DGOSPREY

User Manual

Austin Ladshaw

November 14th, 2016

Table of Contents

Introduction

The following manual is intended to guide new users in the uses and applications of the Discontinuous Galerkin Off-gas SeParation and REcoveY (DGOSPREY) fixed-bed adsorption model. This software was developed using the Idaho National Laboratory (INL) Multi-physics Object-Oriented Simulation Environment (MOOSE). More information on MOOSE can be found at www.mooseframework.org.

DGOSPREY was designed to solve the mass and energy balance equations that result from gas-phase adsorption in cylindrical fixed-bed adsorption columns. The basic equations that the model solves are given below (equations 1 and 2). Equation 1 gives the mass balance of each gas species in the system and equation 2 gives the energy balance for the entire gas phase.

$$\varepsilon_b \frac{\partial C_i}{\partial t} + \nabla\left(\varepsilon_b v C_i\right) = \nabla\cdot\left(\varepsilon_b D_z \nabla C_i\right) - \rho_b \frac{\partial q_i}{\partial t} \tag{1}$$

$$\left(h_g \rho \varepsilon_b + h_s \rho_b\right)\frac{\partial T}{\partial t} + \nabla\left(h_g \rho \varepsilon_b v T\right) = \nabla\cdot\left(\varepsilon_b K_z \nabla T\right) + \rho_b \sum_i \frac{\partial (Q_{st,i} q_i)}{\partial t} \tag{2}$$

Each term in the above equations is given its own unique kernel in the MOOSE environment. Those kernels are used to collectively describe the system of equations that we seek an approximate solution to. Kernels for the time derivative, advection terms, and dispersion terms are unique, i.e. there is only one kernel to use for each term. However, there are many different kernels that can be used for adsorption. The user, at runtime, must specify the specific type of adsorption kernel used in a particular simulation.

Boundary conditions for DGOSPREY differ for mass and energy balances, but always take the same basic form. Equations 3 and 4 show the boundary conditions for the flux of mass and heat from the open ends fixed-bed, while equation 5 shows the boundary condition for the flux of heat from the walls of the fixed-bed.

$$D_z \frac{\partial C_i}{\partial z} = -v\left(C_{in/out,i} - C_i\right) \tag{3}$$

$$K_z \frac{\partial T}{\partial z} = -v h_g \rho \left(T_{in/out} - T\right) \tag{4}$$

$$K_z \frac{\partial T}{\partial r} = -U_w \left(T_{wall} - T\right) \tag{5}$$

Getting Started

Installing MOOSE

MOOSE can be installed on nearly any computer running Mac OS X or a linux operating system. Detailed instructions on installing for your particular system may be found at www.mooseframework.org/getting-started/. Information such as minimum system requirements and how to setup the MOOSE framework can also be found at that web address.

After setting up your computer environment, you need to download/clone MOOSE from the GitHub repository. To do this, start by opening up a command terminal in your computer, then enter in the following commands:

```
mkdir ~/projects

cd ~/projects

git clone https://github.com/idaholab/moose.git

cd ~/projects/moose

git checkout master
```

The first command creates a folder called "projects" under your user directory. Next, you are moving into the directory using the cd command. Then, you are downloading all of the MOOSE files into a new sub-directory called "moose". Finally, you move into that sub-directory and instruct git to move onto the master branch of moose.

Next, you will need to build the libmesh libraries, which MOOSE is dependent on. To do this, you will need to move into the "scripts" directory under the "moose" directory that was just created, and then run the update_and_rebuild_libmesh script. Those commands are as follows:

```
cd ~/projects/moose/scripts

./update_and_rebuild_libmesh.sh
```

NOTE: this process may take several minutes to and hour to complete.

Next, you will need to build the MOOSE from source code using the make command. This will complete the construction of the MOOSE framework. After completion, you should run tests to verify that MOOSE has installed correctly. Use the following commands to build and test MOOSE:

```
cd ~/projects/moose/test
```

```
make –j8

./run_tests –j8
```

The –j8 options from the above commands indicate the number of jobs you are requesting to run or the number of processors to use during the build and test phases. In general, you do not want to specify a number of jobs greater than the number of processors your computer has.

NOTE: building MOOSE can take a very long time (several hours for the first build).

Updating MOOSE

It is very important that you keep your MOOSE build up-to-date with the master branch on the GitHub repository. The primary developers at INL will periodically update MOOSE and libmesh. To stay up-to-date with the software updates, you can join the MOOSE user's group (moose-users@googlegroups.com) by sending an email to moose-users+subscribe@googlegroups.com. More information is available at www.mooseframework.org/getting-started/.

To update MOOSE, open a terminal and navigate to the ~/projects/moose directory. Then, use the git commands to update your local copy of the master branch. The commands for this are as follows:

```
cd ~/projects/moose

git fetch origin

git rebase origin/master
```

NOTE: after updating MOOSE, you will need to rebuild the binaries using the make command discussed in "Installing MOOSE."

To update libmesh, simply rerun the update_and_rebuild_libmesh script as was done in the "Installing MOOSE" section.

NOTE: after updating libmesh, you will also need to rebuild the MOOSE binaries using the make command.

Installing DGOSPREY

The DGOSPREY project has its own GitHub repository with two major branches: (i) a master branch "master" and (ii) a development branch "devel." The repository can be found and explored on the web at https://github.com/aladshaw3/dgosprey.

Generally, you will only want to use the "master" branch, as it will always be the most stable version of the software.

To install DGOSPREY, open up a terminal and navigate to the projects directory where the moose directory is located. Then, clone the DGOSPREY project and switch onto the master branch. The commands for this are as follows:

```
cd ~/projects

git clone https://github.com/aladshaw3/dgosprey.git

cd ~/projects/dgopsrey

git checkout master
```

After downloading the project files, you will need to build binaries and run tests to ensure that the install completed correctly. To do this, enter the following commands into the terminal from the dgosprey directory:

```
make –j8

./run_tests –j4
```

All tests should pass. If they do not, then please report any errors to aladshaw3@gatech.edu. Please put "DGOPREY" in the subject of any emails so ensure that they are not missed or deleted.

After building DGOSPREY and successfully running all tests, you are now ready to begin using DGOSPREY to run fixed-bed adsorption simulations. However, there are a few more considerations that you may want to read about before continuing. Please review "Updating DGOSPREY" and "Branching DGOSPREY" before starting.

Updating DGOSPREY

There are two different commands/group of commands you can use to keep up-to-date with DGOSPREY: (i) the pull command and (ii) the rebase command.

Using the pull command will update your local copy of whatever branch you are currently on with the global version of that branch saved in the repository. For example, to pull down any changes on the master branch, open a terminal and perform the following commands:

```
cd ~/projects/dgosprey

git checkout master

git pull
```

NOTE: these sets of commands can result in an error if there is a discrepancy between your local master copy and the global master copy that cannot be automatically resolved by git. For instance, if you have a local change that has not been committed, then this may result in an error.

The git pull command should be used on your own private branch (see Creating New Branches) or as a quick way to get updates from the master branch. If this command does not work, you can rebase the branch to the global copy by entering the following commands:

```
cd ~/projects/dgosprey

git fetch origin

git rebase origin/master
```

WARNING: using the rebase command will ERASE any changes you have made to your local version of the repository, regardless of whether or not those changes were committed.

As a general rule-of-thumb, you should not make any changes to the master branch unless you really know what you are doing. If you want to make code changes or add new files, such as input files, then you should work on your own branch of DGOSPREY. Creating a branch of DGOSPREY is discussed in the next section.

Branching DGOSPREY

Creating New Branches

Each git repository can handle a number of different code branches that can be used to keep track of several different versions of the code. It is good practice to always keep your own branch of the source code that you will keep up-to-date with the master branch so that you do not make code changes that end up creating errors in the master branch. A general user of the software should NEVER edit the master branch.

To view a list of the available branches in DGOSPREY, open a terminal and navigate to the dgosprey project folder, then execute the git branch command as follows:

```
cd ~/projects/dgosprey

git branch
```

This will display a list of the currently tracked branches of the project and place an * next to the name of the branch in which you are currently working on. To create your own branch and switch to that branch, type in the following commands:

```
git branch branch_name

git checkout branch_name
```

NOTE: you will replace the option branch_name with the name you want to give the branch. DO NOT use spaces when naming the branch. All branch names must be unique.

After creating your own branch you can edit the source files of the project without having to worry about breaking the source code on the master branch. To copy your new local branch to the global repository, you enter the following:

```
git push --set-upstream origin branch_name
```

Updating Branches

Before working on a branch, you will want to make sure that your local branch is up-to-date with any changes on the global branch. The instructions for this are the same as those for "Updating DGOSPREY". However, you would need to replace the keyword "master" with the name of the branch that you are updating.

Committing Changes

After making changes to your own local copy of your own branch, you can save those changes to the global repository. To do this involves committing the changes you have made to your branch and pushing those changes onto the GitHub repository.

First, you will need to make sure you are on the correct branch before making any changes to source code files. Go to the dgosprey directory and check the current branch by entering the following commands to the terminal:

```
cd ~/projects/dgosprey

git branch
```

NOTE: the branch that is displayed with an * next to the name is the current branch. Git will always default to the branch you were on last time you were working in the terminal.

You can check the status of the current branch by typing in the following:

```
git status
```

NOTE: this will display any changes to existing files, new files created, as well as the current working branch you are on.

If you need to add git tracking to any new files, you can do so by using the git add command as follows:

```
git add path/to/file/new_file
```

NOTE: path/to/file is the route to the new_file from the current directory. If you have many files to add, then you can just give the directory to where the new files are.

After adding any new files, you will need to commit all your changes to your local copy of the branch, and then push those changes to the global repository. Those commands are as follows:

```
git commit –a –m “This is the commit message.”

git push
```

NOTE: you are required to provide a message in quotation marks when making a commit.

Merging Branches

Merging of a branch is done to keep different branches of the project up-to-date with each other. Typically, you will want to make sure that you merge your own branch with the master branch to ensure that any changes made to the master branch are also reflected in your own branch. To do this requires a couple of different steps.

First, you will want to update your local copy of the master branch. Open a terminal and navigate to the dgosprey project directory and switch to the master branch, then pull down any changes there might be on the global master branch. Those commands are as follows:

```
cd ~/projects/dgosprey

git checkout master

git pull
```

After updating the master branch, you will need to switch to your own branch, update your branch (if needed), and then execute the merge command. Those commands are as follows:

```
git checkout branch_name

git pull

git merge master
```

NOTE: the "git pull" command is only needed if you are also updating your local copy of your own branch with the global copy of that branch.

WARNING: if there are any changes you have made to your own local copy of your branch AND have not committed those changes, then this will result in a fatal error when trying to merge. Make sure that any and all changes on your own branch are committed before attempting to merge. See "Committing Changes" for more information.

Running DGOSPREY

Terminal Commands

It is highly recommended that you run all DGOSPREY simulations using the terminal commands. This is the simplest and quickest way to run the program, regardless of computer architecture. However, there is a graphical user interface (GUI), distributed with MOOSE, which can be used to run the simulations. See "Peacock" for more information on using the GUI.

Before running any simulations, you will need to make sure that the software is up-to-date and rebuild the binaries if any changes have been made (review "Installing DGOSPREY" and "Updating DGOSPREY" for more information). Once the software is updated and ready, you can run simulations by typing in the following commands from the dgosprey project directory.

```
./dgosprey-opt -i path/to/input/file.i
```

NOTE: the option after –i is the path to the input file for a particular simulation. All input files end with a .i extension.

The above command will run the DGOSPREY using a single computer processor. To make use of multiple processors, you need to prefix the application with the appropriate mpi commands. For example, to run DGOSPREY with 8 processors, you would enter the follow command:

```
mpiexec --n 8 ./dgosprey-opt -i  path/to/input/file.i
```

NOTE: to change the number of processors, simply change the value of the number following the --n argument.

Any output produced from the execution of DGOSPREY will be placed in the same directory as the input file that was just run. Output file names will be named after the input file, but will have "_out" added to the end of the name. Extensions for output files will either be .csv (for excel/spreadsheet output) or .e (for mesh/exodus output) and will depend on the options chosen by the user and indicated in the input file (see "Input Files" for more information).

For example, if the input file run was file.i and the output type selected was csv, then the output file would be named file_out.csv.

NOTE: more than one output file type can be created and saved in the input file directory.

Peacock

Peacock is the GUI that is bundled with MOOSE and can be used to run MOOSE based applications, such as DGOSPREY. Although this application does provide a GUI for running DGOSPREY, the invocation of peacock must still be done through the command terminal. Additional information on peacock can be found at www.mooseframework.org/wiki/Peacock/.

Before you can call peacock, you will need to add the executable to your bash PATH. To do this, navigate to your home directory and open the .bash_profile (Mac OS X) or .bashrc (linux) file for editing. Then, at the end of that file you will need to add the following line:

```
export PATH=/path/to/moose/gui:$PATH
```

NOTE: the .bash_profile and .bashrc files are hidden files in your home directory. To edit them you will need to open a terminal, navigate to your home directory, and then either use the open command (Mac OS X only) or vi editor to edit and save the files.

```
cd ~/

open .bash_profile
```

After adding this line to the file, save the file and exit the terminal. To make sure that the PATH was correctly updated, you can open another terminal and type the following:

```
echo $PATH
```

The output to the console should show a list of all application paths on your computer. If the /path/to/moose/gui is not there, then exit all terminals and try again.

NOTE: the /path/to/moose/gui will vary from computer to computer depending on where your home directory is located and how it is named. To find exactly what your path is, you can open a terminal, navigate to the moose gui folder, and type the pwd command as follows:

```
cd ~/projects/moose/gui

pwd
```

Output from that command will be the exact path to the moose gui folder. You can copy and paste that directory as the /path/to/moose/gui line in the export PATH command (see above).
After the peacock executable is on your PATH, you can call the application from anywhere within the terminal. To run DGOSPREY using peacock, navigate to the dgosprey project folder then type peacock in the terminal:

```
cd ~/projects/dgosprey

peacock
```

NOTE: the first invocation of peacock may take some time.

