Puff UAF
User's Manual

Version 2.2.0

Authors:
Rorik Peterson (rorik@gi.alaska.edu)

Jul 21, 2006

Table of Contents

Introduction

Puff simulates the transport, dispersion and sedimentation of volcanic ash. It requires horizontal wind field data as a function of height on a regular grid covering the area of interest. Puff output includes the location (in 3 dimensions), size, and age-since-eruption of representative ash particles. It can also produce gridded data of relative and absolute ash concentration in the air and on the ground. Puff is a fast and efficient research and operational tool for predicting the trajectories of ash particles, which is essential for hazard assessment.

Puff is developed on Linux/Unix platforms. It has been successfully run on Solaris and MacOSX (although the required UDUNITS package is not offically supported under OSX). There is currently not a Windows version. Puff is written in C and C++ with an attempt to require few additional libraries.

Puff was originally developed at the University of Alaska, Fairbanks (UAF) by Craig Searcy, based on a model conceived by Dr. H. Tanaka at the University of Tsukuba. Subsequent development then split between the Air Force Weather Agency (AFWA) and the National Weather Service (NWS). The AFWA modifications were initially performed by D.S. Tillman at the John Hopkins Applied Physics Laboratory (JHAPL), and others are continuing the development. Their intent was to develop an operational tool specifically for the needs of AFWA. The development at the NWS is maintained by Craig Searcy. An original copy of Searcy's version before branching has been retained at UAF and is available from http://puff.images.alaska.edu/download. Subsequent development of Puff at UAF has begun to dramatically expand the capabilities of Puff. For this reason, it has been renamed Puff-UAF and a new version numbering system started. Puff-UAF is a completely C++ compliant modification of Searcy's code with some of the bells and whistles from the AFWA version. Most importantly, it now contains several new features, efficiency has been improved, and some library dependencies have been simplified.

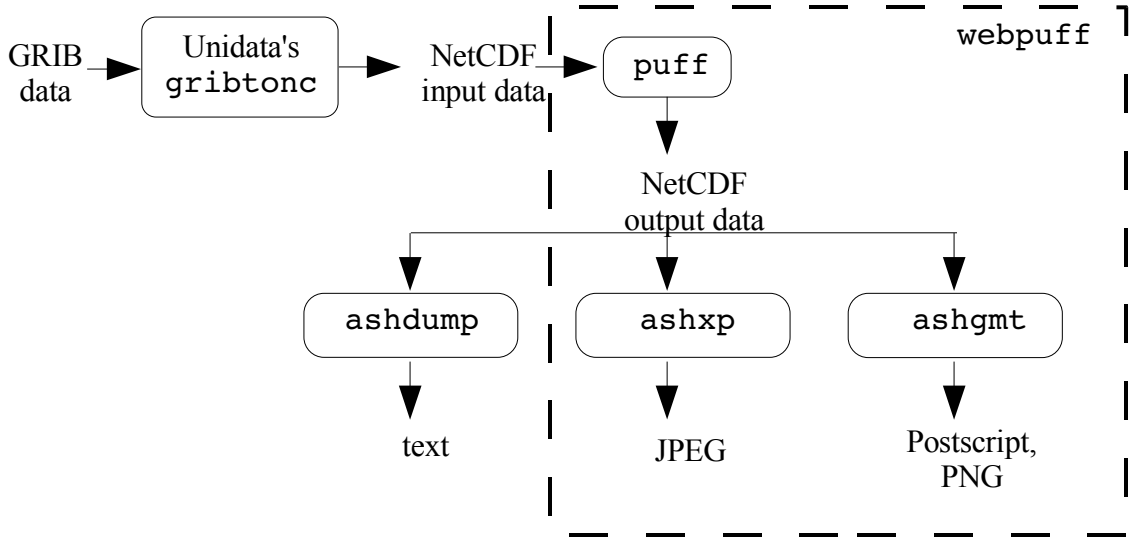
Puff is actually a collection of programs as shown in the figure below. The entire collection is often referred to as Puff suite with a capital 'P'. At the heart of Puff is the main program, which is referred to as simply puff with a lowercase 'p'. You can interact with the main program puff through either the browser-based GUI or directly via the command line.

In addition to specifying eruption parameters, wind field data valid for the time and location must be provided. This data can come in a variety of formats, and Puff has various modules to handle each type of data as shown in the figure. The direct input to puff must be in netCDF format.

The output from puff is also a netCDF file. The post-processors ashxp and ashgmt produce graphical output. The command line utility ashdump dumps column-oriented data to the screen. The browser-based GUI webpuff provides access to puff and direct post-processing of the data via ashxp and ashgmt.

The Puff suite contains the following different programs:

1. **ashdump** Utility for viewing information in the model output files.
2. **ashgmt** Creator of postscript graphics based on the Generic Mapping Tools.
3. **ashxp** Creator of JPEG graphics.
4. **pp2nc** Convert BADC pp files to NetCDF.
5. **puff** Core tracking model.
6. **webpuff** Browser-based GUI.



Name and Version Number

This manual is for the software package Puff-UAF. There are other software programs unrelated to volcanic ash dispersion called Puff. Furthermore, there are a few branches of the original Puff program that have evolved over the past several years. This particular version is developed and maintained at the University of Alaska, Fairbanks (UAF).

The conventional naming scheme for netCDF files is to use the extension .nc. Historically, Puff created netCDF files with the .cdf extension. That extension has been retained for the particle files. The concentration files have the more conventional .nc extension. Using these two different file extensions often makes fileglobbing the different files easier when processing with ashxp or ashgmt.

I recognize that the volcano listing file is misspelled as volcanos.txt. However, that name has been retained for compatibility. You are welcome to change it to volcanoes.txt, or anything else. Be sure to set the environment variable PUFF_VOLCANO_LISTING to this file, however.

Installation

The entire Puff-UAF distribution is divided into a few parts, each of which can be

compiled and installed separately depending on specific site needs. The programs puff, ashdump, ashdump.pl, ashxp, ashgmt, and pp2nc are included in the main Puff-UAF-XX distribution (where XX is the version number).

Binary Distribution

Everything in the binary distribution is contained within a single directory called puff/. Within this are several subdirectories: bin/ data/ etc/ lib/ man/ web/. All the executables are in the bin/ directory, so it is helpful to add that to your PATH environment variable. The data/ directory contains the input wind field data used by Puff. It may currently be empty because the meteorological data files tend to be fairly large. The Puff homepage has current data files you can download and use with Puff at http://puff.images.alaska.edu/data.

Within the etc/ directory is the text file volcanos.txt that lists the name, location, and elevation of many volcanoes. You may edit this file, but keep the fixed-width format the same, as several programs depend on it. The scroll list in the GUI uses this file, so it may be convenient to remove those volcanoes that you are not interested in. Note you can always explicitly specify a volcano location that is not listed in this file. The images/ subdirectory hold the background images for ashxp. You can download any of the GLOBAL files from ftp://ftp.ncgc.noaa.gov/GLOBE_DEM/pictures. The fonts/ subdirectory is a TrueType font specification used by ashxp to label the graphics. The lib/ directory contains the puff libraries and you will likely never need to use them (without shared libraries, they can be deleted). The man/ directory has Unix man pages for ashxp only. You would need to add this directory to you MANPATH environment variable to see them. However, the --help option provides the same information. The web/ subdirectory contains all the files necessary to use the browser-based GUI.

There are a few Perl scripts that need to be edited before Puff will operate correctly. Within the bin/ directory, edit the first few lines of ashgmt to reflect the absolute path location of the executables. (Skip this if you do not plan on using ashgmt). You may find these paths by simply using the Unix 'which' command, i.e.

```
which pscoast
```

If this fails, either you do not have the required program installed, or it is not in your path. See the section Third Party Software for the locations of these other free software packages. In order to use the GUI, edit the top section of web/Webpuff.pm. There is documentation there that describes how to do this.

The binary distribution comes with the UDUNITS library compiled in. However, you if you do not already have the UDUNITS package, you need to get the udunits.dat data file for things to work properly. Furthermore, you need to set the environment variable UDUNITS_PATH to the location of this file before using the command line version of Puff. There is also a setting within Webpuff.pm that specifies the location of this file. You can get a copy of udunits.dat from http://puff.images.alaska.edu/download/.

The binary distribution of ashxp has a default location for a Freetype font file that probably

does not exist on your system. You do not need to put the file in the location it is looking. Set the environment variable ASHXP_FONTFILE to the font file you want to use. A sample TrueType font file that will work should be in the etc/fonts/ directory.

Source Distribution

Building and installation is done using GNU autotools. The most basic build is done by ./configure; make; make check; make install. However, there are several compilation options, which can be viewed using ./configure --help. The most important option is --prefix, which specifies a location when all Puff-related files will be installed. Puff requires Unidata's netCDF C library and header file netcdf.h, and Unidata's UDUNITS library and header file udunits.h to compile. Both packages are available from Unidata at http://www.unidata.ucar.edu.

If these libraries and header files are not in your compiler's search path, specify their location using the environment variables NETCDF_INC and NETCDF_LIB (for example, NETCDF_INC=/usr/local/unidata/include). Often the netCDF and UDUNITS packages are in the same locations, so the above environment variables are sufficient. If not, use the CPPFLAGS and CXXFLAGS variables as described by ./configure --help.

