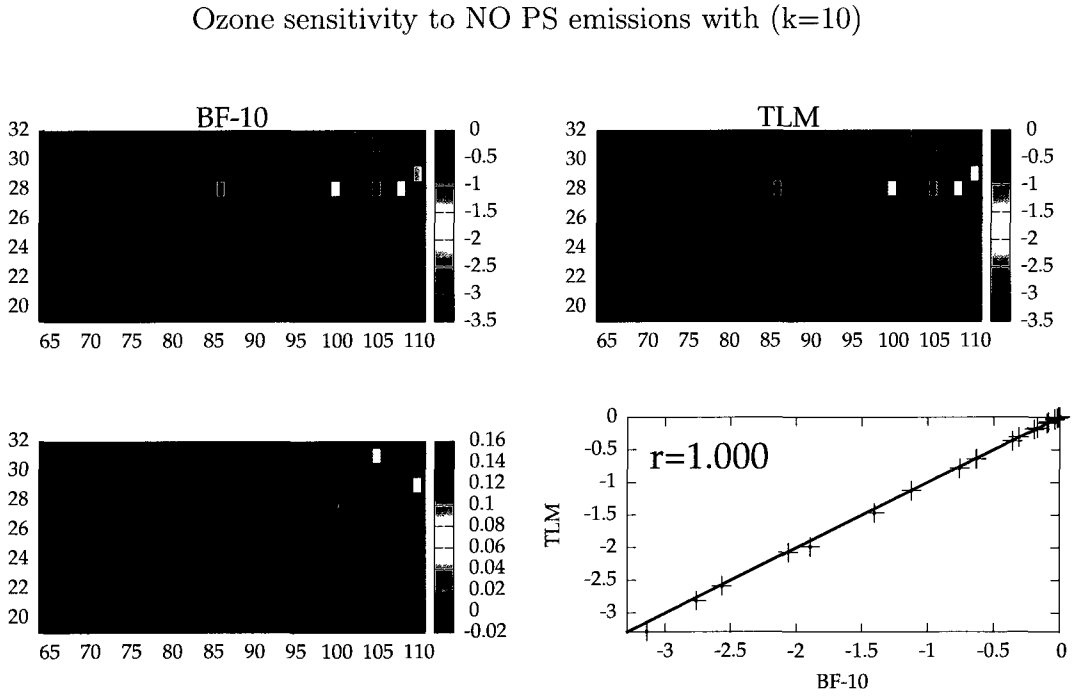


Figure 4.2: Sensitivity of ozone to NO PS emissions without transport after 4 simulated hours.

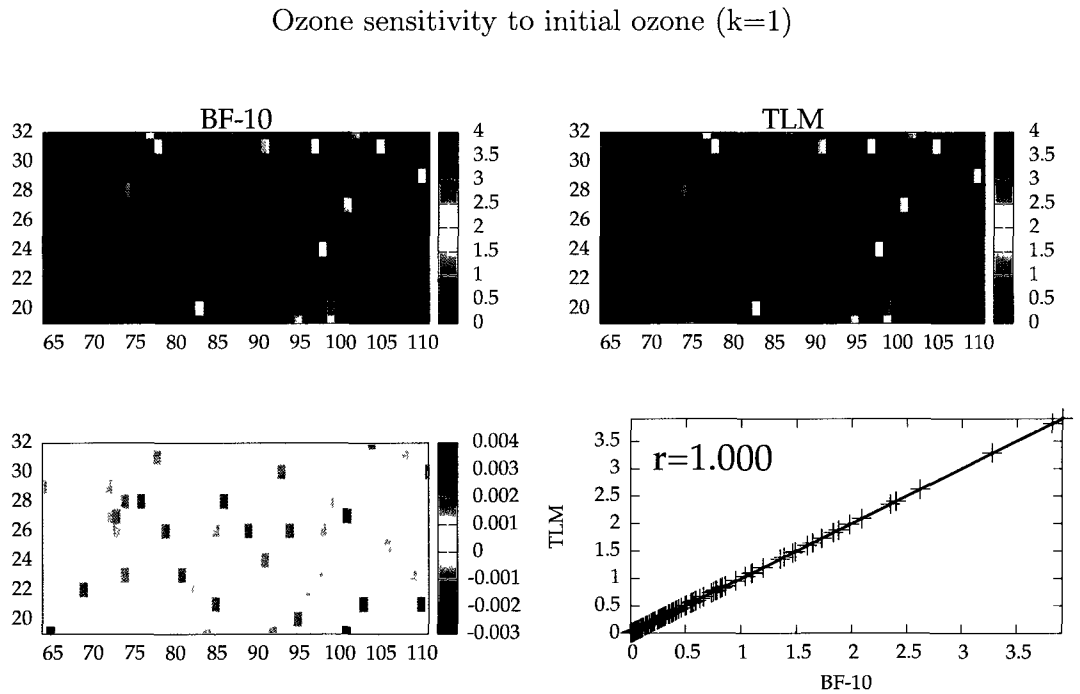


Figure 4.3: Sensitivity of ozone concentration to initial ozone with diffusion after four simulation hours.

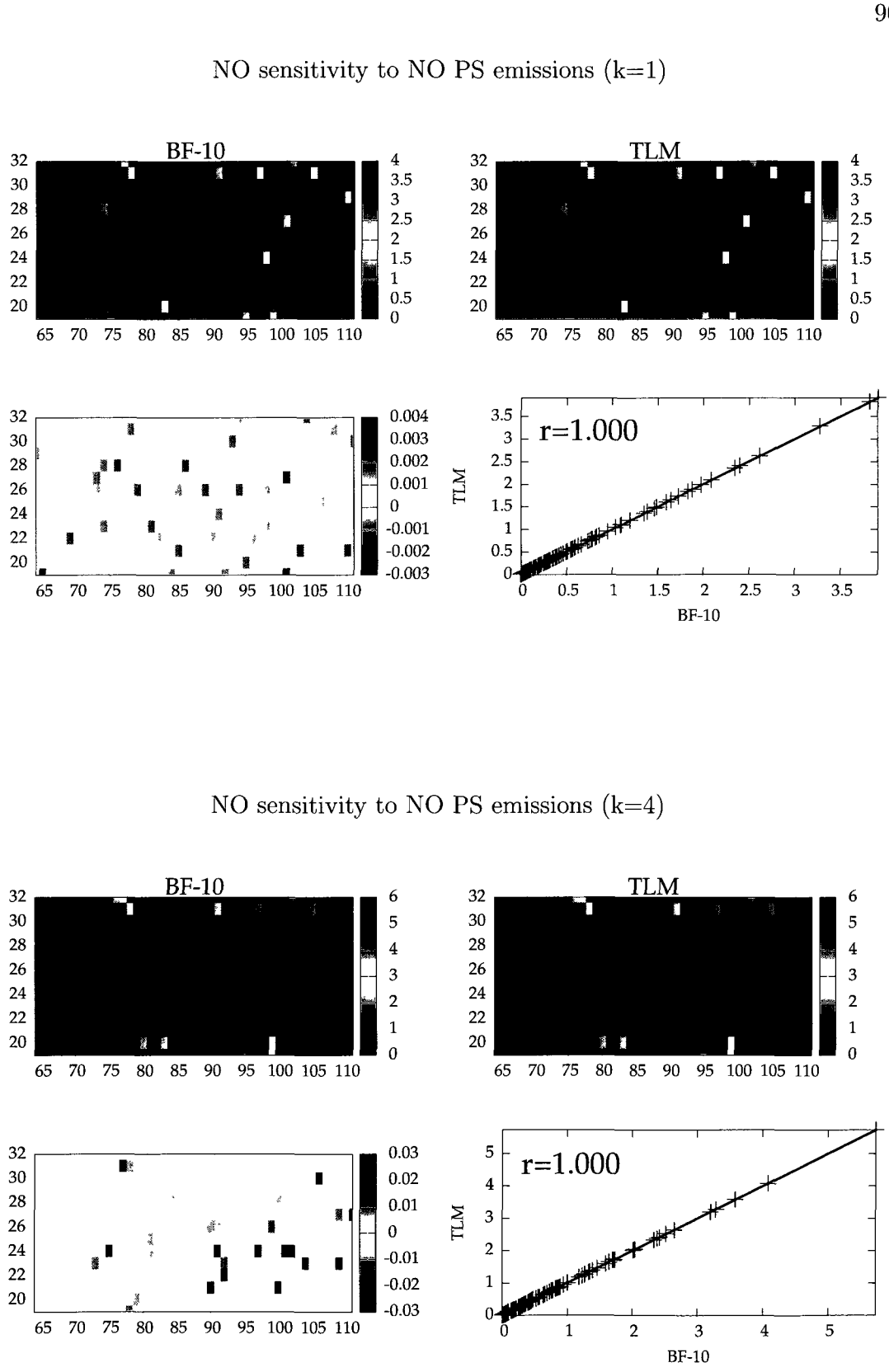


Figure 4.4: Sensitivity of NO to NO PS emissions with diffusion after 4 simulated hours.

is a product of the very large emission values, and that the agreement would continue to improve with smaller perturbations, likely a perturbation in the range that can only be performed with the CVM method.

Figure 4.5 shows that the agreement remains consistent in layer 4.

Due to this reduced agreement, all future results that include FS emissions and chemistry will show this same decrease in compatibility.

As discussed earlier, FLUX SOURCE (FS) emissions, unlike PS emissions, are injected via the diffusion routine. To test this we ran a simulation with the chemistry solver (figure 4.2), diffusion (figure 4.4), and FS emissions and examined the response of ozone to NO FS emissions. The result is shown in figure 4.5 for the surface layer and the layer immediately above it after a simulation period of four hours.

The only remaining gas phase process left to test that is not involved in any of the above results is advection. Advection, unlike the other processes, could not be modified to operate on a small grid. Figure 4.7 shows the sensitivity of ozone to initial ozone with the advection operator at the surface level. Ozone was chosen due to its very heterogeneous initial condition and therefore has a strong signal. Figure 4.7 shows that the TLM implementation of the advection process is properly implemented.

BF-30 sensitivities for the simulation shown in figure 4.7 were also calculated. The BF-30 results, though still compatible to the TLM sensitivities, were - as expected - less than those of BF-10. The agreement between BF and TLM also deteriorated for 1% perturbations, most probably due to numerical errors in the solver. For very small perturbations, the difference between the two concentration fields used in BF approximation becomes comparable in size with the level of numerical error tolerated in the solver. As a result, BF approximations become inaccurate for very small perturbations as well. The lower level for accurate BF perturbations is typically somewhat lower than 1%. This could indicate that the default tolerance levels in the *Rodas-3* need to be set at a tighter level.

