Notes on SOM v7 (through v 7.3.4)

Prof. Chris Cappa (cdcappa@ucdavis.edu)

University of California, Davis

Last update: 07/05/17

This "manual" describes how to run simulations and fit data using the SOM model of Cappa and Wilson (ACP, 2012). It discusses the following:

1. Setting up a SOM simulation using the panel
2. Running a single SOM simulation
    a. Single Component
    b. Multi-component
3. Accessing previously determined fit parameters for various VOCs
4. Accessing "historical" Caltech data
5. Fitting a single chamber observation using SOM
    a. Fitting assuming no vapor wall losses
    b. Determining an optimal k_{wall} and alpha

This "manual" and the program may be difficult to use without a short tutorial. But you are welcome to try.

Running SOM

The IGOR SOM is based on the model described in Cappa and Wilson (ACP, 2012) and Zhang et al. (PNAS, 2014). The various reactions/pathways are simulated using a Forward Euler method to solve the differential equations.

The IGOR SOM is primarily operated from the "sompanel," where various inputs can be adjusted. Boxes with up/down arrows to the right are adjustable. The various controls are described below.

SOM Procedures

Need to make sure that you have at least the following procedures loaded. The first set should be in the Igor file by default as they are embedded:

- Procedure Window (the default)
- OtherSOMStuff
- SOM_Fitting
- aPinene2015

The next set may need to be loaded separately (and may not even be necessary), as they are external procedures.

- Global_Utils_CDC.ipf
- 2015GeneralMacros.ipf
- SizeDistributionProcessing_v1.0.0.ipf

SOM Functions

There are a variety of functions used when running SOM. The two key ones are:

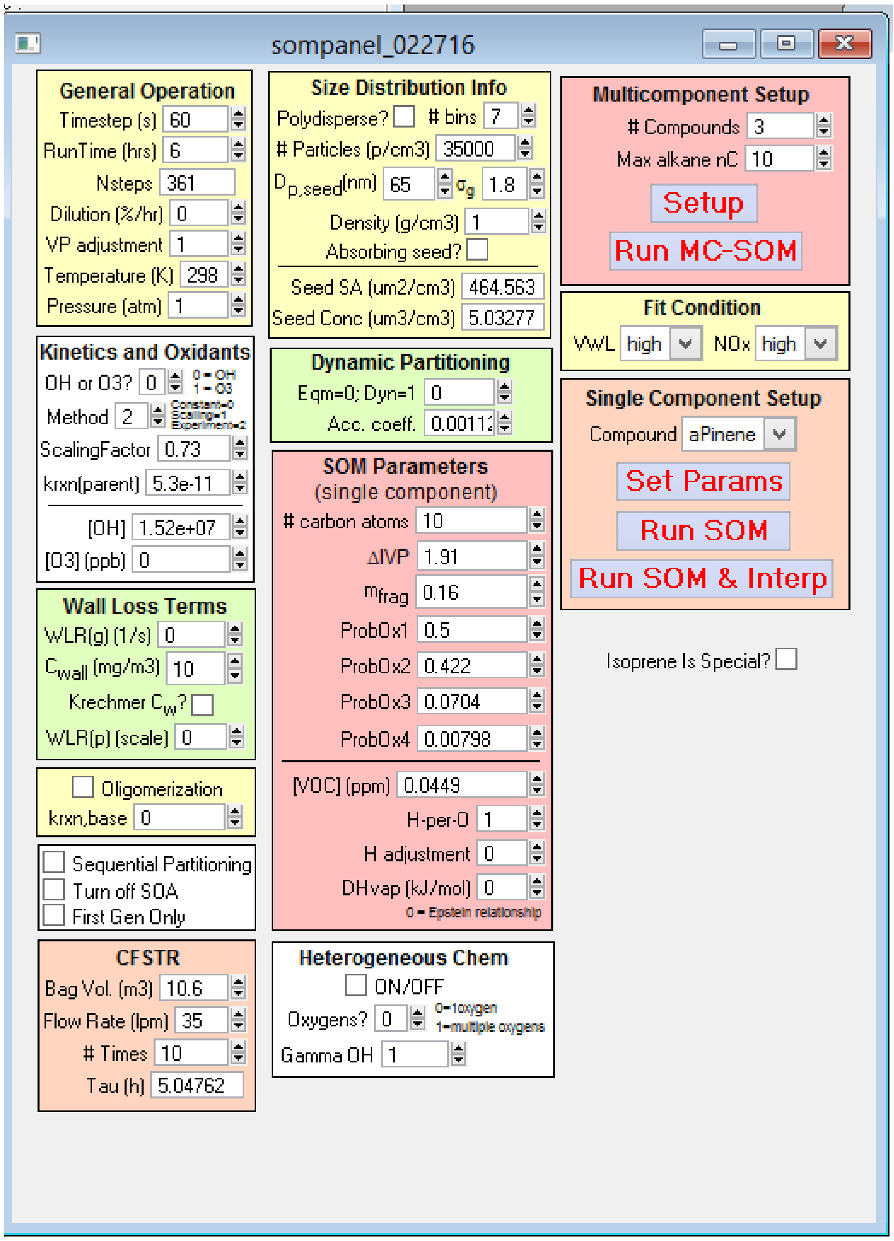
- SOM_V1() = the main SOM program for a single-component simulation
- SOM_MC() = the main SOM program for a multi-component simulation

There are a variety of sub-functions that are accessed. These include:

