12. Analysis Tools for CMAQ

Several tools are freely available for visualizing, analyzing, and evaluating CMAQ inputs and outputs. The list includes CMAQ utility tools, m3tools, PAVE, VERDI, Panoply, the Atmospheric Model Evaluation Tool (AMET), netCDF Operators (NCO), UNIDATA Integrated Data Viewer (IDV), and the NCAR Command-line Language (NCL). Several other commercial packages, including MATLAB and IDL, also support the analysis and visualization of CMAQ inputs and outputs. Most visualization and analysis software that supports netCDF file formats will work with CMAQ output data.

This page briefly describes several of these software utilities and provides links to where they may be downloaded. Refer to the documentation for each piece of software for additional information.

Two classes of analysis tools are presented here

- Command line utilities for manipulating CMAQ input/output data
- netCDF
- appendwrf
- bldoverlay
- combine
- hr2day
- merge_aqs_species
- sitecmp
- sitecmp_dailyo3
- writesite
- m3tools
- netCDF Operators
- Visualization tools for graphical display of CMAQ input/output data
- VERDI
- AMET
- PAVE
- IDV
- NCL

netCDF

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

Almost all of the CMAQ input and output files use the I/O API netCDF file format. If the user has already built the netCDF library for compiling CMAQ, the ncdump utility should also be available on the user's machine. This utility generates an ASCII representation of the netCDF file using the CDF notation developed by NCAR.

The UNIX syntax for invoking ncdump is the following:

```
ncdump [-h] [-c] [-n name] [inputfile]
```

where:

-h produces only the "header" information in the output file; i.e., the declarations of dimensions, variables, and attribute, but no data values for the variables.

-c produces the "header" information in the output file and the data values for coordinate variables (variables that are also dimensions).

-n name is used to specify a different name for the network Common data form Description Language (CDL) description than the default.

Command Line Data Processors

CMAQ Utility Tools

http://www.cmaq-model.org

Several post-processing tools (Fortran-based) are provided along with the CMAQ code/scripts distribution. These are located in the $CMAQ_HOME/POST directory in the CMAQ distribution. These tools work directly with the CMAQ outputs and help in processing, formatting, and preparing datasets from various ambient monitoring networks for subsequent evaluation. These networks include the EPA Air Quality System (AQS)AIRS-AQS, Interagency Monitoring of Protected Visual Environments (IMPROVE), Clean Air Status Trends Network (CASTNET), Speciated Trends Network (STN), National Atmospheric Deposition Program (NADP), Mercury Deposition Network (MDN) and the Southeast Aerosol Research and Characterization Study (SEARCH). The various CMAQ utility tools are described below.

appendwrf

This program concatenates variables from multiple WRF input or output files into a single file along the Time (unlimited) dimension. This can be useful in cases where a user may have WRF input or output files that were generated for shorter time periods and wants to combine them into files with longer (e.g. monthly) duration.

Environment variables used:

`INFILE_1` input file number 1, (max of 15).
`INFILE_2` input file number 2, (max of 15).
`OUTFILE` output file name

bldoverlay

This program creates an observation overlay file that can be imported into either PAVE or VERDI. It requires as input a file containing observed data in a specific format, and then creates a PAVE/VERDI compatible overlay file.

Environment variables used:

`SDATE` start date in the format: YYYYDDD
`EDATE` end date in the format: YYYYDDD
`FILETYPE` type of input file to be used (see information below). Choices are: OBS, SITES (default it OBS)
`OLAYTYPE` type of data for the overlay output file. If input data is daily this should be set to DAILY. If input data is hourly choices are: HOURLY, 1HRMAX, 8HRMAX.
`SPECIES` list of names of the species in the input file (e.g. setenv SPECIES 'O3,NO,CO')
`UNITS` list of units of the species in the input file (e.g. setenv UNITS 'ppb,ppb,ppb')
`INFILE` file containing input observed data
`VALUE` static value to use as "observed" concentration for SITES filetype (default is 1)
`OUTFILE` name of overlay file to create

Input file types and format:

Bldoverlay accepts "OBS" and "SITES" formats (FILETYPE) for the input file. For hourly output data (`OLAYTYPE HOURLY`) the program assumes that observations are in local standard time (LST) and applies a simple timezone shift to GMT using timezones every 15 degrees longitude. For daily output data (`OLAYTYPE DAILY, 1HRMAX or 8HRMAX`) no time shifting is done so the output data remains in LST. In this case the user can use the HR2DAY utility to time shift and average hourly model data to create daily model fields in LST.