Puff also requires the jpeg library (for ashxp) and the conformal mapping tools from NOAA ARL. The source for these is included within the Puff package and will be built if these libraries are not found. It is optional to use the freeType library for labeling the graphics in ashxp (http://www.freetype.org). You need both the library and the development headers to compile ashxp with freeType support (watch the ouput from configure). If freeType is not available, a low-quality, fixed-size font will be used for labeling.

Puff will utilize a shared library between the two programs 'puff' and 'ashdump'. The benefits of using a shared library with Puff is small, and sometimes more trouble than it is worth since no other programs currently use this library. Use the --disable-shared option to configure to turn off shared libraries.

The are a few supporting programs within Puff-UAF that are actually Perl scripts: ashdump.pl, rereference,pl and ashgmt. They are constructed from the template files <file>.in during the execution of './configure'. The primary purpose is to substitute in the absolute path names of several executable files such as 'perl', 'gs' (ghostscript), etc. If the errors “command not found” or “bad interpreter” occur when attempting to run these programs, check that the first line of each script correctly identifies the location of the Perl interpreter.

The GUI interface for Puff is a browser-based set of CGI scripts written in Perl. This interface replaces the previous pufftk interface that utilized PerlTk. The scripts for this interface reside in the web/ directory and are also constructed during configure from a set of <file>.in templates. The Perl module file Webpuff.pm may be need to be modified depending on a particular installation. There is documentation for these modifications within the top few lines of the script. This browser interface can be run locally on a single

machine, within a local user group, or even a public http server. The default behavior is to run on a single machine locally using the 'minihttp' program included with Puff. To start the web interface, use the 'webpuff' script within the web/ directory, which will then start a local httpd service on 'localhost:1234'. Click the 'logout' button on the interface to turn it off. The browser is convenient to use, but lacks some of the versatility and functionality of the command-line based program.

Getting Data

In order to use Puff, you need gridded meteorological data in netCDF format. There are sample files on the project website for testing and doing the tutorial. These testing data files have significantly reduced resolution from the original in order to reduce their size for easy downloading. Depending on your needs, you will have to determine what data you need and how to get it. There are various methods dicussed below. The default location for data files is in the data/ directory within Puff, usually with a different subdirectory for each type of model. However, the data can be located anywhere; just set the correct path in the puffrc resource file.

One option for obtaining current and recent data is directly from the project website data directory at http://puff.images.alaska.edu/data/. Here there are several subdirectories, one for each different type of meteorological model. There is usually a seven-day archive of netCDF files, and a one-day archive of GRIB files. You only need the netCDF files to run Puff. The netCDF files are generated from the GRIB files using the template files with the .cdl extension. If you have Unidata's gribtonc, you can download the GRIB file instead and convert it to netCDF yourself (see Converting GRIB files). GRIB files are compressed and much smaller in size than the netCDF files and will download more quickly.

Another option for obtaining data is directly from a meteorological center such the NOAA's National Weather Service ftp site at ftp://tgftp.nws.noaa.gov/SL.008001/. These files are usually in GRIB format (either version 1 or 2) and will need to be converted to netCDF in order to use with Puff. This site is the most current source of forecast data, but does not maintain much of an archive. There are several other meteorological centers in the world that provide both current and archived weather models that can be used with Puff.

Tests

A set of tests are available in the test/ directory of Puff-UAF, and can be executed using "make check". Each one can also be run manually. These also serve as a set of example demonstrating many of the features of Puff. There is some documentation at the top of each script. The tests are constantly changing and evolving. The basic idea is to execute puff and the other programs in a variety of different ways that will verify that things appear to running appropriately It is safest to only install the programs after all tests have succeeded. During 'make check', if a test succeeds the message PASS: testXX.sh is written to standard output. When a test fails, FAIL:testXX.sh appears and an error log called testXX.err remains in the test/ directory. This log may provide enough information for you to fix the problem. If not, email the error log, along with the program version (puff --version), to the puff maintainer email on the project homepage http://puff.images

alaska.edu.

In order to run these tests, wind field data files that may not be included with the distribution are required. These files should be in the data/ directory. If they are not already available, they will be automatically downloaded from the web site http://puff.images.alaska.edu/download using wget when running 'make check'. If you do not have wget, you will need to retrieve the files manually before running the tests from the project homepage at http://puff.images.alaska.edu/download. A message directing you how to do this will appear when 'make check' fails.

Third-party software

In order to broaden the community that can utilize Puff, third-party software requirements have been minimized. However, there is always a trade-off between expanding the breadth of functionality while minimizing the use of other pre-written software. Fortunately, the modularity of Puff allows some parts to be used even if others are missing. There are two post-processors for creating graphical images of Puff data output: ashxp and ashgmt. The former only requires the JPEG library included with Puff. The latter requires the Generic Mapping Tools (GMT), available for free from http://gmt.soest.hawaii.edu, and the NetCDF Operators (NCO), also available for free from http://nco.sf.net. Ashgmt generates postscript files that are nice for publications. They are automatically converted to PNG graphics if Ghostscript's 'gs' application is found. Also, GIF images are created using ImageMagick 'convert', available from http://www.imagemagick.org. The GIF format is convenient for making animated graphics using 'gifsicle', which is available for free from http://www.lcdf.org/gifsicle. Ashgmt will function without any of 'gs', 'convert', or 'gifsicle', however, some image formats will not be possible; PNG, GIF, and animated GIF, respectively.

Tutorial

You can go through the tutorial sections without first setting up a Puff resource file. However, it might be easier if this is done before hand; see the section on Using the puffrc file. It is also easier if you put the bin/ directory in your path. In any case, make sure you have the required data files in the data/ directory.

Quick Start

In its most simple operation, Puff only needs the location and beginning time of a volcanic eruption. Wind field data that covers this time period is also required. Go into the Puff installation directory. Check that there is wind field data for July 25th, 2006 in the data/gfs/ directory, the name should be 2006072506_gfs.nc. There should be two files: gfs/2006072506_gfs.nc and nam/2006071400_nam216.nc. These files are retrieved using wget when make check is executed and put into the data directory. Optionally, they can be downloaded manually from http://puff.images.alaska.edu/download. Now run puff

```
bin/puff -volc spurr -eruptDate "2006 07 25 06:00" -model gfs
Begin:  Wed Jul 26 08:53:40 2006

Reading u from /home/rorik/puff/data/gfs/2006072506_gfs.nc ... done.
Reading v from /home/rorik/puff/data/gfs/2006072506_gfs.nc ... done.
Converting levels using standard-atmosphere approximation ... done.
Making vertical wind ... done.
Volcano:        spurr ( 207.75 , 61.3 )

Start Time:     2006 07 25 06:00 GMT    (1153807200)
  End Time:     2006 07 26 06:00 GMT    (1153893600)
                ____________________
   RUNNING:
Saving ./200607251200_ash.cdf
Saving ./200607251800_ash.cdf
Saving ./200607260000_ash.cdf
Saving ./200607260600_ash.cdf

Done.
```

If the above happens, all went well and skip to the next section. If not, there are a few common problems.

```
WARNING: A volcano listing has not been specified and name
lookup will not be possible.  Either set the environment
variable PUFF_VOLCANO_LIST or use the command-line option
"-volcFile".
```

Puff needs to know where the volcano 'spurr' is by using the volcano listing file. This file should be in the etc/ directory. Either set the environment variable with an absolute path, or use the --volcFile option.

```
No data resource file found
```

This means Puff could not find the resource file that tells it where the data is stored. There is a default resource file puffrc stored in the etc/ directory. Open it with a text editor (create it if necessary) and add the following two lines (what is shown below might wrap around, but each line begins with model=)

```
model=gfs mask=YYYYMMDDHH_gfs.nc var=u,v path=$PUFFHOME/data/gfs/
model=nam216 mask=YYYYMMDDHH_nam216.nc var=u,v,z,T
path=$PUFFHOME/data/nam
```

Now try it again, specifying the resource file to use.

```
bin/puff -volc spurr -eruptDate "2006 07 25 06:00" -model gfs --rcfile
etc/puffrc
```

Another problem you might encounter is

```
ERROR: PUFFHOME is not defined but appears in rcfile
```

This means you have not defined the environment variable that appears in the puffrc resource file. Either define and export it, or change the path in the resource file to the absolute location of the data files.

Another common error encounter is something like this:

```
WARNING: no "u" data available in /home/rorik/puff/data/gfs prior or
equal to 2006 09 25 06:00
```

Look carefully and you see the eruption date must have been specified incorrectly (month is 9, or September, instead of 7 or July).

If you are still having problems, see the troubleshooting section.

Puff generates data in NetCDF format. You can dump the data using the ashdump utility

```
bin/ashdump 200607251200_ash.cdf | more
```

It is a good idea to pipe the output to a pager since thousands of lines of text will scroll by otherwise. The first few lines give the name of the data file, location and time of the eruption, timestamp for the data file, and the number of particles in the data file. Next is a listing in column format of the location and attributes of each ash particle.

