Overview of the Framework for 0-D Atmospheric Modeling (F0AM)

Version 4.3.1
Last updated 11/3/2023.
Important changes from the previous version are highlighted.
User group mailing list: F0AMusers@googlegroups.com

Table of Contents

1. IMPORTANT USAGE INFORMATION

If you use the model for a publication, please do the following:

1. Cite the model description paper: G. M. Wolfe, M. M. Marvin, S. J. Roberts, K. R. Travis, and J. Liao, The Framework for 0-D Atmospheric Modeling (F0AM) v3.1, Geoscientific Model Development, doi: 10.5194/gmd-2016-175, 2016.
2. Cite appropriate references for any chemical mechanisms used.
3. Add the paper to our list: https://docs.google.com/spreadsheets/d/1fd7mWTzMiWuuqRG9eI9g0iYyt7DymLlVpDv_rArt5Z8/edit?usp=sharing.

The Framework for 0-D Atmospheric Modeling (F0AM, yes that's a zero) is exactly what its name implies: a flexible software interface for simulating chemical systems relevant to atmospheric composition. Let's break it down:

0-D: The model simulates processes at a single point in space. You can think of this point as a uniform box, if you prefer. It does NOT explicitly simulate transport or mixing processes. Users should be cognizant of the inherent limitations of a 0-D framework. Recommended reading includes Chapters 3 and 5 of Jacob's Atmospheric Chemistry.

Atmospheric Modeling: The user specifies a set of initial conditions (chemical concentrations and meteorology) and a chemical mechanism. The model then predicts how concentrations evolve over time. The results that come out of the model are output. This is an important distinction – model output is effectively an educated guess. The term "data" should arguably refer to observations of natural phenomena.

Framework: The model accommodates a variety of typical problems, including photochemical chambers and field observations from ground and aircraft. The provided examples show typical setups. It also includes multiple chemical mechanisms.

This model evolved out of the CAFE 1-D canopy model, developed by Glenn Wolfe during his Ph.D. work in Joel Thornton's lab at the University of Washington. It was originally called the University of Washington Chemical Model (UWCM). Since 2011, the model has undergone heavy modifications. Recognizing that the apple has rolled far from the tree, the model became F0AM in 2016.

Users will need some familiarity with MATLAB to use the model effectively. This document describes the overall structure and implementation of the code. A short presentation, F0AM_GettingStarted.pdf, is also included with this readme.

Questions or comments should be direct to the user group mailing list (F0AMusers@googlegroups.com). This is a community tool, so if you produce any code that others might find useful, please share it via GitHub or correspondence with Glenn.

2. GENERAL OVERVIEW

The following subdirectories are included in the main folder. All folders and subfolders must be added to the MATLAB search path. If you are unfamiliar with how to do this, go to the MATLAB help window and search for "changing the search path."

Setups: Model setup scripts, including the examples.
Tools: Scripts for manipulating UWCM output.
Plots: Scripts for plotting model output.
Chem: Chemical mechanisms, photolysis code
Docs: Documentation and tutorials
Runs: Default folder for saving model output in dated sub-directories
Core: Core scripts and functions for the model.

Model runs are executed with the following function call:

```
S = F0AM_ModelCore(Met,InitConc,ChemFiles,BkgdConc,ModelOptions,SolarParam);
```

INPUTS:

Met: Meteorological variables (2-column cell array)
InitConc: Initial chemical concentrations (3-column cell array)
ChemFiles: Names of all chemistry sub-mechanisms (cell array)
BkgdConc: Background chemical concentrations for dilution (2-column cell array)
ModelOptions: Parameters for model execution and output (structure array)
SolarParam: (optional) Parameters for running in solar cycle mode (structure array)

OUTPUT is given as a structure, S, described in Model Output.

Input and output variables are described in detail below. New users are encouraged to look at the example setups (under \Setups\Examples), which include a range of typical model uses.

ExampleSetup_Chamber
ExampleSetup_LagrangianPlume
ExampleSetup_DielCycle
ExampleSetup_FlightSS
ExampleSetup_MechCompare

NOTE ON TERMINOLOGY: A model *run* refers to a single model call, while a model *step* refers to model execution for a single set of initial meteorological and chemical conditions. There can be multiple *steps* within a *run*.

NOTE ON INPUT REPLICATION: Most of the inputs in Met and InitConc can be specified as either a scalar or a 1-D column array. All variables specified as arrays must be of the same length; this length determines the number of model steps. Any variables specified as scalars are assumed constant for all model steps.

3. METEOROLOGY

Met is a 2-column cell array containing all meteorological inputs. The first column contains variable names, while the second column contains values. The user must specify pressure, temperature, and either H2O or RH. Other variables are optional and setup-dependent.

All Met variables are available for building up reaction mechanisms. Users can add new variables by specifying them in the **InitializeMet** function, located in the \Core directory.

Basic Meteorology

P: Pressure (mbar). Required input.

T: Temperature (K). Required input.

H2O: Water vapor number density (molec cm^{-3}). Takes priority over RH if both specified.

RH: Relative humidity (%).
NOTE: water vapor is a required input (either RH or H2O).

Dilution (also see Dilution)

kdil: First-order rate constant for dilution (s^{-1}). *DEFAULT: 0.*

tgauss: Gaussian dispersion timescale (s). *DEFAULT: Inf.*

Radiation-Related (also see Photolysis Options)

SZA: Solar zenith angle (0 – 90 degrees). Not required if **LFlux** or **SolarParam** are specified. *DEFAULT: 0 degrees.*

LFlux: Name of a text file containing a radiation spectrum (actinic flux vs wavelength). Only needed if you wish to calculate J-values with this spectrum. The text file should have no headers and two columns: wavelength (nm) and photon flux (photons/cm^2/s/nm). See \Setups\Examples\ExampleLightFlux.txt for an example. *DEFAULT: empty*.