OBS format:

The OBS format consists of comma separated values in the format YYYDDD, HH, Site_ID, Longitude, Latitude, Value1[, Value2, Value3,...].

`Note that if the input data is daily that an hour column (HH) is still required in the input data file. In this case HH is ignored so the user could set this value to 0 for all records.`

SITES format:

Set to create a static site file using the value set by VALUE (default is 1). The format is a tab delimited file with the structure Site_ID Longitude Latitude.

block_extract

- add content here

combine

This program combines fields from a set of IOAPI or wrfout input files to an output file. The file assigned to environmental variable SPECIES_DEF defines the new species variables and how they are constructed. This means that all the species listed in the SPECIES_DEF files need to be output when CMAQ is being run. One option is to set the ACONC (or CONC) output to be all species.

Environment variables used:

`GENSPEC` Indicates to generate a new SPECIES_DEF file (does not generate OUTFILE)
`SPECIES_DEF` Species definition file defining the new variables of the output file
`INFILE1` input file number 1, (max of 9).
`OUTFILE` IOAPI output file name, opened as read/write if it does not exist and read/write/update if it already exists

Environment Variables (not required):

`IOAPI_ISPH` projection sphere type (use type #20 to match WRF/CMAQ)
(the default for this program is 20, overriding the ioapi default of 8)

Record type descriptions in SPECIES_DEF file

/ records are comment lines
! records are comment lines
records can be used to define parameters:

All other records are read as variable definition records

Formular expressions supports operators ^ + - * / are evaluated from left to right using precedence order of ^*/+-. Order of evaluation can be forced by use of parentheses. When part of an expression is enclosed in parentheses, that part is evaluated first. Other supported functions include “LOG”, “EXP”,

"SQRT", and "ABS". In addition, expressions can be combined to create conditional statements of the form "expression_for_condition ? expression_if_true : expression_if_false".

Variables from input files are defined by their name followed by its file number enclosed in brackets. Once defined in a species definition file, variables can subsequently be referred to by their name and the number zero enclosed in brackets. Adding a + or - sign before the file number within the bracket instructs combine to use the variable value for the next or previous timestep instead of the current time step when evaluating the expression. This can be used to define variables that are computed as difference between the current and previous time step, for example to compute hourly precipitation as the difference in WRF cumulative precipitation values between successive timesteps.

Examples of possible expressions are shown in the sample SPECIES_DEF files distributed with the CMAQ_TOOLS package.

hr2day

This program creates gridded I/O API files with daily values from gridded I/O API files containing hourly values.

Environment variables used:

Environment Variables (not required):

Species and operator definitions: Defines the name, units, expression and daily operation for each variable in OUTFILE. These definitions are specified by environment variables SPECIES_[n]

merge_aqs_species

This program creates a merged AQS data file from pre-generated files posted on the EPA's AQS website (link below). The user must specify the location where the merged files should be created, the base location of the downloaded AQS files (it is then assumed the files will be in sub-directories from the base directory of /YYYY/hourly and YYYY/daily). The user must also specify the year (YYYY) and whether merging daily or hourly files (the script must be run separately for each time average).

The formatted observation files generated from running this script have been provided in this release for 2001 - 2014. This utility is included to allow the user to generate formatted observation files for different years if needed.

This program requires the R script merge_aqs_species.R. The R code will work with the base installation of R (https://cran.r-project.org/) and does not require installation of any additional libraries.