Perhaps you want to write this data to a text file called ash.dat to be read by anther application. You don't want the header information and are only interested in the latitude and longitude values.

```
bin/ashdump -lat -lon 200607251200_ash.cdf > ash.dat
```

Alternatively, use ashxp to visualize the data. To generate a single JPEG image of the data at 06:00

```
bin/ashxp 200607251200_ash.cdf
```

This image is not that useful since it shows a small region only large enough to show all the ash. A larger region with gridlines on a polar stereographic projection can be generated with

```
bin/ashxp -P polar --gridlines 10 --latlon 50/70/190/230
200607251200_ash.cdf
```

This will generate a file 200607251200_ash.jpg. If you process all ash files at the same time, the bounding box of each image will be the same. Otherwise, the bounding box is adjusted to contain all ash particles (rounded to every 16 pixels) with a minimum size of 256 x 256 pixels. Now create a sequence of JPEG images with a bounding box covering Alaska and showing only ash between the ground and 2km up using 5 pixel particles.

```
../bin/ashxp --hgt-range 0/2000 --pixel 5 2006*ash.cdf
```

You might encounter the error

```
No background file specified.  Either use the --bgfile option or set
the environment variable ASHXP_BG_FILE_NAME
```

The background image can be specified as an option. First, be sure you have a background image in the etc/images/ directory. Any of the GLOBAL images from ftp://ftp.ngdc.noaa.gov/GLOBAL_DEM/pictures will work. Put them in the etc/images directory.

```
bin/ashxp -b etc/images/GLOBALeb3colshade.jpg 2006*ash.cdf
```

GUI

To use the browser-based GUI, go into the web/ directory and run webpuff. You should see

```
starting webpuff... point your browser to http://127.0.0.1:1234/login.pl
```

Type the specified URL into a web browser such as Firefox. You should see a login screen with two buttons. Make up a session name like 'spurr_test', type it into the text window, and press 'Start a New Session'. The next screen should show 3 frames. The top frame runs the puff program. The lower-left frame (blank for now) controls the graphics post processor, and the lower right frame displays the graphics (also blank for now). Select a volcano from the scroll list on the upper left. Press 'Run Puff'. The top frame should change to a logging screen that displays the output you would see on the command line. When finished, press the 'OK' button to return to the main screen. A new scroll menu should appear in the upper right showing the name of the volcano. Click on in and a new window will appear in the lower-left frame. Press 'Map'. After a few moments, a graphical image should appear in lower-right frame. Scroll through the 4 graphical images generated using the scroll menu on the right-hand side. When finished, press 'logout' to stop the 'webpuff' program.

Multiple Eruptions

Puff can simulate multiple eruptions from either the same volcano or a different volcano. There is no pre-set upper limit to the number of eruptions, but memory restrictions obviously apply. Perhaps you are tracking activity on Mt. Etna, which has an initial small eruptive event, is docile for several hours, and then erupts again more violently. In order to simulate this event, you begin Puff with the initial eruption, and then "add" the second eruption when it happens using the -restartFile option. For example, first simulate the initial eruption

```
puff -volc Etna -eruptDate "2006 07 25 07:00" -eruptHours 1 -runHours 5
-saveHours 1 -nAsh 1000
```

Then add the second eruption while loading in the data from the initial eruption in order to

continue tracking it.

```
puff -volc Etna -eruptDate "2006 07 25 12:00" -eruptHours 1.5
-restartFile "200607251200_ash.cdf" -nAsh 5000
```

By specifying a different number of particles for each eruption, different magnitude eruptions can be simulated. If the restart file is not timestamped the same as the next eruption time (--eruptDate, a warning is issued but the simulation continues and the particles from the restart file are 'transposed' in time to the --eruptDate. This might occur if the first simulation did not output files exactly at the time the second eruption began.

The restart capability can be used to simply extend and existing simulation without creating a second event. In this case, only --restartFile and --eruptDate would be required.

Non-point Source Initiation

You can use Puff to simulate the movement of an existing ash cloud. Perhaps you have remote sensing data that indicates there is an existing ash cloud east of Mt. Reventador, Ecuador. The cloud is rectangular with higher elevation ash near the southern end. You can use that information to initialize Puff using the --restartFile option. The restart file contains lat/lon pairs that define a polygon outlining the ash cloud, interior points giving approximate elevation of the ash, and a size distribution. The format of the restart file is

```
# comments have a 'pound' sign before them
# all field have a XML-like structure, and can occur in any order
<verticies>
283.0 -0.0 # lon lat in decimal degrees east
283.5 -0.0
283.5 -1.0
283.0 -1.0
</verticies>
# elevation points are lon lat elevation (meters)
<elevations>
283.3 -0.1 10000
283.3 -0.3 12000
283.3 -0.5 14000
283.3 -0.7 16000
<</elevations>
#size is optional, but is -ashLogMean and -ashLogSdev if specified
<size>
-6 1
</size>
```

The # symbol indicates anything after it on the line is a comment. The size distribution information is optional. No line segments in the defined polygon may intersect, so a figure-8 is an invalid polygonal shape. The number of particles are weighted proportional to the elevation points. To create an area with a greater density of particles, add more elevation points to that area. For example, the following would create a denser particle distribution in the northern part of the cloud in the above example

```
<verticies>
```

```
283.0 -0.0 # verticies are the same
283.5 -0.0
283.5 -1.0
283.0 -1.0
</verticies>
<elevations>
283.3 -0.1 10000
283.3 -0.3 12000
283.3 -0.5 14000
283.3 -0.7 16000
</elevations>
<size>
-6 1
</size>
# more points can be added to the above field, or a new one created.
# repeating elevation points adds more weight to that area of the cloud
<elevations>
283.3 -0.1 10000
283.3 -0.1 10000
283.3 -0.3 12000
</elevations>
```

Multiple Runs

Puff uses Gaussian random motion (simulated Brownian motion) to simulate turbulent diffusion of ash particles. For this reason, the results of each simulation are unique. To obtain an "average" result, several simulations should be run with the same input parameters. In order to do this without simply running Puff over and over, use the --repeat option. For example, to get an ensemble of 10 6-hour results of an Etna eruption, each with unique filenames

```
puff -volc etna -eruptDate "2006 07 25 06:00" -runHours 6 -saveHours 6
-eruptHours 1 -repeat 9
```

Note that a repeat value of 9 produces 10 simulations. The result files will have the name 200607251200_ashxxx.cdf where xxx is a three digit number starting at 000. An average concentration file can be generated using --gridOutput combined with --averageOutput. Without the second option, a separate concentration file will be written for each repeat run. Note that each repeat run will be slightly different due to the random number generator. This can be eliminated by using the --seed option, although the utility of that is not clear.

The repeat option is most useful when creating gridded output with the --gridOutput option. Averaging more runs creates a smoother looking cloud when plotting gridded data. The default grid size of 0.5-degrees in the lat/lon directions, and 2000 meters in the vertical. This can be modified by providing a value to the gridOutput option, i.e.

```
puff --gridOutput=0.25x500
```

which would set the lat/lon grid size to ¼ degree, and 500 meters in the vertical direction. The bounds of the gridded file are adjusted to be just large enough to contain all the ash.

The bounds can be forced to an area larger or smaller than the default.

```
puff --gridBox 175/195/10/300/25000
```

which sets the bounds to 10-30 degrees latitude, 175-195 degrees longitude, and 0/25000 meters in the vertical. The gridded file has a .nc extension and includes the iteration number when used with the repeat option. The date and time in the filename are those of the eruption start time and the file contains all times specified via --saveHours. For example,

```
puff --gridOutput --eruptDate "2006 07 25 06:00" --repeat 10
--averageOutput
```

creates a file 200607250600_conc010.nc.

Particle and Concentration Files

The default output for puff is called a particle file, which is a netCDF files containing a record of all tracer particles. Each particle has the following attributes: latitude, longitude, elevation, age, size, airborne/grounded flag, and an exists flag. You can use ashdump to list these attributes in column format. A separate particle file is generated for every output time. The format of these files is YYYYMMDDHHmm_ashXXX.cdf, where Y is year, M is month, D is day, H is hour, and m is minute. The string XXX is optional and only occurs when the --repeat option is used. In that case, the XXX string will be replaced with the iteration number starting with 000. Generation of particle files can be supressed by setting --ashOutput=false.

An optional concentration file can be created using the --gridOutput option. These files are created by adding up all the tracer particles within a grid box and computing a relative and absolute concentration value. The size of the grid box can be adjusted using --gridOutput=HxV where 'H' is in the horizontal size in degrees and 'V' is the vertical size in meters. The absolute concentrations are calculated based on the eruption mass or eruption volume. Either one can be specified using --eruptMass and --eruptVolume, respectively. They are not independent because a constant density of 2000 kg/m^3 is assumed. Only a single concentration file is generated for each simulation. Several can result if the --repeat option is used without the --averageOutput option. Concentation files are 4-dimensional, x,y,z and time. Concentration files are generally NetCDF complient and can be directly read into many GIS type software packages. The format of concentration files is YYYYMMDDHHmm_concXXX.nc, with the substitutions described above. However, the time/date now corresponds to the beginning of the eruption since data for all subsequent times is included within the file.