- SOM_CreateAtomMatrices(nC,nO,H_per_O,Hadjustment) = creates matrices with number of atoms for SOM species (e.g. oxygens, carbons, hydrogens, molecular weight)
- SOM_RateCoefficients() = determine rate coefficients for SOM species (e.g. C_xO_y) based on particular rules
- SOM_OligomerRateCoef() = determine rate coefficients for oligomerization…in development
- SOM_Fragmentation() = determines the fragmentation matrices
- SOM_ Fragmentation_MP() = same as SOM_ Fragmentation but for multiple components
- SOM_GasPhaseWallLoss() = determines the species-specific wall desorption rates
- SOM_ParticleWallLossRate() = determines particle wall loss rates
- SOM_Heterogeneous() = calculates heterogeneous reaction rates
- EqmCalc() = calculates equilibrium gas-particle distribution based on input conditions
- EqmCalc_MultipleVOC() = same as EqmCalc() but set up to work with multiple VOCs
- SOM_Oligomerization() = runs oligomerization reactions…in development
- SOM_KillWaves() = function to kill unnecessary waves at end of run
- SOM_KillWaves_MultiComponent() = kills additional unnecessary waves in compound-specific folders
- Makelognormaldistn() = function to make a log normal distribution based on the specified diameter, number, spread and number of bins
- MakeSOMwaves() = for multicomponent SOM only; makes folders to hold compound-specific information

And there are some functions to work with observational data and/or process data/graph model results:

- Graph_MC_yield() = graphs the compound-specific mass yield for multi-component simulations
- Graph_MC_GandP() = graphs the compound specific VOC decays and SOA formation as stacked plots for multi-component simulations
- Graph_MCsoa() = graphs the compound specific SOA formation as stacked plots for multi-component simulations
- SOM_SetupSingleComponent() = set SOM parameters based on stored fit
- SOM_SetupMulticomponent() = set SOM parameters based on stored fits for multi-component simulations
- SetFromCaltech() = accesses historical data from the Caltech chamber
- SetFromCaltech2() accesses historical data from the Caltech chamber, specifically data from McVay et al. (2015)
- SOM_BatchFit() = used to perform fit and store results for multiple experiments
- Allatonce_FitCoa() = function used to fit a single experiment



sompanel_022716

General Operation

Timestep (s)	60
RunTime (hrs)	6
Nsteps	361
Dilution (%/hr)	0
VP adjustment	1
Temperature (K)	298
Pressure (atm)	1

Kinetics and Oxidants

OH or O3?	0 (0 = OH, 1 = O3)
Method	2 (Constant=0, Scaling=1, Experiment=2)
ScalingFactor	0.73
krxn(parent)	5.3e-11
[OH]	1.52e+07
[O3] (ppb)	0

Wall Loss Terms

WLR(g) (1/s)	0
C_{wall} (mg/m3)	10
Krechmer C_W?	☐
WLR(p) (scale)	0

☐ Oligomerization
krxn,base 0

☐ Sequential Partitioning
☐ Turn off SOA
☐ First Gen Only

CFSTR

Bag Vol. (m3)	10.6
Flow Rate (lpm)	35
# Times	10
Tau (h)	5.04762

Size Distribution Info

Polydisperse?	☐
# bins	7
# Particles (p/cm3)	35000
$D_{p,seed}$(nm)	65
σ_g	1.8
Density (g/cm3)	1
Absorbing seed?	☐
Seed SA (um2/cm3)	464.563
Seed Conc (um3/cm3)	5.03277

Dynamic Partitioning

Eqm=0; Dyn=1	0
Acc. coeff.	0.0011:

SOM Parameters
(single component)

# carbon atoms	10
ΔlVP	1.91
m_{frag}	0.16
ProbOx1	0.5
ProbOx2	0.422
ProbOx3	0.0704
ProbOx4	0.00798
[VOC] (ppm)	0.0449
H-per-O	1
H adjustment	0
DHvap (kJ/mol)	0 (0 = Epstein relationship)

Heterogeneous Chem

ON/OFF	☐
Oxygens?	0 (0=1oxygen, 1=multiple oxygens)
Gamma OH	1

Multicomponent Setup

# Compounds	3
Max alkane nC	10

Fit Condition

VWL	high
NOx	high

Single Component Setup

Compound	aPinene

Isoprene Is Special? ☐

SOM Inputs

General Operation

- **Timestep**: the timestep for the chemistry calculations.
  - Typically set to 60 seconds
  - When dynamic partitioning is used the mass transfer uses a timestep that is <= the specified timestep, which is important if one wants a stable simulation. This can be very short, meaning long simulations, when alpha is large (> 0.1).
- **Dilution**: rate of dilution, not really important for chamber experiments
- **VP adjustment**: just leave this at 1. This allows for adjustment of the parent VOC vapor pressure from the "standard" case (which is based on alkane backbones)

- **Runtime**: total run time. The number of steps (Nsteps) is calculated based on this and the timestep
- Nsteps: this is calculated, not an input. It is reported in the panel for reference.
- **Temperature** (K): self-explanatory
- **Pressure** (atm): self-explanatory

Kinetics and Oxidants

- **OH or O3**: For simulations using OH or O3 as a reactant. The SOM is really set up for OH reactions (set to zero). O3 can be selected (set to 1), but this is not completely robust.
- **Method**: constant = 0; scaling = 1; experiment = 2
  - Constant = constant OH at the concentration set in the [OH] box
  - Scaling = will use a scaling factor to allow for an exponential decrease in [OH] from the initial [OH] with time. If Method = 1, then set the **scaling factor** input.
  - Experiment = will use a wave with time-varying OH concentrations for the calculations. The wave that is used is named "OH_exp" and must have an associated time wave with name "OH_time". These are interpolated to the appropriate time steps for use in the simulation. This is useful for simulating chamber experiments and you know how [OH] varied with time.
- **krxn(parent)**: units = cm^3 $molecules^{-1}$ s^{-1}; The rate coefficient for the parent VOC
  - set this to zero to use a calculated krxn based on SAR relationships (the base case assumes an alkane).
  - Enter a value if you want your parent species have a different rate coefficient than the SAR relationships. Useful for simulation of aromatics, alkenes, etc. (i.e. those things that don't look like alkanes).
- **[OH]** = OH concentration in molecules/cm^3. Necessary if Method = 0 or 1. If Method = 1 this is the initial [OH].