After the main window opens, you can either create your own custom input file using the peacock interface or open and existing input file to run. Additionally, you can just tell peacock what input file to open when invoking the application. To do this, simply add the path and file as an option in the terminal as follows:

```
peacock path/to/file.i
```

This command will launch peacock and automatically open and load the given input file. For more detailed instructions on running and using peacock, please visit www.mooseframework.org/wiki/Peacock/.

NOTE: whenever peacock is used to run simulations, it will automatically create a lot of junk files in your input file or executable directory. These temporary files are just junk that is no longer needed after completing your simulations in peacock. You should delete these files after using peacock to reduce clutter.

DGOSPREY Input

Input File Structure

The input files for DGOSPREY follow the same basic structure and format as any input files for other MOOSE based applications. For some input syntax information on input files, please refer to www.mooseframework.com/docs/syntax/moose/.

Detailed information for MOOSE input files is found in the MOOSE workshop manual at www.mooseframework.org/static/media/uploads/docs/main.pdf. Input example files are at www.mooseframework.org/static/media/uploads/docs/examples.pdf.

For specific information about DGOSPREY input files, you may refer to the below sub-categories of "Input File Structure." These topics will cover the arguments that are needed (or useful) to run DGOSPREY simulations. Unless otherwise specified, you should treat all of the following input arguments to be necessary to run a successful DGOSPREY simulation.

GlobalParams

You must specify the length of the fixed-bed column (in cm). For example:

```
length = 12.7
```

If you are using the SCOPSOWL kinetics kernels, then you must also specify the initial time step size (in hours). For example:

```
initial_dt = 0.0118
```

Problem

You must specify that the coordinate system for the problem is cylindrical. This is because the fixed-bed columns are cylindrical in shape, but the mesh we generate for the problem is represented as a two-dimensional plane.

```
coord_type = RZ
```

Mesh

MOOSE will generate the mesh, but you need to specify the dimensions of the mesh. In the RZ coordinate system, the x-axis is transformed into the r-axis and the y-axis is transformed into the z-axis. The origin of this new coordinate system (i.e. x = 0 and y = 0) corresponds to the center of the entrance of the column. Example input for the mesh information is as follows:

```
type = GeneratedMesh
dim = 2
nx = 10
ny = 40
xmin = 0
xmax = 1.27
ymin = 0
ymax = 12.7
```

NOTE: xmin and ymin should always be 0 and the dimension should always be 2. nx and ny are the number of element divisions in the x and y axis.

IMPORTANT: xmax should be the inner radius of the column in cm and ymax should always have the same value as length from GlobalParams.

Variables

The variables block of the input file is where all the non-linear variables of the system are declared. This includes each individual gas species, the temperature of the gas in the column, and the temperature of the walls of the column. All need to be declared as order CONSTANT and family MONOMIAL. Those keywords signal that MOOSE use Discontinuous Galerkin methods.

For the wall and gas temperatures, you must also specify the initial temperatures in degrees Kelvin (K). Example input for DGOSPREY is as follows:

```
[./H2O]
      order = CONSTANT
      family = MONOMIAL
[../]

[./wall_temp]
      order = CONSTANT
      family = MONOMIAL
      initial_condition = 298.15
[../]

[./gas_temp]
      order = CONSTANT
      family = MONOMIAL
      initial_condition = 298.15
[../]
```

AuxVariables

Auxiliary variables are variables that are not directly solved for iteratively through MOOSE, but are determined via the solutions to the non-linear variables defined above. These variables do not contribute directly to the residuals in MOOSE and are defined through a different set of kernels called Auxiliary Kernels. The auxiliary variables for DGOSPREY include total gas pressure in the column, ambient temperature around the column, adsorbed concentrations of the gas species, adsorption heats, and a perturbation value for the adsorbed concentrations.

Just like with the variables, all auxiliary variables must also be CONSTANT and MONOMIAL. Additionally, the auxiliary variables all require an initial condition. Examples for the auxiliary variable DGOSPREY input is shown below:

```
[./total_pressure]
      order = CONSTANT
      family = MONOMIAL
      initial_condition = 101.35
[../]

[./ambient_temp]
      order = CONSTANT
      family = MONOMIAL
      initial_condition = 298.15
[../]

[./H2O_adsorbed]
      order = CONSTANT
      family = MONOMIAL
      initial_condition = 0
[../]

[./H2O_perturbed]
      order = CONSTANT
      family = MONOMIAL
      initial_condition = 0
[../]

[./H2O_ads_heat]
      order = CONSTANT
      family = MONOMIAL
      initial_condition = 0
[../]
```

NOTE: the units for total pressure must be given in kPa, units for adsorption and perturbation must be given in mol/kg, and units for adsorption heat are J/kg.

ICs

Initial condition kernels (ICs) are used to set the initial values of non-linear variables in the mesh based on some given functions. In DGOSPREY, the ICs are used to establish the concentrations of the gases in the column prior to simulation of the fixed-bed. To setup these initial conditions requires parameters such as initial mole fraction of the gas species, initial total pressure, and initial gas temperature. You must also identify which non-linear variable the ICs correspond to. For example, to set the initial gas phase concentration of water vapor in the column to zero would require the following:

```
[./H2O_IC]
      type = ConcentrationIC
      variable = H2O
      initial_mole_frac = 0
      initial_press = 101.35
      initial_temp = 298.15
[../]
```

NOTE: units for pressure and temperature must be kPa and K, respectively.

IMPORTANT: each gas species must have their own ICs and the sum of the initial mole fractions should always be 1. Failure to follow this will likely result in simulation errors.

Kernels

Kernels are responsible for adding to the non-linear residuals in the system of equations that MOOSE seeks to solve. Each kernel will represent some piece of the physics/chemistry in our system for each non-linear variable. A more detailed discussion on the individual kernels in DGOSPREY can be found either in the Reference Manual (dgosprey/doc/refman.pdf), the source/header files, or in the "Kernels" section of this manual.

The gaseous species in the system require accumulation, dispersion, advection, and mass transfer kernels. However, for the case of non-adsorbing/inert gas species, you can leave out the mass transfer kernel. For water vapor, inclusion of these kernels would be as follows:

```
[./accumH2O]
      type = BedMassAccumulation
      variable = H2O
      index = 2
[../]

[./diffH2O]
      type = GColumnMassDispersion
      variable = H2O
      index = 2
[../]

[./advH2O]
      type = GColumnMassAdvection
      variable = H2O
[../]

[./massTransH2O]
      type = AdsorptionMassTransfer
      variable = H2O
      index = 2
[../]
```

NOTE: several of the above kernels require an "index" parameter. The value of the index for a gas species is dependent on the order in which the species coupled in the "Materials" block. That ordering MUST be the same throughout a given input file, as it is used to denote which species is which in the main kernels.

IMPORTANT: indexing will always start from 0!

There are four kernels that must be included for the gas phase temperature variable: accumulation, dispersion, advection, and adsorption heat. Those kernels are very similar to the gas concentration kernels, but with some minor differences in parameters and usage. For example, for a system composed of N_2, O_2, and H_2O gases, with adsorption heat variables from each species, the input would look like the following:

```
[./heatAccum]
      type = BedHeatAccumulation
      variable = gas_temp
[../]

[./heatDiff]
      type = GColumnHeatDispersion
      variable = gas_temp
[../]

[./heatAdv]
      type = GColumnHeatAdvection
      variable = gas_temp
[../]

[./heatAdsorption]
      type = AdsorptionHeatAccumulation
      variable = gas_temp
      solid_heats = 'N2_ads_heat O2_ads_heat H2O_ads_heat'
[../]
```

NOTE: the variable names and coupled variable names (i.e. solid_heats) must be the same names of the variables declared in "Variables" and "AuxVariables" blocks.

Lastly, you must also include the necessary kernels for the wall temperature variable. There are three kernels that must be included: accumulation, bed-wall heat transfer, and wall-ambient heat transfer. Those kernels would be put into an input file as follows:

```
[./wallAccum]
      type = WallHeatAccumulation
      variable = wall_temp
[../]

[./wall_bed_trans]
      type = BedWallHeatTransfer
      variable = wall_temp
      coupled = gas_temp
[../]

[./wall_amb_trans]
      type = WallAmbientHeatTransfer
      variable = wall_temp
      coupled = ambient_temp
[../]
```

DGKernels

DGKernels are specialized kernels in MOOSE that allow for access to neighboring element information to construct residuals that will account for fluxes to-and-from nearby elements in the mesh. These kernels are REQUIRED to be used in conjunction with their corresponding dispersion and advection kernels that were prefixed with a "G" from the above "Kernels" section. Mismatching the "DG" and "G" kernels, or leaving out the correct pairs, will result in simulation errors.

Each gas species, as well as the gas temperature variable, will have both a dispersion and advection DGKernel. An example of their usage and syntax for the input file is provide below:

```
[./DGdiff_H2O]
      type = DGColumnMassDispersion
      variable = H2O
      index = 2
[../]

[./DGadv_H2O]
      type = DGColumnMassAdvection
      variable = H2O
[../]

[./DGdiff_heat]
      type = DGColumnHeatDispersion
      variable = gas_temp
[../]

[./DGadv_heat]
      type = DGColumnHeatAdvection
      variable = gas_temp
[../]
```

NOTE: index parameters for the gas species variables must always be the same as all other kernels. If the indices are not consistent throughout the input file, then errors will occur.

AuxKernels

Each auxiliary variable, except ambient temperature, requires an auxiliary kernel to calculate the values of that variable based on the conditions in the fixed-bed column. Ambient temperature does not require an auxiliary kernel because we typically consider it to not change over time. However, you can write your own auxiliary kernel for ambient temperature to allow it to vary with specific conditions.

For total pressure and the heats of adsorption, there is currently only one applicable auxiliary kernel for calculating their respective values. Total pressure in the column

is calculated from the gas concentrations of all species in the column, as well as the temperature in the column, using ideal gas law. The total pressure, for a system of N_2, O_2, and H_2O gases, is specified as follows:

```
[./pressure]
      type = TotalColumnPressure
      variable = total_pressure
      temperature = gas_temp
      coupled_gases = 'N2 O2 H2O'
[../]
```

NOTE: the coupled gases are the names of the gas phase variables and must be given in the same order everywhere in the input file whenever they need to be specified.

NOTE: the "index" parameter from the "Kernels" and "DGKernels" for the gas species comes from the order in which the coupled gases are given. In this example, H2O is the 3rd species listed in the coupled gases parameter; therefore its index is 2.

The adsorption heats contributed by each gas are also given as auxiliary kernels. In DGOSPREY, we calculate heats of adsorption for each gas species from the adsorbed concentrations of those species and the gas phase concentrations of the corresponding species. That information is used to calculate the heat of adsorption from the isotherms for each species, which is done in the Multicomponent Adsorption Generalized Prediction of Isothermal Equilibria (MAGPIE) sub-routines. Declaring the kernel in the input file is done as the following for each adsorption heat variable:

```
[./ads_heat_H2O]
      type = MAGPIE_AdsorptionHeat
      variable = H2O_ads_heat
      solid_conc = H2O_adsorbed
      index = 2
[../]
```

NOTE: for more information on MAGPIE or the isotherms utilized, please refer to the following publications:

Ladshaw, A.; Yiacoumi, S.; Tsouris, C., "A generalized procedure for the prediction of multicomponent adsorption equilibria", *AIChE J.*, 61, 8, 2600-2610, 2015.

Ladshaw, A.; Yiacoumi, S.; Tsouris, C.; DePaoli, D.W., "Generalized Gas-Solid Adsorption Modeling: Single-Component Equilibria", *Fluid Phase Equilibria*, 388, 169-181, 2015.

Each adsorbing gas species must have an auxiliary variable for adsorption, as well as adsorption and perturbation auxiliary kernels. There are several different kernels available for use: MAGPIE adsorption, linear driving force (LDF) adsorption (two types), and bi-porous pellet kinetic adsorption (SCOPSOWL). You are free to use any

type of adsorption auxiliary kernel you want for a given simulation, and the results of a simulation will depend on which kernel you use here. However, you MUST make sure that you use the same kernel type for both the adsorption and perturbation calculations. Each of these different kernels will be discussed in further detail in the "Adsorption Kernels" section of this manual.

To use MAGPIE equilibria adsorption, you would give adsorption and perturbation kernels in the input file as follows:

```
[./H2O_adsorption]
      type = MAGPIE_Adsorption
      variable = H2O_adsorbed
      index = 2
[../]

[./H2O_perturbation]
      type = MAGPIE_Perturbation
      variable = H2O_perturbed
      index = 2
[../]
```

To use LDF kinetic adsorption with a constant LDF parameter, you would give adsorption and perturbation kernels in the input file as follows:

```
[./H2O_adsorption]
      type = MAGPIE_ConstLDF_Adsorption
      variable = H2O_adsorbed
      ldf_coeff = 1
      index = 2
[../]

[./H2O_perturbation]
      type = MAGPIE_ ConstLDF_Perturbation
      variable = H2O_perturbed
      ldf_coeff = 1
      index = 2
[../]
```

NOTE: the coefficient used for the LDF rate of adsorption must be the same value in the adsorption and perturbation kernels. Units for that parameter are 1/hour.

To use LDF kinetic adsorption with a variable LDF parameter, you would give adsorption and perturbation kernels in the input file as follows:

```
[./H2O_adsorption]
      type = MAGPIE_MaterialLDF_Adsorption
      variable = H2O_adsorbed
      index = 2
[../]

[./H2O_perturbation]
```

```
      type = MAGPIE_ MaterialLDF_Perturbation
      variable = H2O_perturbed
      index = 2
[../]
```

To use SCOPSOWL kinetic adsorption, you would give adsorption and perturbation kernels in the input file as follows:

```
[./H2O_adsorption]
      type = Scopsowl_Adsorption
      variable = H2O_adsorbed
      index = 2
[../]

[./H2O_perturbation]
      type = Scopsowl_Perturbation
      variable = H2O_perturbed
      index = 2
[../]
```

NOTE: to use SCOPSOWL adsorption requires an additional material properties object called "ScopsowlProperties." See "Materials" section for more information.

BCs

Boundary conditions (BCs) for DG methods in MOOSE are based on flow rates into and out of the boundary elements of the mesh. As such, there is really only one type of boundary condition kernel to use for each non-linear variable. However, we can apply that boundary condition in limited capacity in order to try to preserve the initial shape of the concentration or temperature wave front.