The default relative concentration is simply the number count of tracer particles at a grid location. A summation of all relative concentration values is necessarily equal to the --nAsh value. Alternatively, the number of levels in relative concentration can be set on a log scale using --gridLevels. For example, if --gridLevels=3 and the maximum number count of particles at any location is 20, a grid point with 2.7 particles has a relative concentration of 1, one with 7.4 particles has a relative concentration of 2, and one with 20

has a relative concentration of 3.

Ash Exposure

One of the major concerns of airborne ash is the potential damage it can do to an aircraft that flies through it. Because aircraft can travel hundreds of kilometers per hour along varying trajectories as an ash cloud moves and disperses, calculation of potential exposure is not straight forward. Puff can be used to calculate the potential exposure to ash an aircraft might encounter by summing up the local ash concentration at each point along a plane's trajectory during the flight. This is accomplished using the --planesFile option combined with --gridOutput. It is also desirable to use --phiDist in this case to remove the occasional large ballistic particle that tends to skew the results.

Aircraft trajectories are specified in a text files in a comma-delimited format such as shown below.

```
FAK101,18/FEB/01,1854,SEA,RJAA,510,340,5640N,14613W,B772
```

The 10 required fields are the following:

1. flight number
2. date
3. time in UTC
4. origin code
5. destination code
6. speed in knots
7. elevation in 100's of feet
8. latitude in DDmm format
9. longitude in DDDmm format
10. plane type

You can manually generate this file, or use the plane_traj.pl script included with puff. The minimum resolution is 1 minute and linear interpolation is used between times. It is advisable to reduce the puff time step to 1 minute or less for more accurate results, i.e. --dtMins=1. Several different flights can be included in the same file, and the --planesFile option accepts Unix 'fileglobs' (i.e. --planesFiles="flight*.dat") Be sure to quote the fileglob to prevent the shell from trying to interpret it. All similar flight numbers on the same day are combined into a single flight when computing exposure. Calculation of exposure is done during the simulation and the results are output to stdout like the following:

```
Calculating plane exposure ...
12.83653 => FAK101-18/FEB/01
```

The units are g/m^2 based on cross section of the target, for example, the engine cross section. A separate line is printed for each separate flight. The exposure to airborne ash a city or town might encounter can be determined by creating a flight with constant latitude and longitude, and an altitude of zero (or whatever the local elevation is). In that case, a two-line file with times corresponding to the beginning and end of the puff simulation are

sufficient. In that case, be sure the times listed in the file are within the simulation bounds; records for times outside are discarded.

Converting BADC pp files

The British Atmospheric Data Center (BADC) produces global gridded windfields that include stratospheric winds (1000-0.1 millibars). Data is available from their website at http://badc.nerc.ac.uk in a binary format with the suffix pp (registration required). The Puff distribution includes a small utility called pp2nc that converts these files into netCDF files that can subsequently be used by puff. To create an initial archive of data in netCDF format, use

```
pp2nc ppassm_operh_y03_m08_d01_h12.pp
```

for example. This command will create the netCDF file with the mask YYYYMMDDHH_badc.nc, or 2003080112_badc.nc in this example.To protect against inadvertent errors, pp2nc will not clobber existing netCDF files, so (re)move them if you desire to create new ones with the same name. To add further data to the archive, use the 'add' command followed by the netCDF archive name, and subsequently a listing of the pp files to add.

```
pp2nc add 2003080112_badc.nc ppassm_operh_y03_m08_d02_h12.pp [...]
```

If you already have a collection of pp files, command-line wildcard expansion may be useful.

```
pp2nc add 2003080112_badc.nc ppassm_operh_y03_m08_*pp
```

Alternatively, if may be useful to create a script that downloads new pp data each day from the badc website and adds it to the archive. Then set this script up to run using cron.

When creating the netCDF archives, remember that Puff only reads in a single data file for each simulation, so include all the data you wish to use in the file. See See Technical Details, for further details on time interpolation and extrapolation.

Converting GRIB files

The recommended way to convert GRIB files into netCDF, which Puff can understand, is to use gribtonc from the Undata decoders. These decoders are fairly robust, very well maintained, and have a relatively large support group. See http://unidata.ucar.edu/packages/decoders.

Some binary distributions of puff come with gribtonc. To use it, first create a template file from a sample GRIB file

```
gribtocdl fh203.00.grib > template.cdl
```

Then use this template to create a NetCDF file that puff can use.

```
gribtonc template.cdl 2006021200_gfs.nc < fh203.00.grib
```

Some data sets do not explicitly contain the x,y or lat,lon dimension values in the NetCDF file (i.e. as a dimension and a variable). This often happens when these values can be determined by the projection and grid size information. However, Puff requires these values to appear as a variable within the data file. You may need to add the x,y or lat,lon values by editing the cdl template file before producing the NetCDF file. This only has to be done once. Be sure to define it as a variable in the 'variables' section and add the actual values in the 'data' section.

If not using gribtonc for some reason, conversion of GRIB files into something Puff understands can be done using grib2pf (grib to puff-friendly). This program is a Perl script, and can be easily modified and customized for different needs. It currently understands the following GRIB grid numbers

- 3: 1-degree x 1-degree global
- 207: 95-km regional Alaska, polar stereographic
- 216: 45-km regional Alaska, polar stereographic
- 240: 1-degree x 1-degree global (nogaps)
- 249: 12-km regional Alaska, polar stereographic (experimental)

To convert a time-series of forecast files downloaded from NCEP for Grid 216 over Alaska

```
grib2pf -t 6 fh.0000_tl.press_gr.awipak fh.0006_tl.press_gr.awipak
```

The -t HH option specified the time differential between the forecast files in hours. It is your responsibility to specify the files in increasing order of forecast time, which is easily done using most UNIX shell expansion utilities. The u and v components of wind are extracted from the input GRIB file using Wesley Ebisuzaki's wgrib and written to a netCDF data file using ncgen. The grid number can be explicitly given using the -g NUM option, however, if it is not one that grib2pf understands, the files cannot be processed. The default output file is based on the valid time of the forecast data, and will be of the form YYYYMMDD_gridXXX.cdf, where XXX is the grid number. Using the -o FILENAME will override this file name. Creation of the netCDF file can be suppressed using -n, and a text file in the netCDF cdl format will remain. Silence is obtained using -s, and verbose output by -v.

To modify grib2pf for a new grid, editing the script is required but quite easy. There are two subroutines at the end of the script: header and footer. Create a new entry in each subroutine for the new grid number. Many fields will remain the same, however, changing the number of grid points in the new grid, and defining the latitude and longitude of the corners must be adjusted as necessary. GRIB files do not contain the dimension values such as latitude, longitude and elevation, and these values must be added to the appropriate location in the footer subroutine.

This script is not really maintained anymore because gribtonc is far superior.

Using the puffrc file

Beginning with Puff version 2.1.0, the plain text file puffrc is used. This file contains information about data files which simplifies the command-line syntax and places the burden of determining the most-recent data on the program. This also makes it very easy to change the meteorological model used by simply changing the --model argument. The search protocol for finding this resource file is to look first in the user's home directory for a file called .puffrc (note the preceding period), then the current directory, and finally $PUFFHOME/etc/puffrc. The home directory is determined by using the environment variable $HOME. If the file is located in either the home or current directory, the name should be pre-pended with a period, i.e. .puffrc. This search protocol may be overridden using the --rcfile option. Failure to locate a resource file results in program termination.

Below is a sample resource file that demonstates many different models and data locations.

```
model=avn mask=YYYYMMDDHH_avn-U.cdf var=u path=/var/data/avn/
model=avn mask=YYYYMMDDHH_avn-V.cdf var=v
model=avn mask=YYYYMMDDHH_avn-\H.cdf var=z
model=gfs mask=YYYYMMDDHH_gfs.nc var=u,v path=$HOME/puff/data/gfs
model=nam216 var=u,v,z,T mask=YYYYMMDDHH_nam216.nc \
path=$PUFFHOME/data/nam
model=reanalysis mask=YYYY.uwnd.nc var=u varUname=uwnd \
path=$HOME/data/reanalysis
model=reanalysis mask=YYYY.vwnd.nc var=u varVname=vwnd \
path=$HOME/data/reanalysis
dem=gtopo30 path=/home/rorik/data/GTOPO30
```

Each line in the file contains a directive. Each directive must end with a carriage return. Lines may be broken by placing a '\' as the final character (no spaces afterwards). In the above example, there are two different types of directives: model and dem. The model directives are used to indicate different model sets that can be used. The directive indicates where meteorological data is located (path), what the file naming scheme is (mask), the type of data located in that file (var), the variable name within the file (i.e. varUname) and a nickname for that data set (model). The default variable names (within the netCDF file) are u,v,T,Z for horizontal wind, vertical wind, temperature, and geopotential, respectively. The names are case-sensitive, so use varUname, varVname, varTname and varZname when necessary.

The above resource file demonstrates how Puff determines the method that different data models are stored. The avn data is broken between three different files. The gfs model only has u,v data (Z and T will be computed from a standard atmosphere approximation) and uses the $HOME environment variable. The nam216 directive makes use of the $PUFFHOME environment variable. Reanalysis data is named uwnd and vwnd, different than the default 'u' and 'v' names. By specifying all this in the resource file, you only need to change the --model= option of the command line to switch between models when running Puff.

