

# Advanced Post-Processing Workshop

Barracuda Virtual Reactor Users' Conference

October 1, 2015

Santa Ana Pueblo, New Mexico

# Outline

- Introduction to Example Simulation ..... 3
- Goal 1: Plot profile of temperature vs height ..... 4
  - Achieving Goal 1: Using 2D Plot Data
- Goal 2: Plot residence time distribution (RTD) of particles exiting system ..... 20
  - Achieving Goal 2: Using Raw Particle Data at Flux Planes
- Goal 3: Show only a certain species of particles in GMV ..... 38
  - Achieving Goal 3: Using Particle Select in GMV
- Goal 4: Make a spinning movie ..... 42
  - Achieving Goal 4: Using BATCHMOVIE.sh

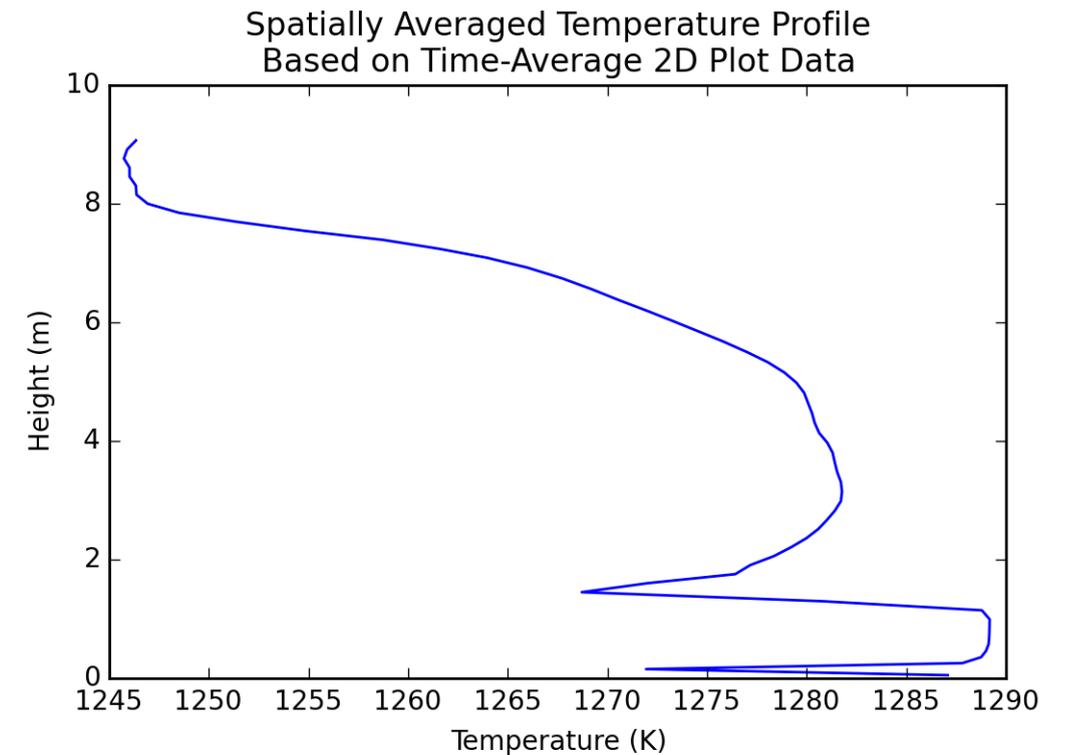
# Introduction to Example Simulation: Wednesday Gasifier

- From Barracuda VR Training Course
  - Coal gasification example
  - Chemistry: 9 volume-average chemical reactions from open literature
  - Thermal: flow streams at various temperatures, and heating coils within fluidized bed
  - Multiple multi-material particle species
    - Initial char bed
    - Fresh coal feed



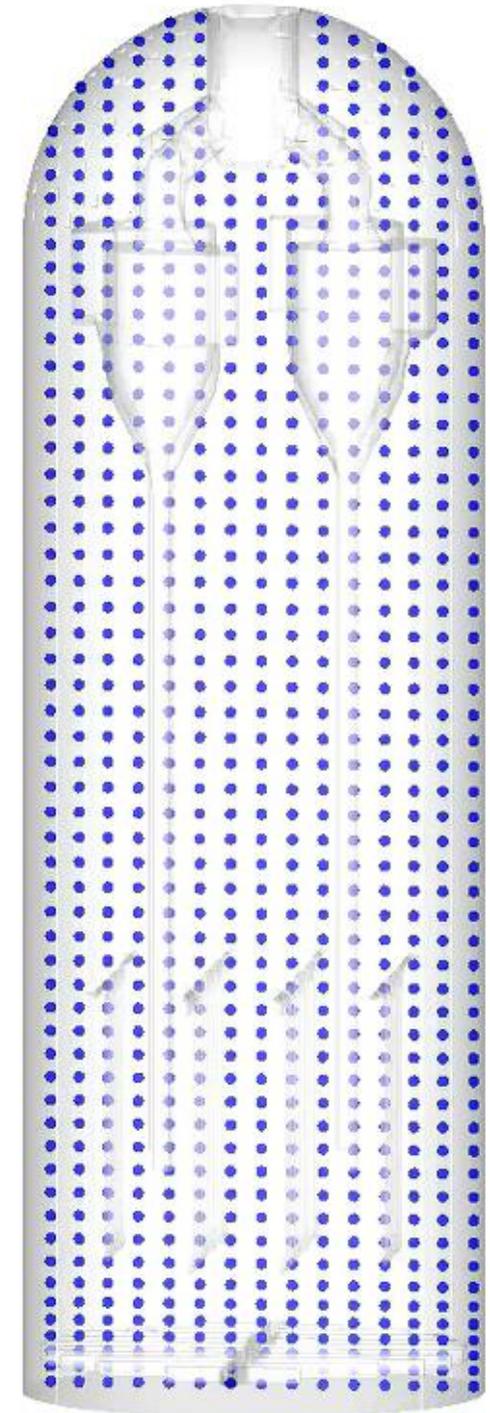
# Goal 1: Plot Profile of Temperature vs Height

- Achieving Goal 1: Using 2D Plot Data
  - Introduction to 2D Plot Data
  - Creating a basic 2D data plot using Plot Manager
  - Calculating a spatially-averaged temperature profile using Python



# What is 2D Plot Data?

- 2D Plot Data gives you information about a selected variable over an entire plane within the simulation domain.
- 2D Plot Data is available for many variables, including instantaneous and time-average:
  - Particle volume fraction
  - Pressure
  - Fluid velocity
  - Fluid temperature
  - Many more...
- 2D Plot Data is output at a user-specified time interval during a simulation.



# How to Enable Output of 2D Plot Data

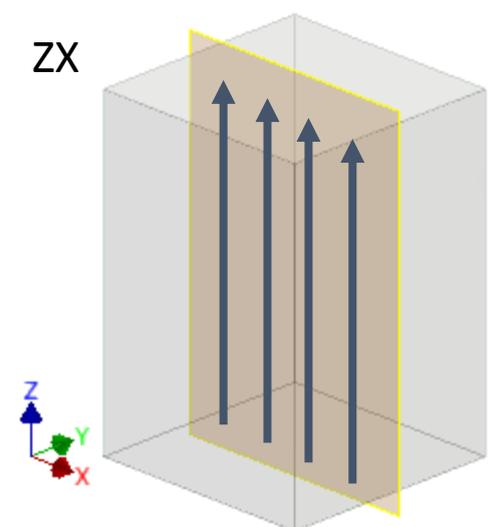
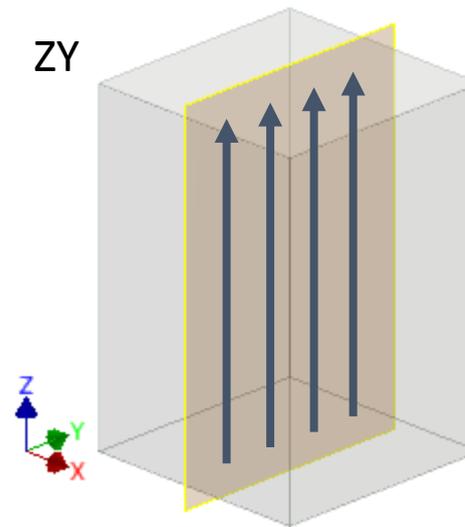
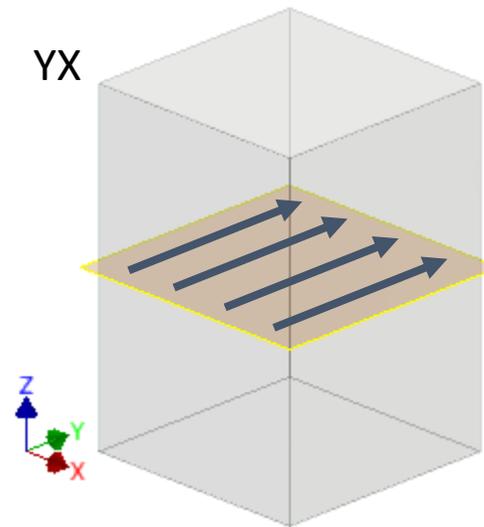
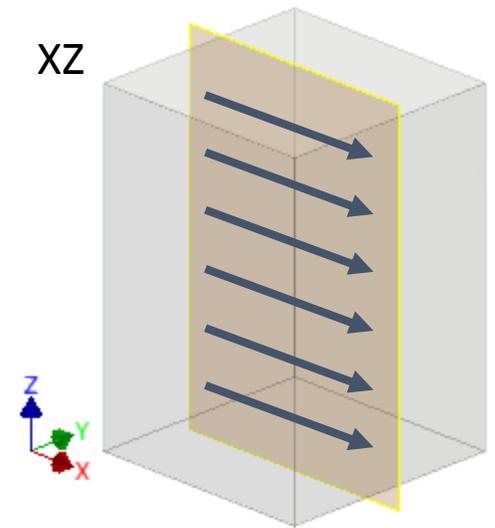
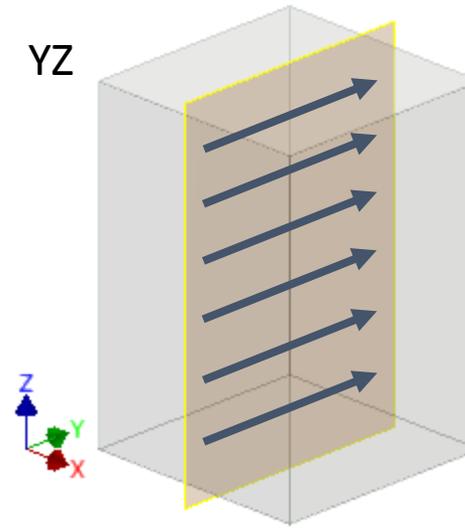
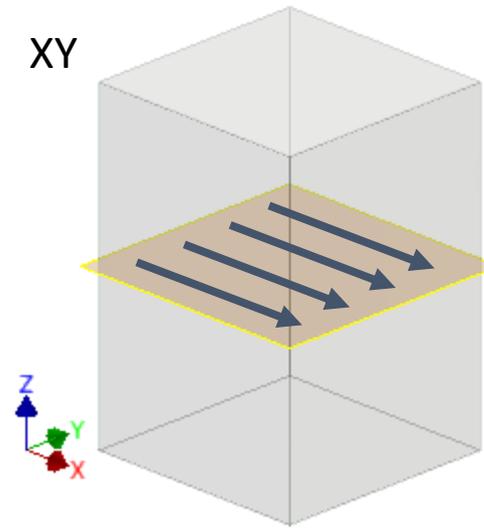
**Choose the orientation(s) and location(s) for 2D Plot Data.**

**Choose the variable(s) for which 2D Plot Data should be created.**

**Specify the time interval at which 2D Plot Data should be generated.**

# Choosing the 2D Plot Data Orientation

- 2D Plot Data can be output in any combination of the 6 orientations shown at right.
- Choosing the correct orientation can make data processing much easier.
- The arrows indicate the arrangement of data columns in output files.



# Output Format of 2D Plot Data

- Each 2D Plot Data file contains text data in space-separated columns

This example is a “zx” 2D Plot Data file. It covers a y-plane.

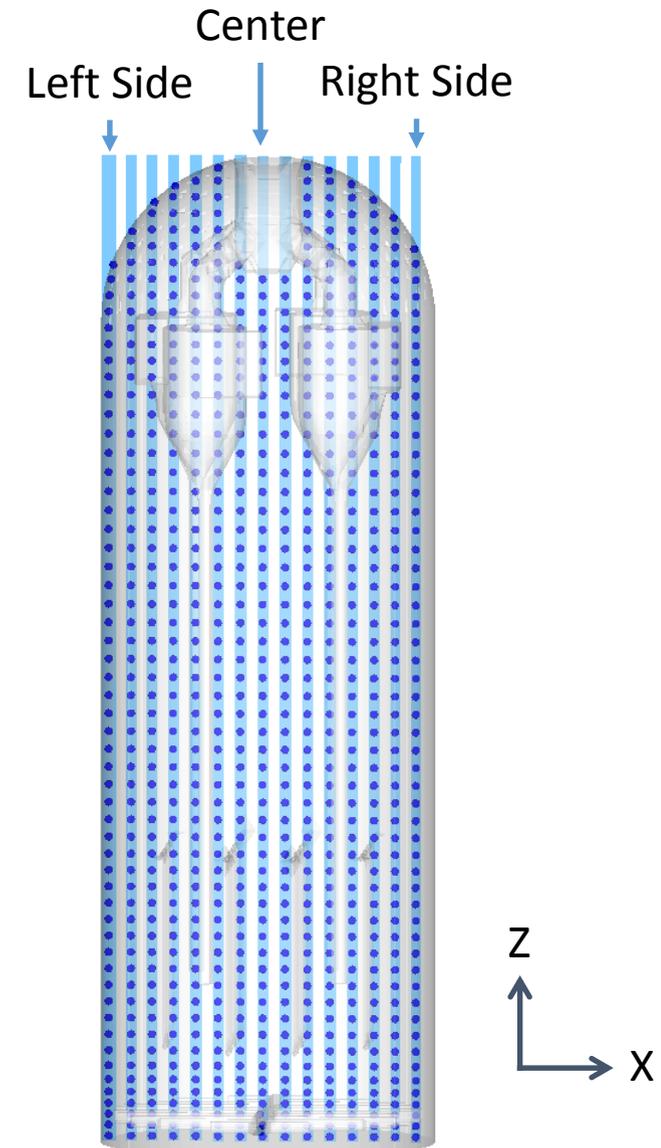
The data is fluid temperature.

```
# Time = 2.00015e+01 (s)
# y = -1.52578e-03 (m) (j=17)
# First column z-distance (m). Following columns are temperature (K) at x-locations
# 1 z
# 2 x=-1.45617e+00m (i= 1)
# 3 x=-1.25138e+00m (i= 3)
# 4 x=-1.06082e+00m (i= 5)
# 5 x=-8.66964e-01m (i= 7)
# ... more column headers ...
5.109293e-02 1.300826e+03 1.303385e+03 1.303440e+03 ... more columns of data ...
1.532788e-01 1.299528e+03 1.301458e+03 1.300700e+03 ... more columns of data ...
2.543134e-01 1.299502e+03 1.301873e+03 1.301868e+03 ... more columns of data ...
3.544176e-01 1.298454e+03 1.301121e+03 1.301696e+03 ... more columns of data ...
4.597586e-01 1.297426e+03 1.294514e+03 1.301106e+03 ... more columns of data ...
5.756338e-01 1.296742e+03 1.294430e+03 1.299496e+03 ... more columns of data ...
7.030965e-01 1.296732e+03 1.296648e+03 1.298808e+03 ... more columns of data ...
8.433054e-01 1.296276e+03 1.300004e+03 1.297590e+03 ... more columns of data ...
... more rows of data ...
```

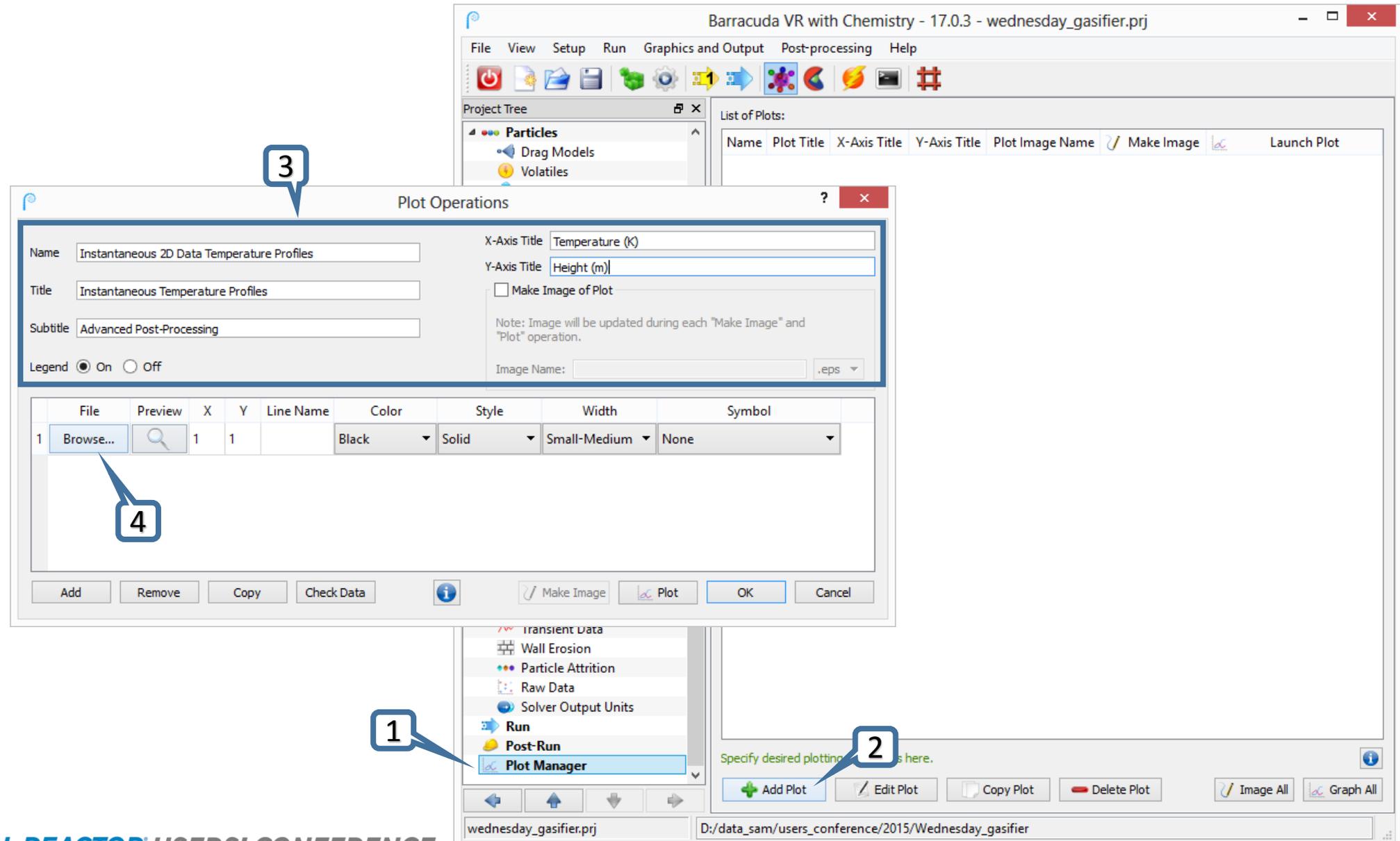
“zx” means the first data column has z-values, and subsequent columns hold data at evenly-spaced x-positions.