For the standard flux boundary conditions, you would the following BC kernels for all gas species and the gas temperature:

```
[./H2O_flux]
      type = DGMassFluxBC
      variable = H2O
      boundary = 'top bottom'
      input_temperature = 298.15
      input_pressure = 101.35
      input_molefraction = 0.00163
      index = 2
[../]

[./Temp_Inlet_Flux]
      type = DGHeatFluxBC
      variable = gas_temp
      boundary = 'top bottom'
      input_temperature = 298.15
[../]
```

```
[./Temp_Wall_Flux]
      type = DGColumnWallHeatFluxBC
      variable = gas_temp
      boundary = 'right left'
      wall_temp = wall_temp
[../]
```

NOTE: the values of the input temperature and input pressure parameters must be the same for all declared boundary conditions. Units for temperature and pressure are K and kPa, respectively.

For the standard flux boundary conditions, you would the following BC kernels for all gas species and the gas temperature:

```
[./H2O_flux]
      type = DGMassFluxLimitedBC
      variable = H2O
      boundary = 'top bottom'
      input_temperature = 298.15
      input_pressure = 101.35
      input_molefraction = 0.00163
      index = 2
[../]

[./Temp_Inlet_Flux]
      type = DGHeatFluxLimitedBC
      variable = gas_temp
      boundary = 'top bottom'
      input_temperature = 298.15
[../]
[./Temp_Wall_Flux]
      type = DGColumnWallHeatFluxLimitedBC
      variable = gas_temp
      boundary = 'right left'
      wall_temp = wall_temp
[../]
```

NOTE: the difference between the standard flux and flux-limited boundary conditions are very minor and will not have a huge impact on the overall simulation results. Flux-limited boundary conditions are made to most closely resemble Dirichlet type BCs at the inlet of the columns, which are actually not available in true Discontinuous Galerkin methods.

Materials

Material properties are where most of the system parameters and other constants will be specified. There are a total of four different material sub-blocks that must be included in every DGOSPREY input file (five total if you are using the SCOPSOWL adsorption kernels). Those materials are: bed properties, flow properties, adsorbent properties, adsorbate properties, and SCOPSOWL properties.

The “BedProperties” material is used to declare physical properties of the fixed-bed column. An example of the necessary sub-block is as follows:

```
[./BedMaterial]
      type = BedProperties
      block = 0
      inner_diameter = 2.54
      outer_diameter = 2.84
      bulk_porosity = 0.421
      axial_conductivity = 6.3E-5
      wall_density = 8.0
      wall_heat_capacity = 0.5
      wall_heat_trans_coef = 6.12
      extern_heat_trans_coef = 6.12
      temperature = gas_temp
      coupled_gases = ‘N2 O2 H2O’
[../]
```

NOTE: the temperature and coupled gases parameters are used to declare the names of the non-linear variables that this material is coupled with.

IMPORTANT: the coupled gases parameters must always have the order of the variables in all objects. This is the order used to determine the “index” parameters that were discussed in “Kernels” and “AuxKernels.”

Table 1: Units of the parameters in BedProperties

Parameter	Units	Parameter	Units
inner_diameter	cm	outer_parameter	cm
bulk_porosity	---	axial_conductivity	J/hr/cm/K
wall_density	g/cm^3	wall_heat_capacity	J/g/K
wall_heat_trans_coef	J/hr/cm^2/K	extern_heat_trans_coef	J/hr/cm^2/K

The “FlowProperties” material is used to declare physical properties of the gas flow in the fixed-bed column. An example of the necessary sub-block is as follows:

```
[./FlowMaterial]
      type = FlowProperties
      block = 0
      molecular_weight = ’28.016 32 18’
      comp_heat_capacity = ‘1.04 0.919 1.97’
      comp_ref_viscosity = ‘0.0001781 0.0002018 0.0001043’
      comp_ref_temp = ‘300.55 292.25 298.16’
      comp_Sutherland_const = ‘111 127 784.72’
      flow_rate = 211680.0
      temperature = gas_temp
      coupled_gases = ‘N2 O2 H2O’
      coupled_adsorption = ‘N2_adsorbed O2_adsorbed H2O_adsorbed’
      coupled_perturbation = ‘N2_perturbed O2_perturbed
                                                H2O_perturbed’
[../]
```

NOTE: the parameters for reference viscosity (comp_ref_viscosity), reference temperature (comp_ref_temp), and Sutherland's constant (comp_Sutherland_const) are used to estimate the overall gas viscosity in the system using the Sutherland equation. More information on the Sutherland equation can be found at the following reference:

Sutherland, W., "The viscosity of gases and molecular force", *Philosophical Mag.*, 36, 507-531, 1893.

IMPORTANT: each parameter, which is represented as a list, must have the values in that list correspond to the correct variable. For example, the molecular weights are listed as 28.016, 32, and 18. These values correspond to the molecular weights of N_2, O_2, and H_2O, respectively, which is the order in which those gases and solids are coupled. Throughout the entirety of the input file, that ordering must be maintained; otherwise simulation errors are likely to occur.

Table 2: Units of the parameters in FlowProperties

Parameter	Units	Parameter	Units
molecular_weight	g/mol	comp_heat_capacity	J/g/K
comp_ref_viscosity	g/cm/s	comp_ref_temp	K
comp_Sutherland_const	K	flow_rate	cm^3/hr

The "AdsorbentProperties" material is used to declare physical properties of the adsorbent pellets in the fixed-bed column. Parameters for binder fraction and crystal radius are optional parameters that are only applicable for bi-porous commercial adsorbents. An example of the necessary sub-block is as follows:

```
[./AdsorbentMaterial]
      type = AdsorbentProperties
      block = 0
      binder_fraction = 0.175
      binder_porosity = 0.27
      crystal_radius = 1.5
      pellet_diameter = 0.236
      macropore_radius = 3.5E-6
      pellet_density = 1.69
      pellet_heat_capacity = 1.045
      ref_diffusion = '0 0 0.8814'
      activation_energy = '0 0 0'
      ref_temperature = '0 0 267.999'
      affinity = '0 0 0'
      temperature = gas_temp
      coupled_gases = 'N2 O2 H2O'
 [../]
```

NOTE: the parameters for reference diffusion, activation energy, reference temperature, and affinity are used to calculate the surface/crystalline surface

diffusivity parameter for each adsorbing species. If reference diffusion is given as 0 for a species, then no adsorption will occur for that species.

IMPORTANT: each parameter, which is represented as a list, must have the values in that list correspond to the correct variable, as before.

Table 3: Units of the parameters in AdsorbentProperties

Parameter	Units	Parameter	Units
binder_fraction	---	binder_porosity	---
crystal_radius	μm	pellet_diameter	cm
macropore_radius	cm	pellet_density	g/cm^3
pellet_heat_capacity	J/g/K	ref_diffusion	$\mu m^2/hr$
activation_energy	J/mol	ref_temperature	K
affinity	---		

The "MagpieAdsorbateProperties" material is used to declare physical/chemical properties of the isotherms for each gas species. Parameters come from the Generalized Statistical Thermodynamic Adsorption (GSTA) isotherm model. An example of the necessary sub-block is as follows:

```
[./AdsorbateMaterial]
      type = MagpieAdsorbateProperties
      block = 0
      number_sites = '0 0 4'
      maximum_capacity = '0 0 11.67'
      molar_volume = '0 0 13.91'
      enthalpy_site_1 = '0 0 -46597.5'
      entropy_site_1 = '0 0 -53.7'
      enthalpy_site_2 = '0 0 -125024'
      entropy_site_2 = '0 0 -221'
      enthalpy_site_3 = '0 0 -193619'
      entropy_site_3 = '0 0 -356'
      enthalpy_site_4 = '0 0 -272228'
      entropy_site_4 = '0 0 -567'
      enthalpy_site_5 = '0 0 0'
      entropy_site_5 = '0 0 0'
      enthalpy_site_6 = '0 0 0'
      entropy_site_6 = '0 0 0'
      temperature = gas_temp
      coupled_gases = 'N2 O2 H2O'
      total_pressure = total_pressure
 [../]
```

NOTE: the maximum number of isotherm equilibria parameters that are allowed is six, which is why the enthalpy and entropy parameters only go up to 6. For species that do not adsorb, you will give the number of sites as 0 and the maximum capacity as zero. For the enthalpies and entropies beyond the number of sites for a given species, you will put in all zeros.

IMPORTANT: each parameter, which is represented as a list, must have the values in that list correspond to the correct variable, as before.

Table 4: Units of the parameters in MagpieAdsorbateProperties

Parameter	Units	Parameter	Units
number_sites	---	maximum_capacity	mol/kg
molar_volume	mol/cm^3	enthalpy_site_#	J/mol
entropy_site_#	J/K/mol		

Additional information on the GSTA isotherm can be found in the following reference:

Ladshaw, A.; Yiacoumi, S.; Tsouris, C.; DePaoli, D.W., "Generalized Gas-Solid Adsorption Modeling: Single-Component Equilibria", *Fluid Phase Equilibria*, 388, 169-181, 2015.

The "ScopsowlProperties" material is used to declare properties needed for the bi-porous pellet kinetics model. Most of the properties are Boolean statements to determine the type of kinetic simulation to run and several of the parameters are optional. An example of the necessary sub-block is as follows:

```
[./ScopsowlMaterial]
      type = ScopsowlProperties
      block = 0
      dirichlet_bc = false
      heterogeneous = true
      surface_diffusion = true
      macro_spheres = true
      micro_spheres = true
      macro_length = 1
      micro_length = 1
      coupled_gases = 'N2 O2 H2O'
      coupled_adsorption = 'N2_adsorbed O2_adsorbed
                                                H2O_adsorbed'
 [../]
```

NOTE: the only required parameters are heterogeneous and surface diffusion. All other parameters will default to the above given values if they are not specified in the input file. Failure to give correct structural information, or any required information, for the particular adsorbent particles of interest will result in simulation errors.

The interpretation, units, and meaning of each of the above parameters for "ScopsowlProperties" are given in the table below:

Table 5: Descriptions and units of the parameters in ScopsowlProperties

Parameter	Information
dirichlet_bc	If false, film mass transfer is included in the simulations for adsorption kinetics. Otherwise, film mass transfer is neglected.
heterogeneous	If false, then it is assumed that the adsorbent pellet is made up of a single phase. Otherwise, the pellet has two phases: a binder matrix and a set of small adsorbent crystals throughout that matrix. **NOTE**: if true is selected, then the micro_spheres parameter is required!
surface_diffusion	If false, then surface diffusion is neglected. Otherwise, the kinetics simulation will include surface diffusion.
macro_spheres	If false, then the adsorbent pellet is assumed cylindrical in shape and you are required to also give the macro_length. Otherwise, the pellet is assumed to be spherical.
micro_spheres	**Only required if heterogeneous = true**. If false, then it is assumed that the adsorbent crystals are cylindrical in shape and you are also required to give the mirco_length. Otherwise, it is assumed that the crystals are spherical in shape.
macro_length	**Only required if macro_spheres = false**. Length of the cylindrical pellets in cm.
micro_length	**Only required if micro_spheres = false**. Length of the cylindrical crystals in μm.

Postprocessors

Postprocessors in MOOSE allow the user to compile specific information about the overall system of interest. They typically involve integrating a particular non-linear variable over the domain, or a sub-domain, of the mesh. For DGOSPREY, the major types of information that we would be interested in are: breakthrough of a gas species at the exit of the column, breakthrough of gas temperature at the column exit, total gas pressure at the column exit, average temperature of the column wall, and average adsorbed concentration in the column. The units of all of these values will be the same units as the non-linear variables themselves: gas concentrations are in mol/L, temperatures are in K, pressure is in kPa, and adsorption concentrations are in mol/kg.

To have DGOSPREY calculate the breakthrough concentrations of species or the breakthrough temperatures and pressures, you will need to specify a set of postprocessors as shown in the example below:

```
[./H2O_exit]
      type = SideAverageValue
      boundary = 'top'
      variable = H2O
      execute_on = 'initial timestep_end'
[../]

[./temp_exit]
      type = SideAverageValue
      boundary = 'top'
      variable = gas_temp
      execute_on = 'initial timestep_end'
[../]

[./press_exit]
      type = SideAverageValue
      boundary = 'top'
      variable = total_pressure
      execute_on = 'initial timestep_end'
[../]
```

NOTE: if you do not specify to execute on initial and timestep_end, then the values will be calculated out of sync/phase in the simulation and create false results.

NOTE: setting boundary to "top" instructs the postprocessor to estimate the values of the variables at the exit of the fixed-bed column.

For calculating the average column wall temperature, you will need to setup a similar postprocessor, but replace the keyword "top" with "right." The right side of the mesh represents the inner edge of the column wall in our coordinate system. An example of the postprocessor input is given below:

```
[./avg_wall_temp]
      type = SideAverageValue
      boundary = 'right'
      variable = wall_temp
      execute_on = 'initial timestep_end'
[../]
```

To have DGOSPREY calculate the average value of a variable through the entire mesh domain, you would use the "ElementAverageValue" postprocessors. For example, to calculate the average adsorption of water vapor in our fixed-bed column would be as follows:

```
[./H2O_ads]
      type = ElementAverageValue
      variable = H2O_adsorbed
      execute_on = 'initial timestep_end'
[../]
```

Executioner

The executioner block in the input file is where all of the solver options are specified. In many cases, MOOSE will provide some default solver options. However, the defaults are not always the most suitable for DGOSPREY simulations. Below are the solver options that are recommended for DGOSPREY simulations:

```
type = Transient
scheme = implicit-euler
nl_rel_tol = 1E-6
nl_abs_tol = 1E-6
nl_rel_step_tol = 1E-10
nl_abs_step_tol = 1E-10
l_tol = 1E-6
l_max_its = 100
nl_max_its = 10
solve_type = newton
line_search = none
start_time = 0
dtmin = 1E-8
petsc_options_iname= '-pc_type -pc_hypre_type -ksp_gmres_restart'
petsc_options_value = 'hypre boomeramg 100'
```

NOTE: although many of these options are not actually used if the solve type is "newton," it is still recommended that you go ahead and have these values set so that if you change "newton" to "pjfnk," the solver will already be setup correctly.