Figure 4.8 shows ozone sensitivity to NO emissions with all (except aerosol) processes. This figure shows good agreement between BFM and TLM sensitivity fields. The error plot

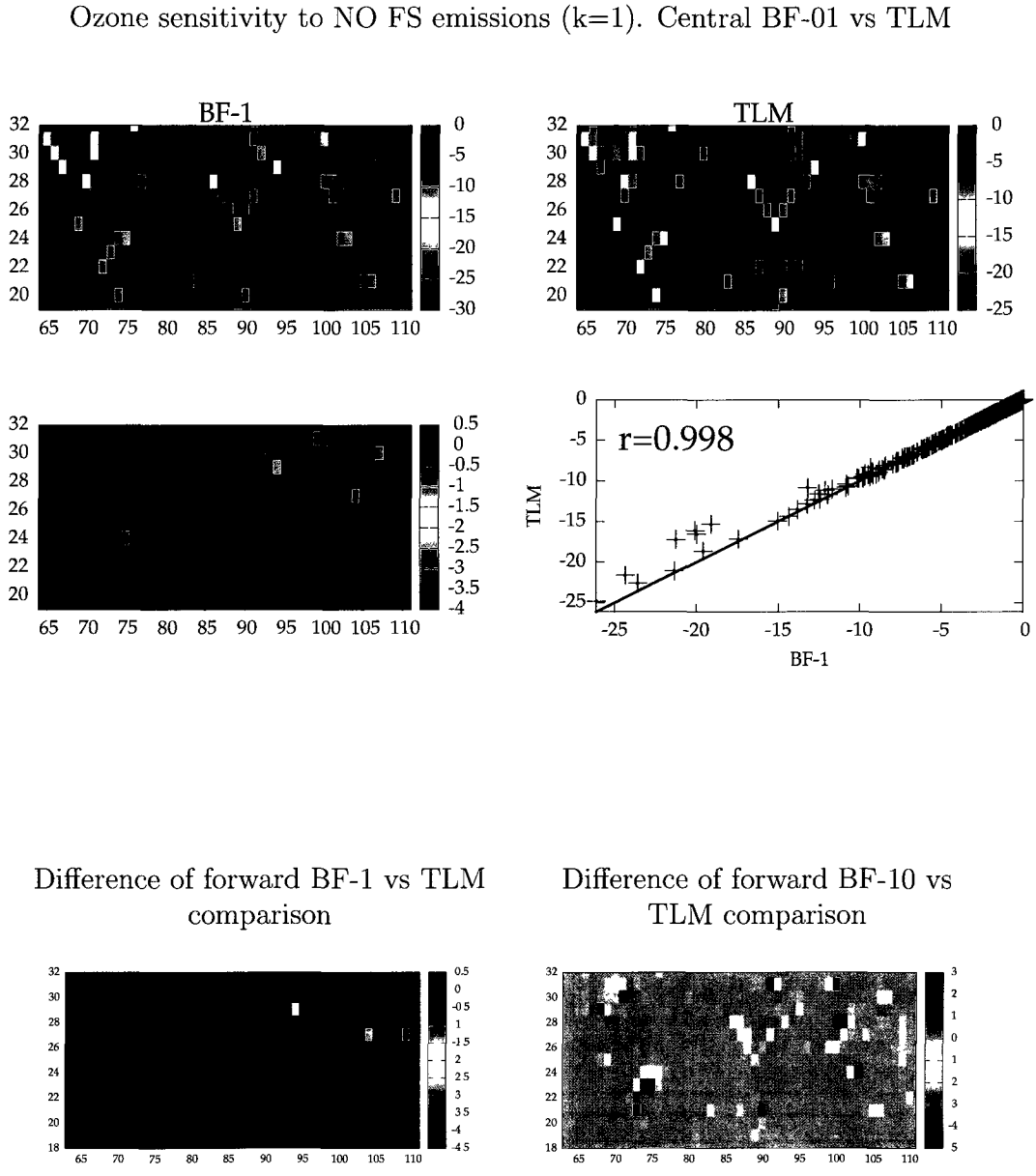


Figure 4.5: Ozone sensitivity to NO FS emissions with diffusion after 4 hours of simulation. Shown in this figure are the surface level sensitivities (top) using a 1% central difference perturbation, and analysis of the BFM/TLM agreement using different perturbation sizes and differential schemes (bottom.)

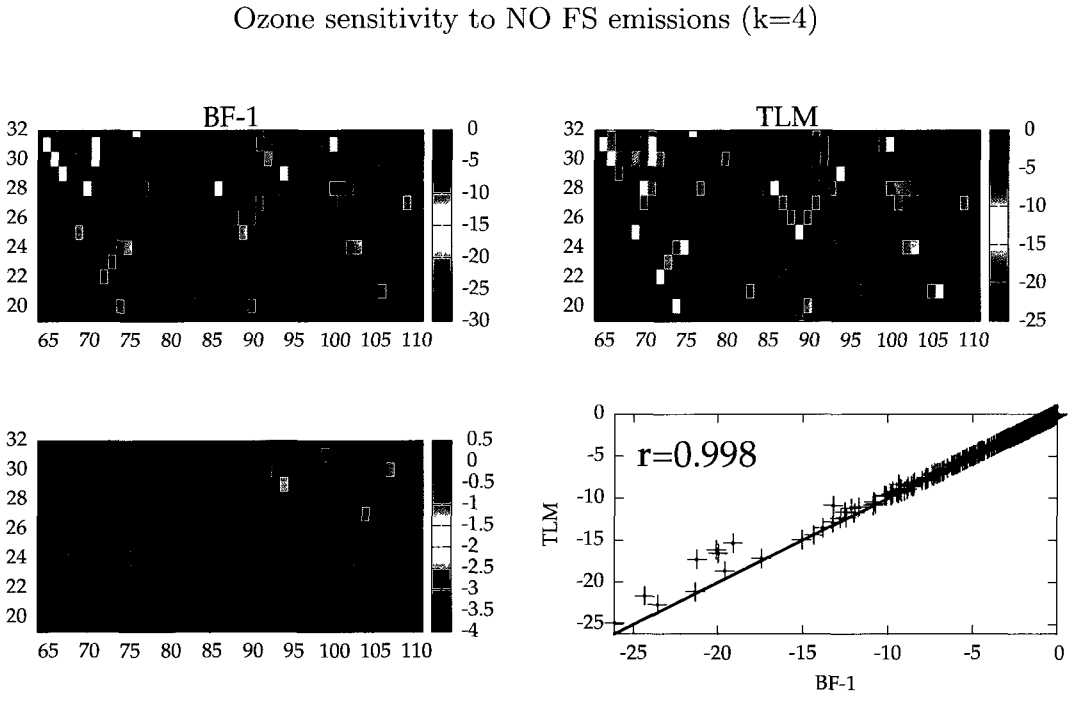
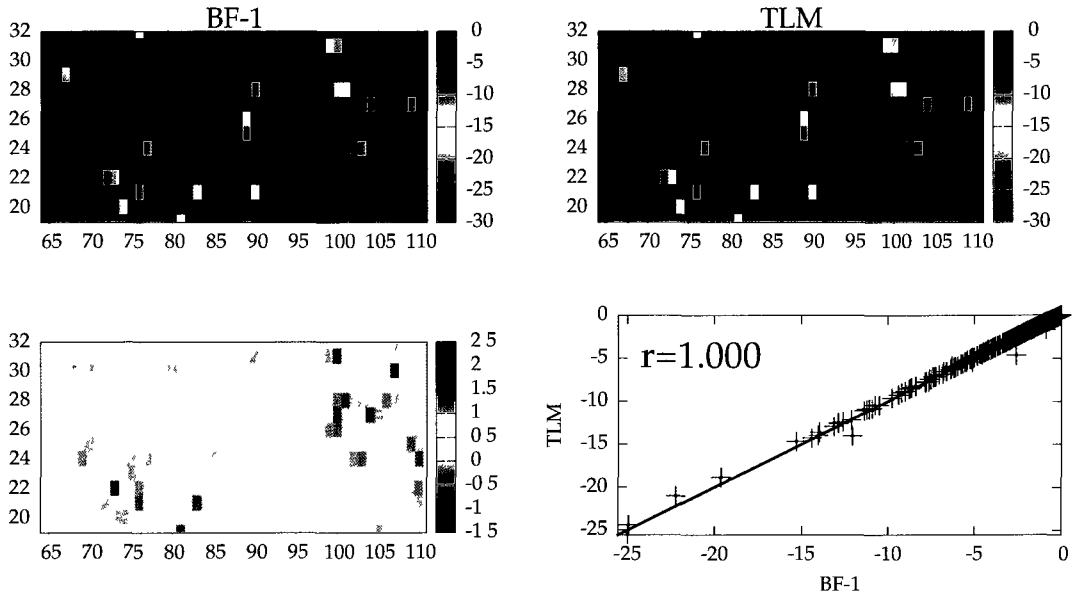


Figure 4.5: Ozone sensitivity to NO FS emissions with diffusion after 4 hours of simulation. Fourth level.

Ozone sensitivity to no emissions without advection (k=1)



Ozone sensitivity to no emissions without advection (k=5)

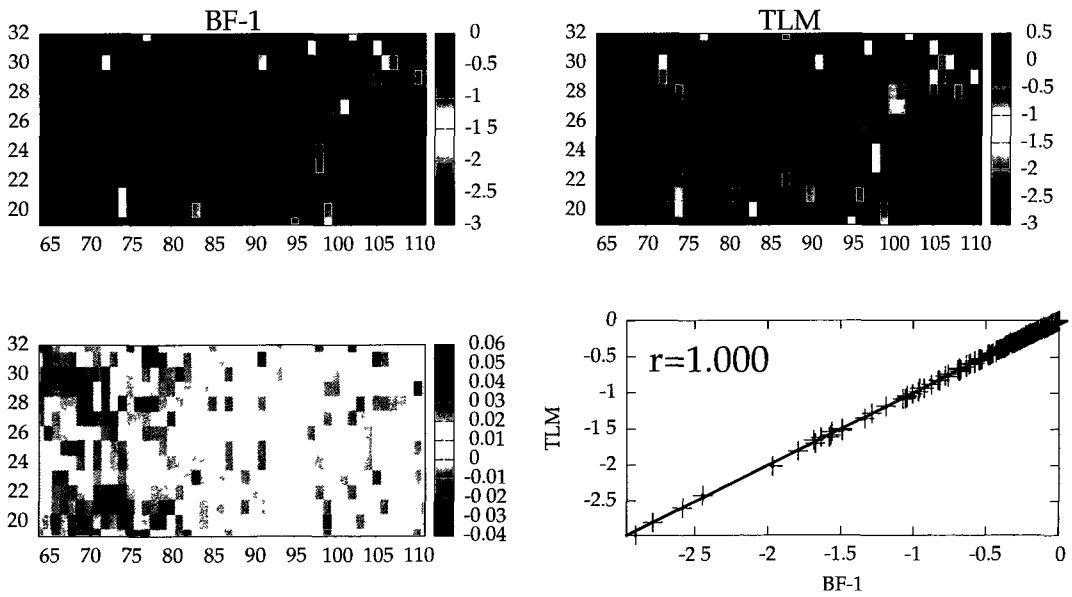


Figure 4.6

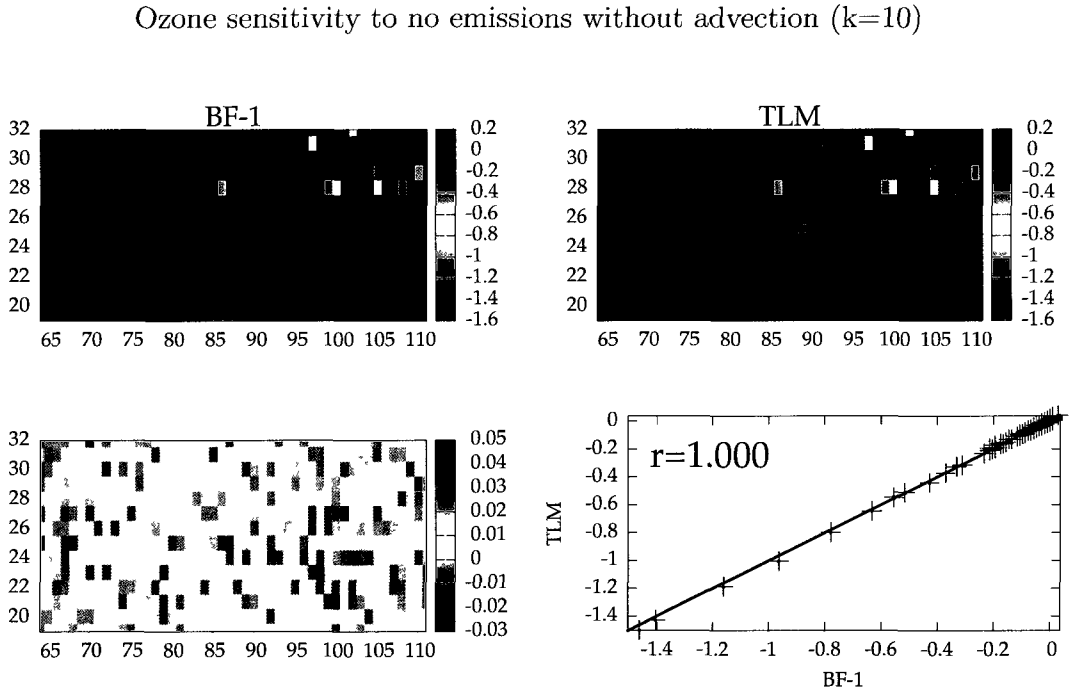


Figure 4.6: Ozone sensitivity to all NO emissions without advection after 2 hours of simulation.