Wall Loss Terms

- **WLR(g)** = gas-phase wall loss rate in 1/s. Currently it is assumed that this is the same for all species.
- **WLR scaling** = scaling factor (legacy, set to zero)
- **Cwall** = effective absorbing mass concentration (mg/m^3) of the wall
- **Krechmer C_w?** = decide whether to assume that the specified C_{wall} is the same for all species or whether the effective Cwall varies with C* (as suggested by Krechmer et al. (ES&T, 2016)).
  - Unchecked = constant Cwall
  - Checked = use Krechmer relationship
- WLR(p) = scaling factor for particle wall loss rate (usually set to zero). If set to 1, particles deposit irreversibly to the wall following the functional form in Loza et al. (ACP, 2010). This is a multiplicative factor onto the Loza et al. relationship (so setting to e.g. 2 means that particles are lost twice as fast as the observations suggest

Size Distribution Info

- This sets the properties of the seed particles (if they exist). Only important when doing dynamic partitioning (and actually may not work with eqm. Partitioning).
- **Polydisperse?**: check box to decide whether to use a single size particle or assume a distribution of particle sizes. Note that things slow down considerably for polydisperse. But can be used to examine changes in shape of size distribution. If "yes" (checked) then a log-normal distribution is created that has **# bins** having a geometric diameter **$D_{p,seed}$** with a spread of **σ_g** and a total number concentration of **# particles** (in p/cm^3).
- **# Bins:** the number of sizes to use if a polydisperse distribution is assumed. Use of 7 seems reasonable in most cases to get the general idea.
- **# particles:** (p/cm^3) the number concentration of particles. The total seed surface area and volume depends on this. You can adjust the # of particles and the seed diameter to get the "correct" (i.e. observed) seed surface area when doing monodisperse particles.
- **$D_{p,seed}$:** in nm. Geometric median diameter for distribution or monodisperse size
- **σ_g:** the size distribution spread
- **Density:** (g/cm^3) the assumed material density of the seed. Not really important unless you are assuming an absorbing seed.
- **Absorbing seed:** no check = seed does not participate in partitioning (i.e. does not mix). Check = seed participates in partitioning (i.e. mixes)
- Seed SA: the calculated seed surface area concentration
- Seed Conc: the calculated seed volume concentration

Dynamic Partitioning

- **Eqm/Dyn** : select instantaneous equilibrium (0) or dynamic (1) partitioning. Dynamic runs a lot slower than eqm. How slow depends on what the accommodation coefficient is set to. Larger accommodation coefficients correspond to slower calculations. Very slow if you have the accommodation coefficient > 0.1. But in this case you can probably just assume equilibrium partitioning with minimal problems.
- **Acc. Coeff** : accommodation coefficient (only used when Eqm/Dyn = 1)

SOM Parameters

- **# carbon atoms**: for a single component simulation, this is the # of carbon atoms of the parent VOC compound
- **ΔLVP:** the assumed decrease in volatility to occur for each oxygen that is added. This is in log(C*) units, where C* is micrograms/m^3.
  - typical range = 1-2.5
- **m_{frag}:** the fragmentation parameter, that determines the fragmentation probability. Smaller numbers correspond to greater fragmentation. Set to 10 if you want (essentially) zero fragmentation. Set to 0.01 if you want a lot of fragmentation. Do not set to zero.
- Oxygen probability array: the probabilities of adding some number of oxygen atoms per reaction. These must sum to 1. However, you can enter whatever numbers you want and they will be normalized in the calculation.
  - **ProbOx1** = probability of adding 1 oxygen upon reaction
  - **ProbOx2** = …2 oxygens

    - **ProbOx3** = …3 oxygens
    - **ProbOx4** = …4 oxygens
- **[VOC] (ppm):** the concentration of the parent VOC, for single component simulations
- **H-per-O:** the number of hydrogens assumed to be lost per oxygen added. Default is 1. This affects the MW calculations, and thus the vapor pressures (slightly). It mostly impacts the calculated H:C, basically sets the value.
- **H-adjustment:** ignore this and set to 0. can be used to adjust the parent MW to account for "missing" hydrogens relative to an alkane, for example as would happen for an aromatic compound. This can be set to the exact value, but can be left at zero too. If it is left at zero, the other fit parameters will simply be a little different than they would be if this is set to some other value. I suggest just leaving at 0.
- **DHvap (kJ/mol):** ignore this and set to 0. Potentially important for T-dependent simulations. Characterizes the variation in vapor pressures with temperature. 0 assumes the relationship given by Epstein et al.

Heterogeneous Chem, etc.