# Plot Temperature Profiles at Left Side, Center, and Right Side

- Using 2D Plot Data output files, we will plot instantaneous and time-average temperature profile curves.
- We will plot data at  **$t = 100$  s**.
- To visualize the temperature variation across the reactor, we will plot profiles at the **Left Side**, **Center**, and **Right Side** of the vessel.

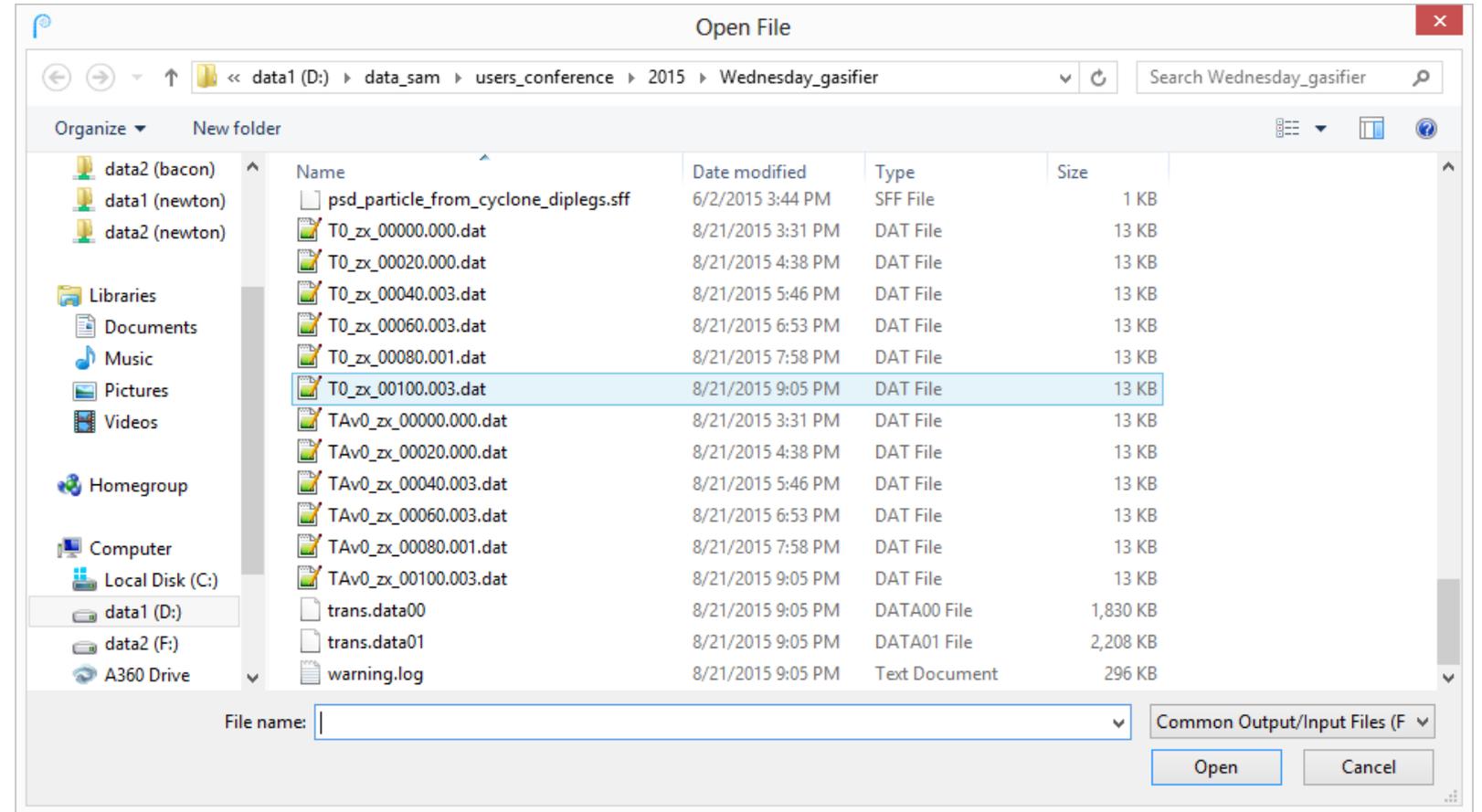


# Use Plot Manager in the Barracuda VR GUI



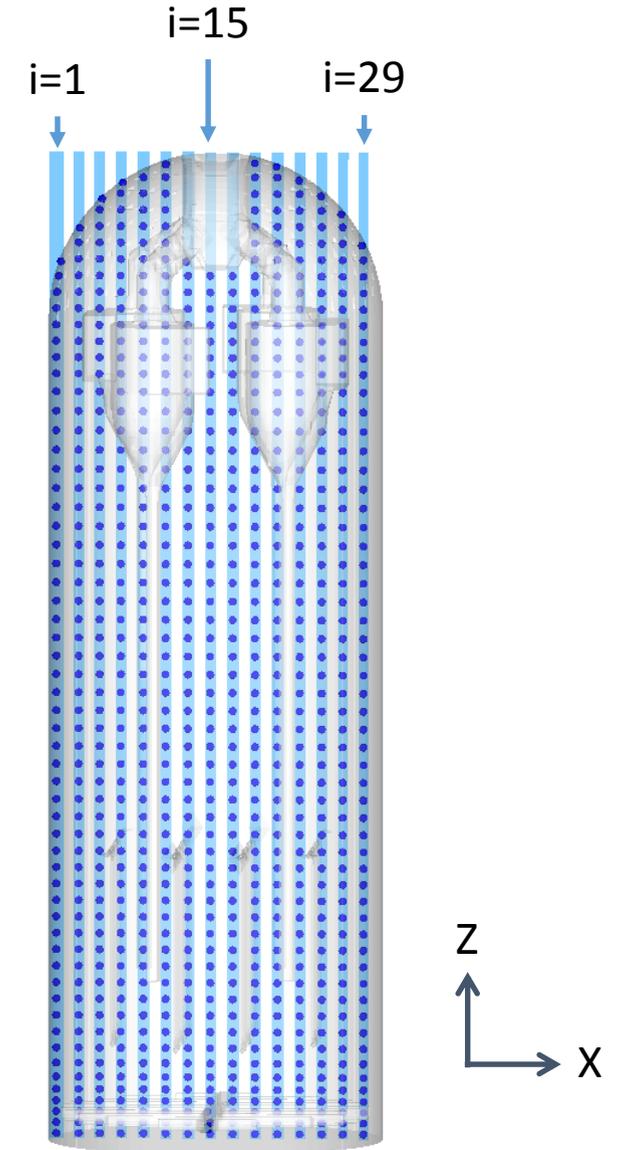
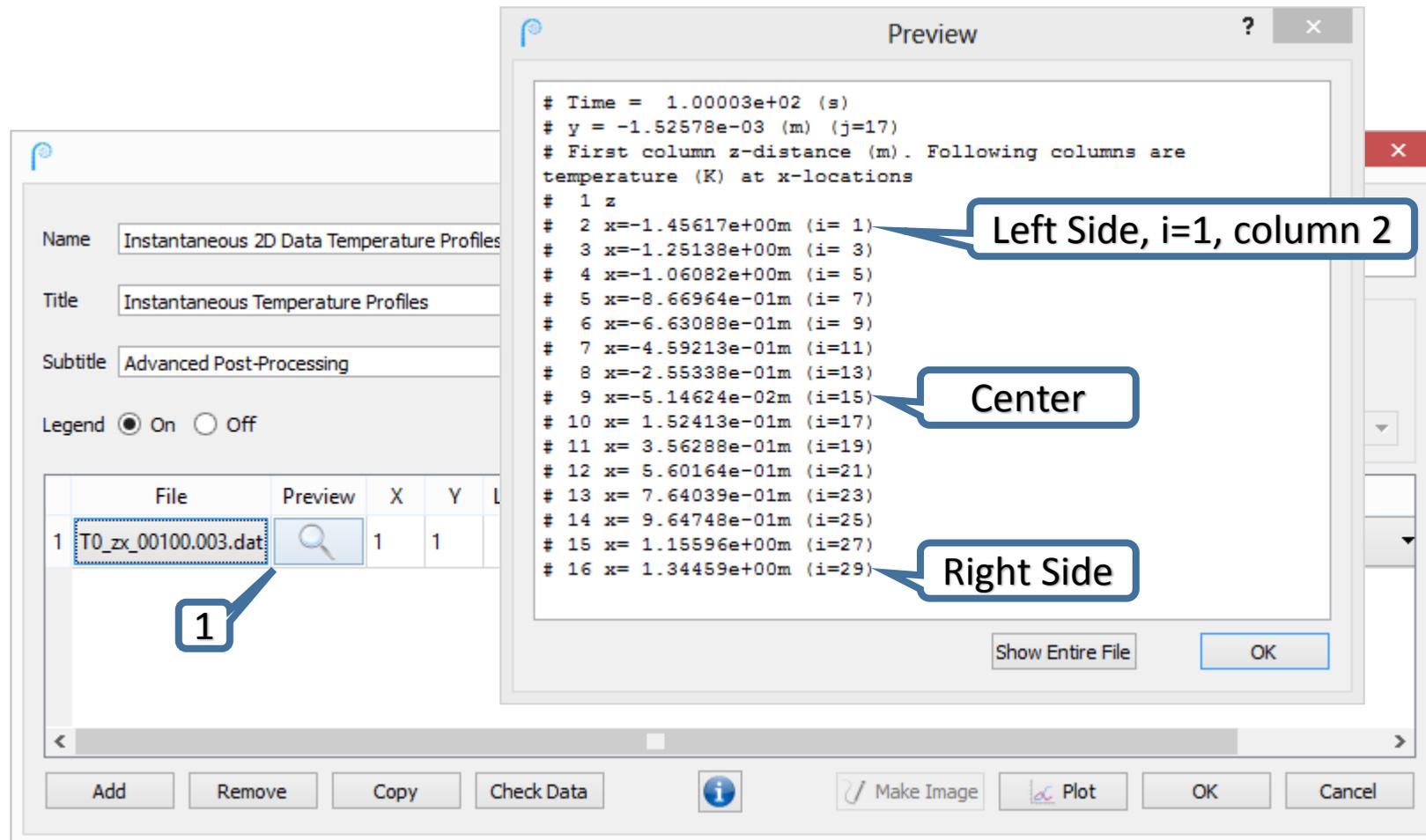
# Select the Instantaneous Fluid Temperature File at t=100 s

- Note the file naming convention:
  - **T0** indicates instantaneous fluid temperature data.
  - The simulation time,  $t = 100.003$  s, is embedded in the file name.
- A table listing all 2D Plot Data output file names is available in the [Barracuda User's Manual](#)

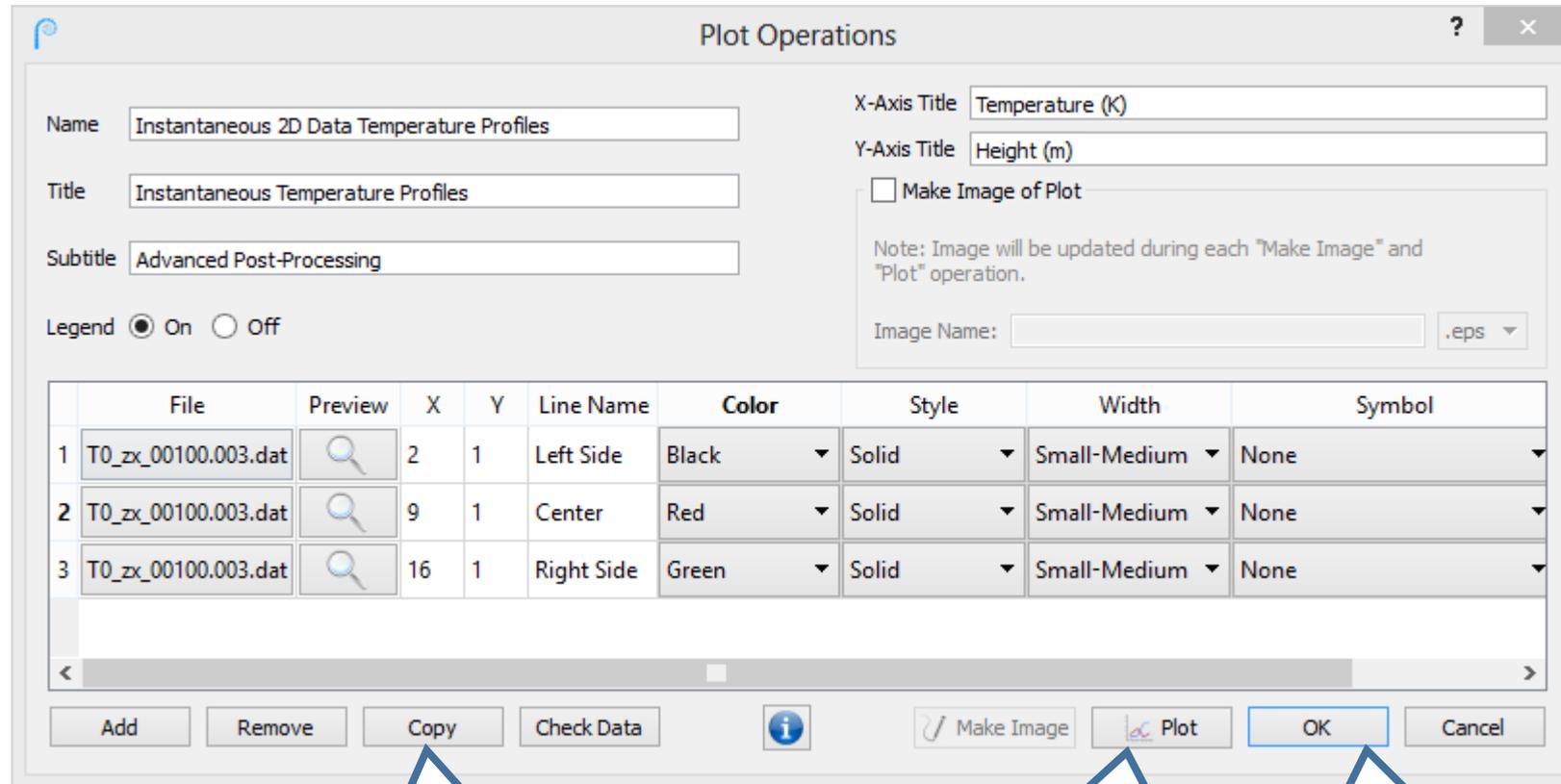


# Use Preview to Choose Data Columns for Plotting

- The **Preview** button quickly displays the header section of a data file.



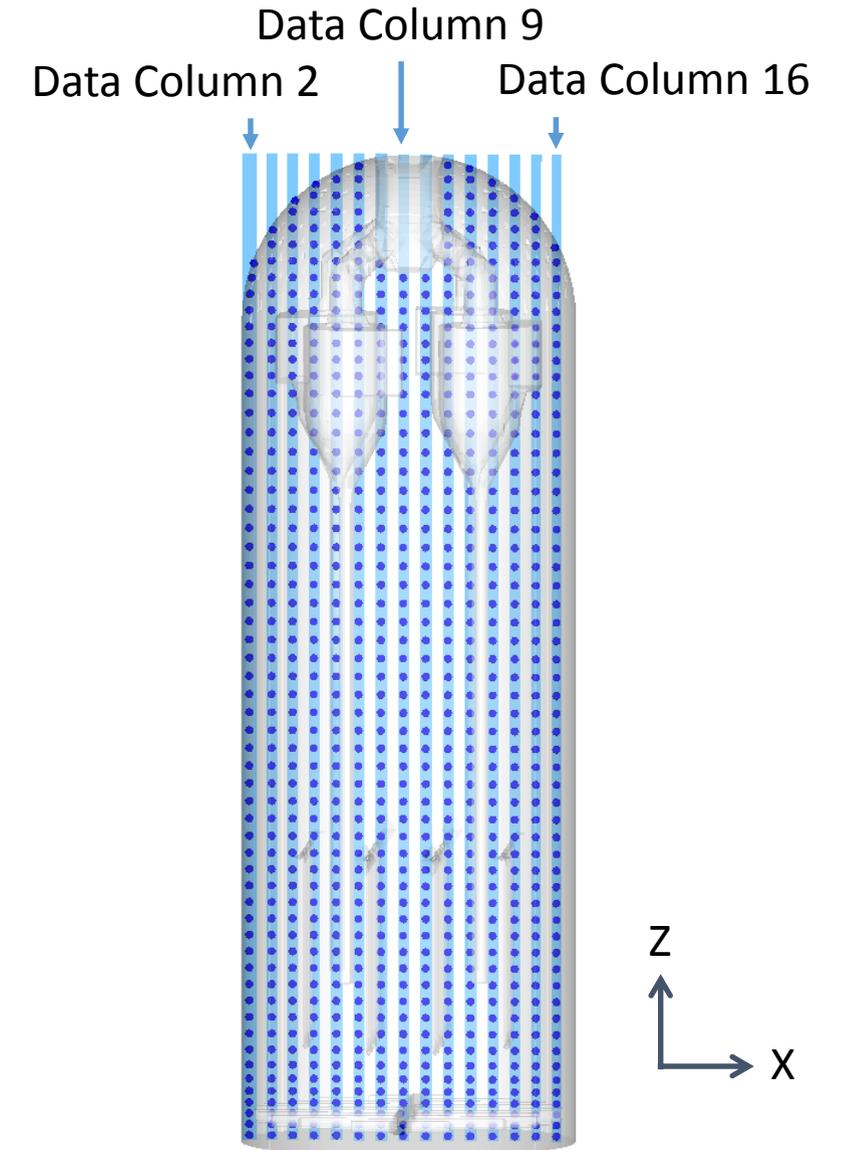
# Define the X and Y Columns for Plotting