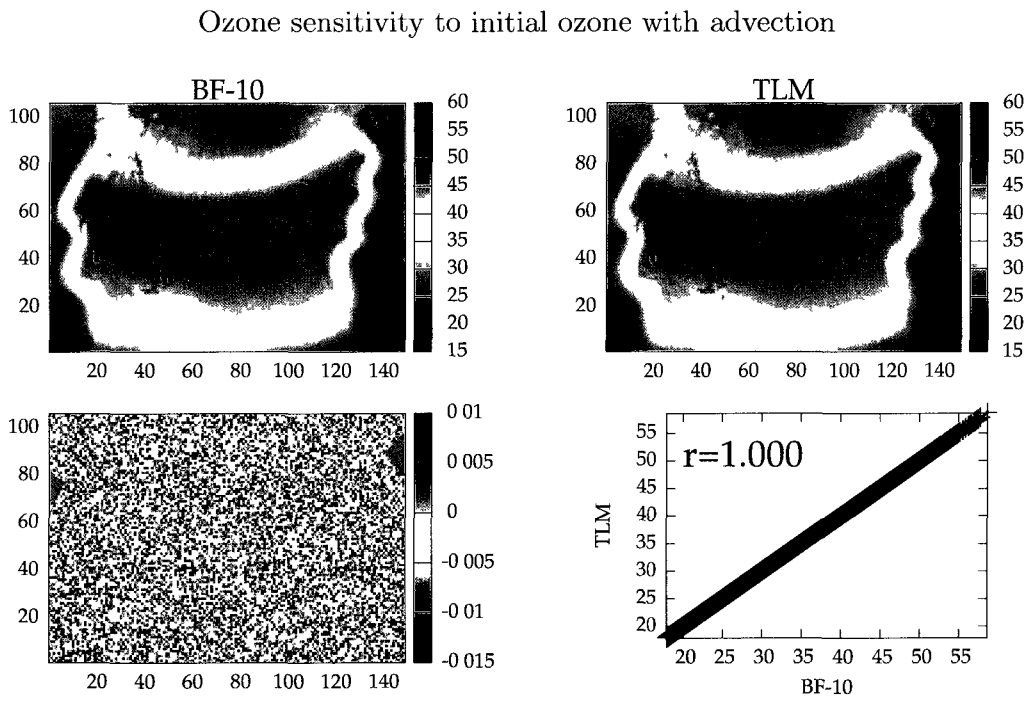
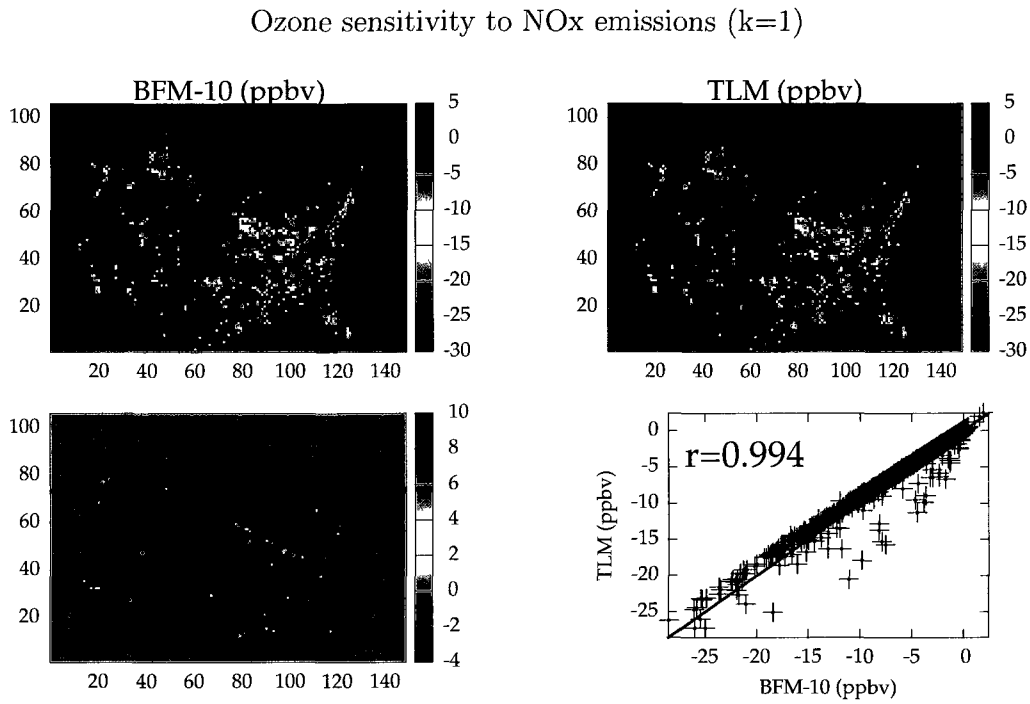


Figure 4.7: Ground level sensitivity of ozone concentration to initial ozone with advection after two simulation hours.



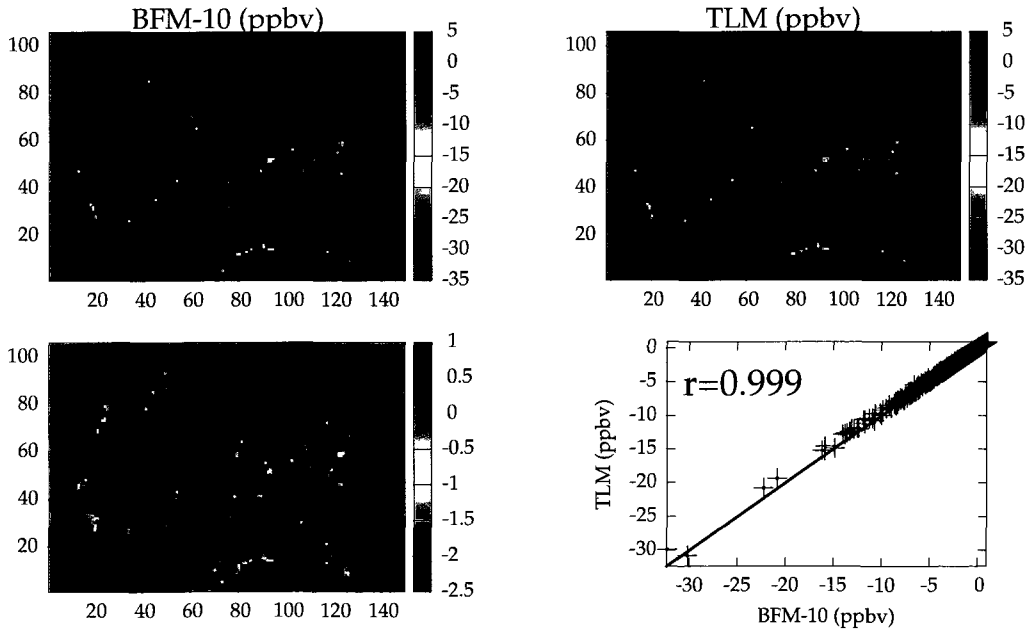


Figure 4.8: Ozone sensitivity to NOx emissions with all processes (except aerosols) after 4 hours of simulation.

Table 4.2: Summary of gas phase processes in AURAMS.

Test	Processes	Figure
O_3 sens. to initial ozone	Chemistry	4.1
O_3 sens. to NO PS emissions	Chemistry	4.2
O_3 sens. to initial ozone	Diffusion	4.3
NO sens. to NO PS emissions	Diffusion + PS Emissions	4.4
O_3 sens. to NO FS emissions	Chemistry + Diffusion + FS Emissions	4.5
O_3 sens. to NO emissions	Chemistry + Diffusion + Emissions	4.6
O_3 sens. to initial ozone	Advection	4.7
O_3 sens. to NO emissions	Advection + Diffusion + Emissions + Chemistry	4.8

at the lower left of figure 4.8 shows the same range as the error in figure 4.5, and as such it is the FS emissions that are suspected of contributing this error.

For clarity, table 4.2 lists the tests performed and the relevant figure number.

4.3 Evaluation of TLM implementation in the Canadian Aerosol Module (CAM)

Figure 4.9 shows the correlation coefficients of the three methods used in this research to calculate sensitivity, namely, BFM, TLM and CVM. This figure illustrates the poor compatibility of the BFM to the TLM sensitivity coefficients compared to the CVM. Aerosol processes contain sharp non-linearities that cannot be captured by the BFM – the perturbation is too large or is too small to be distinguishable from numerical noise. For this reason, all sensitivity comparisons in this chapter are given as TLM compared to CVM. All CVM sensitivities in this chapter were performed with a semi-normalized perturbation of 10^{-9}. This perturbation was scaled down even further for processes that use double precision.

To evaluate the TLM of CAM, the CVM and TLM sensitivities of NH_4, NO_3 and SO_4 in all 12 bins were obtained with respect to various gas and aerosol species. These species were chosen as they are processed by most aerosol operators and have a better understood chemistry compared to other species such secondary organic aerosols.

Pairwise comparisons were conducted for the surface layer throughout the whole computational domain and demonstrated by scatter plots below. Note that for brevity, many of the titles are shown in a form such as "NH_4, k=18, t=3." This is short form for the sensitivity of total NH_4 (the sum of all 12 NH_4 bins), at vertical layer 18, after three time steps to some parameter given in the figure caption. Actual layer heights are given in table 4.1.

Figures 4.10 to 4.12 show the compatibility between CVM and TLM sensitivities for a perturbation in all species. It is important to note that a perturbation in all species has no physical meaning and is only used to here as an initial test to ensure a sensitivity response for all species. Figures 4.13-4.15 again show the comparison of the two methods but with a perturbation made in only NH_3.

Overall, the two methods yield very similar results, as observed in figures 4.10 to 4.15,

where the red crossings represent CVM and TLM sensitivity pairs for all grid cells at the given layer, and the black line represent one-on-one line. Results from single-step runs show in general a good agreement between CVM and TLM. For three-step runs, though in some cases there are several points off the line, the overall accuracy remains good (figures 4.12, 4.15)

The results in this section were chosen as they represent the response of all species throughout the entire domain. Because of the number of figures, most individual discussion on each figure is given in the caption.

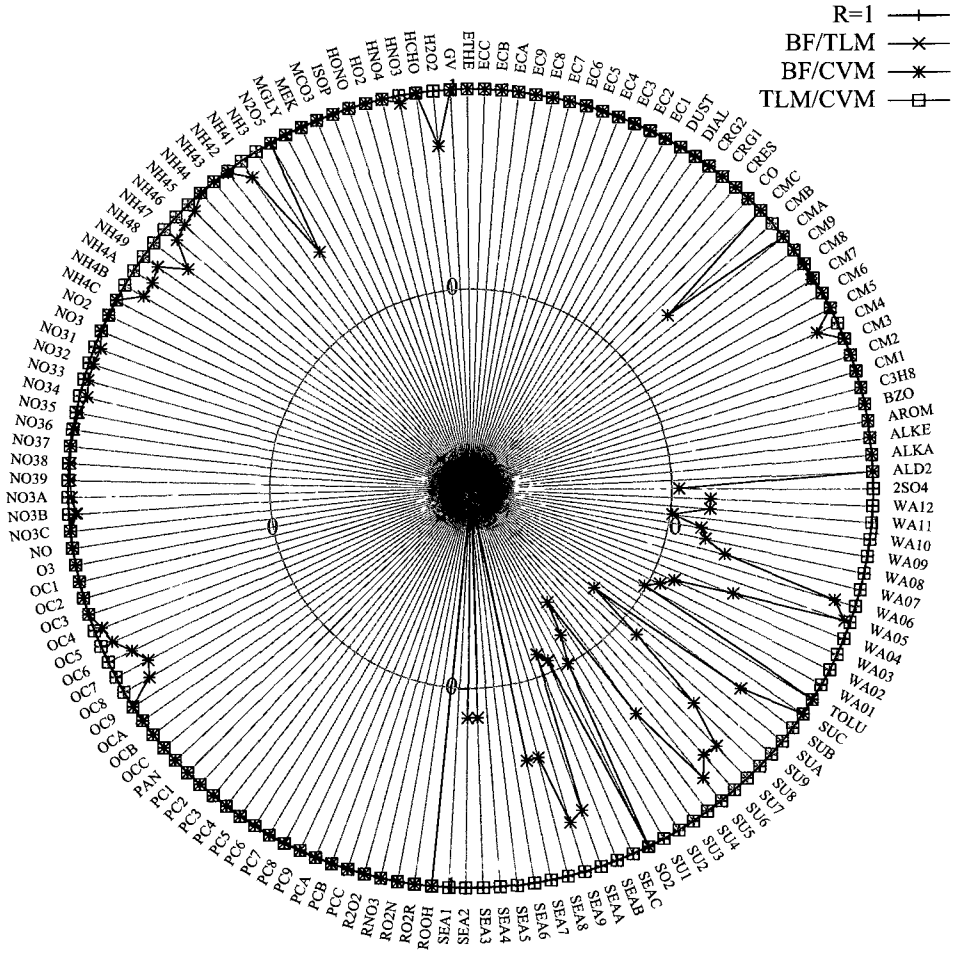


Figure 4.9: This radar plot shows that that the TLM/CVM sensitivities are very compatible for all species when running all aerosol processes in the CAM after one time step. The sensitivity coefficients in this figure are for an initial perturbation in all species.