This utility also requires .csv files downloaded from the EPA's AQS website:

http://aqsdr1.epa.gov/aqsweb/aqstmp/airdata/download_files.html

The required files from that site are:

By default, the species merged are

sitecmp

This program generates a csv (comma separated values) file that compares CMAQ generated concentrations with an observed dataset.

Environment Variables (required):

Environment Variables (not required):

Species Definitions: Defines the data columns for your output file. Each can specify the observed and modeled variables of the species you are analyzing. These definitions are specified by environment variables [species-type]_[1-50], where species type is one of the following {AERO, GAS, WETCON, WETDEP, PREC}. See the sample run scripts for additional examples beyond those listed below.

File formats:

sitecmp_dailyo3

This program generates a csv (comma separated values) file that compares various daily ozone metrics computed from hourly CMAQ generated and observed ozone concentrations. The metrics included in the output file are daily maximum 1-hr ozone concentrations, daily maximum 1-hr ozone concentrations in the nine cells surrounding a monitor, time of occurrence of daily maximum 1-hr ozone concentrations, daily maximum 8-hr ozone concentrations, daily maximum 8-hr ozone concentrations in the nine cells surrounding a monitor, time of occurrence of daily maximum 8-hr ozone concentrations, the daily W126 ozone value, and the daily SUM06 ozone value.

Environment Variables (required):

Environment Variables (not required):

File formats:

writesite

This program generates a csv file from an IOAPI data file for a set of species at defined site locations.

Options:

- Program can shift to local standard time for hourly data based on default time zone file
- Data at all cells or at defined site locations can be specified
- Date range can be specified
- Grid layer can be specified

Environment variables:

Environment Variables (not required):

M3tools

<https://www.cmascenter.org/ioapi/>

An extensive set of utility programs called *m3tools* that use the I/O API library have been developed and made available for the modeling community. These utility routines assist in manipulating dates and times, performing coordinate conversions, storing and recalling grid definitions, sparse matrix arithmetic, etc., as well as in data manipulation and statistical analyses. All *m3tools* can be run at the command line, and the various options can be provided interactively, or all of them can be stored in a file and executed as scripts.

A list of these utility programs and brief descriptions is provided below.

- **airs2m3**: Imports air quality monitor data from an AIRS AMP350-format ASCII file and puts them into an I/O API "observational data" file
- **bcwndw**: Extracts data from a gridded file to the boundary of a subgrid window (see **m3wndw** later in this list for extracting to the window itself)
- **datshift**: Takes calendar date (form YYYYMMDD) and a number of days D, and reports the date D days later.
- **gregdate**: Computes calendar-style date "Month DD, YYYY", day-of-week (Sunday, Monday, ..., Saturday), and whether or not Daylight Saving Time is in effect from Julian date YYYYDDD, or from "yesterday", "today", or "tomorrow"
- **juldate**: Computes Julian date YYYYDDD, day-of-week (Sunday, Monday, ..., Saturday), and whether or not Daylight Saving Time is in effect from calendar-style date "Month DD, YYYY", or from "yesterday", "today", or "tomorrow".
- **m3combo**: Computes linear combinations of sets of variables from an I/O API input file, and writes the resulting variables to an I/O API output file
- **m3cple**: Copies to the same grid, or interpolates to another grid, a time sequence of all variables from a source file to a target file, under the optional control of an I/O API coupling-mode "synch file"
- **m3diff**: Computes statistics for pairs of variables and for the results of applying various comparison ("differencing") operations to those variables in a pair of files.
- **m3edhdr**: Edits header attributes/file descriptive parameters