Use the **Copy** button to create rows 2 and 3

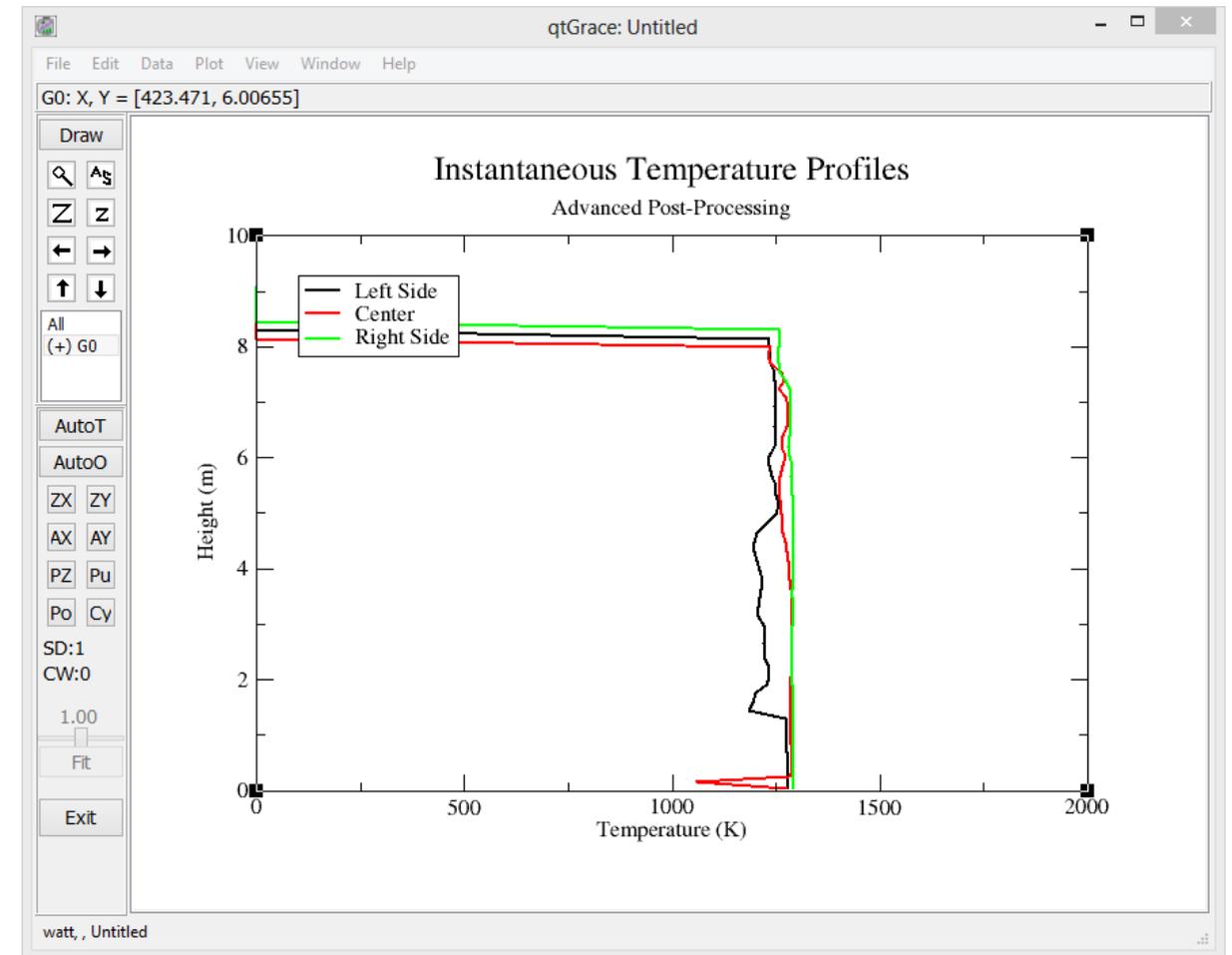
Click **Plot** to launch the plotting utility (Xmgr / qtGrace)

Click **OK** to exit the **Plot Operations** dialog



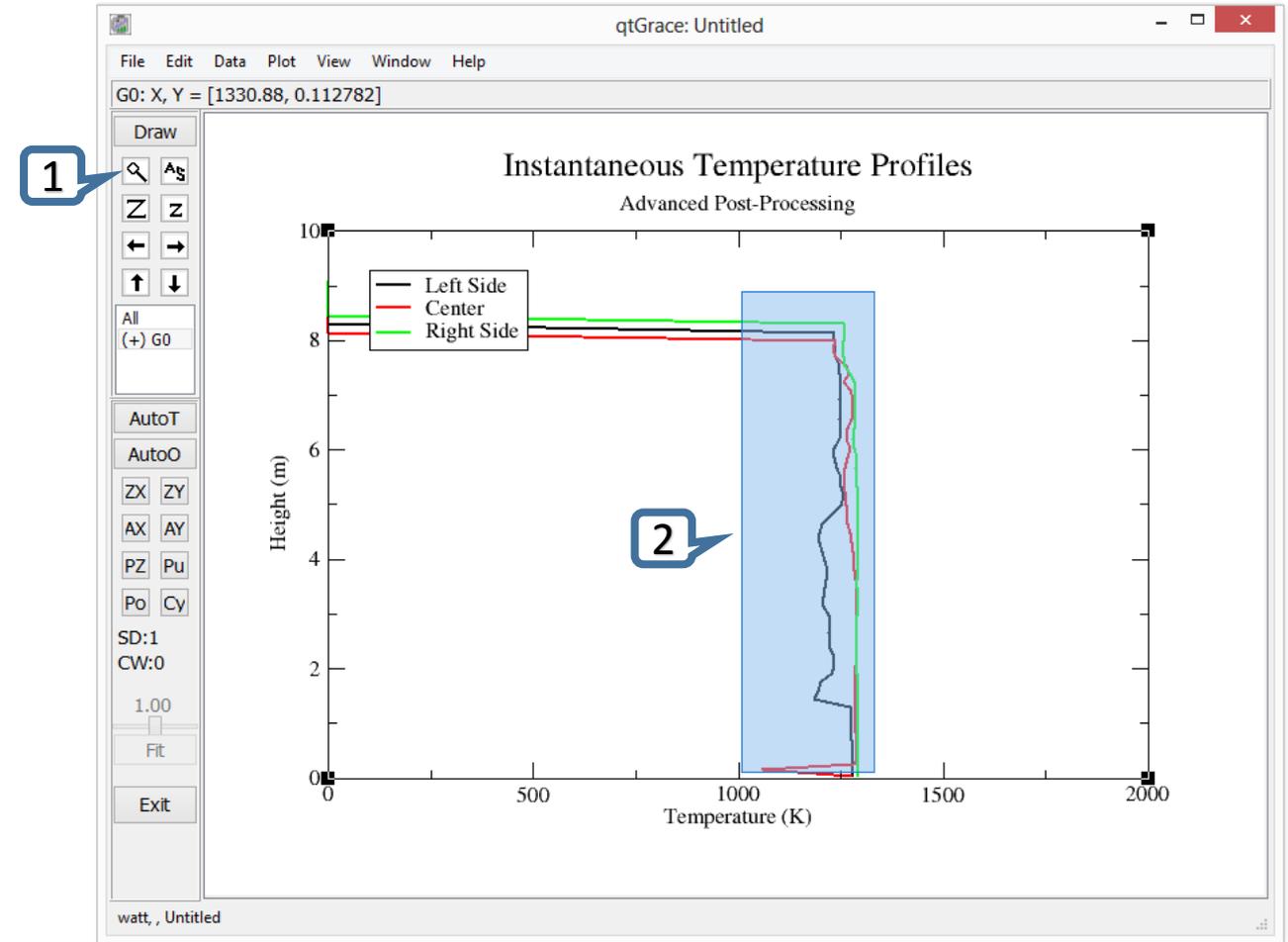
# Interpreting the 2D Plot Data Instantaneous Temperature Profile

- Three curves are plotted, as defined in the **Plot Operations** dialog.
- Values of **0** or **-1** are reported for null cells.
- A low-temperature spike is seen in the **Center** profile, corresponding to the steam sparger Injection BC.



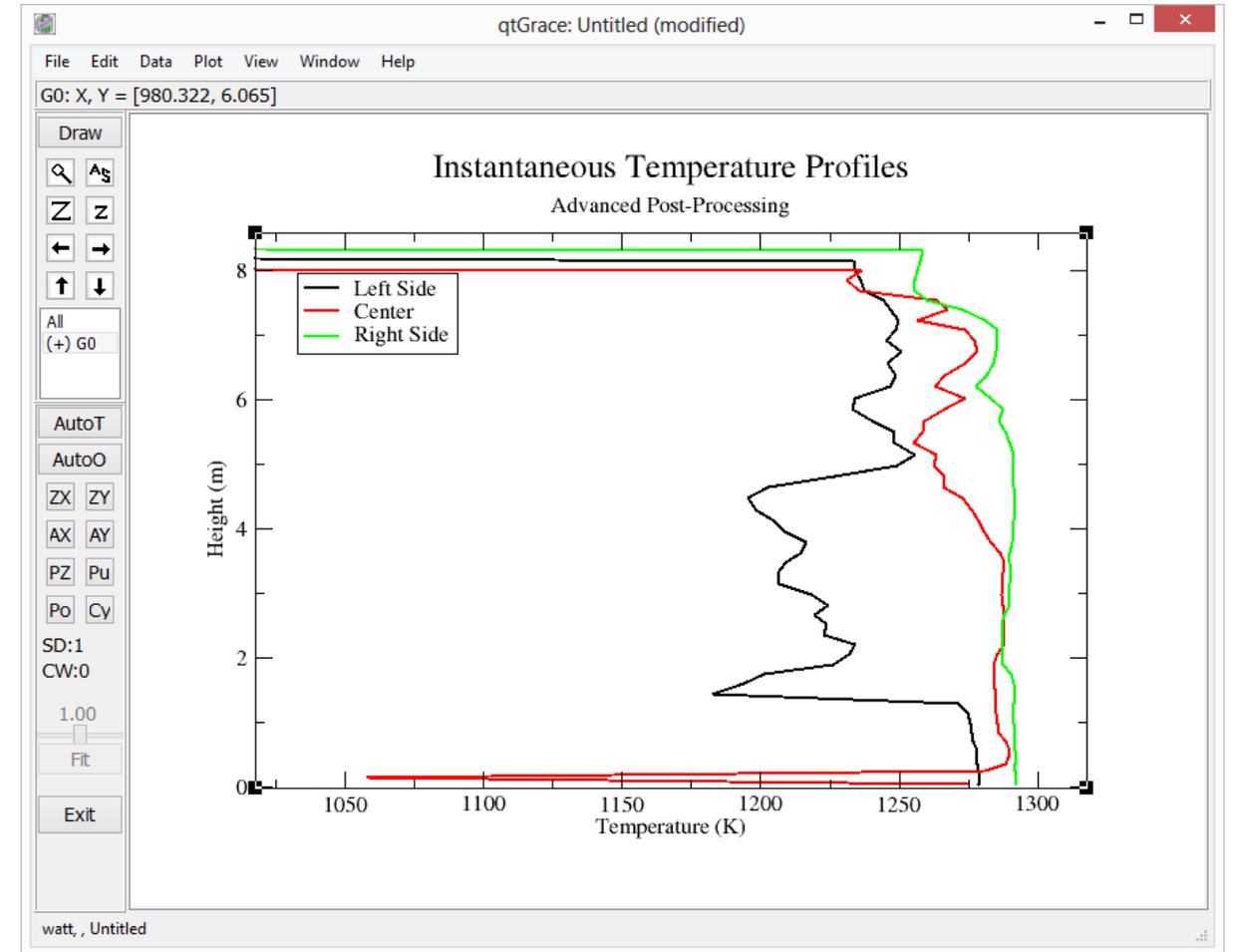
# Use the Zoom Tool to Select a Smaller Region for Plotting

- The 0 and -1 values are not valid temperature data. We want to focus on the temperature region above  $T = 1000$  K.
- Click the **Zoom** tool (magnifying glass)
- Draw a rectangle around the region of interest.
- You can specify exact x- and y-ranges using “World Scaling”
  - On Windows: Plot → Graph Appearance → Viewport
  - On Linux: Plot → World scaling



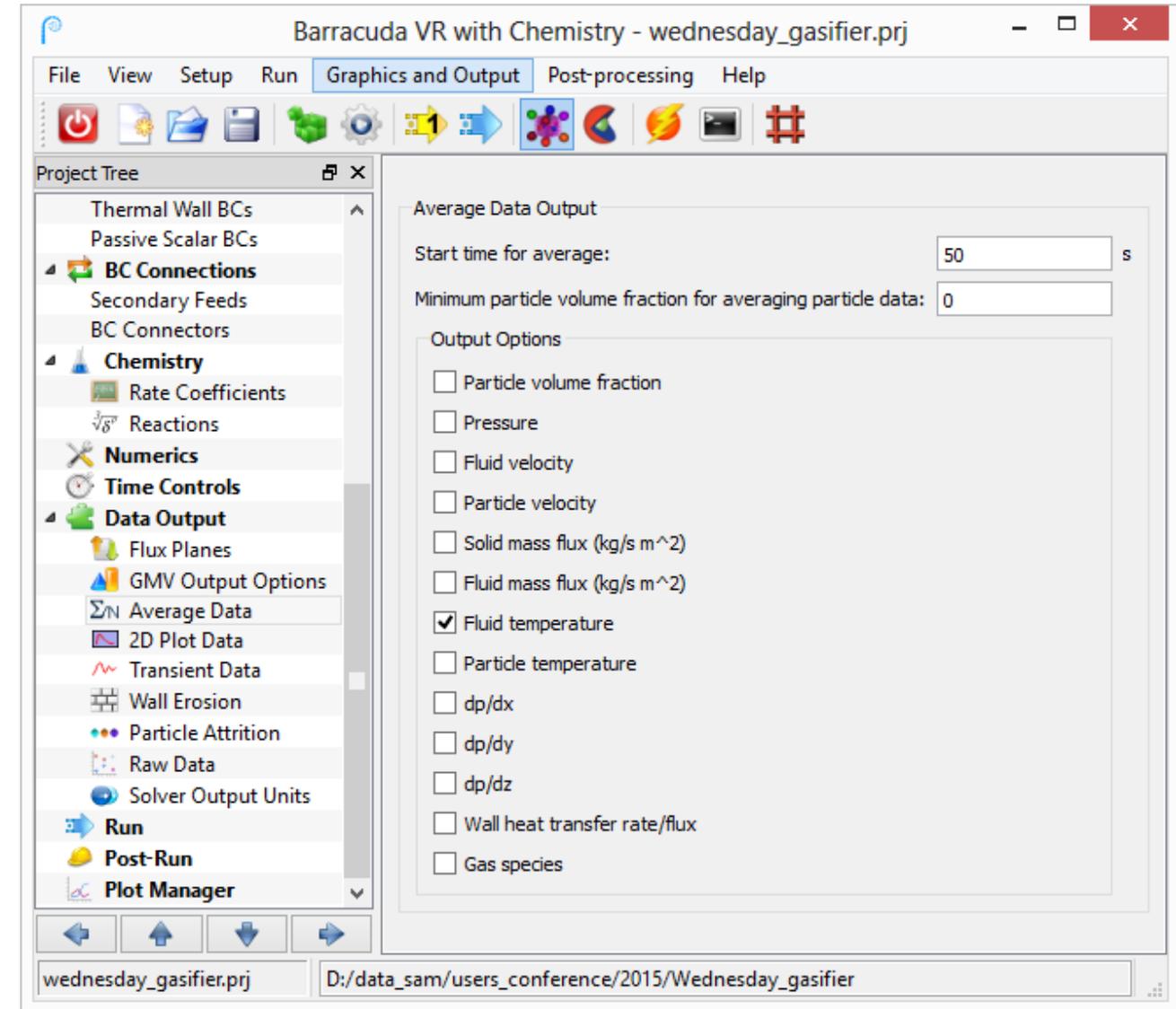
# Final View of Instantaneous Fluid Temperature (2D Plot Data)

- The **Center** curve has a low-temperature spike at the location of the steam sparger.
- The **Left Side** is at the lowest temperature in the upper portion of the vessel.
  - Devolatilization of fresh coal feed (endothermic)
  - The Center and Right Side are hotter, and more uniform from top to bottom.



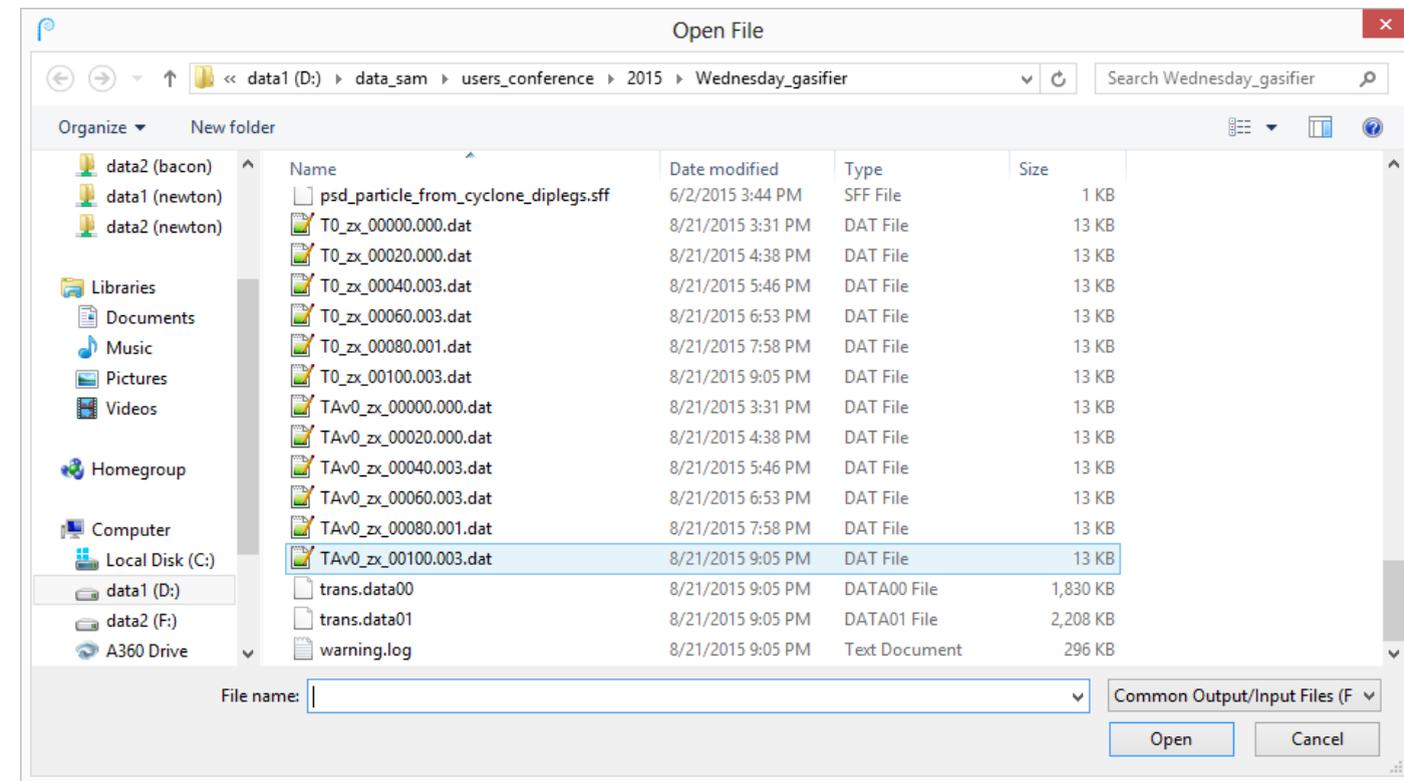
# Time-Averaged Temperature Profiles from 2D Plot Data

- Previous plot was **instantaneous** data at  **$t = 100$  s**.
- In this simulation, **time averaging** was started at  **$t = 50$  s**.
- Time-average data is often very useful in Barracuda VR simulations, because it can better represent “steady state” behavior.



