Appendix for

Development of a Source Oriented version of the WRF-Chem Model and its Application to the California Regional $PM_{10}/PM_{2.5}$ Air Quality Study

Hongliang Zhang[1], Steve P. DeNero[1], David K. Joe[1], Hsiang-He Lee[2], Shu-Hua Chen[2], John Michalakes[3],and Michael J. Kleeman[1,]*

[1]Department of Civil and Environmental Engineering, University of California, Davis. One Shields Avenue, Davis CA. [2]Department of Land, Air, and Water Resources, University of California, Davis. One Shields Avenue, Davis CA. [3]National Renewable Energy Laboratory, Golden CO

*Corresponding author. Tel.: +1 530 752 8386; fax; +1 530 752 7872. E-mail address: mjkleeman@ucdavis.edu (M.J. Kleeman).

Description of the source oriented WRF Chem model (SOWC)

This appendix describes the changes made to the original WRF-Chem source code to create the SOWC, and is separated into the major processes and methods therein. The changes fall within three main branches of the code: The Registry, the chemistry driver, and the Eulerian-mass conservation dynamic core. Figures A1, A2, and A3 provide an illustrative view of the flow of information within the framework of the SOWC model. Figure 1illustrates the WRF call structure that controls this time-looping behavior. Figure 2 illustrates the call overall call structure of the chemistry driver, and names the files that were edited or created with the creation of the SOWC model. Figure A3 provides this same insight within the Eulerian mass conservation dynamic core.

1 THE REGISTRY

The purpose and functionality of the WRF Registry is described in the *WRF Tiger Team Documentation: The Registry* (Michalakes and Schaffer, 2004). In brief, the Registry is used to store information regarding every variable within the modeling framework. The information stored includes the

variable's name, the symbol used model-wide to represent it, the number and specifications of its dimensions, which sections of WRF-Chem it is to be included in, and the variable's units. At the time the code is compiled, the Registry also provides low-level information to the automated scripts which are responsible for writing most of WRF-Chem's structure.

The goal of this study was to track source-oriented and size resolved particles across the modeling domain, and observe their interaction with the regional meteorology. The particle variables created for this study were therefore configured to have six dimensions; three dimensions for Cartesian coordinates; one for the particle size bin; one for the source origin; and one final dimension for the individual chemical species. Particle phase concentrations were stored in a six-dimensional array with these properties, which was simply named AQC for Air Quality Concentration. Appendix A summarizes AQC and other variables created within the SOWC model to support source-oriented calculations. In addition to the AQC array, separate arrays were also created for particulate matter emissions from area sources (EMIS_AQC), particulate matter emissions from point sources (EMISP_AQC), gas-phase species (AQCG), gas-phase emissions from area sources (EMIS_AQCG), and a final variable array used for writing time-averaged particulate matter concentrations to the history files (AQC_OUT).

The EMIS_AQC array has six dimensions, matching AQC in all aspects except the length of the 2^{nd} and 6^{th} dimensions. The 2^{nd} dimension is reserved in both arrays for vertical layers. EMIS_AQC's 2^{nd} dimension has a length of one (1) since all area sources are ground based. The 6^{th} dimension is reserved for pollutant species and is shorter than that of AQC because EMIS_AQC only describes primary pollutants while AQC must describe primary and secondary pollutants. The EMISP_AQC array has six total dimensions. The first two are not used (dimension of 1) but are maintained only for consistency in the I/O conventions of the SOWC model. This subtle I/O feature is discussed in Appendix Section A. The AQCG array has four dimensions only as it is not source-oriented. The EMIS_AQCG array has five dimensions, maintaining its source-oriented information because it is processed at the same time as the source-oriented particle emissions EMIS_AQC. This feature of the emissions pre-processor is described in greater detail in Appendix Section C. Emissions of each gas phase pollutant from all sources described in EMIS_AQCG are summed together within the SOWC emissions framework (see Section 4). Finally, the AQC_OUT array has six dimensions, matching AQC in all aspects except the length of the 2^{nd} "vertical extent" dimension, which has a default value of 1 that can be changed at run-time as required for each application of the model. The value of 1 represents the ground, surface-layer of the model, and is used here as a dimension for two purposes. The first of these is that the most relevant pollutant concentrations are those at the surface. All the monitoring data used in the comparisons are from surface stations in California. The other purpose for setting this value to 1 is to save on disk memory space. The

output files for these simulations are several gigabytes each, and the removal of one dimension of these output arrays saves a significant amount of space as well as time spent by the processor writing these arrays to file.

The arrays described above are defined in the file Registry/registry.aqc. This file was created specifically for the SOWC model and is incorporated into the main Registry code via an include statement in the file Registry/registry.chem. Having a separately referenced Registry file for source-oriented variables provides organizational structure and it facilitates the removal of the source-oriented feature for testing purposes. Initial testing with the source-oriented AQC array revealed that the process of writing out the entire 6-dimension array consumed a considerable fraction of the simulation time. Writing the history output file took approximately 30 times longer using 5 source types, 8 size bins, and 40 individual pollutant species compared to time required to write history output files with the original WRF CHEM model. The AQC_OUT array with a maximum vertical dimension of variable 'KAQC' was created partly to reduce the time required for writing output files. The AQC_OUT array averages the content of AQC with respect to time over all dimensions except that the vertical extent KAQC is typically set lower than the vertical extent of the full WRF model (default value of KAQC=1 as described earlier). The value of KAQC can be changed at runtime by editing the 'KAQC' value under the '&Chem' section of the namelist.input file. The value of 1 (one) signifies that only the first (bottom, surface) layer of information stored in AQC in the Z- (or, interchangeably K-) direction will be transferred from AQC to AQC_OUT, and then output to the history files. Note that increasing the value of KAQC will lengthen the amount of time spent on writing history output files, and it will increase the amount of storage space needed to hold all of AQC_OUT. The SOWC model can write AQC (instantaneous concentrations) and/or AQC_OUT (time-averaged concentrations) independently to output files depending on the specification of input/output (I/O) flags in Registry/registry.aqc. Michalakes (2004) provides an overview of each possible Registry I/O flag.