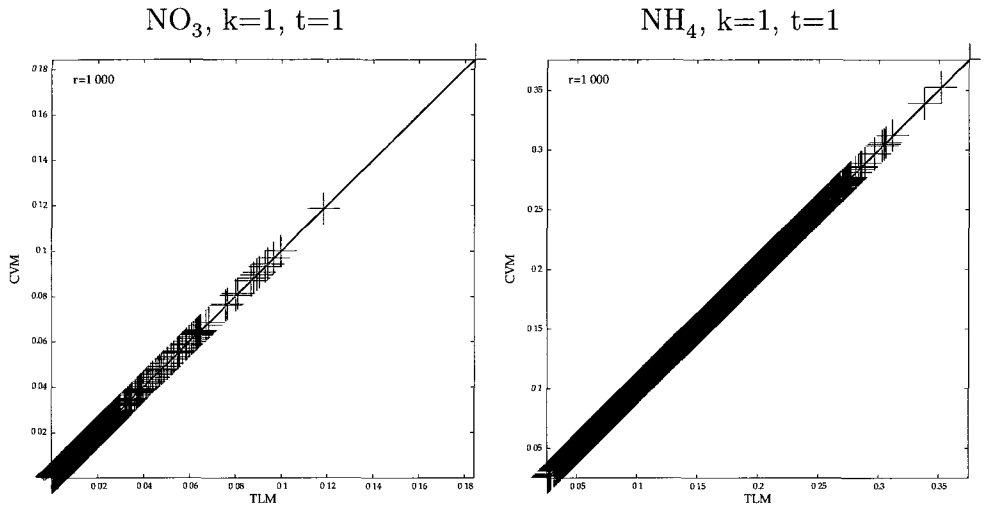


Figure 4.10: TLM/CVM sensitivities for NO_3 and NH_4 to an initial perturbation in all species at the surface layer for one time step.

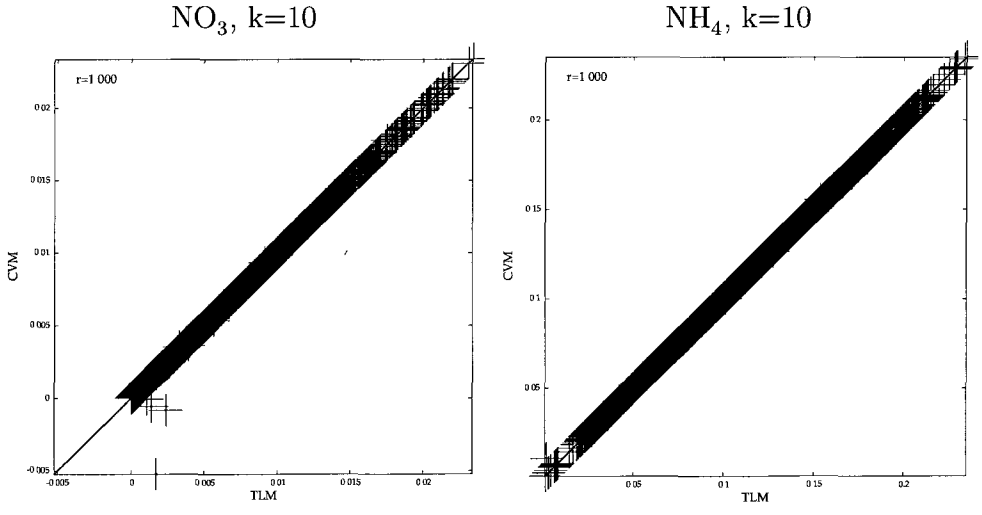


Figure 4.11: TLM/CVM sensitivities for NO_3 and NH_4 to an initial perturbation in all species at layer 10 after one time step for a perturbation in all species.

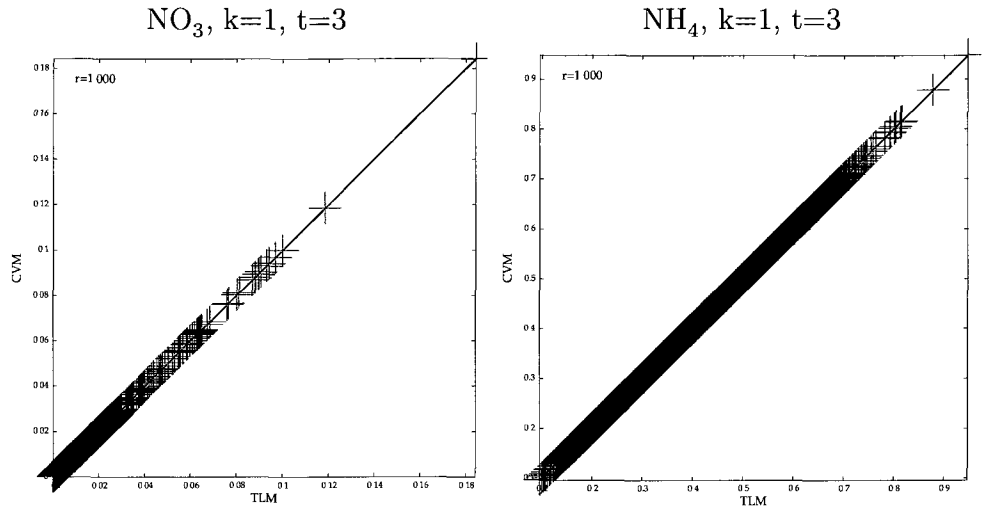


Figure 4.12: TLM/CVM sensitivities for NO_3 (sum of all bins) to perturbation in all species at surface layer after three time steps.

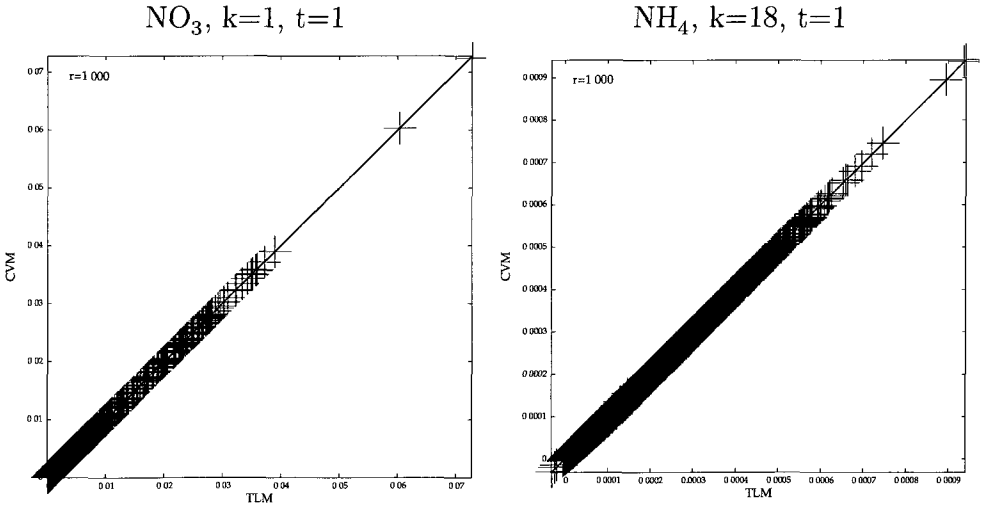


Figure 4.13: TLM/CVM sensitivities for NO_3 (sum of all bins) to perturbation in NH_3 after one time step.

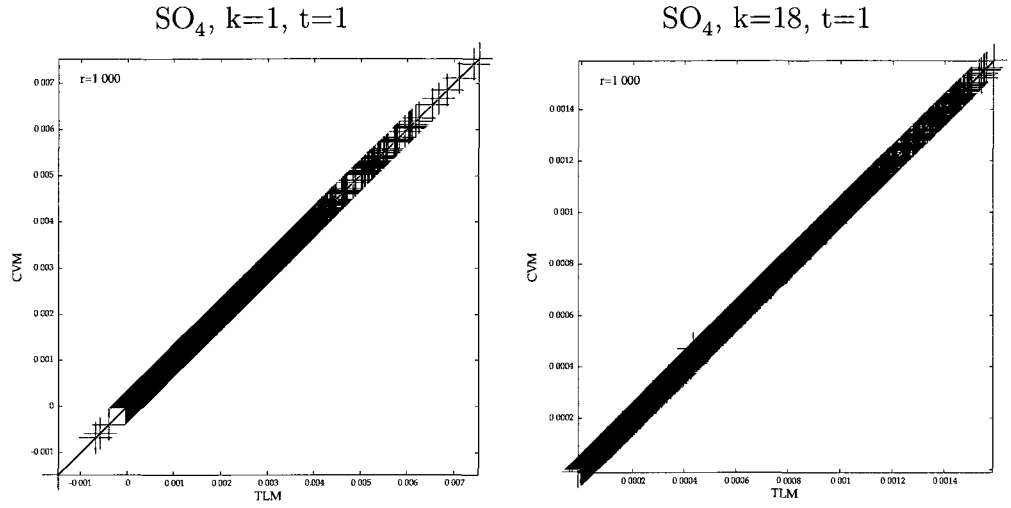


Figure 4.14: TLM/CVM sensitivities for SO_4 (sum of all bins) to perturbation in NH_3 after one time step.

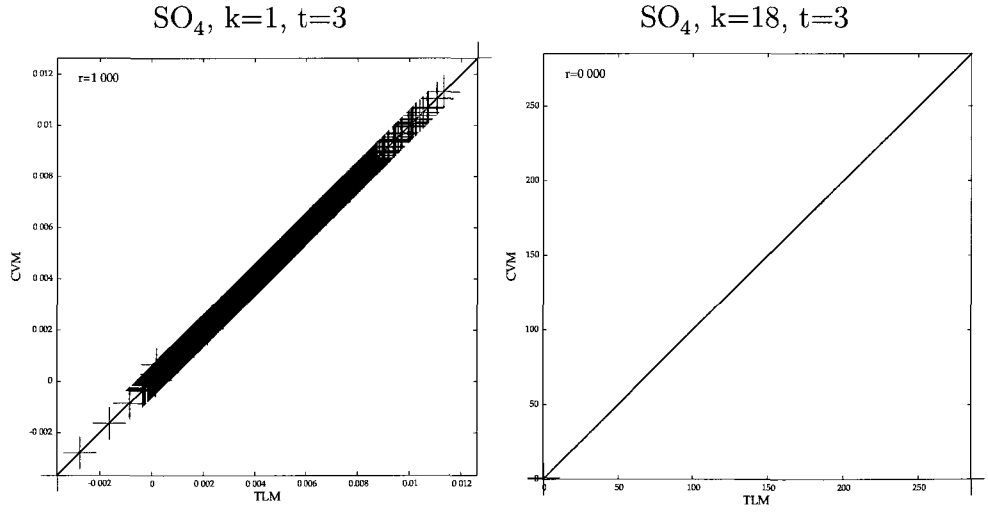


Figure 4.15: TLM/CVM sensitivities of SO_4 (sum of all bins) to perturbation in NH_3 after three time steps, layer 18.