In addition to the above options, you will need to specify an end time for the simulation, the maximum allowable step size, and the "TimeStepper" sub-block, which includes the type of time stepper and the initial time step. For this additional information, there are some general guidelines that you should follow to run a successful DGOSPREY simulation.

First of all, your choice of time step size is very important in the synchronization of the multi-scale physics between micro-scale kinetics and macro-scale mass transport. Recall from "GlobalParams" that you needed to specify an "initial_dt" parameter. The value given for that parameter must ALWAYS match the "dt" parameter of the "TimeStepper" sub-block. Otherwise, the simulation results will be de-synchronized and useless.

In addition to synchronizing the initial time steps, you will also want to take reasonably small time steps throughout the simulation. Generally, smaller time steps will result in higher accuracy. For simulations involving the micro-scale pellet kinetics, it is recommended that your initial time step be no larger than $1/10^{th}$ of the radius of the adsorbent pellets. For example, if the pellet is 0.236 cm in diameter, then your initial time step size is recommended to be 0.0118 hr.

Lastly, you will most likely want to minimize the maximum allowable time step size that DGOSPREY will try to take in the simulation when using a time adaptive time stepper. A good rule-of-thumb for the maximum time step size is to be no larger than 10x the initial time step size. Example input for these timing options are given below:

```
end_time = 60
dtmax = 0.118

[./TimeStepper]
      type = SolutionTimeAdaptiveDT
      dt = 0.0118
[../]
```

In the above example, the simulation will be run to an end time of 60 hours and use a time adaptive stepper that will increase the initial time step up to the value of "dtmax" when successive time steps are successful.

Alternative choices to some of the solver options from above are given below:

```
scheme = bdf2
solve_type = pjfnk
line_search = bt

[./TimeStepper]
      type = ConstantDT
      dt = 0.0118
[../]
```

NOTE: when using "pjfnk" as the solve type, it is recommended that you use a line search algorithm to help smooth convergence of the non-linear iterations.

NOTE: the scheme "bdf2" is usually considered to by more accurate than "implicit-euler," but it is less stable.

NOTE: the "ConstantDT" time stepper function produces very good, smooth looking data, since all time steps are the same size. However, this method will generally require longer simulation times.

Preconditioning

Preconditioning is completely optional and is only useful when using the "pjfnk" solver method (see "Executioner"). Currently, there are no custom sub-routines for preconditioning DGOSPREY simulations. For more information on preconditioning in MOOSE, please refer to the MOOSE manual:

http://mooseframework.org/static/media/uploads/docs/main.pdf

Outputs

The outputs block is where you will specify the types of output files you want to be produced from the DGOSPREY simulation. MOOSE has a variety of different output options available (see http://mooseframework.org/wiki/MooseSystems/Outputs/). In DGOSPREY, it is most common to just use the following three options:

```
exodus = true
csv = true
print_linear_residuals = true
```

NOTE: the above options will produce an exodus file (.e), a csv file (.csv) for the postprocessors requested, and print linear residuals to the console window during runtime. Exodus files maintain all simulation information for all variables across all time steps for the entire mesh. However, they require specific software in order to read and visualize the data (see "Data Visualization").

Input File Examples

The DGOSPREY software comes bundled with a host of sample input files that you can examine and run to become familiar with the input file syntax, input format, and execution of the application. Those files are located under the "input_files" folder of the dgosprey project folder (~/projects/dgosprey/input_files). Provided in the next two sub-sections are sample input files: (i) for water vapor adsorption with an MS3A adsorbent and (ii) Kr and Xe adsorption using a AgZ adsorbent.

NOTE: to comment out lines in the input file, simply place a # before that line.

Water Vapor on MS3A

```
[GlobalParams]

    length = 12.7
    initial_dt = 0.0118

[] #END GlobalParams

[Problem]

    coord_type = RZ

[] #END Problem

[Mesh]

    type = GeneratedMesh
    dim = 2
    nx = 10
    ny = 40
    xmin = 0.0
    xmax = 1.27 #cm
    ymin = 0.0
    ymax = 12.7 #cm

[] # END Mesh

[Variables]

    [./N2]
      order = CONSTANT
      family = MONOMIAL
    [../]

    [./O2]
      order = CONSTANT
      family = MONOMIAL
    [../]

    [./H2O]
      order = CONSTANT
      family = MONOMIAL
    [../]

    [./wall_temp]
      order = CONSTANT
      family = MONOMIAL
      initial_condition = 298.15
    [../]

    [./column_temp]
      order = CONSTANT
      family = MONOMIAL
      initial_condition = 298.15
```

```
    [../]

[] #END Variables

[AuxVariables]

    [./total_pressure]
      order = CONSTANT
      family = MONOMIAL
      initial_condition = 101.35
    [../]

    [./ambient_temp]
      order = CONSTANT
      family = MONOMIAL
      initial_condition = 298.15
    [../]

    [./H2O_Adsorbed]
      order = CONSTANT
      family = MONOMIAL
      initial_condition = 0.0
    [../]

    [./N2_Adsorbed]
      order = CONSTANT
      family = MONOMIAL
      initial_condition = 0.0
    [../]

    [./O2_Adsorbed]
      order = CONSTANT
      family = MONOMIAL
      initial_condition = 0.0
    [../]

    [./H2O_Perturb]
      order = CONSTANT
      family = MONOMIAL
      initial_condition = 0.0
    [../]

    [./N2_Perturb]
      order = CONSTANT
      family = MONOMIAL
      initial_condition = 0.0
    [../]

    [./O2_Perturb]
      order = CONSTANT
      family = MONOMIAL
      initial_condition = 0.0
    [../]

    [./N2_AdsorbedHeat]
```

```
      order = CONSTANT
      family = MONOMIAL
      initial_condition = 0.0
    [../]

    [./O2_AdsorbedHeat]
      order = CONSTANT
      family = MONOMIAL
      initial_condition = 0.0
    [../]

    [./H2O_AdsorbedHeat]
      order = CONSTANT
      family = MONOMIAL
      initial_condition = 0.0
    [../]

[] #END AuxVariables

[ICs]

    [./N2_IC]
      type = ConcentrationIC
      variable = N2
      initial_mole_frac = 0.79
      initial_press = 101.35
      initial_temp = 298.15
    [../]

    [./O2_IC]
      type = ConcentrationIC
      variable = O2
      initial_mole_frac = 0.21
      initial_press = 101.35
      initial_temp = 298.15
    [../]

    [./H2O_IC]
      type = ConcentrationIC
      variable = H2O
      initial_mole_frac = 0.0
      initial_press = 101.35
      initial_temp = 298.15
    [../]

[] #END ICs

[Kernels]

    [./accumN2]
      type = BedMassAccumulation
      variable = N2
      index = 0
    [../]
```

```
[./diffN2]
  type = GColumnMassDispersion
  variable = N2
  index = 0
[../]

[./advN2]
  type = GColumnMassAdvection
  variable = N2
[../]

[./accumO2]
  type = BedMassAccumulation
  variable = O2
  index = 1
[../]

[./diffO2]
  type = GColumnMassDispersion
  variable = O2
  index = 1
[../]

[./advO2]
  type = GColumnMassAdvection
  variable = O2
[../]

[./accumH2O]
  type = BedMassAccumulation
  variable = H2O
  index = 2
[../]

[./H2O_MT]
  type = AdsorptionMassTransfer
  variable = H2O
  solid_conc = H2O_Adsorbed
[../]

[./diffH2O]
  type = GColumnMassDispersion
  variable = H2O
  index = 2
[../]

[./advH2O]
  type = GColumnMassAdvection
  variable = H2O
[../]

[./wallAccum]
  type = WallHeatAccumulation
  variable = wall_temp
[../]
```

```
    [./wall_bed_trans]
      type = BedWallHeatTransfer
      variable = wall_temp
      coupled = column_temp
    [../]

    [./wall_amb_trans]
      type = WallAmbientHeatTransfer
      variable = wall_temp
      coupled = ambient_temp
    [../]

    [./columnAccum]
      type = BedHeatAccumulation
      variable = column_temp
    [../]

    [./columnConduction]
      type = GColumnHeatDispersion
      variable =column_temp
    [../]

    [./columnAdvection]
      type = GColumnHeatAdvection
      variable =column_temp
    [../]

    [./columnAdsHeat]
      type = AdsorptionHeatAccumulation
      variable = column_temp
      solid_heats = 'N2_AdsorbedHeat O2_AdsorbedHeat H2O_AdsorbedHeat'
    [../]

[] #END Kernels

[DGKernels]

    [./dg_disp_N2]
      type = DGColumnMassDispersion
      variable = N2
      index = 0
    [../]

    [./dg_adv_N2]
      type = DGColumnMassAdvection
      variable = N2
    [../]

    [./dg_disp_O2]
      type = DGColumnMassDispersion
      variable = O2
      index = 1
    [../]

    [./dg_adv_O2]
```

```
      type = DGColumnMassAdvection
      variable = O2
    [../]

    [./dg_disp_H2O]
      type = DGColumnMassDispersion
      variable = H2O
      index = 2
    [../]

    [./dg_adv_H2O]
      type = DGColumnMassAdvection
      variable = H2O
    [../]

    [./dg_disp_heat]
      type = DGColumnHeatDispersion
      variable = column_temp
    [../]

    [./dg_adv_heat]
      type = DGColumnHeatAdvection
      variable = column_temp
    [../]

[] #END DGKernels

[AuxKernels]

    [./column_pressure]
      type = TotalColumnPressure
      variable = total_pressure
      temperature = column_temp
      coupled_gases = 'N2 O2 H2O'
    [../]

    [./nitrogen_adsorption]
      type = Scopsowl_Adsorption
      variable = N2_Adsorbed
      index = 0
    [../]

    [./oxygen_adsorption]
      type = Scopsowl_Adsorption
      variable = O2_Adsorbed
      index = 1
    [../]

    [./water_adsorption]
      type = Scopsowl_Adsorption
      variable = H2O_Adsorbed
      index = 2
    [../]

    [./nitrogen_perturbation]
```

```
      type = Scopsowl_Perturbation
      variable = N2_Perturb
      index = 0
    [../]

    [./oxygen_perturbation]
      type = Scopsowl_Perturbation
      variable = O2_Perturb
      index = 1
    [../]

    [./water_perturbation]
      type = Scopsowl_Perturbation
      variable = H2O_Perturb
      index = 2
    [../]

    [./nitrogen_adsorption_heat]
      type = MAGPIE_AdsorptionHeat
      variable = N2_AdsorbedHeat
      solid_conc = N2_Adsorbed
      index = 0
    [../]

    [./oxygen_adsorption_heat]
      type = MAGPIE_AdsorptionHeat
      variable = O2_AdsorbedHeat
      solid_conc = O2_Adsorbed
      index = 1
    [../]

    [./water_adsorption_heat]
      type = MAGPIE_AdsorptionHeat
      variable = H2O_AdsorbedHeat
      solid_conc = H2O_Adsorbed
      index = 2
    [../]

[] #END AuxKernels

[BCs]

    [./N2_Flux]
      type = DGMassFluxLimitedBC
      variable = N2
      boundary = 'top bottom'
      input_temperature = 298.15
      input_pressure = 101.35
      input_molefraction = 0.78863
      index = 0
    [../]

    [./O2_Flux]
      type = DGMassFluxLimitedBC
      variable = O2
```

```
      boundary = 'top bottom'
      input_temperature = 298.15
      input_pressure = 101.35
      input_molefraction = 0.20974
      index = 1
    [../]

    [./H2O_Flux]
      type = DGMassFluxLimitedBC
      variable = H2O
      boundary = 'top bottom'
      input_temperature = 298.15
      input_pressure = 101.35
      input_molefraction = 0.00163
      index = 2
    [../]

    [./Heat_Gas_Flux]
      type = DGHeatFluxLimitedBC
      variable = column_temp
      boundary = 'top bottom'
      input_temperature = 298.15
    [../]

    [./Heat_Wall_Flux]
      type = DGColumnWallHeatFluxLimitedBC
      variable = column_temp
      boundary = 'right left'
      wall_temp = wall_temp
    [../]

[] #END BCs

[Materials]

    [./BedMaterials]
      type = BedProperties
      block = 0
      inner_diameter = 2.54
      outer_diameter = 2.84
      bulk_porosity = 0.421
      axial_conductivity = 6.292E-05
      wall_density = 8.0
      wall_heat_capacity = 0.5
      wall_heat_trans_coef = 6.12
      extern_heat_trans_coef = 6.12
      temperature = column_temp
      coupled_gases = 'N2 O2 H2O'
    [../]

    [./FlowMaterials]
      type = FlowProperties
      block = 0
      molecular_weight = '28.016 32 18'
      comp_heat_capacity = '1.04 0.919 1.97'
```

```
  comp_ref_viscosity = '0.0001781 0.0002018 0.0001043'
  comp_ref_temp = '300.55 292.25 298.16'
  comp_Sutherland_const = '111 127 784.72'
  flow_rate = 211680.0
  temperature = column_temp
  total_pressure = total_pressure
  coupled_gases = 'N2 O2 H2O'
  coupled_adsorption = 'N2_Adsorbed O2_Adsorbed H2O_Adsorbed'
  coupled_perturbation = 'N2_Perturb O2_Perturb H2O_Perturb'
[../]

[./AdsorbentMaterials]
  type = AdsorbentProperties
  block = 0
  binder_fraction = 0.175
  binder_porosity = 0.27
  crystal_radius = 1.5
  pellet_diameter = 0.236
  macropore_radius = 3.5e-6
  pellet_density = 1.69
  pellet_heat_capacity = 1.045
  ref_diffusion = '0 0 0.8814'
  activation_energy = '0 0 0'
  ref_temperature = '0 0 267.999'
  affinity = '0 0 0'
  temperature = column_temp
  coupled_gases = 'N2 O2 H2O'
[../]

[./AdsorbateMaterials]
  type = MagpieAdsorbateProperties
  block = 0
  temperature = column_temp
  total_pressure = total_pressure
  coupled_gases = 'N2 O2 H2O'
  number_sites = '0 0 4'
  maximum_capacity = '0 0 11.67' #mol/kg
  molar_volume = '0 0 13.91' #cm^3/mol
  enthalpy_site_1 = '0 0 -46597.5'
  enthalpy_site_2 = '0 0 -125024'
  enthalpy_site_3 = '0 0 -193619'
  enthalpy_site_4 = '0 0 -272228'
  enthalpy_site_5 = '0 0 0'
  enthalpy_site_6 = '0 0 0'

  entropy_site_1 = '0 0 -53.6994'
  entropy_site_2 = '0 0 -221.073'
  entropy_site_3 = '0 0 -356.728'
  entropy_site_4 = '0 0 -567.459'
  entropy_site_5 = '0 0 0'
  entropy_site_6 = '0 0 0'
[../]

[./KineticMaterials]
  type = ScopsowlProperties
```

```
      block = 0
      dirichlet_bc = false
      heterogeneous = true
      surface_diffusion = true
      coupled_adsorption = 'N2_Adsorbed O2_Adsorbed H2O_Adsorbed'
      coupled_gases = 'N2 O2 H2O'
    [../]

[] #END Materials

[Postprocessors]

    [./N2_exit]
      type = SideAverageValue
      boundary = 'top'
      variable = N2
      execute_on = 'initial timestep_end'
    [../]

    [./O2_exit]
      type = SideAverageValue
      boundary = 'top'
      variable = O2
      execute_on = 'initial timestep_end'
    [../]

    [./H2O_exit]
      type = SideAverageValue
      boundary = 'top'
      variable = H2O
      execute_on = 'initial timestep_end'
    [../]

    [./temp_exit]
      type = SideAverageValue
      boundary = 'top'
      variable = column_temp
      execute_on = 'initial timestep_end'
    [../]

    [./press_exit]
      type = SideAverageValue
      boundary = 'top'
      variable = total_pressure
      execute_on = 'initial timestep_end'
    [../]

    [./wall_temp]
      type = SideAverageValue
      boundary = 'right'
      variable = wall_temp
      execute_on = 'initial timestep_end'
    [../]

    [./H2O_solid]
```

```
      type = ElementAverageValue
      variable = H2O_Adsorbed
      execute_on = 'initial timestep_end'
    [../]

    [./H2O_heat]
      type = ElementAverageValue
      variable = H2O_AdsorbedHeat
      execute_on = 'initial timestep_end'
    [../]

[] #END Postprocessors

[Executioner]

    type = Transient
    scheme = implicit-euler
    nl_rel_tol = 1e-6
    nl_abs_tol = 1e-6
    nl_rel_step_tol = 1e-10
    nl_abs_step_tol = 1e-10
    l_tol = 1e-6
    l_max_its = 100
    nl_max_its = 10
    solve_type = newton
    line_search = none    # Options: default shell none basic l2 bt cp
    start_time = 0.0
    end_time = 60.0
    dtmin = 1e-8
    dtmax = 0.118
    petsc_options_iname = '-pc_type -pc_hypre_type -ksp_gmres_restart'
    petsc_options_value = 'hypre boomeramg 100'

    [./TimeStepper]
      type = SolutionTimeAdaptiveDT
      dt = 0.0118
    [../]

[] #END Executioner

[Preconditioning]

[] #END Preconditioning

[Outputs]

    exodus = true
    csv = true
    print_linear_residuals = true

[] #END Outputs
```