Several variables related to the dimensions of the source-oriented variables are assigned default values in Registry/Registry.EM_CHEM that can be changed at runtime through entries in the namelist.input file under the &time_control section. The variables *first_bin* and *last_bin* control the number of aerosol particle size bins in the simulation, and the variables *first_src* and *last_src* allow the user to control the number of source types in the simulation. *First_bin* and *first_src* have a default value of one (1), *last_bin* has a default value of eight (8), and *last_src* has a default value of ten (10). The *max_point* variable represents the maximum total number of point sources in the emissions array EMISP_AQC. *Max_point* has a default value of 4550, which is the number of point emissions processed in the current study. Lastly, the variables *aqc_adv_opt* and *aqcg_adv_opt* found in namlist.input under the

&dynamics section control the advection options for the species within AQC and AQCG. The treatment of advection in SOWC will be discussed later on in Section 3.10. *Aqc_adv_opt* and *aqcg_adv_opt* have default values of one (1), which signify positive-definite transport.

The WRF modeling system includes a C program in the 'external' directory that reads the contents of the Registry files and then generates portions of FORTRAN subroutines that are included in the actual WRF model. A greater overview of this C program and the changes made to it in order to accommodate the source-oriented variables can be found in Appendix B. The programming approaches adopted in the C program require that all dimensions past three (which normally correspond to the three Cartesian coordinate dimension) have their value appended to the end of the variable name. As an example, the CHEM array included in the original WRF CHEM model has four dimensions whereas AQC included in the new SOWC model has six dimensions. Both CHEM and AQC contain an entry to describe concentrations of elemental carbon (EC). In CHEM, the EC index variable is ECJ that is dimensioned as IKJ. In AQC, the EC index variable is AQCEC that is dimensioned IKJ-Size-Source. During I/O as well as in any input or output files where these variables are included, the CHEM variable is still named ECJ, but the AQC variable has the size bin and source type appended to the end. Since these two dimensions follow the 3rd dimension "J", a new mechanism to retain the pollutant's identity is required. The 6th dimension describing the pollutant species is already accounted for in the name of the variable (AQC*EC*). However, the standard WRF-Chem framework would not be able to distinguish source-types and size bins from one another, and all information would be stored in AQCEC and overwrite everything preceding it. Therefore, the additional two dimensions are appended to the name of the variable. As an example, the variable describing AQCEC concentration values from the fifth size bin and fourth source type is named AQCEC_00005_00004. This methodology is only applied to variable arrays that exceed four dimensions. These changes were made to the C program that reads in the Registry files and automatically generates the first few levels of WRF-Chem's source-code. It was therefore necessary to add two dummy dimensions to EMISP_AQC in order to push the dimension count above 4. In doing so, the appropriate variable suffixes were automatically generated by the C program which corresponds to the target AQC variable. EMISP_AQC will be discussed in more detail in Section 3.

2. CODE CONFIGURATION, COMPILATION FLAGS, AND ENVIRONMENTAL FLAGS

Instructions on how to build and configure the original WRF-Chem code can be found in the WRF user guide online at http://www.mmm.ucar.edu/wrf/users/docs/. It is important to note that the SOWC model framework rests entirely within the Eulerian mass-coordinate solver, or EM for short. This is important because an attempt to build the SOWC model in any of WRF-Chem's other solver options

would not be successful. The steps taken to configure and compile the SOWC for this specific study will be discussed here briefly. Also to be covered are the additions to the code and computing environment to help the user switch back and forth between the original code and the source-oriented version.

The most recent tests for this study have been built and run on 64-bit intel architecture (intel i5 cpu with 8G of RAM) but initial testing demonstrated that the code compiles and runs in the 32-bit environment. The Intel 'ifort compiler with icc for distributed memory in parallel' option was tested in the 32-bit and 64-bit environment. The "PGI compiler with gcc for distributed memory in parallel' option was tested in the 32-bit environment. It was found that the ifort compiler generated slightly faster executable programs, and this compiler was used to produce results in this report.

The memory allocation demands of the SOWC model required the addition of the "-heap_arrays 1024" flag to the 'FCBASEOPTS' group in the file arch/configure.new_defaults when using the ifort compiler. This flag instructs the compiler to store all arrays >1024 KB and place them in heap memory instead of stack memory. The default optimization level within the 'CFLAGS_LOCAL' and 'FCOPTIM' groups was decreased to two (-O2) from the original WRF code default of three (-O3). The –O3 level was observed through iterative testing to lead to decreased processing performance and slower simulation time.

The environment flag *SOURCE_ORIENTED* must be set to true in order to build and run the SOWC model. Setting *SOURCE_ORIENTED* to false will result in the compilation passing over all the SOWC code changes, generating the original WRF CHEM executable. Note that *SOURCE_ORIENTED* must also be set in the run directory's environment. This approach follows in the methodology followed by the *WRF_CHEM* environment variable in the base model.

3 UPDATING PARTICLE RADII AND NUMBER CONCENTRATIONS

The last two species within the AQC array are number concentration and particle radius, respectively (see Appendix A). The radius and number concentration values are carried around in AQC throughout the SOWC model, and are accessible at any point for every simulated size bin and source type. AQC is processed by many operators in the simulated atmosphere that add or remove mass and number from each size bin and source type. Coagulation also combines particles in different size bins and sources types. The radius, mass and number concentration are updated after each operator step in order to maintain internal consistency. These operator steps include the addition of new mass and number through emissions, initial conditions, boundary conditions and condensation of gaseous material; the removal of mass through deposition; the displacement of mass from horizontal and vertical transport; and the re-distributing of mass through evaporation and coagulation. The *densit* subroutine is used to recalculate the

radius and number concentration throughout the SOWC model and can be found in the file 'chem/module_ucd_vvel'. Subroutine *Densit* generally conserves mass and number, making adjustments to particle radius due to the operators described above. Densit applies a default radius for each size bin that contains a mass concentration less than $1x10^{-6}$ $\mu g/m^3$.

4. EMISSIONS

The process of adding fresh source-oriented emissions into the SOWC variable arrays starts within 'chem/emissions_driver' for AQC and for AQCG. AQCG also uses subroutine *add_anthropogenics* found in the file 'chem/module_emissions_athropogenics'. AQC is fed by both the area emissions array EMIS_AQC and the point emissions array EMISP_AQC. AQCG is currently only fed by the area emissions array EMIS_AQCG. The plume heights calculated for point emissions are calculated by subroutine *plumht* in the file 'chem/module_ucd_plume'. This section covers the steps taken to integrate this process into the SOWC model.