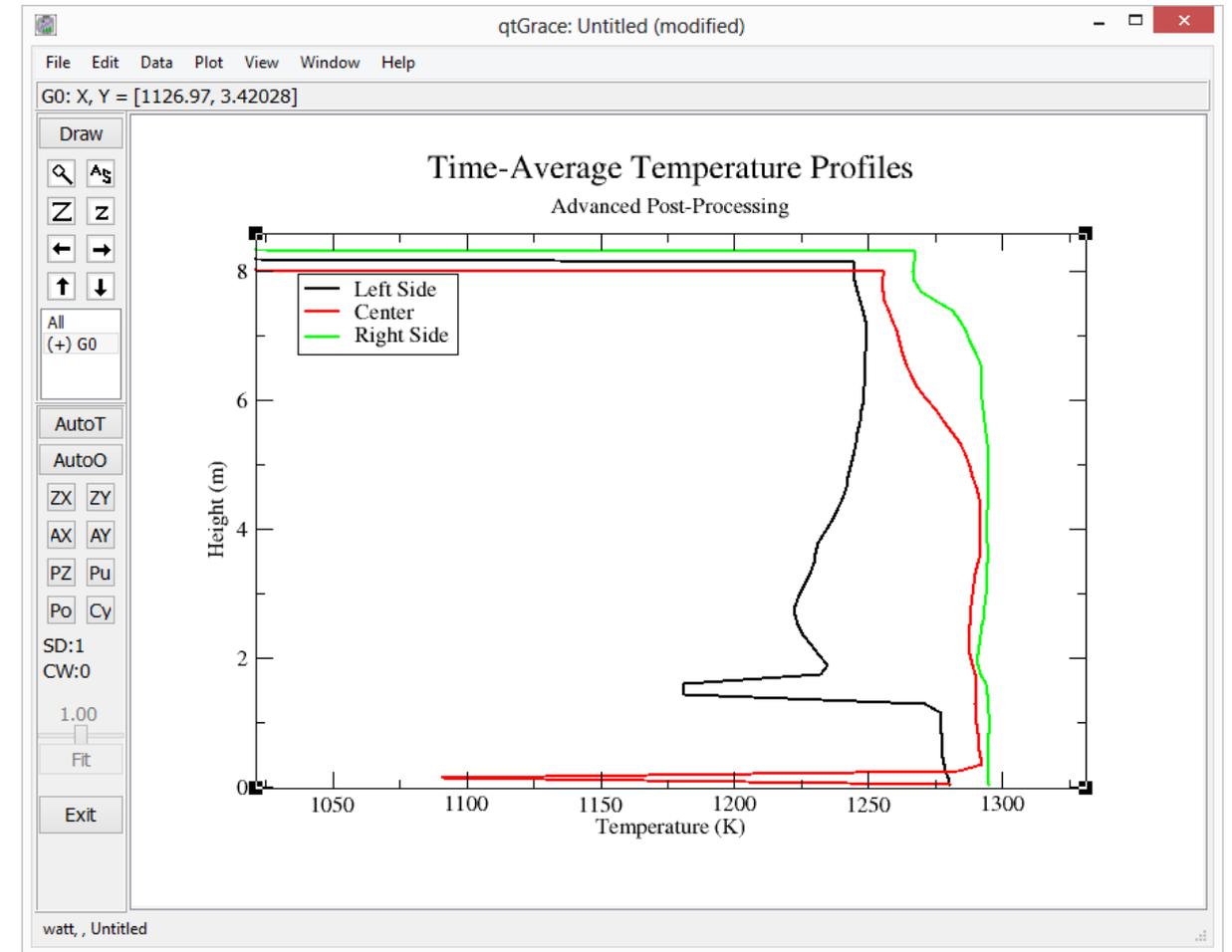
# Select the Time-Average Fluid Temperature File at t=100 s

- Use the **Plot Manager** procedure from previous slides and choose **time-average file** instead of **instantaneous file**
- Note the file naming convention:
  - **TAv0** indicates time-average fluid temperature data.
  - The simulation time,  $t = 100.003$  s, is embedded in the file name.
- Use the **Copy** feature in **Plot Manager**, since this plot is very similar to the previous one.



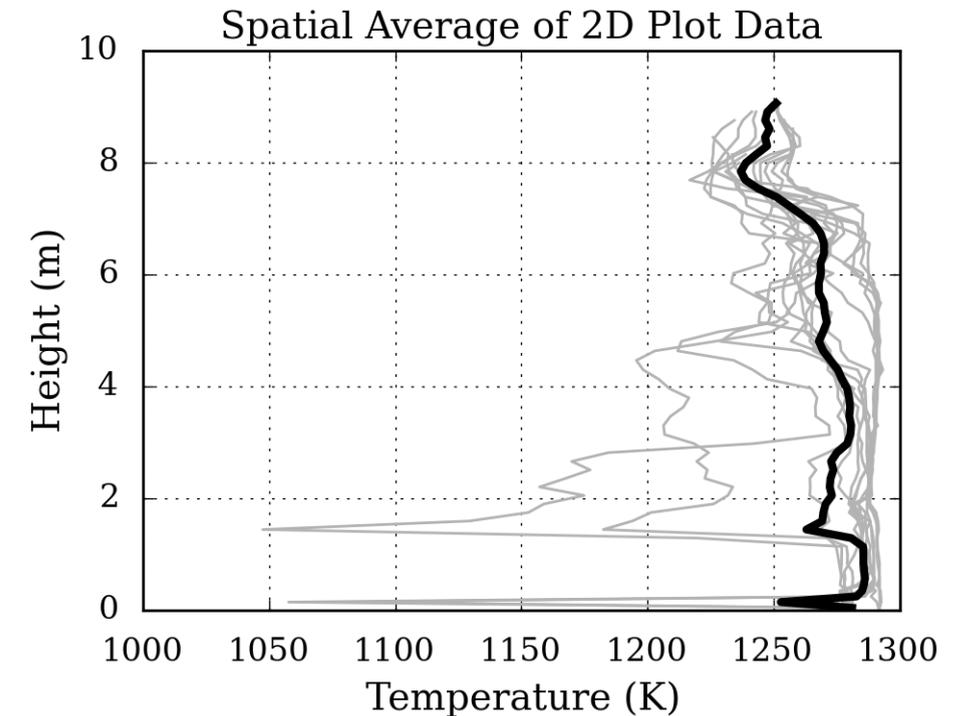
# Final View of Time-Average Fluid Temperature (2D Plot Data)

- Lines are more smooth than instantaneous data (as expected).
- **Left Side** is still showing a lower temperature than **Center** and **Right Side**.
- When plotting 2D Data:
  - Values of 0 and -1 are reported for null cells.
  - Use the Zoom feature (magnifying glass) to select the region of interest.



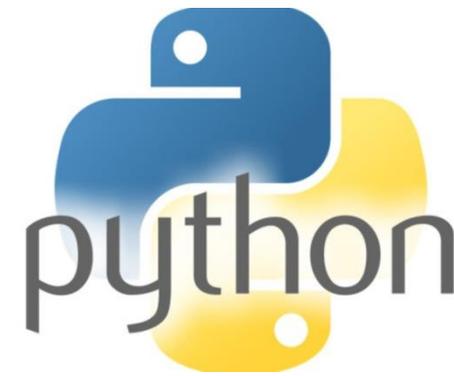
# When **Plot Manager** is not enough

- More complex data analysis may require using a different tool
- Spatial averaging of multiple columns of 2D Plot Data is not supported in **Plot Manager**
- There are many options for approaching this task, including:
  - Excel
  - Matlab
  - Mathematica
  - Enight
  - Python



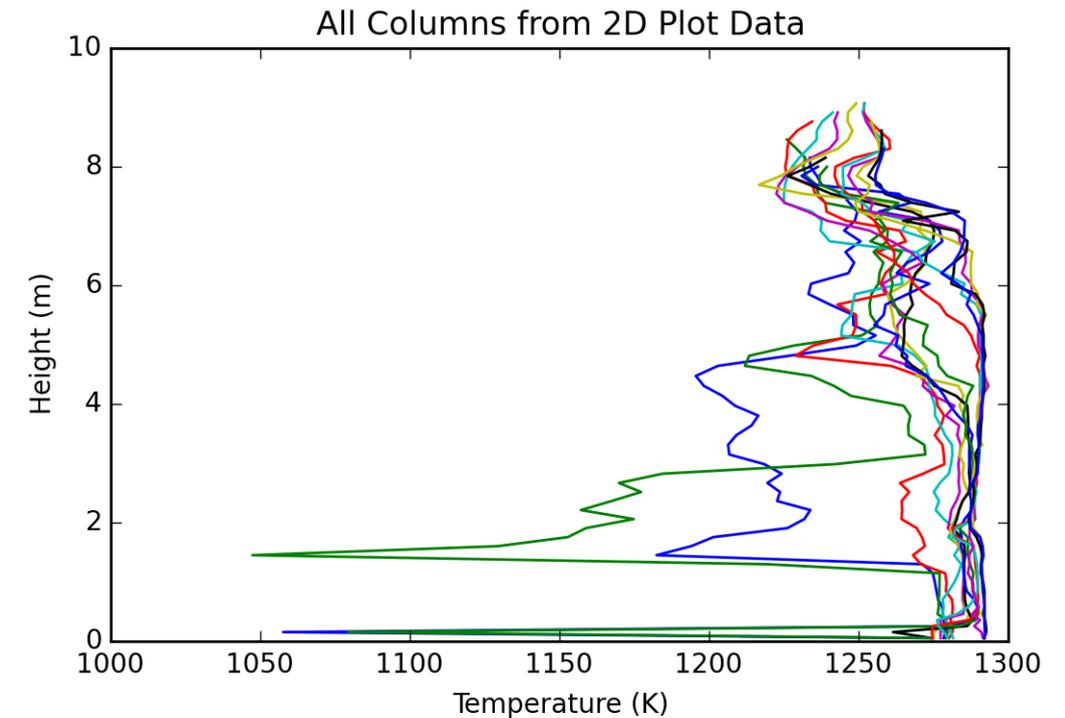
# About Python

- Python is a programming language that is popular among engineers and scientists due to its numerical analysis capabilities.
- Python can be freely downloaded. The distribution we recommend is from <http://continuum.io/downloads>
- A great deal of information about Python is available online and in print
  - <https://docs.python.org/3/>
  - <https://wiki.python.org/moin/PythonBooks>



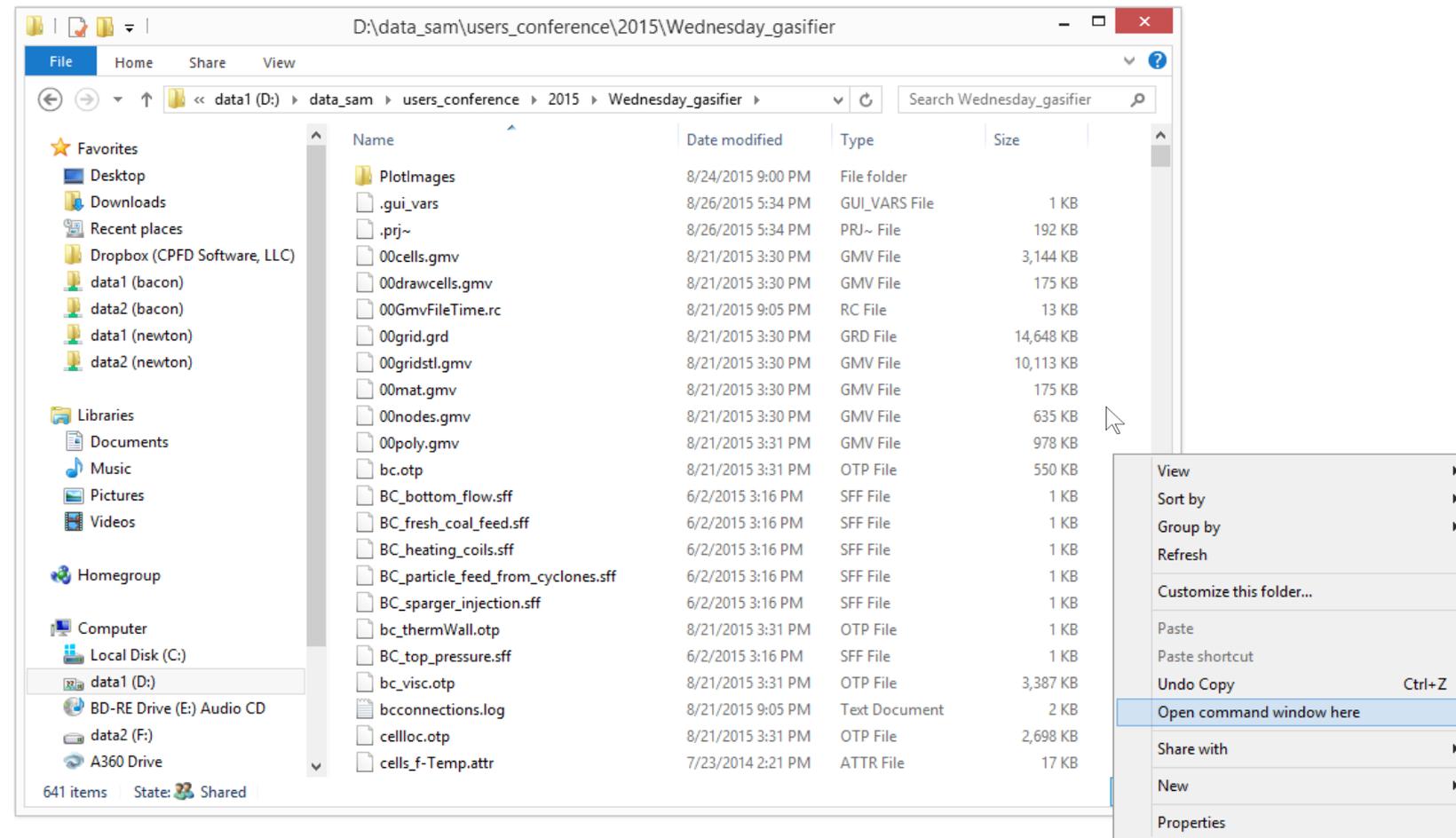
# Calculating Spatial Averages from 2D Plot Data

- Previous plots were for specific x-locations.
- Sometimes it is desirable to calculate a single **spatially-averaged** profile based on all x-locations from the 2D Plot Data output file.
- The 2D Plot Data output file contains 15 columns of data at different x-positions.
- We will calculate a “simple-minded” average of the values in the 2D data file. No weighting based on cell volumes.



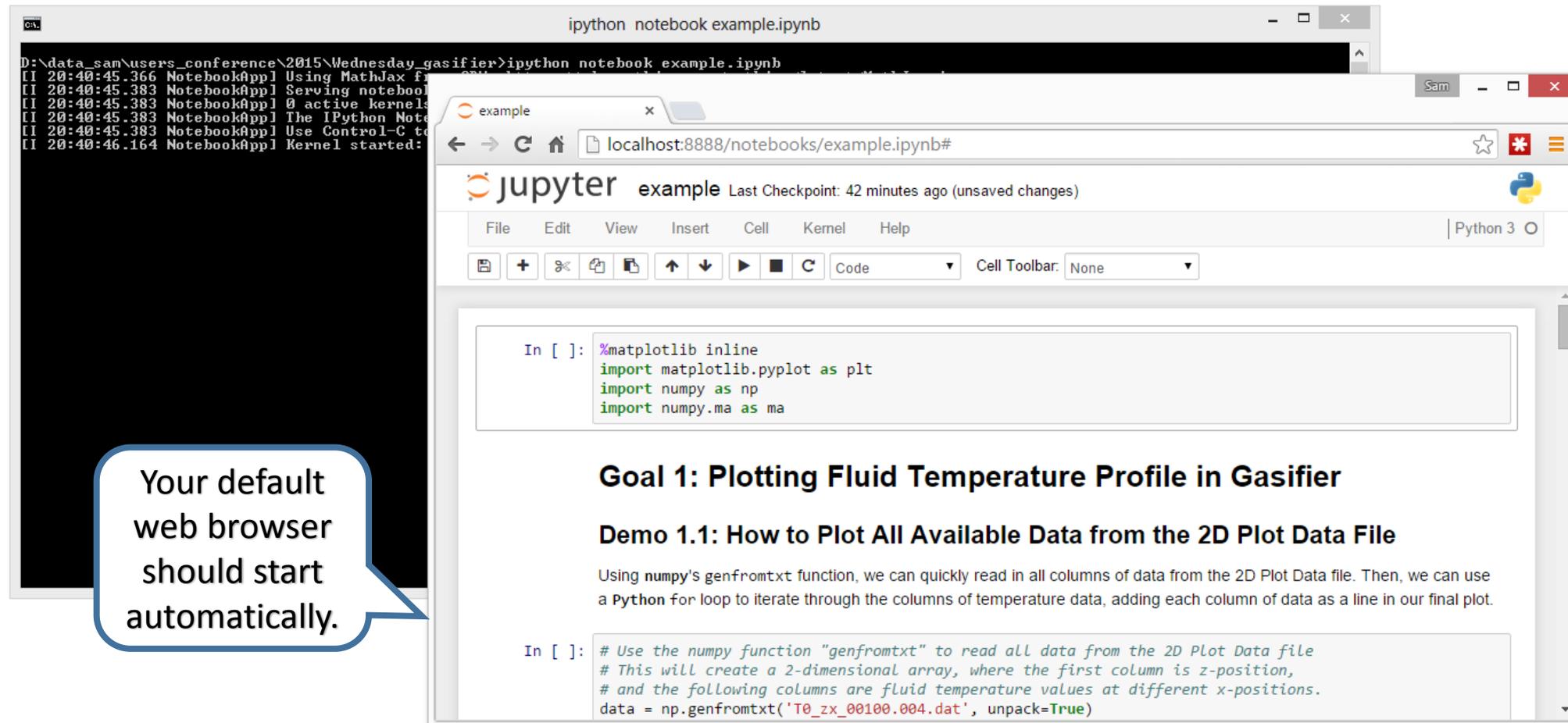
# Open a Command Window in the Current Directory

- On Windows: Shift + Right-click → Open command window here
- On Linux: open a terminal in the normal way



# Start the iPython Notebook Server

- In the command window, type:  
`ipython notebook example.ipynb`



The image shows a terminal window on the left and a Jupyter Notebook browser interface on the right. The terminal window displays the command `ipython notebook example.ipynb` and its output, including messages about using MathJax, serving the notebook, and starting the kernel. The Jupyter Notebook interface shows the URL `localhost:8888/notebooks/example.ipynb#` and the notebook content. The notebook content includes a code cell with the following code:

```
In [ ]: %matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import numpy.ma as ma
```

Below the code cell, there is a section titled "Goal 1: Plotting Fluid Temperature Profile in Gasifier" and a sub-section titled "Demo 1.1: How to Plot All Available Data from the 2D Plot Data File". The text under "Demo 1.1" explains that using `numpy`'s `genfromtxt` function can quickly read in all columns of data from the 2D Plot Data file. Then, a `Python for` loop can be used to iterate through the columns of temperature data, adding each column of data as a line in the final plot.

Below the text, there is another code cell with the following code:

```
In [ ]: # Use the numpy function "genfromtxt" to read all data from the 2D Plot Data file
# This will create a 2-dimensional array, where the first column is z-position,
# and the following columns are fluid temperature values at different x-positions.
data = np.genfromtxt('T0_zx_00100.004.dat', unpack=True)
```

Your default web browser should start automatically.

# Example iPython Notebook Overview

- The first cell has several import commands to bring in the functions we'll be using for reading data, creating plots, etc.

```
In [ ]: %matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import numpy.ma as ma
```

Instead of remembering the exact syntax of the import commands, it is recommended that you keep a template somewhere on your system as a starting point for new iPython notebooks.