J[n]: Measured J-values (s^{-1}), used to overwrite the default parameterized values. The variable name is specific to the utilized chemistry scheme (e.g., the NO2 photolysis frequency is **J4** in MCM and **JNO2** in CB05). Use one row for each variable.

jcorr: Correction factor used to scale all J-values that are not explicitly input. This can be
1) a numeric scalar or array with values >0
2) a string specifying the name of an input J-value, e,g. 'J4'
3) a cell array of strings of multiple input J-values, e.g. {'J4','J1'}
In the second case, a correction factor will be calculated as the observed/calculated J-value ratio. The same is true in the third case, except jcorr is the average of correction factors for all specified inputs. *DEFAULT: 1.*

ALT: Altitude, m. Identical to (though separate from) the "alt" input field in SolarParam. Only used if the "HYBRID" J-value method is selected. *DEFAULT: 500 m.*

O3col: Overhead ozone column, DU. Only used if "HYBRID" J-value method is selected. *DEFAULT*: 300 DU (typical of mid-latitudes).

albedo: Surface reflectance, unitless (range 0-1). Only used if "HYBRID" J-value method selected. *DEFAULT*: 0.1 (typical of vegetated surfaces).

Emissions/Deposition (also see Emissions and Deposition)

BLH: Boundary layer depth, m. *DEFAULT: 1000 m.*
PPFD: photosynthetic photon flux density, umol/m^2/s. *DEFAULT: 0 umol/m^2/s.*
LAI: leaf area index, m^2/m^2. *DEFAULT: 0.*

Aerosol (also see Heterogeneous Chemistry)

pH: Liquid drop pH. *DEFAULT: 7.*
rpaerosol: mean aerosol radius, cm
Naerosol: organic number density, #/cm^3
Saerosol: organic surface area density, cm^2/cm^3
Vaerosol: organic volume density, cm^3/cm^3
rpice, etc.: As above, but for ice particles
rpaqueous, etc.: As above, but for liquid drops

Useful functions (located in \Tools\):

ConvertHumidity: Converts between standard representations of atmospheric water vapor content.
sun_position: Calculates SZA based on date, time and location coordinates.
ReplaceNaN: Replaces NaNs in a matrix with linear interpolations (between rows). This is handy if your observational data has holes but should be used with due caution.
ScaleData: Linearly scales a vector to a new range. Useful if, for example, you want a dilution rate constant that is scaled to wind speed or boundary layer height.

4. CHEMICAL CONCENTRATIONS

4.1 THE INITCONC INPUT

InitConc is a 3-column cell array containing information for all initial concentrations. All species not specified here will have a starting concentration of 0.

First column: Species names. These must be the same as those found in the chemical mechanisms.

Second column: Chemical mixing ratios in parts per billion (ppb) or family definition (see below).

Third column: This is a scalar flag, **HoldMe,** specifying how constraints are handled for each model step.

1 – Hold constant throughout model step.
0 – Initialize but do not hold constant. Behavior depends on value of **ModelOption.LinkSteps**:
LinkSteps = 0: Initialize at beginning of each model step.
LinkSteps = 1: Initialize at beginning of first step only.

Useful functions (located in \Tools\)

NumberDensity: Calculates atmospheric number density at a given T and P. Useful for converting between concentration (molec/cm^3) and mixing ratio.

ReplaceNaN: replaces NaNs in a matrix with linear interpolations (between rows). This is handy if your observational data has holes but should be used with caution. Gap-filling is an art, and this is a relatively coarse fix.

DataCleaner: Like ReplaceNaN but on steroids. Provides more options for filling in gaps (nans and/or negatives) with interpolation, mean, median, etc.

4.2 FAMILY CONSERVATION

NOTE: This functionality supersedes the FixNOx option, though both remain available. The two are not compatible.

ANOTHER NOTE: This code is still in development and seems to work fine for NOx but not for larger families (e.g. Bry). If it breaks on you, try changing the order of the family member cell array. Sorry . . . this is a hard problem.

It may be desirable to hold the sum of a family (or class) of species constant while allowing individual species within that family to evolve freely. In **ExampleSetup_FlightSS.m**, for example, NO and NO2 are initialized by observations, but their balance must respond to the solar cycle so they cannot realistically be held constant (or at least NO cannot). Without NOx sources (e.g. emissions), NOx would decay over the course of the step. In other cases it might be appropriate to constrain total NOy, total Cl_y, etc. Families can be defined in InitConc as follows.

First column: Family name.

Second column: Cell array of family members. This can also include multipliers for species. Multipliers must be at the beginning of the string and must be separated from the species name with an asterisk (e.g., '2*N2O5').

Third column: empty array, [].

In addition, families must meet the following criteria:

1. at least one member of each family must be initialized in InitConc.
2. Any initialized family members must have HoldMe = 0.

An example is shown below. Implementation of family conservation is based on the "mass matrix" option embedded within the ode15s solver. It has been tested within ExampleSetup_DielCycle, but it is still experimental and may lead to "unexpected" behavior for some situations. In particular, testing has shown that model output is sometimes sensitive to the order in which family members are specified: {'NO','NO2'} may give a slightly different output than {'NO2','NO'}. This is a consequence of how the ODE solver handles this type of problem (using a "mass matrix"). Read up on ode15s options if you want to know the gory details. Use with caution.

```
InitConc = {...
            'NO'        0.1                          0
            'NO2'       0.4                          0
            'NOy'       {'NO','NO2','NO3','2*N2O5'}  []
           };
```

5. CHEMISTRY

5.1 THE CHEMFILES INPUT

ChemFiles is a cell array of strings specifying functions and scripts for the chemical mechanism. The first and second cells are always functions for generic/complex rate constants and J-values, respectively. Subsequent cells are scripts for mechanisms and sub-mechanisms. For an MCM scheme, the input might look as follows:

```
ChemFiles = {...
             'MCMv331_K(Met)';...
             'MCMv331_J(Met,0)';...
             'MCMv331_Inorg_Isoprene'};
```

The output of the K and J functions must be a structure containing all calculated rate constants/J values. If, for some bizarre reason, your mechanism does not have functions for K's and J's, either or both of the first two cells can be empty. Available mechanisms are listed below.