EMIS_AQC has six dimensions corresponding to those used by AQC. The process of adding the emissions to the concentration array (AQC) is very straightforward. The units of EMIS_AQC coming into the SOWC model are micro-grams per square meter per second ($\mu g/m^2/s$). AQC has units of micrograms per cubic meter ($\mu g/m^3$). Therefore, to introduce the emissions into the AQC array, a simple conversion factor involving the chemistry time step and cell thickness are applied to the EMIS_AQC value. The product of that operation is then used to update the pre-existing AQC value. It is important to note that the area emissions match AQC in number of sources and sizes, so that in a test involving more than one source type and more than one size bin, the emissions from each source type and each size bin are kept separate as they are added to AQC.

EMISP_AQC technically also has six dimensions. However, as previously reported in section 3.1, the first two of these dimensions are dummy variables with default lengths of one (1) each, so they don't take up additional memory. This is done for the I/O naming convention of variables within the SOWC model so that EMISP_AQC can be matched easily to AQC. Appendix section A, Source Oriented Array Description, gives an overview of the EMISP_AQC array and its dimensioning. The third dimension of EMISP_AQC is equivalent in length to the total number of point emissions taken from the namelist.input state variable *max_point*. This use of the number of point sources and two dummy variables instead of the standard Cartesian coordinates is a memory-saving technique. Most likely very few cells within a given modeling domain will have point source emissions. Therefore, each point was assigned an index value ranging from one (1) to *max_point*. In this way, EMISP_AQC only allocates memory for active point sources and avoids the allocation of a large number of null values that would be inescapable with

Cartesian dimensioning. The point index value inherent in EMISP_AQC must be accompanied by two other arrays that store information about the physical location and emissions properties of each point source. EMISP_INT includes the X and Y grid location of each point source taken from the emissions pre-processor grid. EMISP_REAL stores site-specific latitude and longitude information as well as physical characteristics of each stack. Based on these arrays, emissions from each point source are allocated to their proper x-y grid cell location and the effective plume rise of the emissions can be calculated, allowing emissions to be injected into the proper location within the given WRF modeling domain.

5. INITIAL CONDITIONS

The initial conditions for gases and particulate matter were specified using a subroutine specifically tailored for the target modeling domain and study period. The code for the initial conditions can be found in 'chem/module_chemics'. Concentrations measured during the CRPAQS winter field campaign were interpolated to create a uniform grid following the methods of Goodin et al (1979; 1980). These initial and boundary conditions were used extensively in previous modeling studies (Ying et al., 2008a; Ying et al., 2008b; Ying et al., 2009). In the present study, the interpolated concentrations on the western (upwind) edge of the modeling domain were averaged as a best estimate of background aerosol concentrations.

The other valid option to generate ICs for model calculations involves the use of a spin-up cycle. Liu et al (2001) showed that air quality calculations generally become insensitive to initial conditions after approximately 3 days of time during a typical air pollution episode. Arbitrary initial conditions can therefore be specified if an additional three day period is added to the beginning of the model episode. (for example, starting the test on December 12th instead of December 15th in the present study). The concentration field at the end of the three day initialization period can optionally be saved to efficiently start new calculations. If a spin up period cannot be simulated, an alternative procedure involves using the first three days of the actual episode to create a restart file that is then renamed to the date and time corresponding to the start of the episode. For example, if December 15th were the first day of the episode, then the user would initialize the model with arbitrary concentrations and perform calculations to create a restart file on December 18th. After the December 18th restart file is created, rename that file as December 15th and then start the test anew with the *restart* namelist flag turned on. This will load concentrations from the third day of simulation into the model at the beginning of the first day and the test can then start as if valid ICs had been written into the code structure. This second option is the preferred treatment of this issue when measured initial conditions are not available.

6. BOUNDARY CONDITIONS

Boundary conditions for the current study are based on measured concentrations interpolated using the methods of Goodin et al (1979). All boundary conditions have been used extensively in previous air quality studies. New routines to apply boundary condition to gas and particulate matter concentrations were created within the original WRF framework and implemented for the SOWC model. Calls to the routines *flow_dep_bdy_aqc* and *flow_dep_bdy_aqcg* were added to the file 'dyn_em/solve_em'. The routines themselves are found in 'chem/module_input_chem_data'. The new routines take in the concentration arrays AQC and AQCG one species at a time, determine where the MPI process lies within the modeling domain, and then determines whether to implement boundary conditions to the current grid cell based on wind direction. In order for boundary conditions to be applied in a given grid square, a number of criteria have to be met. First of all, the domain under consideration must be flagged as "specified". This is a flag in the namelist run-time file. Only the outer-most domain is normally specified. Next, the MPI process in question must have an edge that is part of the outer boundary of the domain. Then the grid square being considered must also fall on that outer edge boundary. Finally, the winds at that boundary edge/square must be blowing into the domain. If the grid square in question is not on an outer edge of a specified domain with inward blowing winds, then boundary conditions are not applied to the cell. Boundary conditions are not source-oriented. All boundary conditions are mapped to the largest source-type simulated by the SOWC model. This feature does not apply to *flow_dep_bdy_aqcg* because AQCG is not source-oriented.

flow_dep_bdy_aqc and *flow_dep_bdy_aqcg* call *set_aer_bc_ucd* and *set_gas_bc_ucd,* respectively. Subroutines *set_aer_bc_ucd* and *set_gas_bc_ucd* apply the boundary conditions specified by Ying et al. (2008a; 2008b; 2009) for the CRPAQS field study. The general framework allows the user to specify any value for boundary conditions, including zeroing out these concentrations (normally used for debugging). The same boundary conditions were specified along all four boundaries in the current study, but a framework exists to specify unique boundaries along each edge. A range of boundary conditions can also be specified along the same edge.

7 DRY DEPOSITION

New routines were added to the original WRF framework to perform source-oriented aerosol calculations in the SOWC model. The AQCG gaseous species follow the framework already present in the original WRF chemistry code. The dry deposition of gas and PM routines starts in the file 'chem/dry_dep_driver'. Both PM and gas deposition use the *vertmx* subroutine found in 'chem/module_vertmx_wrf'. Gas deposition also uses subroutine *wesely_driver* in

'chem/module_dep_simple', while the SOWC treatment of PM dry deposition uses subroutine *drydep* from 'chem/module_ucd_vvel'. This sub-section will discuss the treatment of the AQC and AQCG concentration arrays within these subroutines.