The dem directive indicates where the data for each type of DEM is located (path) and a nickname for it (dem). There is no limit to the number of directives that may be in the file,

and incorrect directives (i.e. file locations that do not exist) are not checked for validity unless that model is being used. The first directive on the line must be either model= or dem=, however, the following parameter/value pairs can occur in any order (i.e. mask, path, var, etc).

To indicate the data set for a simulation, use the --model option with the corresponding nickname. Puff will then read the resource file to determine the file naming protocol for that model type, and then look in the appropriate location for the most recent data preceding the eruption start time. As an example, assume the following simulation using the example resource file above.

```
puff -model=gfs -eruptDate "2006 05 10 19:30" -volc Reventador
```

Puff will look in the directory $HOME/puff/gfs for data files. The naming protocol will use the mask YYYYMMDDHH_gfs.nc, where YYYY is a 4-digit year, MM is a 2-digit month, DD is the two-digit day, and HH is the 2-digit hour. The components of wind may be located in the same or different files, and the var= part of the directive is used to specify the data within that file. For the gfs data, 'u' and 'v' components are in the same file and there is no T or Z data. Assuming data arrives in every 12 hours, Puff would find all wind components in the file 2006051012_gfs.nc However, if data arrived every 6 hours, Puff would find the file 2006051018_gfs.nc.

For the avn data, each component is in a separate file. Note that the avn geopotential height data files (var=z) have a literal H in the name. In order to differentiate literals from year, month, day, and hour specifications, they are escaped using the \ character.

If the vertical coordinate is in millibars, the geopotential height Z will be used to convert these levels to elevation. However, if the letter z had not appeared in the var specification (i.e. var=u,v), Puff would not know that there was geopotential data in the file. Since no other directive exists for model gfs that says where geopotential data is, Puff assumes that data is not available and uses a standard atmosphere approximation. The temperature field is not required for constant Stokes sedimentation; see the --sedimentation option for other sedimentation physics.

The environment variables $HOME and $PUFFHOME may be used, but no other environment variables will be interpreted.

Options For puff

A complete listing of all options available to each program is available by using the --help option. All options for puff and ashdump are long names and either a single or double dash may be used (i.e. -eruptDate and --eruptDate are equivalent) This is not true for ashxp, which uses single dashes for short options that can be concatenated, and double dashes for long options. To see the help menu in ashxp, use -h or --help, but not -help which is equivalent to -h -e -l -p). To keep things confusing, ashgmt only acceps long options but only enough characters to be unambiguous are required. In ashgmt, -h results in

```
Option h is ambiguous (help, hgt-range)
```

The different options puff accepts fall generally into three different categories: description of the original ash plume, control of program input and output, and customization of the simulation. Boolean options are turned on when given without an argument, and true, True, false and False are acceptable arguments. Some options have an optional argument, such as --gridOutput. In that case, an '=' sign is required when passing an argument, i.e. --gridOutput=0.5x2000 but not --gridOutput 0.5x2000. It is always safe to use the equal sign with all options.

Initial Ash Cloud Description

There are several different characteristics of the initial ash cloud that can be customized. If the option is not given, the default value is used. The only required argument is either --volc or --latLon. and the beginning of the eruption --eruptDate. (The combined options -volcLat and --volcLon accomplish the same thing.) If a restart file is used with the --restartFile option, the volcano name/location specification is optional. The following table lists all the options for describing the initial ash cloud.

option	**Default**	**Description**
ashLogMean	-5	base-10 log of the mean particle size in meters
ashLogSdev	1	standard deviation in the particle size distribution
lonLat	none	longitude/latitude of the volcano
nAsh	2000	number of ash particles
noFallout	FALSE	do not let particles fall gravitationally
plumeMax	16000	maximum plume height in meters
plumeMin	0	minimum plume height in meters
plumeHwidth	0	initial horizontal spread
plumeZwidth	3	initial vertical spread
plumeShape	linear	vertical distribution of particles
volc	none	name of the volcano, without volcLon & volcLat
volcLon	none	longitude of the volcano, over rides volcanos.txt file
volcLat	none	latitude of the volcano, over rides volcanos.txt file

A distribution of ash-particle sizes is created at initiation. There are two different ways to specify the particle distribution. One options is a gaussian distribution with an average particle radius and size distribution that is controlled through the options --ashLogMean and --ashLogSdev. For example, -ashLogMean=-5 -ashLogSdev=2 will result in a distribution where the average particle radius is 10^{-5} meters = 10 microns. One standard deviation of the particle number (~73.5%) will occur between 10^{-3} and 10^{-7} meters, or 1 mm to 0.1 microns.

Alternatively, a custom size distribution can be specified using a phi distribution with the --phiDist option. The phi distribution is a lognormal distribution

$$\phi = -\log_2 d$$

where d is the particle diameter in millimeters. The --phiDist option takes a string specifying value pairs of phi and the percentage of particles by mass that are in that bin. An example would be

```
puff --phiDist="1=25.5;2=50;3=24.5"
```

which means that 25.5% of the particles (by mass) have a diameter of 2^{-1} millimeters, 50% have a diameter of 2^{-2} millimeters, and 24.5% have a diameter of 2^{-3} millimeters.. If the percentages do not sum to 100, they are normalized. The quotes are necessary to protect the semi-colons from the shell. Because mass is proportional to the diameter cubed, phi distributions can cause the number distribution to be skewed toward the smaller particles. For example, --phiDist "3=50;6=50" means that only about 1 in 500 particles are the larger size. This makes mapping ash fall deposits tricky because most of the particles appear to blow away and only a few deposit on the ground. From a number density point of view, that is probably true. However, it might be helpful to use a mono-sized particle distribution when mapping ash fall deposits, and use the full phi distribution when analyzing the relative fall out amounts.

The particle size effects the settling velocity. The default settling law is a constant Stokes law (i.e. --sedimentation=constant)

$$V = \frac{2}{9} r^2 \Delta \rho \mu^{-1} = r^2 G$$

where G = 1.08 x 10^9 is a constant. The option --noFallout forces the constant G = 0, so vertical motion is only caused by diffusion and wind velocity. The effect of pressure and temperature on the particle/air density difference can be accounted for by setting --sedimentation=stokes. A standard temperature atmosphere is assumed if no temperature data is provided.

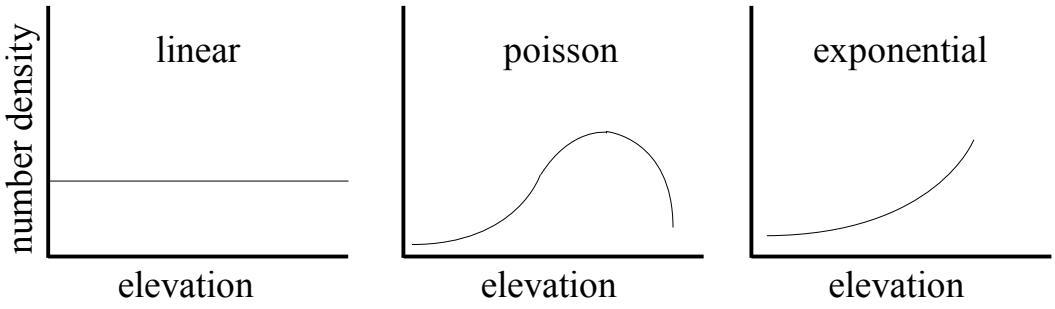
Variable fall dynamics based on the Reynolds number are possible by specifying --sedimentation=reynolds. In this case, the fall velocity is calculated for each particle at each time step based on its size and environment. There are three flow regimes possible with this option: turbulent, transitional, and laminar (Stokes). This option should only be used for high-altitude plumes, large particle sizes, or when very detailed fall information is

desired due to the extra run-time required.

When using variable fall dynamics, it is recommended that a gridded temperature field be given in addition to the wind fields. To do this, the variable T is put into the .puffrc resource file (i.e. var=T,u,v). If there is no temperature data, a 'standard atmosphere' is assumed and T should not be added to the var= directive.

Particle Distributions

The --plumeMax and --plumeMin options control the minimum and maximum vertical extent of the initial cloud. The --plumeZwidth option has a different meaning based on the vertical ash distribution selected using the --plumeShape option. The plume shape actually controls the number distribution of particles in the vertical dimension and accepts three three different arguments: linear, poisson, and exponential.



Linear Ash Distribution The ash particles are linearly distributed using a random function with a uniform probability distribution function over the desired range.

Exponential Ash Distribution The following formula is used to place sample ash particles in the vertical distribution:

$$S = (0.25) \text{ x } (\text{plumeZwidth}) \cdot (R)$$

$$\text{height} = \text{plumeMax} - (\text{plumeZwidth}) \cdot (E) + S$$

where --plumeZwidth represents an approximate vertical range over which the ash will be distributed (default: 3000 meters), R is a uniformly distributed random number between 0 and 1, --plumeMax is the approximate top extent of the ash (meters), and E is a positive random number from an exponential distribution of unit mean.