5.2 STRUCTURE OF INDIVIDUAL CHEMISTRY SCRIPTS

Each chemistry script has two main sections: species names and chemical reactions.

Species Names

This section is where all chemical species and RO2 names are specified. These are given as cell arrays of strings and added to the relevant lists by calling the script **AddSpecies**. For example, for a mechanism involving oxidation of methane:

```
SpeciesToAdd = {'CH4'; 'CH3O2'; 'CH3OOH'; 'HCHO'; 'OH'; 'HO2'; 'NO'; 'NO2'};
RO2ToAdd = {'CH3O2'};
AddSpecies
```

A few notes on this:

- **RO2ToAdd** is optional and only necessary for mechanisms that use total peroxy radicals as an operator (like MCM). In such a case, all RO2 species must be included in *both* the **CnamesToAdd** and **RO2ToAdd** cell arrays.
- Generally, it is good practice to give the names of *all* reactants and products included in the sub-mechanism. However, this section can be omitted if the mechanism will only be used in conjunction with another mechanism that includes all reactant/products.
- The same species can be added in multiple sub-mechanisms without fear of generating duplicate species. This is not the case with reactions.

Chemical Reactions

This section contains blocks of code for chemical reactions. Each block has the following structure.

```
i=i+1;
Rnames{i} = 'CH4 + OH = CH3O2';
```

```
k(:,i) = 1.85e-12.*exp(-1690./T);
Gstr{i,1} = 'CH4'; Gstr{i,2} = 'OH';
fCH4(i) = -1; fOH(i) = -1; fCH3O2(i) = 1;
```

Rnames: A string specifying the name of the reaction

k: Reaction rate constant. Note that this will be a column vector if multiple initial conditions are given, hence the use of vectorized operators (.*, ./ and .^).

Gstr: 3-column cell array of strings specifying reactant names. One or more of the Gstr columns can be left blank; if the reaction is 0^{th}-order, like emissions, you need not specify any value here. Yes, it is short for "G-string." Giggle if you must.

fX: Stoichiometric coefficients for each species for a given reaction. In the above example, one molecule each of CH4 and OH are lost and 1 molecule of CH3O2 is formed, thus the respective fX are -1, -1 and 1. fX for all species not participating in the reaction are 0 by default.

Limiting Reagent Reactions

This is a special feature that is currently only used in the "HalogenAerosol_Sherwen2016" MCMv331 sub-mechanism to approximate aqueous phase chemistry. For such reactions, the rate constant is 1^{st}-order and the instantaneous reaction rate is determined by k*min(conc_reactant_1, conc_reactant_2). If you want a reaction treated this way, add the line "lr_flag(i) = 1" to your reaction block. Note, these reactions currently only support 2 reactants.

Additional Notes

- ORDER: The order in which reactions are entered does not matter.
- DUPLICATE REACTIONS: If duplicate reactions are found, a warning will appear in the command window during model initialization, but the model will still run. These do occur in the MCM, typically involving two separate but numerically-identical photolysis reactions. In general, however, duplicate reactions should not be present.
- All **Met** variables and generic rate constants/photolysis frequencies (the latter specified in the first two ChemFiles input functions) can be used when defining reaction rates constants. Addition of Met variables beyond those listed in Meteorology requires first adding them to the list in the **InitializeMet.m** function.

5.3 GENERATING MCM REACTIONS FILE

The full version of the MCM, as well as a few subsets, are included with the examples under \Chem\MCMv331\. In many cases, however, a user will only have constraints for a subset of all VOC. For computational efficiency, it is recommended that users generate their own MCM subsets in these cases. Here's how.

1) Go to the MCM website, https://mcm.york.ac.uk/MCM/
2) Near the top, click "Browse."
3) Click through the categories to find species you want.
4) Click the green "+" to include species. Species can be removed by clicking the red "–" here or in the Mark List (upper right basket icon). You can also add species using the search tools.
5) Near the top, click "Export."

6) Select “FACSIMILE input format” and “Include inorganic reactions.” Unselect “include generic rate coefficients.”
7) Click “Download” to download to a text file (default name is mcm_export.fac)
8) Give this file a more descriptive name and move it to somewhere on your MATLAB search path (e.g., F0\Chem\MCMv331\alkanes.fac).
9) In the MATLAB command window, call the **FAC2F0AM** function:

```
FAC2F0AM(MCM_flnm, save_flnm)
```

Here, **MCM_flnm** is the name of the FACSIMILE text file (including extension) and **save_flnm** is the desired name for the script that will be written. This will generate the sub-mechanism as a script (.m file) in the same directory as the text file. **FAC2F0AM** has been tested with the entire MCM reaction set, but it may fail for other FACSIMILE-formatted mechanisms. Some translation code for KPP-formatted mechanisms is also available on request, though it may need some tinkering.

Do not use multiple MCM-extracted mechanisms simultaneously! This will lead to duplicate reactions.

5.4 MODIFYING MCM REACTIONS

Sometimes, it may be necessary to modify the rate constant or yield of a reaction in the MCM mechanism. This can be done in the mechanism script; however, it is highly recommended that users save a separate script—with a different name—if any modifications are made to the base MCM mechanism. This will reduce confusion and errors when performing multiple model experiments.