There are two main steps involved in the treatment of particulate matter dry deposition. The first step involves calculating the deposition velocity in subroutine *drydep*. *drydep* takes in various physical characteristics of the particles and environment as inputs including particle density and radius, surface roughness, and friction velocity. The deposition velocity calculated for each particle is assumed to hold in any vertical layer. The *drydep* subroutine itself is a version of the dry deposition scheme found in the UCD/CIT model (Kleeman, 2001) that was modified to work in the WRF framework. The deposition velocities calculated in *drydep* are then passed on to the subroutine *vertmx* that handles vertical mixing and species removal due to surface deposition.

Deposition of gaseous AQCG species follows the treatment used for the CHEM variable in the original WRF-CHEM model. Subroutine *wesley_driver* takes information from the physical surroundings and outputs a dry deposition velocity for each chemical species. It does this in a three-step process that follows as (1) calculation of surface resistance, (2) calculation of surface deposition velocity, and (3) calculation of species-specific deposition velocity based on local meteorology and land use. The deposition velocities for AQCG species are passed into subroutine *vertmx* that calculates the vertical mixing and dry deposition loss rates.

There was no precipitation during the CRPAQS winter intensive period, and so there was no wet deposition of the particles or gases from December 15 – 30, 2000. The source code mechanism to carry out this process has therefore not yet been fully implemented into the SOWC model but this model exists in other source-oriented chemical transport models (Mahmud et al., 2010) and can be easily ported to the WRF code.

8. THE SAPRC90 GAS-PHASE MECHANISM

One of the largest additions of new code to develop the SOWC model was the addition of the SAPRC90 gas-phase mechanism. The standard WRF-Chem model has several options for gas-phase reaction mechanisms, but the SOWC model represents the first application of the SAPRC mechanism in WRF. The driving subroutine for the SAPRC90 mechanism, named *saprc90_driver*, is referenced from 'chem/mechanism_driver' and is found in 'chem/module_saprc90'. There are also four (4) data modules that work in conjunction with the SAPRC90 mechanism. They are *module_modlspc*, *module_parameter*, *module_gaskin*, and *module_common*. These are all found in 'chem/module_data_saprc90' and are

referenced in multiple places throughout module_saprc90. The following sub-section will provide an overview of the flow of information in the SAPRC90 mechanism in the SOWC model. There are also mechanism versions from 1999 and 2007 that, with a few minor changes discussed later on in this sub-section, could be implemented with relative ease.

The *saprc90_driver* subroutine is referenced from the *mechanism_driver* subroutine, which acts as a hub for all other possible gas-phase mechanisms. In order to successfully run the SOWC model with SAPRC90, the user's run directory must have two (2) files prior to starting the simulation. The files are referenced within the subroutine *rdmod*, which is called from *saprc90_driver* and are named 'soam_rev29c.mod' and 'soam_ims95a.rxp'. The 'soam_rev29c.mod' file stores properties pertaining to the species involved in the mechanism. For example, it contains the number of active, buildup, constant, and total species in the mechanism as well as a list of all species involved with initial conditions for the constant species. It also contains molecular weights of all species involved, steady state variable names, variable and constant coefficient names, wavelength-dependent light absorption coefficient and quantum yield values, and information regarding the reaction rates. Finally, it lists out all the reaction pathways used in SAPRC90. The 'soam_ims95a.rxp' file meanwhile contains information on the mechanistic and kinetic parameters for lumped species. Once this information is read in, a third file (soam_rev29c.doc, 'status=unknown') will be created in the run directory that gives a full description of the mechanism including everything that is found in the SAPRC input files.

The process of reading from and writing to the files listed above only needs to be performed once. There are two main subroutines that are utilized to read in and set-up the reaction mechanisms, but only need to be called on the first pass through the mechanism. The first subroutine is *rdmod* which is responsible for reading in the kinetic model and the species parameters that are used within the kinetic model. The other subroutine that is called only on the first pass is named *constr*. Subroutine *constr* calculates the values of the reaction rate constants and variable and constant coefficients. A logical variable was therefore created in 'registry.aqc' named 'saprc_first_call'. It has a default value of 'true' that is switched to 'false' after *constr* to prevent further calling of the initialization routines.

Once the mechanism files are read in, the mechanism goes on to start the reaction calculations. These can be broken down into three main parts: calculation of photolysis rates, updating the temperature-dependent rate constants, and integrating the chemical reaction system. All three of these routines are looped over for each grid box within the Cartesian domain of each MPI process.

The subroutine *citphk* is implemented to compute the photolysis rate constants during the daylight hours. This routine takes in much of the basic species information read in from the MOD file as well as

the cosine of the solar zenith angle, grid square latitude and longitude, and the solar declination angle. Note that *citphk* is only called from *saprc90_driver* when the sun is above the horizon.

The subroutine *newrk* is used to calculate rate constants based on the most recent temperature values for each grid square. The integration of the system over the operator time step is carried out in subroutine *integr2*. This routine makes use of a predictor/corrector scheme to converge on gaseous concentration values (Young and Boris, 1977). Most of the inputs into *integr2* are parameters that control the numerical integration that have been tuned based on past experience with photochemical mechanisms. Among them is the convergence criterion and minimum concentration threshold. The detailed list of the control parameters used for *integr2* can be found within the comments section of that subroutine. Other inputs to *integr2* include the gaseous concentration array, the number of equations to integrate over, and the integration time. In this study, the integration time was set equal to the WRF chemistry time, typically 3-4 minutes. The actual time step used inside *integr2* is variable depending on the convergence of the implicit equation solver as it spans the total integration time.

This study utilized the 1990 version of the SAPRC gas-phase mechanism so that the results could be directly compared to previous air quality modeling studies. There are a few fairly easy steps the user can take to update this mechanism to either the 1999 or 2007 versions if they choose. The user must substitute the appropriate versions of the '.mod' and '.rxp' files found in the simulation run directory as well as the subroutines *constr*, *bldup*, and *difun*. Subroutine *bldp* calculates the formation rates of non-reacting species while *difun* calculates the formation and loss rates for active model species (those that act as either reactants or products). The rest of the mechanism's structure is independent of the mechanism version.