Figures 4.10-4.14 show excellent comparison. These figures represent the compatibility of the TLM and CVM sensitivities throughout the majority of the model. There are however points that on the surface appear to be incompatible. Figure 4.15 investigates the response of total sulphate to a perturbation in NH_3 at layer 18 after 3 time steps. There is a single point of disagreement (TLM appears to be wrong) which we have not been able to resolve. Though this peculiarity remains under investigation, we hypothesise that this error could be the result of applying stress to the CAM by running it repeatedly without the host model.

4.4 Comparison of AURAMS and CMAQ Sensitivity Fields

AURAMS like many other CTMs has been subject to various model inter-comparison. In such studies concentration predictions from two or more models are compared. Inter-model comparisons are performed to increase confidence in participating models. In model inter-comparisons, only concentration fields are compared, and furthermore, generally an effort is made to make models as comparable as possible by coordinating inputs, approaches, meteorology, chemical mechanisms, etc.

In this section we compare both concentration and sensitivity fields from AURAMS and the US EPA's CMAQ (Byun & Schere, 2006). Furthermore, we intentionally make no effort to make the two models comparable as we claim those differences can be considered as a crude realization of modelling uncertainty. We aim to test the hypothesis that models of different origin and varying inputs are more consistent in sensitivity fields than concentration fields. This conjecture is somewhat expected as sensitivities are inherently fields of differences, and therefore, likely to be free from systematic errors in concentrations fields (the bias.)

The two models used in this comparison are AURAMS[1] as described in 4.1 and CMAQ

[1]The version of AURAMS delivered to us in 2009 contained an error where wind fields were effectively being doubled, this issue was spotted and fixed at ENVIRONMENT CANADA on July 20,

4.5.1. CMAQ, like AURAMS, is an off-line CTM. In this simulation CMAQ was configured to run on a Lambert projection of the REGIONAL PLANNING ORGANIZATION (RPO) domain with a 36 km resolution for 13 vertical layers.

CMAQ was chosen because although the suite of process parameterization is similar to AURAMS, the individual parameterizations are different in every instance. AURAMS and CMAQ are thus independent descriptions of the processes governing atmospheric ozone as well as other species (Moran, 2010).

The precise purpose of the simulation is to compare ozone sensitivity to NO emissions. Both models were restricted to gas phase processes only. At the time of this comparison, the CMAQ DDM module was not implemented so BFM sensitivities were calculated. This was considered acceptable because of the high compatibility between BFM and TLM typical in gas phase processes.

The meteorological and emission inputs were derived from two separate sources. Again, this was done to make the comparison more realistic as the motivation of this study is to compare results of two models despite differences in parameterizations, grid projects, and even input source.

CMAQ is initialized with a uniform (profile) field for ozone, thus for this comparison CMAQ was "spun-up" for two simulation days before the day to be compared in order to establish a realized ozone field. AURAMS fortunately has a spatially variant initial ozone field and thus was not spun-up.

In order to compare sensitivities calculated by AURAMS and CMAQ, we developed software to copy AURAMS output onto the CMAQ grid. This allowed us to compare our results on a point-by-point basis. The actual process of projecting AURAMS data onto the CMAQ Lambert domain involved writing subroutines that could transform x,y coordinates into *lat/lon* coordinates[1]. The program that projects the domain loops through

2010. This fix was ported into our version of AURAMS before running this simulation.

[1]EC provided us a routine developed in John D. Henderson in 1975 that could transform x,y coordinates to lat/lon coordinates for a polar stereographic projection, we inverted the routine to be able to transform lat/lon coordinates into x/y coordinates. This routine is available at

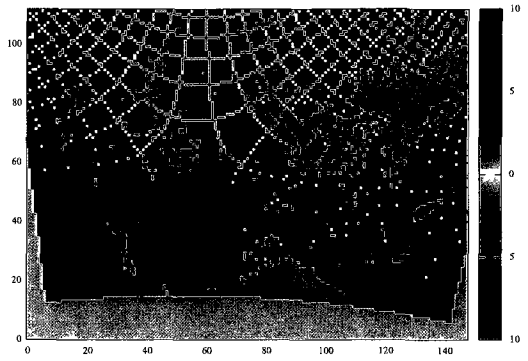


Figure 4.16: This figure is an example of imposing the AURAMS grid - a polar stereographic projection - onto the CMAQ grid - a Lambert projection. One can see artificial lines where grid points could not be matched.

all AURAMS grid cells and uses the transformation routine to query for a corresponding coordinate in the CMAQ domain. In this way, the AURAMS data is "*pushed*" down onto the CMAQ domain. This ensures only points within the receiver domain are transferred, e.g. points that do not exist in the receiver domain, (e.g., points in Newfoundland, the arctic, or Mexico), are not projected. Because of the different grid sizes, certain cells (approximately 700 grid cells per layer) have duplicate mapping. Hence, this homomorphic mapping can neither be considered one-to-one or onto. The effect of this is illustrated in figure 4.16. For the purposes of this comparison, unmatched grid points are not considered in any calculated statistics of the comparison.

Figure 4.17 shows a comparison of ozone concentrations after 23 simulation hours. The motivation before 23 hours is that because the simulation time was limited to a day, it was decided to sample concentration as late in the simulation as possible. The actual limits on the difference plot in 4.17 suggest that the concentrations do not compare well in magnitude, but the scatter plot suggests a decent spatial comparison.

The sensitivity results shown in figure 4.18 below are ozone sensitivity to NO emissions.

`http://www.cccma.ec.gc.ca/data/grids/geom_llfij.shtml`

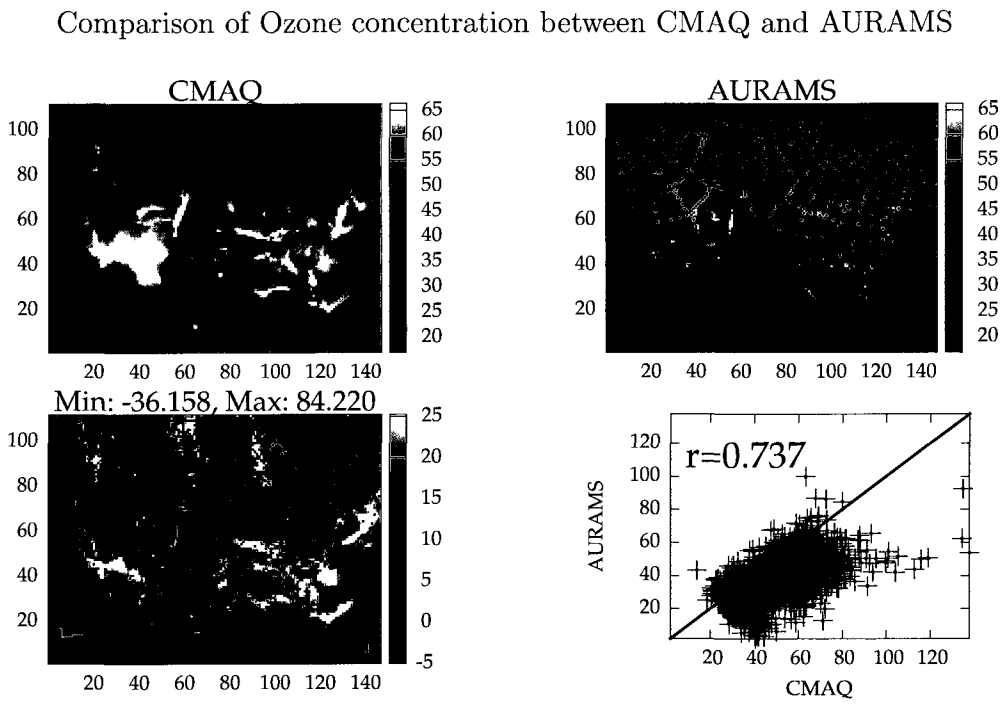


Figure 4.17: Comparison of ozone output from AURAMS and CMAQ after 23 simulation hours (showing 6 pm EST) at the surface layer (0 to 50 meters above surface.) Colour bar extremes set within a z-score of ± 2.

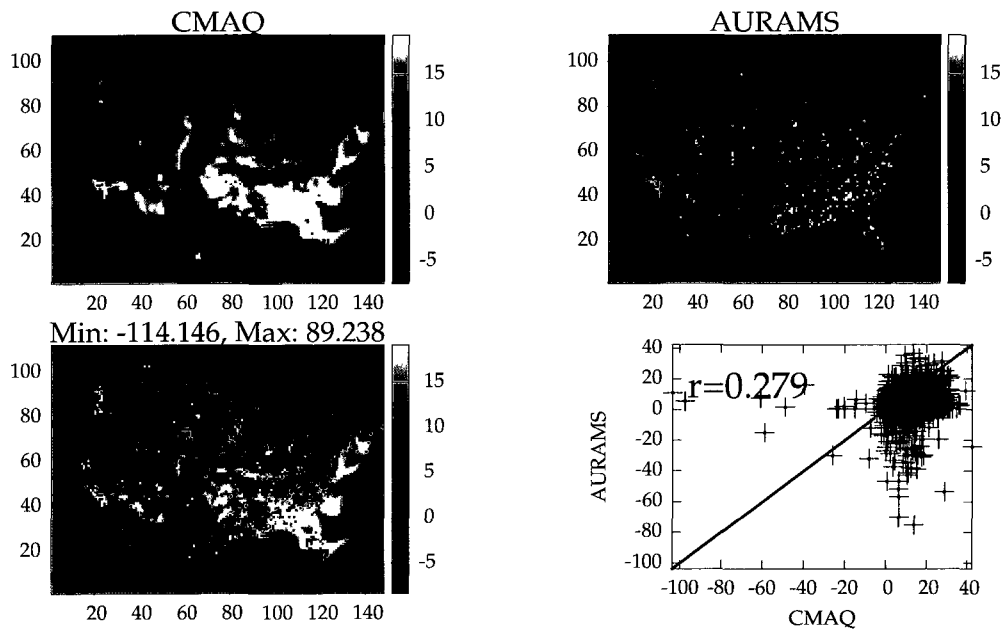


Figure 4.18: Comparison of ozone sensitivity to NO emissions from CMAQ and AURAMS after 23 simulation hours (showing 6 pm EST) at the surface layer (0 to 7 meters above surface.) Colour bar extremes set within a z-score of ± 2.

CMAQ sensitivities were derived using a 5% brute forced central difference, AURAMS sensitivities were calculated using the TLM implemented by this research and discussed throughout this document. Both the extremes of the difference plot and the scatter plot suggest disagreements.

Limited results presented above show that our hypothesis cannot be confirmed. Although due to short duration of simulation a fully developed sensitivity response is not formed, but the results seen here suggest that a better metric for quantifying compatibility between two fields (concentrations and sensitivities) may be necessary.

Chapter 5

Conclusion

The TANGENT LINEAR MODEL (TLM) was implemented into AURAMS in order to compute local first order forward semi-normalized sensitivities to various parameters. This implementation was applied to all gas phase and particulate phase processes. The implementation supports computing the sensitivity of concentration to multiple parameters simultaneously. KPP was used to generate a chemistry routine for the forward model including direct sensitivities using the *Rodas-3* solver for the ADOM-II chemical mechanism. TAPENADE was extensively used to generate the TLM version of the CANADIAN AEROSOL MODULE. The latter process however involved considerable post code generation manual modifications due to issues of insufficient numerical precision for derivative values.

The primary method of validating the TLM implementation in the gas phase processes was to compare the TLM sensitivity coefficients against BRUTE-FORCE METHOD (BFM) calculated sensitivities coefficients. A perturbation size of 10% often yielded excellent compatibility. Diffusion with flux emissions was the exception to this, thus a perturbation size of 1% is shown. The compatibility significantly increases with this smaller perturbation but is still not as compatible as the other processes. It is speculated that smaller perturbation sizes than the BFM can use would provide increased compatibility. For this, we recommend that in future work the CVM method be applied for gas phase processes.