- ON/OFF button: turn heterogeneous chemistry on or off
- Oxygens? = number of oxygens to add per heterogeneous reaction. 0 = 1 oxygen, 1 = multiple oxygens based on Oxygen Probability Array. The use of multiple oxygens is not yet robust so use with caution
- Gamma OH = OH effective uptake coefficient

Other

- **Isoprene is Special:** there have been some arguments made (especially by Jose Jimenez) that we know "enough" about isoprene chemistry that we should be treating it more explicitly and with more knowns/constraints. The "isoprene is special" button overrides the default SOM multigeneration oxidation scheme to treat a few species uniquely. Details are given in Hodzic et al. (ACP, 2016). Note that the fit parameters for use with Isoprene Is Special need to be entered by hand.
- **Oligomerization**: This is a work in progress. Leave the check box unchecked
- **Sequential partitioning**: Check to use the "sequential partitioning model" from Cappa and Wilson (2011). It works just fine with eqm. Partitioning, but probably doesn't work correctly with the dynamic partitioning. A work in progress.
- **Turn Off SOA**: does not allow for SOA to form so that one can examine vapor wall loss only.
- **First Gen Only**: Turns off multi-generational oxidation, only allowing for production of "first generation" products. This is essentially the same as assuming things react with O3 (if there are not multiple double bonds and OH is not generated). Can be used to examine the influence that the multi-generational ageing has on SOA formation.
- **Eqm**. **Method**: (LEGACY; now set to default in code, not panel) When instantaneous equilibrium is used (or really dynamic partitioning too) one can choose to calculate the Raoult's Law vapor pressure depression based on mass concentrations or molar concentrations. The default is to have this checked and use the mass concentrations (after Donahue et al., ES&T, 2006)

Single Component Setup

- **Compound:** If you want to use best fit parameters (i.e. the SOM parameters) for some previously fit dataset you can select the compound that you want here. This is used in conjunction with the **Set Params** button and the **VWL** and **NOx** conditions under Fit Conditions.
- **Set Params:** click this to have the krxn(parent), # carbon atoms, ΔLVP, mfrag and ProbOx values set based on previous fits to observations. Is linked to **Compound**, **VWL** and **NOx**
- **Run SOM**: click this to run the model based on the parameters that have been set. This will run the function SOM_v1().
- **Run SOM & Interp**: click this to run the model based on the parameters that have been set, but then to interpolate the final results to the same timebase as a given experiment. There must exist a wave with name "Time_Exp" that has the time (in hours) to which you want to interpolate the model results. This is useful and important when doing data fitting.

Fit Condition

- You can access SOM parameters based on previously performed fits (c.f. Zhang et al., PNAS, 2014 and Jathar et al., GMD, 2015). The Fit Condition specifies what the assumption was regarding vapor wall losses during fitting and the NOx condition.
- **VWL:** if you want to setup to do calculations based on some previous fit, this specifies the assumption regarding vapor wall losses used in doing the original fit.
    - **Zero:** the fit was done assuming no wall loss (kwall = 0)
    - **Low:** the fit was done assuming "medium" wall loss (kwall = 1e-4)
    - **High:** the fit was done assuming "high" wall loss (kwall = 2.5e-4)
- **NOx:** fits to observations were performed for data collected under "high" and "low" NOx conditions. This specifies which fit to use.

Multicomponent Setup

Some additional details on the multi-component SOM are available in Notebook0 in the Igor experiment

- It is possible to perform simulations assuming that you have more than one parent compound. This requires some user effort to get things set up correctly
- **# compounds:** the number of parent compounds to consider
- **Max alkane nC:** if alkanes are being considered, this is the maximum number of carbon atoms to be used. This is because if you want to have a multicomponent simulation with alkanes it is actually quite easy to do (although hard to track the evolution on a compound-by-compound basis). If you specify e.g. **max alkane nC** = 10, then you can have a multicomponent system with alkane precursors having carbon numbers ranging from 1 to 10.
- **Setup:** Will pop up a dialog and table to allow you to specify the details of your multicomponent system. More details are given below.
    - **Dialog:**
        - **Compounds:** select the names of the compounds that you want to consider.
        - **Concentrations:** enter the concentrations of the compounds (in ppb). Enter these as a semi-colon separated string (no spaces)
    - **Table:**

- An example is shown below in the figure for a 4-component system with isoprene, benzene and n-alkanes and branched alkanes (with an imposed upper limit of 10 carbons, for this example)
- **ParameterNames**: just identifies the parameter names you are working with
- **SOMParams**: a matrix that holds the SOM parameters (e.g. # carbons, mfrag) for each compound. The assumed compound name is indicated. These are prepopulated with values based on prior fits and the specified fit conditions (VWL and NOx). You can adjust as desired
- **InitialConcMatrix**: (units = ppm) This is an *p* x *q* matrix, where *p* is the maximum number of carbon atoms for any of the species being considered and *q* is the number of species being considered.
    - Non-alkanes: For all compounds except alkanes, the initial concentration is entered in the row corresponding to the carbon number of the species of interest (counted from the bottom). These will automatically be entered for you based on the concentrations you entered in the pop-up dialog. But you can adjust them here without having to run the setup again.
    - Alkanes: For alkanes, the concentration in each row is the concentration of an alkane with a number of carbon atoms equal to |*p*-Max alkane nC+1|. So, if **Max alkane nC** = 10, the first row is for the species with 10 carbon atoms, the second for species with 9, etc. This is populated automatically with a distribution based on Zhao et al. (ES&T, 2015) where the total concentration across all species is equal to the value that you entered in the pop-up dialog. If you only want one alkane compound (e.g. only the C10 alkane) then just set all other rows in that column to zero.

- **Run MC-SOM:** will run a simulation for a multicomponent system. Runs SOM_MC().

