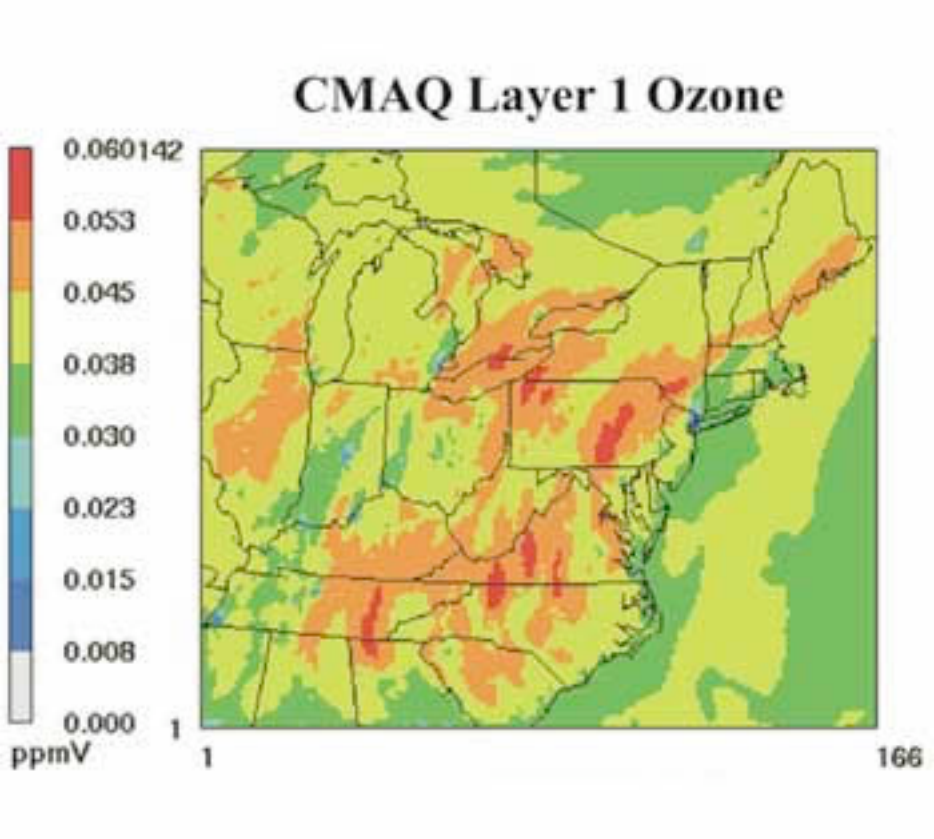
Community Multiscale Air Quality (CMAQ) Modeling System



Introduction to CMAQ Version 5.0

TRAINING MANUAL

Community Modeling and Analysis System Center
UNC Institute for the Environment
Chapel Hill, NC

Disclaimer

The statements in this training manual are those of the University of North Carolina at Chapel Hill and not necessarily those of the United State Environmental Protection Agency. The material in this training manual has not been subjected to the U.S. EPA's peer and administrative review, nor has it been approved for publication as an EPA document. The mention of commercial products, their source, or their use in connection with material reported herein is not to be construed as actual or implied endorsement of such products.

Acknowledgements

The CMAS Center is grateful to the U.S. National Park Service Air Resources Division for providing the Rocky Mountain Atmospheric Nitrogen and Sulfur (RoMANS) study data used in this training course.

Contents: Introduction to CMAQ version 5.0

ACRONYMS

BCs	Boundary Conditions
BCON	Boundary Conditions processor
CB-IV (or CB4)	Carbon Bond-IV chemical mechanism
CB05	Carbon Bond '05 chemical mechanism
CCTM	CMAQ Chemical Transport Model
CEMPD	Center for Environmental Modeling for Policy Development (UNC-IE)
CMAS	Community Modeling and Analysis System
CMAQ	Community Multiscale Air Quality model
CTM	Chemical Transport Model
CVS	Concurrent Versions System
GMT	Greenwich Mean Time
I/O API	Input/Output Applications Programming Interface
ICs	Initial Conditions
IE	Institute for the Environment (UNC)
JPROC	Photolysis Rate processor
MCIP	Meteorology-Chemistry Interface Processor
MM5	Fifth Generation Mesoscale Model
NCAR	National Center for Atmospheric Research
NOAA	National Oceanic and Atmospheric Administration
NO_x	Oxides of nitrogen
O_3	Ozone
PAVE	Package for Analysis and Visualization of Environmental data
PBL	Planetary Boundary Layer
PDM	Plume Dynamics Model
PinG	Plume-in-Grid
PM	Particulate Matter
$PM_{2.5}$	Particulate Matter less than 2.5 μm in diameter
PPB	Parts Per Billion
PPM	Parts Per Million or Piecewise-Parabolic Method advection scheme
PSU	Pennsylvania State University
QSSA	Quasi-Steady State Approximation model
RADM	Regional Acid Deposition Model
RADM2	Regional Acid Deposition Model version 2
SAPRC	State Air Pollution Research Center chemical mechanism
SMOKE	Sparse Matrix Operator Kernel Emissions processor
SO_x	Oxides of Sulfur
UNC	University of North Carolina
UTC	Universal Time Coordinate
WRF	Weather Research and Forecasting model

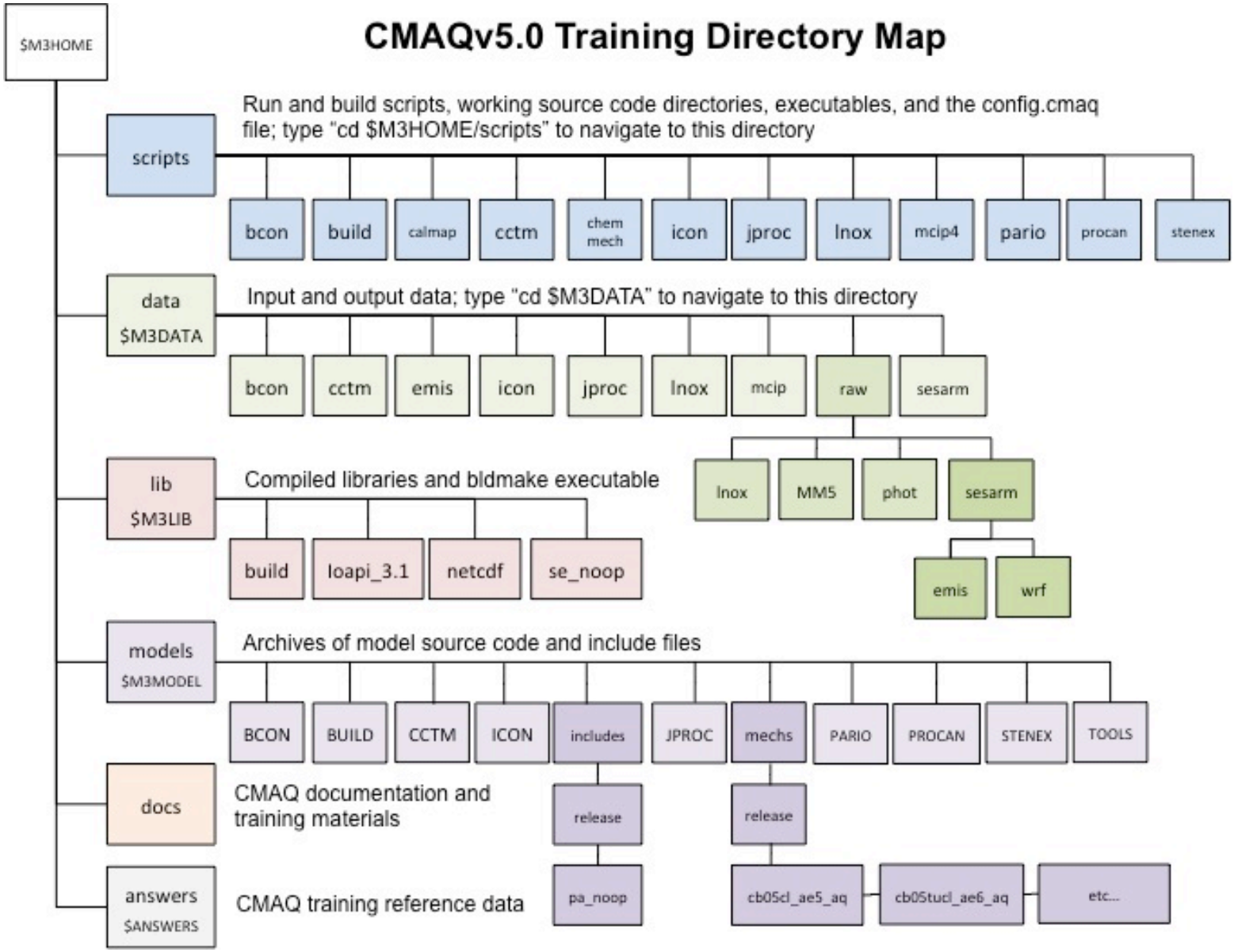
COURSE AGENDA

Day 1

8:30-9:45	CMAQ basics lecture
9:45-10:00	Break
10:00-11:00	Hands-on overview lab
11:00-12:15	Hands-on MCIP lab
12:15-1:15	Lunch
1:15-2:15	Hands-on BCON lab
2:15-3:15	Hands-on ICON lab
3:15-3:30	Break
3:30-5:00	Hands-on CCTM lab

Day 2

8:30-8:45	Review
8:45-9:45	Hands-on Nesting lab
9:45-10:00	Break
10:00-11:00	Hands-on Nesting lab (con't)
11:00-12:00	Hands-on Multiday/Restart lab
12:00-1:00	Lunch
1:00-1:45	Hands-on Lightning NOx Emissions lab
1:45-2:30	Advanced CMAQ topics Lecture
2:30-2:45	Break
2:45-5:00	Hands-on Problem Solving case study
4:30-5:00	Q&A (Overlap)


CMAQv5.0 Training Directory Map
$M3HOME
scripts
Run and build scripts, working source code directories, executables, and the config.cmaq file; type "cd $M3HOME/scripts" to navigate to this directory
bcon
build
calmap
cctm
chem mech
icon
jproc
lnox
mcip4
pario
procan
stenex
data
$M3DATA
Input and output data; type "cd $M3DATA" to navigate to this directory
bcon
cctm
emis
icon
jproc
lnox
mcip
raw
sesarm
lnox
MM5
phot
sesarm
emis
wrf
lib
$M3LIB
Compiled libraries and bldmake executable
build
loapi_3.1
netcdf
se_noop
models
$M3MODEL
Archives of model source code and include files
BCON
BUILD
CCTM
ICON
includes
JPROC
mechs
PARIO
PROCAN
STENEX
TOOLS
docs
CMAQ documentation and training materials
release
release
answers
$ANSWERS
CMAQ training reference data
pa_noop
cb05cl_ae5_aq
cb05tucl_ae6_aq
etc...

(Page left intentionally blank)

1 CMAQ Training — Overview

1.1 Introduction

During this training overview, you will:

- Learn basic UNIX commands needed for using CMAQ from scripts
- Learn how to navigate around the main CMAQ directories
- Learn how to use the **config.cmaq** script to configure your computing system for compiling and running CMAQ
- Explore the input and output directories, and review the locations of important CMAQ input files
- Learn how to compile and use the **bldmake** utility for creating CMAQ executables, and learn about the source-code control utility CVS
- Learn about the I/O API, netCDF, and other libraries needed by CMAQ, and compile the CMAQ stencil exchange library
- Get specifics about the training case you will be simulating

NOTE: The symbol ">" is used throughout the training materials to represent the UNIX prompt. When this symbol appears as the first entry on a line in these instructions, you do not need to type it. Also, `Courier font` is used to indicate text that you should type at the prompt as you work through the exercises.

The login information for your training laptop is provided below. Keep this information handy for reference throughout the class.

Login: _____

Password: _____

1.2 UNIX commands and information needed for training

A. The following is a brief tutorial of basic UNIX commands needed for the CMAQ exercises.

NOTE: In UNIX the case is important (lowercase and uppercase are not interpreted the same way), so please follow the instructions carefully.

A. **Start an xterm window.**
When you log in to your training machine, double click on the "CMAQ Training" icon on the desktop. This icon launches an X-terminal (xterm), which is an interface to the Linux cluster that you will use for running CMAQ. You can start a new **xterm** window by

either clicking on the desktop icon or by typing the following command in an existing xterm window, and then hit the <Enter> key:

```
> xterm &
```

NOTE: The "&" symbol is used at the Linux command line to execute a command in the background. When you type the "&" after a command line argument, the command preceding the symbol is run in the background, meaning that the command line will continue to be available to run other commands while the previous command continues to run.

B. **Get a list of the contents of a directory.**
To list the contents of a directory, use the **ls** command, followed by the <Enter> key:

```
> ls
```

C. **Change directories.**
To change to a different directory, use the **cd** command. The syntax of this command is:

cd <directory name>

For example, to change to the CMAQ training directory from your home directory, type the following command:

```
> cd /home/training/CMAQv5.0
```

To back out of a directory, type the following command:

```
> cd ../
```

To back out two directory levels, type the following command:

```
> cd ../../
```

D. **Use an environment variable.**
The following example uses the environment variable M3HOME to change to the root directory of the CMAQ system. To use an environment variable, the dollar sign must precede the environment variable with no space between the dollar sign and the environment variable. Type the following command to change to the CMAQ root directory (we will use capital letters for all environment variables):

```
> cd $M3HOME
```

NOTE: For the training, M3HOME has already been set for you in your environment.

E. **Determine the value of an environment variable.**
The **echo** command permits you to see the value of an environment variable. For example, to see the value of the M3HOME environment variable, type the following:

```
> echo $M3HOME
```

The contents of the variable will be displayed, and the UNIX prompt will then return.

F. **Find out what directory you are currently in.**
The **pwd** command allows you to see the directory currently available at the UNIX prompt. Depending on the configuration of your computer, this may also be displayed along with the UNIX prompt. Type the following command to see your directory:

```
> pwd
```

G. **Copy a file.**
The **cp** command is used to make a copy of a file and give that copy a different name. The syntax of the copy command is:

cp <from filename> <to filename>

1.3 The CMAQ system directories

B. The main CMAQ system directory is $M3HOME. From the xterm window, go to the main CMAQ system directory and look at its contents by typing:

```
> cd $M3HOME                                   changes the directory
> ls                                           shows directory contents
```

Several directory names will be displayed in your window:

- The **answers** directory is the location of the reference data for the training exercises.
- The **data** directory is the location of the input and output files.
- The **lib** directory is the location of the supporting CMAQ utilities and libraries.
- The **scripts** directory is the location of the installation, compilation, and run scripts.
- The **models** directory is the location of the CMAQ source code archives.

1.4 CONFIG.CMAQ: Setting the CMAQ Environment

C. The file **config.cmaq** is used to set up the environment variables that CMAQ needs for compiling and executing. It is a script that sets up directory locations and compiler options required by the CMAQ system. By default the config.cmaq script is located in the $M3HOME/scripts directory. Navigate to this directory by typing the following command in the xterm:

```
> cd $M3HOME/scripts
```

NOTE: The M3HOME environment variable is already set in your environment.

View the config.cmaq script using the command less:

```
> less config.cmaq
```

NOTE: "less" is a Unix command used to view text files. Scroll forward in the file by typing "f", scroll backward by typing "b", and quit the command by typing "q".

The config.cmaq script contains two sections:

- **model archive location** sets the location of important CMAQ directories
- **architecture & compiler specific settings** sets compiler configurations for building CMAQ

The **model archive location** section of the config.cmaq script sets environment variables for three directories needed to build and run CMAQ:

- M3HOME = the base CMAQ installation location
- M3MODEL = the location of the CMAQ source code repository
- M3DATA = location of the CMAQ input and output data

The **architecture & compiler specific settings** section contains options for building CMAQ using different Fortran compilers. The first four lines of this section are independent of the compiler and are used to set environment variables for identifying the operating system that will be used to build CMAQ.

This section then contains settings for the Intel, Portland Group, and Gnu Fortran compilers. The "#" is used to comment out lines in the config.cmaq script. All text following the "#" are ignored when the script is invoked. The training version of the config.cmaq script is set up for the Intel Fortran compiler. Notice how the settings for the Portland Group and Gnu compilers are deactivated or commented out through the use of the "#" symbol. To switch compiler settings in the config.cmaq script, you must comment out the lines (i.e. add a "#" symbol at the beginning of the line) for the compiler that you do not want to use and uncomment the lines for the compiler that you do want to use.

The final sections of the config.cmaq script set the location of the library directory needed for compiling CMAQ, an executable identifier (EXECID) used to tag the CMAQ programs with information about the operating system and compiler, and CVS remote archive information.

Quit from the less command and invoke the config.cmaq script using the following command:

```
> source config.cmaq
```

This source command applies the settings in the config.cmaq script to the xterm in which the command is run. All of the environment variables that are listed on uncommented lines in the config.cmaq script are now set in your xterm window. For example, use the Linux command echo to print the setting of the environment variables "compiler":

```
> echo $compiler
```

This command should return the result "ifort" telling you that you will be compiling CMAQ with the Intel Fortran compiler. Notify the instructor if this echo command returns any result other than "ifort".

IMPORTANT: Each time you open a new xterm you need to "source" the config.cmaq file to set the CMAQ environment in the new window. As a reminder, the steps required to use the config.cmaq file for new xterms are listed below:

```
> cd $M3HOME/scripts
> source config.cmaq
```

1.5 Searching the contents of the data directory

D. Go to the CMAQ **data** directory and look at its contents by typing:

```
> cd $M3DATA
> ls
```

Several directory names will be displayed in your window:

- The **bcon** directory: location for the output of the boundary conditions processor.
- The **cctm** directory: location for the output of the chemical transport model.
- The **emis** directory: location for the output of the emissions model (e.g., SMOKE).
- The **icon** directory: location for the output of the initial conditions processor.
- The **jproc** directory: location for the output of the photolysis rate processor.
- The **lnox** directory: location for the output of the lightning NOx emissions processor.
- The **mcip** directory: location for the output of the meteorology data processor.
- The **raw** directory: location for all of the input data for the CMAQ models.
- The **sesarm** directory: location for the input data for the Lab 9 problem solving case study.

You can now navigate and explore the various data directories using the **cd** command to access the directories and the **ls** command to list their contents.

Exercise 1-1:
What are the names of the four subdirectories contained in the $M3DATA/raw directory?

Answer 1-1:

Exercise 1-2:
In which directory is the meteorology input data located?

Answer 1-2:

1.6 Using bldmake and CVS

E. This section includes brief explanations of the CMAQ utility **bldmake** and the source-code control utility **CVS** as they apply to the CMAQ modeling system. Explanations of the scripts used to create and run the bldmake executable will be presented in detail but they will not be used to actually create executables.

A. The CMAQ utility **bldmake** is a program that controls the configuration management, source-code archival, and compilation operations of the CMAQ system. The various components of CMAQ are configured and compiled using scripts that invoke **bldmake**. **Bldmake** dynamically generates a Fortran Makefile for compiling the CMAQ programs. The user has the option of allowing **bldmake** to automatically execute this Makefile to compile the source code into an executable or to run the Makefile manually.

B. The Concurrent Versions System (**CVS**) is a source-code version control system that operates on a hierarchical directory structure to manage code releases and control the concurrent editing of files among multiple authors. CMAQ source code is distributed as a **CVS** archive. In the CMAQ system, **CVS** is used to maintain a centralized repository of the distributed scripts and code in a secured and regulated location.

C. The first step that **bldmake** implements when compiling a CMAQ module is to "check out" the appropriate files from the **CVS** archive. After the correct version of the source code is copied into a working directory, **bldmake** then invokes a compiler to build the source files into an executable program file.

D. Initializing the installation of the CMAQ modeling system begins with creating the **bldmake** utility. As mentioned in Step 2 above, the $M3HOME/scripts directory

contains the installation, compilation, and run scripts for all of the CMAQ programs, including **bldmake**. Go to the **bldmake** scripts directory by typing:

```
> cd $M3HOME/scripts/build
```

View the **bldmake** build script by typing:

```
> gedit bldit.bldmake &
```

Scroll down the script to look at how the **bldmake** script calls CVS and the Fortran compiler to create the program **bldmake**. The first non-comment line of the script sources the config.cmaq script. This next lines search for the existence of the $M3MODEL and $M3LIB directories, as these contain the CMAQ source code and libraries. The next section of the script uses the UNIX utility `uname` to determine the operating system for which **bldmake** will be created. Note how the script installed with the training is configured for Linux.

Exercise 1-3:

Substituting the environment variables for their actual values, what is the full path of the directory where **bldmake** will be placed by this script?

Answer 1-3:

The next few lines of the script set the `echo` command, the location of the working **bldmake** directory, and name of the executable. The location of the CVS repository for the **bldmake** code and the CVS release tag are defined next, followed by the location of the Fortran compiler and any compilation flags used to build the program. Note how the Fortran compiler and compilation flags are set by the environment variables myFC and myFFLAGS, respectively. These environment variables are set in the config.cmaq script and passed into the **bldmake** compilation script.

The final section of the **bldmake** script sets the target directory for the source code to be checked out from the CVS repository, checks the code out of CVS, and invokes the Fortran compiler to create the **bldmake** executable. Note how after the name of the Bld directory is set, the script checks to see if the directory exists (if ! –e "$Bld") and then creates it if it does not (mkdir $Bld). If the environment variable Bld is set, the script checks to see if it points to an existing directory. If it does not, the script prints a notification and exits. If Bld does point to an existing directory, the contents of the directory are deleted and the script changes directories into Bld before invoking CVS. A CVS export command is then invoked to check the **bldmake** source code and include files out of the repository and into the Bld directory. The Fortran compiler is then invoked to build the executable. If the executable does not exist after the Fortran compiler command completes, the script prints a notification and exits. If the executable does exist after the Fortran compiler command completes, the executable is moved into

the destination directory defined earlier in the script. Finally, the UNIX command `chmod` is used to set the correct permissions of the **bldmake** executable.