9. Gas-Particle Partitioning in ISORROPIA and Coagulation

Gas-particle interaction is a complex process that consumes a sizeable fraction of the total chemistry-related computing time in the SOWC model. The file 'chem/aerosols_driver' contains subroutine *aerosol_driver*, which is the starting point for all the aerosol packages included in WRF. The SOWC model was introduced through the framework of the Secondary Organic Aerosols Module (SORGAM) and was built-up by replacing the SORGAM subroutines with source-oriented SOWC subroutines. The original SORGAM routines are found in the standard WRF-Chem package. The subroutines used to prep the input for these gas-particle partitioning calculations are found in 'chem/module_aerosols_sorgam', and the thermodynamic calculations are performed in 'chem/module_isrpia' in subroutine *isoropia*. The algorithm employed in *sorgam_driver* pre-screens AQC and AQCG concentrations within each MPI process to determine if concentrations are sufficiently

large to merit gas-particle conversion calculations. Concentrations are copied to a working array and the units are converted from μg m^{-3} to μmoles m^{-3}, which is the standard unit base for aqueous chemistry. Each particle size bin and source type corresponding to diameters larger than 100 nm are copied to the working variable. The SOWC model uses the APDC approach outlined by Jacobson (2005) for gas-particle conversion of inorganic species. In this approach, the ammonium ion is held in equilibrium while the anion concentrations are solved dynamically. Mass and charge balance equations are then used to determine final concentrations of each component. This numerical solution is stable at larger time steps (150-300s as compared to 5-30s), which greatly reduces the computational burden of gas-particle conversion. The vapor pressure of inorganic gases HNO_3, HCl, H_2SO_4, and NH_3 immediately above the particle surface are calculated using the ISORROPIA equilibrium solver (Nenes et al., 1998). The concentration of particulate water is calculated during these steps using the Zdanovskii-Stokes-Robinson (ZSR) approach by the isoropia thermodynamics solver (Stokes and Robinson, 1966). These concentrations are updated for every particle source and size bin at each time step taken. This procedure updates particle water content based on a rigorous thermodynamic calculation that depends on the particle composition. The ZSR method has been widely used in other studies (Clegg and Seinfeld, 2004). The current study resolves the distribution of aerosol water among source-oriented aerosols, which may result in modified optical properties (Beaver et al., 2010) provided that there is a difference in chemical composition between the particles being surveyed (Fuller et al., 1999). This will be discussed further in section 11.

A source-oriented coagulation calculation is performed immediately following the gas-particle exchange calculations. The subroutine *coagrate_ucd* and the subroutines that follow are all originally from the UCD/CIT model (Ying et al., 2008a). The fastest coagulation rates occur between the smallest particles that have high Brownian diffusivity and the largest particles that provide a large target for collisions. The source-oriented algorithm transfers the mass of smaller particles involved in coagulation events to the larger particles, and reduces the number concentration of the smaller particles. The "source-origin" of the larger particles is preserved, at least approximately, since the mass added by coagulation events is generally small relative to the total mass in these size fractions. The integration time in this process matches that of the rest of the chemistry code, which in this test was four (4) minutes. All coagulation routines are located entirely within 'chem/module_aerosols_sorgam' and start out of subroutines *aeroproc* and *coagrate_ucd.*

10. ADVECTION AND DIFFUSION

Mass transport of the scalar arrays is performed within the dynamic core of the WRF model. The SOWC model can only be used with the Eulerian mass-conservation dynamic core (EM_CORE) at present. All of the mass transport and inter-processor communications occur either directly within 'dyn_em/solve_em', or in a routine referenced by *solve_em*. A Runge-Kutta (RK) algorithm is employed within *solve_em* to solve the set of ordinary differential equations pertaining to the tendency (dC/dt) of the scalar arrays. The RK solver allows for first, second, or third-order time integrations choices based on a run-time selection in the 'namelist.input' file. The default value for time integration is third-order. The purpose of this subsection is to describe how the scalar array AQC was introduced into this branch of the WRF code, and to illustrate the flow of information that occurs through *solve_em*. Note that the treatment of the AQC and the AQCG variables in this section of the code is identical.

The mediation level routine *solve_interface* calls *solve_em* every 'time_step' seconds and *chem_driver* every 'chemdt' seconds. Figure A1 illustrates the WRF call structure that controls this time-looping behavior. The main sections of *solve_em* that concern scalar array AQC are the communications between MPI processes and the RK-solver loop. MPI processes communicate in WRF through calls to the HALO routine using the portion of the scalar variable in the 'Memory' grid domain. Each MPI process has a "Memory" grid that overlaps with the "Tile" grid of the neighboring process. Communication with neighboring processes occurs both before and after horizontal transport and the RK loop so that the scalar arrays are updated with the most current information. The file 'Registry/Registry.EM_CHEM' lists all of the variables that are exchanged between neighboring MPI processes, as well as the specific HALO routine that is used for each communication.

The Runge-Kutta loop within solve_em is used to calculate the tendency of each scalar array variable, and then solve the corresponding ordinary differential equation in order to update the scalar value at the next time step. AQC is passed through many different subroutines within the RK loop (see Figure A3). These subroutines can be separated into three groups: tendency calculations, scalar updates, and boundary condition updates.

The tendency calculation routines include all the processes involved in mass transport and solar radiation. Long- and Short-wave radiation calculations, which are classified here as a tendency calculation routine, will be discussed in the following sub-section. Subroutines *first_rk_step_part1*, *first_rk_step_part2*, and *rk_scalar_tend* are the routines responsible for calculating the tendency values for each variable integrated by the WRF dynamic core. These tendency calculation routines were not modified in the SOWC model. The first two subroutines are only referenced during the first time step within the RK solver loop. Subroutine *first_rk_step_part1* initializes the scalar tendencies of AQC with a

call to subroutine *zero_tend*, and then follows this by referencing the short-wave and long-wave radiation modules. Subroutine *first_rk_step_part2* is calculates the first set of scalar horizontal and vertical diffusion tendencies. *rk_scalar_tend* is called during every time-iteration of the RK loop to calculate tendency associated with horizontal and vertical advection and diffusion. All tendencies related to AQC are stored in *aqc_tend*. This variable has the same dimensions as AQC, representing a significant memory burden in the calculation.