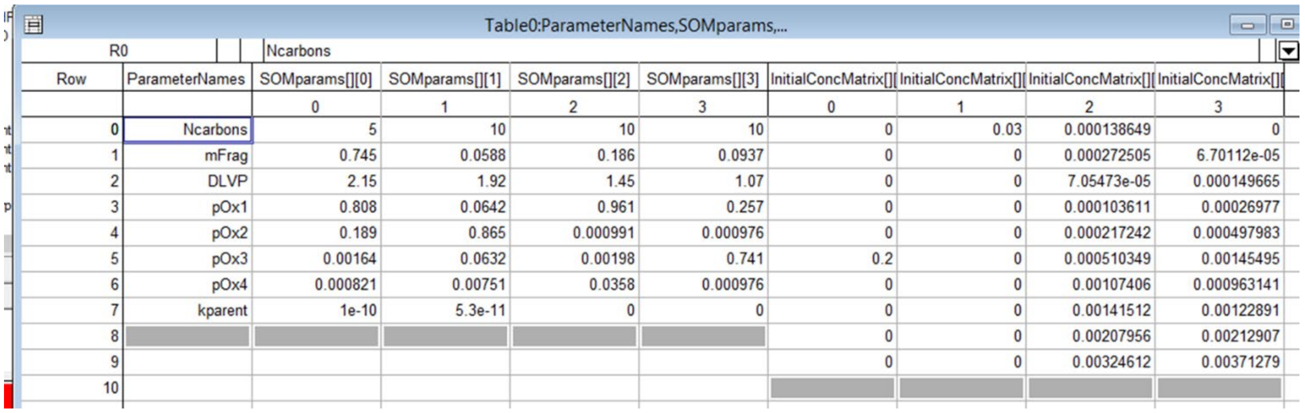


Table0:ParameterNames,SOMparams,...

R0 | Ncarbons

Row	ParameterNames	SOMparams[][0]	SOMparams[][1]	SOMparams[][2]	SOMparams[][3]	InitialConcMatrix[][	InitialConcMatrix[][	InitialConcMatrix[][	InitialConcMatrix[][
		0	1	2	3	0	1	2	3
0	Ncarbons	5	10	10	10	0	0.03	0.000138649	0
1	mFrag	0.745	0.0588	0.186	0.0937	0	0	0.000272505	6.70112e-05
2	DLVP	2.15	1.92	1.45	1.07	0	0	7.05473e-05	0.000149665
3	pOx1	0.808	0.0642	0.961	0.257	0	0	0.000103611	0.00026977
4	pOx2	0.189	0.865	0.000991	0.000976	0	0	0.000217242	0.000497983
5	pOx3	0.00164	0.0632	0.00198	0.741	0.2	0	0.000510349	0.00145495
6	pOx4	0.000821	0.00751	0.0358	0.000976	0	0	0.00107406	0.000963141
7	kparent	1e-10	5.3e-11	0	0	0	0	0.00141512	0.00122891
8						0	0	0.00207956	0.00212907
9						0	0	0.00324612	0.00371279
10									

SOM Outputs

Single Component SOM

Key Waves

When the single component SOM is run a number of waves are generated. The key outputs are:

- **Coa_time** = SOA mass concentration (ug/m^3)
- **O2C_time** = O:C ratio
- **HC_ppb_time** = parent hydrocarbon concentration in ppb.
- **deltaHC_time** = amount of parent VOC reacted, in ppb
- **TimeW** = time wave for run, in hours

If you chose to interpolate waves to an experimental time base the key output waves are:

- **Coa_time_interp** = SOA mass concentration (ug/m^3)
- **O2C_time_interp** = O:C ratio
- **deltaHC_time_interp** = amount of parent VOC reacted, in ppb

Other Waves

A number of other waves are generated that store the results from a simulation. Many more are generated and killed at the end. The killing of waves is controlled by the SOM_KillWaves() function. If you want something to show up at the end comment it out in the SOM_KillWaves() function

- **Diameter_time** = time-series of particle diameters for each size bin
- **Diameter_final** = diameter by bin for last time point in the run
- **dNdlogDp_time** = time series of dN/dlogDp (p/cc) for polydisperse simulations
- **dNdlogDp_final** = final size distribution
- **GasMass_time** = 3D matrix with time-series of each SOM species in the gas phase
- **ParticleMass_time** = 3D matrix with time-series of each SOM species in the particle phase
- **GasMass_Matrix** = 2D matrix of mass concentrations for each gas-phase species at last step
- **ParticleMass_Matrix** = 2D matrix of mass conc. for each particle-phase species at last step

Comparing to and Fitting Data

Experimental Data

If one wants to fit data (or compare to data), the data must live in the root folder and have the following names

Particle Phase

- Coa_experiment = time-series of SOA concentrations (ug/m3)
- Coa_experiment_err = time-series of SOA concentration uncertainties (not necessarily used, but populate nonetheless
- Coa_experiment_mask = wave that indicates whether to include the associated Coa_experiment data point as part of the fitting. 1 = include. 0 = exclude
- Time_experiment = wave with time elapsed, in hours; must be same length as Coa_experiment
- O2C_experiment = time-series of SOA O:C ratios

- O2C_experiment_err = time-series of O:C uncertainties
- O2C_experiment_mask = same as Coa_experiment_mask

Gas Phase

- VOC_ppm = time-series of VOC concentrations; does not need to be same length as Coa_experiment; must match Time_VOC wave
- Time_VOC = wave with elapsed time for VOC data; in hours
- OH_exp = time-series of OH concentrations (probably derived from the VOC decay waves separately); units = molecules/cm3
- OH_exp_time = wave with elapsed time for OH concentrations; in hours
    - Generally try and make sure that the longest time here is longer than the run time in the model. For example, if the last point is as 9.3 h then you should only run for 9.29 h max.