The TLM implementation in the CAM was tested primarily with the CVM. This method

allowed us to use perturbations small enough to "follow" the same narrow control pathways as concentration. The compatibility between TLM and CVM calculated sensitivities were excellent for all species through the entire domain with the exception of layer 18 which has contained one anomalous outlier point. The compatibility remained high when running for multiple time steps.

To our surprise, regardless of the initial complex perturbation to the concentration field when calculating semi-normalized CVM sensitivity coefficients, after many time steps (on the order of 40) of the CAM the truncation error grew to the point of influencing the real portion of the concentration field resulting in mass inconsistencies. Investigation into this issue continues, however preliminary results suggest that the CVM, though a powerful tool and excellent for comparison in the first few time steps, may not be an appropriate tool for long simulations.

As of this writing, the TLM implementation has not been tested for both gas and aerosol processes simultaneously. Such a test would require that the CVM be implemented in the gas phase processes. This is both possible and is highly recommended for future work.

To date, AURAMS is only the second active TLMs to have a full TLM implementation in gas and aerosol phases. Our implementation in AURAMS is considerably more accurate than the other implementation (in US EPA's CMAQ.) Overall, we believe AURAMS TLM could play a significant role both as a policy evaluation tool and as a building block in development of instrumented Canadian CTMs.

For future work, we suggest that this study be extended in the following ways:

- Port the current implementation into the more recent versions of AURAMS and possibly into GEM-MACH
- Use AURAMS TLM for evaluation of long-range transport of pollution
- Extend the TLM implementation to calculate higher order sensitivity coefficients.
- Perform a long-range transport study using the TLM implementation in AURAMS to

investigate the performance of the TLM over extended simulation periods.

- Use high order gas-phase AURAMS TLM for construction of ozone isopleths across Canadian cities using higher resolution grids.

Development on AURAMS TLM is ongoing. Some of the proposed works above have already been planned and will happen as a continuation of this work. This thesis should be thought of as a snap-shot in time of the progress of this research.

List of References

Bartholomew-Biggs, M. (1998). Using forward accumulation for automatic differentiation of implicitly-defined functions. *Computational Optimization and Applications*, *9*(1), 65–84.

Basrur, S. (2000). *Air Pollution Burden of Illness in Toronto.* Toronto, Ontario, Canada: City of Toronto, Community and Neighbourhood Services, Toronto Public Health, Health Promotion & Environmental Protection Office.

Bell, M., & Davis, D. (2001). Reassessment of the lethal London fog of 1952: novel indicators of acute and chronic consequences of acute exposure to air pollution. *Environmental health perspectives*, *109*(Suppl 3), 389.

Box, G., Hunter, W., & Hunter, J. (1978). *Statistics for experimenters: an introduction to design, data analysis, and model building.* 111 River Street, Hoboken, NJ 07030: Wiley New York.

Boylan, J., Odman, M. T., Wilkinson, J., Russell, A. G., Doty, K., Norris, W., et al. (2002). Development of a comprehensive, multiscale. *Atmospheric Environment*, *36*(23), 3721–3734.

Brook, R., Brook, J., Urch, B., Vincent, R., Rajagopalan, S., & Silverman, F. (2002). Inhalation of fine particulate air pollution and ozone causes acute arterial vasoconstriction in healthy adults. *Circulation*, *105*(13), 1534.

Byun, D. (1999). Dynamically consistent formulations in meteorological and air quality models for multi-scale atmospheric applications: Part II. Mass conservation issues. *J. Atmos. Sci*, *56*, 3808–3820.

Byun, D., & Schere, K. (2006). Review of the governing equations, computational algorithms, and other components of the Models-3 Community Multiscale Air Quality (CMAQ) modeling system. *Applied Mechanics Reviews*, *59*, 51.

Byun, D., Young, J., Pleim, J., Odman, M. T., & Alapaty, K. (1999). Numerical transport algorithms for the Community Multiscale Air Quality (CMAQ) chemical transport model in generalized coordinates. *Science Algorithms of the EPA Models-3 Community Multiscale Air Quality (CMAQ) Modeling System, EPA/600/R-99*, *30*, 7–1.

Chen, J. (2009, June). personal communication.

Cohan, D. S., Hakami, A., Hu, Y., & Russell, A. G. (2005). Nonlinear response of ozone to emissions: Source apportionment and sensitivity analysis. *Environ. Sci. Technol*, *39*(17), 6739–6748.

Damian, V., Sandu, A., Damian, M., Potra, F., & Carmichael, G. R. (2002). The

kinetic preprocessor KPP-a software environment for solving chemical kinetics. *Computers & Chemical Engineering*, *26*(11), 1567–1579.

Dickinson, R. P., & Gelinas, R. J. (1976). Sensitivity analysis of ordinary differential equation systems–A direct method* 1. *Journal of computational physics*, *21*(2), 123–143.

Dominici, F., Peng, R., Bell, M., Pham, L., McDermott, A., Zeger, S., et al. (2006). Fine particulate air pollution and hospital admission for cardiovascular and respiratory diseases. *Jama*, *295*(10), 1127.

Dunker, A. M. (1981). Efficient calculation of sensitivity coefficients for complex atmospheric models. *Atmospheric Environment (1967)*, *15*(7), 1155–1161.

Dunker, A. M. (1984). The decoupled direct method for calculating sensitivity coefficients in chemical kinetics. *Journal of Chemical Physics*, *8*, 2385-2394.

Dunker, A. M., Yarwood, G., Ortmann, J., & Wilson, G. (2002). The decoupled direct method for sensitivity analysis in a three-dimensional air quality model - implementation, accuracy, and efficiency. *Environmental Science Technology*, *36*, 2965-2976.

Giles, M., Ghate, D., & Duta, M. (2008). Using automatic differentiation for adjoint CFD code development. *COMPUTATIONAL FLUID DYNAMICS JOURNAL*, *16*(4), 434.

Glarborg, P., Miller, J A., & Kee, R. J. (1986). Kinetic modeling and sensitivity analysis of nitrogen oxide formation in well-stirred reactors. *Combustion and Flame*, *65*(2), 177 - 202. Available from `http://www.sciencedirect.com/science/article/B6V2B-497S6M8-V/2/aaafac981419aa`

Gong, S., Barrie, L., Blanchet, J., Von Salzen, K., Lohmann, U., & Lesins. (2003). Canadian Aerosol Module: A size-segregated simulation of atmospheric aerosol processes for climate and air quality models. *Journal of geophysical research*, *108*(D1), 4007.

Griewank, A. (2003). A mathematical view of automatic differentiation. *Acta Numerica*, *12*(-1), 321–398.

Gruntz, D. (1997). Automatic differentiation and bisection. *MapleTech, Maple Technical Newsletter*, *4*, 14–19.

Hakami, A. (2003). *Direct sensitivity analysis in air quality models.* Doctoral dissertation, Georgia Institute of Technology.

Hakami, A., Odman, M. T., & Russell, A. G. (2003). High-order, direct sensitivity analysis of multidimensional air quality models. *Environmental Science Technology*, *37*, 2442-2452.

Hakami, A., Odman, M. T., & Russell, A. G. (2004). Nonlinearity in atmospheric response: A direct sensitivity analysis approach. *Journal of Geophysical Research*, *107*, D15303.

Hansen, J., Sato, M., Ruedy, R., Lacis, A., & Oinas, V. (2000). Global warming in the twenty-first century: An alternative scenario. *Proceedings of the National Academy of Sciences of the United States of America*, *97*(18), 9875.

Hascoet, L., & Pascual, V. (2004). TAPENADE 2.1 user's guide.

Hinds, W. (1999). *Aerosol technology: properties, behavior, and measurement of airborne particles.* 111 River Street, Hoboken, NJ 07030: Wiley. Available from http://books.google.com/books?id=ORxSAAAAMAAJ

Houyoux, M., Vukovich, J., Coats, C., Wheeler, N., & Kasibhatla, P. (2000). Emission inventory development and processing for the Seasonal Model for Regional Air Quality(SMRAQ) project. *Journal of Geophysical Research, 105*(D7), 9079–9090.

Jacob, D. J. (1999). *Introduction to atmospheric chemistry* (1st ed.). Princeton, New Jersey: Princeton University Press. (available online at: http://www-as.harvard.edu/people/faculty/djj/book)

Jacobson, M. (1994). *Developing, Coupling, and Applying a Gas, Aerosol, Transport, and Radiation Model to Study Urban and Regional Air Pollution.* Unpublished doctoral dissertation, University of California, Los Angeles.

Jiang, W. (2003). Instantaneous secondary organic aerosol yields and their comparison with overall aerosol yields for aromatic and biogenic hydrocarbons. *Atmospheric Environment, 37*(38), 5439–5444.

Koo, B., Dunker, A. M., & Yarwood, G. (2007). Implementing the decoupled direct method for sensitivity analysis in a particulate matter air quality model. *Environmental science & technology, 41*(8), 2847–2854.

Kourti, N., & Schatz, A. (1998). Solution of the general dynamic equation (GDE) for multicomponent aerosols. *Journal of Aerosol Science, 29*(1-2), 41–55.

Larson, R., Edwards, B., & Heyd, D. (1999). *Calculus Early Transcendental Functions: Single Variable and Student Study Guide and CD-ROM, Second Edition and Internet 6 Month.* 222 Berkeley st, Boston, MA 02116-3764: Houghton Mifflin Harcourt (HMH). Available from http://books.google.com/books?id=Zq_fAAAACAAJ

Leamer, E. E. (1985). Sensitivity analyses would help. *The American Economic Review, 75*(3), pp. 308-313. Available from http://www.jstor.org/stable/1814801

Makar, P., Bouchet, V., & Nenes, A. (2003). Inorganic chemistry calculations using HETV–a vectorized solver for the SO42–NO3–NH4+ system based on the ISORROPIA algorithms. *Atmospheric Environment, 37*(16), 2279–2294.

Mauzerall, D., & Wang, X. (2003). Protecting agricultural crops from the effects of tropospheric ozone exposure: reconciling science and standard setting in the United States, Europe, and Asia. *Energy and the Environment, 26*, 237-268.

McRae, G. J., Goodin, W. R., & Seinfeld, J. H. (1982). Numerical solution of the atmospheric diffusion equation for chemically reacting flows. *Journal of Computational Physics, 45*(1), 1–42.

Menon, S., Hensen, J., Nazarenko, L., & Luo, Y. (2002, September). Climate effects of black carbon aerosols in china and india. *Science, 297*.

Moran, M. (2010, October). personal communication.

Moran, M. (2011, March). personal communication.

Moran, M., Dastroor, A., Gong, S., Gong, W., & Makar, P. A. (1998). *Conceptual design for the aes unified regional air-quality modelling system* (Tech. Rep.). Air Quality Modelling and Integration Division, Air Quality Research Brance, Atmospheric Environment Service, Downsview, Ontario, M3H 5T4, Canada: Environment Canada.

Napelenok, S. L., Cohan, D. S., Hu, Y., & Russell, A. G. (2006). Decoupled direct 3D sensitivity analysis for particulate matter (DDM-3D/PM). *Atmospheric Environment*, *40*, 6112-6121.

Nowak, U. (1982). *A short users guide to LARKIN* (Tech. Rep.). Technical report, Konrad-Zuse Zentrum fuer, Informationstechnik Berlin.

Odman, M. T. (1998). *Research on numerical transport algorithms for air quality simulation models.* Research Triangle Park, NC 27711: United States Environmental Protection Agency, Research and Development, National Exposure Research Laboratory.

Odman, M. T., Boylan, J., Wilkinson, J., Russell, A. G., Mueller, S., & Imhoff. (2002). Integrated modeling for air quality assessment: The Southern Appalachians Mountains initiative project. In *Journal de physique iv (proceedings)* (Vol. 12, pp. 211–234).

Oh, S., Bissett, E., & Battiston, P. (1993). Mathematical modeling of electrically heated monolith converters: model formulation, numerical methods, and experimental verification. *Industrial & Engineering Chemistry Research*, *32*(8), 1560–1567.