To configure the **bldmake** script for a new system, change the environment variable settings for the Fortran compiler and compiler flags in the config.cmaq script. The training script is set up for the training servers to use the Intel Fortran compiler. Exit from the script and invoke the following command to build **bldmake**:

```
> bldit.bldmake |& tee bldmake.log
```

After the script completes, check to be sure the **bldmake** executable exists by listing the contents of the directory $M3LIB/build and looking for the **bldmake** executable:

```
> ls $M3LIB/build
```

Note: Linux files that have executable permissions are identified with an "*" at the end of the filename. The "*" is not actually part of the filename, it is added by the system to denote which files have execute permissions. Notice how the bldmake file that you just created has an "*" appended to the file name. The bldit.bldmake script also has an "*" at the end of file name.

1.7 I/O API and netCDF libraries

F. Library files provide an easy way for programs to share commonly used subroutines. You need only name the library when linking the program, and those library modules that resolve references in the program are linked and merged into the executable file. CMAQ uses the Input/Output Applications Programming Interface (I/O API) and the network Common Data Form (netCDF) libraries to access a standard set of I/O routines and file format specifications for CMAQ. Fortran libraries are identified with the file extension *.a and often have names beginning with the prefix "lib" or just the letter "l".

The I/O API is an easy-to-learn, easy-to-use programming library for data storage and access, available for both Fortran and C. It is the standard data access library for CMAQ, MCIP, and SMOKE. The I/O API also contains an extensive set of utility routines for manipulating dates and times, performing coordinate conversions, storing and recalling grid definitions, performing sparse matrix arithmetic, etc., as well as a set of data-manipulation and statistical analysis programs.

NetCDF is an interface for array-oriented data access and a library that provides an implementation of the interface. The netCDF library also defines a machine-independent format for representing scientific data. Together, the interface, library, and format support the creation, access, and sharing of scientific data.

All of the libraries required by CMAQ are already set up for this training. A requirement for building CMAQ and MCIP is the availability of the I/O API and netCDF libraries. For this

training CMAQ version 5.0 will be built and run using the I/O API version 3.1 and netCDF version 4.1.1. These libraries can be downloaded and installed from the following locations:

I/O API: http://www.baronams.com/products/ioapi/AVAIL.html

netCDF: http://www.unidata.ucar.edu/software/netcdf/

For compiling both the I/O API and netCDF libraries for use with CMAQ, it is important to understand that the libraries and CMAQ must all be compiled with consistent compiler flags and options. This consistency in compiler options is especially true for compiling on Linux.

The default Portland Group Fortran 90 (pgf90) compiler options used for building CMAQ executables include:

- *-Mfixed* – use fixed-form formatting for f90 source code
- *-Mextend* – allow 132-column source code lines
- *-O3* – level 3 optimization for the executable

The default Intel Fortran 90 (ifort) compiler options used for building CMAQ executables include:

- *-fixed* – use fixed-form formatting for f90 source code
- *-132* – allow 132-column source code lines
- *-fno-alias* – disallow common memory locations to be accessed through different symbolic names in the program
- *-mp1* – improve floating point precision and consistency
- *-O3* – level 3 optimization for the executable

The default Gnu Fortran (gfort) compiler options used for building CMAQ executables include:

- *-ffixed-form* – use fixed-form formatting for f90 source code
- *-ffixed-line-length-132* – allow 132-column source code lines
- *-O3* – level 3 optimization for the executable
- *-funroll-loops* – optimization option for improving performance on iterative DO loops in the source code
- *-finit-character=32* – initialize local character variables to string of 32 bytes

This training uses the Intel Fortran compiler (ifort). The default Intel compiler options are augmented with the following compiler setting:

- *-vec-report0* – suppress messages about vectorizer diagnostics

1.8 CMAQ libraries

G. In addition to the I/O API and netCDF libraries, CMAQ also uses libraries required for parallel processing called the stencil exchange library, or Stenex for short; Parallel Input/Output library, or pario for short; and Dynamic Memory libraries, or dynmem for short.

   A. The stenex library is created using a **bldmake** script distributed with the CMAQ modeling system. Change directories to $M3HOME/scripts/stenex and list the directory contents with the following commands:

      ```
      > cd $M3HOME/scripts/stenex
      > ls
      ```

      Two **bldmake** scripts are located in this directory, one for compiling the Stenex library for multiprocessor applications (bldit.se) and the other for single-processor applications (bldit.se_noop).

   B. View the noop Stenex build script by typing:

      ```
      > gedit bldit.se_noop
      ```

      Note the similarities between the **bldmake** script for the Stenex library and the build script for the program **bldmake** examined previously in step 5.

   C. Exit from the Stenex library build script and execute it with the following command:

      ```
      > bldit.se_noop |& tee stenex_noop.log
      ```

      When executed successfully, the build script for Stenex writes the following message to the log:

      Finish building /home/training/CMAQv5.0/lib/x86_64/ifort/se_noop/libsef90_noop.a

   D. Check to be sure the Stenex library for single-processor applications exists by listing the contents of the directory $M3LIB/se_noop and looking for the Stenex library (remember library files are typically prepended with the string “lib” and end with the extension “.a”):

      ```
      > ls $M3LIB/se_noop
      ```

Exercise 1-4:
What is the name of the Stenex library file that will be used for this training?

Answer 1-4:

E. The Pario library is also created using **bldmake** scripts distributed with the CMAQ modeling system. To look in the directory housing the build script, type the following commands:

```
> cd $M3HOME/scripts/pario
> ls
```

As this training uses serial or single processor CMAQ executables, this library will not be created. The **bldmake** script that creates this library is similar to the script used to create the Stenex library and is invoked the same way. This library must be compiled to build the CMAQ chemical transport model for multiprocessor applications.

1.9 Training case

The CMAQ simulation that we will be working through during most of this training was developed from the U.S. National Park Service Air Resource Division (NPS-ARD) Rocky Mountain Atmospheric Nitrogen and Sulfur (RoMANS) study. A brief description of the case was presented during the classroom section that preceded this lab. The first part of the training involves configuring and running all of the CMAQ programs for a 12-km grid simulation that starts on July 7, 2006 and runs for 24 hours. Initial and boundary conditions are prepared using static profiles, meteorology inputs from MM5 are run through MCIP to create CMAQ-ready meteorology, clear-sky photolysis rates are computed using precompiled input data, and the emissions are derived from SMOKE outputs prepared prior to this training. The second part of the training involves configuring and running the necessary CMAQ programs for a 4-km grid simulation that is nested within the 12-km domain. Initial and boundary conditions for this nested grid are derived from the output concentration file from the 12-km simulation. MCIP is again used to prepare the meteorology inputs, and the emissions were prepared prior to this training. A restart of the 12-km simulation will then be performed to illustrate how to set up multiday simulations. This simulation will be initialized with the output concentration file from the previous 12-km simulation. The last part of the training involves setting up all of the CMAQ programs with only a description of the modeling case to guide the configuration and modeling process.

The following is a brief summary of some of the technical details of the simulations that will be run for most of this training. Refer back to this summary throughout the training if questions about the simulation arise.

Parameter	**Description**
Temporal Parameters	
Start Date	July 7, 2006
Start Time	00:00:00 GMT
Duration	48 hours
Spatial Parameters	
Description	12- and 4-km resolution grids focused on Denver, CO
Grid Names	(1) ROMANS12_50X50 (2) ROMANS4_50X50
Grid Resolution	(1) 12 km; (2) 4 km
Sigma Levels	"1.0, 0.997499, 0.995, 0.9899, 0.985, 0.9799, 0.97, 0.9599, 0.95, 0.93, 0.91, 0.88, 0.84, 0.65, 0.45, 0.25, 0.00"

This completes the CMAQ Overview exercise. You are now ready to move on to the second exercise, which addresses MCIP, the Meteorology-Chemistry Interface Processor.

2 CMAQ Training — MCIP, the Meteorology-Chemistry Interface Processor

2.1 Introduction

During the MCIP training, you will:

- Examine and execute a Fortran Makefile for building an executable
- Copy the MCIP run script, examine the script, and change options
- Create a set of meteorology input files for a coarse-grid CMAQ simulation
- Examine the MCIP log file
- Examine the MCIP output files in PAVE or VERDI

Recall that the symbol ">" is used throughout the training materials to represent the UNIX prompt. When this symbol appears as the first entry on a line in these instructions, you do not need to type it. Also, `Courier font` is used to indicate text that you should type at the prompt as you work through the exercises.

NOTE: To proceed with this exercise, you must have invoked the config.cmaq file, as described in Step 3 of the CMAQ Overview Exercise.

2.2 Creating the MCIP executable

1. In this step, you will learn to use a UNIX Makefile to build an MCIP executable. MCIP is distributed separately from the rest of the CMAQ modeling system. As such, it is configured to use a Makefile for building the MCIP executable, rather than relying on **bldmake** as the rest of the CMAQ modules do. The following steps guide you through the process of using a Makefile to build the MCIP executable.

   A. Go to the directory that contains the MCIP scripts.

   ```
   > cd $M3HOME/scripts/mcip4
   ```

   B. Go to the directory called src:

   ```
   > cd src
   ```

   C. Edit the Makefile using the following command:

   ```
   > gedit Makefile &
   ```

You will need to confirm that the Makefile is configured to create an executable for the operating system of your computer. The default Makefile is configured for Linux systems with Portland Group (PGF90) compilers. To configure the Makefile for a Linux system with Intel compilers, comment out the PGF90 compiler options using the "#" symbol. Six lines must be commented out under the PGF90 compiler options section of the Makefile (lines 28 and 33-37). Uncomment the Intel Fortran compiler options in the Makefile. Remove the "#" symbol from the lines that contain the Linux compiler options (lines 47 and 50-54). **NOTE:** You can display line numbers in the text editor that you are using to edit this file. Under the Preferences menu in gedit, invoke the "Statistics Line" option to see the file line numbers. Confirm that the Makefile is configured to use the correct path for the Fortran 90 compiler, I/O API library, netCDF library, and the netCDF include file on the training computer.

To build the MCIP executable, invoke the Makefile and redirect the standard output to a log file by typing the following command from the BLD directory:

```
> make |& tee make.mcip.log
```

2.3 Configuring and executing the MCIP run script for a 12-km simulation

2. In this step, you will copy the MCIP run script and configure it to create a set of CMAQ meteorology files from MM5 binary outputs.

   A. Copy the MCIP run script to one that uses the training tag ("training") and indicates this as a 12-km resolution simulation.

   ```
   > cd ..
   > cp run.mcip run.mcip_training_12k
   ```

   B. View your MCIP run script using the following command:

   ```
   > gedit run.mcip_training_12k &
   ```

   C. The APPL variable assigns a scenario tag to the MCIP simulation. Change the setting of the APPL variable to the word "training".

   D. Explore the script to get a feel for the different execution options available in MCIP version 4 and to answer the following questions.

Exercise 2-1:

(1) What is the start date and start time of the simulation that the MCIP script is set up to run?

(2) A feature of MCIP version 4 is the ability to read in fractional land use values from an MM5 TERRAIN file in order to calculate the percentage of urban area. If you wanted this new output, what three variables would you need to set?

(3) Another feature of MCIP version 4 is the ability for MCIP to use all of the layers from the input meteorology without specifying the full list in the script (*note that we will not use this option in this lab*). If you want to use this option, what should CTMLAYS be set to?

Answer 2-1:

(1)

(2)

(3)

E. Configure the MCIP run script to process meteorology data that starts on July 7, 2006 at 00 UTC and ends on July 8, 2006 at 00 UTC. (HINT: Change the MCIP_START and MCIP_END script variables). After checking the rest of the run script configuration (grid name, data directories, input file names, vertical layer structure), save the run script and exit from the editor.

F. From the $M3HOME/scripts/mcip directory, run the MCIP script, redirecting the output to a log file using the following command. MCIP should take less than 1 minute to complete this exercise.

```
> run.mcip_training_12k |& tee mcip_12k.log
```

G. View your log file using the following command:

```
> gedit mcip_12k.log &
```

If the program completes successfully, you would see this message at the bottom of the log file:

NORMAL TERMINATION

H. After getting MCIP to run to completion, examine the log file and answer the following question.

Exercise 2-2:

What are the output domain dimensions for this MCIP simulation (column, row, layers)?

Answer 2-2:

Exit from the MCIP log file after examining it for a few minutes. Notice how the log file contains metadata about the configurations of the horizontal and vertical domains. There is also information about which meteorology variables are and are not present in the input MM5 data.

2.4 Examining the MCIP output files

3. In this step, you will examine the MCIP outputs with the `ncdump` utility and use PAVE to compare them to the answer files. Note that the MCIP files are located in a directory that uses the name of the output grid in its path.

   A. Go to the MCIP output directory and examine each of the output files from MCIP:

   ```
   > cd $M3DATA/mcip/ROMANS12_50X50/2006188
   ```

   In this directory there should be 11 different output files produced by MCIP. Briefly, the mmheader and GRIDDESC files are text files describing the headers of the MM5 input file and the MCIP output grid, respectively. The MCIP GRID files are time-independent 2-D and 3-D files, and the MET files are time-dependent 2-D and 3-D files. The namelist file is a Fortran input produced by the MCIP run script that contains all of the settings for your simulation.

   B. Examine the METCRO3D file with the `ncdump` utility, using the following command:

   ```
   > ncdump METCRO3D_training | less
   ```

NOTE: `ncdump` is a netCDF utility (not a CMAQ utility) that allows you to view binary netCDF files as text. While in `ncdump`, you can scroll forward with the <space bar>, backward with <b>, and use <q> to quit.

Exercise 2-3:

(1) How many time steps are there in this file?
(2) What are the units of the variable PRES?

Answer 2-3:

(1)

(2)

Exit from `ncdump` ("q" to quit) when you are finished with this exercise.

C. Now start VERDI and look at the METCRO2D output file. (Make sure that you are still in the MCIP output directory.) **NOTE: If you are unfamiliar with VERDI, you may need assistance from the instructor(s) after starting it.**

```
> verdi -f $cwd/METCRO2D_training -f $ANSWERS/mcip/METCRO2D_training
```

- Create a tile plot of the variable PBL. There are three tabs at the bottom of the left pane on the VERDI interface: Datasets, Formulas, and Areas. From the Datasets tab, click on the name of the METCRO2D_training file in the top box of this tab. Next find PBL (M) in the Variables box and double click on it. Switch to the Formulas tab by single clicking the Formulas thumbnail at the bottom of the left VERDI pane. You should see that PBL was added to the Formulas box. Note that it will be appended with a "[1]" to denote that it's a variable from the first dataset loaded into VERDI. Single click on the PBL[1] variable in the Formulas box and then single click "Fast Tile Plot" along the top of the main VERDI window.

  You can then step through the different hours of the file using either the "Time Step" box or the animation buttons (<, |>, and >) to see how PBL heights change through the simulation day.

- Continue to experiment and explore with PAVE. For example, use the formula pop-up window to add the formula PBL[1]-PBL[2]. Create a tile plot of this formula to ensure that the meteorology fields for PBL in the file that you created are consistent with those in the answer file. Also, try different formulas/species and zoom functions.

- You can probe the values in different cells by selecting the Probe function from the Controls pull down menu above the plot. As you mouse over the plot you will see the cell number where the mouse is located at the bottom right of the VERDI window (column, row). Single clicking on the plot prints the value in that cell at the bottom left of the VERDI window with the format: (hour, layer, column, row): value. Dragging a box over several cells prints out an array of values in the selected cells.

Exercise 2-4:

What is the PBL height in cell (37,37) on July 7, 2006, at 11:00 UTC?

Answer 2-4:

Exercise 2-5:

Into which two MCIP output files is the variable DENS written?
(Hint: DENS is the total density of air and it is calculated in three dimensions.)

Answer 2-5:

This concludes the MCIP exercise. The concepts that you learned in this exercise will be applied later in Exercises 7 and 8 problem-solving exercises that integrate all of the CMAQ modules. You are now ready to move on to the third exercise, which addresses BCON, the boundary conditions processor.

3 CMAQ Training — BCON, the Boundary Conditions Processor

3.1 Introduction

During the BCON processor training, you will:

- Examine and execute the BCON build script
- Copy the BCON run script, examine the script, and change options
- Examine a BCON input file
- Execute BCON to create a boundary conditions (BC) input file for a coarse-grid CMAQ simulation
- Examine the BCON log file
- Examine the BCON output file in PAVE

NOTE: To proceed with this exercise, you must have invoked the config.cmaq file, as described in Step 3 of the CMAQ Overview Exercise.

3.2 Configuring BCON through the bldmake script

1. In this step, you will copy the BCON processor build script and configure it to create a BCON executable for a non-nested simulation.

   A. Go to the directory that contains the BCON scripts.

   ```
   > cd $M3HOME/scripts/bcon
   ```

   B. View the BCON build script using the following command:

   ```
   > gedit bldit.bcon &
   ```

   C. Examine the script. Note that at the top of the script, there is a pair of conditional statements that check to see if the $M3MODEL and $M3LIB directories exist. If these directories are not found, or the environment variables are not set, the following warning will be displayed:

   ```
   $M3MODEL or $M3LIB directory not found
   ```

   If you execute this script and see this warning, you need to go back and source the **config.cmaq** file to set the CMAQ environment variables (see Step 3 of Exercise 1).

D. In the script, see how the Project and GlobInc variables are used to point to the directories where the BCON source codes and libraries are located, respectively.

E. The APPL variable assigns a scenario tag to the BCON executable. The scenario tag should contain information on the nature of the BCON source data (more on this later).

F. The next section of the BCON build script lists several options for the program **bldmake**. These options provide **bldmake** instructions concerning what action to take in compiling the program. The default option, <verbose>, will compile and link together the Fortran object files to create a BCON executable in verbose mode, whereby all CVS, compiler, and linker output is printed to the terminal.

G. The "various modules" section of the build script provides the different configuration options for building BCON. The important configuration options in this section are the input data (ModType) and the chemical mechanism conversion (ModMech) flags.

   ModType. The BCON processor can generate BC files from one of three input sources:

   - *m3conc* – existing 3-D Models-3 concentration file, i.e., CCTM output file
   - *profile* – time-invariant set of vertical concentration profiles from a text file
   - *tracer* – chemically inert tracers

   ModMech. The BCON processor reads boundary condition concentration for different chemical mechanisms:

   - *cb05* –CB05 speciation
   - *saprc99* – SAPRC99 speciation
   - *saprc07t* – SAPRC07t speciation

H. The "mechanism" section of the build script is where you define the chemical mechanism that you will be modeling. The available mechanisms include:

   - *cb05cl_ae5_aq*: CB05 photochemical mechanism with chlorine chemistry, aerosol module 5, and aqueous chemistry
   - *cb05tucl_ae5_aq*: CB05 photochemical mechanism with updated toluene chemistry and chlorine chemistry, aerosol module 5, and aqueous chemistry
   - *cb05tucl_ae6_aq*: CB05 photochemical mechanism with updated toluene chemistry and chlorine chemistry, aerosol module 6, and aqueous chemistry
   - *cb05tump_ae6_aq*: CB05 photochemical mechanism with updated toluene chemistry, air toxics (including mercury and chlorine), aerosol module 6, and aqueous chemistry. Note that this is the CMAQ multi-pollutant (mp) mechanism
   - *saprc99_ae5_aq*: SAPRC99 photochemical mechanism, aerosol module 5, and aqueous chemistry
   - *saprc99_ae6_aq*: SAPRC99 photochemical mechanism, aerosol module 6, and aqueous chemistry

- *saprc07tb_ae6_aq*: SAPRC07tb photochemical mechanism, aerosol module 6, and aqueous chemistry
- *saprc07tc_ae6_aq*: SAPRC07tc photochemical mechanism, aerosol module 6, and aqueous chemistry

I. Exit from the default build script and copy it to a new script called bldit.bcon.profile using the following command:

```
> cp bldit.bcon.ifort bldit.bcon.profile
```

J. Edit the new BCON **bldmake** script and configure it to create an executable that uses time-invariant RADM2 input data for modeling the CB05 mechanism. To do this, change the settings for BCON to:

- ❑ set the APPL environment variable to *profile*
- ❑ read in profile data
- ❑ use CB05 speciation
- ❑ generate a BC file for modeling gas-phase chemistry with the CB05cl mechanism, heterogeneous chemistry with the 5th-generation aerosol model, and aqueous chemistry

Exercise 3-1:
List the variables that you changed or checked, and their values.

Answer 3-1:

Save the build script (but no need to exit from gedit).

3.3 Building the BCON executable

2. In this step, you will execute the BCON build script for creating a binary executable. From the $M3HOME/scripts/bcon directory, invoke the BCON build script, redirecting the standard output to a logfile:

```
> bldit.bcon.profile |& tee bcon.profile.log
```

Look for the executable file **BCON_profile_Linux2_x86_64ifort** in the $M3HOME/scripts/bcon directory to see if the build script completed successfully. If you do not see this file, examine the build log and make the changes to the configuration options in the build script that will allow it complete successfully.

Exercise 3-2:

What is the name of the directory that the build script created for installing and compiling a working version of the BCON code? (Hint: the directory name begins with the string "BLD", is located in the same directory as the build script, and uses the setting of the APPL environment variable in its name.)

Answer 3-2:

3.4 Configuring the BCON run script

3. Now you will copy the BCON processor run script and configure it to create a BC file based on a set of time-invariant chemical profiles.

   A. Copy the BCON run script to one that is identified with the tag *profile*:

   ```
   > cp run.bcon run.bcon.profile
   ```

   B. View your BCON run script using the following command:

   ```
   > gedit run.bcon.profile &
   ```

   C. The CFG variable assigns a configuration tag to the BCON executable. In this example, the CFG tag will be used to indicate that this is a training configuration. The APPL variable is used as an identifier for the BCON simulation and also used to identify the BCON executable. Change the setting of CFG to *training* and APPL to *profile*.