- **m3fake**: Builds a file according to user specifications, filled either with dummy data or with data read in from a set of user-supplied files
- **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
- **m3pair**: Builds an ASCII file of paired values for two variables from two files, within a user-selected window into the grid, according to user specifications
- **m3stat**: Computes statistics for variables in a file
- **m3tproc**: Computes time period aggregates (e.g., 08:00-16:00 gridded daily maxima) and writes them to an output file. Can be used to create running averages, (e.g., 8-h O3 data from 1-h O3), daily averages, daily maxima, etc.
- **m3tshift**: Copies/time-shifts data from a file
- **m3wndw**: Windows data from a gridded file to a subgrid (see **bcwndw** earlier in this list for extracting to the boundary of the subgrid window)
- **m3xtract**: Extracts a subset of variables from a file for *<time interval>* .Can also be used to concatenate data from two or more files with different time periods into one file
- **m4filter**: Converts first-edition Models-3 files to current version
- **mtxblend**: Uses a sparse-matrix file to interpolate/transform data from an input file to the grid of a "base" file and to merge it with data from the "base" file
- **mtxbuild**: Builds a sparse-matrix transform file from user-supplied ASCII coefficient inputs
- **mtxcalc**: Builds a grid-to-grid sparse-matrix transform file using a sub-sampling algorithm
- **mtxcple**: Uses a sparse-matrix file to interpolate a time sequence of all variables from a source file to a target file, under the optional control of an I/O API coupling-mode "synch file"
- **presterp**: Interpolates from a 3-D sigma-coordinate file to a new 3-D pressure-coordinate file, using coefficients from PRES_CRO_3D
- **selmrg2d**: Selects multiple 2-D layer/variable combinations from multiple gridded input files, and writes result to merged 2-D gridded output file
- **utmtool**: Performs coordinate conversions and grid-related computations for lat-lon, Lambert, and UTM coordinate systems.
- **vertot**: Computes vertical-column totals of variables in a file

netCDF Operators (NCO)

http://nco.sourceforge.net/

The netCDF Operators (NCO) are a suite of programs known as operators. Each operator is a stand-alone, command-line program that is executed at the UNIX shell level, similar to the commands ls or mkdir. The operators take netCDF files

as input, then perform a set of operations (e.g., deriving new data, averaging, hyperslabbing, or metadata manipulation) and produce a netCDF file as output. The operators are primarily designed to aid manipulation and analysis of gridded scientific data. The single command style of NCO allows users to manipulate and analyze files interactively and with simple scripts, avoiding the overhead (and some of the power) of a high-level programming environment.

NCO achieves flexibility by using *command-line options*. These options are implemented in all traditional UNIX commands as single-letter *switches*, e.g., 'ls -l'. NCO supports both short-format (single letter) and long-format (multiletter) options.

An overview of the various netCDF operators is given below.

- **ncap (netCDF Arithmetic Processor)**: ncap and ncap2 arithmetically process netCDF files. The processing instructions are contained either in the NCO script file fl.nco or in a sequence of command-line arguments.
- **ncatted (netCDF Attribute Editor)**: ncatted edits attributes in a netCDF file. ncatted can *append*, *create*, *delete*, *modify*, and *overwrite* attributes (all explained below). Furthermore, ncatted allows each editing operation to be applied to every variable in a file. This saves time when changing attribute conventions throughout a file.
- **ncbo (netCDF Binary Operator)**: ncbo performs binary operations on variables in *file_1* and the corresponding variables (those with the same name) in *file_2* and stores the results in *file_3*. The binary operation operates on the entire files.
- **ncea (netCDF Ensemble Averager)**: ncea performs grid-point averages of variables across an arbitrary number (an *ensemble*) of *input-files*, with each file receiving an equal weight in the average. Each variable in the *output-file* will be the same size as the same variable in any one of the *input-files*, and all *input-files* must be the same size. ncea averages entire files, and weights each file evenly. This is distinct from ncra (discussed later in this list), which averages only over the record dimension (e.g., time), and weights each record in the record dimension evenly; ncea *always averages* coordinate variables, regardless of the arithmetic operation type performed on the noncoordinate variables. All dimensions, including the record dimension, are treated identically and preserved in the *output-file*.
- **ncecat (netCDF Ensemble Concatenator**): ncecat concatenates an arbitrary number of input files into a single output file. A new record dimension acts as the glue to bind the input files data together. Each variable in each input file becomes one record in the same variable in the output file. All *input-files* must contain all extracted variables (or else there would be "gaps" in the output file). Each extracted variable must be constant in size and rank across all *input-files*. The *input-files* are stored consecutively as a single record in *output-file*. Thus, the *output-file* size is

the sum of the sizes of the extracted variable in the input files.