Another option is to apply the correction in a separate sub-mechanism that appears in **ChemFiles** after the MCM sub-mechanism. For example, the following script updates the branching for the MACRO2 + NO reaction in MCMv3.2:

```
i=i+1;
Rnames{i} = 'MACRO2 + NO =  MACRNO3';
k(:,i) = KRO2NO.*0.15;
Gstr{i,1} = 'MACRO2'; Gstr{i,2} = 'NO';
fMACRO2(i)=-1; fNO(i)=-1; fMACRNO3(i)=1;

RxnToReplace = 'MACRO2 + NO =  + ACETOL + CO + HO2 + NO2';
kToReplace = KRO2NO.*0.85;
ReplaceRxn
```

The first section contains a new reaction. The second section adjusts the yield of the default MCM reaction from 1 to 0.85 by altering the rate constant. **RxnToReplace** is the name of the reaction to be fixed (which you can find in the MCM sub-mechanism), and **kToReplace** is the new rate constant. Calling the script **ReplaceRxn** then applies the correction. This script is currently not capable of replacing fX or Gstr values, but could be modified to do so if necessary.

5.5 AVAILABLE MECHANISMS

The below table lists currently-available chemical mechanisms and sub-mechanisms, which can be found in the \Chem\ folder. These folders also contain relevant documentation. If users create more such mechanisms in the course of their work, they are encouraged to share with the community. Text readers are available for creating F0AM mechanism scripts from native mechanism text files for SAPRC (MEC2F0AM.m) and GEOS-CHEM (EQN2F0AM.m). These are included in their respective \Chem\ folders. They may or may not work for versions other than those on which they were tested.

NOTE OF CAUTION regarding condensed mechanisms and photolysis. Some mechanisms specify cross sections and quantum yields, but these can be outdated relative to current panel recommendations and/or not well documented. The mechanisms as included in F0AM do not use mechanism-specific photolysis data, but instead map all photolysis frequencies to a common spectral database (see **PhotoDataSources.xlsx** and further discussion below). So, when comparing F0AM output to another model (e.g., CMAQ, CAMX, etc.), users should consider that J-values may not be parameterized in the same fashion even though the mechanisms are otherwise the same. Whether or not J-values parameterizations are "part" of a mechanism is an area of philosophical debate.

Mechanism	Chemistry	Generic K function	J-value function(s)
MCMv3.3.1	MCMv331_AllRxns MCMv331_Inorg_Isoprene MCMv331_Methane custom subsets (see above) *Sub-mechanisms* CH4_O1D CH3ONO_hv HO2NO2_hv CH3O2_OH C3H4_OH HMHP_OH Slow_PAA_OH Cl_VOC_Riedel2014 Halogens_MECCA MTSQT_Wolfe2011 Halogens_Sherwen2016 HalogenAerosol_Sherwen2016 MCMv331_AllRxns_NOAABB[1]	MCMv331_K(Met)	MCMv331_J(Met, Jmethod)
MCMv3.2	MCMv32_Inorg_Isoprene custom subsets (see above)	MCMv32_K(Met)	MCMv32_J(Met,Jmethod)
CB05	CB05_AllRxns	CB05_K(Met)	CB05_J(Met, Jmethod)

CB6r2	CB6r2_AllRxns	CB6r2_K(Met)	CB6r2_J(Met, Jmethod)
RACM2	RACM2_AllRxns	RACM2_K(Met)	RACM2_J(Met, Jmethod)
GEOS-CHEM	GEOSCHEMv902_AllRxns GEOSCHEMv1207_AllRxns	GEOSCHEM_K(Met)	GEOSCHEM_J(Met, Jmethod)
SAPRC07B	SAPRC07B_AllRxns	SAPRC07_K(Met)	SAPRC07_J(Met,Jmethod)
CRACMMv1.0	CRACMM1_aq_AllRxns	CRACMM1_aq_K(Met)	CRACMM1_aq_J(Met,Jmethod)

[1]Full MCMv331 mechanism with modifications/additions.

5.6 PHOTOLYSIS OPTIONS

Several options are available for calculating J-values. All options are contained in the J-value functions of each mechanism and are selected with the "Jmethod" input. Jmethod can be a string or scalar, and options are 'MCM', 'BOTTOMUP', OR 'HYBRID' (0, 1, or 2 if given as a scalar). The default is 'MCM'.

It is always highly preferable to scale model-calculated J-values to an observed J-value using the jcorr input. The radiation models underlying the MCM and HYBRID methods represent "typical" tropospheric conditions but do not reflect variability in overhead ozone column, surface albedo, aerosol optical depth, clouds, solar eclipses, really big birds, etc., all of which affect the radiation field. In short: *do not trust an unconstrained parameterization to give an accurate J-value estimate.*

MCM

This is the trigonometric SZA function found in MCM. The actual function is

$$J = I * \cos(SZA)^m * \exp(-n * \sec(SZA))$$

Here, I/m/n are constants derived from least-squares fits to J-values derived from a radiative transfer model run at 0.5 km and literature cross sections/quantum yields. The origin of the parameterization is discussed in Jenkin et al. (1997). As of MCMv3.3.1, there seems to be substantial differences between this parameterization and values calculated from the NCAR TUV radiation model. See \Chem\Photolysis\PhotoDataSources.xlsx for a comparison. Some mechanisms include photolysis reactions that are not found in MCM. For such reactions, HYBRID values are used instead, with a fixed altitude of 0.5km to match the MCM assumption, and O3 column of 350 DU and albedo of 0.01 to optimize agreement between MCM and HYBRID J-values.

BOTTOMUP

J-values are calculated from scratch by integrating a user-specified actinic flux spectrum (as specified in the **Met.LFlux** input) and literature-derived cross sections and quantum yields. Cross sections and quantum yields are contained in \Chem\Photolysis\, and the spreadsheet **PhotoDataSources.xlsx** documents their origin, last update, and translations into various mechanisms.