   D. The MECH variable must be set to be consistent with the "Mechanism" setting in the BCON build script. Change MECH in the BCON run script to cb05cl_ae5_aq

   E. The horizontal grid for the CMAQ modeling system is defined by the environment variable GRID_NAME, which is contained within the file GRIDDESC. Verify that the GRIDDESC file exists before proceeding with the simulation. Next, verify that the name and definition of the 12-km grid assigned to the GRID_NAME variable are present in the GRIDDESC file. The GRID_NAME used in this exercise is tagged with the grid resolution and the number of rows and columns (i.e., ROMANS12_50X50).

   F. BCON uses the METCRO3D file to assign the vertical layer structure of the output BC file. Verify that the LAYER_FILE variable is pointing to the METCRO3D file that you created in the MCIP lab.

Exercise 3-3:

(1) What is the name and path of the GRIDDESC file for the 12-km training grid?
(2) What is the name of the grid that you are modeling in this exercise?
(3) What is the name and path of the file that the LAYER_FILE environment variable should point to for the 12-km training simulation?

Answer 3-3:

(1)

(2)

(3)

G. The variable BC in the BCON run script determines whether the program will input a text file of vertical profiles or a netCDF 3-D concentration file.

Exercise 3-4:

Based on how you configured the BCON executable in Step 1 above, what should the setting for the variable BC be in this run script?

Answer 3-4:

H. Depending on the setting of the variable BC, the BCON run script points to different input files from which to generate a CMAQ boundary conditions file. If BC is set to <profile>, the environment variable BC_PROFILE points to a set of time-invariant vertical profiles. If BC is set to <m3conc>, the environment variable BC_PROFILE points to a netCDF 3-D concentration file.

Verify that the correct BCON input file for this simulation is being referenced by the BCON run script. The CMAQ boundary conditions concentration profiles are installed in the BLD directory automatically by **bldmake** based on the ModMech setting used to compiler the BCON executable.

Exercise 3-5:

What is the path and name of the BCON input file that will be used for this simulation?

Answer 3-5:

Save the BCON run script and exit from the editor.

3.5 Examine the BCON input file

4. Open the BCON input file in a text editor and examine the format of the file.

   A. Change directories to the CMAQ input file location and list the contents of the directory using the following commands:

```
> cd $M3HOME/scripts/bcon/BLD_profile
> ls
```

   B. Open the input data file for this exercise using the gedit text editor.

```
> gedit bc_profile_CB05.dat &
```

   C. The BCON input file can be broken down into four sections.

   - *Header* – A default text header describes that the vertical structure of the file uses a terrain-following sigma coordinate. The numeric header defines the format of the data in the file; the first number lists the number of sigma layers or columns of data, the second number describes the number of chemical species or rows of data, and the third through *n*th numbers are the values of the sigma coordinates.

   - *Date/Time Stamp* – A start date (YYYY-MM-DD) is used for creating a time-dependent BC file. For time-independent boundary conditions, these fields are ignored.

   - *Boundary Switch* – The BC input file is structured to allow the creation of independent chemical conditions at each of the four horizontal faces of the modeling domain. The boundary switch denotes which of the four faces (North, South, East, West) are described in the data section (next).

   - *Data Section* – Listing of the boundary conditions for every modeled chemical species at each sigma level as defined in the file header. The columns in the data section follow the order of the sigma values listed in the header.

Exercise 3-6:

Are the boundary conditions in the file bc_profile_CB05.dat uniform on all four sides of the domain? What are the values of the layer 1 boundary condition for CO on each of the four sides of the domain?

Answer 3-6:

North:	South:
East:	West:

3.6 Running BCON

5. In this step, you will execute the BCON run script to create a netCDF boundary conditions input file to CMAQ, examine the log file, and visualize the output file with VERDI.

   A. From the $M3HOME/scripts/bcon directory, invoke the BCON run script, redirecting the output to a log file using the following command:

   ```
   > run.bcon.profile |& tee bcon.profile_12k.log
   ```

   B. View your BCON log file using the following command:

   ```
   > gedit bcon.profile_12k.log &
   ```

   Look for the "successful completion" message at the bottom of the log file:

   ```
   >>----> Program BCON completed successfully <----<<
   ```

   If you do not see this message, you cannot proceed and must ask the instructor for help. If you do see this message, examine the rest of the BCON log file.

Exercise 3-7:

What are the units of the model species PAN as output by the BCON processor? (Hint: Look in the "Output File Section" of the log file.)

Answer 3-7:

You can exit from the BCON log file after completing this exercise.

C. Now go to the BCON output directory and examine the output file from BCON:

```
> cd $M3DATA/bcon
> ncdump BCON_profile_ROMANS12_50X50_training | less
```

NOTE: `ncdump` is a netCDF utility (not a CMAQ utility) that allows you to view binary netCDF files as text. While in `ncdump`, you can scroll forward with the <space bar>, backward with <b>, and use <q> to quit.

Exercise 3-8:

What is the value of the *x*-origin variable (XORIG) in the header of the BCON_profile_ROMANS12_50X50_training file? (Hint: You will have to scroll down to the "global attributes" section of the file header to get this information.)

Answer 3-8:

Exit from `ncdump` when you are finished with this exercise.

D. Now start PAVE and look at the BCON output file. (Make sure that you are still in the BCON output directory.) **NOTE: If you are unfamiliar with PAVE, you may need assistance from the instructor(s) after starting PAVE.**

```
> pave -f $cwd/BCON_profile_ROMANS12_50X50_training -f
$ANSWERS/bcon/BCON_profile_ROMANS12_50X50_training
```


- Create a tile plot using the "Graphics" menu item "Create Tile Plot". Select a variable in the PAVE Species List window. From the Graphics pull down menu on the main PAVE window, select "Create Tile Plot".
- You can do the same with the BCON profile file in the $ANSWERS/bcon directory to make sure you have processed the BCON file correctly. You will know that you have processed correctly if the tile plot of your file and tile plot of the reference data match exactly.
- Continue to experiment and explore with PAVE. For example, after loading the answers file into PAVE, use the formula pop-up window to add the formula O3a-O3b. Create a tile plot of this formula to ensure that the BCs for O_3 in the file that you created are consistent with the BCs for O_3 in the answer file. Also, try different formulas/species and zoom functions. As the BCON file is a 3-D file, you can also look at the boundary conditions in layers other than layer 1. To choose another layer to display, use "Select layer ranges matching current formula" under the "Formulas" pull-down menu.
- Compare the raw boundary conditions data in the BCON input file with the data in the gridded netCDF files and observe how the data have been reformatted by the BCON processor. You can open the raw BC text file using the following command from an xterm window:

```
> gedit $M3HOME/scripts/bcon/BLD_profile/bc_profile_CB05.dat &
```

Exercise 3-9:

What is the boundary condition in layer 1 for O_3 along the eastern edge of the domain? Hint: you can probe values in a cell or series of cells in PAVE by clicking or dragging on a cell or series of cells with your mouse.

Answer 3-9:

Exercise 3-10:

What is the boundary condition in layer 6 for O_3 along the southern edge of the domain?

Answer 3-10:

This concludes the BCON exercise. The concepts that you learned in this exercise will be applied later in Exercises 7 and 8 problem-solving exercises that integrate all of the CMAQ modules. You are now ready to move on to the fourth exercise, which addresses ICON, the initial conditions processor.

(Page left intentionally blank)

4 CMAQ Training — ICON, the Initial Conditions Processor

4.1 Introduction

During the ICON processor training, you will:

- Examine and execute the ICON build script.
- Copy the ICON run script, examine the script, and change options
- Examine an ICON input file
- Execute ICON to create an initial conditions (IC) input file for a coarse-grid CMAQ simulation
- Examine the ICON log file
- Examine the ICON output file in PAVE

NOTE: To proceed with this exercise, you must have sourced the config.cmaq file.

The ICON processor is very similar to BCON in configuration, execution, and input file requirements. To limit redundancy between the exercises, we have omitted the step-by-step instructions in this exercise for the exercises that closely parallel the BCON exercise. If you are not sure how to complete an exercise, either refer back to the similar step in the BCON exercise or ask your instructor.

4.2 Configuring ICON through the bldmake script

1. In this step, you will copy the ICON processor build script and configure it to create an ICON executable for a non-nested simulation.

   A. Go to the directory that contains the ICON scripts and view the ICON build script.

   ```
   > gedit bldit.icon &
   ```

   B. Examine the ICON build script, taking note of the similarities to the BCON build script.

   C. The APPL variable assigns a scenario tag to the ICON executable. The scenario tag should contain information on the nature of the ICON source data (more on this later).

   D. The next section of the ICON build script lists several options for the program **bldmake.** These options provide **bldmake** instructions concerning what action to take in compiling the program. The default option, <verbose>, will compile and link together the Fortran

object files to create an ICON executable in verbose mode, whereby all CVS, compiler, and linker output is printed to the terminal.

E. The "various modules" section of the build script provides the different configuration options for building ICON. The important configuration options in this section are the input data (ModInpt) and the chemical mechanism conversion (ModMech) flags.

ModType. The ICON processor can generate IC files from one of three input sources:

- *m3conc* – existing 3-D Models-3 concentration file, i.e., CCTM output file
- *profile* – time-invariant set of vertical concentration profiles from a text file
- *tracer* – chemical inert tracers

ModMech. The BCON processor reads boundary condition concentration for different chemical mechanisms:

- *cb05* –CB05 speciation
- *saprc99* – SAPRC99 speciation
- *saprc07t* – SAPRC07t speciation

F. The "mechanism" section of the build script is where you define the chemical mechanism that you will be modeling. The available mechanisms include:

- *cb05cl_ae5_aq*: CB05 photochemical mechanism with chlorine chemistry, aerosol module 5, and aqueous chemistry
- *cb05tucl_ae5_aq*: CB05 photochemical mechanism with updated toluene chemistry and chlorine chemistry, aerosol module 5, and aqueous chemistry
- *cb05tucl_ae6_aq*: CB05 photochemical mechanism with updated toluene chemistry and chlorine chemistry, aerosol module 6, and aqueous chemistry
- *cb05tump_ae6_aq*: CB05 photochemical mechanism with updated toluene chemistry, air toxics (including mercury and chlorine), aerosol module 6, and aqueous chemistry. Note that this is the CMAQ multi-pollutant (mp) mechanism
- *saprc99_ae5_aq*: SAPRC99 photochemical mechanism, aerosol module 5, and aqueous chemistry
- *saprc99_ae6_aq*: SAPRC99 photochemical mechanism, aerosol module 6, and aqueous chemistry
- *saprc07tb_ae6_aq*: SAPRC07tb photochemical mechanism, aerosol module 6, and aqueous chemistry
  saprc07tc_ae6_aq: SAPRC07tc photochemical mechanism, aerosol module 6, and aqueous chemistry

G. Exit from the default build script and copy it to a new script called bldit.icon.profile using the following command:

```
> cp bldit.icon bldit.icon.profile
```


H. Now configure the ICON **bldmake** script to create an executable that uses time-invariant input data speciated with the CB05 mechanism. To do this, change the settings for ICON to:

- ❑ set the APPL environment variable to *profile*
- ❑ read in profile data
- ❑ use the CB05 mechanism
- ❑ generate an IC file for modeling gas-phase chemistry with the CB05cl mechanism, heterogeneous chemistry with the 5th-generation aerosol model, and aqueous chemistry

Save the build script (but no need to exit from gedit).

4.3 Building the ICON executable

2. In this step, you will execute the ICON build script to create an executable file. From the $M3HOME/scripts/icon directory, invoke the ICON build script, redirecting the standard output to a log file:

```
> bldit.icon.profile |& tee icon.profile.log
```

Look for the executable file **ICON_profile_Linux2_x86_64ifort** in the ICON scripts directory to see if the build script completed successfully. If you do not see this file, examine the build log and make the changes to the configuration options in the build script that will allow it complete successfully.

4.4 Configuring the ICON run script

3. Now you will copy the ICON processor run script and configure it to create an IC file based on a set of time-invariant chemical profiles.

A. Copy the ICON run script to one that is identified with the tag profile:

```
> cp run.icon run.icon.profile
```

B. View your ICON run script. Set the CFG variable to *training* and set the APPL variable to the identifier used to tag the model executable (*profile*). Set the MECH variable to mechanism used to build the executable (cb05cl_ae5_aq). Set the GRIDDESC and GRID_NAME environment variables to point to the grid description file and the name of the grid definition contained in the grid description file for this training. Also set the LAYER_FILE environment variable to point to the METCRO3D file you created with MCIP.

Exercise 4-1:

What is the name of the horizontal grid for this exercise? (This is the setting for the variable GRID_NAME.)

Answer 4-1:

C. The variable IC in the ICON run script determines whether the program will input a text file of vertical profiles or a netCDF 3-D concentration file.

Exercise 4-2:

What should the setting for the variable IC be in this run script?

Answer 4-2:

D. Depending on the setting of the variable IC, the ICON run script points to different input files from which to generate a CMAQ initial conditions file. If IC is set to <profile>, the environment variable IC_PROFILE points to a set of time-invariant vertical profiles. If IC is set to <m3conc>, the environment variable IC_PROFILE points to a netCDF 3-D concentration file. Verify that the ICON run script is referencing the correct ICON input file for this simulation.

Exercise 4-3:

What is the path and name of the ICON input file that you need to use for this simulation?

Answer 4-3:

Save the ICON run script and exit from the editor.

4.5 Examine the ICON input file

4. Open the ICON input file in a text editor and examine the format of the file. Note the similarities to the BCON input file in the format of the data.

Exercise 4-4:
The previous exercise described the BCON input text file as having four sections. Which of the four sections in missing in the ICON input file?

Answer 4-4:

Exercise 4-5:
Why is this section missing from the ICON input file, and how does it affect the contents of the file?

Answer 4-5:

4.6 Running ICON

5. In this step, you will execute the ICON run script to create a netCDF initial conditions input file to CMAQ, examine the log file, and visualize the output file with PAVE.

   A. From the $M3HOME/scripts/icon directory, invoke the ICON run script, redirecting the output to a log file using the following command:

   ```
   > run.icon.profile |& tee icon.profile_12k.log
   ```

   B. View your ICON log file and look for the "successful completion" message at the bottom of the log file:

   ```
   >>----> Program ICON completed successfully <----<<
   ```

   If you do not see this message, you cannot proceed and must ask the instructor for help. If you do see this message, examine the rest of the ICON log file.

Exercise 4-6:

What are the units of the model species ASOIL as output by the ICON processor? (Hint: Look in the "Output File Section" of the log file.)

Answer 4-6:

You can exit from the ICON log file after completing this exercise.

C. Now go to the ICON output directory and examine the output file from ICON:

```
> cd $M3DATA/icon
> ncdump ICON_profile_ROMANS12_50X50_training | less
```

NOTE: Recall that while in `ncdump` you can scroll forward with the <space bar>, backward with <b>, and use <q> to quit.

Exercise 4-7:

What is the setting for the number of variables (var) in the header of the ICON_profile_ROMANS12_50X50_training file?

Answer 4-7:

Exit from `ncdump` when you are finished with this exercise.

D. Now load the ICON output file into VERDI. (Make sure that you are still in the ICON output directory.) **NOTE: If you are unfamiliar with VERDI, you may need assistance from the instructor(s) after starting VERDI.**

```
> verdi -f $cwd/ICON_profile_ROMANS12_50X50_training -f
$ANSWERS/icon/ICON_profile_ROMANS12_50X50_training
```

- Create a fast tile of different IC species
- You can do the same with the ICON output file in the $ANSWERS/icon directory to make sure you have processed the file correctly.
- Continue to experiment and explore with VERDI. For example, after loading the reference file into VERDI, use the formula pop-up window to add the formula NOa-NOb. Create a tile plot of this formula to ensure that the ICs for NO in the file that you created are consistent with the ICs for NO in the answer file. Also, try different formulas/species and zoom functions.

- Compare the raw initial conditions data in the ICON input file with the data in the gridded netCDF files and observe how the data have been reformatted by the ICON processor. You can open the raw IC text file using the following command from an xterm window:

```
> gedit $M3HOME/scripts/icon/BLD_profile/ic_profile_CB05.dat &
```

Exercise 4-8:

What is the initial condition in layer 1 for both the I and J modes of the aerosol species sulfate (ASO_4)? (Hint: you will need to sum both sulfate modes in the formula pop-up window, ASO4I+ASO4J.)

Answer 4-8:

Exercise 4-9:

What is the initial condition in layer 5 for CO?

Answer 4-9:

This concludes the ICON exercise. The concepts that you learned in this exercise will be applied later in Exercises 6 and 7 problem-solving exercises that integrate all of the CMAQ modules. You are now ready to move on to the fifth exercise, which addresses CCTM, the CMAQ Chemistry-Transport Model.

(Page left intentionally blank)

5 CMAQ Training — CCTM, the CMAQ Chemistry-Transport Model

5.1 Introduction

During the CCTM training, you will:

- Examine and execute the CCTM build script.
- Copy the CCTM run script, examine the script, and change options
- Use the input files that you prepared in the previous exercises to carry out a CCTM simulation
- Examine the CCTM log file
- Examine the CCTM output files in VERDI

NOTE: To proceed with this exercise, you must have sourced the config.cmaq file.

5.2 Configuring the CCTM through the bldmake script

1. In this step, you will copy the CCTM processor build script and configure it to create a CCTM executable for a non-nested simulation.

   A. Go to the directory that contains the CCTM scripts.

   ```
   > cd $M3HOME/scripts/cctm
   ```

   B. Create a working copy of the build script by copying the default CCTM build script to a new script called bldit.cctm.training using the following command:

   ```
   > cp bldit.cctm bldit.cctm.training
   ```

   C. View your CCTM build script using the following command:

   ```
   > gedit bldit.cctm.training &
   ```

   D. Change the APPL environment variable to *training*.

   E. The next section of the CCTM build script lists several options for the program **bldmake**. These options provide **bldmake** instructions concerning what action to take in compiling the program. The default option, <verbose>, will compile and link together the FORTRAN object files to create a CCTM executable in verbose mode, whereby all CVS, compiler, and linker output is printed to the terminal.

F. The "various modules" section of the build script provides the different configuration options for building the CCTM. The CCTM has quite a few configuration options. Brief details of these options follow.

ModHadv. The CCTM has one horizontal advection configuration option:
- *hyamo* – globally mass conserving scheme based on PPM

ModVadv. There are two vertical advection configuration options:
- *vwrf* – WRF-based omega calculation
- *vyamo* – globally mass conserving scheme based on PPM

ModHdiff. There is one horizontal diffusion option:
- *multiscale* – diffusion coefficient based on local wind deformation

ModVdiff. There are two vertical diffusion options:
- *acm2* – vertical diffusion calculation using the advanced convective method
- *acm2_mp* – ACM2 with multi-pollutant capabilities

ModDepv There are two deposition velocity options:
- *m3dry* – CMAQ dry deposition calculation, moved from MCIP into CMAQ
- *m3dry_mp* – CMAQ dry deposition with multi-pollutant capabilities

ModEmis Inline emissions processing module

ModBiod Inline biogenic (BEIS3) module

ModPlmrs Inline point source plume rise module

ModPhot. There are two photolysis driver options:
- *phot_table* – interpolate clear-sky photolysis rates from look up tables and apply cloud correction factors
- *phot_inline* – adjust photolysis calculations with predicted aerosols and radiation

ModCgrds There are two options for specifying chemistry input parameters:
- *cgrid_spcs_nml* – namelist file called at execution with species definitions
- *cgrid_spcs_icl* – include files called at compilation with species definitions

ModChem. There are eight chemistry solver options:
- *ros3* – Rosenbrock solver
- *smvgear* – Gear solver optimized for sparse matrix vector calculations, mechanism independent
- *ebi_cb05cl* – EBI solver configured for the CB05 mechanism with chlorine chemistry
- *ebi_cb05tucl* – EBI solver configured for the CB05 mechanism with updates to the toluene mechanism and including chlorine chemistry

- *ebi_cb05tump* – EBI solver configured for the CB05 mechanism with updates to the toluene mechanism and including air toxics, chlorine, and mercury; this is the multi-pollutant mechanism for CMAQ
- *ebi_saprc99* – EBI solver configured for the SAPRC99 mechanism
- *ebi_saprc07tb* – EBI solver configured for the SAPRC07tb mechanism
- *ebi_saprc07tc* – EBI solver configured for the SAPRC07tc mechanism

ModAero. There are three options for modeling aerosols:
- *aero5* – 5th generation CMAQ aerosol mechanism
- *aero6* – 6th generation CMAQ aerosol mechanism
- *aero6_mp* – 6th generation CMAQ aerosol mechanism, including air toxics and mercury

ModCloud. There are three options for modeling cloud physics and chemistry:
- *cloud_acm_ae5* – subgrid-scale cloud mixing algorithm based on the advanced convective method (ACM) with the 5th generation CMAQ aerosol mechanism
- *cloud_acm_ae6* – ACM cloud module with the 6th generation CMAQ aerosol mechanism
- *could_acm_ae6_mp* – ACM cloud module with the 6th generation CMAQ aerosol mechanism, including air toxics and mercury

G. The "mechanism" section of the build script is where you define the chemical mechanism that you will be modeling. A couple of notes on the nomenclature of the chemical mechanisms:

- *cb05cl_ae5_aq*: CB05 photochemical mechanism with chlorine chemistry, aerosol module 5, and aqueous chemistry
- *cb05tucl_ae5_aq*: CB05 photochemical mechanism with updated toluene chemistry and chlorine chemistry, aerosol module 5, and aqueous chemistry
- *cb05tucl_ae6_aq*: CB05 photochemical mechanism with updated toluene chemistry and chlorine chemistry, aerosol module 6, and aqueous chemistry
- *cb05tump_ae6_aq*: CB05 photochemical mechanism with updated toluene chemistry, air toxics (including mercury and chlorine), aerosol module 6, and aqueous chemistry. Note that this is the CMAQ multi-pollutant (mp) mechanism
- *saprc99_ae5_aq*: SAPRC99 photochemical mechanism, aerosol module 5, and aqueous chemistry
- *saprc99_ae6_aq*: SAPRC99 photochemical mechanism, aerosol module 6, and aqueous chemistry
- *saprc07tb_ae6_aq*: SAPRC07tb photochemical mechanism, aerosol module 6, and aqueous chemistry
- *saprc07tc_ae6_aq*: SAPRC07tc photochemical mechanism, aerosol module 6, and aqueous chemistry

H. The "set process analysis linkages" section of the build script sets the location of the process analysis configuration files.