Kr and Xe on AgZ

```
[GlobalParams]

    length = 22.86

[] #END GlobalParams

[Problem]

    coord_type = RZ

[] #END Problem

[Mesh]

    type = GeneratedMesh
    dim = 2
    nx = 10
    ny = 40
    xmin = 0.0
    xmax = 0.8636 #cm
    ymin = 0.0
    ymax = 22.86 #cm

[] # END Mesh

[Variables]

    [./Kr]
      order = CONSTANT
      family = MONOMIAL
    [../]

    [./Xe]
      order = CONSTANT
      family = MONOMIAL
    [../]

    [./He]
      order = CONSTANT
      family = MONOMIAL
    [../]

    [./wall_temp]
      order = CONSTANT
      family = MONOMIAL
      initial_condition = 253.15
    [../]

    [./column_temp]
      order = CONSTANT
      family = MONOMIAL
      initial_condition = 253.15
    [../]
```

```
[] #END Variables

[AuxVariables]

    [./total_pressure]
      order = CONSTANT
      family = MONOMIAL
      initial_condition = 101.35
    [../]

    [./ambient_temp]
      order = CONSTANT
      family = MONOMIAL
      initial_condition = 253.15
    [../]

    [./He_Adsorbed]
      order = CONSTANT
      family = MONOMIAL
      initial_condition = 0.0
    [../]

    [./Kr_Adsorbed]
      order = CONSTANT
      family = MONOMIAL
      initial_condition = 0.0
    [../]

    [./Xe_Adsorbed]
      order = CONSTANT
      family = MONOMIAL
      initial_condition = 0.0
    [../]

    [./He_Perturb]
      order = CONSTANT
      family = MONOMIAL
      initial_condition = 0.0
    [../]

    [./Kr_Perturb]
      order = CONSTANT
      family = MONOMIAL
      initial_condition = 0.0
    [../]

    [./Xe_Perturb]
      order = CONSTANT
      family = MONOMIAL
      initial_condition = 0.0
    [../]

    [./Kr_AdsorbedHeat]
      order = CONSTANT
      family = MONOMIAL
```

```
      initial_condition = 0.0
    [../]

    [./Xe_AdsorbedHeat]
      order = CONSTANT
      family = MONOMIAL
      initial_condition = 0.0
    [../]

    [./He_AdsorbedHeat]
      order = CONSTANT
      family = MONOMIAL
      initial_condition = 0.0
    [../]

[] #END AuxVariables

[ICs]

    [./Kr_IC]
      type = ConcentrationIC
      variable = Kr
      initial_mole_frac = 0.0
      initial_press = 101.35
      initial_temp = 253.15
    [../]

    [./Xe_IC]
      type = ConcentrationIC
      variable = Xe
      initial_mole_frac = 0.0
      initial_press = 101.35
      initial_temp = 253.15
    [../]

    [./He_IC]
      type = ConcentrationIC
      variable = He
      initial_mole_frac = 1.0
      initial_press = 101.35
      initial_temp = 253.15
    [../]

[] #END ICs

[Kernels]

    [./accumKr]
      type = BedMassAccumulation
      variable = Kr
      index = 0
    [../]

    [./Kr_MT]
      type = AdsorptionMassTransfer
```

```
  variable = Kr
  solid_conc = Kr_Adsorbed
[../]

[./diffKr]
  type = GColumnMassDispersion
  variable = Kr
  index = 0
[../]

[./advKr]
  type = GColumnMassAdvection
  variable = Kr
[../]

[./accumXe]
  type = BedMassAccumulation
  variable = Xe
  index = 1
[../]

[./Xe_MT]
  type = AdsorptionMassTransfer
  variable = Xe
  solid_conc = Xe_Adsorbed
[../]

[./diffXe]
  type = GColumnMassDispersion
  variable = Xe
  index = 1
[../]

[./advXe]
  type = GColumnMassAdvection
  variable = Xe
[../]

[./accumHe]
  type = BedMassAccumulation
  variable = He
  index = 2
[../]
[./diffHe]
  type = GColumnMassDispersion
  variable = He
  index = 2
[../]

[./advHe]
  type = GColumnMassAdvection
  variable = He
[../]

[./wallAccum]
```

```
      type = WallHeatAccumulation
      variable = wall_temp
    [../]

    [./wall_bed_trans]
      type = BedWallHeatTransfer
      variable = wall_temp
      coupled = column_temp
    [../]

    [./wall_amb_trans]
      type = WallAmbientHeatTransfer
      variable = wall_temp
      coupled = ambient_temp
    [../]

    [./columnAccum]
      type = BedHeatAccumulation
      variable = column_temp
    [../]

    [./columnConduction]
      type = GColumnHeatDispersion
      variable =column_temp
    [../]

    [./columnAdvection]
      type = GColumnHeatAdvection
      variable =column_temp
    [../]

    [./columnAdsHeat]
      type = AdsorptionHeatAccumulation
      variable = column_temp
      solid_heats = 'Kr_AdsorbedHeat Xe_AdsorbedHeat He_AdsorbedHeat'
    [../]

[] #END Kernels

[DGKernels]

    [./dg_disp_Kr]
      type = DGColumnMassDispersion
      variable = Kr
      index = 0
    [../]

    [./dg_adv_Kr]
      type = DGColumnMassAdvection
      variable = Kr
    [../]

    [./dg_disp_Xe]
      type = DGColumnMassDispersion
      variable = Xe
```

```
    index = 1
  [../]

  [./dg_adv_Xe]
    type = DGColumnMassAdvection
    variable = Xe
  [../]

  [./dg_disp_He]
    type = DGColumnMassDispersion
    variable = He
    index = 2
  [../]

  [./dg_adv_He]
    type = DGColumnMassAdvection
    variable = He
  [../]

  [./dg_disp_heat]
    type = DGColumnHeatDispersion
    variable = column_temp
  [../]

  [./dg_adv_heat]
    type = DGColumnHeatAdvection
    variable = column_temp
  [../]

[] #END DGKernels

[AuxKernels]

  [./column_pressure]
    type = TotalColumnPressure
    variable = total_pressure
    temperature = column_temp
    coupled_gases = 'Kr Xe He'
  [../]

  [./krypton_adsorption]
    type = MAGPIE_MaterialLDF_Adsorption
    variable = Kr_Adsorbed
    index = 0
  [../]

  [./xenon_adsorption]
    type = MAGPIE_MaterialLDF_Adsorption
    variable = Xe_Adsorbed
    index = 1
  [../]

  [./helium_adsorption]
    type = MAGPIE_MaterialLDF_Adsorption
    variable = He_Adsorbed
```

```
    index = 2
  [../]

  [./krypton_perturbation]
    type = MAGPIE_MaterialLDF_Perturbation
    variable = Kr_Perturb
    index = 0
  [../]

  [./xenon_perturbation]
    type = MAGPIE_MaterialLDF_Perturbation
    variable = Xe_Perturb
    index = 1
  [../]

  [./helium_perturbation]
    type = MAGPIE_MaterialLDF_Perturbation
    variable = He_Perturb
    index = 2
  [../]

  [./krypton_adsorption_heat]
    type = MAGPIE_AdsorptionHeat
    variable = Kr_AdsorbedHeat
    solid_conc = Kr_Adsorbed
    index = 0
  [../]

  [./xenon_adsorption_heat]
    type = MAGPIE_AdsorptionHeat
    variable = Xe_AdsorbedHeat
    solid_conc = Xe_Adsorbed
    index = 1
  [../]

  [./helium_adsorption_heat]
    type = MAGPIE_AdsorptionHeat
    variable = He_AdsorbedHeat
    solid_conc = He_Adsorbed
    index = 2
  [../]

[] #END AuxKernels

[BCs]

  [./Kr_Flux]
    type = DGMassFluxLimitedBC
    variable = Kr
    boundary = 'top bottom'
    input_temperature = 253.15
    input_pressure = 101.35
    input_molefraction = 0.000131792
    index = 0
  [../]
```

```
    [./Xe_Flux]
      type = DGMassFluxLimitedBC
      variable = Xe
      boundary = 'top bottom'
      input_temperature = 253.15
      input_pressure = 101.35
      input_molefraction = 0.000863107
      index = 1
    [../]

    [./He_Flux]
      type = DGMassFluxLimitedBC
      variable = He
      boundary = 'top bottom'
      input_temperature = 253.15
      input_pressure = 101.35
      input_molefraction = 0.999005101
      index = 2
    [../]

    [./Heat_Gas_Flux]
      type = DGHeatFluxLimitedBC
      variable = column_temp
      boundary = 'top bottom'
      input_temperature = 253.15
    [../]

    [./Heat_Wall_Flux]
      type = DGColumnWallHeatFluxLimitedBC
      variable = column_temp
      boundary = 'right left'
      wall_temp = wall_temp
    [../]

 [] #END BCs

[Materials]

    [./BedMaterials]
      type = BedProperties
      block = 0
      inner_diameter = 1.7272
      outer_diameter = 1.905
      bulk_porosity = 0.798
      axial_conductivity = 6.292E-05
      wall_density = 7.7
      wall_heat_capacity = 0.5
      wall_heat_trans_coef = 9.0
      extern_heat_trans_coef = 90.0
      temperature = column_temp
      coupled_gases = 'Kr Xe He'
    [../]

    [./FlowMaterials]
```

```
  type = FlowProperties
  block = 0
  molecular_weight = '83.8 131.29 4.0026'
  comp_heat_capacity = '0.25 0.16 5.1916'
  comp_ref_viscosity = '0.00023219 0.00021216 0.0001885'
  comp_ref_temp = '273.15 273.15 273.15'
  comp_Sutherland_const = '266.505 232.746 80.0'
  flow_rate = 2994.06
  temperature = column_temp
  total_pressure = total_pressure
  coupled_gases = 'Kr Xe He'
  coupled_adsorption = 'Kr_Adsorbed Xe_Adsorbed He_Adsorbed'
  coupled_perturbation = 'Kr_Perturb Xe_Perturb He_Perturb'
[../]

[./AdsorbentMaterials]
  type = AdsorbentProperties
  block = 0
  binder_fraction = 0.175
  binder_porosity = 0.27
  crystal_radius = 1.5
  pellet_diameter = 0.236
  macropore_radius = 3.5e-6
  pellet_density = 1.69
  pellet_heat_capacity = 1.045
  ref_diffusion = '0 0 0'
  activation_energy = '0 0 0'
  ref_temperature = '0 0 0'
  affinity = '0 0 0'
  temperature = column_temp
  coupled_gases = 'Kr Xe He'
[../]

[./AdsorbateMaterials]
  type = MagpieAdsorbateProperties
  block = 0
  temperature = column_temp
  total_pressure = total_pressure
  coupled_gases = 'Kr Xe He'
  number_sites = '2 3 0'
  maximum_capacity = '1.716 1.479 0' #mol/kg
  molar_volume = '20.785 25.412 0' #cm^3/mol
  enthalpy_site_1 = '-44696.86 -18455.18 0'
  enthalpy_site_2 = '-65465.52 -35511.74 0'
  enthalpy_site_3 = '0 -53315.13 0'
  enthalpy_site_4 = '0 0 0'
  enthalpy_site_5 = '0 0 0'
  enthalpy_site_6 = '0 0 0'
  entropy_site_1 = '-170.45 -23.25 0'
  entropy_site_2 = '-248.55 -62.45 0'
  entropy_site_3 = '0 -100.10 0'
  entropy_site_4 = '0 0 0'
  entropy_site_5 = '0 0 0'
  entropy_site_6 = '0 0 0'
[../]
```

```
 [] #END Materials

[Postprocessors]

    [./Kr_exit]
      type = SideAverageValue
      boundary = 'top'
      variable = Kr
      execute_on = 'initial timestep_end'
    [../]

    [./Xe_exit]
      type = SideAverageValue
      boundary = 'top'
      variable = Xe
      execute_on = 'initial timestep_end'
    [../]

    [./He_exit]
      type = SideAverageValue
      boundary = 'top'
      variable = He
      execute_on = 'initial timestep_end'
    [../]

    [./temp_exit]
      type = SideAverageValue
      boundary = 'top'
      variable = column_temp
      execute_on = 'initial timestep_end'
    [../]

    [./press_exit]
      type = SideAverageValue
      boundary = 'top'
      variable = total_pressure
      execute_on = 'initial timestep_end'
    [../]

    [./wall_temp]
      type = SideAverageValue
      boundary = 'right'
      variable = wall_temp
      execute_on = 'initial timestep_end'
    [../]

    [./Kr_solid]
      type = ElementAverageValue
      variable = Kr_Adsorbed
      execute_on = 'initial timestep_end'
    [../]

    [./Kr_heat]
      type = ElementAverageValue
      variable = Kr_AdsorbedHeat
```

```
    execute_on = 'initial timestep_end'
  [../]

  [./Xe_solid]
    type = ElementAverageValue
    variable = Xe_Adsorbed
    execute_on = 'initial timestep_end'
  [../]

  [./Xe_heat]
    type = ElementAverageValue
    variable = Xe_AdsorbedHeat
    execute_on = 'initial timestep_end'
  [../]

[] #END Postprocessors

[Executioner]

  type = Transient
  scheme = implicit-euler
  nl_rel_tol = 1e-6
  nl_abs_tol = 1e-4
  nl_rel_step_tol = 1e-10
  nl_abs_step_tol = 1e-10
  l_tol = 1e-6
  l_max_its = 100
  nl_max_its = 10
  solve_type = newton
  line_search = none    # Options: default none basic l2 bt cp
  start_time = 0.0
  end_time = 50.0
  dtmax = 0.1
  petsc_options_iname = '-pc_type -pc_hypre_type -ksp_gmres_restart'
  petsc_options_value = 'hypre boomeramg 100'

  [./TimeStepper]
    type = SolutionTimeAdaptiveDT
    dt = 0.035
  [../]

[] #END Executioner

[Preconditioning]

[] #END Preconditioning

[Outputs]

  exodus = true
  csv = true
  print_linear_residuals = true

[] #END Outputs
```