- Any additional Python modules you need can be imported similarly, and by convention should be imported at the top of the notebook.

# Demo 1.1: Plotting All Data from a 2D Plot Data File

- The next cell shows how to plot all of the available data from the instantaneous 2D Plot Data file.

```
In [ ]: # Use the numpy function "genfromtxt" to read all data from the 2D Plot Data file
# This will create a 2-dimensional array, where the first column is z-position,
# and the following columns are fluid temperature values at different x-positions.
data = np.genfromtxt('T0_zx_00100.004.dat', unpack=True)
numColumns = data.shape[0]

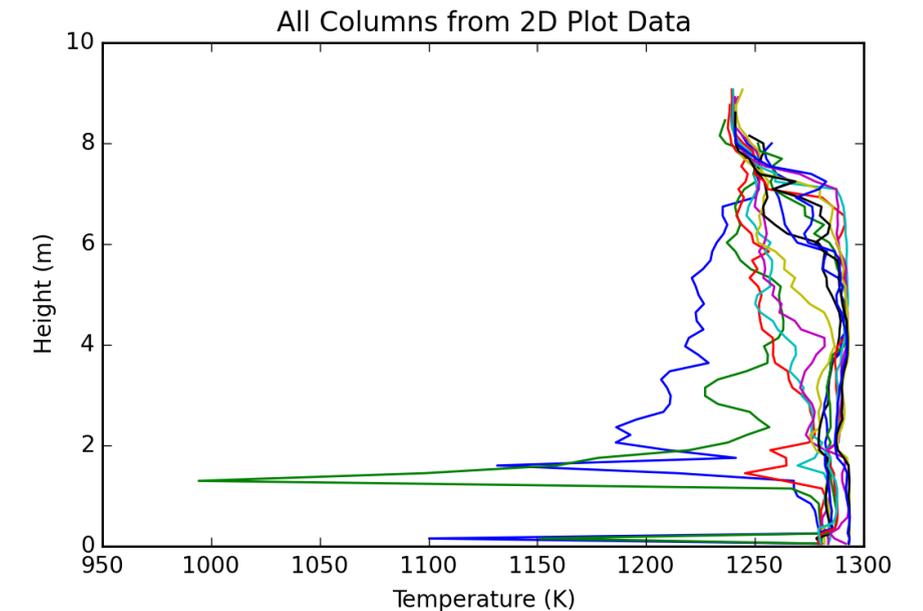
fig, ax = plt.subplots()
ax.set_title('All Columns from 2D Plot Data')
ax.set_xlabel('Temperature (K)')
ax.set_ylabel('Height (m)')

for i in range(1, numColumns):
    x = data[i]
    y = data[0]
    validData = (x > 0)
    ax.plot(x[validData], y[validData])

p = "all_data_columns_from_instantaneous_2d_plot_data"
fig.savefig(p + '.png', format='png', dpi=200)
```

Use this syntax whenever you want to start a new plot.

Add lines to the plot by using the ax.plot( ) command.



# Demo 1.2: Calculating a Spatial Average of 2D Plot Data

- The next cell shows how to use a special feature of `numpy` called “masked arrays” to calculate a spatial average of the **instantaneous** 2D Plot Data for fluid temperature.

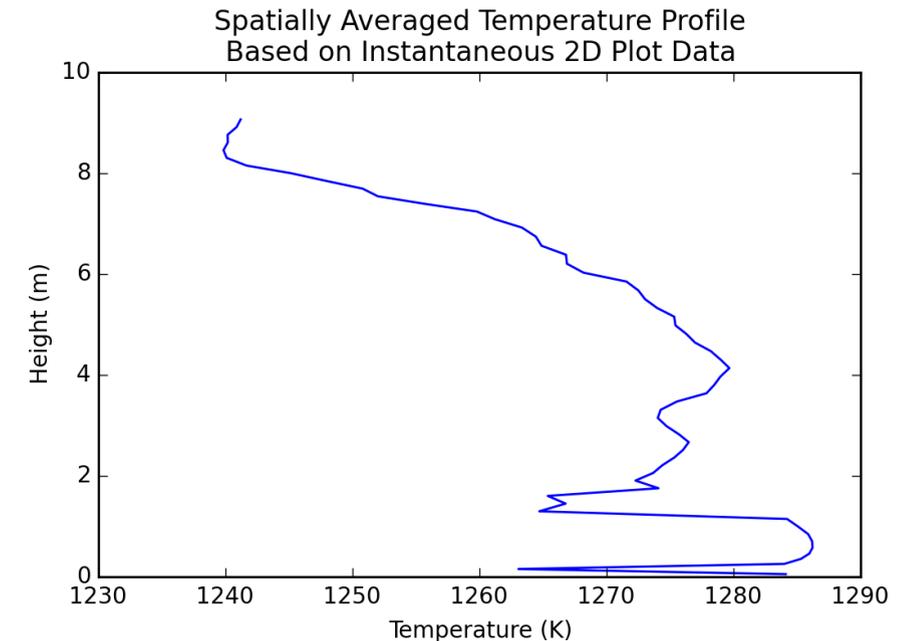
```
In [ ]: # The vessel height is the first column of the 2-dimensional "data" array.
height = data[0]

# We can use numpy's *masked array* feature to calculate a spatial average
# and intentionally ignore values of 0 and -1 reported at null cells in
# Barracuda's 2D Plot Data output files.
x = ma.masked_less_equal(data[1:], 0)
averageTemperature = ma.average(x, axis=0)

fig, ax = plt.subplots()
ax.set_title('Spatially Averaged Temperature Profile\nBased on Instantaneous 2D Plot Data')
ax.set_xlabel('Temperature (K)')
ax.set_ylabel('Height (m)')
ax.plot(averageTemperature, height)

p = "spatial_average_of_instantaneous_2d_plot_data"
fig.savefig(p + '.png', format='png', dpi=200)
```

These two lines calculate the spatial average, ignoring values  $\leq 0$ .



# Demo 1.3: Spatial Average of Time-Average 2D Plot Data

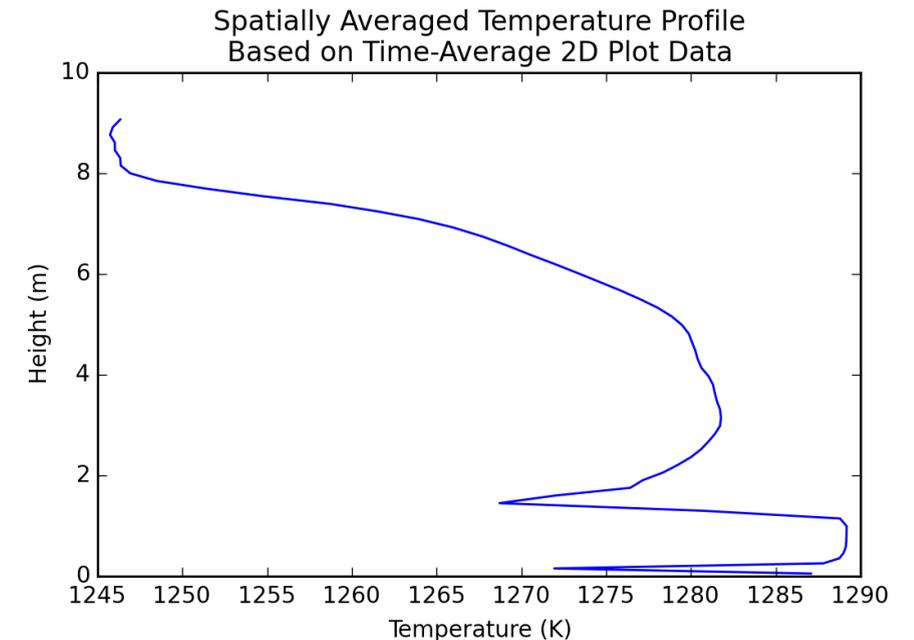
- The next cell is very similar, but in this case we use the **time-average** 2D Plot Data file.

```
In [ ]: # Following the principles above, we can calculate the spatial average
# for *time-average* temperature data.
data = np.genfromtxt('TAv0_zx_00100.004.dat', unpack=True)
height = data[0]
x = ma.masked_less_equal(data[1:], 0)
averageTemperature = ma.average(x, axis=0)

fig, ax = plt.subplots()
ax.set_title('Spatially Averaged Temperature Profile\nBased on Time-Average 2D Plot Data')
ax.set_xlabel('Temperature (K)')
ax.set_ylabel('Height (m)')
ax.plot(averageTemperature, height);

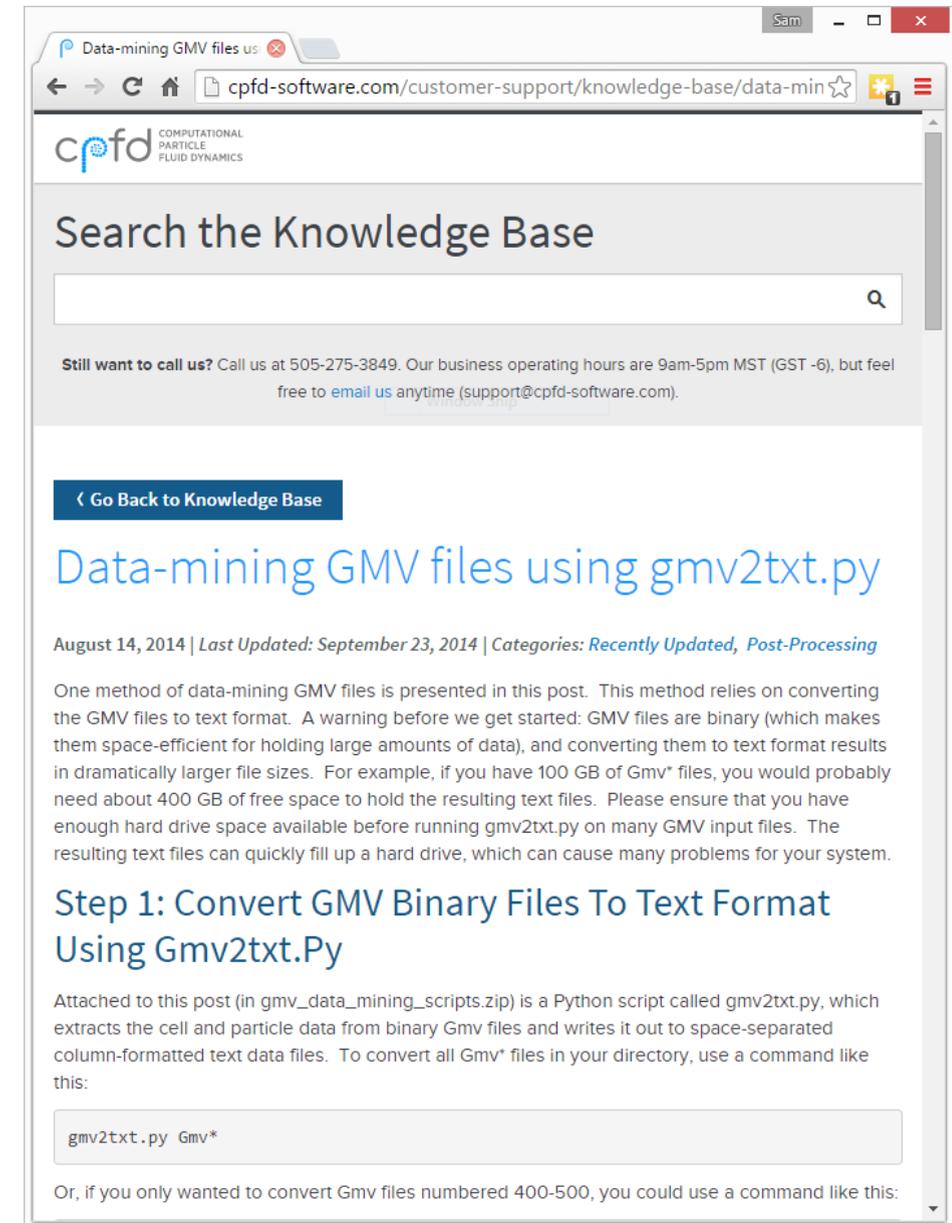
p = "spatial_average_of_time-average_2d_plot_data"
fig.savefig(p + '.png', format='png', dpi=200)
```

The file name passed to `genfromtxt()` is the only change.



# Further Reading for Better Spatial Averaging

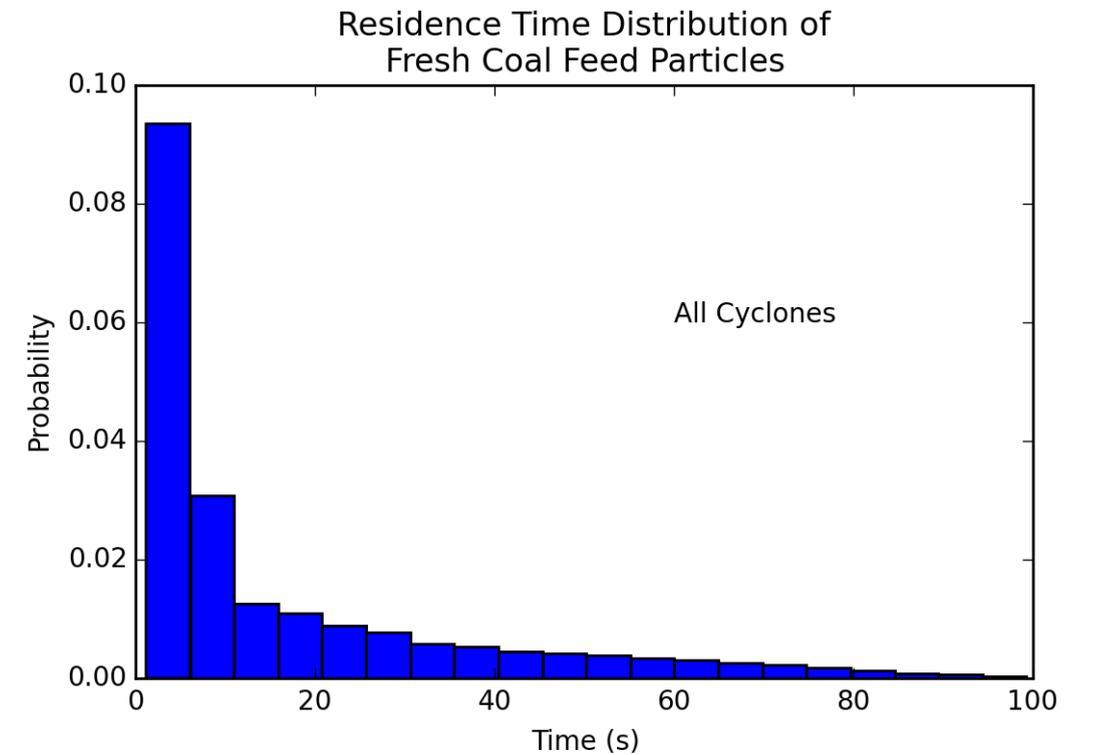
- As stated previously, the spatial average calculated here using the 2D Plot Data is “simple-minded”
  - Numerical average of the data
  - No weighting based on cell size
  - Only uses data from one 2D plane
- More sophisticated methods are available for calculating spatial averages to address these shortcomings. One specific method is discussed in the CPFD Support Site post [Data-Mining GMV Files Using gmv2txt.py](#). See this post for complete details.



The screenshot shows a web browser window with the URL [cpfd-software.com/customer-support/knowledge-base/data-min](http://cpfd-software.com/customer-support/knowledge-base/data-min). The page features the CPFD logo (Computational Particle Fluid Dynamics) and a search bar. Below the search bar, there is a contact information section: "Still want to call us? Call us at 505-275-3849. Our business operating hours are 9am-5pm MST (GST -6), but feel free to email us anytime (support@cpfd-software.com)." A blue button labeled "Go Back to Knowledge Base" is visible. The main content area displays the article title "Data-mining GMV files using gmv2txt.py" in blue, followed by the date "August 14, 2014 | Last Updated: September 23, 2014" and categories "Recently Updated, Post-Processing". The article text begins with "One method of data-mining GMV files is presented in this post. This method relies on converting the GMV files to text format. A warning before we get started: GMV files are binary (which makes them space-efficient for holding large amounts of data), and converting them to text format results in dramatically larger file sizes. For example, if you have 100 GB of Gmv\* files, you would probably need about 400 GB of free space to hold the resulting text files. Please ensure that you have enough hard drive space available before running gmv2txt.py on many GMV input files. The resulting text files can quickly fill up a hard drive, which can cause many problems for your system." The article then introduces "Step 1: Convert GMV Binary Files To Text Format Using Gmv2txt.Py" and mentions a Python script named gmv2txt.py. A code block shows the command `gmv2txt.py Gmv*`. The text concludes with "Or, if you only wanted to convert Gmv files numbered 400-500, you could use a command like this:".

# Goal 2: Residence Time Distribution (RTD) of Particles Exiting System

- Achieving Goal 2: Using **Raw Particle Data at Flux Planes**
  - Introduction to Raw Particle Data
    - What is it?
    - How to enable it?
  - Calculating and plotting RTD using Python



# Raw Particle Data at Boundary Condition Flux Planes

- Detailed information is recorded for each computational particle that crosses the BC flux plane.
  - Time at which particle crossed plane
  - Unique particle ID
  - Cloud mass
  - Particle radius
  - Residence time
  - Species
  - Material composition

```
1 # Cyclone 1 pressure BC
2 #
3 #@ 1 "Time" "s"
4 #@ 2 "Unique particle ID" ""
5 #@ 3 "Particle cloud mass" "kg"
6 #@ 4 "Particle radius" "micron"
7 #@ 5 "Particle density" "kg/m^3"
8 #@ 6 "Number of particles in cloud" ""
9 #@ 7 "Residence time (on)" "s"
10 #@ 8 "Species" ""
11 #@ 9 "Particle x-velocity" "m/s"
12 #@ 10 "Particle y-velocity" "m/s"
13 #@ 11 "Particle z-velocity" "m/s"
14 #@ 12 "Particle temperature" "K"
15 #@ 13 "Mass fraction of solid material (Ash)" ""
16 #@ 14 "Mass fraction of solid material (C)" ""
17 #@ 15 "Mass fraction of non-solid material (Volatiles)" ""
18 7.376229e+00 394899 1.184436e-03 2.273864e+01 8.212680e+02 2.928497e+07 3.047600e+00 2 3.
19 7.508660e+00 394952 9.154500e-04 4.275332e+01 8.264344e+02 3.383983e+06 3.081339e+00 2 4.
20 7.518999e+00 393263 8.805444e-04 5.391718e+01 8.006598e+02 1.675071e+06 5.019400e+00 2 4.
21 7.612704e+00 394030 1.185997e-03 4.076749e+01 8.067776e+02 5.179636e+06 4.925256e+00 2 3.
22 7.621108e+00 393070 1.128864e-03 5.707190e+01 7.983230e+02 1.815965e+06 5.171938e+00 2 4.
23 7.754625e+00 394895 8.774637e-04 2.221392e+01 7.991252e+02 2.391387e+07 3.433623e+00 2 2.
24 7.772742e+00 394896 8.865401e-04 7.215038e+01 8.073552e+02 6.979580e+05 3.451741e+00 2 4.
25 7.792435e+00 393690 1.088730e-03 5.346482e+01 7.977591e+02 2.131845e+06 5.236681e+00 2 4.
26 7.820005e+00 394832 9.012132e-04 3.397386e+01 7.935011e+02 6.914433e+06 3.616411e+00 2 3.
27 7.839698e+00 393249 8.636590e-04 7.070242e+01 7.853039e+02 7.428703e+05 5.340099e+00 2 4.
28 7.844621e+00 394875 8.401807e-04 2.022194e+01 7.873912e+02 3.080529e+07 3.561802e+00 2 3.
29 7.859391e+00 394841 9.892950e-04 3.501765e+01 7.881765e+02 6.978352e+06 3.644092e+00 2 3.
30 7.864314e+00 394854 9.148053e-04 4.186526e+01 7.973009e+02 3.732988e+06 3.623293e+00 2 2.
31 7.950361e+00 394845 9.509896e-04 5.568040e+01 7.981462e+02 1.647774e+06 3.725416e+00 2 3.
```

# Enabling Raw Particle Data Output for BC Flux Planes

- Pressure, Flow, and Injection BCs support Raw Particle Data output.