Pandis, S. N. (1991). *Studies of physicochemical processes in atmospheric particles and acid deposition. dissertation (ph.d.).* Doctoral dissertation, California Institute of Technology. Available from `http://thesis.library.caltech.edu/1649/`

Pandis, S. N., Paulson, S., Seinfeld, J. H., & Flagan, R. (1991). Aerosol formation in the photooxidation of isoprene and [beta]-pinene. *Atmospheric Environment. Part A. General Topics*, *25*(5-6), 997–1008.

Pascual, V., & Hascoet, L. (2006). Extension of TAPENADE toward Fortran 95. *Automatic Differentiation: Applications, Theory, and Implementations*, 171–179.

Pryor, S., & Barthelmie, R. (1996). PM10 in Canada. *Science of The Total Environment*, *177*(1-3), 57–71.

Pudykiewicz, J., Kallaur, A., & Smolarkiewicz, P. (1997). Semi-Lagrangian modelling of tropospheric ozone. *Tellus B*, *49*(3), 231–248.

Rabitz, H., Kramer, M., & Dacol, D. (1983). Sensitivity analysis in chemical kinetics. *Annual review of physical chemistry*, *34*(1), 419–461.

Reich, P. (1987). Quantifying plant response to ozone: a unifying theory. *Tree Physiology*, *3*(1), 63.

Reynolds, S., Roth, P., & Seinfeld, J. H. (1973). Mathematical modeling of photochemical air pollution–I:: Formulation of the model. *Atmospheric Environment (1967)*, *7*(11), 1033–1061.

Roth, P., Roberts, P., & Mei-Kao Liu Steven, D. (1974). Mathematical modeling of

photochemical air pollution–II. A model and inventory of pollutant emissions. *Atmospheric Environment (1967)*, *8*(2), 97–130.

Russell, A. G., & Dennis, R. (2000). NARSTO critical review of photochemical models and modeling. *Atmospheric Environment*, *34*(12), 2283–2324.

Russell, M., Zhao, S., & Hakami, A. (2011, March). *Development of a tangent linear model for aurams for sensitivity analysis* (Tech. Rep.). 4095 Dufferin Street, Toronto, Ontario, Canada: Carleton University, Ottawa.

Sandhu, H. (1998). *Ambient particulate matter in Alberta.* Alberta, Canada: The Branch.

Sandu, A., Daescu, D., & Carmichael, G. R. (2003). Direct and adjoint sensitivity analysis of chemical kinetic systems with KPP. *Atmospheric Environment*, *37*, 5083-5096.

Sandu, A., Daescu, D., Carmichael, G. R., & Chai, T. (2005). Adjoint sensitivity analysis of regional air quality models. *Journal of Computational Physics*, *204*(1), 222–252.

Sandu, A., Verwer, J., Blom, J., Spee, E., Carmichael, G. R., & Potra, F. (1997). Benchmarking stiff ODE solvers for atmospheric chemistry problems II: Rosenbrock solvers. *Atmospheric environment*, *31*(20), 3459–3472.

Seinfeld, J. H., Atkinson, R., Berglund, R., Chameides, W., Cotton, W., et al. (1991). Rethinking the Ozone problem in urban and regional air pollution. *National Acad. Press, Washington, DC*.

Seinfeld, J. H., & Pandis, S. N. (2006). *Atmospheric chemistry and physics - from air pollution to climate change* (2nd ed.). New York, New York: Wiley-Interscience.

Sillman, S., Vautard, R., Menut, L., & Kley, D. (2003). O3-NO x-VOC sensitivity and NO x-VOC indicators in Paris: Results from models and Atmospheric Pollution Over the Paris Area (ESQUIF) measurements. *Journal of Geophysical Research*, *108*(D17), 8563.

Smolarkiewicz, P., & Pudykiewicz, J. (1992). A class of Semi-Langragian approximations for fluids. *Journal of the atmospheric sciences*, *49*(22), 2082–2096.

Squire, W., & Trapp, G. (1998). Using complex variables to estimate derivatives of real functions. *Siam Review*, *40*(1), 110–112.

Stieb, D., Burnett, R., Smith-Doiron, M., Brion, O., HWASHIN, H., & Economou, V. (2008). A New Multipollutant, No-Threshold Air Quality Health Index Based on Short-Term Associations Observed in Daily Time-Series Analyses. *Journal of the Air & Waste Management Association*, *58*(3), 435–450.

Stobbs, N. T. (1952, December). *Photograph of Nelson's Column during the 1952 Great Smog event in London.* (Image Copyright N T Stobbs. This work is licensed under the Creative Commons Attribution-Share Alike 2.0 Generic Licence. To view a copy of this licence, visit http://creativecommons.org/licenses/by-sa/2.0/ or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.)

Stockwell, W., & Lurmann, R. (1989). *Intercomparison of the ADOM and RADM*

Gas-Phase Chemical Mechanisms (Tech. Rep.). 3412 Hillview Avenue, Palo Alto, Ca.: Electrical Power Research Institute Topical Report.

Stull, R. (1988). *An introduction to boundary layer meteorology.* 101 Philip Drive, Norwell, MA 02061: Springer.

Talagrand, O., & Courtier, P. (1987). Variational assimilation of meteorological observations with the adjoint vorticity equation. I: Theory. *Quarterly Journal of the Royal Meteorological Society, 113*(478), 1311–1328.

U.S. EPA. (2006). *U.S. EPA. Air Quality Criteria for Ozone and Related Photochemical Oxidants (2006 Final)* (Tech. Rep.). Washington, DC: U.S. Environmental Protection Agency.

Vedal, S. (1997). Ambient particles and health: lines that divide. *Journal of the Air & Waste Management Association, 47*(5), 551–581.

Venkatram, A., Karamchandani, P., & Misra, P. (1988). Testing a comprehensive acid deposition model. *Atmospheric Environment (1967), 22*(4), 737–747.

Wang, K., Lary, D., Shallcross, D., Hall, S., & Pyle, J. (2001). A review on the use of the adjoint method in four-dimensional atmospheric-chemistry data assimilation. *Quarterly Journal of the Royal Meteorological Society, 127*(576), 2181–2204.

Wexler, A., Lurmann, F., & Seinfeld, J. H. (1994). Modelling urban and regional aerosols–I. model development. *Atmospheric Environment, 28*(3), 531–546.

Winkler, P. (1973). The growth of atmospheric aerosol particles as a function of the relative humidity–II. An improved concept of mixed nuclei. *Journal of Aerosol Science, 4*(5), 373–387.

World Health Organization. (2000). *Air Quality Guidelines for Europe* (2nd ed.; Tech. Rep. No. 91). World Health Organization, Regional Office for Europe, Copenhagen.

Yang, Y.-J., Wilkinson, J. G., & Russell, A. G. (1997). Fast, direct sensitivity analysis of multidimensional photochemical models. *Environmental Science Technology, 31*, 2859-2868.

Zhang, L., Moran, M., Makar, P., Brook, J., & Gong, S. (2002). Modelling gaseous dry deposition in AURAMS: a unified regional air-quality modelling system. *Atmospheric Environment, 36*(3), 537–560. Available from http://www.sciencedirect.com/science/article/B6VH3-44SJFVX-B/2/a8850883a83296

Appendix A

Sensitivity Module Documentation

The intent of the sensitivity module is to allow the user to easily implement multiple sensitivities parameters into the AURAMS model without having to deal with the details of memory management, field initialization, or even inputting the details of the TLM sensitivity to be calculated. This modules provides broad and detailed memory management, can input variable number of sensitivity input parameters from a namelist file, and can be extended for higher order derivatives.

Note that because AURAMS is developed to be compiled with a FORTRAN95 compiler, objects are not available. Therefore this module was implemented with structures and subroutines that operate on those structures. Though there are methods that can be used to "simulate" object behaviour in FORTRAN95, these methods would add too much overhead to each "objects" methods, computational overhead in each method, and would increase the difficulty in testing the module. Despite this, this document might refer to the structures in this module as objects, this is simply conceptual and does not reflect an object oriented implementation. If however this module is ever placed in a FORTRAN model, it is recommended that the module be upgraded to use a full object oriented implementation.

A.1 User Documentation

The sensitivity module must be loaded into the model using FORTRAN95's `USE` keyword which must occur before the `IMPLICIT NONE` call, e.g.

```
USE Sensitivities
IMPLICIT NONE;
```

Before any sensitivity fields can be accessed, the module must be initialized. This step loads all the sensitivity input parameters for a *sensitivities.namelist* file (format discussed in section A.3) that must be present in the run directory. To initialize the module, call,

```
! Declare the field sizes
CALL setFldSizes(ubound(ma), ubound(accfld), ubound(wrkacc), ubound(flxt))

! Load sensitivity Objects
CALL loadSensitivityObjects()
```

The first call specifies to the module the sizes of all the fields and work fields it will need to later allocate. The `loadSensitivityObjects()` routine initializes all the required sensitivities structures that were specified in the *sensitivities.namelist*. Each sensitivity object holds several bits of information, namely, the *name* of the sensitivity, *type*, and *list of species* to calculate the sensitivity with respect to (this parameter is an array rather than a species identifier such that sensitivities can be with respect to lump species such as NO_x or VOC.) The name is used for the final output and is currently constrained to 4 characters by AURAMS and the FST functions in *librmn.a*. The type specifies how the sensitivity parameter are used, for instance, there are currently two separate types of sensitivities, sensitivity to initial concentration and sensitivity to emissions (the actual configuration allows finer grain settings to distinguish between emission types.) How these sensitivities are processed using these types is handled in the CTM, not the module.

At this point, the user has two choices, to allocate all the fields that are used in AURAMS at the beginning, or use fine grained memory management throughout the code.

Because AURAMS is much more computationally intensive than memory intensive, we recommend that the fields are initialized right away. This can be done with,

```
do i=1,nOrder(1)
        ! Create the wrkfld, accfld, wfkfld, flxb, flxt
        CALL createSensitivityWorkFields(getSa_1(i));
enddo
```

At this point, all the fields and work fields on the sensitivity object are allocated. Note the 1 in *nOrder*, as well as the "_1" in *getSa_1(i)*; this "1" denotes that these are objects intended to be used to calculate first order sensitivities. A second order version of this object as well as the subroutines to control it, however at the time of this writing they are not as developed as the first order components.

There are several routines to access the fields on the sensitivity object, these are:

```
! Fetch the sensitivity field
POINTER(:,:,:,:) getFld(sa)

! Fetch the 3D work field
POINTER(:) getWkfld3d(sa)

! Fetch the accfld
POINTER(:,:,:,:) getAccFld(sa)

! Fetch the work field for the accumulated field
POINTER(:,:,:,:) getWrkAcc(sa)

! Fetch the flux fields
getFlxFlds(sa, POINTER(:), POINTER(:))
```

where *sa* denotes the sensitivity object, and the upper case terms denote the output type of the function (first 4) or subroutine (last.)

Once the model is done with the temporary work fields, they should be destroyed in order to save memory. This can be done with,

```
do i=1,nOrder(1)
        ! Create the wrkfld, accfld, wfkacc, flxb, flxt
        CALL destroySensitivityWorkFields(getSa_1(i));
enddo
```

So, for example, in *ctm()*, the code to inject point source emissions into the sensitivity fields would look something like,

```
! Inject emissions into concentration field
CALL emfld(qa, MA(1,1,1,iadv(isv)),rho, ngp,                    &
           nis, njs, nks, nlvem, iadv(isv), dtm,                &
           istart+1, istop-1, jstart+1, jstop-1,i_no,no_cor) &

do j=1,nOrder(1)
        p_sa1 => getSa_1(j);
        if (isv .in. p_sa1 .and. &
            iand(getSensitivityType(p_sa1), sens_type_emit_ps) /= 0) THEN

                ! Fetch field
                p_sa_fld => getFld(p_sa1)

                ! Inject emissions into sensitivity field
                CALL emfld(qa, P_SA_FLD(1,1,1,iadv(isv)),rho,ngp, &
                   nis, njs, nks, nlvem, iadv(isv), dtm,          &
                   istart+1, istop-1,jstart+1,jstop-1,i_no,no_cor)&
        endif
enddo
```

In the code above, first emissions are injected into the concentration field, then the CTM loops through all the first order sensitivity objects. Before injecting emissions into a sensitivity field, the code first checks to see whether the sensitivity object is used for point source emissions and if it is currently dealing with a species the sensitivity object is sensitive to. The latter is done using the custom `.in.` operator to check is the species `isv` is `.in.` the `p_sa1%species` array, the former is done by doing a binary comparison of the sensitivity type and the required type to do emission sources.