The AQC variable (and all scalar variables) are updated in the dynamic core with a call to *rk_update_scalar* and/or *rk_update_scalar_pd*. Both of these subroutines take in the scalar tendency array, aqc_tend, and use it to update a scalar array. *rk_update_scalar_pd* also requires the current value of the scalar array as an input. The AQC example of this input is named AQC_OLD which must be dimensioned identically to AQC, representing yet more memory burden. The *rk_update_scalar_pd* routine updates the values within the AQC_OLD array through use of a positive-definite routine. The subroutine *rk_update_scalar* meanwhile takes in the previous time step scalar array (aqc_old), the tendency array (aqc_tend), and the next time step's scalar array (aqc). *rk_update_scalar* uses the old and tendency values to update AQC in the next time step array with the most current value.

The routines that update boundary conditions are the last main group within the Runge-Kutta integration solver. Depending on the configuration of the grid domain being processed, the AQC scalar array can be processed by several different routines. The main distinction for the domain in question is whether it is the outer-most parent domain, where the boundary conditions are considered to be 'specified'. A nested domain is one that sits inside another, larger domain. **Error! Reference source not found.** illustrates the idea of parent and nested domains with the configuration used in the preliminary simulations for the current study. The domain identifiers for 'parent' and 'nested' are defined by the user at runtime and can be found in the &bdy_control section of the namelist.input file in Appendix Section D: Runtime Configuration.

If the domain under consideration is the parent, specified domain, then the AQC array will be processed through subroutine *flow_dep_bdy_aqc* to have its boundary conditions updated. Note that *flow_dep_bdy_aqc* (as well as *flow_dep_bdy_aqcg* for the AQCG array) were created specifically to support the boundary condition calculations of the SOWC model arrays. If the domain in question is a nested domain then the scalar arrays will be passed into two other subroutines: *relax_bdy_scalar* and *spec_bdy_scalar*. The purpose of these routines is to add tendency values in the boundary relaxation and boundary specified regions. The specified boundary width is a feature of each MPI process domain regardless of whether or not the domain is 'specified' in the sense used above. The specified boundary

width is controlled by the namelist option 'spec_bdy_width' and has a default value of five (5). This means that on the edge of every processor domain, five extra cells are added for boundary condition. The outermost one (1) cell of those five is the 'specified zone' (see namelist variable 'spec_zone'). The next, inside four (4) cells are considered the 'relaxation zone' (see namelist variable 'relax_zone'). These two routines therefore add tendencies to the outermost edges of nested domains.

11 LONG-WAVE AND SHORT-WAVE PHYSICS / AEROSOLS-RADIATION FEEDBACK

The SOWC model currently uses the radiation modules developed by the Goddard Space Flight Center (GSFC). The standard WRF source code comes with the GSFC short-wave radiation module. The current study introduced a comparable GSFC long-wave radiation module to the SOWC model framework (Chen et al., 2010). The radiation routines are called from subroutine *radiation_driver* located in 'phys/module_radiation_driver' which is called from subroutine first_rk_step_part1. The short-wave routine is found within the file 'phys/module_ra_gsfcsw' and the long wave routine is found within 'phys/module_ra_gsfclw'. The radiation scheme is specified at runtime through an entry in the namelist.input. The SOWC model is currently only configured to work properly when the GSFC radiation schemes are selected. The ability to calculate feedback effects of source-oriented aerosol concentrations on the radiation budget is enabled by setting the namelist parameter 'aer_ra_feedback' to 2.

The standard WRF code used a pre-defined concentration profile of internally mixed pollutants for all aerosol optics calculations. A new subroutine was implemented within the SOWC model to calculate layer-averaged optical properties of the size and source resolved aerosols. The new source-oriented routines represent a major improvement in simulating realistic aerosol feedback effects on meteorology. Subroutine *aerosol_opt_ucd* in 'phys/module_ra_gsfcsw' is responsible for calculating the layer-averaged optical properties for both the long-wave and short-wave routines. The refractive index for each particle size and source within each layer is calculated using a core and shell approach. Each refractive index contribution from each source and size is then summed together to give an averaged value for that cell. The refractive index of the non-black carbon components is calculated and then combined into a volume-weighted value. This averaged refractive index is applied as the shell to the particle core. This framework is discussed thoroughly in Stelson (1990) which also provides the values for the individual-component refractive index values used in the current study. Any water that is added to the particle through the gas-particle conversion routines discussed in section 9 also contributes to the refractive index of the shell layer. From this averaged refractive index calculation, values for single scattering albedo, asymmetry parameter, and optical thickness are calculated using Mie scattering theory. These three optical

parameters are then used as inputs with the standard radiation transfer code that is part of the standard WRF model.

References:

Beaver, M.R., Freedman, M.A., Hasenkopf, C.A., Tolbert, M.A., 2010. Cooling Enhancement of Aerosol Particles Due to Surfactant Precipitation. J Phys Chem A 114, 7070-7076.

Chen, S.H., Wang, S.H., Waylonis, M., 2010. Modification of Saharan air layer and environmental shear over the eastern Atlantic Ocean by dust-radiation effects. J. Geophys. Res.-Atmos. 115.

Clegg, S.L., Seinfeld, J.H., 2004. Improvement of the Zdanovskii−Stokes−Robinson Model for Mixtures Containing Solutes of Different Charge Types. The Journal of Physical Chemistry A 108, 1008-1017.

Fuller, K.A., Malm, W.C., Kreidenweis, S.M., 1999. Effects of mixing on extinction by carbonaceous particles. J. Geophys. Res.-Atmos. 104, 15941-15954.

Goodin, W.R., McRa, G.J., Seinfeld, J.H., 1979. A Comparison of Interpolation Methods for Sparse Data: Application to Wind and Concentration Fields. Journal of Applied Meteorology 18, 761-771.

Goodin, W.R., McRae, G.J., Seinfeld, J.H., 1980. An Objective Analysis Technique for Constructing Three-Dimensional Urban-Scale Wind Fields. Journal of Applied Meteorology 19, 98-108.

Jacobson, M.Z., 2005. A solution to the problem of nonequilibrium acid/base gas-particle transfer at long time step. Aerosol Sci. Technol. 39, 92-103.