- **ncflint (netCDF File Interpolator)**: ncflint creates an output file that is a linear combination of the input files. This linear combination is a weighted average, a normalized weighted average, or an interpolation of the input files. Coordinate variables are not acted upon in any case; they are simply copied from *file_1*.
- **ncks (netCDF Kitchen Sink)**: ncks combines selected features of ncdump, ncextr, and the nccut and ncpaste specifications into one versatile utility. ncks extracts a subset of the data from *input-file* and prints it as ASCII text to stdout, writes it in flat binary format to binary-file, and writes (or pastes) it in netCDF format to *output-file*.
- **ncpdq (netCDF Permute Dimensions Quickly)**: ncpdq performs one of two distinct functions—packing or dimension permutation—but not both. ncpdq is optimized to perform these actions in a parallel fashion with a minimum of time and memory.
- **ncra (netCDF Record Averager)**: ncra averages record variables across an arbitrary number of *input-files*. The record dimension is, by default, retained as a degenerate (size 1) dimension in the output variables.
- **ncrcat (netCDF Record Concatenator)**: ncrcat concatenates record variables across an arbitrary number of *input-files*. The final record dimension is by default the sum of the lengths of the record dimensions in the input files.
- **ncrename (netCDF Renamer)**: ncrename renames dimensions, variables, and attributes in a netCDF file. Each object that has a name in the list of old names is renamed using the corresponding name in the list of new names. All the new names must be unique.
- **ncwa (netCDF Weighted Averager)**: ncwa averages variables in a single file over arbitrary dimensions, with options to specify weights, masks, and normalization.

Visualization Tools

Visualization Environment for Rich Data Interpretation (VERDI)

http://www.verdi-tool.org

The Visualization Environment for Rich Data Interpretation (VERDI) is a flexible and modular Java-based visualization software tool that allows users to visualize multivariate gridded environmental datasets created by environmental modeling systems such as SMOKE, CMAQ and WRF, namely gridded concentration and deposition fields that users need to visualize and compare with observational data both spatially and temporally. VERDI has been designed keeping most of the functionality of PAVE in mind, and hence can help users analyze and visualize model outputs in a very similar vein, using both command-

line driven scripts as well as using a Graphical User Interface (GUI). Further, VERDI is under active development to enhance its features beyond PAVE.

Atmospheric Model Evaluation Tool (AMET)

http://www.cmascenter.org

The Atmospheric Model Evaluation Tool (AMET) is a suite of software designed to facilitate the analysis and evaluation of meteorological and air quality models. AMET matches the model output for particular locations to the corresponding observed values from one or more networks of monitors. These pairings of values (model and observation) are then used to statistically and graphically analyze the model's performance. More specifically, AMET is currently designed to analyze outputs from MM5, WRF, CMAQ, and CAMx as well as MCIP-postprocessed meteorological data (surface only).

The basic structure of AMET consists of two ''fields"and two *processes*. The two fields (scientific topics) are MET and AQ, corresponding to meteorology and air quality data. The two processes (actions) are database population and analysis. Database population refers to the underlying structure of AMET; after the observations and model data are paired in space and time, the pairs are inserted into a MySQL database. Analysis refers to the statistical evaluation of these pairings and their subsequent plotting. Practically, a user may be interested in using only one of the fields (either MET or AQ), or may be interested in using both fields. That decision is based on the scope of the study. The three main software components of AMET are MySQL (an open-source database software system), R (a free software environment for statistical computing and graphics), and perl (an open-source, cross-platform programming language).