DGOSPREY Output

CSV Files

CSV files are comma-separated value files that can be opened and imported directly into excel or another spreadsheet software. The output stored in these files comes from the postprocessors specified in the DGOSPREY input files (see "Postprocessors"). As such, they contain only the time stamped values of those postprocessors and do not retain any mesh dependent information.

NOTE: location of all output files will be the same directory from where the input file was located.

NOTE: the name of the output file will have the original input file name as a prefix, followed by "_out.csv" as the extension. For example, given an input file named "KrXe253K.i", the CSV output file would be "KrXe253K_out.csv".

CSV Output Example

The following is a sample of what the output looks like when produced from the simulations of the "Kr and Xe on AgZ" example file from the above section:

Table 6: Small sample of what the CSV output file will look like

time	He_exit	Kr_exit	Kr_heat	Kr_solid	Xe_exit	Xe_heat	Xe_solid	press_exit	temp_exit	wall_temp
0	0.0481	0	0	0	0	0	0	101.35	253.15	253.15
0.035	0.0481	1.10E-08	6.940	1.55E-04	9.26E-56	1.229	6.67E-05	101.56	253.86	253.21
0.0735	0.0481	1.27E-06	7.462	1.67E-04	1.95E-53	3.179	1.72E-04	101.87	254.69	253.27
0.11585	0.0481	3.45E-06	7.808	1.74E-04	7.67E-52	5.858	3.17E-04	102.10	255.26	253.31

NOTE: the CSV file does not output the units for the non-linear variables. Units of time will always be in hours, concentrations in mol/L, temperatures in K, pressures in kPa, adsorption in mol/kg, and heat of adsorption in J/kg.

CSV files can be opened using excel, or another spreadsheet software. After importing the data into spreadsheet software, you can create various plots and charts, such as the gas species breakthrough curves. Figure 1 below shows a plot of the concentrations Kr and Xe gases at the exit of the column over time. The blue line, along with the left y-axis, shows the breakthrough curve for Kr gas, while the red line, with the right y-axis, gives the breakthrough curve for Xe gas. For Kr gas, breakthrough appears to occur immediately, which is mimicked in the tabulated data above (Table 6). The Xe gas takes about 20 hours to breakthrough, which generally indicates that the strength of adsorption is higher for Xe than it is for Kr.

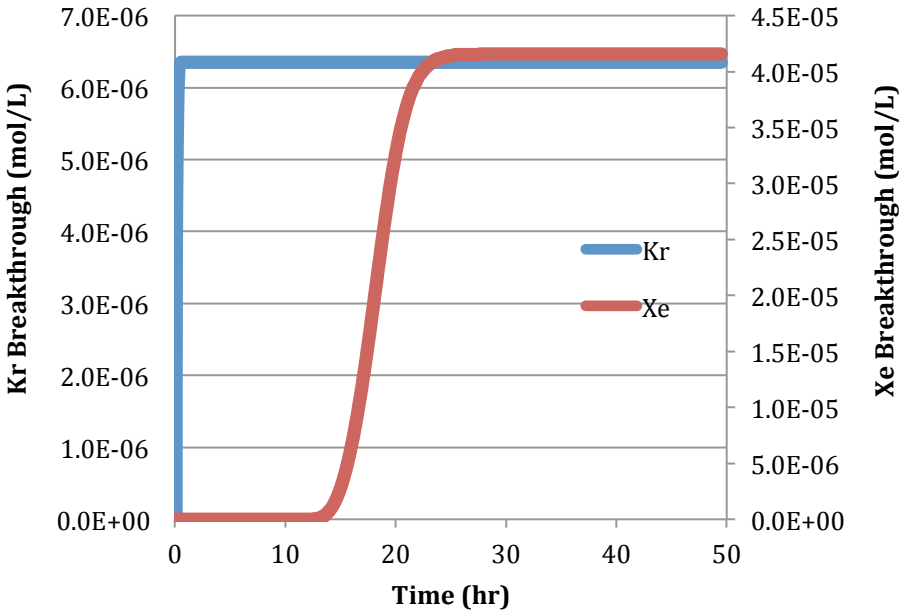

Figure 1: Breakthrough curves of Kr and Xe on AgZ

Exodus Files

Exodus files retain all original mesh information that was produced, including the values of all variables and auxiliary variables in the entire domain at all time steps. These files can only be opened and viewed in special software. One such software is a program called ParaView (www.paraview.org).

ParaView is an open-source software that can be downloaded and installed onto most any operating system, including Mac OS X and linux. You can download ParaView for free at www.paraview.org/download/. Exodus files can be imported into ParaView to view and create videos of the simulation using contour coloring of the non-linear variables on the actual mesh object for a particular simulation.

Exodus Output Example

Figure 2 was produced as a snapshot of the Exodus output from the simulation of the "Water Vapor on MS3A" example. This image was produced using the ParaView software and shows the water vapor concentration in the fixed-bed column roughly mid-way through the simulation. The left side of Figure 2 shows a heat map of water vapor and the right side shows the concentration profile of water vapor on the centerline of the mesh. In the next section, we will cover a quick tutorial of using ParaView to produce images and movies such as this.

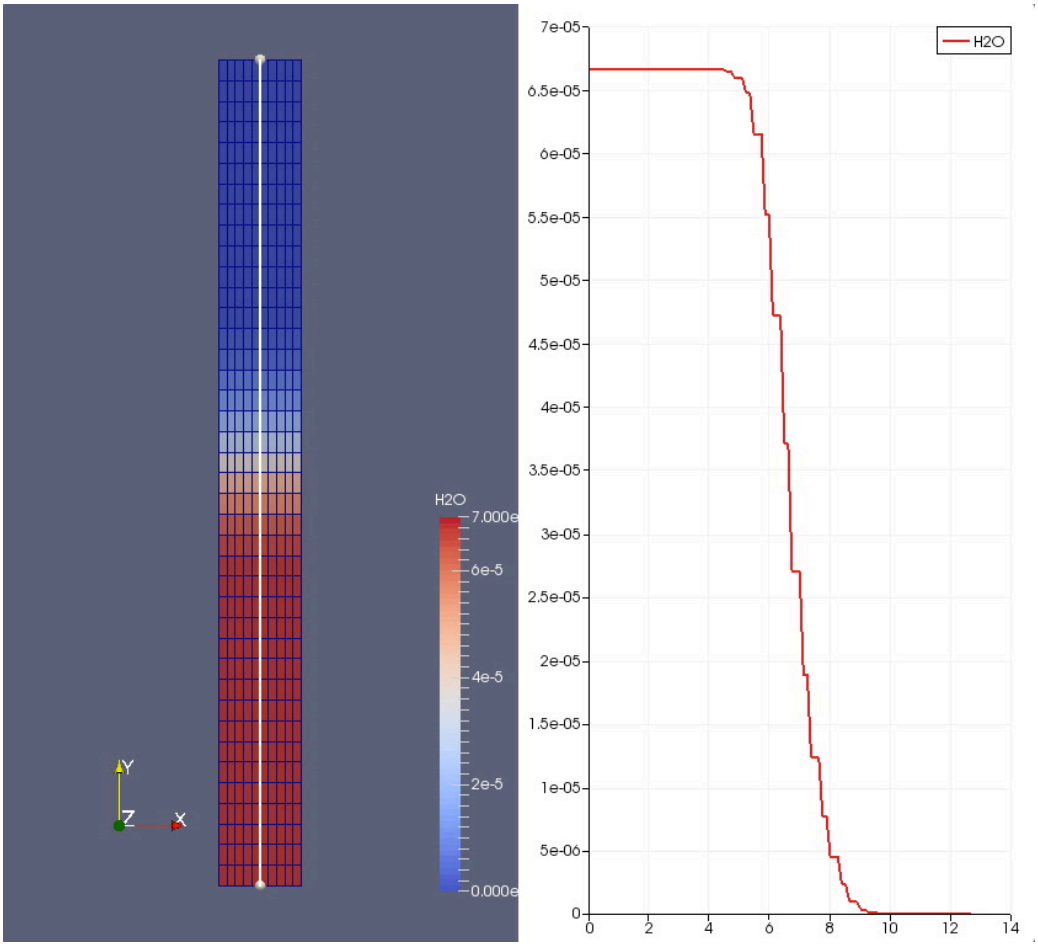

Figure 2: (Left) Heat map of water vapor inside of the fixed-bed column at roughly mid-way through the simulation. (Right) Concentration profile of water vapor, at that same time snapshot, taken from the centerline of the mesh.

ParaView Tutorial

ParaView is the application that was used to visualize the Exodus data produced by DGOPSREY. The program is distributed for free and can be downloaded at the following web address: www.paraview.org/download/. After you have downloaded and installed the software, you can follow the following tutorial to learn how to utilized this application for DGOSPREY data visualization.

Opening Exodus Files

To open the Exodus files produced from DGOSPREY, open the ParaView application then click on the "Open" button on the user interface as shown below (Figure 3):

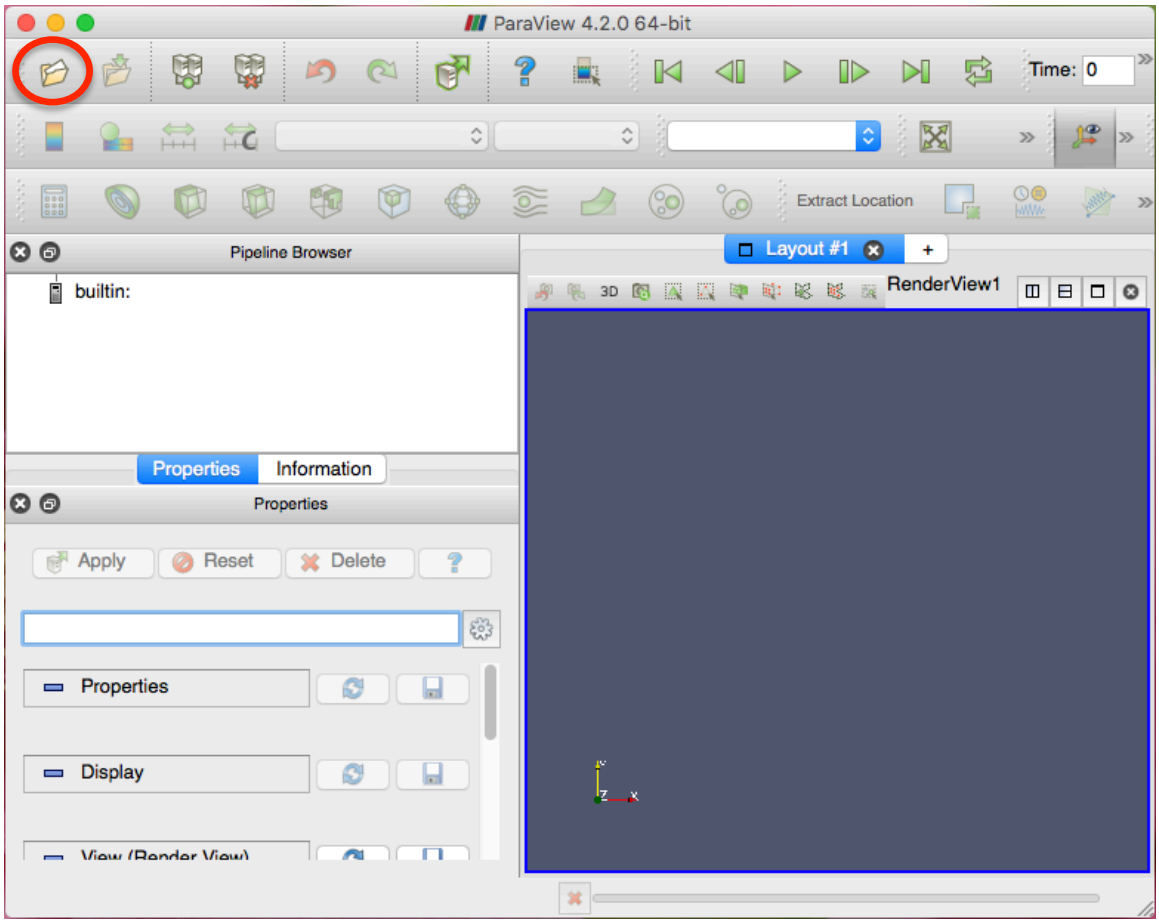

Figure 3: ParaView window showing the "Open" file button. Press this button to open a menu allowing you to search for the Exodus file you want to view.