Accessing Historical Data

Historical data from Caltech are stored in the folder root:Caltech_Data. This folder contains subfolders with the data this includes:

- Alkanes
    - C12 compounds
        - Dodecane
        - Cyclododecane
        - HexylCycloHexane
        - Methylundecane
- Aromatics
    - Benzene
    - mXylene
    - naphthalene
    - Toluene
- Biogenics
    - aPinene
    - isoprene

There exist data for both high/low NOx photooxidation experiments. You can access the data and have the e.g. Coa_experiment waves in root populated using the function

Setting "Experiment" values based on a given experiment

1. You can grab the experimental conditions for a given experiment by using the function "SetFromCaltech" or "SetFromCaltech2".
2. SetFromCaltech: This works for "historical" data that were collected under either high or low NOx conditions. Entries would look like e.g. SetFromCaltech("dodecane","low") for a low-NOx experiment using dodecane as the precursor VOC
3. SetFromCaltech2: This works for the data in McVay et al. (2016) performed for alpha-pinene under variable seed conditions and variable UV condtions. Entries would look like SetFromCaltech2("aPinene","low","low") where the first "low" refers to the UV condition and

the second "low" to the seed concentration. This can be run from the command line or from a function.

a. Data are stored in root:Caltech_data:biogenics:OH:aPinene_2016 and associated subfolders
    i. Data in subfolders are separated according to the UV and seed surface area during a given experiment
b. Acceptable combinations of UV and seed conditions are:
    i. Low, low
    ii. Low, high
    iii. High, low
    iv. High, med
    v. High, high
c. The observed Coa time series will be copied to root:Coa_experiment and root:Time_experiment
    i. The "mask" wave that is stored in the data folder will be copied to Coa_experiment_mask
    ii. The "uncertainty" wave that is stored in the data folder will be copied to Coa_experiment_err
d. The particle number and seed diameter will be set to values that will give the correct seed surface area assuming monodisperse particles
e. The OH method will be set to "2" (= experimental value) and the observationally constrained OH time series will be copied to root:OH_exp and root:OH_exp_time
f. The observed VOC decay time series will be copied to root:VOC_ppm and root:Time_VOC

Fitting Data – One compound and one condition

- Igor has a built in fit routine that can be accessed either from dropdown menus or from the command line/functions. Data can be fit to the SOM using either.
- To fit to a function that is not built-in to Igor (such as SOM) requires that one write a function that defines a new fit routine. There is a procedure window titled "SOM_Fitting". The fit routines can be found here.
    - To fit just concentration data, use the "allAtOnce_FitCoa" function.

Fitting using the dropdown menu

- This is most easily done by putting the data that you want to fit on a graph and going from there. So, start by making a graph of Coa_experiment vs time_experiment (or just use the "fitGraph" that is already created).
- Make sure that the data you want to deal with is stored in the root folder in the waves Coa_experiment and time_experiment.
- Create a wave in root (if it doesn't already exist) called CoefWave (or whatever you want to call it) using the command make/o/d/n=(6) CoefWave = {0.5,1.75,0.25,0.25,0.25,0.25}. This wave stores the fit parameters, in this case {mfrag, DLVP, Pfunc1, Pfunc2, Pfunc3, Pfunc4}.
- Bring the graph to the front.
- Go to "Analysis:Curve Fitting". A pop up dialog will come up.

    - On the “Function and Data” tab:
        - Select “allatonce_FitCoa” from the “functions” dropdown.
        - Select “From Target”
            - This should set your “Y data” to “Coa_experiment” and your “X data” to “Time_experiment”. If this doesn’t happen, you can set them by hand.
    - On the “Data Options” tab you can choose to restrict the range over which the data are fit. I usually have a separate wave (Coa_experiment_mask) that indicates whether data should be included or not. In this wave 1 = include and 0 = exclude. If you want to fit all of the data then this doesn’t matter and you can leave it blank.
    - Click on the “Coefficients” tab. The first time you do this you might get a warning saying that it needs you to enter values for a user defined function. This is okay.
        - Under “Coefficient Wave” select the CoefWave that you generated above.
            - Enter your initial guesses here. When in doubt, try the following:
            - Mfrag = 1
            - DLVP = 1.8
            - Pfunc1-4 = 0.25
        - Under “Constraints” select “From Coefficients List”.
            - This puts limits on the values for e.g. DLVP, mfrag, Pfunc. This is important as none of these can be negative.
            - For mfrag (i.e. CoefWave_0) use the range 0.01 to 10
            - For DLVP (CoefWave_1) use the range 0.7 to 2.5
            - For Pfunc1-4 (CoefWave_2-5) use the range 0.01 to 100. Don’t worry that the largest value is > 1…this gets renormalized in the code.
        - I typically click on “Graph Now” to test whether things seem to be working, or to decide whether I want to change my initial guesses.
    - Once you’re happy with your inputs, click on “Do It” on the bottom left. This will start things running. Depending on what type of calculations you are doing this can be “fast” or very slow. The dynamic partitioning is comparably slow, especially when alpha is large.
- Important: Make sure that the “RunTime” in the panel is longer than max value in the “Time_Experiment” wave. Otherwise you will run into errors.

Fitting Data – Multiple compounds, one condition at a time

THIS ONLY WORKS FOR FITTING THE HISTORICAL CALTECH DATA. But the process can be adapted.