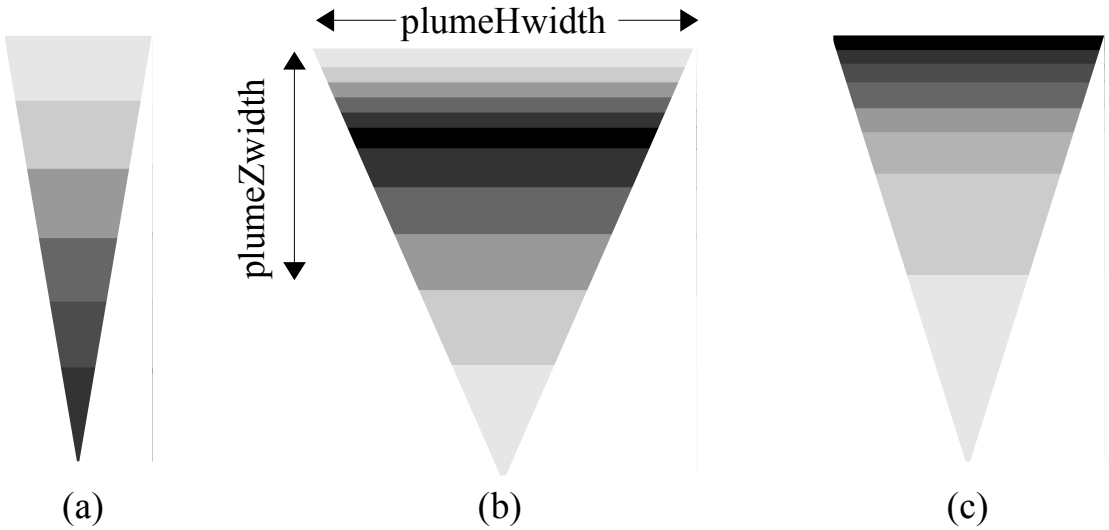
Poisson Ash Distribution The following formula is used to place sample ash particles in the vertical distribution:

$$S = (0.5) \cdot (\text{plumeZwidth}) \cdot (R)$$

$$\text{height} = \text{plumeMax} - (\text{plumeZwidth}) \cdot (P) + S$$

where --plumeZwidth is a vertical scale over which the ash will be distributed (default: 3000 meters), R is a uniformly distributed random number between 0 and 1, --plumeMax is the top extent of the ash (meters), and P is an integral value drawn from a Poisson distribution of unit mean.

The option --plumeHwidth affects the horizontal distribution of the ash particles. It uses a uniformly distributed random process to determine the ash particle location in a circle of radius --plumeHwidth (at the plumeMax height) centered on the volcano site. The horizontal displacement from a vertical line above the volcano is a random ±plumeHwidth value multiplied by the ratio of the particle height to plumeMax. So the net shape of the plume is an inverted cone where particles are located directly over the volcano at the lowest level and extend out further horizontally with increasing plume height.



The figure above shows an example of the three different initial cloud specifications with the attributes shown in the table. Shading indicates relative number density of particles with darker regions having more particles. Smaller values of plumeZwidth compress the dark banded region in (b) and (c).

	plumeShape	**PlumeHwidth [km]**	**PlumeZwidth[km]**
(a)	linear	1	N/A
(b)	poisson	3	0.5
(c)	exponential	2	0.25

The density of particles (shading) in the linear distribution does not appear linear in the figure because of the increasing plume width with higher elevations. The number of particles at any elevation is the same, but the cloud is more diffuse as the width increases. The shading would be constant with a plume width of zero. All plume shapes accept a plume width of zero, which corresponds to a line source. More control of the initial plume shape and concentration is possible by creating a restart file instead.

Specification of the volcano location is done using either --volc or --lonLat (the combination of --volcLon and --volcLat accomplish the same thing.) The decimal values should be separated with a slash, such as --lonLat 212.2W/44N. If a north/south/east/west direction is not specified (with a capital letter after the degree), degrees east and north are assumed. Specification of minutes or seconds is not allowed, and should be converted to decimal values instead (i.e. 63'20" = 63.333). When a volcano name is specified using the --volc option, the file volcanos.txt is consulted to find the volcano latitude, longitude, and elevation. (The PUFF_VOLCANO_LIST environment variable supersedes this, and the --volcFile option supersedes both). If the volcano name specified cannot be found in that file, an error results. Specification of the name is uppercase/lowercase insensitive, however, correct spelling is required. A shortened name can be used if it is unambiguous, so --volc=bezy is valid for Bezymianny, however, --volc=KORO is ambiguous (both Koro in Fiji and Korosi in Africa). The command 'puff --showVolcs' will produce a listing of all the known volcanoes, which is useful to check for correct spelling. The distribution comes with a default volcanos.txt file, which can be edited without recompiling the program. The location of the file is set using the environment variable PUFF_VOLCANO_LIST, and cannot alternatively be specified on the command line using the --volcFile option.

If the volcano name is not among the known list of volcanoes, the latitude and longitude can be specified instead. These options supersedes the --volc option. The option --lonLat can be used in place of --volcLon and --volcLat.

Program Input/Output Control

The following table of options control input and output from the program:

Option	**Default**	**Description**
argFile	none	read arguments from this file
fileDate	none	date/time sring for the input data
fileU.fileV,fileZ,fileT	none	specify the exact file to use for this data field
fileAll	none	use this file for all data fields
gridBox	none	force the bounds of the concentration file
gridOutput	FALSE	write a gridded concentration file
logFile	none	send all output to this file
newline	FALSE	write a new line at each time step
opath	./	write data to this directory
quiet	FALSE	write nothing to STDOUT
rcfile	none	use this resource file
RegionalWinds	none	reduce extents of global data
saveWinds	FALSE	save a netCDF file with the wind and dispersion coefficient, optionally specify file name.

sort	yes	sort ash by height before writing each file
verbose	FALSE	be verbose

Since the number of options specified can be numerous, they can be put in an argument file and Puff will read the file using --argFile. Each option should be listed on its own line, An initial '\' character may appear on the line for backwards compatibility but is not required. Comments can occur on their own line or after the arguments and are identified with the # character. Blank lines are ignored. For example, the following file is valid

```
# input file describing the February 19 eruption of Mt. Cleveland
volc Cleveland
eruptDate "2001 02 19 14:10"  # this is the second eruption that day
runHours=5
```

The argument file is processed in the order it appears on the command line. Values appearing in the argument file can be overridden on the command line by specifying the option after the argument file.

```
puff -argFile args.in -runHours 12
```

This would set the runHours to 12, over riding the 5 specified in the argument file. Several argument files can be specified as well, each being processed as they appear on the command line.

Data for the u and v components of wind are required. All data for a single variable must be in the same netCDF file. Different variable may be in the same or separate files. The location of these input data are specified using the puffrc resource file. Puff attempts to use the most recent data available based on the file mask. This can be overridden by using the --fileU, --fileV, etc. options.

The exact name of these wind components appearing in the data file must also be specified using the --varU and --varV options. Default values are lowercase u and v for longitudinal and latitudinal velocities, respectively, and upper case T and Z for temperature and geopotential height, respectively. Alternatively, the resource file may specify the names using varUname, varVname, varZname, and varTname. Puff will calculate a vertical component of wind using a divergence method. Use the --saveWinds option to save these calculated wind values along with temperature and horizontal diffusivity data in a netCDF data file. If no option value is given, the file will be named wind_puff.nc. If a value is given to the option, that name will be used in place of wind, for example --saveWinds=20040123 will save data to 20040123_puff.nc. This file can be used as an input file for further simulations if an appropriate line appears in the puffrc resource file. This may speed up further simulations because the vertical wind will not need to be calculated.

The extent of a global data set can be reduced using the --regionalWinds=X, where X is degrees. The extent of the global data set is reduced to a box ±X degrees in both directions around the volcano site. Using this option can improve speed because much less

computation occurs, such as calculating vertical winds and horizontal diffusion coefficients at locations never encountered. A significant improvement may occur when the data storage requirement of the global data would cause use of disk swap space, but regional data does not. When this option is used, the wind field becomes a regional data set. Puff does not read additional data when a particle exits the lateral extents. This option is ignored with input data that is already a regional set. The vertical extent of the input data cannot be reduced.

The default output directories are the current directory, and can be changed using the --opath option. One level of subdirectory can be created if it does not exist already. Existing files will be clobbered without warning.

While the simulation is running, a timer will appear in the lower left corner. If you want a new line to be printed at each time step, use the --newline option. This is generally not desirable, but could be piped to another process and used as a progress monitor. The --verbose flag will show several values of interest in the simulation, and should probably only be used when debugging. The --quiet option eliminates all output and is useful when used in scripts or cron jobs.

Log files can be specified using --logFile. Both stdout and stderr are redirected there. The --quiet option is set when using log files to suppress the time/date counter from filling the file.

The data written to the output ash files can be sorted prior to writing, which is useful when post-processing the data with another application. The default is to sort the particles based on height just prior to writing each file. Sorting can occur only at the beginning of the simulation by assigning the option no. In this case, larger particles may fall below smaller particles that have slower sedimentation velocities. If the argument 'never' is used, sorting is never done and the ordering is essentially random. The variable to sort on can also be changed, for example --sort=lon. Default behavior is to sort by elevation prior to writing each particle file.