HYBRID

J-values are calculated as a function of SZA, altitude, overhead ozone column and albedo using lookup tables. The lookup tables are calculated using solar spectra from the NCAR TUV v5.2 radiation model (available online) and the same literature cross sections/quantum yields used in the BOTTOMUP method. TUV setup included the following parameters:

Parameter	Range or Value	Units
SZA	0:5:90	Degrees
Altitude	0:1:15	Km
Ozone column	100:50:600	DU
Albedo	0:0.2:1	None
Ground altitude	0	Km
AOD	0.235	None
T, P	US Standard Atmosphere	

This method was designed as a compromise between the MCM parameterization (which is incomplete when paired with other mechanisms and optimized for surface conditions) and running the full TUV model inline (which is computationally expensive, and not easily modified). The **PhotoDataSources.xlsx** spreadsheet documents sources for all photolytic data and also shows a comparison of the HYBRID output against both TUVv5.2 and MCM. Most of the differences are due to choices: JPL vs. IUPAC recommendations, wavelength ranges, etc. Users are encouraged to verify data on reactions relevant to their work.

5.7 GENERATING HYBRID LOOKUP TABLES

Adding or modifying J-values for the hybrid method requires regenerating the whole set of lookup tables. Steps for doing so are provided here. All files referenced below are found in \Chem\Photolysis\.

1) Acquire any necessary cross section and quantum yield (CS/QY) data and put it in the appropriate folders in \Chem\Photolysis\. These can be functions (.m) or comma-delimited tables (.csv). See comments within **IntegrateJ.m** for further details on format. DO NOT overwrite files already present; if you are updating CS/QY, give your update a new filename.
2) Modify **J_BottomUp.m** as needed. If updating CS/QY, comment out the old lines rather than deleting. If creating new J-values, DO NOT use the "Jn##" naming scheme found in this function, as this could create conflicts in future official F0AM updates.
3) Rename **HybridJtables.mat** to **HybridJtables_old.mat**.
4) Test your updates by calling **calc_HybridJtables.m** with a test_flag of 1:

```
calc_HybridJtables(1)
```

This will evaluate J-values for a representative solar cycle and generate an inspection plot for all J-values vs SZA. Sanity-check both the shape and the magnitude of max values. Squash bugs as needed. If you suspect an issue with CS/QY values, you can plot these using **IntegrateJ.m**.

5) Call **calc_HybridJtables.m** with a test_flag of 0. This will generate a new **HybridJtables.mat** file. Note, this function can take hours to complete. If you have the parallel computing toolbox, you can speed this up by setting the par_flag input to 1.
6) If you are using a mechanism other than MCM, update the relevant J function (e.g., **GEOSCHEM_J.m**) to make the J-values available within your mechanism.
7) Document any changes in **J_BottomUp.m** and **PhotoDataSources.xlsx**. Don't skip this step! It may seem tedious but it will help you in the future, when your mind starts to rot.

If users need to derive J-values for conditions outside those listed in the above table, they will need to generate new TUV solar spectra. Code for doing so is available upon request.

5.8 HETEROGENEOUS CHEMISTRY

Currently, none of the mechanisms in F0AM include heterogeneous chemistry. There are a few examples showing how this could be done in the \Chem\Aerosol folder.

The Thornton group at UW has developed an aerosol module specifically for isoprene aerosol growth, which can be found at https://www.atmos.washington.edu/~thornton/washington-aerosol-module and is described in D'Ambro et al. (2017). Note that this code branches from F0AMv3.1 and may not include the same features or functionality as newer versions of F0AM. WAM will be merged into F0AM as a module in a future release.

5.9 EMISSIONS AND DEPOSITION

Theoretically, emissions and (dry) deposition can be treated just like chemistry in a 0-D box model: emissions as a 0^{th}-order source and deposition as a 1^{st}-order sink. There are challenges to doing this.

- For emissions, one must assume instantaneous dilution into the whole box, which may or may not be fair depending on the situation.
- For deposition, experimental constraints on deposition velocities (Vd) are limited. It is non-trivial to constrain or predict Vd for lots of species (e.g. in the MCM).
- Both require knowledge of the mixing height and surface characteristics (plant functional type, LAI, surface wetness, etc).

A few crude examples are included in the \Chem\Emission and \Chem\Deposition folders; use or modify these at your own peril. More advanced formulations are available outside of F0AM. Thomas Karl provides MATLAB code (including GUIs) for canopy resistances and MEGAN v2.1 isoprene emissions at http://homepage.uibk.ac.at/~c7071028/. Jennifer Kaiser (Kaiser et al., 2016) has also developed some code to calculate deposition velocities for select MCM species (in a SE US forest) using SMILES strings; please contact Glenn if you would like to see this code.

5.10 INTEGRATION OF CHEMICAL EQUATIONS

ModelOptions.IntTime specifies the length of time to integrate each model step. The model uses MATLAB's ode15s solver, which is specifically designed for stiff systems. Initial concentrations for each step are set to 0 molec cm^{-3} unless otherwise specified in **InitConc**. To see how the chemical rates are evaluated, the enterprising user is invited to look at **IntegrateStep** and **dydt_eval** in \Core\. In a nutshell:

1) The index **iG** (which is generated using **Gstr**) is used to calculate the matrix **G**, which is the product of reactant concentrations for each reaction;

2) Multiplication of **G** by rate constants, **k**, gives the rate for each reaction;

3) Multiplication of the rates by **f** gives the net rate of change for each species. This last line is a matrix multiplication, so it is actually a two-step process: multiplication of each rate by the stoichiometric coefficients, and summation of these weighted rates across all reactions for each species.

In mathematical terms, for any species X,

$$\frac{d[X]}{dt} = \sum_{i=1}^{\#\,Rxns} f_i^X k_i G_i$$

Several special cases are also handled after calculating rates.