After clicking "Open" you will be able to search for the Exodus file you want to open in ParaView. In this example, we are going to open the water vapor on MS3A output, which is named "H2O_SCOPSOWL_out.e" for this tutorial. The name of your output file will depend on the input file you run.

Viewing Heat Maps

When the Exodus file first opens in ParaView you will still not be able to see any mesh object in the window (Figure 4). However, the window will display the file object under the "Pipeline Browser" box (red circle in Figure 4). Underneath this box is a box called "Properties" that holds the names of all DGOSPREY variables and auxiliary variables from the input file (green square in Figure 4).

To display the mesh object and prepare ParaView to view the variable values on the mesh, you need to check the boxes next to all variable names that you want to display. Use the scroll bar (blue circle in Figure 4) to scroll through the list of

variable names and check all the variables that you want ParaView to display on the mesh. Then, click the blue "Apply" button under the "Properties" box.

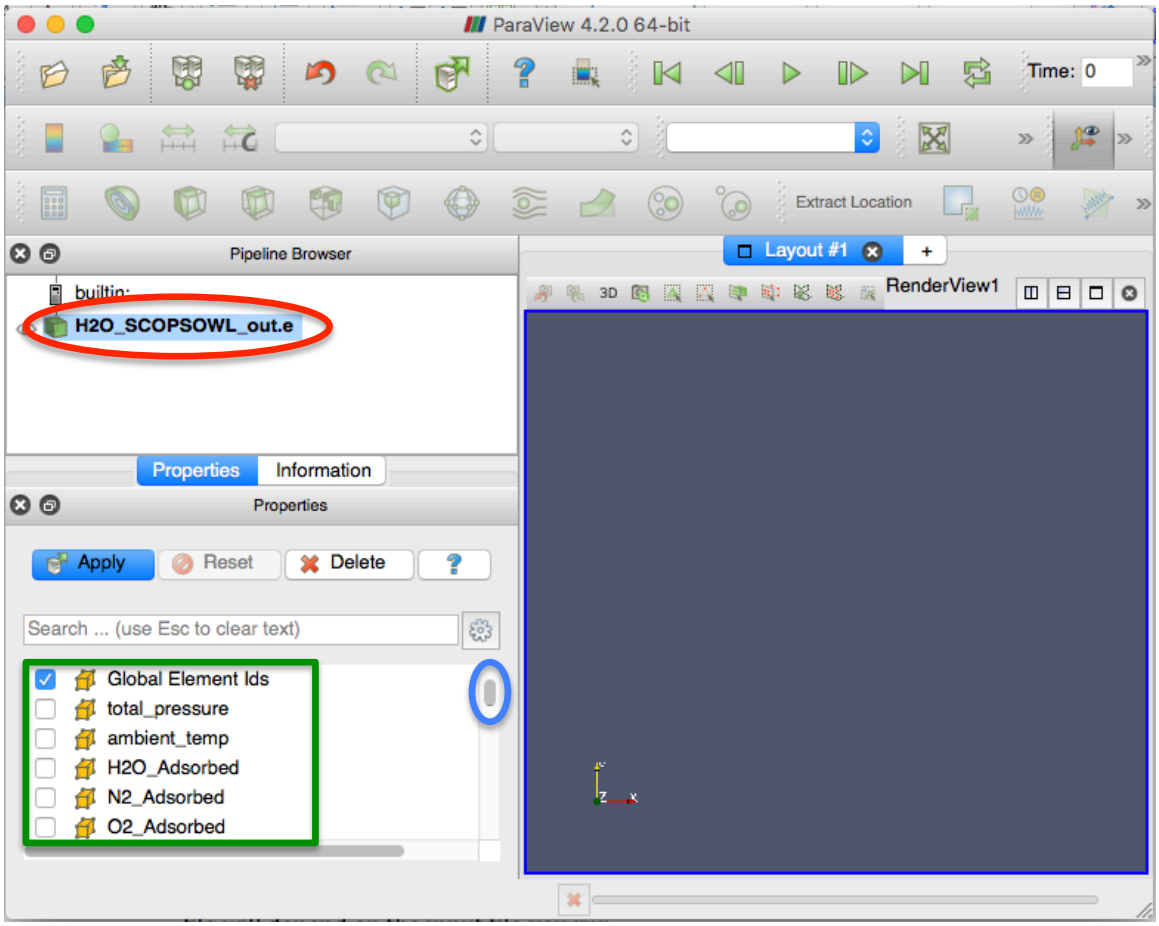

Figure 4: ParaView window after opening the Exodus file. The red circle shows the name of the object that was opened. The blue circle is used to scroll through the different variables that were declared in the DGOSPREY input file. Check all the variables you are interested in viewing then press "Apply".

After hitting "Apply", a gray rectangle will appear on the right side of the window under "Layout". This rectangle is the mesh object, although it does not show a mesh or any variable heat map. To get the mesh to display, click on the dropdown menu at the top right of the window (red circle in Figure 5) and select "Surface with Edges". Once selected, the gray rectangle will be filled with black lines representing the mesh.

To get the mesh to show a heat map for a variable, you will need to choose a variable from the dropdown menu in the top left of the window (blue circle in Figure 5). Select the name of the variable that you want to view the heat map for. In this example, we will select the "H2O" variable for this simulation. Figure 6 shows the new view of the mesh, which has turned all blue to represent the initial condition for water vapor in the fixed-bed.

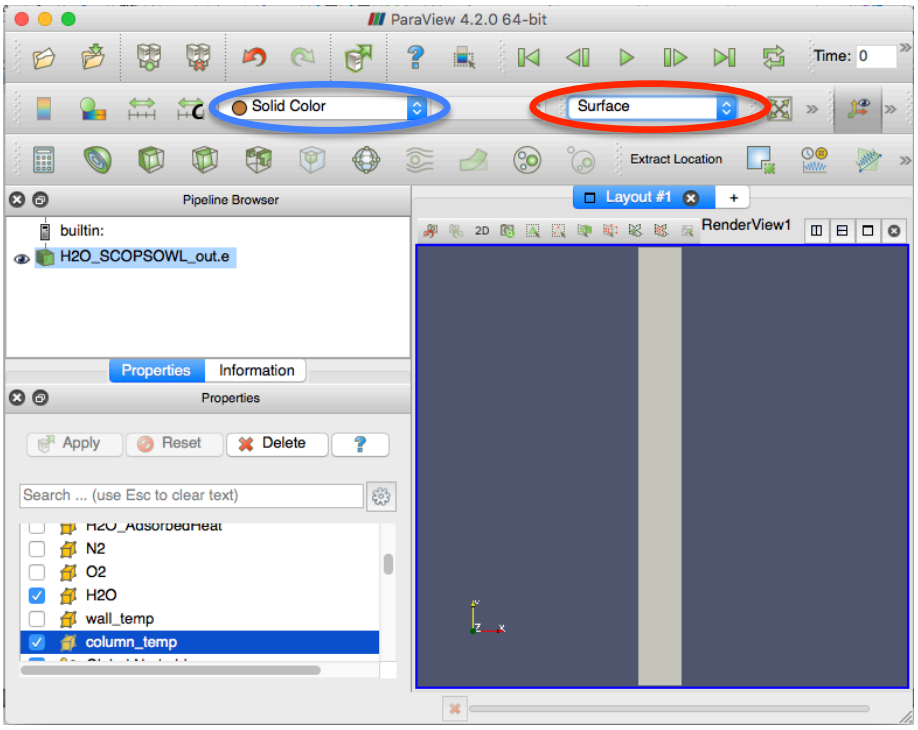

Figure 5: ParaView window showing the domain of the mesh. To show the actual mesh, select "Surface with Edges" from the dropdown menu circled in red. To show a heat map for a variable, select the name of the variable from the dropdown menu circled in blue.

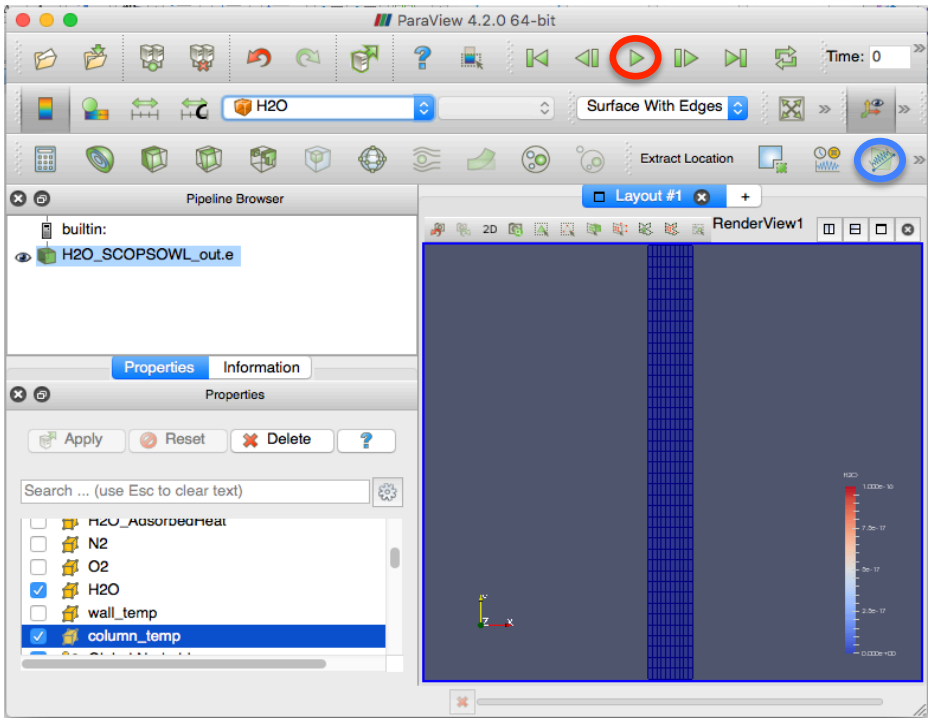

Figure 6: ParaView window showing the heat map of H2O in the mesh. To watch the heat map movie, press the play button circled in red. The blue circle shows the "Plot Over Line" button.

Viewing Profile Plots

To view a profile of a non-linear variable along the length of the bed (y-axis of the mesh) you need to select the "Plot Over Line" button, which is circled in blue on Figure 6 above. After clicking that option, the window will change to what is shown in Figure 7 below. Under the "Properties" box in the bottom left, scroll until you see the "Y Axis" button (red circle in Figure 7). Click it then press "Apply" above. This will open another box to the right of the mesh where the profile plots are shown. These curves represent the values of all non-linear variables along the centerline of the mesh (Figure 8). You can view the profiles more clearly by closing the mesh view by pressing the "Close Box" button circled in red on Figure 8.

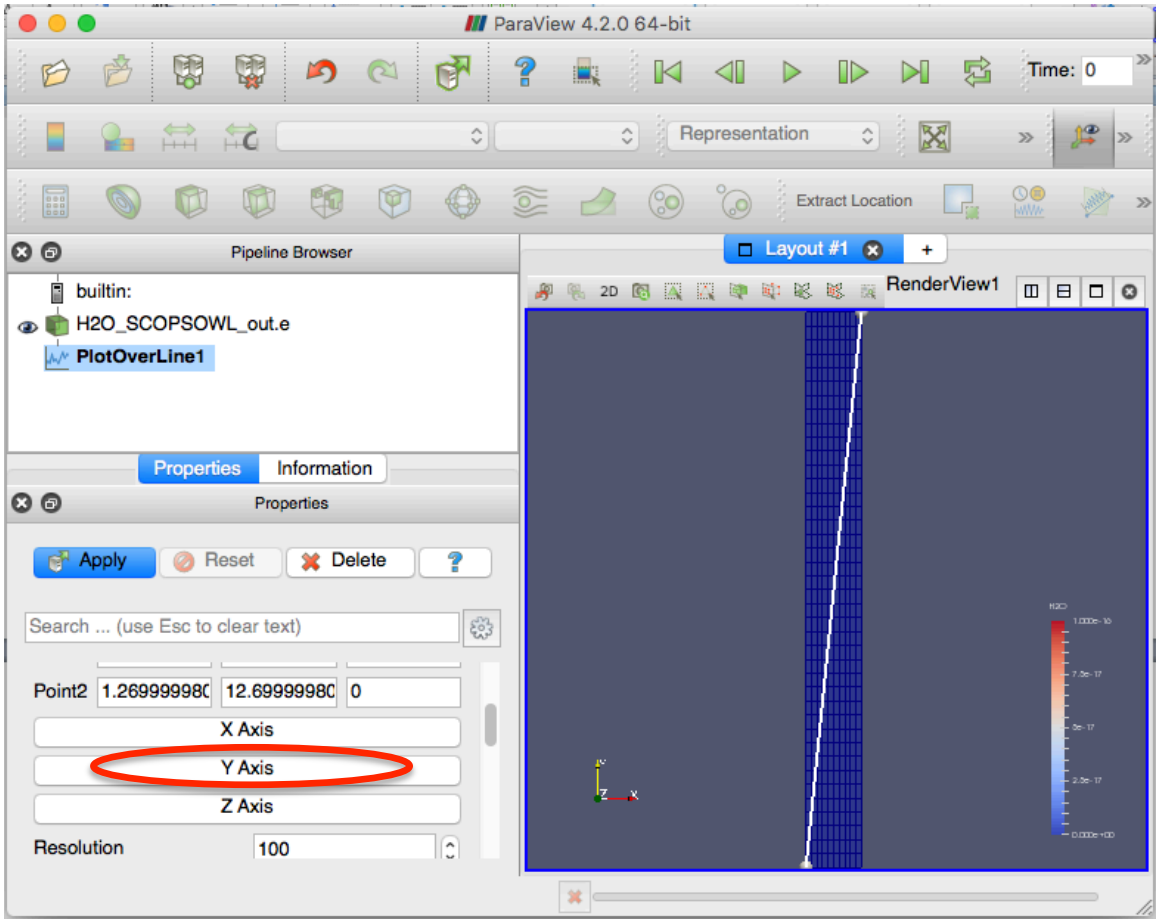

Figure 7: ParaView window showing the "Plot Over Line" options. Select the "Y Axis" button then press "Apply" to display the plots of all variables along the centerline of the mesh. This opens up another box on the bottom right next to the mesh, which shows the profiles of all non-linear variables that were selected when the mesh was displayed.