Package for Analyses and Visualization of Environmental Data (PAVE)

http://paved.sourceforge.net

PAVE is a flexible and distributed application to visualize multivariate gridded environmental datasets. Features include

- baseline graphics with the option to export data to high-end commercial packages
- the ability to access and manipulate datasets located on remote machines
- support for multiple simultaneous visualizations
- an architecture that allows PAVE to be controlled by external processes
- low computational overhead
- no software distribution cost

PAVE is very widely used by the air quality modeling community, and it can produce various types of plots, including scatter plots, time-series plots, 2-D tile

plots, 3-D surface plots, bar plots, wind-vector plots, etc. The source code for PAVE is also distributed under the terms of the GNU General Public License Version 2. PAVE can be run at the Linux command prompt, and the various commands/options can be invoked using the graphical user interface (GUI), or all of them can be stored in a script file and executed by running the script. However, note that PAVE is not being updated any more, and CMAS has ceased support for PAVE, and encourages the user community to move towards VERDI (discussed next).

Integrated Data Viewer (IDV)

http://www.unidata.ucar.edu/software/idv/

The Integrated Data Viewer (IDV) from Unidata is a Java™-based software framework for analyzing and visualizing geoscience data. The IDV release includes a software library and a reference application made from that software. It uses the VisAD library (<http://www.ssec.wisc.edu/~billh/visad.html>) and other Java-based utility packages.

The IDV is developed at the Unidata Program Center (UPC), part of the University Corporation for Atmospheric Research in Boulder, CO, which is funded by the National Science Foundation. The software is freely available under the terms of the GNU Lesser General Public License.

The IDV "reference application" is a geoscience display and analysis software system with many of the standard data displays that other Unidata software (e.g., GEMPAK and McIDAS) provides. It brings together the ability to display and work with satellite imagery, gridded data (for example, numerical weather prediction model output), surface observations, balloon soundings, NWS WSR-88D Level II and Level III RADAR data, and NOAA National Profiler Network data, all within a unified interface. It also provides 3-D views of the earth system and allows users to interactively slice, dice, and probe the data, creating cross-sections, profiles, animations and value read-outs of multidimensional data sets. The IDV can display any Earth-located data if they are provided in a supported format.

IDV includes the capability to read I/O API netCDF formatted files and a scripting interface to create and manipulate images and movies. The scripting is accomplished through an XML file: *IDV Scripting Language* (ISL). The ISL file can be opened from a running IDV, or one can be passed to the IDV as a command-line argument:

```
runIDV capture.isl
```

NCAR Command Language (NCL)

http://www.ncl.ucar.edu

The NCAR Command Language (NCL) is a free, interpreted language designed specifically for scientific data processing and visualization. NCL has robust file input and output. It can read in netCDF, HDF4, HDF4-EOS, GRIB, binary, and ASCII data. The output graphics from NCL are of high quality, and highly customizable.

It runs on many different operating systems, including Solaris, AIX, IRIX, Linux, MacOSX, Dec Alpha, and Cygwin/X running on Windows. It is available for free in binary format.

NCL can be run in interactive mode, where each line is interpreted as it is entered at the user's workstation, or it can be run in batch mode as an interpreter of complete scripts. The user can also use command-line options to set options or variables on the NCL command line.

The power and utility of the language are evident in three areas:

- file input and output
- data analysis
- visualization

NCL has many features common to modern programming languages, including types, variables, operators, expressions, conditional statements, loops, and functions and procedures. The various NCL commands can be executed at the NCL command prompt, or all the commands can be stored in a file and invoked as follows:

ncl commands.ncl

NCL comes with numerous predefined functions and resources that the user can easily invoke. These include functions for data processing, visualization, and various mathematical and statistical analyses, such as empirical orthogonal functions (EOFs) and singular value decomposition (SVD). As an example, contributed.ncl is a library of user-contributed functions within NCL. This library is distributed with NCL, and loading the script at the beginning of the user's NCL script therein can access the functions.

```
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/contributed.ncl"
```

NCL also has a capability to call external Fortran or C routines. This is enabled through an NCL wrapper script called WRAPIT. The wrapper file is a C program that describes all of the arguments and passes them back and forth between the function/procedure the user wants to call, and the NCL script that is calling it. Thus, when the user invokes WRAPIT on the Fortran code that needs to be called from NCL, it creates a special C wrapper file, compiles it and the Fortran file, and generates a *.so file that the user can then load into NCL using the "external" statement.
***