Specify a Flux file name

Click the checkbox.

# Demo 2.1: Plotting Residence Time Distribution (Single Cyclone)

- This cell demonstrates the basic procedure for plotting RTD as a histogram.

```
In [ ]: # From the flux plane raw particle data, we only care about these columns:
#@ 7 "Residence time (s)" "s"
#@ 8 "Species" ""
# Note that genfromtxt's usecols function requires columns counted from 0,
# so subtract 1 from the Barracuda header numbers, which start at 1.
resTime, species = np.genfromtxt('FLUXBC_cyclone_1_pressure_raw_particle', usecols=(6,7), unpack=True)

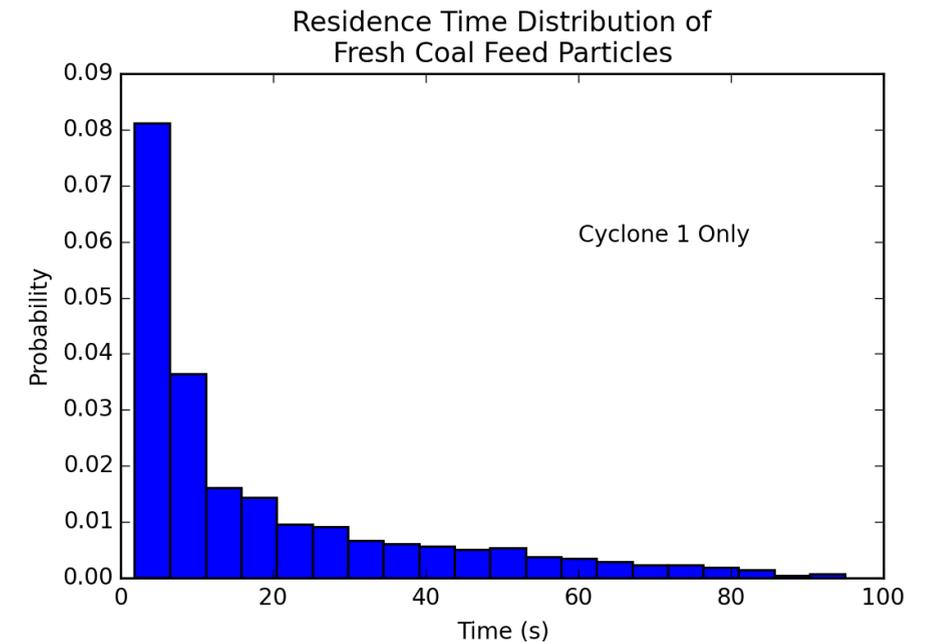
# We only care about *Species 2* particles, Fresh Coal Feed
selectedSpecies = (species == 2)

print("Number of data points all species = ", len(resTime))
print("Number of data points (species 2) = ", len(resTime[selectedSpecies]))

fig, ax = plt.subplots()
ax.set_title('Residence Time Distribution of\nFresh Coal Feed Particles')
ax.set_xlabel('Time (s)')
ax.set_ylabel('Probability')
ax.annotate('Cyclone 1 Only', xy=(60,0.06))
ax.hist(resTime[selectedSpecies], bins=20, normed=True);

p = "pdf_rtd_of_fresh_coal_feed_particles_cyclone_1"
fig.savefig(p + '.png', format='png', dpi=200)
```

File naming convention.



## Demo 2.2: Plotting RTD Based on Data from All Cyclones

- The next cell extends the plot to include data from all 4 cyclones.

```
In [ ]: fList = ['FLUXBC_cyclone_1_pressure_raw_particle',
                'FLUXBC_cyclone_2_pressure_raw_particle',
                'FLUXBC_cyclone_3_pressure_raw_particle',
                'FLUXBC_cyclone_4_pressure_raw_particle']

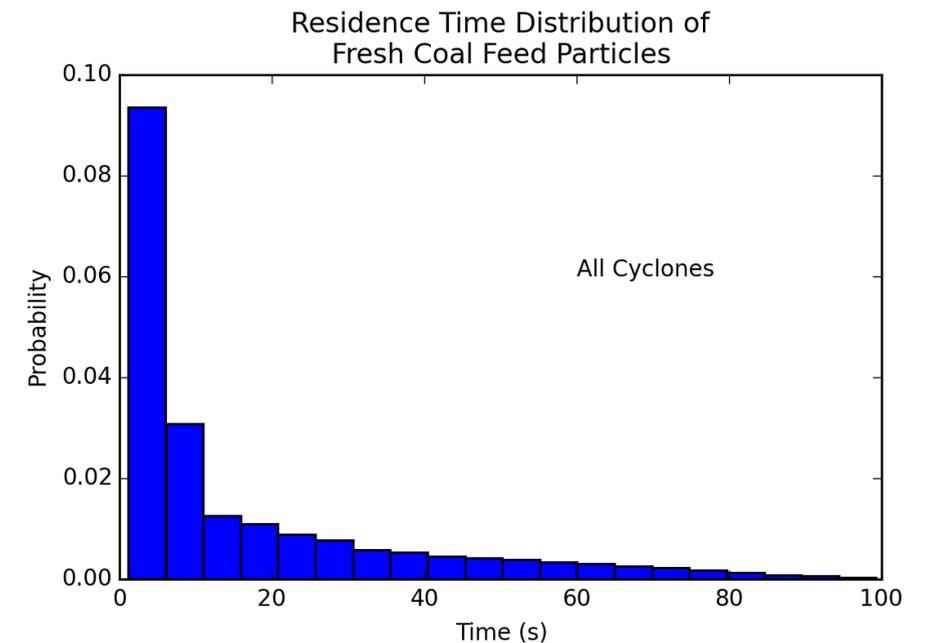
resTime = []
species = []
for f in fList:
    resTime = np.concatenate((resTime, np.genfromtxt(f, usecols=(6))))
    species = np.concatenate((species, np.genfromtxt(f, usecols=(7))))

selectedSpecies = (species == 2)

print("Number of data points all species = ", len(resTime))
print("Number of data points (species 2) = ", len(resTime[selectedSpecies]))

fig, ax = plt.subplots()
ax.set_title('Residence Time Distribution of\nFresh Coal Feed Particles')
ax.set_xlabel('Time (s)')
ax.set_ylabel('Probability')
ax.annotate('All Cyclones', xy=(60,0.06))
ax.hist(resTime[selectedSpecies], bins=20, normed=True);

p = "pdf_rtd_of_fresh_coal_feed_particles_all_cyclones"
fig.savefig(p + '.png', format='png', dpi=200)
```



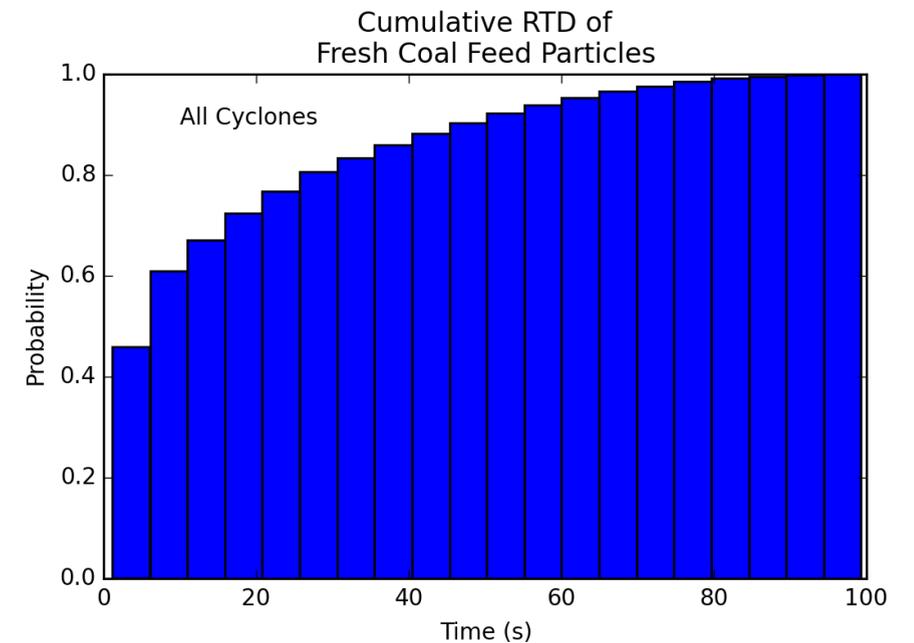
# Demo 2.3: Plotting RTD as a Cumulative Distribution Function

- The final cell shows:
  - How to plot RTD as a cumulative distribution function
  - The concept of using data calculated in a previous iPython notebook cell

```
In [ ]: fig, ax = plt.subplots()
ax.set_title('Cumulative RTD of\nFresh Coal Feed Particles')
ax.set_xlabel('Time (s)')
ax.set_ylabel('Probability')
ax.annotate('All Cyclones', xy=(10,0.9))
ax.hist(resTime[selectedSpecies], bins=20, normed=True, cumulative=True);

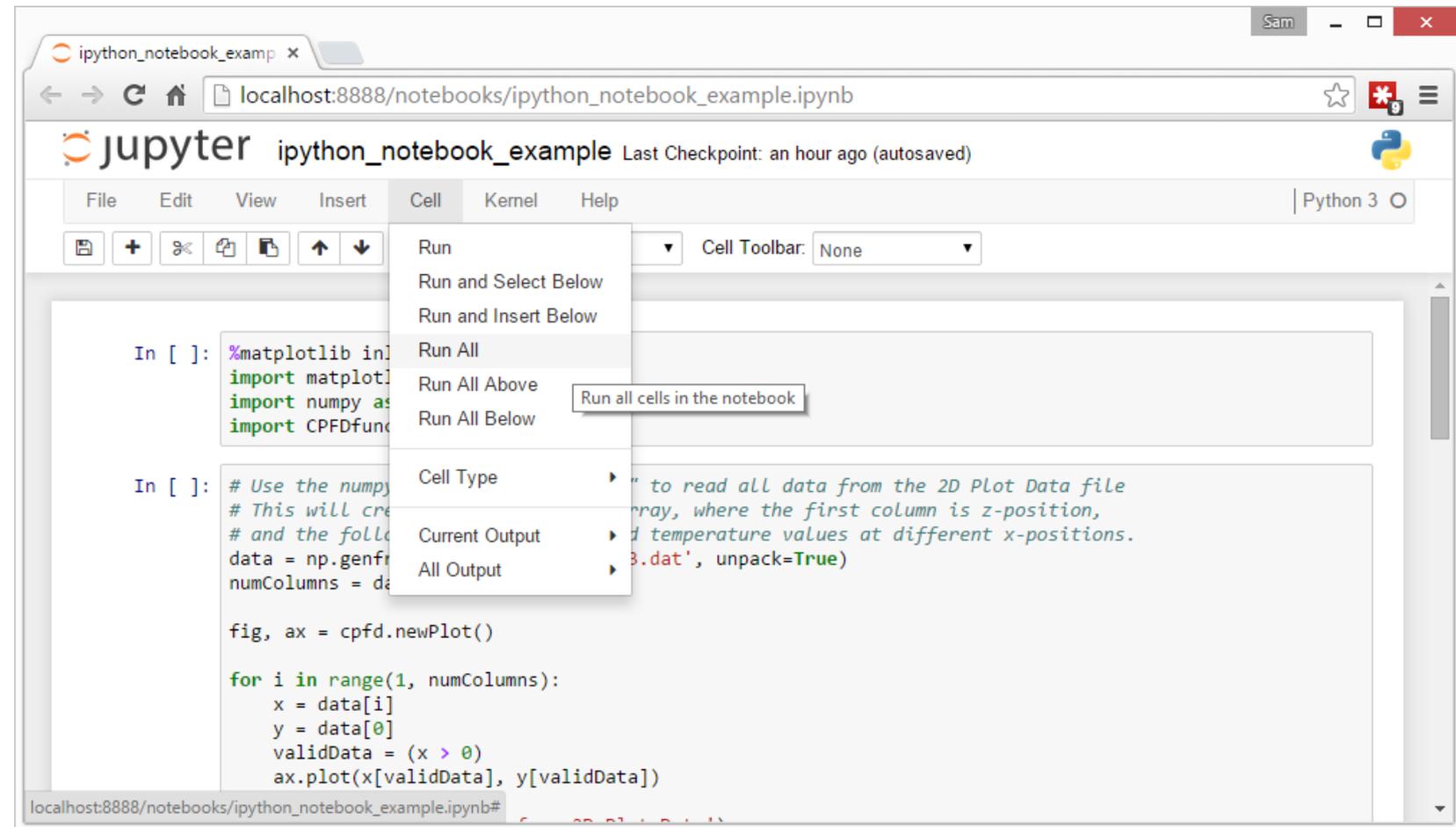
p = "cdf_rtd_of_fresh_coal_feed_particles_all_cyclones"
fig.savefig(p + '.png', format='png', dpi=200)
```

Using the cumulative option changes plot from probability to cumulative.



# How to Run the Notebook

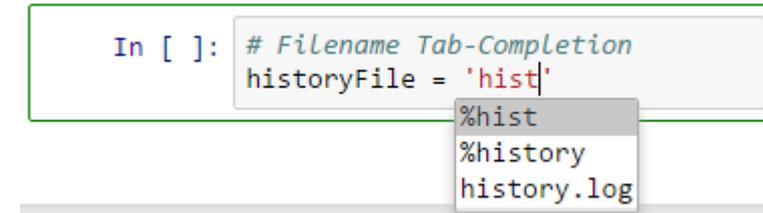
- The entire notebook can be run at once by using: Cell → Run All
- An individual cell can be run by using:
  - Cell → Run
  - Ctrl + Enter



# Other Notable Features of iPython Noteboook

- Tab-completion of commands and filenames
  - When typing, press Tab and a dropdown will appear with available options
- Markdown
  - Cell → Cell Type → Markdown
  - Allows you to document your work
  - Supports LaTeX equations
  - Supports headers and other formatting

```
In [ ]: # Filename Tab-Completion
historyFile = 'hist|'
```



```
# My first heading
## My second heading

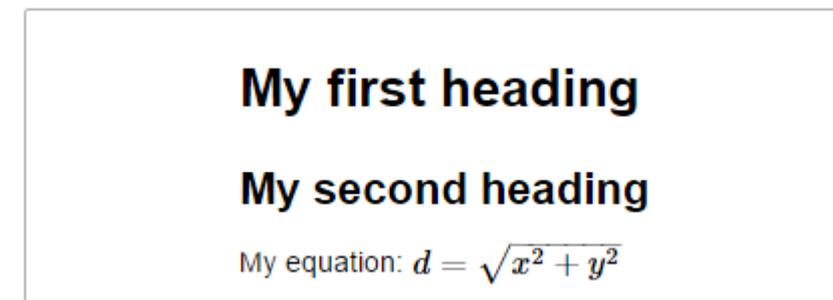
My equation: $d = \sqrt{x^2 + y^2}$
```



**My first heading**

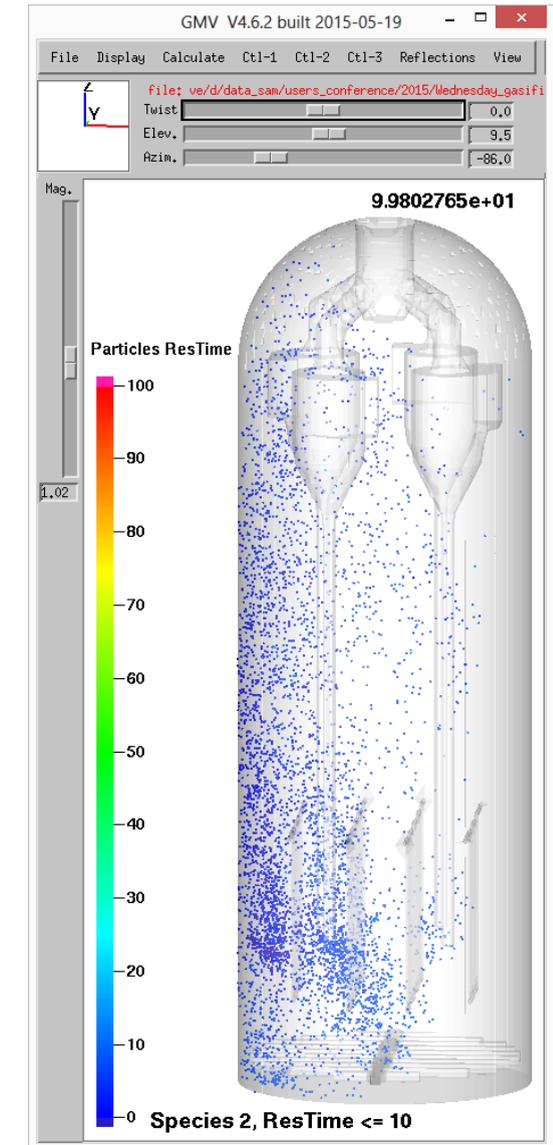
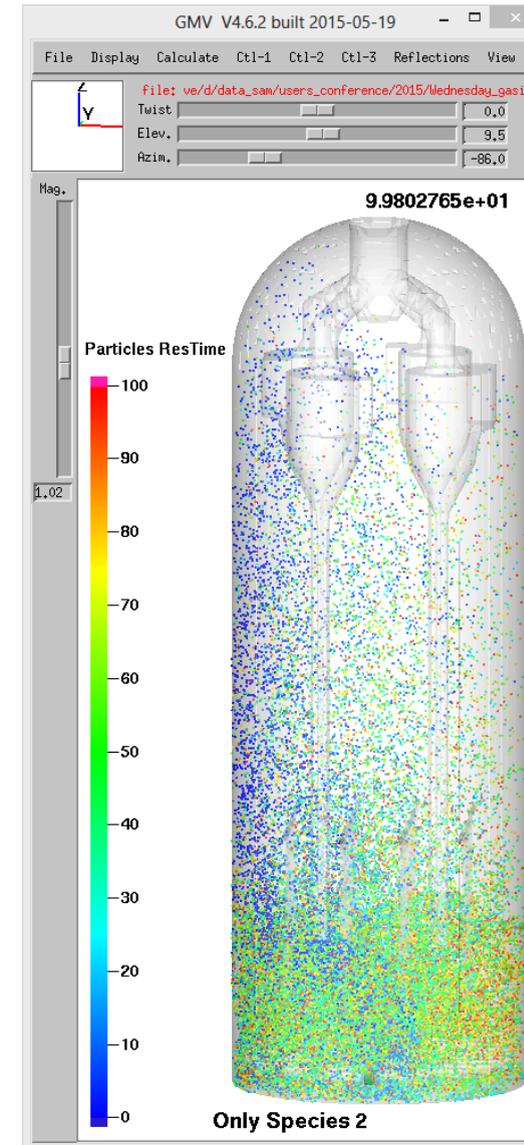
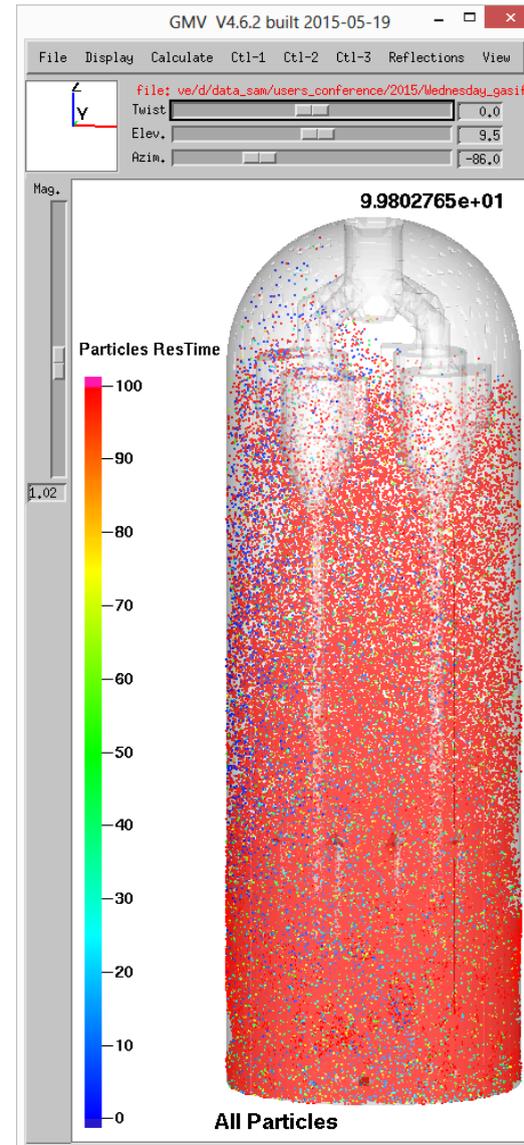
**My second heading**