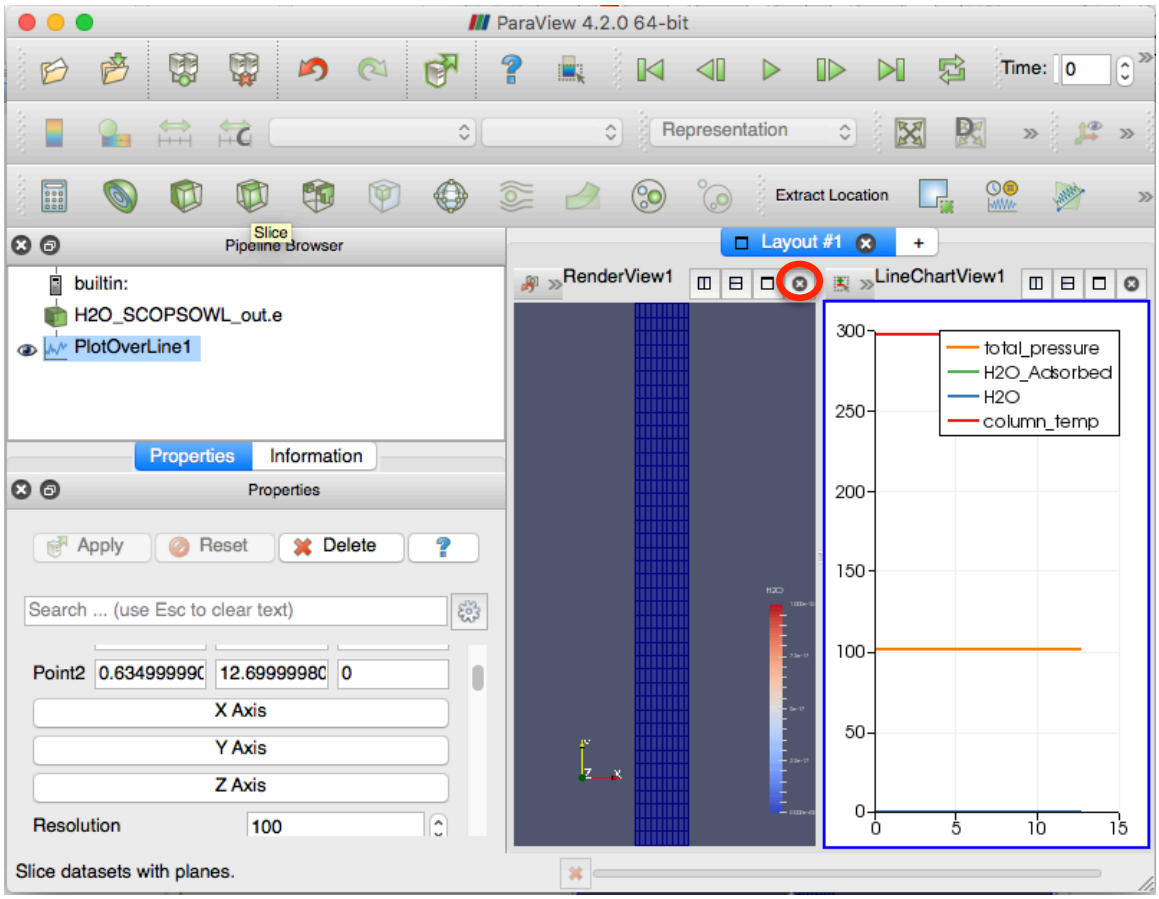

Figure 8: ParaView window showing the mesh view side-by-side with the profile view. You can close the mesh view by clicking the "Close Box" button circled in red.

After closing the mesh view, you now have a better view of the profile plots. However, by default it will show all profiles for all variables and you will want to just view one variable at a time. To select and view just a single profile, go to the "Properties" box on the bottom left and deselect the other variables. In this example, we just want to view the water vapor profile, so we deselect all variable names except for H2O (red circle in Figure 9).

Now that only one variable is selected, we will be able to more clearly view the profile for that non-linear variable. However, ParaView does a poor job of providing reasonable scales for each variable. Notice how in Figure 9 the y-axis goes from a value of -1e-07 to +1e-07. You will need to manually fix the scaling of the y-axis for each non-linear variable's profile you wish to view. To do this, scroll down through the "Properties" box on the bottom left until you find the "Left Axis Range" sub-category. Select the "Left Axis Use Custom" check box then enter in custom values for the lower and upper bounds of the y-axis (red rectangular area of Figure 10). For this example, we will select the lower value to be 0 and the upper value to be 7e-05. You can then press the play button at the top to view the profile change with each simulation time step (Figure 10).

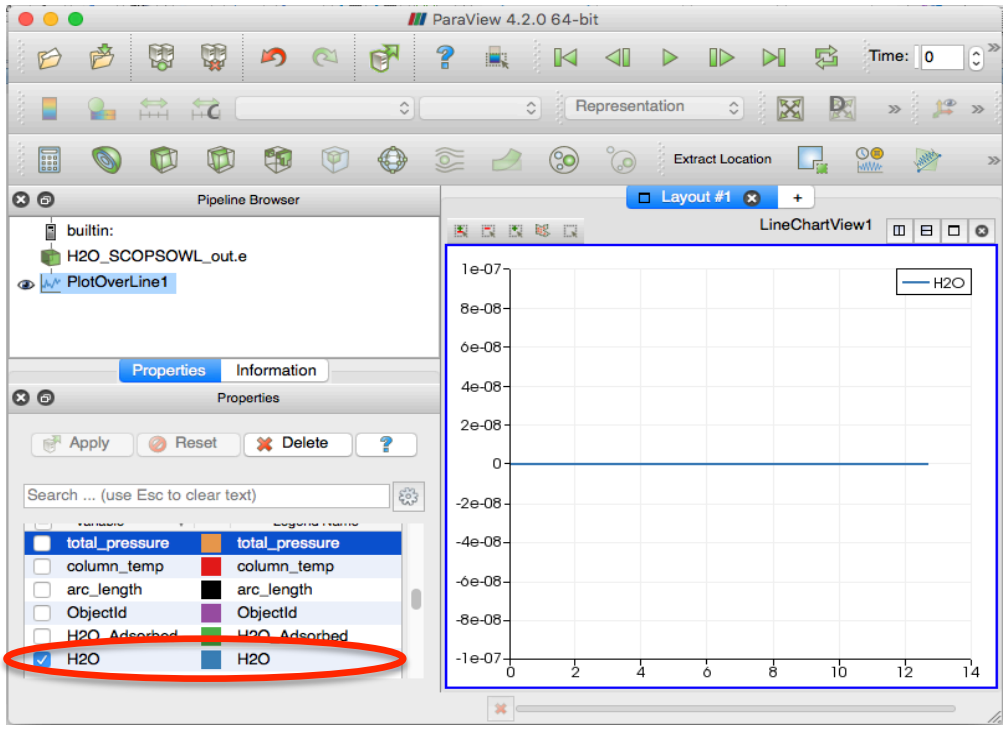

Figure 9: ParaView window showing the larger view of the profile of water vapor on the centerline of the mesh. The red square shows where you would select the non-linear variable to view in the window.

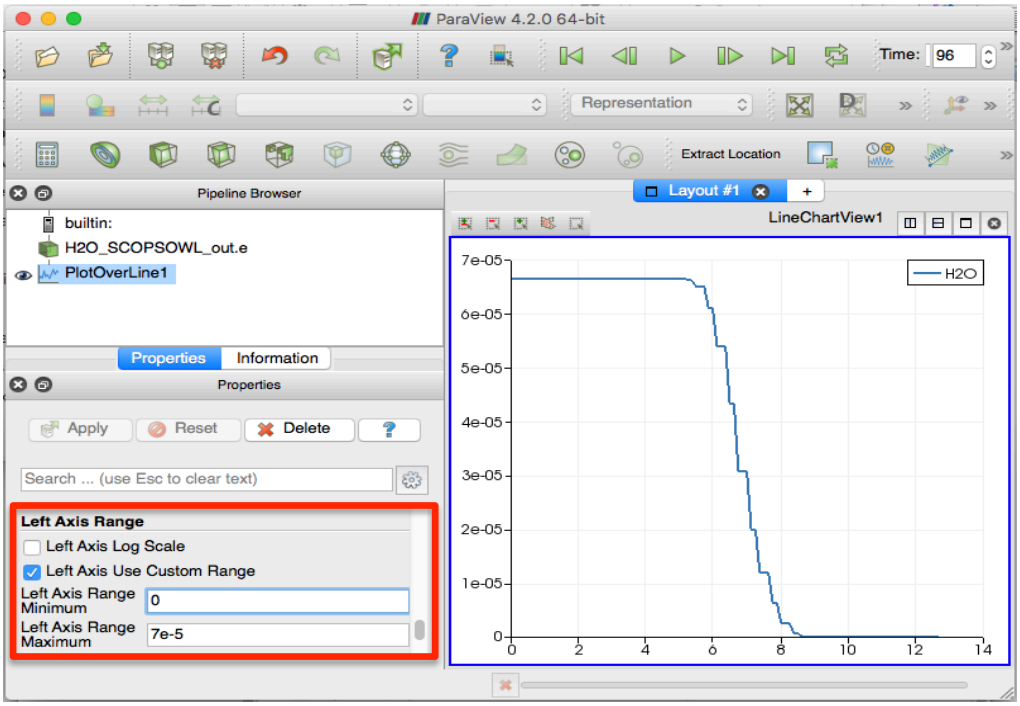

Figure 10: ParaView window showing where and how (red rectangle) to customize the y-axis of the profile plot on the bottom right. From here, you can press the play button to view how the profile changes with the simulation time steps.

Creating Movies

You can create movies in ParaView that are made from whatever view is currently showing in the "Layout" box at the bottom right of the ParaView window. For example, if we continue from the last section ("Viewing Profile Plots") then the movie will be for the water vapor concentration profiles in the column over time. To do this, you must open the dropdown menu under the "File" option for the application (Figure 11). From this menu, you need to select the "Save Animation" option (red circle in Figure 11), which will bring up a second window with some animation options. Typically, the default options are fine, so there is nothing additional to change.

NOTE: the location of the "File" dropdown menu may be different for your operating system. For Mac OS X, the "File" menu is located at the top of your computer screen, next to the apple symbol.

NOTE: if you want to create a movie of the heat map, then follow the instructions for "Viewing Heat Maps" before selecting "Save Animation".

NOTE: movie file formats can be chosen when you are asked where to save the file.

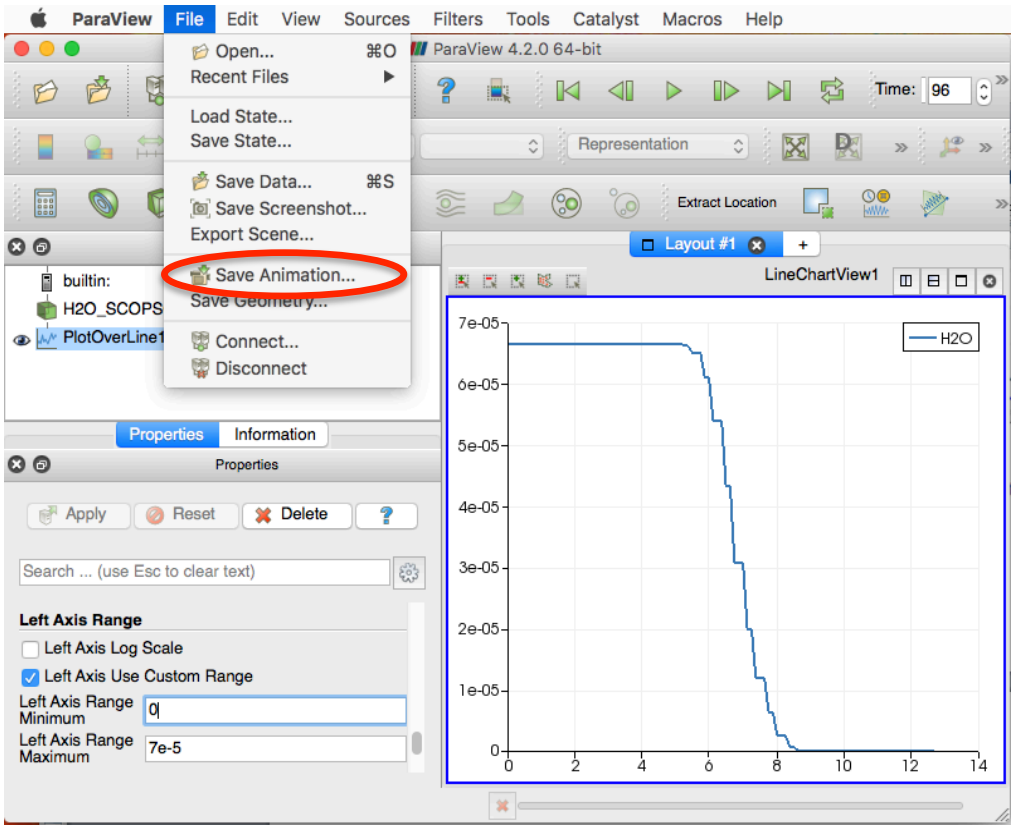

Figure 11: ParaView window showing the menu where you will find the "Save Animation" option for creating movies in ParaView.