Kleeman, M.J., Cass, G. R., 2001. A 3d Eulerian source-oriented model for an externally mixed aerosol. Environmental Science and Technology 35, 4834.

Liu, T.-H., Jeng, F.-T., Huang, H.-C., Berge, E., Chang, J.S., 2001. Influences of initial conditions and boundary conditions on regional and urban scale Eulerian air quality transport model simulations. Chemosphere - Global Change Science 3, 175-183.

Mahmud, A., Hixson, M., Hu, J., Zhao, Z., Chen, S.H., Kleeman, M.J., 2010. Climate impact on airborne particulate matter concentrations in California using seven year analysis periods. Atmos. Chem. Phys. 10, 11097-11114.

Michalakes, J., Schaffer, D., 2004. WRF Tiger Team Documentation.

Nenes, A., Pandis, S., Pilinis, C., 1998. ISORROPIA: A New Thermodynamic Equilibrium Model for Multiphase Multicomponent Inorganic Aerosols. Aquatic Geochemistry 4, 123-152.

Stelson, A.W., 1990. Urban aerosol refractive index prediction by partial molar refraction approach. Environmental science & technology 24, 1676-1679.

Stokes, R.H., Robinson, R.A., 1966. Interactions in Aqueous Nonelectrolyte Solutions. I. Solute-Solvent Equilibria. The Journal of Physical Chemistry 70, 2126-2131.

Ying, Q., Lu, J., Allen, P., Livingstone, P., Kaduwela, A., Kleeman, M.J., 2008a. Modeling air quality during the California Regional PM10/PM2.5 Air Quality Study (CRPAQS) using the UCD/CIT source-oriented air quality model – Part I. Base case model results. Atmospheric Environment 42, 8954-8966.

Ying, Q., Lu, J., Kaduwela, A., Kleeman, M., 2008b. Modeling air quality during the California Regional PM10/PM2.5 Air Quality Study (CPRAQS) using the UCD/CIT Source Oriented Air Quality Model - Part II. Regional source apportionment of primary airborne particulate matter. Atmospheric Environment 42, 8967-8978.

Ying, Q., Lu, J., Kleeman, M.J., 2009. Modeling air quality during the California Regional PM10/PM2.5 Air Quality Study (CRPAQS) using the UCD/CIT sourceoriented air quality model – part III. Regional source apportionment of secondary and total airborne particulate matter. Atmospheric Environment 43, 419-430.

Young, T.R., Boris, J.P., 1977. A numerical technique for solving stiff ordinary differential equations associated with the chemical kinetics of reactive-flow problems. The Journal of Physical Chemistry 81, 2424-2427.

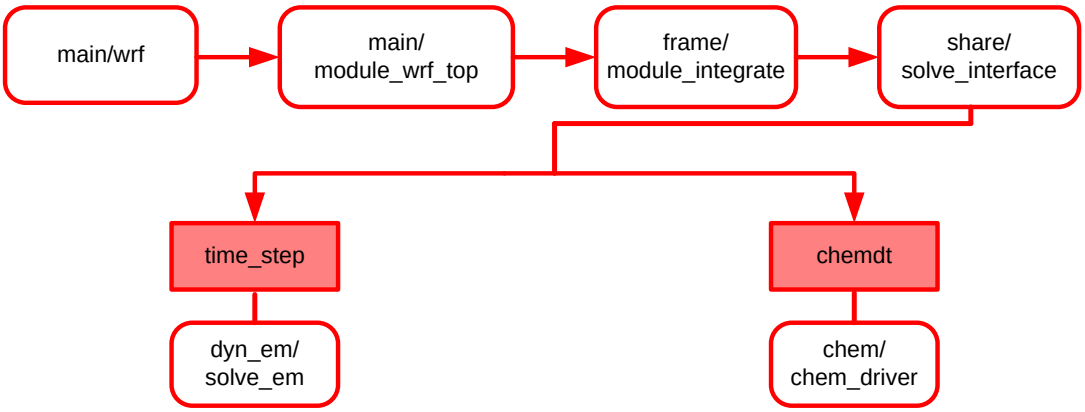

Figure A1. Schematic of WRF's overall time-looping structure, showing the alternation between dynamic-core calculations and the chemistry-core calculations. At every time step advance of variable length *time_step* the WRF model (and therefore SOWC model) will call the dynamic Eulerian mass conservation core. With ever time step advance of variable length *chemdt*, the WRF-Chem (and SOWC) model will call the chemistry driver.