My equation:  $d = \sqrt{x^2 + y^2}$

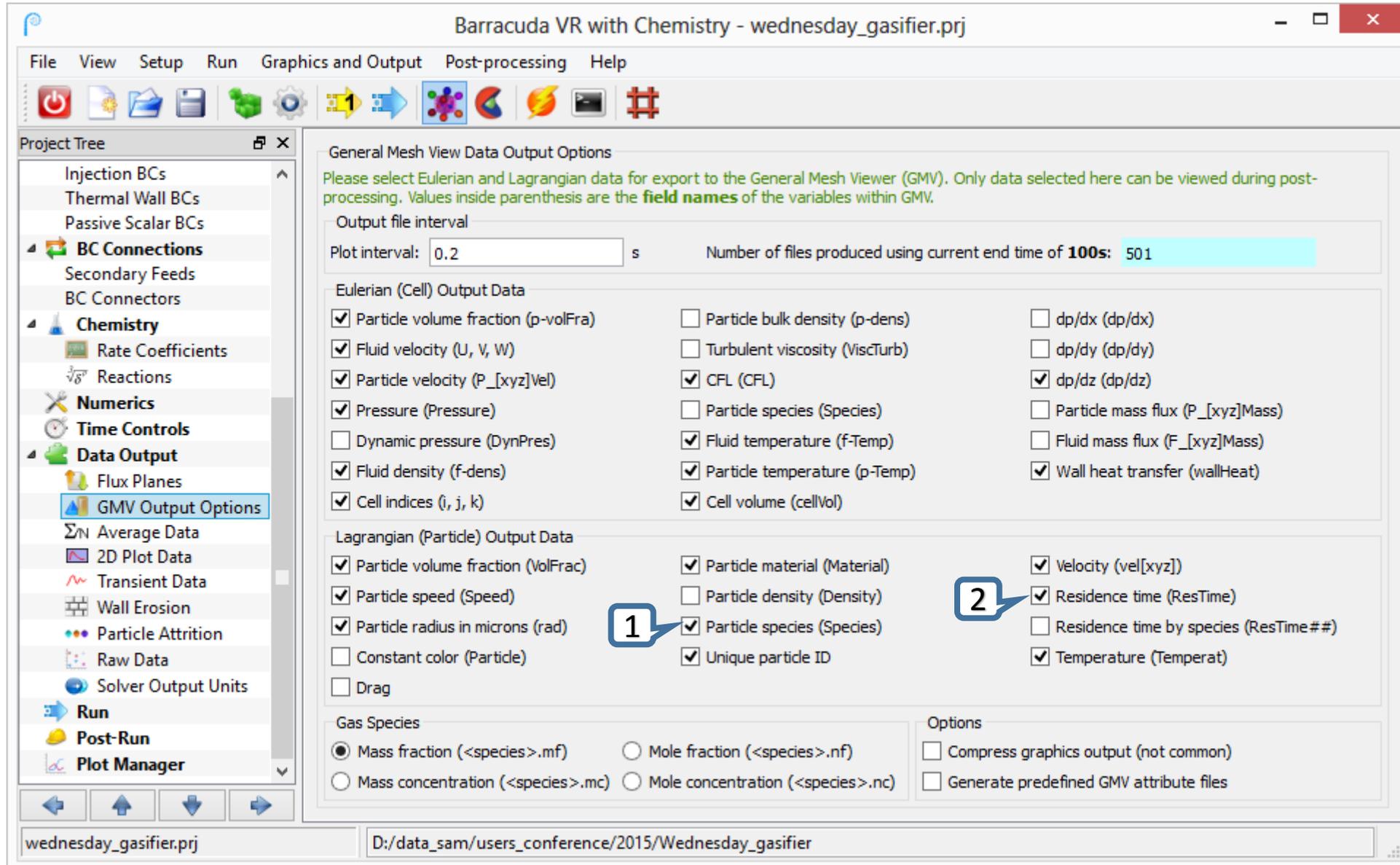


# Goal 3: Viewing a Single Particle Species in GMV

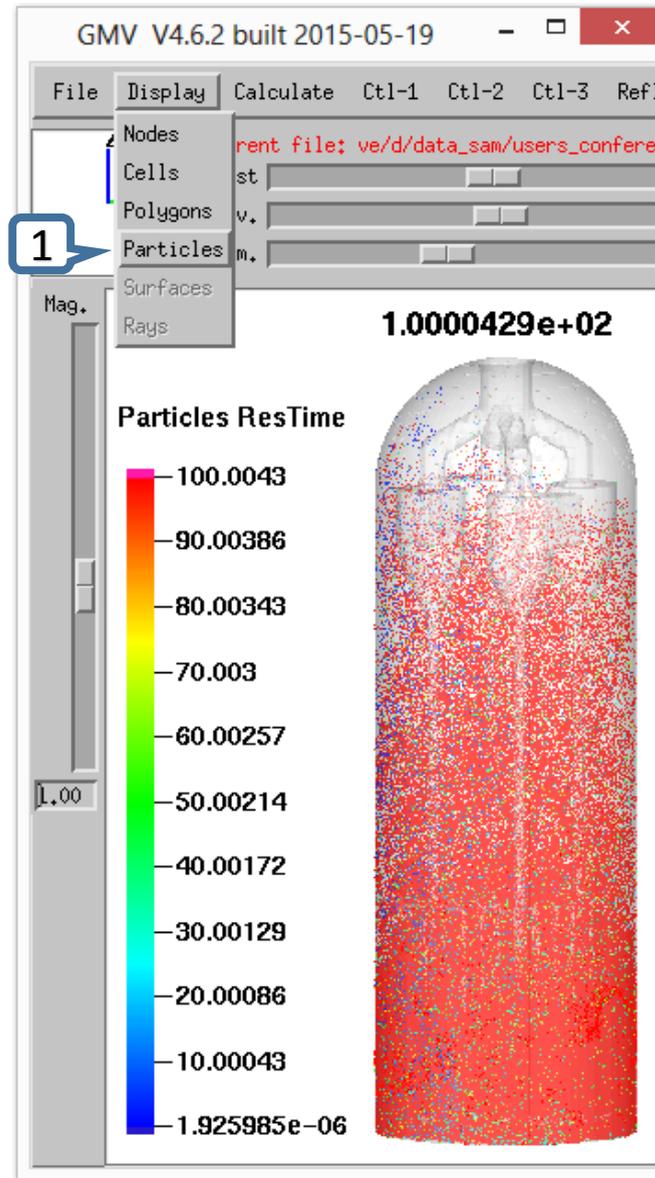
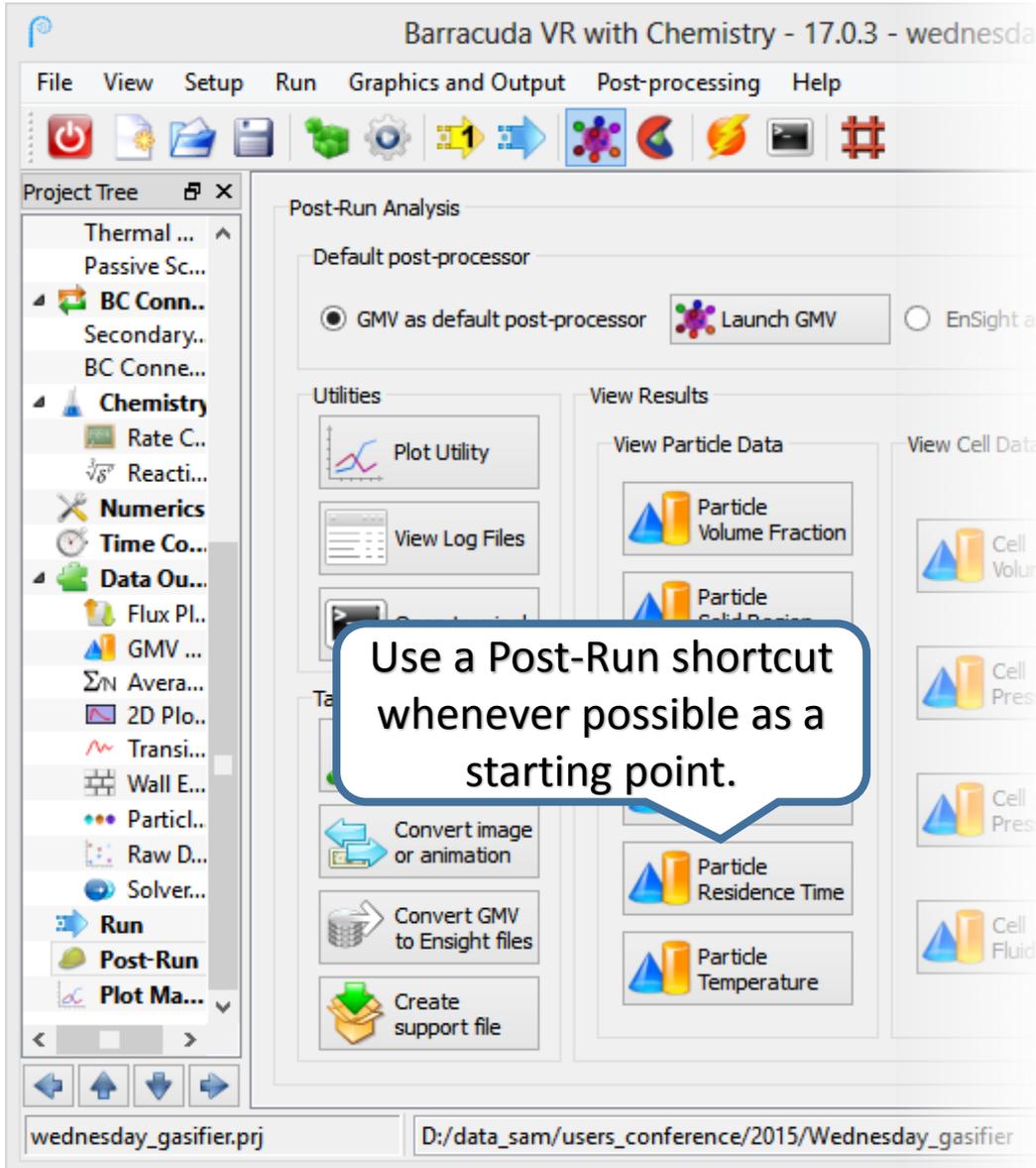
- Achieving Goal 3: Using Particle Select in GMV
  - Introduction to Particle Select
  - Selecting based on multiple criteria



# Variables Used in GMV Must Be Selected in GMV Output Options



# Using Particle Select to Show Only Species 2



2

3

4

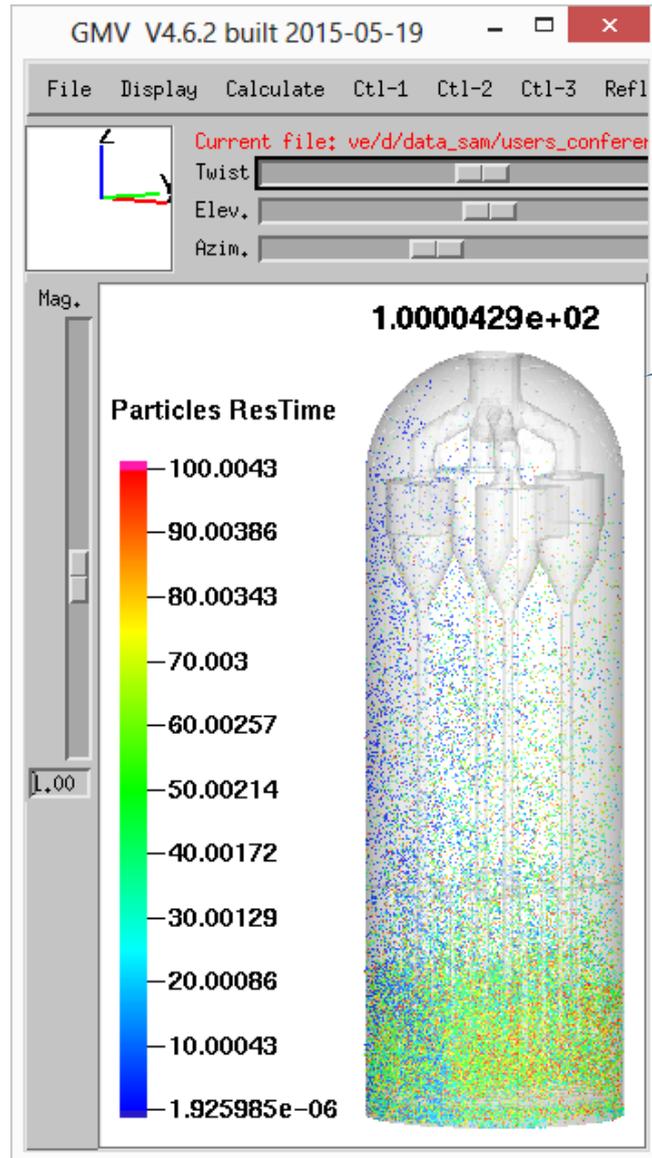
5

6

7  
Min=2  
Max=2

8

# Selecting Particles Based on Multiple Criteria



View resulting from steps shown on previous slide. Only particles with Species = 2 are shown.

View resulting from steps shown on this slide. Only particles with Species = 2 and ResTime ≤ 10 are shown.

Particle Select - Part...

Field: ResTime  **1**

Min: 1.9259848614e-06  
Max: 100.00428772

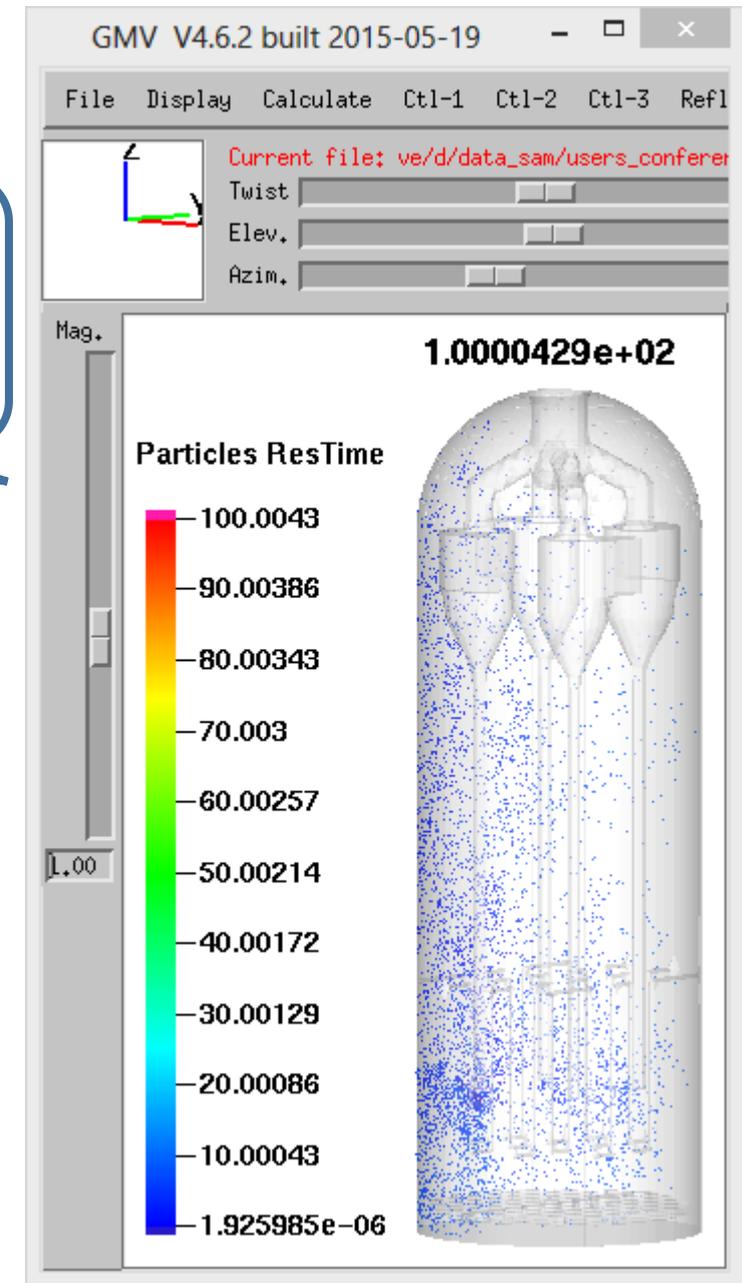
RANGE  
Min:  **3**  
Max:  **3**

S...