This works if you have stored your data (concentrations vs. time, OH vs. time, etc.) in folders. There is a table called “FitSetup” where you will specify the compound to fit

- Create a new line (or over-write an old line) in the “FitSetup” table. Note that there are a lot of different waves in this table, but they must all be filled in with the correct information.
- Waves to populate before fitting:

    - “CompoundName” is the compound name associated with the appropriate folder where the data are stored.
    - “NOx” is the NOx condition (low vs. high). Note that “NOx” might not simply be low or high as I had to accommodate situations where there are multiple data sets for a given compound for a given NOx condition. Each label (e.g. low vs. low2) has an associated data folder, that can be checked by looking at the GetDataFolderName_Caltech(Compound,NOx) function.
    - HoldWave = ignore this and set to 0
    - MaskWave: 0 = no mask applied during fitting; 1 = mask (Coa_experiment_mask) applied during fitting
    - FitTo: Coa = fit to Coa_experiment only (use this); “Coa and O2C” = fit to both Coa and O2C (don’t use this)
    - O2Cerr_scale: set to 1 and ignore
    - FragType: decide whether to use mfrag (Pfrag = (O:C)^mfrag) or cfrag (Pfrag = No*cfrag). Use mfrag
    - Het_Wave: 0 = no heterogeneous chemistry; 1 = heterogeneous chemistry
    - WLRgas_wave: vapor wall loss rate in 1/s
    - InitialGuesses: initial guesses for mfrag, DLVP, Pfunc1, Pfunc2, Pfunc3, Pfunc4
    - NpWave = seed particle number concentration (p/cc)
    - GammaOH_wave: OH uptake coefficient for use when Het_Wave = 1
    - Cwall_Wave: effective wall concentration (mg/m^3)
    - WLRg_scaling: some legacy scaling factor. Set to 0.
    - WLRp_wave: 0 = no particle loss; 1 = particle loss according to equation given in SOM mechanism (derived based on comparison with real Caltech chamber observations, i.e. Loza et al.)
    - SPMwave: to use the sequential partitioning model, set this to 1. Default is zero. Note that SPM has not been confirmed to work correctly with dynamic partitioning.
    - kOH_wave: what method to use for specifiying the rate coefficient matrix. Use 4.
    - ErrorWave: legacy. Ignore and set to 0.
    - HoldStr: set to none (unless you really want to hold some of the coefficients constant)
    - DpSeed_Wave: seed particle diameter
    - Alpha_wave: mass accommodation coefficient. Only used if DynamicPartitioning_Wave = 1.
    - DynamicPartitioning_Wave: 0 = instantaneous eqm; 1 = dynamic partitioning.
- Waves that get populated upon fitting
    - FitResults: results for mfrag, DLVP, Pfunc1, Pfunc2, Pfunc3, Pfunc4
    - ChiSq: chisq associated with best fit…actually, this is always calculated incorrectly for a reason that I still haven’t figured out. You can get the correct value by running “RecalculateChi2_Batch(start,stop)”
    - RunTime: when did you do the fit
    - ChiSq_O2C: chisq associated with O2C fit. Ignore this.
- To do the fit:

    - Run the function SOM_BatchFit(startpos,stoppos,dataset) where startpos and stoppos are the indices associated with the data that you want to fit and dataset = "Caltech". For example, based on what is currently set up in the FitSetup table, if you run SOM_BatchFitCaltechData(0,4) this will perform fits for dodecane, methylundecane, cyclododecane and hexylcyclohexane low NOx experiments. This function will automatically grab data from the appropriate folder and copy it to Coa_experiment and Time_experiment in the root folder. (It also grabs O2C_experiment and a few other things.) Once you set this going it should start to fit dodecane. Once it completes that fit it will start on the next compound, etc., etc.
    - When you run this it also grabs the starting concentration from the "RunParameters" wave that exists in each data folder. It also grabs some additional information from this wave, including how to treat the [OH] (e.g. constant, time-dependent according to some function or based on some input wave), the number of carbon atoms in that molecule, the rate coefficient associated with degradation of the parent species (important for aromatics and alkenes).
    - Usually the fit algorithm does pretty well in finding a solution. But it can get lost or stuck, especially if you give a poor initial guess. So watch out for errors, which stop the fitting routine. To kill a fit use "ctrl-shift-break". The normal "abort" button on the bottom left of the Igor screen does not seem to work all the time.
- After you've performed the fits, if you want to reload data for a given compound *and* you want to show the SOM simulation results for the compound based on a given fit, then you can run RecalculateChi2_Batch(start,stop), where start and stop serve the same purpose as startpos and stoppos above.

Fitting Data – Determine optimal k_{wall} and alpha (accommodation coefficient)

The goal is to determine the k_{wall} and alpha value pair that provides for the best fit of SOM to the observations. This can be done for a single experiment (i.e. a given UV/seed pair) or one can aim to find the (k_{wall},alpha) that gives the best global fit for a given UV condition. Descriptions of the functions that can be used to perform batch fitting are given below. These functions are stored in the "aPinene2016" procedure window.

1. **BatchFit_SOM_kwall_alpha:** A *single* experiment (UV/seed combination) can be fit using the function BatchFit_SOM_kwall_alpha(compound,[NOx,UV,seed,reset]). The observations will be fit using SOM_v1, assuming dynamic mass transfer.
    a. The inputs are
        i. Compound always = "aPinene"
        ii. UV can be either "low" or "high"
        iii. Seed can be "low", "med" or "high"
        iv. Reset is by default 0, meaning no reset. If Reset = 1, the results matrices are re-created and set to default values.
    b. The kwall and alpha values to be considered are hard wired in the code and can be changed. The min and max values considered are specified, along with the total number of points between (and including) the min/max values