I. Next, you need to configure the CCTM **bldmake** script to create an executable that uses the 5th-generation aerosol model, uses the EBI chemistry solver configured for CB05, no plume-in-grid, and no process analysis. To do this, change the settings for the CCTM to:

- ❑ use yamo for the driver module
- ❑ use gencoor for the concentration coupling module
- ❑ use the yamo method for both horizontal and vertical advection
- ❑ use multiscale horizontal diffusion
- ❑ use acm2 vertical diffusion
- ❑ use m3dry for the deposition velocity calculations
- ❑ use the namelist option for the configuring the chemical parameters
- ❑ use photolysis rates calculated inline
- ❑ use the EBI chemistry solver configured for CB05cl
- ❑ use the 5th-generation CMAQ aerosol mechanism
- ❑ use the ACM cloud model configured for the 5th generation CMAQ aerosol mechanism
- ❑ do not create process analysis output
- ❑ use CB05 speciation with chlorine chemistry, the 5th generation CMAQ aerosol model, and aqueous chemistry

Exercise 5-1:
List the variables that you changed or checked, and their values.

Answer 5-1:

Save the build script (but no need to exit from gedit).

5.3 Building the CCTM executable

2. In this step, you will execute the CCTM build script to create an executable file. From the $M3HOME/scripts/cctm directory, invoke the CCTM build script, redirecting the standard output to a log file:


```
> bldit.cctm.training |& tee cctm.training.log
```

Look for the executable file **CCTM_training_Linux2_x86_64ifort** in the CCTM scripts directory to see if the build script completed successfully. If you do not see this file, examine the build log and make the changes to the configuration options in the build script that will allow it to complete successfully.

5.4 Configuring the CCTM run script

3. Now you will copy the CCTM run script and configure it to run a 12-hour simulation of the 12-km modeling domain.

   A. Copy the CCTM run script to one that uses a tag to denote this simulation as a 12-km resolution training simulation:

   ```
   > cp run.cctm run.cctm.training_12k
   ```

   B. View your CCTM run script. Set the APPL variable to *training* to identify this as a training simulation. Set the CFG variable to *romans12* to identify this simulation as running on the 12-km RoMANS modeling grid.

   C. The "timestep run parameters" section of the CCTM run script defines the beginning date, beginning time in GMT, number of time steps, and the time step length for the simulation.

Exercise 5-2:

What is the default start date that the CCTM run script is set to run?

Answer 5-2:

Exercise 5-3:

If you wanted to simulate two days, rather than just 12 hours as this script is currently set up to do, what variable would you change and what would its value be?

Answer 5-3:

After completing the above two exercises, configure the script to start on July 7, 2006, at 000000 GMT and run for 12 hours.

D. To produce a CCTM log file, the environment variable "LOGFILE" must be set; it is located in the "set log file" section of the run script. Keep the "LOGFILE" environment variable commented out. We will manually create a log file by redirecting the standard output when we invoke the CCTM run script.

E. The environment variable "DISP" determines how existing output files are managed. By default, this is set to "keep", meaning that if output files already exist, the CCTM will not try to overwrite the existing files and will thus not execute successfully.

F. The horizontal grid for the CMAQ modeling system is defined by the environment variable GRID_NAME, which is contained within the file GRIDDESC. Confirm that the setting of GRID_NAME is consistent with the rest of the modeling completed thus far. Change the GRID_NAME setting if it is not set to the name of the grid that is being modeled.

G. The CCTM gives users the option of specifying the species and layers to write to the hourly instantaneous concentration file. The CONC_SPCS and CONC_BLEV_ELEV environment variables can be set to the model output species and layer range over which hourly concentrations will be written to the CCTM output CONC file. These options provide a way to reduce the size of the CCTM output files. The training run script is configured to output all layers and all species to the CONC file (the CONC_SPCS and CONC_BLEV_ELEV variables are commented out). In the Nested Simulation Lab you will need a complete 3-D CONC file for extracting boundary conditions for a nested 4-km domain. Do not change the settings of these variables for this training.

H. The CCTM creates integral-averaged concentration files over the time interval that you are running. These files are created in addition to the hourly concentration files, deposition files, and aerosol files that the CCTM creates by default. You can specify the layer range over which the integration will take place with the environment variable "ACONC_BLEV_ELEV". The training run script is configured to integrate over layer 1 only (setting = "1 1"). The ACONC file is not used for extracting boundary conditions concentrations.

Exercise 5-4:

If you wanted to produce integral-averaged concentrations over layers 3 through 5, what should "ACONC_BLEV_ELEV" be set to? (**NOTE:** Do not invoke this setting; keep the default.)

Answer 5-4:

I. The ACONC_END_TIME environment variable is used to specify whether the timestamp on the ACONC file is set to the beginning or end of each hour. The default behavior is to time stamp the ACONC files at the beginning of each hour. Setting this variable to "T" or "Y" will change this behavior and timestamp the ACONC file at the end of each hour.

J. The variables CTM_MAXSYNC and CTM_MINSYNC set the maximum and minimum synchronization time steps, respectively, for the model. The maximum synchronization time step must be adjusted for different model domain grid resolutions. Higher resolution domains require smaller synchronization time steps. Decreasing the setting of CTM_MAXSYNC from its default value of 720 seconds will improve the accuracy of the model solution, but it also creates a performance penalty to the model run time. While there is no rule of thumb for the CCTM in how CTM_MAXSYNC should be set, in general it's practical to set it toward the upper limit of where the model will converge on a solution.

K. The next several lines of the CCTM run script contain settings that control the behaviors of several run-time configuration options. Refer to the CMAQ Operational Guidance Document, available on the CMAQ Wiki through http://www.cmaq-model.org, for explanations of these settings.

L. CMAQ uses a dynamic minimum vertical diffusivity for urban and rural land use types. The environment variable KZMIN controls whether this feature is invoked during a CMAQ simulation. This feature in CMAQ requires that an additional meteorology variable that defines the urban fraction of grid cell be available. This urban land use fraction variable, PURB, is available with meteorology generated with either MM5 or WRF. KZMIN is set to "TRUE" by default in CMAQ. To deactivate this setting you must manually add the KZMIN environment variable to the CCTM run script and set it to "F".

M. The OCEAN_1 environment variable points to the optional ocean mask CCTM input file that is required when the CCTM is compiled with the aero4 or aero5 aerosol modules. The ocean file, also known as a sea salt mask (SSMASK) identifies the grid cells in the modeling domain that will have sea salt emissions. The ocean mask file can be generated

with the Spatial Allocator developed by the UNC Institute for the Environment and distributed by the CMAS Center[1].

N. The remainder of the script lists the CCTM input files and their locations. Examine these sections to confirm that the input files created in the previous exercises are consistent with the settings of this script.

O. Configure the CCTM run script to use the following settings:

- ❑ run on a single processor
- ❑ start on day 2006188 and run for 12 hours
- ❑ simulate the ROMANS 12km training grid
- ❑ output layer 1 O3, NO, CO, NO2, NO2, ASO4I, ASO4J, and NH3 to the ACONC file
- ❑ do not simulate windblown dust
- ❑ do not simulate lightning NOx
- ❑ use dynamic minimum vertical diffusivity
- ❑ calculate deposition velocities inline
- ❑ do not calculate landuse-specific deposition velocities
- ❑ calculate bi-directional fluxes for neither ammonia nor mercury
- ❑ simulate surface HONO interactions
- ❑ use neither inline biogenic nor plume rise calculations

Exercise 5-5:
List the variables that you changed or checked, and their values.

Answer 5-5:

Save the run script and exit from the editor.

[1] http://www.ie.unc.edu/cempd/projects/mims/spatial/

5.5 Executing the CCTM run script

4. In this step, you will execute the CCTM run script to create a set of netCDF output files, examine the log file, and visualize the output files with VERDI.

   A. From the $M3HOME/scripts/cctm directory, invoke the CCTM run script, redirecting the output to a log file using the following command:

   ```
   > run.cctm.training_12k |& tee cctm.training_12k.log
   ```

   B. The CCTM will take about 15-20 minutes to complete. While waiting for the run to finish, skip to **Section 7: Problem Solving – Nested Simulations**

   Once the CCTM has finished running, use the following command to view the CMAQ log file:

   ```
   > gedit cctm.training_12k.log &
   ```

   To ensure that the CCTM completed successfully, look for the normal completion message at the end of the log file:

   ```
   >>----->  Program completed successfully  <-----<<
   ```

   Exercise 5-6:
   To what file does the environment variable EMIS_1 point?

   Answer 5-6:

   You can exit from the CCTM log file after completing this exercise.

   C. Now go to the CCTM output directory and examine the output files from the CCTM:

   ```
   > cd $M3DATA/cctm
   > ncdump CCTM.ACONC.romans12.ACONC.2006188_000000.ncf | less
   ```

Exercise 5-7:

What is the start date and start time of the ACONC file? (Hint: You will have to scroll down to the *global attributes* section of the file header to get this information.)

Answer 5-7:

Start Date =

Start Time =

Exit from ncdump when you are finished with this exercise.

D. Now start VERDI and look at the CCTM ACONC file. (Make sure that you are still in the CCTM output directory.) **NOTE: If you are unfamiliar with VERDI, you may need assistance from the instructor(s) after starting VERDI.**

```
> verdi -f $cwd/CCTM.romans12.ACONC.2006188_000000.ncf -f
$ANSWERS/cctm/CCTM.romans12.ACONC.2006188_000000.ncf
```

- Create an O_3 tile plot using the "Fast Tile Plot" option in VERDI.
- Create a time-series plot using the "Time Series" option in VERDI.
- You can do the same with the CCTM ACONC file in the $ANSWERS/cctm directory to make sure you have processed correctly.
- Continue to experiment and explore with VERDI. For example, after loading the answers file into VERDI, use the formula window to add the formula O3[1]-O3[2]. Create a tile plot of this formula to ensure that the concentrations for O_3 in the file that you created are consistent with the concentrations for O_3 in the answer file. Also, try different formulas/species and zoom functions.
- Load the other CCTM files into VERDI and compare them to the answer files in the $ANSWERS/cctm directory. Make sure that you have reproduced the answers exactly.

This concludes the CCTM exercise. The concepts that you learned in this exercise will be applied in Exercises 6 and 7, problem-solving exercises that integrate all of the CMAQ modules. The first of these two exercises covers problem solving with nested simulations.

6 CMAQ Training — Problem Solving: Nested Simulations

NOTE: This exercise assumes that you have completed the previous six CMAQ exercises and thus have a basic knowledge of how to build and run the CMAQ modules. For complete details on how to use the **bldmake** scripts and invoke the run scripts, refer back to the exercises for the specific CMAQ modules.

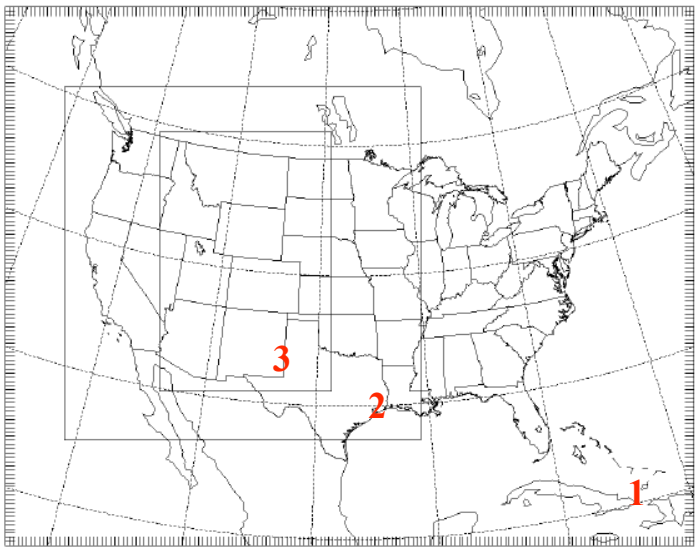
6.1 Introduction

During this problem-solving exercise, you will:

- Learn how to configure and run MCIP for a nested simulation
- Learn how to configure and run the ICON processor for a nested simulation
- Learn how to configure and run the BCON processor for a nested simulation
- Learn how to configure and run the CCTM for a nested simulation
- Examine the nested CCTM output files in VERDI

NOTE: To proceed with this exercise, you must have sourced the config.cmaq file**.**

In the context of air quality modeling, a nest refers to a fine-grid subdomain that draws its boundary conditions from a coarser-grid parent domain. Model domain nesting may be recursive, where consecutively finer scale domains extract boundary conditions from a preceding nested domain. For example, in the figure below Domain 1 represents the parent domain. Domain 2 is a nest that uses boundary conditions extracted from domain 1. Domain 3 is an inner nest that uses boundary conditions from domain 2.



In this exercise, you will be simulating a 4-km grid resolution nest within the 12-km resolution parent grid that you modeled in the previous exercises.

Operationally, performing a nested simulation requires changes to the way in which the initial and boundary condition files are prepared for the CCTM. In this exercise, BCs for the nested 4-km simulation will be extracted from the CCTM concentration file for the 12-km domain that you simulated in the previous CMAQ exercises. The nested configuration will require you to create a new executable for the BCON processor so it can use a different type of input data than in the previous run. The nested configuration for ICON requires generating ICs on the fine-grid domain using the same input data and executable as we used previously.

The Meteorology-Chemistry Interface Processor (MCIP, discussed in Exercise 2) and the CMAQ Chemistry-Transport Model (the CCTM, discussed in Exercise 5) do not extract the nested, fine-grid data from the coarse domain. Instead, MCIP and the CCTM require input data (e.g., from MM5 or WRF, processed through MCIP) that is at the same resolution as the grid for which they will be running (either coarse or fine). For MCIP and the CCTM we will use the same executables that we created in those previous exercises, and will simply adjust the grid definitions to produce simulations at a grid cell resolution of 4 km.

Emissions inputs for the 4-km domain were prepared in advance, outside of this course.

6.2 Generating meteorology files for a nested simulation

1. In this step, you will edit the MCIP run script and configure it to create a set of CMAQ meteorology files from MM5 binary outputs for a 4-km grid resolution domain. The MCIP simulation for the 12-km domain was run for 25-hours in order to generate meteorology input data for the two 12-hour simulation periods. MCIP will be used to process only 13-hours of meteorology data for the 4-km nested domain.

   A. Go to the directory that contains the MCIP scripts:

   ```
   > cd $M3HOME/scripts/mcip4
   ```

   B. Create and view an MCIP run script for creating 4-km resolution meteorology data for input to the CCTM:

   ```
   > cp run.mcip_training_12k run.mcip_training_4k
   > gedit run.mcip_training_4k &
   ```

   C. The APPL environment variable should already reflect that this is a training exercise. Verify that this variable is set correctly.

   D. Next you will define the new domain. To do this, adjust the “GridName” variable in the run script by changing it’s setting from the name of the 12-km grid to the name of the 4-km grid. (HINT: Look at the last page of the Overview lab to find the name of the 4-km grid).

E. The variable "InMetFiles" points to the input files for MCIP. Change the input data file names to point to the MM5 binary files tagged "DOMAIN3" for the 4-km grid. Look in the directory $M3DATA/raw/MM5/DOMAIN3 to find the names of the input meteorology files for the 4-km grid.

F. Also change the input data file name to point to the MM5 Terrain files tagged "DOMAIN3".

G. Change the MCIP_END variable to stop the simulation at hour 12 on July 7, 2006.

H. Because the 4-km MM5 domain is larger than the 4-km CMAQ modeling domain, MCIP will be used to window the meteorology domain to the desired specifications. The MM5 and CMAQ 4-km grid definitions are provided below.

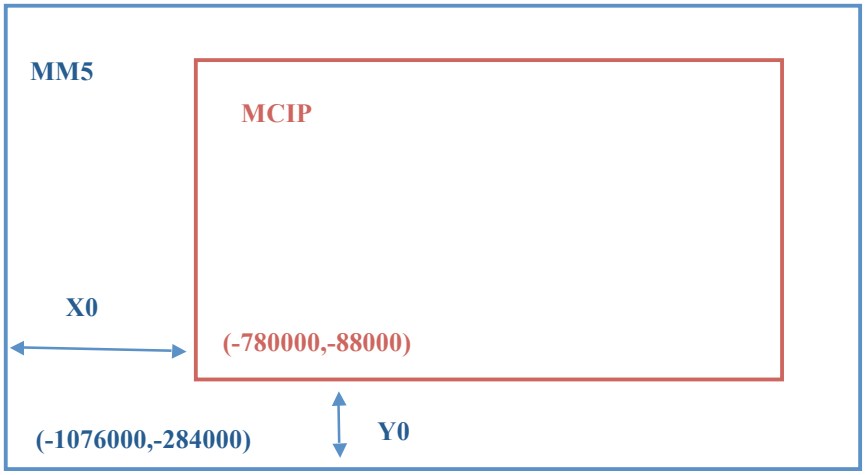
Parameter	**MM5**	**CMAQ**
Alpha (degrees)	33.0	33.0
Beta (degrees)	45.0	45.0
Gamma (degrees)	-97.0	-97.0
X-center (degrees)	-97.0	-97.0
Y-center (degrees)	40.0	40.0
Columns(	160	50
Rows	115	50
X-origin (m)	-1,076,000	-780,000
Y-origin (m)	-284,000	-88,000
dx (m)	4,000	4,000
dy (m)	4,000	4,000

The MCIP variables BTRIM, X0, Y0, NCOLS, and NROWS are used to select a subset of the meteorology domain. BTRIM is used to set the number of meteorology boundary points to remove on each of the four horizontal sides of the MCIP domain. Setting BTRIM = 0 will use the maximum number of grid cells in the input meteorology. Setting BTRIM = -1 indicates that a specific subdomain will be selected using X0, Y0, NCOLS, and NROWS.

An MCIP subset domain is configured by setting BTRIM = -1. The following parameters are used to define the meteorology subdomain:

- X0 is the X-coordinate of the lower left corner of the MCIP domain (including the MCIP lateral boundary) relative to the meteorology domain.
- Y0 is the Y-coordinate of the lower left corner of the MCIP domain (including the MCIP lateral boundary) relative to the meteorology domain.
- NCOLS and NROWS set the number of columns and rows, respectively, for the output MCIP domain.

For this exercise the figure below illustrates how to calculate X0 and Y0 to define the 4-km MCIP subset domain. Note the addition of the single lateral boundary cell to each calculation



X0 = (-780000 – -1076000)/dx = 998000/4000 = 74 + 1 = 75
Y0 = (-88000 – -284000)/dy = 196000/4000 = 49 + 1 = 50

Based on this calculation and the data in the table above, set X0 = 75, Y0 = 50, NCOLS = 50, and NROWS = 50 in the MCIP runs script for processing the 4-km MM5 data.

I. Leave the rest of the settings in the MCIP script the same. Save the run script (but no need to exit from gedit).

J. Invoke the MCIP run script, redirecting the output to a log file:

```
> run.mcip_training_4k |& tee mcip_4k.log
```

Upon completion of MCIP, look at the log file to ensure that this step completed successfully.

Exercise 6-1:

What are the output domain dimensions for this MCIP simulation (column, row, layers)?

Answer 6-1:

Exit from the MCIP log file after examining it for a few minutes.

6.3 Generating initial conditions for a nested simulation

2. In this step, you will edit the ICON run script and use it to create initial conditions for a 4-km grid that is nested within the 12-km domain that you modeled in earlier exercises. You will simply be assigning the profile data you used in the earlier exercise to the new grid.

   A. Create a new ICON run script and view it using the following commands:

   ```
   > cp run.icon.profile run.icon.profile_4k
   > gedit run.icon.profile_4k &
   ```

   B. Configure the script to create ICs on the nested domain. In the "horizontal grid definition" section of the run script, change the grid name from the name of the 12-km grid, "ROMANS12_50X50", to the name of the 4-km grid, "ROMANS4_50X50".

   C. Configure the 3-D Layer File to point to the file for the 4-km grid. Then save the run script (but no need to exit from gedit).

   D. Invoke the ICON run script, redirecting the output to a log file:

   ```
   > run.icon.profile_4k |& tee icon.profile_4k.log
   ```

   Look at the log file to ensure that this step completed successfully.

6.4 Generating boundary conditions for a nested simulation

3. In this step, you will edit the BCON build script and apply it to create an executable that will use an existing coarse-grid CCTM concentration file as input.

   A. Go to the directory that contains the BCON scripts:

   ```
   > cd $M3HOME/scripts/bcon
   ```

   B. Create a new BCON **bldmake** run script using the following command:

   ```
   > cp bldit.bcon.profile bldit.bcon.m3conc
   ```

   C. Configure the new BCON **bldmake** script to create an executable that uses a Models-3 concentration input file with no mechanism conversion. To do this, change the settings for BCON to:

   - ❑ change the APPL variable to m3conc to indicate that the new executable will be used to generate BCs from an existing concentration file
   - ❑ read in m3conc data
   - ❑ use the gas-phase mechanism cb05
   - ❑ generate a BC file for modeling gas-phase chemistry with the CB05cl mechanism, heterogeneous chemistry with the 5th-generation CMAQ aerosol model, aqueous chemistry.

Exercise 6-2:
List the variables that you changed or checked, and their values.

Answer 6-2:

Save the build script (but no need to exit from gedit).

D. Invoke the BCON build script and redirect the standard output to a log file:

```
> bldit.bcon.m3conc |& tee bcon.m3conc.log
```

Check the log file to ensure that the build completed successfully. If it did not, make the changes to the configuration options that will allow the build script to complete successfully.