VolFrac  
pid  
rad  
Species  
mf-Ash  
mf-C  
mf-volatile  
Temperat  
**ResTime** **2**  
Speed  
velx  
vely  
velz  
SIZE

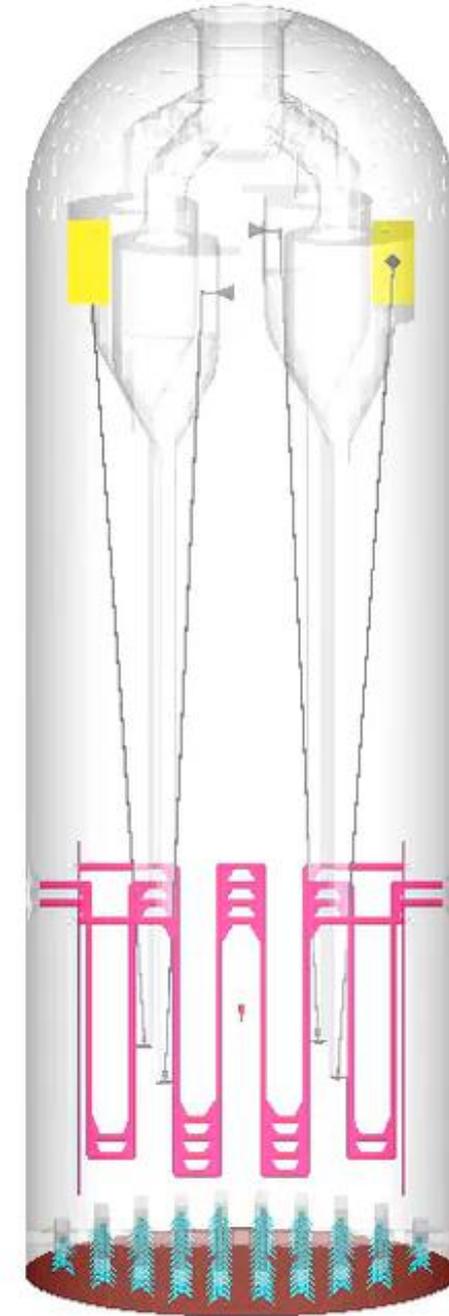
Particle Select   **4**

Select On Particle Field Data Range  
 Select On Search Sphere  
 Select On Particle Numbers  
 Select On Search Box



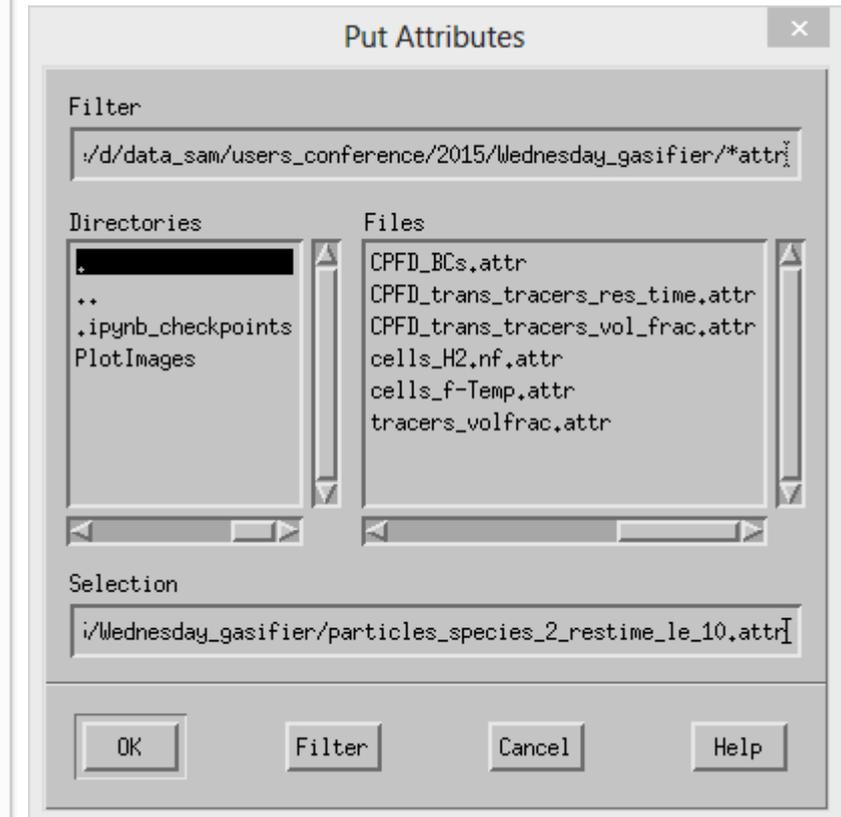
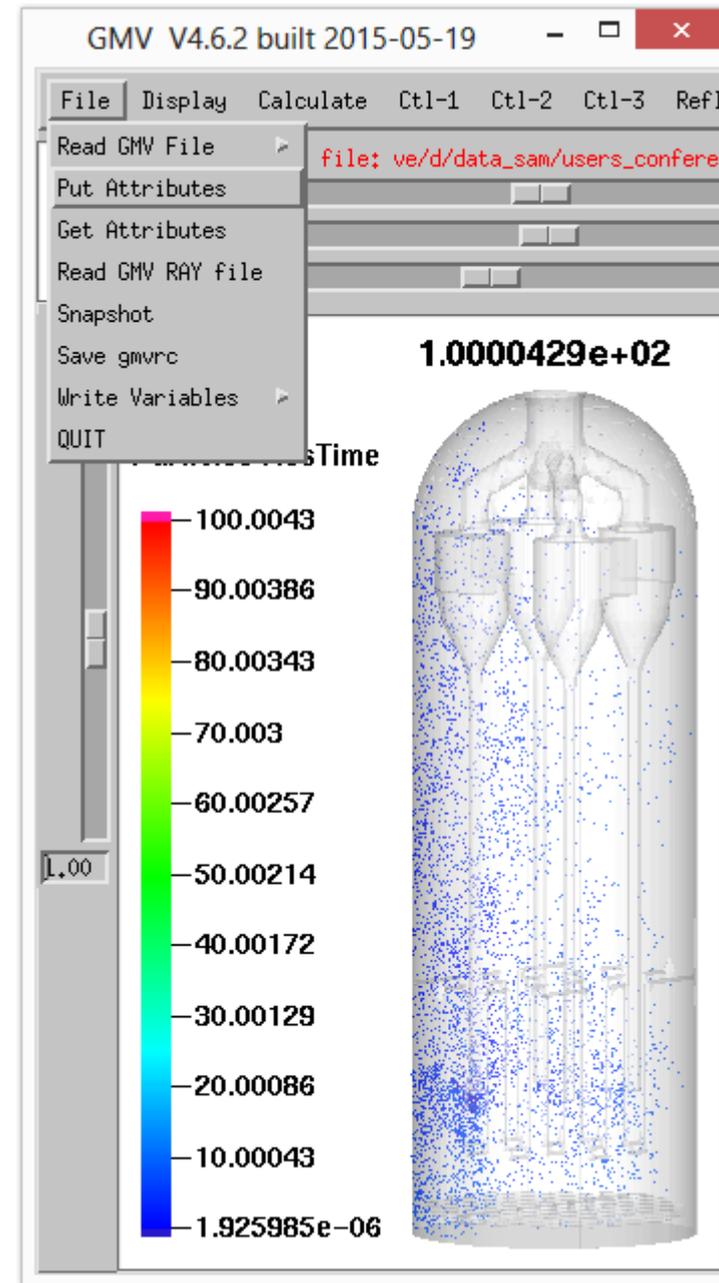
## Goal 4: Creating a Spinning Movie

- Achieving Goal 4: Using BATCHMOVIE.sh
  - Introduction to BATCHMOVIE.sh
  - Creating a spinning movie using the `-azim` flag in BATCHMOVIE.sh



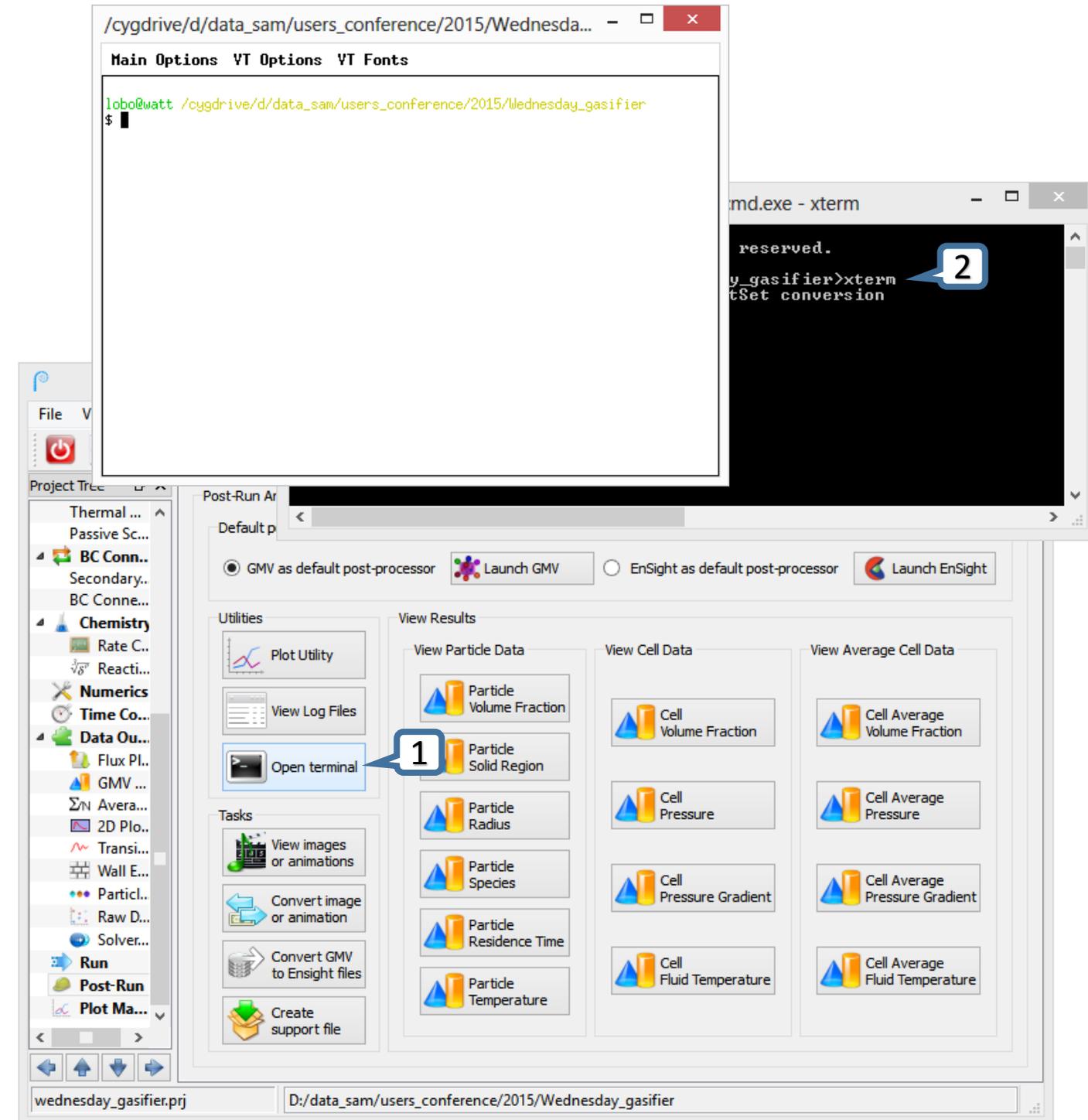
# Save an Attribute File

- BATCHMOVIE.sh creates images for each Gmv\* file by using an **attribute file** describing the view.
- Create the view you want in GMV, then use the menu: File → Put Attributes
- Use a good, descriptive name (with no spaces) for the attribute file.



# Start a Terminal

- BATCHMOVIE.sh is a command-line utility, so we need to open a terminal in the current project's directory to use it.
- In Windows:
  - Post-Run → Open Terminal
  - In cmd.exe enter: xterm
- In Linux:
  - Post-Run → Open Terminal



# Basic BATCHMOVIE.sh Command

- Create a “regular” movie:

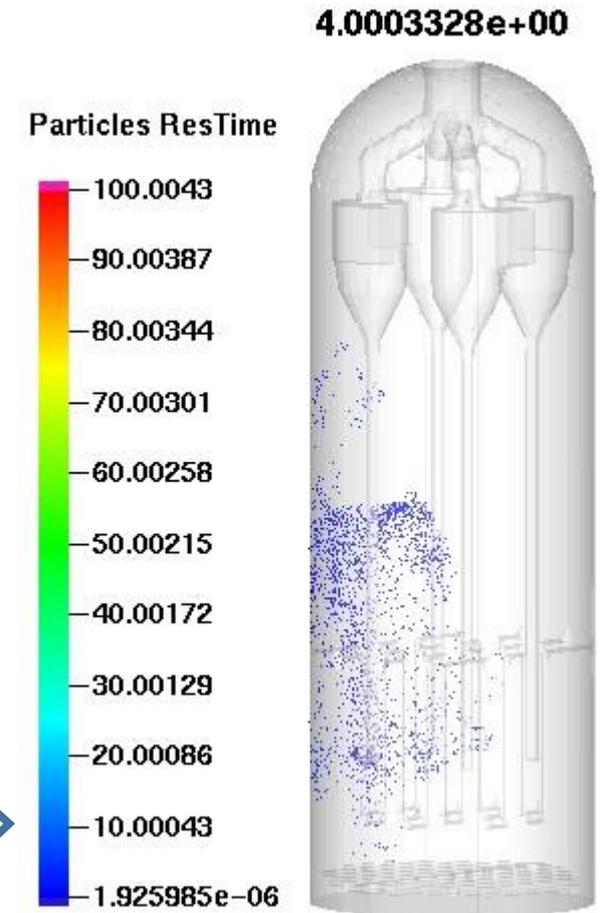
```
BATCHMOVIE.sh particles_species_2_restime_le_10.attr
```

```
lolo@watt /cygdrive/d/data_sam/users_conference/2015/Wednesda...
Main Options VT Options VT Fonts

lolo@watt /cygdrive/d/data_sam/users_conference/2015/Wednesday_gasifier
$ BATCHMOVIE.sh particles_species_2_restime_le_10.attr
Using mencoder
Attribute File: particles_species_2_restime_le_10.attr
Using window size embedded in attribute file (particles_species_2_restime_le_10.attr): 298x477
Image Prefix: particles_species_2_restime_le_10
Target Directory: particles_species_2_restime_le_10
Animation Name: particles_species_2_restime_le_10.mpg
Now creating: particles_species_2_restime_le_10Gmv.00000.jpg
Now creating: particles_species_2_restime_le_10Gmv.00001.jpg
Now creating: particles_species_2_restime_le_10Gmv.00002.jpg
Now creating: particles_species_2_restime_le_10Gmv.00003.jpg
Now creating: particles_species_2_restime_le_10Gmv.00004.jpg
Now creating: particles_species_2_restime_le_10Gmv.00005.jpg
Now creating: particles_species_2_restime_le_10Gmv.00006.jpg
Now creating: particles_species_2_restime_le_10Gmv.00007.jpg
Now creating: particles_species_2_restime_le_10Gmv.00008.jpg
```

BATCHMOVIE.sh will process all Gmv\* files in the current directory, creating images and a final animation based on the specified attribute file.

The resulting movie will match the view defined in the attribute file, including particle selection. Only particles with Species = 2 and ResTime  $\leq 10$  will be shown.



## Using the `-azim` Option to Create a Spinning Movie

- BATCHMOVIE.sh has a `-azim` option to rotate the movie as time progresses. In this example, let's spin the geometry two full turns over the course of the 100 s of simulation time.
  - In order to use the `-azim` option in Windows, the `tclsh` package must be installed in Cygwin. If this package is not already on your system, see [this knowledge base post](#) for details about how to install it.
- Let's also use the `-name` option, to give the resulting images and movie a different prefix than they would have by default.

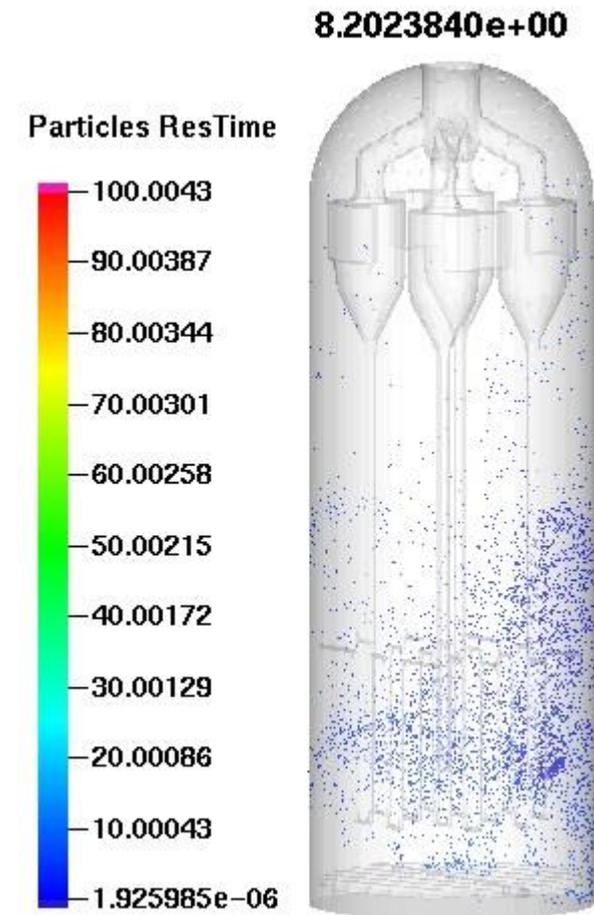
```
BATCHMOVIE.sh particles_species_2_restime_1e_10.attr -azim 0 720  
-name spinning_movie
```

Put this BATCHMOVIE.sh command  
on a single line in the terminal.

# Final Spinning Movie

- BATCHMOVIE.sh will run, creating the images and final movie with spinning geometry.

```
lobo@watt /cygdrive/d/data_sam/users_conference/2015/Wednesday_gasifier
$ BATCHMOVIE.sh particles_species_2_restime_le_10.attr -azim 0 720 -name spinning_movie
Using mencoder
Attribute File: particles_species_2_restime_le_10.attr
Using window size embedded in attribute file (particles_species_2_restime_le_10.attr): 298x477
Image Prefix: spinning_movie
Target Directory: spinning_movie
Animation Name: spinning_movie.mpg
Now creating: spinning_movieGmv.00000.jpg
Now creating: spinning_movieGmv.00001.jpg
Now creating: spinning_movieGmv.00002.jpg
Now creating: spinning_movieGmv.00003.jpg
Now creating: spinning_movieGmv.00004.jpg
Now creating: spinning_movieGmv.00005.jpg
```



# Further Tools for Creating Fancy Movies

- `-mp4` flag for BATCHMOVIE.sh, and standalone script `jpg2mp4`
  - The .mpg files generated by BATCHMOVIE.sh sometimes cannot be played by Windows Media Player or Microsoft PowerPoint
  - You can create a .mp4 file instead, which will usually be able to play successfully on Windows
- MULTIFRAME.tcl
  - Script distributed with Barracuda VR
  - Can combine frames from multiple animations into a single final animation
  - See Barracuda VR Training Materials for details, or [this support site post](#).
- CEI EnSight
  - Alternative post-processor that supports GMV files
  - Modern GUI design with many features
  - Also scriptable through a built-in Python interface
  - Sold and distributed by [CEI](#)