   c. The maximum number of iterations per (kwall,alpha) pair is hard wired at 20
   d. Saved information includes
      i. ChiSq_matrix = the V_chisq results from fitting
      ii. FitResults_matrix = the SOM fit parameters determined
      iii. FitError_matrix = indicates whether an error occurred
      iv. FitQuitReason_matrix = indicates the reason that the fit ended, which can be used to identify which (kwall,alpha) pairs led to the max iterations being reached (or some other error)
   e. Note: when alpha > 0.1, the fitting proceeds very slowly because SOM runs very slowly (due to the need for a small time-step to deal with fast mass transfer). However, alpha = 0.1 typically gives very similar results to alpha = 1 because gas-particle mass transfer is sufficiently fast when alpha = 0.1 that the gas and particles (approximately) reach equilibrium on short (< 1 min) timescales. Thus, the code is currently hardwired to use alpha_max = 0.1, but then to add one additional run using alpha = 1 but where equilibrium partitioning is assumed.
2. **BatchFit_SOM_aP2016():** All of the experiments can be individually fit using the function BatchFit_SOM_aPinene2016(). This function loops over the different experimental conditions (i.e. UV/seed combinations) and runs BatchFit_SOM_kwall_alpha() for each condition. The resulting results matrices have "_XY" appended to the end, where X = the UV condition (either L or H) and Y = the seed condition (L, M or H). The user can add an optional additional suffix to the results matrix names by changing the "special_suffix" string. If you have this as "" then no suffix is appended. The use of special_suffix can be helpful if you want to run a batchfit for alternative conditions (e.g. using the Jimenez/Krechmer Cw parameterization).
3. **BatchFit_SOM_kwall_alpha_Refit:** If you have performed a batch fit for a single experiment and find that one (or more) of the (kwall,alpha) pairs gave a particularly bad result you can rerun the fitting and update the results matrices for just these (kwall,alpha) pairs. This function relies on the "Refit_matrix" wave. Re-fitting will be performed for only those elements that are set to 1. One can use this function to restart a run that was aborted early or to redo some select subset of (kwall,alpha) pairs. This can only be run after BatchFit_SOM_kwall_alpha has been run at least once in "reset" mode, as it relies on waves having already been created. You can use the "special_suffix" string.
4. **BatchFit_SOM_aP2016_compare:** Once you have run BatchFit_SOM_aP2016() to determine the best fit for every (kwall,alpha) pair for a given experiment, you can run this function to see how well these best fits reproduce the complementary experiment for a given UV condition. For example, the SOM parameter sets determined from fitting of the "low UV, low SA" experiment are used to simulate the complementary "low UV, high SA" experiment. Chi square values are calculated for each of the experiments and for the combination. You can use the "special_suffix" string.
   a. For low UV, two matrices are created that have 3 layers. Layer 0 corresponds to the ChiSq values for the experiment to which the SOM fit was originally performed (low or high SA). Layer 1 corresponds to the ChiSq values for the complementary experiment (high or low SA). And Layer 2 corresponds to the sum of the ChiSq's from the first two layers.

b. For high UV, it is similar to low UV except there are now 4 layers to account for there being three seed conditions (low, medium, high), with the last layer the sum over the first three.
c. The resulting chisq matrices are named according to the data set that was originally fit. For example, ChiSq_matrix_LL_compare, ChiSq_matrix_LH_compare, etc., and where the "_compare" indicates that you are comparing between experiments.

To perform a batch "optimization" for every experimental condition, the following should be done.

1. Run BatchFit_SOM_aP2016(). This will run through every condition (UV,Seed pair) and determine a matrix of chi-square values associated with each (kwall,alpha) pair. The best-fit SOM parameters for each (kwall,alpha) pair are also stored. The results waves are appended with e.g. "_LL" to indicate the UV and seed condition.
    a. This runs the BatchFit_SOM_kwall_alpha() function automatically for each UV,seed pair
    b. This may take a very long time to complete
2. Run BatchFit_SOM_aP2016_compare() for each condition considered. This function is set up to operate only on a single experimental condition, so will need to be run 5 times (or you can write a function that calls this 5 times and loops through the different experimental conditions).
3. Make image plots showing the results (i.e. the chi-square matrices) for each condition.

When starting out, it may be useful to perform the optimization using a limited set of values for (kwall,alpha). You can examine the general behavior using a coarse resolution, and then increase the resolution (i.e. number of points) later. The number of points is set using the nsteps_kwall and nsteps_alpha variables in BatchFit_SOM_kwall_alpha(). Starting values of 5 for both seems reasonable, but higher resolution (or a more limited range) should ultimately be considered. Using nsteps_kwall = 13 and nsteps_alpha = 14 seem reasonable for "high resolution" optimization.

The optimization process is slow. During an optimization some fits proceed fast while others progress only slowly, due either to a greater number of iterations being required or a smaller timestep being required. You can run multiple optimizations (e.g. for the different cases above) simultaneously by running multiple instances of Igor. Igor does not operate like e.g. Excel in that you cannot by default have multiple instances open. However, this can be circumvented by holding ctrl while opening a file. This opens up a second instance by running the executable again. You can actually have the same file open twice, although this is not recommended. Instead, it is recommended that you duplicate your file (*.pxp) of interest and then open the original and the copy. Most computers seem to handle two instances reasonably well, and since most computers now have dual cores they can multi-task so running two optimizations does not slow down one. However, if you start to run too many instances of Igor things can get unstable. Igor does not recover like Microsoft products, so if something crashes and you have not saved recently then your work is lost. (You could add the SaveExperiment command to run at various times during the optimization so that the file is automatically saved. Just don't have this run too often.) In any case, you could, for example, perform the "default" optimization simultaneous with the "Krechmer alternative" optimization.

When doing a "batch fit" the "abort" button does not work. Normally, the abort button (that pops up in the lower-left of Igor when a procedure is running) will stop the function from executing. But this doesn’t work when doing fitting (for some reason I still don’t understand). However, you can force a quit by using ctrl+break or ctrl+pause. (On my laptop it is ctrl+break. On my keyboard it is ctrl+pause, I think because there is no break button and pause = break on the keyboard.) This can be useful if you want to force stop an optimization to make some change and then restart.