If the condition holds, the sensitivity field is fetched (pointed to by `p_sa_fld`) and given to the `emfld` function for emission injection.

A.2 Developer Documentation

The sensitivity module is designed to allow the developer to be conservative with their memory use by providing a suit to easily allocate and deallocate fields when necessary.

This module contains two sensitivity structures, one to hold the information for first order sensitivities, and one for second order sensitivities. Because these two structures are conceptually very similar (in fact, would be siblings of a parent class if FORTRAN95 supported it) and because the subroutines for for order sensitivities are much better developed, only the first order sensitivities will be discussed here.

The sensitivity object is defined as,

```
! This is a structure, not an object.  This fields here
! are initialized (nullified) in addFirstOrderSensitivity
type, PUBLIC :: sens_first_order; PRIVATE
        ! To keep track of all the sensitivities, the CTM
        ! calls a method here to produce a "unique" species name
        ! based on which sensitivity this is
        integer :: id

        integer :: order
        integer :: sens_type
        character(len=sname_length) :: sname ! Len=4 comes from AURAMS
        real, dimension(:,:,:,:), pointer :: fld

        ! Additional fields used for the accumulated species
        real, dimension(:,:,:,:), pointer :: accfld
        real, dimension(:,:,:,:), pointer :: wrkacc

        ! For a wrkfld, being named 3D and being 1D is confusing,
        ! and limits performance boosts I could get with pointers
        ! but I have to stay consistent with AURAMS
        real, dimension(:), pointer :: wkfld3d

        ! Flux emissions that get sent to diffusion
        real, dimension(:), pointer :: flxb, flxt

        integer, dimension(:), pointer :: species;
end type sens_first_order
```

These types are populated in `loadSensitivityObjects()` which calls the private routine `addFirstOrderSensitivity()` to initialize all the fields on the sensitivity object and store the sensitivity object it self in a container structure of type `sensitivity_meta`.

The following functions and routines exist for creating/getting/setting/destroying these fields,

```
! Fetch the sensitivity field
POINTER(:,:,:,:) getFld(sa)

! Create, fetch, save, and destroy sensitivity work field
createWkfld3d(sa, integer species_id, logical optional first_destroy_field)
getWkfld3d(sa)
saveWkfld3d(sa, integer species_id, logical optional destroy_wkfld_after_save)
destroyWkfld3d(sa)

! Create, fetch and destroy the accfld
createAccFld(sa, logical optional first_destroy_field)
POINTER(:,:,:,:) getAccFld(sa)
destroyAccFld(sa)

! Create, fetch and destroy the work field for the accumulated field
createWrkAcc(sa, logical optional first_destroy_field)
POINTER(:,:,:,:) getWrkAcc(sa)
destroyWrkAcc(sa)

! Create, fetch and destroy the flux fields
createFlxFlds(sa, logical optional first_destroy_field)
getFlxFlds(sa, POINTER(:), POINTER(:))
destroyFlxFlds(sa)
```

where `sa` is a first order sensitivity object.

Each sensitivity structure has a type, these types are stored as binary such that the type integer can represent multiple types, these types are:

```
! Initial Conditions
integer, parameter :: sens_type_initial = 1

! Point Source Emissions
```

```
integer, parameter :: sens_type_emit_ps = 2

! Flux Source Emissions
integer, parameter :: sens_type_emit_fx = 4
```

The private function `checkSensType(integer sensitivity_type)` is called by `addFirstOrderSensitivity()` to apply constraints to the type. i.e. a sensitivity can either be for initial conditions or for an emission type, but not both; also, a sensitivity can be for both emission types (2&4 = 6.) The architecture allows this module to easily be expanded to permit sensitivities to other aspects of the model, such as boundary conditions or rate constants. Two other subroutines related to the sensitivity type are:

```
! Get the type of sensitivity
integer getSensitivityType(sens_first_order sensitivity)

! Get a string containing the sensitivity type, \eg, "initial conditions"
character(len=30) sensitivityTypeStr(integer sensitivity_type)
```

Each sensitivity object also has a list of species that sensitivities are calculated with respect to. These species are stored in the sensitivity structures `species` attribute. As mentioned above, the `.in.` operator was developed to easily check to see if a species is in a list of species in the sensitivity structure. To receive a list of these species, use the following routines:

```
! Get the species array (this is an allocated copy of the array)
integer(:) getSpecies(sa)

! Print the species array to standard output
printSpecies(species)
```

Because this module allows for multiple sensitivities to be calculated in one run, the model output requires some special considerations. Using the current output mechanisms in AURAMS right now, there are two options for outputting multiple sensitivities, outputting a separate file per sensitivity (one file for the sensitivity of species to say, NO emissions,

another for all sensitivity of all species to initial ozone, etc), or outputting all sensitivities into one file with an identifier in their name. Currently the latter option was implemented. The sensitivity module has a routine for returning a unique name for the species based on the sensitivity object,

```
! Return a unique species name, \eg, O3 becomes 2O3 on the second
! sensitivity object.
CHARACTER (len=4) saName(sa, CHARACTER (len=4) species_name)

! Example use in AURAMS
sa_name = saName(p_sa1, rpn(isv));
```

A.3 Description of *sensitivities.namelist*

The *sensitivities.namelist* file allows for full configuration of the TLM sensitivities one plans to calculate. To set up the TLM sensitivities, the user must configure two structures in this file. The `&smeta` object that houses the number of sensitivity objects to expect, and each sensitivity object, namely `&s1_data` objects for first order sensitivities.

The setup as well as examples are shown in the sample *sensitivities.namelist* file below,

```
! Sensitivity options

! Enter the number of sensitivity coefficients to calculate
&smeta
        n_first_order = 1
        n_second_order = 0
/

! To add a sensitivity, add s#_data options below in the format:
! First order sensitivity coefficient
!
! &s1_data
!    sens_type = # where  1 = initial conditions
!                         2 = point source emissions
!                         4 = flux source emissions
!                         6 = all emissions (6=2+4)
```

```
!    sname = 'cccc' where c's are characters.  This is the name
!                   of the sensitivity, one example would be
!                   'NOX_' for sensitivity to NOx.  The 4 character
!                   restriction makes it difficult to use a good
!                   name
!
!    species = #,#,#...    Species IDs to calculate sensitivities
!                          with respect to.  Look in static_masterlist_file
!                          for definitions of these species.
!                          Example values are:
!                          1 = SO2
!                          2 = SO4
!                          3 = NO
!                          4 = NO2
!                          5 = O3
!                         28 = CO
!                         28 = CO
!                         36 = OH
!                         ...
!                         NOx  = 3,4
!                         VOCs = 8,9,10,11,12,13,23
! /
!
! Or, for a second order sensitivity,
! &s2_data
!    sens_type = #
!    sname_1   = 'cccc'
!    species_1 = #,#,#,...
!    sname_2   = 'cccc'
!    species_2 = #,#,#,...
! /
! There are two species entries above such that you may calculate
! an ozone isopleth for example, where the following coefficients
! needed: S'_NOx, S'_VOC, S''_NOx,NOx, S''_NOx,VOC, S''_VOC,VOC
! So, for that example, you need 5 &s2_data objects

! Sensitivity to initial ozone
!&s1_data
!        sens_type = 1       ! Initial condition
!        sname     = 'iO3_'
!        species   = 5       ! Ozone
!/

!! Sensitivity to NO PS emissions
!&s1_data
```

```
!        sens_type = 2       ! PS emissions
!        sname     = 'NO_'
!        species   = 3 ! NO
!/

!! Sensitivity to NO emissions
!&s1_data
!        sens_type = 6       ! All emissions
!        sname     = 'eNO_'
!        species   = 3 ! NO
!/

! Sensitivity to NOx emissions
&s1_data
        sens_type = 6       ! All emissions
        sname     = 'NOx_'
        species   = 3,4 ! NOx
/

!! Sensitivity to SO2 emissions
!&s1_data
!        sens_type = 6       ! All emissions
!        sname     = 'eSO2'
!        species   = 1 ! SO2
!/
```

Appendix B

Example of sensitivity to chemistry

Consider the following set of reactions:

$$NO_2 + h\nu \xrightarrow{k_1} NO + O \tag{R1}$$

$$NO + O_3 \xrightarrow{k_2} NO_2 + O_2 \tag{R2}$$

$$O + O_2 + M \xrightarrow{k_3} O_3 + M \tag{R3}$$

for some rate constants k_1, k_2, and k_3, and $h\nu$ being the amount of sunlight radiation at an appropriate wavelength, λ.

Let $\tilde{\mathbf{x}}$ be the concentration input vector,

$$\tilde{\mathbf{x}} = \begin{bmatrix} [NO] \\ [NO_2] \\ [O] \\ [O_3] \end{bmatrix} (t=0), \qquad \tilde{\mathbf{y}} = \begin{bmatrix} [NO] \\ [NO_2] \\ [O] \\ [O_3] \end{bmatrix} (t)$$

where the square brackets denote concentration. Note that we exclude O_2 and M as it is assumed to have an effective constant mixing ratio. It can then be shown that the

production rates are:

$$\tilde{\mathbf{R}} = \begin{bmatrix} R_{\mathrm{NO_2}} \\ R_{\mathrm{NO}} \\ R_{\mathrm{O}} \\ R_{\mathrm{O_3}} \end{bmatrix} = \begin{bmatrix} \frac{\mathrm{d[NO_2]}}{\mathrm{d}t} \\ \frac{\mathrm{d[NO]}}{\mathrm{d}t} \\ \frac{\mathrm{d[O]}}{\mathrm{d}t} \\ \frac{\mathrm{d[O_3]}}{\mathrm{d}t} \end{bmatrix} = \begin{bmatrix} k_1[\mathrm{NO_2}] - k_2[\mathrm{NO}][\mathrm{O_3}] \\ -k_1[\mathrm{NO_2}] + k_2[\mathrm{NO}][\mathrm{O_3}] \\ k_1[\mathrm{NO_2}] - k_3[\mathrm{O}][\mathrm{O_2}][\mathrm{M}] \\ -k_2[\mathrm{NO}][\mathrm{O_3}] + k_3[\mathrm{O}][\mathrm{O_2}][\mathrm{M}] \end{bmatrix}$$

Given this, the Jacobian is calculated as,

$$J = \frac{\partial \tilde{\mathbf{R}}}{\partial \tilde{\mathbf{x}}} = \begin{bmatrix} \frac{\partial R_1}{\partial x_1} & \frac{\mathrm{d}R_1}{\mathrm{d}x_2} & \cdots \\ \frac{\partial R_2}{\partial x_1} & \frac{\mathrm{d}R_2}{\mathrm{d}x_2} & \cdots \\ \vdots & \vdots & \vdots \\ \cdots & \cdots & \frac{\partial R_4}{\partial x_4} \end{bmatrix}$$

$$J = \begin{bmatrix} -k_2[\mathrm{O_3}] & k_1 & 0 & -k_2[\mathrm{NO}] \\ k_2[\mathrm{O_3}] & -k_1[\mathrm{NO_2}] & 0 & k2[\mathrm{NO}] \\ 0 & k_1 & -k_3[\mathrm{O_2}][\mathrm{M}] & 0 \\ -k_2[\mathrm{O_3}] & 0 & k_3[\mathrm{O_2}][\mathrm{M}] & -k_2[\mathrm{NO}] \end{bmatrix} \tag{B.1}$$

What can be read from (B.1) is that for example, the sensitivity of the rate of production of NO to the concentration of available $[\mathrm{NO_2}]$ is k_1, and that the production of $\mathrm{NO_2}$ is not directly sensitive to O.