6.5 Configuring and executing the BCON run script

4. In this step, you will edit the BCON run script and use it to create boundary conditions for an 4-km grid that is nested within the 12-km domain that you modeled in earlier exercises.

A. Create a new BCON run script and view it using the following commands:

```
> cp run.bcon.profile run.bcon.m3conc
> gedit run.bcon.m3conc &
```

B. Change the input data reference in the APPL environment variable to “m3conc”.

C. To define the nested domain, look in the “horizontal grid definition” section of the run script, and change the grid name from the name of the 12-km grid, “ROMANS12_50X50”, to the name of the 4-km grid, “ROMANS4_50X50”.

D. Change the directory path to the correct layer file for the 4-km grid.

E. Change the setting of the BC environment variable, which determines the nature of the input data. In Step 3 of Exercise 3, we set BC to *profile*, which caused the environment variable BC_PROFILE to point to a set of time-invariant vertical concentration profiles from a text file. Here, change the BC variable in the script to *m3conc*, which causes BC_PROFILE to point to a CMAQ concentration file as input rather than a static set of profiles.

F. Change the date for the output file name to the Julian date for July 7, 2006 and change the run length to 12 hours. Confirm that the 12-km concentration file created in the

CCTM exercise is being used as input for this exercise. You must point BCON to the 12-km CONC file in order to extract the 4-km boundary conditions from this file.

Save the run script (but no need to exit from gedit).

G. Invoke the BCON run script, redirecting the output to a log file:

```
> run.bcon.m3conc |& tee bcon.m3conc_4k.log
```

Look at the log file to ensure that this step completed successfully.

6.6 Running a nested simulation with the CCTM

5. In this step, you will edit and execute the CCTM run script to create a set of netCDF output files for a 4-km resolution domain that is nested within the 12-km domain that you ran previously. You will then visualize the output files with PAVE and compare them to the 12-km simulation results.

   A. Go to the directory that contains the CCTM scripts:

   ```
   > cd $M3HOME/scripts/cctm
   ```

   B. Create and view a CCTM run script for the nested 4-km resolution simulation.

   C. Verify that the APPL environment variable already reflects that this is a training exercise.

   D. Configure the CCTM run script to simulate the 4-km grid resolution case that you have prepared meteorology, IC, and BC inputs for. To do this, change the following settings for the CCTM run script:

   - ❑ Set the CFG variabile to *romans4*
   - ❑ set the "GRID_NAME" environment variable to the name of the 4-km grid
   - ❑ confirm that you are using a GRIDDESC file that contains the grid name
   - ❑ set the emissions input variable to point to the 4-km resolution gridded emissions
   - ❑ use the 4-km ocean mask file (OCEAN_1)
   - ❑ use the 4-km IC file generated from the profiles file
   - ❑ use the 4-km BC file generated from the 12-km CMAQ concentration (CONC) file
   - ❑ set the meteorology input directory to the location of the 4-km resolution files

Exercise 6-3:

List the variables that you changed or checked.

Answer 6-3:

Save the run script (but no need to exit from gedit).

E. From the $M3HOME/scripts/cctm directory, execute the CCTM run script, redirecting the output to a log file using the following command:

```
> run.cctm.training_4k |& tee cctm.training_4km.log &
```

The CCTM will take about 15-20 minutes to complete.

Upon completion of the CCTM, look at the log file to ensure that this step completed successfully. Exit from the CCTM log file after examining it for a few minutes.

6.7 Examining the nested output files with VERDI

6. In this step, you will load the 4-km and 12-km CCTM ACONC (concentration) output files into VERDI and compare the results from the two simulations.

Go to the CCTM output directory and import the 12- and 4-km CONC files using the VERDI command line option "-f". **NOTE: If you are unfamiliar with VERDI, you may need assistance from the instructor(s) after starting VERDI.**

- Create tile plots of O_3 using the "Fast Tile Plot" feature for both output files.
- Zoom in on the area of the 12-km domain that is covered by the 4-km simulation.
- Normalize the scales on the two plots so that they are the same.
- Note that the Denver, CO metropolitan area is located right in the middle of the 4-km domain. It is more highly resolved in the 4-km simulation than the 12-km simulation.
- Continue to experiment and explore with VERDI. Look at other pollutants. Load in the reference data for the 4-km simulation located in the $ANSWERS/nesting directory, then check to make sure you ran the nesting exercise correctly.

This concludes the nesting exercise. You are now ready to move on to the seventh exercise, "Problem Solving: Multiday Simulations."

7 CMAQ Training — Problem Solving: Multiday Simulations

NOTE: This exercise assumes that you have completed Lab 6 and thus have a basic knowledge of how to build and run the CMAQ modules. For complete details on how to use the **bldmake** scripts and invoke the run scripts, refer back to the exercises for the specific CMAQ modules.

7.1 Introduction

During this problem-solving exercise, you will:

- Learn how to configure and run the CCTM to restart a simulation
- Practice different scripting techniques for running the CCTM for multiple days

NOTE: To proceed with this exercise, you must have sourced the config.cmaq file.

This exercise teaches you to restart a CCTM simulation using output from a previous run for initialization. By learning to restart the CCTM from a point at which it exited normally, you will become familiar with the technique of configuring the CCTM for a multiday simulation. Techniques for chaining multiple days of runs together will also be discussed at the end of this exercise. Additional discussion on running multi-day CMAQ simulations is available in Chapter 10 of the CMAQ Operational Guidance Document.

7.2 Restarting a CCTM simulation

1. In this step, you will edit and execute the CCTM run script to conduct a 12-km simulation for July 7, 2006 from 12 to 24 UTC, which continues the simulation started in the CCTM exercise (Lab 6). After completing this step, you will visualize the output files with VERDI and compare them to output from the previous time period.

   A. Go to the directory that contains the CCTM scripts:

   ```
   > cd $M3HOME/scripts/cctm
   ```

   B. Create and view a CCTM run script for the 12-km resolution simulation using the following commands:

   ```
   > cp run.cctm.training_12k run.cctm.training_12k_restart
   > gedit run.cctm.training_12k_restart &
   ```


C. Configure the CCTM run script to simulate a 32-km grid resolution case for July 3, 1999 that uses the meteorology, IC, and BC inputs that you prepared in Labs 2 through 5. Change the following settings for the CCTM run script:

- ❑ set the start date to 2006188
- ❑ set the start time 120000
- ❑ set the "GRID_NAME" environment variable to the name of the 12-km grid
- ❑ set the emissions input variable to point to the 12-km resolution gridded emissions for July 7, 2006
- ❑ use the 12-km CGRID file created in the CCTM exercise (Lab 6) as the IC file
- ❑ use the 12-km BC file generated from the vertical profiles file
- ❑ set the meteorology input directory to the location of the 12-km resolution MCIP output files

Exercise 7-1:
List the variables that you changed or checked.

Answer 7-1:

Save the run script (but no need to exit from gedit).

D. From the $M3HOME/scripts/cctm directory, invoke the CCTM run script, redirecting the output to a log file using the following command:

```
> run.cctm.training_12k_restart |& tee cctm.training_12k_restart.log
```

The CCTM will take about 15-20 minutes to complete. When it has finished, look at the log file to ensure that this step completed successfully. Exit from the CCTM log file after examining it for a few minutes.

7.3 Examining the restarted CCTM output files with VERDI

2. In this step, you will load the 12-km CCTM CONC files for the hours from 00-12 and 12-24 GMT into VERDI and compare the results from the two simulations.

Go to the CCTM output directory, start VERDI, and import both of the 12-km CONC files that you created during this training.

- Create tile plots of O_3 using the "Fast Tile Plot" VERDI option. Confirm that the simulated O_3 concentration in the last hour of the simulation started at 0Z are the same as the first hour of the simulation started at 12Z.
- Continue to experiment and explore with VERDI. Look at other pollutants. Load in the reference data for the July 7, 2006 12-km simulation located in the $ANSWERS/restart directory, then check to make sure you ran the multiday simulation exercise correctly.

7.4 Scripting for multiday simulations

3. Regional air quality models like CMAQ are almost always used to simulate multiday time periods ranging from several days to a year or more. The length of time that you choose for your simulation is determined in part by the temporal coverage of your meteorology and emissions input data. The meteorology simulations are commonly performed for 4- or 5-day periods (the period used is often dependent on the spatial extent of the modeling domain). While it is possible to mirror the meteorology time period when preparing the related emissions and air quality modeling simulations, output file sizes for simulations with durations of 4-5 days quickly become unmanageable. So, although it is somewhat a matter of personal style, it is usually a good idea to set up simulations in 24-hour intervals. Along these lines, you can envision four situations that you may encounter in setting up multiple day CMAQ simulations:

    1. Single-day input files → Single-day output files
    2. Multiday input files → Multiday output files
    3. Single-day input files → Multiday output files
    4. Multiday input files → Single-day output files

    The easiest way to configure your CMAQ run scripts is to have the time period covered by your output files be the same as the time period covered by your input files. It is possible, however, to accommodate any of the above situations in the run scripts. A few points to consider when setting up simulations over multiple days:

    - Be consistent with the names of the time-dependent input files. Minimizing the variation in the file names for sequential input files will simplify the scripting for calling these files in a simulation.
    - Use environment variables in the naming of input and output files as much as possible. The less you hard-wire file names in your scripts, the easier it will be to modify the scripts for new grids and episodes.
    - Minimize the number of scripts that you create for running multiday simulations. Let the scripts do the work for you. It is unnecessary to create a script for each day you are simulating. It is more efficient and less error-prone to create one script that uses conditional scripting techniques (such as "if", "while", and "foreach" statements) to loop over multiple days.

Now you will practice scripting for extended simulation periods.

A. Copy the CCTM run script to a new script.

B. Configure the new CCTM run script to run for five days, assuming a situation where you have both *multiday* input files and *multiday* output files.

Exercise 7-2:
How did you modify the script to set up a five-day simulation?

Answer 7-2:

C. Configure the new CCTM run script to run for 5 days, assuming a situation where you have both *single-day* input files and *single-day* output files.

Exercise 7-3:
How did you modify the script to set up a five-day simulation? (Note: there are several ways to accomplish this task.)

Answer 7-3:

This concludes the multiday simulation exercise. You are now ready to move on to the eigth exercise, “Problem Solving: Lightning NOx Emissions”.

8 CMAQ Training — Problem Solving: Lightning NOx Emissions

8.1 Introduction

During this problem solving exercise, you will:

- Review the steps required for interpolating lightning strike data to a new modeling grid
- Build and run the CMAQ lightning NOx emissions preprocessor
- Plot the lighting NOx preprocessor output data and examine the result
- Execute a new CCTM simulation that includes lightning NOx emissions and compare these results to the simulation that did not include lightning NOx.

NOTE: To proceed with this exercise, you must have sourced the config.cmaq file.

8.2 Preparing lightning NOx input data for the CCTM

1. In this step, you will review the steps necessary for preparing lighting NOx (herein referred to as LNOx) input data for a CMAQ simulation. Lightning is the largest source of nitrogen in the upper troposphere. NO_x produced by lightning influences ambient nitrogen concentrations and wet deposition of total nitrates within the boundary layer. To simulate the impacts of LNOx emissions on ambient gas and particle concentrations and on deposition, CMAQv5 is instrumented with an LNOx module. The CCTM includes three options for calculating LNOX emissions:

   - Read an input file of LNOx emissions that were calculated off-line
   - Compute LNOx emissions in-line during a simulation directly from lightning flash counts estimated from the convective precipitation field in the input meteorology
   - Compute LNOx emissions in-line using a combination of observed flash counts and simulated convective precipitation.

   The second option is recommended for hemispheric-scale simulations or when no flash-count observations are available, while the third option is recommended for all simulations at less than 50-km horizontal grid resolutions. The National Lightning Detection Network (NLDN) is a source of flash count observations for North America that can be used as an input to CCTM. The third option will be used in this exercise to estimate the impacts of lightning NOx on air quality concentrations in the 12-km modeling domain.

   The general approach that you will follow in this exercise is to run the CMAQ LNOx preprocessor, LTNG_2D_DATA, to generate a lightning parameters file using input data provided in the CMAQ distribution package. Before running the program

LTNG_2D_DATA it's necessary to run a series of scripts that interpolate the input data to the modeling grid that you are simulating with the CCTM. For the sake of time, these data have already been preprocessed for the 12-km training domain. The following datasets were prepared in advance for this training exercise:

- Inter-cloud to cloud-to-ground ratios (ICCG)
- Ocean mask
- NLDN monthly flash count totals

A. Go to the directory that contains the LNOx pre-processing scripts

```
> cd $M3HOME/scripts/lnox
```

B. Build the program LTNG_2D_DATA

```
> cd src
> make -f Makefile.LTNG_2D_DATA
```

Confirm that the executable LTNG_2D_DATA was generated by listing the contents of the src directory. If the executable exists, back out one directory to the location of the run scripts.

E. Create a copy of the script run.LTNG_2D.csh and edit the script to prepare input data for the RoMANS12 modeling grid. There are four variables at that top of the script that control different processes that are run by the script. You do not need to create the ocean mask file or flash ratios file. Configure the script to create a CMAQ input LNOx file and to create LNOx plots. Other configuration options to check or change in this script include:

- ❑ Simulation time period (month and year)
- ❑ RoMANS 12-km METCRO2D file (METFILE)
- ❑ RoMANS 12-km monthly flash density file (NLDNFILE)

C. After configuring the run.LTNG_2D script, save and run to generate a CCTM LNOx parameters file. This script will create an output netCDF file in the directory $M3DATA/lnox. Note that this is the file that will be used as input to the CCTM. The script also writes a PDF file of graphics to the directory $M3HOME/scripts/lnox/R-out. Both of these files contain the same information. Either use VERDI to visualize the variables in the netCDF output file or use the command "display" to view the contents of the graphics PDF file. Notice the differences in the flash counts estimated from the MCIP precipitation field (LTstrk) and the flash counts in the NLDN dataset (NLDNstrk).

D. Now configure the CCTM to run a 12 km simulation that includes inline LNOx emissions. Here are a few hints to help set this simulation up:

- ❑ Change the CFG variable to label the outputs from the simulation as being a lightning NOx sensitivity. This step is important because you don't want to overwrite the results from the base 12-km simulation.
- ❑ Turn on the lightning NOx capability in the CCTM through the variable CTM_LTNG_NO
- ❑ Configure the lightning NOx module to calculate the emissions "InLine" and to use the lightning parameters file that you created with the LTNG_2D_DATA preprocessor.
- ❑ Generate a diagnostic file of lightning NOx emissions.

Exercise 8-1:
List the variables that you changed or checked, and their values.

Answer 8-1:

The rest of the settings for the simulation will stay the same as those used in Lab 5. Confirm that the script is configured for the RoMANS12 grid and for a 12-hour long simulation that starts at 0Z on July 7, 2006. Save the CCTM run script, exit, and run it. Remember to capture the output from the script to a log file.

8.3 Examining the CCTM LNOx simulation output files with VERDI

2. In this step, you will load the 12-km CCTM CONC files from the base simulation (Lab 5) and the LNOx simulation into VERDI and compare the results.

   Go to the CCTM output directory, start VERDI, and import the base and LNOx 12-km CONC files that you created during this training. Create tile plots of O_3 using the "Fast Tile Plot" VERDI option. Create difference plots of both O_3 and NO to see how the CMAQ LNOx module impacts the simulation results.

This concludes the lightning NOx problem solving exercise. You are now ready to move on to the ninth and final exercise in the CMAQ training course: "Problem Solving: Regional Modeling Case Study." This final exercise integrates all of the information and techniques you learned as you worked through Exercises 1 through 7.

(Page left intentionally blank)

9 CMAQ Training — Problem Solving: Regional Modeling Case Study

NOTE: This exercise assumes that you have completed the previous CMAQ exercises and thus have a basic knowledge of how to build and run the CMAQ modules. For complete details on how to use the **bldmake** scripts and invoke the run scripts, refer back to the exercises for the specific CMAQ modules.

9.1 Introduction

During this problem-solving exercise, you will:

- Learn how to configure and run CMAQ based on a statement of a problem

NOTE: To proceed with this exercise, you must have sourced the config.cmaq file.

Please work through the problem presented below and compare your results with the answers in the $ANSWERS/sesarm directory.

9.2 Regional Modeling for the Southeast U.S.

As a modeler for a state air quality agency in the Southeast U.S. you are tasked with studying the effects of interstate transport on ozone non-attainment areas in your state. Your agency recently received an archive of CMAQ input data for a 12-km modeling domain covering the region. The archive includes annual 2007 WRF output files, annual SMOKE emissions files, and a description of the domain that you are to model. Your modeling approach entails evaluating the base model results and then performing a series of emissions sensitivities to evaluate the impacts of different source regions on receptors in your state. For the first step in this process, you must perform a base simulation with the data that you just received.

One of the analysts in your organization installed the data archive in the following directory: $M3DATA/raw/sesarm. She also set up the directory structure for your CMAQ input and output files. Configure the CMAQ components to run this simulation. Specifics about the horizontal and vertical domains, as well as the CCTM set up, are included below.

Set up the simulation to start at 00Z on July 22, 2007 (2007203) and run for 12 hours. The modeling configuration will simulate gas-phase chemistry using the CB05cl photochemical mechanism with the EBI solver and aerosols with the 5th-generation CMAQ aerosol model. To optimize the CCTM for WRF meteorology, you will need to modify the CCTM configuration and rebuild the model. You will be able to use the executables that you created for ICON and BCON in the previous exercises in this course (i.e., you do not need to recompile these two programs). This is not a nested simulation, so you will be using the CB05 IC and BC profiles that are packaged with CMAQv5.0.

The emissions files for this simulation were prepared to compute vertical distributions inline in the CCTM. You will need to configure the CCTM run script to read in the emissions files and calculate plume rise for the point sources.

Table 3 lists the full configuration options to use for the CCTM.

Set the configuration variable (CFG) in run scripts to *sesarm*.

Upon completion of the simulation, look at the results in VERDI and compare them to the answers for this exercise (located in $ANSWERS/sesarm).

Table 1. Horizontal domain specifications

Parameter	Value
Col	168
Row	177
Layers	22
True Lat 1	33
True Lat 2	45
Standard Lon	-97
X-center	-97
Y-center	40
X-Origin	108000
Y-Origin	-1620000
X-Cell	12000
Y-Cell	12000

Table 2. WRF domain specifications

Parameter	Value
Col	250
Row	250
Layers	34
True Lat 1	33
True Lat 2	45
Standard Lon	-97
X-center	-86.89108
Y-center	37.76529
X-Origin	-600000
Y-Origin	-1680000
X-Cell	12000
Y-Cell	12000

Table 3. Vertical domain specifications

Layer	sigma
1	1.00
2	0.9974
3	0.994
4	0.99
5	0.9854
6	0.9796
7	0.9723
8	0.9635
9	0.9528
10	0.9401
11	0.9252
12	0.9079
13	0.8882
14	0.8659
15	0.841
16	0.7828
17	0.7133
18	0.6323
19	0.5406
20	0.3895
21	0.2378
22	0.1056
23	0.0000

Table 4. CCTM configuration specifications

Configuration Option	Setting
Model Driver	ctm_wrf
Model Concentration Coupling	gencoor_wrf
Model Initialization	init_yamo
Horizontal Advection	hyamo
Vertical Advection	vwrf
Horizontal Diffusion	multiscale
Vertical Diffusion	ACM2
Deposition Velocities	m3dry
Species Configuration	namelist
Photolysis	inline
Chemistry Solver	EBI (CB05cl)
Aerosol Model	aero5
Clouds	acm_ae5
Process Analysis	none
Windblown Dust	No
Lightning NOx	No
Recalculate KzMin?	Yes
Inline Deposition Velocities	Yes
Landuse-based Deposition Velocities	No
Bi-directional NH3 Flux	No
Bi-directional Mercury Flux	No
Surface HONO Interaction	Yes
Inline Biogenic Emissions	No
Inline Plume Rise	Yes