In addition to the default output data files that contain the location of each individual particle, a gridded relative concentration file can be written using the --gridOutput option. A file is written with the name template YYYYMMDDHHmm_conc.nc, where the date corresponds to the beginning of the eruption specified by --eruptDate. Contrary to particle files, only one file is written that contains data for all times specified by --saveHours. The size of the grid can be specified by giving the --gridOutput option a value in the form dHorz/dVert, for example

```
--gridOutput=0.75/1500
```

which creates a grid 0.75 by 0.75 degrees in the horizontal direction and 1500 meters in the vertical. The latitude and longitude grid size must be the same. The lon/lat/height/time grid is just big enough to contain all the data. The size of the box can be specified using the --gridBox option with the format x1:x2/y1:y2/z1:z2 with values in decimal degrees east and meters high. Particles that move outside of the specified box are still tracked in the model and will appear in the particle files, but not in the concentration file. The time

dimension can not be modified and is necessarily zero to --runHours, every --saveHours.

Simulation Parameters

The simulation characteristics are controlled using the following options:

option	default	description
diffuseH	10000	Horizontal diffusion coefficient in m^2/s
diffuseZ	10	Vertical diffusion coefficient in m^2/s
dtMins	10	time step in minutes
eruptDate	none	date/time of the eruption start
eruptHours	3	duration of the eruption
runHours	24	simulation duration
saveHours	6	save interval
sedimentation	constant	type of sedimentation physics

Dispersion due to turbulence and Fickian diffusion is controlled by --diffuseH and --diffuseZ. The default value for horizontal diffusion is very conservative and generally overestimates dispersion observed by remote sensing techniques. Probably a more accurate estimation of the horizontal diffusion is based on local wind shear. To have puff calculate the horizontal diffusion based on local wind shear, use --diffuseH=turbulent. You can see the values of horizontal diffusion calculated by also using the --saveWinds option which creates a file wind_puff.nc that will contain the horizontal diffusion coefficient named kh.

The time step of the simulation is controlled using the option --dtMins. The default value is 10 minutes, which is small enough for simulations more than ~6 hours. For finer resolution, smaller values of --dtMins may help, however, the time resolution of the wind field data may be much more coarse precluding any actual improvement in simulation quality. For short duration simulations, the ash will appear to "puff" out if this value is too large. The value of dtMins must be an integer, for fine resolution append "sec" to the option (i.e. --dtMins=5sec). The minimum time step is 1 second.

The duration of the simulation can be changed the --runHours option. There is no set upper limit on the duration of a simulation. However, when time exceeds the range of the input wind field data, data for the final time will be used. Extrapolation does not occur. For this reason, wind field data files may need to be concatenated prior to use. Fractional values are acceptable, and will be rounded to the nearest second.

Data files are saved every interval defined by --saveHours. This value may be fractional, and a numerical value is expected. Optionally, the save interval in minutes can be specified by appending min, such as --saveHours=15min, which is equivalent to 0.25 hours.

The duration of the eruption is specified using --eruptHours. Fractional values are acceptable. The total number of particles ejected will be linearly distributed throughout the eruption hours. A value of zero causes all particles to be ejected at the commencement of

the simulation. There is currently no way to vary the intensity during a single eruption, although it is possible with the use of a restart file.

The date and time of the beginning of the eruption must be specified using --eruptDate. There are two acceptable formats: absolute and relative. Absolute times are of the form "YYYY MM DD HH:mm" or YYYYMMDDHHmm. Relative times are of the form "-N" where N is the number of hours prior to the current time (note the preceding dash). Fractional values are acceptable for relative time.

There are three physical descriptions of sedimentation physics available that can be specified using --sedimentation. They are 'constant', 'stokes', and 'reynolds'. The default is 'constant' and corresponds to a constant-temperature Stokes settling law. The other two options require temperature data and will adjust the air density and viscosity accordingly. The 'reynolds' option does not assume laminar flow (i.e. Stokes flow) and will calculate the local Reynolds number of each particle at each time step and use an empirical equation that accounts for turbulent, transitional, and laminar flow.

Digital Elevation Models

Puff can optionally use a digital elevation model (DEM) to determine when ash particles hit the ground. The type of DEM to use is specified with the --dem option, and the value corresponds to the name used in the resource file directive dem=..., see See The puffrc file. There are currently three types of DEM supported, GTOPO30 (http://edcdaac.usgs.gov/gtopo30/gtopo30.html) , DTED0 (geoengine.nima.mil), and USGS DEM in gridFloat format (http://seamless.usgs.gov). The names in the puffrc resource file must be 'gtopo30', 'dted0' or 'ned', respectively. GTOPO30 is a freely available DEM on a 30 arc-second grid, and is based on DTED0. GTOPO30 data comes in 32 tiles. DTED0 data primarily has a 30 arc-second resolution, however, extreme latitude data is 60 arc-second in longitude. DTED0 data comes in 1-degree by 1-degree tiles anywhere there is land. The USGS data comes in varying sized tiles from the Seamless site, depending on the data requested. This data set is referred to generically as ned for National Elevation Dataset, but actually can be any DEM data in gridFloat format with the appropriate header file.

The location of these data files must be specified in the puffrc resource file in order to Puff to find them. Puff expects a specific directory structure within the dem location (specified with path= in the puffrc resource file). For example, consider the following directives for all three DEM's

```
dem=gtopo30 path=$HOME/dem/gtopo30
dem=dted0 path=$HOME/dem/dted0/dted
dem=ned path=$HOME/dem/ned
```

These 3 names 'gtopo30', 'dted0' and 'ned' (lowercase) are the only accepted arguments to the --dem option. The directory names storing the data is arbitrary, however, in the example below they correspond. The expected directory structure would look like this:

```
gtopo30/
      W180N90.HDR
```

```
      W180N90.DEM

dted0/
      dted/
            e010/
                  n30.dt0
                  n31.dt0
                  n32.dt0
            e010/
                  n30.dt0
                  n31.dt0
                  n32.dt0

ned/
      11645938/
            11645938.flt
            11645938.hdr
```

It is advisable not to have any non-DEM files laying around in these directories because Puff may crash trying to interpret them as DEM files. The relative resolution can be reduced, which can significantly increase simulation speed, particularly when using GTOPO30. The resolution reduction occurs by factors of two and is specified optionally after the dem type using a colon, i.e. --dem=gtopo30:2, which would be a 4-fold (2 squared) reduction in resolution. Only values that evenly divide the number of DEM data points are valid. This option is not available for DTED0, but is not really necessary because of the small tile size. Because GTOPO30 tiles are larger, there is a larger overhead when reading higher resolutions, and using a lower resolution can have significantly improved speed.

Each tile is read into memory when an ash particle crosses into its boundary (contrary to how wind field data is handled). If a particle DEM tile does not exist, the DEM for that location will be ignored and the default ground-level value of zero meters will be used. Puff does not issue any warnings when a DEM tile is not found. To be sure the DEM tiles are actually being read, watch the screen output for something like

```
Reading DEM tile W180N90.DEM
```

In general, using a DEM significantly increases run time, and should probably only be used if fallout or near-surface transport information is desired.

ashdump Options

usage: ashdump [options] <file>

The <file> is a puff-generated particle file (i.e. ends in ash.cdf) All options are long names and accept either single or double dashes. Ashdump is a fairly simply utility that dumps data from the puff-generated NetCDF files to standard output. This output can be piped to other applications or a text file. When used without options, it will print a brief header with some details about the eruption and the data contained in the file, followed by a listing of

the location, size, and age of each particle in column format, one line per tracer particle. To see just the header, use the --header option. To see only some statistics (and the header), use --stats. A summary of the input parameters is displayed with --showParams. Puff generated concentration files (end in .nc) cannot be processed with ashdump. Use Unidata's ncdump or preferably NCO's ncks.

Filtering

Filtering is possible by location, and size using --range, --height and --size. For example, to only list particles in the box 210 to 220 degrees longitude, 55 to 65 degrees latitude, 3000 to 15000 meters, and of a size smaller than 100 microns

```
ashdump --range=210/220/55/65 --height=3000/15000 --size=0/1e-4
./200209260000_ash.cdf
```

The default behavior is to show both airborne particles and fallout. You can turn either one off using --fallout=false or --airborne=false. The "Total particles" reported in the header are those in the simulation, including those that have not yet erupted. Particles that have not erupted yet will not be shown or tallied as being in the specified range. This would happen with --eruptHours is greater than --saveHours and dumping one of the early particle files.

Output Control

A listing of any subset of the particle attributes is possible using the options --lat, --lon, --size, --height, and --age. These are boolean options and optionally accept 'true' or 'false' values. If no option is given, 'true' is assumed, so --lon is equivalent to --lon=true. Any number of these options can be combined. They can also be specified in a variable list using --variables=lon,lat,size. The column width is controlled using --width, but shouldn't be set to values less than necessary to print the values. The number of digits to the right of the decimal point is controlled using --precision. The default elevation unit is meters, use --height-in-feet to print in feet. Note that the --height filtering option will be interpreted in the same units.