- Dilution, if used, is added simultaneously for all species.
- For species with HoldMe = 1, d[X]/dt is set to 0.
- For conserved families (with members X_i), d[X]/dt for the first member is replaced with a conservations law: 0 = sum(X_i) – (Initial family concentration).

F0AM also uses several ODE options.

- The Jacobian is calculated explicitly within **Jac_eval.m**. This speeds integration substantially.
- A mass matrix is calculated within **Mass_eval.m**. This is needed for family conservation but is an identify matrix otherwise.

Advanced users may need to change code within the core to implement custom reaction classes that are not handled within F0AM. If you have to do so, you should read the ode15s documentation and be aware that changes to **dydt_eval.m** are almost always accompanied by changes to **Jac_eval.m**, and possibly **Mass_eval.m**.

6. DILUTION

6.1 SIMPLE 1ST-ORDER

Dilution can be parameterized following the simple functional form

$$\frac{d[X]}{dt} = -k_{dil}([X] - [X]_b)$$

Where k_{dil} is a 1st-order dilution rate coefficient and $[X]_b$ is a fixed background concentration. Note that this is equivalent to a 0th-order source ($k_{dil}[X]_b$) and a 1st-order sink ($-k_{dil}[X]$). More information on this parameterization, including possible methods of determining k_{dil}, can be found elsewhere (Dillon et al., 2002; Wolfe and Thornton, 2011).

kdil is specified as a parameter in **Met**. Setting this to 0 will negate dilution (unless **tgauss** is specified, see below).

BkgdConc is a 2-column cell array that determines background concentrations. The first column gives species names and the second give values, analogous to **InitConc**. The first row must contain the name 'DEFAULT' and a value of 0 or 1, which determines the default concentration for non-specified species. Setting this to 0 assumes concentrations of 0, while setting it to 1 assumes background concentrations equal to those found in **InitConc** (and 0 for those not in **InitConc**).

Many 0-D box model simulations include an additional 24-hour lifetime for all species to keep secondary species from building up to unreasonable levels. This can be achieved here by setting **kdil** = 1/86400s and setting the DEFAULT in **BkgdConc** to 0.

This scheme is a dramatic simplification of a complex physical process, and effectively encompasses all physical sinks (e.g. deposition, entrainment, etc).

6.2 GAUSSIAN DISPERSION

Another option for dilution is Gaussian dispersion, which is typically used for modeling of discrete plumes (e.g. fires or power plants). In this case, the equation is

$$\frac{d[X]}{dt} = \left(\frac{-4K_y}{y_0^2 + 8K_y t}\right)([X] - [X]_b) = \left(\frac{-1}{t_{gauss} + 2t}\right)([X] - [X]_b)$$

Here, K_y is the diffusion coefficient (~10^4 m^2/s), y_0 is the initial plume width, and $t_{gauss} = y_0^2/4K_y$ is an initial dilution timescale. Comparison with the 1st-order case above shows that this is a very similar formulation but with a time-dependent dilution coefficient. To use this method, users must specify a value for **tgauss** in the Met inputs. When measurements are available, this parameter is often estimated by fitting to a conserved tracer like CO or CO_2 (see, e.g., ExampleSetups_LagrangianPlume.m).

If both **tgauss** and **kdil** are specified in **Met**, the model will default to 1st-order dilution. You cannot do both!

7. MODEL OPTIONS

ModelOptions is a structure containing any of the following fields, which affect model execution and output handling.

IntTime: Integration time for each step in seconds. Required input.

LinkSteps: Flag for using end concentrations from one step to initialize the next step (0 or 1, default = 0). Note that this behavior is superseded for any species that have **HoldMe=1** in **InitConc**.

Verbose: Flag for displaying verbose model execution information in command window, including progress and run times (default = 1).
0: Warnings and errors only
1: Above + Initialization messages, run time, and save name
2: Above + step increment and time
3: Above + convergence progress
4: Above + solar cycle increment
5: Above + integration progress of ODE solver

EndPointsOnly: Flag specifying whether to output concentrations for entire model step integration period (0, default) or last point of each step only (1). In Solar Cycle mode, setting this to 0 will provide output along each solar mini-step (with output time as determined by **IntTime**).

TimeStamp: Vector of times for initial conditions. This will overwrite the model output variable **Time**. Useful if you are modeling a time series of observations and want the same time base for both model and observations.

SavePath: Path for saving output. Multiple options here:
1) A full path including extension, e.g 'C:\CoolScience\MyResults.mat'.
2) A directory, e.g. 'C:\CoolScience\'. A dated directory will be created in the directory (format YYYYMMDD). A save file will be created in this directory with a name of YYYYMMDD_##.mat, where ## is incremented (starting at 01).
3) A filename, e.g. 'MyResults.mat'. Output is saved with this filename in the \Runs\ directory.
4) If left empty or unspecified, the default save path is \Runs\YYYYMMDD\YYYYMMDD_##.mat.
5) Set to "DoNotSave" to not save output. Obviously.

GoParallel: Flag for executing model steps in parallel (0 or 1, default = 0). This will only work if the following conditions are met:
1) You must have the parallel computing toolbox.
2) Each step must be independent, i.e. **LinkSteps = 0.**

This option can significantly speed up execution for setups with many steps (e.g. SS simulation of a flight mission or sweeping large parameter spaces). By default the model will use the number of workers specified by your local cluster profile, but

users can change this behavior (e.g. use a remote cluster) by modifying the parpool call in F0AM_ModelCore. Refer to parfor documentation for more information.

FixNOx: Flag for scaling total NOx to match initial conditions (0 or 1, default = 0). To use this, constraints for both NO and NO2 must be specified in InitConc with their HoldMe flags set to 0. Between each step (or mini-step if in Solar Cycle mode), model NO and NO_2 are scaled so that their sum matches the sum of input NO+NO2. It is only useful if LinkSteps = 1 or in Solar Cycle mode, and it works best when IntTime is small (600 seconds or smaller).