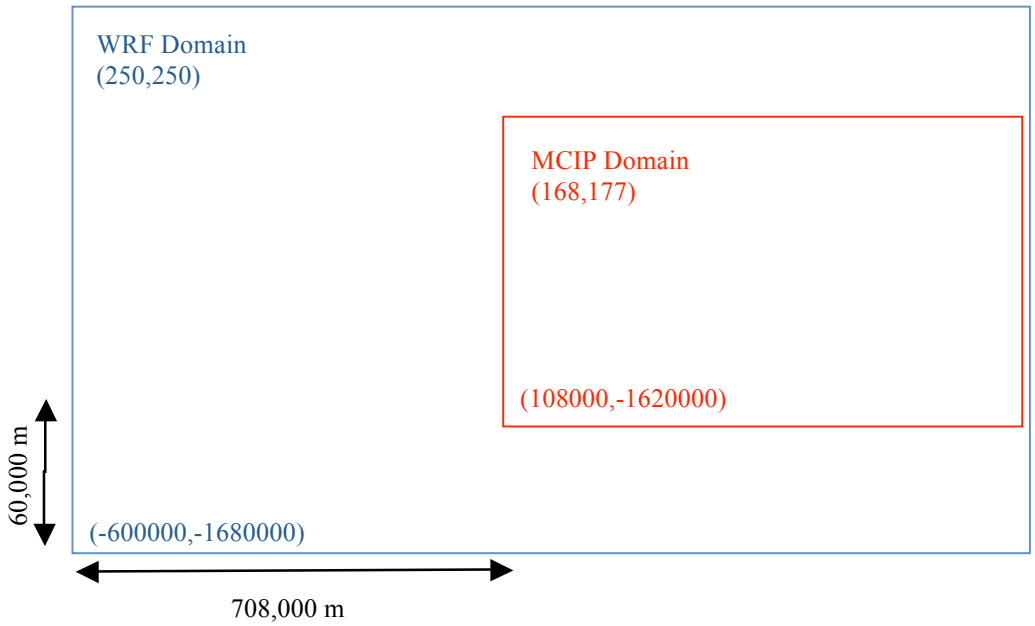
CMAQ version 5 Training Answer Sheet

Exercise	Answer
1-1	lnox, MM5, phot, sesarm
1-2	$M3DATA/raw/MM5
1-3	/home/training/CMAQv5.0/lib/x86_64/ifort/build
1-4	libsef90_noop.a
2-1	(1) July 22, 2001 00:00 UTC (2) InTerDir, IfTer, and InTer File (3) CTMLAYS = -1.0
2-2	(50,50,16)
2-3	(1) 25 (2) Pa (Pascals)
2-4	17.415 m
2-5	METCRO3D, METBDY3D
3-1	• Variable: ModType Value: profile • Variable: ModMech Value: cb05 • Variable: Mechanism Value: cb05cl_ae5_aq • Variable: APPL Value: profile
3-2	BLD_profile
3-3	(1) $M3DATA/mcip/ROMANS12_50X50/2006188/GRIDDESC (2) ROMANS12_50X50 (3) $M3DATA/mcip/ROMANS12_50X50/2006188/METCRO3D_training
3-4	profile
3-5	$M3HOME/scripts/bcon/BLD_profile/bc_profile_CB05.dat
3-6	No North: 0.08 ppmV South: 0.07 ppmV East: 0.08 ppmV West: 0.08 ppmV
3-7	ppmV
3-8	-1,056,000 meters
3-9	30 ppbV (0.030 ppmV)
3-10	32 ppbV (0.032 ppmV)
4-1	ROMANS12_50X50
4-2	profile
4-3	$M3HOME/scripts/bcon/BLD_profile/ic_profile_CB05.dat
4-4	boundary switch
4-5	
4-6	
4-7	80
4-8	0.601
4-9	80 ppbV (0.08 ppmV)
5-1	• Variable: ModDriver Value: ctm_yamo • Variable: ModCpl Value: gencoor • Variable: ModHadv Value: hyamo • Variable: ModVadv Value: vyamo

	• Variable: ModHdiff Value: multiscale • Variable: ModVdiff Value: acm2 • Variable: ModDepv Value: m3dry • Varaible: ModCgrds Value: cgrid_spcs_nml • Variable: ModPhot Value: phot_inline • Variable: ModPing Value: ping_noop • Variable: ModChem Value: ebi_cb05cl • Variable: ModAero Value: aero5 • Variable: ModCloud Value: cloud_acm_ae5 • Variable: Mechanism Value: cb05cl_ae5_aq • Variable: PAOpt Value: pa_noop
5-2	2001100
5-3	NSTEPS = 480000
5-4	"3 5"
5-5	• Variable: NPCOL_NPROW Value: "1 1" • Variable: NPROCS Value: 1 • Variable: STDATE Value: 2006188 • Variable: NSTEPS Value: 120000 • Variable: GRID_NAME Value: ROMANS12_50X50 • Variable: AVG_CONC_SPCS Value: "O3 NO CO NO2 ASO4I ASO4J NH3" • Variable: CTM_WB_DUST Value: N • Variable: CTM_LTNG_NO Value: Y • Variable: KZMIN Value: "1 1" • Variable: CTM_ILDEPV Value: Y • Variable: CTM_MOSAIC Value: N • Variable: CTM_ABFLUX Value: N • Variable: CTM_HGBIDI Value: N • Variable: CTM_SFC_HONO Value: Y • Variable: CTM_BIOGEMIS Value: N • Variable: CTM_PT3DEMIS Value: N
5-6	egts_l.2006188.1.ROMANS12_50X50.Pre06b.cmaq.cb5p25.1lay.ncf
5-7	Start Date = 2006188 Start Time = 0
6-1	(50,50,16)
6-2	• Variable: APPL Value: m3conc • Variable: ModInpt Value: m3conc • Variable: ModMech Value: cb05 • Variable: Mechanism Value: cb05cl_ae5_aq
6-3	CFG, GRID_NAME, GRIDDESC, EMISfile, OCEANfile, GC_ICfile, GC_BCfile, METpath
7-1	STDATE, STTIME, GRID_NAME, EMISfile, GC_ICpath, GC_ICfile, GC_BCfile, METpath
7-2	Changed the number of time steps (NSTEPS) to 1200000
7-3	Implement a "foreach" loop that cycles through the five Julian dates. Set the number of time steps (NSTEPS) to 240000

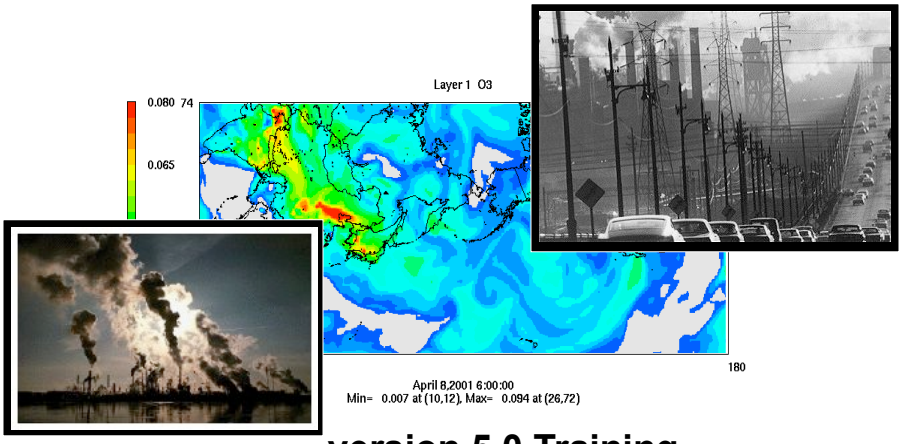
8-1	• Variable: CFG Value: romans12_lnox • Variable: CTM_LTNG_NO Value: Y • Variable: LTNGNO Value: InLine • Variable: LTNGPARAM Value: Y • Variable: LTNGPARM_FILE Value: $M3DATA/lnox/LTNG_RATIO.2006.07.ioapi • Variable: LTNGDIAG Value: Y
Lab 9	• With all of the scripts/files that you create for this exercise, it is recommended that you copy the original to a new file and edit that new file. Reference or use the new files during this exercise. The new files will include build scripts and run scripts. Use a unique label to identify this case. Instead of labeling the executables and runs with "training", label them with "sesarm". • You can use the MCIP4 executable that you created in Lab 2 for this problem solving exercise. • Configure and run the MCIP4 run script for this new case to create the meteorology input for this simulation. Change the GridName in the script and the input and output directory paths to $DataPath/raw/sesarm/wrf and $DataPath/sesarm/mcip/$GridName, respectively. Change the name of the input meteorology files to the names of the wrfout files in the wrf input data directory. Start the simulation on July 22, 2007, at 00Z and end the simulation on July 23, 2007, at 00Z. • Edit the sigma layer definitions (CTMLAYS) in the MCIP run script to be consistent with the configuration in Table 3 of Chapter 9 in this manual. • The horizontal WRF domain is much larger than the domain that you will be simulating. You will need to use the MCIP subdomain option. The MCIP4 script contains documentation stating that in order to use the subdomain option, you must set the value of BTRIM = -1. Thus, set BTRIM to this value. Note that the WRF grid starts at (1,1) and not (0,0); this is important when you calculate the offset from the origin for your MCIP domain. The X-origin and Y-origin of the WRF domain are -600,000 m and -1,680,000 m from the center of the domain, respectively. The setting for Y0 trims the N-S or Y-axis of the parent domain, while the setting for X0 trims the E-W or X-axis of the parent domain. At 12-km resolution, to get the X-origin to be reset to 108,000, you will need to trim 59 cells from the WRF domain, or set X0 = 60 (equivalent to the starting cell in the WRF domain (1) plus the number of offset cells). Similarly, to reset the Y-origin to -1,620,000, you will need to trim 5 cells from the MM5 domain or set Y0 = 6 (equivalent to the starting cell in the WRF domain plus the number of offset cells). See the diagram below for an illustration of this calculation. • Set the WRF Lambert Conformal Reference Latitude (WRF_LC_REF_LAT) to 40.0 to force the correct centroid latitude in the MCIP output. If you do not set this variable, the MCIP output domain will not match the domain of the emissions files that were prepared in advance of this exercise. • Run BCON with the default CB05 profile data and write the new BCON output files to $M3DATA/sesarm/bcon. Remember to change the name of the grid to the name that you defined in the MCIP run. You also need to point the script to the new MCIP output file that you created. • Run ICON with the default CB05 profile data and write the new ICON output files to $M3DATA/sesarm/icon. Remember to change the name of the grid to the name that you defined in the MCIP run. You also need to point the script

	to the new MCIP output file that you created. • Configure and run the build script to create a new CCTM executable that will use the configuration listed in Table 4. • Configure the CCTM run script to use the new grid and meteorology data that you created with MCIP for this simulation. • Configure the CCTM run script to process plume rise "inline" by setting CTM_PT3DEMIS = Y. Find the section of the CCTM run script where the input emissions files are defined. Processing emissions inline requires three types of emissions files to be set in the CCTM run script: (1) layer 1 (or non-elevated) sources, (2) stack groups files for each elevated source sector, and (3) emissions data for each elevated source sector. The environment variable used to set the type (1) sources in the CCTM run script is EMISfile. Note that there can only be one non-elevated emissions file input to the CCTM. The type (2) and (3) emissions files are set with the STK_GRPS_## and STK_EMIS_## environment variables, respectively. Note that there can be multiple elevated emissions files input to the CCTM, where the "##" in the environment variables above correspond to an ordered number of files. For this problem solving exercise there are 8 elevated source sectors. Set the environment variable NPTGRPS to "8" to specify the number of elevated sectors that will require plume rise calculations. You must then set the STK_GRPS and STK_EMIS variables accordingly for each of the 8 sectors. For example, you will see that there are a pair of emissions files for a sector tagged "sesarm_ptf" in the $M3DATA/raw/sesarm/emis directory. Set the CCTM run script variable STK_GRPS_01 to point to the stack_groups file and set the variable STK_EMIS_01 to point to the inln_mol file for this sector. In setting these script variables, the order of the sectors is not important. It is necessary however to ensure that the STK_GRPS and STK_EMIS sectors correspond with each other. If you use the variable STK_GRPS_02 to point to the input file for the emissions sector tagged sesarm_ptcem, you must also set the STK_EMIS_02 variable to point to the corresponding inln_mol file for that sector. • Configure the CCTM run script to use the correct ocean file (OCEAN_1) for the 12-km southeast U.S. domain. • Update the name of the IC and BC files in the CCTM run script to use the files that you created for this simulation.



Y0	=	[(WRF Y-orig – MCIP Y-orig)/12,000]+ 1
	=	[(-1,680,000 - -1,620,000)/12,000] + 1
	=	6
X0	=	[(WRF X-orig – MCIP X-orig)/12,000]+ 1
	=	[(-600,000 – 108,000)/12,000] + 1
	=	60

Community Multiscale Air Quality (CMAQ) Modeling System



version 5.0 Training

http://www.cmaq-model.org

1

Training Overview

Classroom 1

- AQM overview
- CMAQ basics
- CVS, I/O API, netCDF
- CMAQ scripts
- CMAQ programs and options

Classroom 2

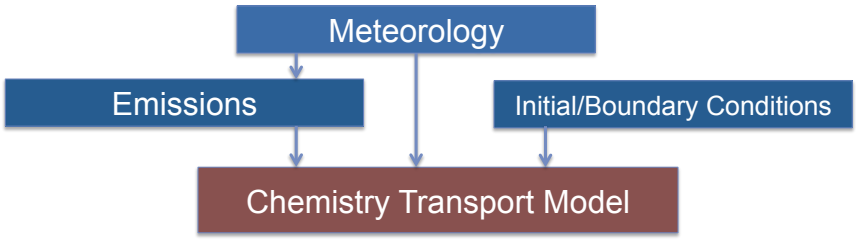
- CMAQ concepts
- Evaluation and QA

Laboratory

- **Overview lab**
- **MCIP lab**
- **ICON lab**
- **BCON lab**
- **CCTM lab**
- **Nesting lab**
- **Multi-day lab**
- **LNOX lab**
- **Case Study**

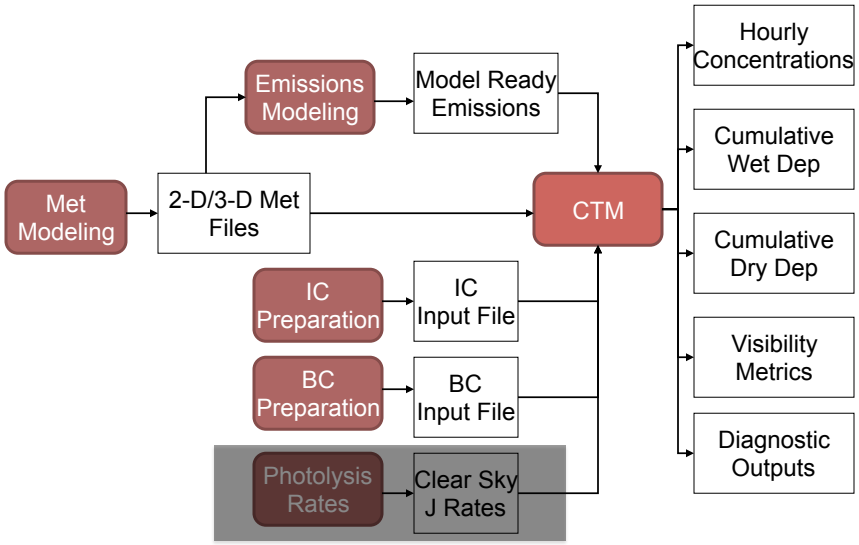
2

Chemistry-Transport Modeling



- Inputs to CTMs include
  - Meteorology – e.g. winds, temperature, precipitation
  - Emissions – pollutant fluxes from industrial, mobile, natural, and other sources
  - IC/BC – fluxes at the model boundaries
- Outputs from CTMs include
  - Hourly concentrations of gases and particles
  - Hourly wet and dry deposition totals
- CTMs are open-source, Linux software designed to run on large computing clusters

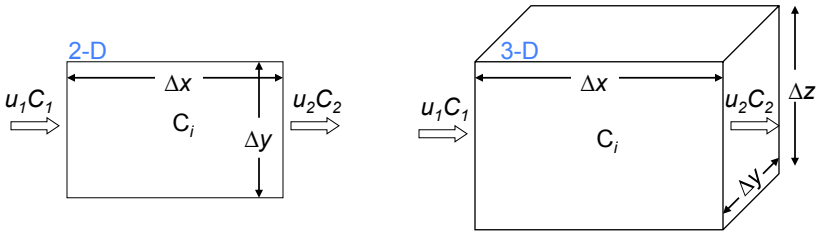
CTM Process Schematic



4

Conceptual approach to CTMs

- Extend the 2-D box model to three dimensions



u = wind vector

C_i = concentration of species i

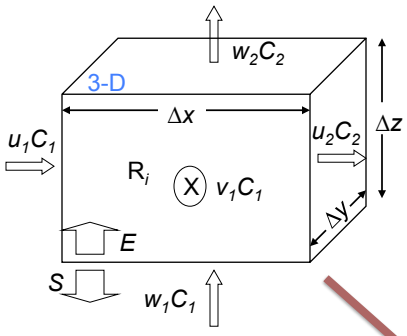
Basic Continuity Equation (flux in 1 direction):

$\Delta C\ \Delta x\ \Delta y\ \Delta z = u_1C_1\Delta y\ \Delta z\ \Delta t - u_2C_2\ \Delta y\ \Delta z\ \Delta t$

Divide by Δt and volume: $\partial C/\partial t = -\partial(uC)/\partial x$

5

Expanded Continuity Equation



Expanded Continuity Equation Derivation:

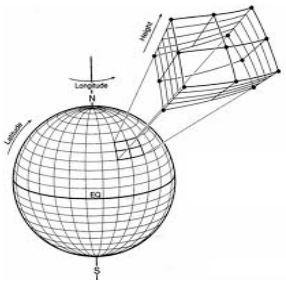
Expand to flux three dimensions:

$\partial C/\partial t = -\partial(uC)/\partial x - \partial(vC)/\partial y - \partial(wC)/\partial z$

$= -\nabla\cdot(vC)$ (flux divergence form)

Add additional production and loss terms:

$\partial C/\partial t + \nabla\cdot(vC) = D\nabla^2 C + R + E - S$



u,v,w = wind vectors

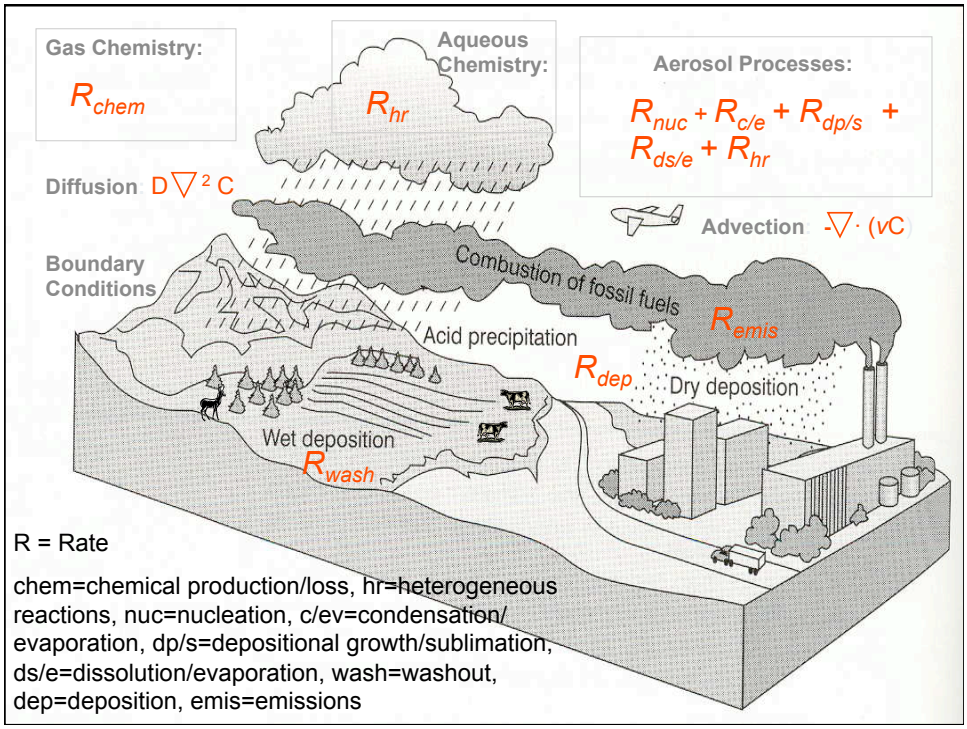
E = emissions

S = loss processes

R_i = Chemical formation of species I

D = Molecular diffusion coefficient

6



Full Continuity Equations

- Gas Continuity Equation

$$\partial C/\partial t + \nabla \cdot (vC) = D\nabla^2 C + R_{chemg} + R_{emisg} + R_{depg} + R_{washg} + R_{nucg} + R_{c/eg} + R_{dp/sg} + R_{ds/eg} + R_{hrg}$$

- Particle Continuity Equation (number)

$$\partial n/\partial t + \nabla \cdot (vn) = D\nabla^2 n + R_{emisn} + R_{depn} + R_{sedn} + R_{nucn} + R_{washn} + R_{coagn}$$

- Particle Continuity Equation (volume concentration)

$$\partial V/\partial t + \nabla \cdot (vV) = D\nabla^2 V + R_{emisv} + R_{depv} + R_{sedv} + R_{nucv} + R_{washv} + R_{coagv} + R_{c/ev} + R_{dp/sv} + R_{ds/ev} + R_{egv} + R_{qgv} + R_{hrv}$$

8

Atmospheric Diffusion Equation (ADE)

- ADE* represents mass balance in which emissions, transport, diffusion, chemical reaction and removal processes are represented by:

$$\frac{\partial C_i}{\partial t} + \frac{\partial (uC_i)}{\partial x} + \frac{\partial (vC_i)}{\partial y} + \frac{\partial (wC_i)}{\partial z} =$$

$$\frac{\partial}{\partial x}\left(K_H \frac{\partial C_i}{\partial x}\right) + \frac{\partial}{\partial y}\left(K_H \frac{\partial C_i}{\partial y}\right) + \frac{\partial}{\partial z}\left(K_V \frac{\partial C_i}{\partial z}\right) + R_i + S + L_i$$

* **Several assumptions included above**

9

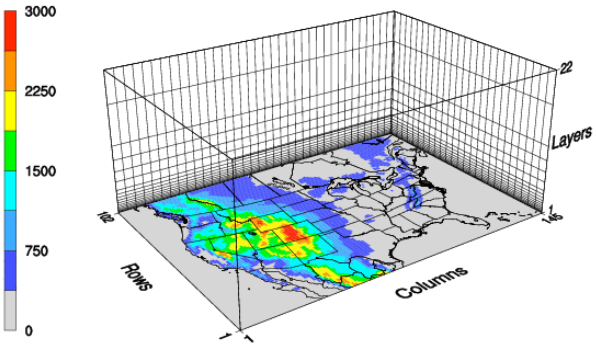
ADE (..Contd.)

where,

- i = chemical species, i (where $i = 1, 2, \ldots N$)
- C_i = pollutant concentration of species, i
- u, v, w = horizontal and vertical wind speed components = $f(x, y, z, t)$
- K_H, K_V = horizontal and vertical turbulent diffusion coefficients = $f(x, y, z, t)$
- R_i = rate of formation of i by chemical reactions = $f(C_1, C_2, .., C_i, .. C_N)$
- S = emission rate of all precursors
- L_i = net rate of removal of pollutant i by surface uptake processes = $f(x, y, z, t)$

10

3-D Box Representation



11

Numerical Solution of Partial Differential Equations (PDEs)