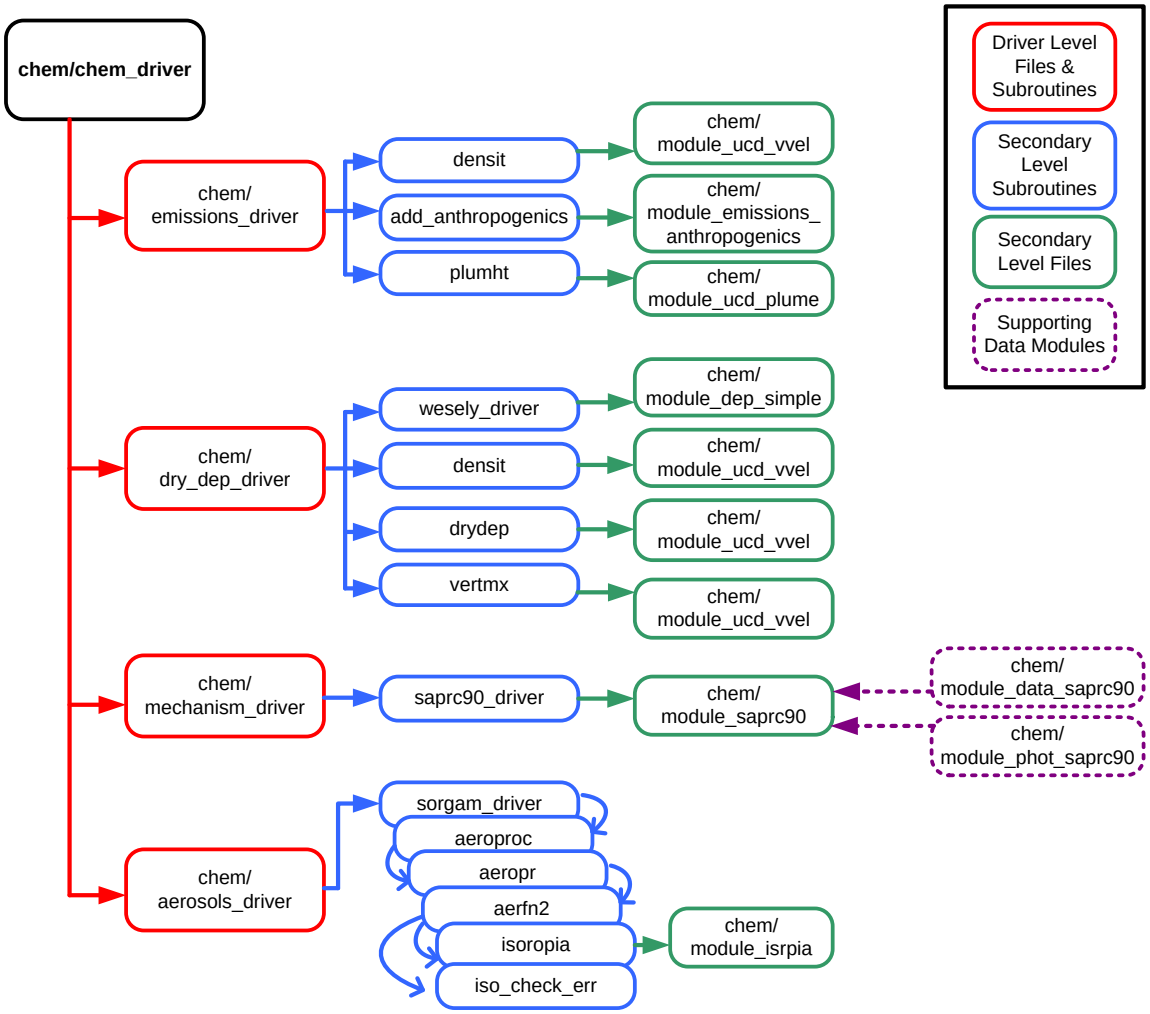

Figure A2. Schematic showing the flow of information within the SOWC model's chemistry driver. All files, routine calls, and modules listed here were altered or newly added to the WRF framework to build the SOWC model just within the chem driver section of the original WRF code.

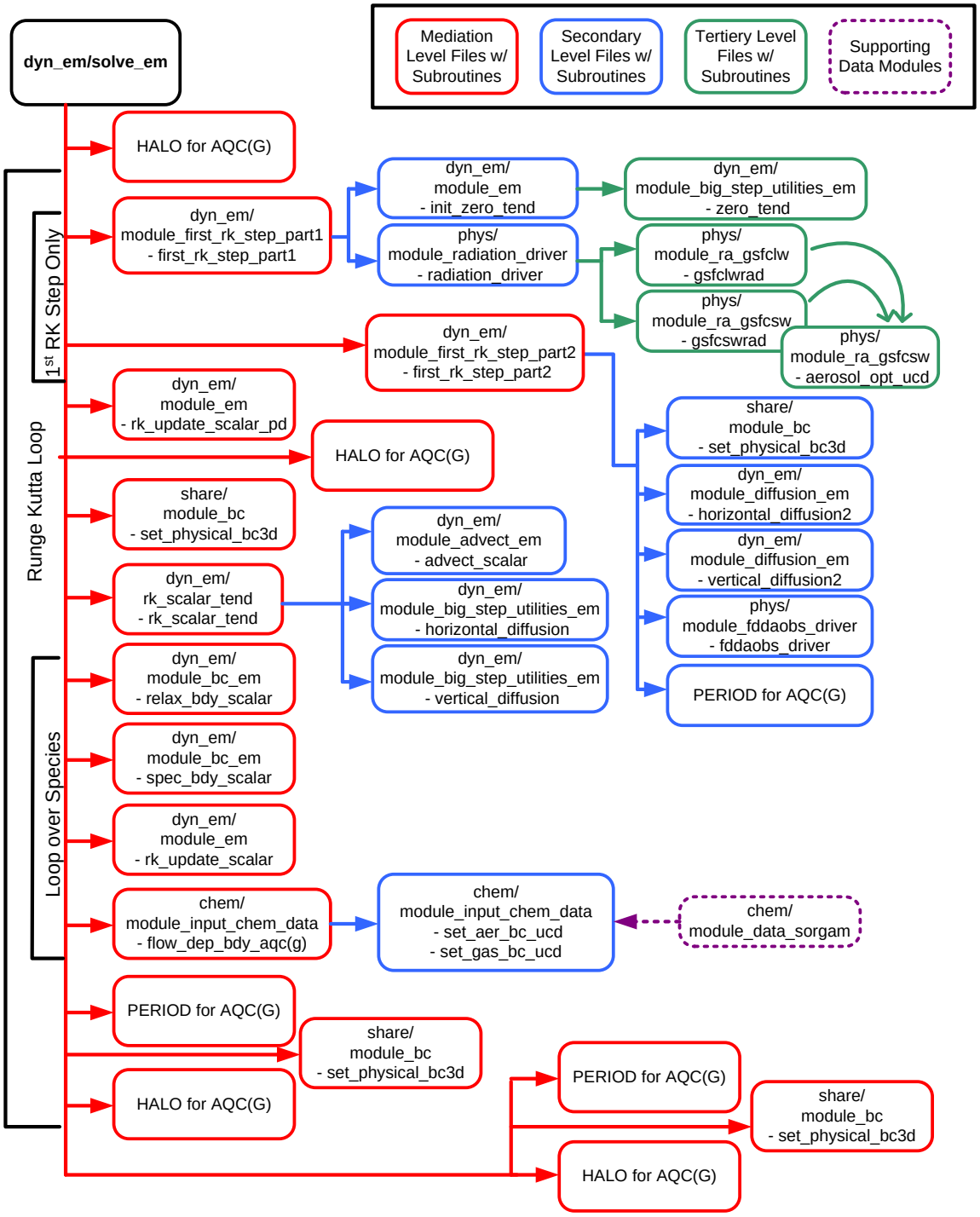

Figure A3. Schematic showing the flow of information within the SOWC model's dynamic-Eulerian mass conservation core. All files, routine calls, and modules listed here were altered or newly added to the WRF framework to build the SOWC model just within the dynamic driver section of the original WRF code.