FixNOx performs the same basic task as family conservation, but it is functionally different. Fundamentally, FixNOx is a more coarse adjustment, but it is also more stable than family conservation. Note that FixNOx and family conservation are not compatible, so you can only use one or the other.

DeclareVictory: Set it to 1 and see what happens. Make sure your speakers are on. Useful for difficult modeling problems.

8. SOLAR CYCLE PARAMETERS

SolarParam is a structure containing information needed to run the model in a "solar cycle" mode. In this configuration, the model will allow SZA, and thus photolysis frequencies, to evolve in "real time" over the course of a model step (basically by taking mini-steps and updating SZA). This type of setup is typically employed for steady-state simulations along a flight transect. **SolarParam** is an optional input for calling the model, but if it is used then the first five fields are required. Lat, lon, alt, and startTime should have the same length as the number of inputs in **Met** and **InitConc**.

lat: Latitude, degrees (-90 to 90). North of the equator is positive.
lon: Longitude, degrees (-180 to 180). East of the meridian is positive.
alt: Altitude, meters above sea level.
startTime: 6-column matrix of start times in Universal time (UTC).
Columns are [year month day hour min sec].
This format is the same as MATLAB's "date vector" format.
nDays: Integer number of days to loop through solar cycle.
resetConcDaily: A flag (0 or 1) specifying that all species in InitConc will be reset to their initial values at the start of each day of a solar cycle run.
Converge: A structure specifying options for running in a special "convergence" mode. See below.

Convergence Mode

This is a special mode where the model will run one day at a time until it converges on a steady-state solution. "Convergence" is defined by a maximum percent change in concentrations between successive 24-hour intervals. To enable convergence mode, set SolarParam.nDays = -1. The following additional options (not required) can be specified in the SolarParam.Converge sub-structure.

Species: Cell array of strings indicating species to use for convergence calculation. Default is all species in the mechanism with concentration > 1 cm^{-3}. *DEFAULT: {'all'}*
MaxPctChange: Scalar value, indicating threshold percent concentration change for successful convergence. *DEFAULT: 0.1%.*
MaxDays: Scalar value, indicating maximum number of days to execute before giving up (akin to a timeout). *DEFAULT: 10 days.*

Example use (also see **ExampleSetup_FlightSS**):

```
SolarParam.nDays = -1;
SolarParam.Converge.Species = {'OH','HO2','NO','NO2'};
SolarParam.Converge.MaxPctChange = 1; % I am impatient
SolarParam.Converge.MaxDays = 20; % but not really
```

Other Details

The **ModelOptions.EndPointsOnly** input has special behavior in the case of a solar cycle run:

ModelOptions.EndPointsOnly = 1: output last point at end of step.
ModelOptions.EndPointsOnly = 0: output end points at intervals of **ModelOptions.IntTime** (e.g. at the end of each min-step) along the model step. In other words, output whole diel cycle(s).

9. MODEL OUTPUT

Model output is a structure that contains all of the relevant model parameters and results. This output is saved automatically to a location that depends on the **ModelOptions.SavePath** input.

Name	Type	Description	Dimension[a]
Met	Structure	Meteorological constraints	nIp x 1
Met.jcorr	Numerical Array	Default J-value correction factor	nIp x 1
Met.jcorr_all	Numerical Matrix	Correction factors for all J-values	nIp x nJ
InitConc	Structure	Initial/constraint concentrations (ppb)	nIp x 1
BkgdConc	Structure	Background concentrations (ppb)	nIp x 1
ModelOptions	Structure	Model options	Varies
SolarParam	Structure	Solar cycle parameters	Varies
SolarParam.SZAcycle	Numerical Array	SZA for each mini-step along all solar cycles	nSC x 1
Cnames	Cell Array	Species names	nSp x 1
Conc	Structure	Modeled concentrations (ppb)	nOp x 1
Time	Numerical Array	Model time (seconds, unless overwritten by ModelOptions.TimeStamp)	nOp x 1
StepIndex	Numerical Array	Integer index for model step	nOp x 1
iRO2	Numerical Array	Index for location of RO2 species in Cnames	Varies
Chem	Structure	Chemistry parameters	Varies
Chem.ChemFiles	Cell Array	Chemistry sub-mechanism names	Varies
Chem.Rnames	Cell Array	Reaction names	nRx x 1
Chem.Rates	Numerical Array	Reaction rates (ppb/s)	nOp x nRx
Chem.f	Sparse Matrix	Stoichiometric coefficients	nRx x nSp
Chem.iG	Numerical Array	Integer index for location of reactants	nRx x 2
Chem.k	Numerical Array	Reaction rate constants (1^{st} order: s^{-1}; 2^{nd} order: $cm^3\ molec^{-1}\ s^{-1}$).	nIp x nRx
Chem.iHold	Numerical Array	Index for species in Cnames held constant	Varies
Chem.iInit	Numerical Array	Index for initialized species	Varies
Chem.DilRates	Structure	Dilution rates for each species (ppb/s)	nOp x 1
Chem.Family	Structure	Information on chemical families, including member names, indices, and scaling values.	Varies
Chem.iLF	Numerical Array	Integer index for "limiting reagent" reactions	Varies

[a]nSp = number of species
nRx = number of reactions
nOp = number of output points
nIp = number of input constraints
nJ = number of J-values
nSC = number of solar cycle mini-steps (nIp*SolarParam.nDays*ModelOptions.IntTime/(86400 sec/day))

9. TOOLS

The \Tools\ folder contains functions that are useful for manipulating input and output. For more info on these, see the help sections at the top of each function.