- E.g. Model Application with 100 Columns X 100 Rows X 25 Layers = 250,000 grid-cells
- CB-IV Chemical Mechanism with 79 species and 94 reactions
- Number of differential equations to be solved simultaneously per model output time step = 250,000 X 79 = 19,750,000
- For 1 day, # ODEs = 19.75 X 24 = 474 million
- For 1 month, # ODEs = 19.75 X 24 X 30 = 14.22 billion
- For 1 year, # ODEs = 19.75 X 24 X 365 = 173 billion

12

CTM Coordinate Systems

- Convert all motion equations from Cartesian to spherical coordinates

$$dx = (R_e \cos\varphi) d\lambda_e \qquad dy = R_e d\varphi$$

- Horizontal grids typically on the order of 1 to 36 km
- Recent applications extending to 500m and 108 km
- Lambert conformal, polar stereographic, and Mercator are the most common modeling projections

13

CTM Coordinate Systems

- Vertical grids extend from the surface to 10 km
- Altitude coordinate: layers are defined as surfaces of constant height with variable pressure
- Pressure coordinate: layers are defined as surfaces of constant pressure with variable height
- Sigma-pressure coordinate: layers defined as surfaces of constant σ, where

$$\sigma = \frac{p_a - p_{a,top}}{p_{a,\,surf} - p_{a,top}} \qquad (0,1)$$

14

CMAQ Basics

- Definitions & Acronyms
- CMAQ Basics
- CMAQ Modules
- CMAQ Scripts

15

Definitions & Acronyms

- SMOKE: Sparse Matrix Operator Kernel Emission processing system
- CCTM: CMAQ Chemical Transport Model
- BCON: Boundary Conditions processor
- ICON: Initial Conditions processor
- JPROC: Photolysis rates processor
- MCIP: Meteorology Chemistry Interface Processor
- CB05cl: Carbon Bond version 2005 chemical mechanism with chlorine chemistry

16

CMAQ Basics

- CMAQ designed under the community modeling paradigm:
  - Modular
  - Multiscale
  - One-atmosphere
  - Portable
  - Accessible
- Can model from urban (few km) to regional (hundreds of kilometers) to inter-continental (thousands of kilometers) scales of transport
- Tropospheric O_3, acid deposition, visibility, particulate matter, toxics, mercury
- Requires inputs from emissions and meteorology models, initial and boundary conditions
- CMAQ = Chemical Transport Model + preprocessors

17

CMAQ Modules

- ICON
- BCON
- JPROC
- LNOx
- MCIP
- CCTM

18

CMAQ Modules – ICON (1)

- ICON is the CMAQ initial conditions preprocessor
- Generates IC's from:
  - Text-based vertical profiles
  - netCDF CTM output
- Grid windowing
- Default and custom mechanism conversion
- Creates netCDF IC output file
- Example text-based input file:

19

CMAQ Modules – ICON (2)

Optional initial condition:

The vertical coordinate of the model to generate these i.c. is the terrain-following sigma coordinate. The number of sigma layers and defined sigma levels are listed below.

```
 6    55   1.00  0.98  0.93  0.84  0.60  0.30  0.00
 1988180  00
"SO2     " 0.300E-03  0.200E-03  0.100E-03  0.100E-03  0.200E-04  0.100E-04
"SULF    " 1.000E-30  1.000E-30  1.000E-30  1.000E-30  1.000E-30  1.000E-30
"NO2     " 0.167E-03  0.167E-03  0.084E-03  0.000E+00  0.000E+00  0.000E+00
"NO      " 0.083E-03  0.083E-03  0.042E-03  0.000E+00  0.000E+00  0.000E+00
"O3      " 0.350E-01  0.350E-01  0.400E-01  0.500E-01  0.600E-01  0.700E-01
```

20

CMAQ Modules – BCON (1)

- BCON is the CMAQ boundary conditions preprocessor
- Generates BC's from:
  - Text-based vertical profiles
  - netCDF CTM output
- Static and time-dependent BC's
- Grid windowing capabilities
- Default and custom mechanism conversion
- Creates netCDF BC output file
- Example text-based input file:

21

CMAQ Modules – BCON (2)

Optional boundary condition:

The vertical coordinate of the model to generate these b.c. is the terrain-following sigma coordinate. The number of sigma layers and defined sigma levels are listed below.

```
   6     55   1.00  0.98  0.93  0.84  0.60  0.30  0.00
1988180  00
North
"SO2     " 0.300E-03  0.200E-03  0.100E-03  0.100E-03  0.200E-04  0.100E-04
"SULF    " 1.000E-30  1.000E-30  1.000E-30  1.000E-30  1.000E-30  1.000E-30
"NO2     " 0.167E-03  0.167E-03  0.084E-03  0.000E+00  0.000E+00  0.000E+00
"NO      " 0.083E-03  0.083E-03  0.042E-03  0.000E+00  0.000E+00  0.000E+00
"O3      " 0.350E-01  0.350E-01  0.400E-01  0.500E-01  0.600E-01  0.700E-01
```

22

CMAQ Modules – JPROC (1)

- JPROC is the CMAQ photolysis rate preprocessor
- Generates daily photolysis lookup tables from:
  - Tabulated molecular cross-section/quantum yield data as a function of wavelength
- Creates ASCII PHOT output files
- Example input file format:

ACROLEIN

FAC = 1.0

250.0 1.803E-21 1.000
251.0 1.805E-21 1.000
252.0 2.052E-21 1.000 …

23

CMAQ Modules – JPROC (2)

```
1999193  (yyyyddd) Julian Date for the file
   7  LEVELS (m)
     0.0  1000.0  2000.0  3000.0  4000.0  5000.0 10000.0
   6  LATITUDES (deg)
   10.0  20.0  30.0  40.0  50.0  60.0
   9  HOUR ANGLES (from noon)
    0.0   1.0   2.0   3.0   4.0   5.0   6.0   7.0   8.0
   6  PHOTOLYTIC REACTIONS
    'NO2_CBIV88      ',    1.0
   1   1   1
5.1139301E-01  5.0172305E-01  4.6737903E-01  4.0507138E-01
3.0828261E-01 1.7023879E-01  3.4761846E-02  0.0000000E+00
0.0000000E+00
```

24

CMAQ Modules – LNOx (1)

- LNOx is the lightning NOx emissions preprocessor
- Three options for simulating the air quality impacts of lightning NO emissions in CMAQ:
  1. Input file of LNOx emissions that were calculated off-line
  2. Compute LNOx emissions in-line during a simulation directly from lightning flash counts estimated from the convective precipitation field in the input meteorology
  3. Compute LNOx emissions in-line using a combination of observed flash counts and simulated convective precipitation
- A preprocessor is used to prepare a lightning NOx parameters input file for simulations that use option 3.

25

CMAQ Modules – MCIP

- MCIP is the CMAQ meteorology preprocessor
- Generates gridded 3-d and 2-d netCDF met inputs
- Source data are MM5 binary MMOUT files or WRF netCDF files
- MCIP3 optionally reads MM5 TERRAIN files for calculation of land-use based K_v in CMAQ
- MCIP outputs can also be used with SMOKE for emissions modeling

26

CMAQ Modules - CCTM

- The CCTM is an Eulerian air quality model
- Configuration options for CCTM science modules determined at compilation
- Dynamic horizontal grid allocation and vertical layer structure determined at execution
- Basic outputs include hourly concentrations for gaseous and aerosol pollutants, wet and dry deposition fluxes, and visibility metrics
- Supplementary outputs include inline emissions diagnostic files, land-use deposition files (MOSAIC), and process analysis outputs

27

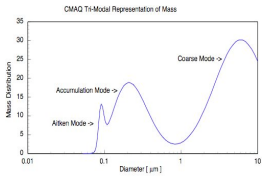
CCTM Science Process Modules

- DRIVER → controls model data flows
- HADV → horizontal advection
- VADV → vertical advection
- ADJCON → adjusts mixing ratios of pollutants
- HDIFF → horizontal diffusion
- VDIFF → vertical diffusion and deposition
- CHEM → gas-phase chemistry
- CLOUD → aqueous-phase chemistry, cloud-mixing
- AERO → aerosol dynamics and size distributions
- PHOT → computes photolysis rates
- AERO-DEPV → computes particle size-dependent dry deposition velocities
- UTIL → collection of utility subroutines

28

PM Treatment in CMAQ

- Particle size distribution is represented as the superposition of 3 lognormal sub-distributions (modes)
  - Aitken mode (up to ~ 0.1 microns) (typically for fresh particles)
  - Accumulation mode (0.1 - 2.5 microns) (aged particles)
  - Coarse mode (2.5 – 10 microns)
  - PM_{10} is the sum of all three modes
- For each mode, model predicts
  - Various chemical components of $PM_{2.5}$
    - Primary: Organics, Elemental Carbon, Crustal
    - Secondary: Sulfate, Nitrate, Ammonium, Organic Aerosol (Anthropogenic, Biogenic)
- Additional outputs
  - particle number, total surface area and total mass



Ref: Binkowski et al, JGR, 2003

29

CMAQ Scripts

- Build scripts and Makefiles
- Run scripts

30

CMAQ Build Scripts (1)

- The CMAQ system is entirely composed of Fortran programs
- Bldmake is used for configuration management and compilation in the CMAQ system
- Bldmake functions:
  - checks out working copies of the CMAQ source code from a repository
  - generates an Makefile
  - executes the Makefile to create an executable
- Each CMAQ module has a build script associated with it that calls Bldmake (MCIP uses a Makefile)

31

CMAQ Build Scripts (2)

- Operating system dependencies are handled by the Bldmake scripts (i.e, location and type of Fortran compiler, compilation flags, location of source code on system)
- The build scripts contain configuration information that is used to check copies of the CMAQ source code out of a Concurrent Version System (CVS) code repository
- The CMAQ build scripts provide the option of either generating executables directly or creating Makefiles

32

Makefiles

- Alternative to build scripts for compiling executables
- Require that the source code is online, i.e. no CVS options built into the script
- Operating system dependencies
- MCIP is the only CMAQ module that is currently distributed with only a Makefile option for compilation

33

Run Scripts

- User interface for CMAQ
- CMAQ run scripts are C-shell files that set necessary environment variables, directory and file locations, and call executables
- Settings defined in the CMAQ run scripts include:
  - Input and output file nomenclature
  - Horizontal grid definitions
  - 3-d meteorology file for setting vertical layer structure
  - Start date, start time, time step, and run duration
  - Logfile options
  - Debug options
- The CCTM run script also includes several scientific configuration options

34

CMAQ Libraries and Utilities

- I/O API
- netCDF
- mpich
- Stenex
- Pario
- CVS
- VERDI

35

CMAQ Libraries and Utilities

- Library files provide an easy way for programs to share commonly used subroutines; three libraries used by CMAQ
- I/O API - an easy-to-learn, easy-to-use programming library for data storage and access, available for both Fortran and C
- netCDF - an interface for array-oriented data access and a library that provides an implementation of the interface
- mpich – message passing interface for parallel computing

36

CMAQ Libraries and Utilities

- Stenex - the Stencil Exchange library is required for parallel processing
- Pario – a set of library routines that control CMAQ input/ output for multiprocessor applications

37

I/O API Utilities

- The Input/Output Applications Programming Interface contains an extensive set of utility routines for manipulating dates and times, performing coordinate conversions, storing and recalling grid definitions, sparse matrix arithmetic, etc., as well as a set of data-manipulation and statistical analysis programs
- Command line programs that are easy to script for automating analysis and post processing
- http://www.baronams.com/products/ioapi/
- Examples of I/O API utilities

38

I/O API Utilities

- m3diff: for computing statistics for pairs of variables and for applying various comparison ("differencing") operations to those variables in a pair of files
- m3edhdr: edit header attributes/file descriptive parameters
- m3merge: merges selected variables from a set of input files for a specified time period, and writes them to a single output file, with optional variable-renaming in the process
- vertot: compute vertical-column totals of variables in a file

39

netCDF Library

- The Network Common Data Form is an interface to a library of data access functions for storing and retrieving data in the form of arrays
- An abstraction that supports a view of data as a collection of self-describing, network-transparent objects that can be accessed through a simple interface
- By using direct file access, netCDF achieves the goal of supporting efficient access to small subsets of large datasets
- www.unidata.ucar.edu/packages/netcdf/

40

CVS

- Concurrent Versions System is a source code version control system that operates on a hierarchical directory structure to manage code releases and control the concurrent editing of files among multiple authors
- Version control systems are primarily used in development environments where multiple programmers may be accessing source code simultaneously
- In CMAQ, CVS is used simply as a code repository for protecting the distributed files from direct access

41

CVS

- Instead of directly accessing the source code in the CMAQ distribution, you use CVS to access copies of the code
- If you manipulate these copies when they are checked out of CVS, you can check them back into the repository, where CVS will record information of the nature and timing of the file revisions
- http://www.cvshome.org
- Examples of CVS commands

42

CVS

- cvs checkout foo – checks out your own working copy of the source for ‘foo’ into the current directory
- cvs commit foo – checks your working copy of the source for ‘foo’ into the CVS archive
- cvs –n update foo – lists the files that have changed in the source for ‘foo’ since checking them out of the CVS archive
- cvs add bar – adds a node to the CVS archive for the source for ‘bar’

43

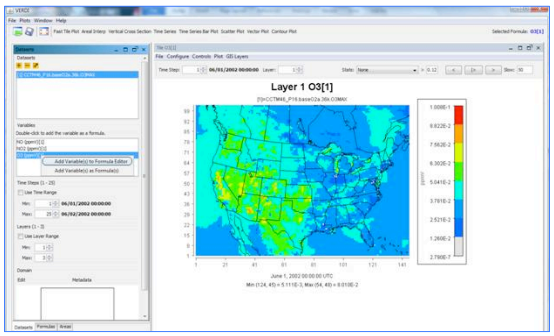
VERDI

- Visualization Environment for Rich Data Interpretation is Java graphics tool for I/O API-netCDF formatted data
- VERDI was developed as a replacement for PAVE
- Easily scriptable to automate plot generation
- Creates 2-d tile plots, time series, 3-d mesh plots, bar charts, scatter plots, and integrates observations into graphics
- http://www.verdi-tool.org

44

VERDI

- Exports images and data in GIF, MPEG, Animated GIF, ascii text, and postscript
- VERDI currently can be used to create 2-d tile plots, vertical cross sections, scatter plots, line and bar timeseries, contour plots, vector plots, and vector-tile plots
- Intuitive GUI



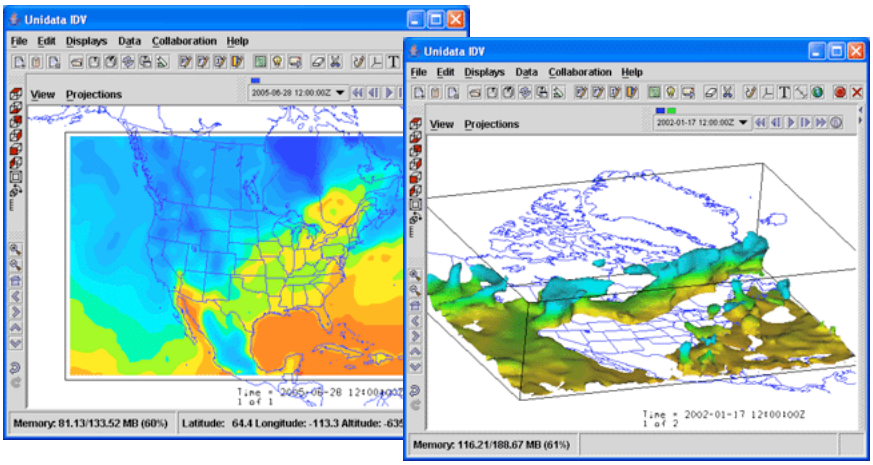
45

IDV

- Integrated Data Viewer is a Java graphics tool that supports multiple data formats
- IDV is a 3-d visualization tool
- Scriptable to automate plot generation
- In addition to standard output formats (gif, mpeg, etc), also can output kmz files.
- http://www.unidata.ucar.edu/software/idv/

46

IDV



47

CMAQ Programs and Options

- Compiler and execution options for:
  - ICON
  - BCON
  - JPROC
  - MCIP
  - LNOx
  - CCTM

48

ICON

- Compilation Options
  - Executable nomenclature
  - m3bld options – e.g. compile_all, verbose, Makefile
  - Input data type
  - Debug mode?
  - Chemical mechanism
- Execution Options
  - Episode nomenclature
  - Horizontal grid resolution and grid definition
  - Vertical layer structure
  - Input/output directories and file names
  - For nested runs: start date and time of parent data
  - Executable name and location

49

BCON

- Compilation Options
  - Executable nomenclature
  - m3bld options – e.g. compile_all, verbose, Makefile
  - Input data type
  - Debug mode?
  - Chemical mechanism
- Execution Options
  - Episode nomenclature
  - Horizontal grid resolution and grid definition
  - Vertical layer structure
  - Input/output directories and file names
  - For nested runs: start date and time of parent data, run length
  - Executable name and location

50

JPROC

- Compilation Options
  - Executable nomenclature
  - m3bld options – e.g. compile_all, verbose, Makefile
  - Chemical mechanism
  - Debug mode?
- Execution Options
  - Episode nomenclature
  - Time parameters: start date and end date
  - Input/output directories and file names
  - Executable location

51

MCIP – Compiler Options

- Makefile configuration, no build script
- Executable nomenclature
- Platform-specific compiler options

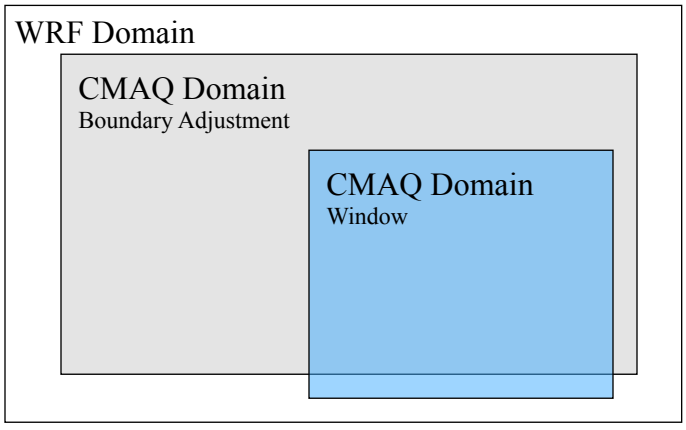
52

MCIP – Execution Options

- Episode nomenclature
- Coordinate and grid names
- Input/output directories and file names
- Control options
  - Compute and output potential vorticity
  - Output vertical velocity
  - Output u- and v- component winds on a C-grid
  - Replace model-derived input with GOES observed clouds
- Episode start and end dates/times
- Sigma values of vertical layer boundaries
- Meteorology boundary adjustment
- Horizontal domain windowing options
- Executable name and location

53

MCIP – Window vs Boundary Trim



54

CCTM – Compiler Options

- Executable nomenclature
- m3bld options – e.g. compile_all, verbose, Makefile
- Science modules
  - Horizontal and vertical advection module
  - Horizontal and vertical diffusion module
  - Deposition velocity module
  - Chemistry input parameters module
  - Gas-phase chemistry solver
  - Aerosol chemistry module
  - Cloud module
  - Chemical mechanism
  - Process analysis configuration
- Debug mode?