Function	Description
breakout	Converts from a cell array/matrix pair of names/values to a structure of individual variables. Each column in the matrix is assumed to correspond to a variable name. If no output argument is assigned, variables will be written to the caller workspace.
breakin	Converts from a structure of variables (such as the **InitConc** output) to a cell array/matrix pair that contains the names and values of each variable. In the matrix, each column is a variable.
ConvertHumidity	Converts between various units for water vapor content.
DataCleaner	Perform various cleaning operations on a dataset to remove or replace NaNs and negatives. The output dataset should be appropriate for input into a box model. Should be used with awareness of the inherent assumptions in any gap-filling.
ExtractRates	Isolate and sort all reaction rates for a chemical species.
ExtractSpecies	Grab and sort a subset of chemical species concentrations. Input can be either a cell array of species names or an index (such as **iRO2**).
HYSPLIT2F0AM	Generates Met inputs for F0AM based on output from HYSPLIT trajectories. Meant to be used with the trajectory output files from the "HYSPLITcontrol" toolbox, available at https://github.com/AirChem/HYSPLITcontrol.
IndexEQ	Creates a 2-column index specifying the location of all equilibrium reactions.
IndexNOy	Creates an index for all reactive N species. Meant for MCM; not perfect.
InputReplicate	Replicates F0AM inputs (e.g. for spinning up a fully constrained diel cycle).
InputInterp	Interpolates F0AM inputs to a finer time basis.
lifetime	Calculates total chemical lifetime of a model species.
MergeRuns	Combine model output from multiple output structures.
NumberDensity	Calculates atmospheric number density at a given temperature and pressure.
ReplaceNaN	Replace NaNs in a vector with linear interpolation of nearest non-NaN points.
Rparts	Takes a cell array of reaction names and breaks them apart into cell arrays of reactant and product names.
Run2Init	Generates model initial conditions (Met and InitConc) from a subset of model results. Useful if you want to use results from one run to initialize another run.
ScaleData	Applies a linear scaling to an array.
SplitRun	Splits a model output structure into a series of new structures containing results from individual steps or repetitions within the run. Also includes the option for a user-specified custom index to split results.
struct2var	Extracts fields from a structure and reassigns them as variables in caller workspace.

sumRates_NOx	Calculates net rates of production and loss for the NOx family. Designed for MCM.
sumRates_ROx	Calculates net rates of production and loss for the ROx family. Designed for MCM.
sun_position	Calculates solar zenith and azimuth angle for the Earth at any time/location.
SMILES (folder)	SMILES (simplified molecular input line-entry system) is a way to represent molecules with ASCII strings This folder contains an experimental function, **SearchSMILES**, to identify MCM species with specific functionalities using SMILES strings, returning their names, molecular weights and atom counts. This function has not been rigorously tested, but most of the patterns should be captured.

10. PLOTS

The \Plots\ folder contains a handful of functions for generating common plots of model output.

Function	Description
PlotConc	Plots time series of a species or an arithmetic combination of species (e.g. NO/NO2). Can accept multiple model structure inputs.
PlotConcGroup	Plots time series of a group of species. Useful for, e.g., looking at the distribution of RO2 or NOy.
PlotRates	Plots time series of production and loss rates for a single species or a family of species.
PlotRatesAvg	Plots production and loss rates for a single species averaged over some subset of outputs.
PlotReactivity	Plots time series speciated reactivity, which is the inverse lifetime of a species. Useful for, e.g., plotting OH reactivity. Requires some unique user-defined inputs.
PlotYield	Calculates and plots the yield of a product from the oxidation of a reactant. Typically used for looking at chamber experiments.
purtyPlot	User-preferred settings for plot decorations
fillcolors.mat	3-column RGB matrix for colors used in multi-species plotting

All plot functions will also return the plotted variables if an output variable is assigned when calling the function. Also, all of these functions accept several options, which are input as name-value pairs. The specific options supported depends on the function; see comments in each function for full details. Options shared across most functions are listed below.

PlotFun(...,'ptype',value): Indicates type of plot. Values vary depending on function.

PlotFun(...,'unit',value): Specify the unit.

PlotFun(...,'scale',value): specify a scalar multiplier. For functions that plot groups of things, setting this to 0 causes normalization by the sum of the group.

REFERENCES

D'Ambro, E. L., Møller, K. H., Lopez-Hilfiker, F. D., Schobesberger, S., Liu, J., Shilling, J. E., Lee, B. H., Kjaergaard, H. G., and Thornton, J. A.: Isomerization of Second-Generation Isoprene Peroxy Radicals: Epoxide Formation and Implications for Secondary Organic Aerosol Yields, Env. Sci. Technol., 51, 4978-4987, 2017.

Dillon, M. B., Lamanna, M. S., Schade, G. W., Goldstein, A., and Cohen, R. C.: Chemical evolution of the Sacramento urban plume: Transport and oxidation, J. Geophys. Res., 107, 4045, 2002.

Jenkin, M. E., Saunders, S. M., and Pilling, M. J.: The tropospheric degradation of volatile organic compounds: A protocol for mechanism development, Atmos. Env., 31, 81-104, 1997.

Kaiser, J., Skog, K. M., Baumann, K., Bertman, S. B., Brown, S. B., Brune, W. H., Crounse, J. D., de Gouw, J. A., Edgerton, E. S., Feiner, P. A., Goldstein, A. H., Koss, A., Misztal, P. K., Nguyen, T. B., Olson, K. F., St. Clair, J. M., Teng, A. P., Toma, S., Wennberg, P. O., Wild, R. J., Zhang, L., and Keutsch, F. N.: Speciation of OH reactivity above the canopy of an isoprene-dominated forest, Atmos. Chem. Phys. Disc., doi: 10.5194/acp-2015-1006, 2016. 1-20, 2016.

Wolfe, G. M. and Thornton, J. A.: The Chemistry of Atmosphere-Forest Exchange (CAFE) Model - Part 1: Model Description and Characterization, Atmos. Chem. Phys., 11, 77-101, 2011.