55

CCTM – Execution Options(1)

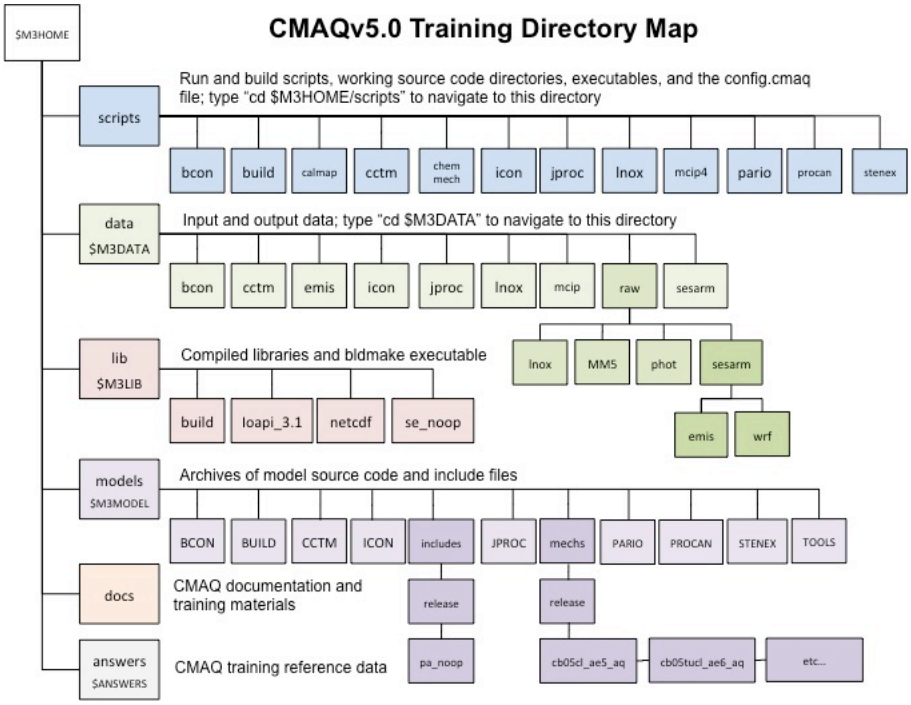
- Episode nomenclature
- Episode start date, start time, duration and time step
- Logfile and diagnostic options
- Input/output directories and file names
- Horizontal grid name and grid definition
- Synchronization time step
- Specifications for integral averaged species output
- Diagnostic output file options
- Process analysis domain specifications
- Executable name and location

56

CCTM – Execution Options(2)

- Inline and science options
  - Calculate inline windblown dust emissions
  - Include the impacts of erodible agricultural land on windblown dust emissions
  - Calculate lightning NOx emissions
  - Use a land-use based vertical diffusivity
  - Calculate deposition velocities inline
  - Calculate land-use specific deposition velocities with MOSAIC
  - Use the ammonia bi-directional flux module
  - Use the mercury bi-directional flux module
  - Simulate atmosphere-surface HONO interactions
  - Compute biogenic emissions inline with the BEIS3 model
  - Compute plume rise for point sources inline

57



Training details

- Training case was created from National Park Service modeling study (RoMANS)
  - July 7, 2006 on two domains centered on Denver, CO
  - 50x50x16 cell, 12 km resolution
  - 50x50x16, 4 km resolution
- CB05 chemical mechanism (EBI solver), aqueous chemistry, and aerosols (ISORROPIA for thermodynamics) – Mechanism cb05cl_ae5_aq
- MCIP4.0
- ACM clouds
- No process analysis

59

About This Training

- Specific to Linux and portable to most Linux platforms
- Run CMAQ from scripts
- Emphasis on operational, not technical aspects of CMAQ
- Basic CMAQ functions covered

60

Air Quality Modeling Concepts

- Model initialization/spin-up
- Multi-day simulations
- Nested simulations

61

Initialization/Spin-up

- Initialization is performed to minimize the effects of initial conditions on the simulation results
- Spin-up: A period at the beginning of a simulation with the purpose of initializing the model. Spin-up period results are typically discarded
- No rule of thumb for the CMAQ spin-up duration
  - Dependent on the grid resolution, magnitude of forcing from emissions and boundary conditions, and strength of horizontal and vertical transport
  - Convention is to use a 2-week spin-up for regional modeling simulations

62

Multi-day Simulations

- CMAQ simulations are typically continuous
- No re-initialization from clean conditions after the spin-up period
- To keep output file sizes manageable, multi-day simulations are stopped and restarted for each simulation day
  - Subsequent days use the simulated concentrations from the previous day as initial conditions

	Day 1	Day 2	Day 3	Day 4	Day 5
IC	clean	$CONC_1$	$CONC_2$	$CONC_3$	$CONC_4$
Output	$CONC_1$	$CONC_2$	$CONC_3$	$CONC_4$	$CONC_5$

63

Nested Simulations

- Nested modeling refers to embedding consecutively higher resolution modeling grids within each other
- Nesting is used to simulate the lateral boundary conditions for a modeling domain
- In the image to the right, grid 2 is "nested" in grid 1 and grid 3 is "nested" in grid 2
- The BCs for grid 1 are "clean" and uniform in time and space
- BCs for grids 2 and 3 are dynamic and extracted from previous grid

64

CMAQ Analysis Concepts

- Process Analysis
- Model Performance Evaluation and QA Procedures

65

Process Analysis (PA)(1)

- Eulerian grid models are based on partial differential equations that define the time-rate of change in species concentrations due to chemical and physical processes
- PA is a configuration system within Eulerian models that provides quantitative information about the impacts of individual processes on the cumulative chemical concentrations
- PA is an optional feature of CMAQ that provides insight into the reasons for a model's predictions

66

Process Analysis (PA)(2)

- Two classes of PA
  - Integrated reaction rates (IRR)
  - Integrated process rates (IPR)
- PA is useful for
  - Identifying sources of error
  - Interpreting model results
  - Determining the important characteristics of chemical mechanisms (IRR)
  - Determining the important characteristics of different implementations of physical processes (IPR)
  - IPR quantifies the contribution of each source and sink process for a particular species at the end of each time step

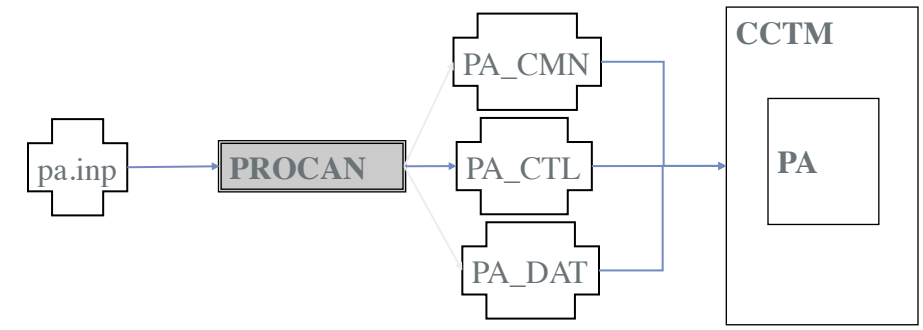
67

Process Analysis (PA)(3)

- CMAQ implementation requires compiling the CCTM with PA include files generated by the PROCAN preprocessor
- PA include files specify
  - IRR or IPR
  - Chemical species or groups to collect PA information about
- A PROCAN configuration file defines the contents of the include files
- A PROCAN run script uses information in the configuration file and calls the executable to create the include files

68

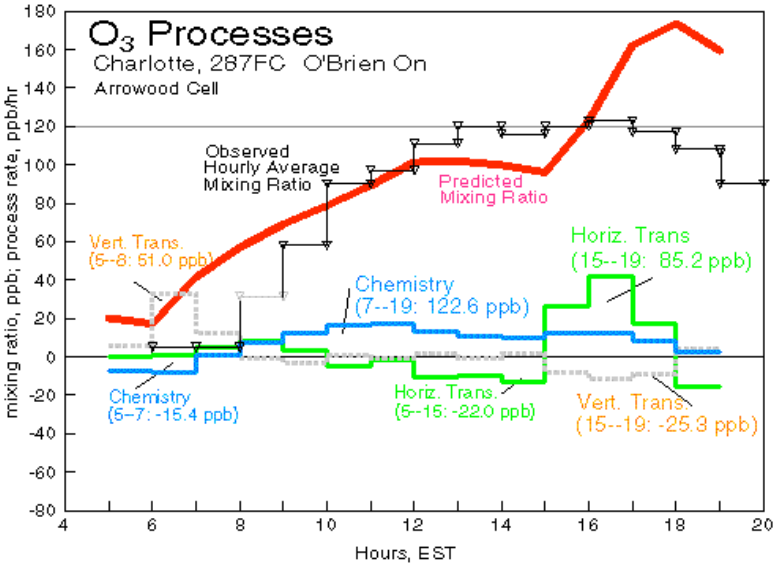
Process Analysis (PA)(4)



Configuration File

Include Files

69



70

Process Analysis (PA)(7)

- IRR quantifies the mass throughput of a particular reaction within a chemical mechanism
- IRR can diagnose mechanistic and kinetic problems within the chemistry model
- IRR can reveal NO_x vs. VOC sensitivity regimes
- IRR generally more difficult to interpret than IPR

71

Model Performance Evaluation (MPE)

- Question why a model is doing what it is doing
- What are the inherent uncertainties and how do they impact the model results
- Qualitative and quantitative evaluation
- Diagnostic versus operational evaluation
- Comparisons against observations
- Evaluate at different temporal and spatial scales
- Categorical model evaluation (used for Forecasting)
  - Contingency Table, False Alarm Rate, Skill Scores, CSI, etc.

72

Quantitative vs Qualitative

- Qualitative model evaluation targets intuitive features in results
  - Effects of urban areas
  - Boundary layer effects
  - Effects of large point sources and highways
  - Diurnal phenomena
- Quantitative evaluation provides statistical evidence for model performance
  - Daily, seasonal, annual comparisons with observed data
  - At coarse grids, compare observations with the concentrations in the model grid cell in which the monitor is located
  - At fine grids, compare observations with the concentrations in a matrix of cells surrounding the cell in which the monitor is located

73

Problems/Issues

- Modeling scales have grown tremendously both spatially and temporally
  - Datasets becoming larger
  - Need to process and digest voluminous amount of information
- Heterogeneous nature of observational datasets
  - Vary by network, by quality, by format, by frequency
- Measurement or model artifacts
  - What is modeled is not always measured
  - Need adjustments before comparisons
- Problem of incommensurability
  - Comparing point measurement with volume average

74

Observational Databases

- AIRS (hourly) (~4000)
- IMPROVE (every 3rd day) (~160)
- CASTNET (hourly, weekly) (123)
- NADP (weekly) (over 200)
- EPA Supersites (sub-hourly) (8)
- EPA STN (hourly) (215)
- PAMS (hourly) (~130)
- AERONET
- Special field campaigns
  - e.g. AIRMAP, ASACA, BRAVO, CCOS, CRPAQS, NARSTO, SEARCH, SOS, TXAQS, etc.
  - Aircraft Data
- Remote Sensing Data (AURA, MODIS, etc.)

75

Operational Evaluation (mostly quantitative)

- Compute suite of statistical measures of performance
  - Peak Prediction Accuracy, Bias metrics (MB, MNB, NMB, FB), Error metrics (RMSE, FE, GE, MGE, NMGE), etc.
  - "Goodness-of-fit" measures (based on correlation coefficients and their variations)
  - Various temporal scales
- Time-series analyses
  - Hourly, weekly, monthly
- Grid (tile) plots
- Scatter plots
- Pie-charts

76

Diagnostic Evaluation (qualitative and quantitative)

- Compute various ratios
  - Metrics different for each problem being diagnosed / studied
    - O_3/NO_z, H_2O_2/HNO_3 for NO_x versus VOC limitation
    - NO_z/NO_y for chemical aging
    - PM species ratios such as NH_3/NH_x, NO_3/(total nitrate) for gas-particle partitioning, NH_4/SO_4, NH_4/NO_3, etc.
    - Others?
- Innovative Techniques
  - Empirical Orthogonal Functions
  - Principal Component Analyses
  - Process Analyses
  - Source Apportionment (available for Carbon and Sulfur)
  - Decoupled-direct method (DDM)
  - Others?

77

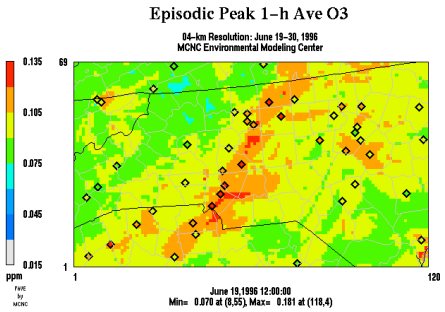
Analyses Tools for MPE

- *sitecmp* to prepare obs-model pairs
  - Part of CMAQ Distribution
- VERDI
  - http://www.verdi-tool.org
- I/O API Utilities
  - http://www.baronams.com/products/ioapi
- netCDF Operators
  - http://nco.sourceforge.net
- NCAR Command-line Language
  - http://www.ncl.ucar.edu
- Python I/O API Tools
  - http://www-pcmdi.llnl.gov/software-portal/Members/azubrow/ioapiTools/index_html
- Atmospheric Model Evaluation Tool (AMET)
  - http://www.cmascenter.org

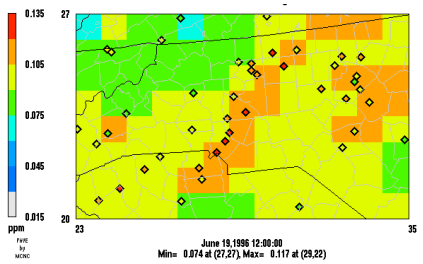
78

MPE Example 1

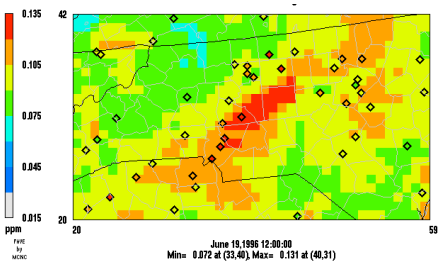
Grid Resolution Variability



4-km



36-km



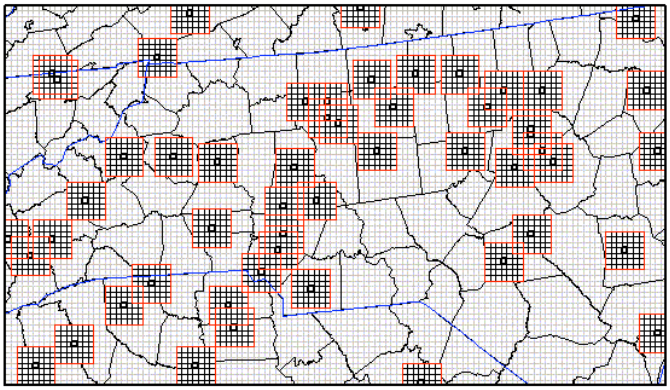
12-km

79

MPE Example 2

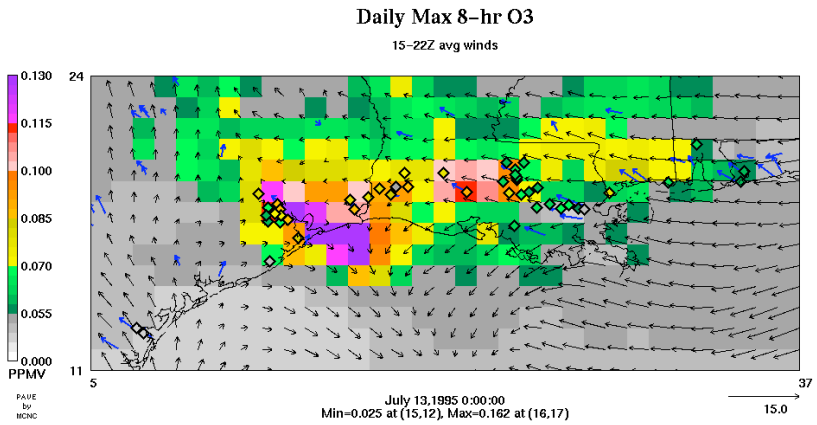
Spatial Variability of Peak Predictions

7x7 Grids and Monitor Locations
(1996 Episode)



80

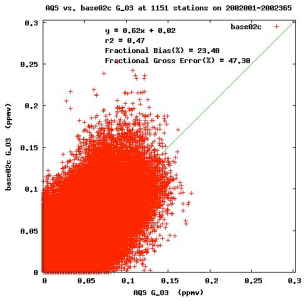
MPE Example 3
Wind and Obs Overlay



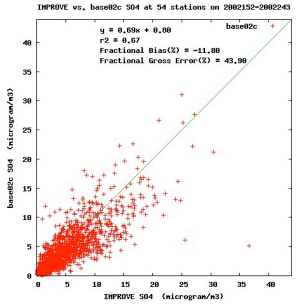
81

MPE Example 4
Scatter Plot Analyses

- Regression analyses present model results across multiple observation points or time periods



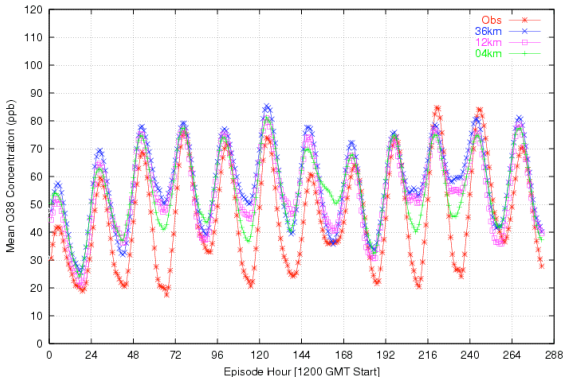
O_3



SO_4

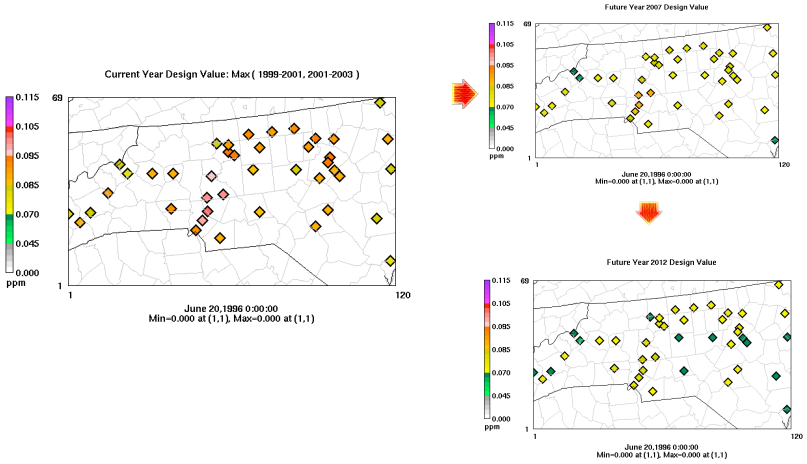
82

MPE Example 5
Time Series Analyses



83

MPE Example 6
Attainment Demonstration for O_3

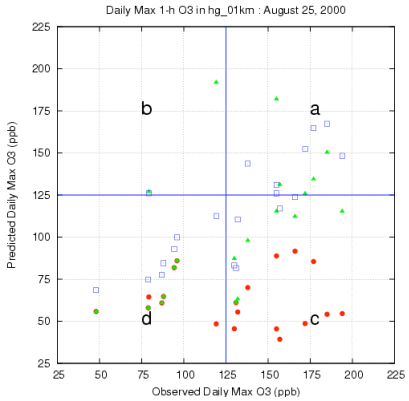


84

MPE Example 7
Forecast Model Evaluation

		Observed	
		No	*Yes*
Predicted	*Yes*	b	a
	No	d	c

		Observed	
		No	*Yes*
Predicted	*Yes*	False Alarm	Hit
	No	Correct Negative	Miss



85

CMAS Training – UNIX Basics

Basic commands

A command is simply a program that tells the computer to do something. In Windows, you usually start programs using an icon; in UNIX, you start programs by typing a command. Most of the commands you will need for this class have the form:

command *argument*

The *argument* indicates what the command should perform its action on, usually a file. Some commands don't require an *argument*, while other commands may need more than one. Commands in UNIX are case sensitive; **command** and **Command** are not the same.

Here are some commands you'll probably use in this class:

Command	***Action***
man *command*	Get help for the given *command*
ls *directory*	List the contents of *directory*; if no *directory* is given, **ls** lists the contents of the directory you are currently in
cd *directory*	Change to *directory*
pwd	Print the path of the directory you are currently in
cp *file1 file2*	Copy *file1* to *file2*; **cp** will overwrite *file2* if it exists
mv *file1 file2*	Move (rename) *file1* to *file2*; **mv** will overwrite *file2* if it exists
rm *file*	Remove (delete) *file1*
head *file*	Display the first 10 lines of *file*
tail *file*	Display the last 10 lines of *file*
more *file*	Display *file* one page at a time
grep *pattern file*	Search for the given *pattern* in *file*
setenv *name value*	Assign *value* to the environment variable named *name*
echo $*name*	Print the value associated with the environment variable named *name*; note that the $ character immediately precedes *name* (no space)
command1 | **command2**	Send the output from **command1** to **command2**; usually **command2** is the **more** command to display the output from **command1** one page at a time
ncdump *file*	Dump the NetCDF file *file* as text
pave -f *file1* -f *file2* … -f *file*n	Run **pave** and load in *file1*, *file2*, … *file*n; you must give the full path and file name for each file loaded

Files and directories

Files in UNIX are organized into directories, similar to folders in Windows. While Windows has the base directory **C:**, the UNIX base directory is just **/**. The files you will use in this class are located in the directory **/home/training/smoke** for the SMOKE course and **/home/training/cmaq** for the CMAQ course. You will use the **cd** command to move into different directories and the **ls** command to list the files in a particular directory.

One useful construct in UNIX is the concept of relative paths. The parent of any directory can be referenced as "**..**". For example, if you are currently in the directory **/home/training/smoke/data/ge_dat** and want to change to the directory **/home/training/smoke/data/inventory**, you can use the command:

```
cd ../inventory
```

This command indicates that you should first go up one directory (into **/home/training/smoke/data**) and then go into the **inventory** directory.

When providing file names as arguments to UNIX commands (i.e. **more** *file*), you must tell the system exactly where *file* is located. One way to do this is to use the full path (also called the absolute path) and file name. Suppose you want to look at the **scc_desc.txt** file in the **/home/training/smoke/data/ge_dat** directory. Regardless of what directory you are currently in, you could use the following command to view the file:

```
more /home/training/smoke/data/ge_dat/scc_desc.txt
```

Most UNIX commands will look in the current directory when just given a file name. So, if you are already in the **/home/training/smoke/data/ge_dat** directory, you can just use the command:

```
more scc_desc.txt
```

Another option is to use relative path names. If you are in the directory **/home/training/smoke/data**, you can use the command:

```
more ge_dat/scc_desc.txt
```

You can also use environment variables to store directory or file names. For example, the environment variable SMKROOT is set to **/home/training/smoke**. This means you can use the following command to view the **scc_desc.txt** file:

```
more $SMKROOT/data/ge_dat/scc_desc.txt
```

This last command is the same as using the absolute path and file name, so it can be used from any directory.

Useful shortcuts

The up and down arrows on the keyboard allow you to scroll through previously used commands. If you listed the contents of a directory with **ls** three commands ago, you can hit the up arrow three times to get back to that command. While scrolling through the commands, the up arrow shows you older commands and the down arrow shows you more recently used commands.

The Tab key can be used to complete file and directory names while you're typing commands. Suppose you have a directory containing three files:

bfac.summer.txt bfac.winter.txt costcy.txt

If you want to view the **bfac.summer.txt** file with the **more** command, you can type the following command:

```
more b<Tab>
```

where <Tab> means press the Tab key. The computer will automatically fill in the command as far as it can:

```
more bfac.
```

Since there are two file names that start with **bfac.**, the computer doesn't know which one you want, so you'll need to type more of the name:

```
more bfac.s<Tab>
```

Now the computer can fill in the rest of the file name to complete the command:

```
more bfac.summer.txt
```

If you wanted to view the **costcy.txt** file, you could simply type:

```
more c<Tab>
```

The file name would get filled in completely since there is only one file in the directory whose name starts with the letter "c". Using the <Tab> key to complete file names is especially useful for long file names since you can avoid typing the whole name (and making typos).

The computer you are using for this training also allows you to copy and paste text using the mouse. If you highlight text with the left mouse button, you can paste that text by pressing the middle mouse button.