Options for ashxp

usage: ashxp [options] <file(s)>
One or more puff-generated particle files can be processed at once. The postprocessor ashxp generates JPEG images from the data in puff-generated ash files. A bounding box is defined such that all particles are contained in the image with a minimum size of 256 x 256 pixels. If more than one ash file is specified on the command line, the bounding box for all images will the same, which is useful for creating animations or comparing results side by side.

option	default	description
--airborne [-a]	TRUE	Plot airborne particles
--bgfile [-b]	none	specify a background JPEG image

option	default	description
--border [-B]	1	# pixels in the border
--color [-c]	grey	particle coloring scheme
--fallout [-f]	TRUE	show grounded particles
--fontsize [-F]	12	fontsize for labels
--grayscale [-g]	FALSE	convert background image into a greyscale
--gridlines [-G]	NONE	specify lat/lon gridlines
--hgt-range [-r]	NONE	plot particles in this elevation range in meters
--include-out-of-bounds	FALSE	show particles that have left the boundary of the windfield
--latlon [-l]	NONE	force map bounds to this range
--label [-L]	NONE	add a label at a given lat/lon location
--label-file	NONE	specify a file containing a list of labels
--magnify [-m]	0	(de)magnify the image
--max-height [-h]	30000	maximum height in meters of plotted particles
--min-size [-x]	256	minimum map size in pixels
--no-background [-X]	FALSE	do not use a background image
--no-stamp	FALSE	do not add a date/time stamp
--output [-O]	(out=in) =~ s/cdf$/jpg/	write output JPEG image to this file name.
--patch	no	patch background image for missing pixels
--pixels [-p]	2	particle size in pixels
--projection [-P]	varies	use this map projection
--quiet	FALSE	quell all output
--report [-R]	FALSE	output short report
--size	NONE	only plot particles in this size range in millimeters
--sorted	FALSE	sort particles by height before plotting
--temp [-t]	FALSE	use an OS-derived temporary file for output
--timestamp [-T]	TRUE	print a date/time stamp
--usage	FALSE	print usage and exit
--verbose [-v]	FALSE	be verbose
--version [-V]	FALSE	print version and exit
--whole [-w]	FALSE	use entire background image

The default background image is set using the environment variable

ASHXP_BG_FILE_NAME, and can be overridden using the --bgfile option. To suppress a background image and plot the data on a black background, set --bgfile="" or use the --no-background flag. Processing speeds are increased when not using background images. Any JPEG image can be used as a background file, however, it only makes sense for it to be a global image otherwise the particles will not be mapped to the proper location. The background image can have any dimensions and ashxp will use the full resolution available to it. There are several color background images included with the distribution: high, medium, and low resolution.

By default, both airborne and grounded particles are mapped. Using either --fallout or --airborne suppressed mapping of the other. These options are exclusive. The available coloring protocols are 'gray', 'red', 'rainbow' and 'isopach'. The rainbow option produces color-coded elevation maps. Both 'isopach' and 'red' are density plots. The 'isopach' option color codes by the integrated relative density of ash at a pixel location. Note that the 'isopach' option is not restricted to fallout in the case of more conventional geologic isopach maps. They include all particles within the height range selected. Using --fallout produces a conventional isopach map. Three log-scale levels are shown in isopach maps, scaled by the maximum value. The 'red' option is similar to 'isopach' but a continuous, linear red-to-black scale is used. No elevation information is available with 'isopach' and 'red' mapping. However, height range filtering (--hgt-range) can be used in conjunction with this density mapping to create maps of relative density in a given elevation slab, like an aviation flight level for example.

The map projection can be specified using the --projection option. Valid values are 'lambert', 'polar', and 'mercator'. A default projection will be assigned based on the overall region being mapped. Polar projections are not allowed when ash crosses the equator. The conformal mapping tools from NOAA ARL are used to map the background image into the desired projection. This mapping sometimes results in holes in the projected image, particularly with larger values of --magnify. An attempt to patch these holes using a nearest neighbor interpolation is possible using --patch.

Turning on --sorted causes ashxp to sort all particles based on elevation, and plots higher elevation ash on top of lower elevation ash. This option creates images similar to what is seen in remote sensing data. It is turned off by default since puff sorts output by default (see the --sort option for puff). The --whole causes the entire background image to be used instead of clipping out only the relevant area. This is useful when creating a global perspective, but should be used judiciously in conjunction with high-resolution background images. This option is particularly useful for creating images to use with xplanet (http://xplanet.sourceforge.net); a powerful global image processing program.

Options for grid2pf

The GRIB to 'puff-friendly' utility does not need any non-filename arguments under most conditions. However, several options are available to help the utility create the correct data file.

option	default	description
-d	none	explicitly set date/time of the first input file
-g	none	explictly set the GRIB grid number
-n	FALSE	do not create a netCDF file
-o	YYYYMMDDHH_gridXXX.cdf	set the output file name
-s	FALSE	be silent
-v	FALSE	be verbose

Options for ashgmt

usage: ashgmt [options] <ashfile(s)>

All options are long, but just enough characters to be unambiguous are required. The options for ashgmt are very similar to ashxp, and both programs are available in the GUI. For more information on using ashgmt from the command line, see the extensive help menu using --help.

You need the Generic Mapping Tools to run ashgmt. If you used a binary installation, the absolute path to various GMT programs needs to be specified at the top of the ashgmt program. Use a text editor to edit it.

GUI (webpuff)

To start the browser-based GUI, run the webpuff program within the web/ directory. At initiation, you should see

```
starting webpuff... point your browser to http://127.0.0.1:1234/login.pl
```

See the Tutorial section for how to operate. Pressing 'logout' will stop the httpd daemon running on localhost, and you need to run the webpuff script again to restart it. All simulations are stored in a directory under the session ID name within the web/ directory, and can be recalled by resuming that session at login. The first eruption simulation will be named after the volcano, and appears in the 'previous' menu. Subsequent simulations will be numbered by volcano. A simulation listed in the 'previous' list can be deleted by selecting it and pressing the 'delete' button. When a previous simulation is selected, the simulation parameter values display will change to the values of that simulation. Therefore, the effect of changing only one parameter is possible by selecting the previous simulation and then manually changing only the parameter of interest. Press 'reset' to change them all to the default values specified in Webpuff.pm. The default values for webpuff are specified in the Webpuff.pm file and supersede the default values for the command-line puff. You can change the default values in Webpuff.pm if you wish. A short description of each option is available if you click on the option name. The models in

the 'model' drop down list should correspond to your puffrc resource file that resides in your home directory. If nothing appears, you do not have a resource file and the GUI will not work. The list of DEM's is also retrieved from the puffrc resource file.

Technical Details

This section describes some of the technical details about how the Puff model works. It is not required reading in order to run puff, but might be insightful when trying to interpret the results or push the limits of resolution.

Data Specification

Puff uses four-dimensional wind field data: three space dimensions and time. Only one data file per variable is used in each simulation, although several files may contain different variables. NetCDF files easily accommodate multiple records (a record is one entry in the time dimension), and it is fairly easy to concatenate several records into a single file. The NCO operator ncrcat works well (http://nco.sf.net). Which file is used is determined by the puffrc resource file or perhaps the --fileAll option. If more records are in the file than are required to cover the simulation length, those records are not read.

Time Interpolation

The input data in the time dimension does not need to cover the entire duration of the simulation. In fact, data for a single time is sufficient as long as it is before the eruption time. If you do not supply wind field data before the eruption time, an error will result and Puff will not execute.

If you specify a time series of data, Puff will interpolate between the time values. For example, assume at a specific location that there is a westerly wind of 10 mph at 12:00 pm. Furthermore, it is forecast that the westerly wind will increase to 22 mph by 6pm. Puff will linearly interpolate to obtain values at all time between 12pm and 6pm. Thus, there would be a 14mph wind at 2pm and a 18 mph wind at 4pm.

Puff does NOT extrapolate in time, but use the last values for which there is data and assumes a constant value there after. To continue the above example, if forecast data is not available after 6pm, a constant westerly wind of 22 mph will be assumed for all subsequent times.

Space Interpolation

A spline interpolation scheme similar to that used for the time series is also used for space. However, there is no extrapolation allowed in the space dimensions. If an ash particle travels beyond the extent of data either laterally or vertically, the 'exists' flag is turned off and further tracking of that particle is terminated. The simulation is terminated when all particles are either out of bounds or grounded. When an entire global wind field data set is

used (and the --regionalWinds option is not set), particles can travel around the globe repetatively. Because Puff internally uses a latitude/longitude grid, difficulties may arrise when particles travel over the poles where the changes in longitude at each time step can become very large. Setting a small value of --dtMins may help.

Missing Data

If part of the input wind field data is missing, Puff will attempt to patch it. Missing data in the netCDF file must be identified with the _MissingValue value. If the _MissingValue attribute does not exist, -9999 is assumed. This patching is done by puff on the fly and can be turned off by giving the --noPatch flag. The patching is done using spline interpolation. Patching can be done in all four dimensions: time and space. The amount of missing data in the input data is displayed when the wind field data is first read by puff and appears on a line such as this

```
Reading u from /home/rorik/data/gfs/200607260000_gfs.nc ... done.
Patching u data (10.23 %bad) ...
```

If this message does not appear, either the --noPatch option was used, or there was no missing data. It is a good idea to keep an eye on this value, because Puff will attempt to run with any percentage of bad data.

Puff does not extrapolate in the spatial direction. Therefore, there is a ceiling to the model domain. If a particle exits the ceiling, the 'exists' flag is set to false and further transport is not calculated. Particles that do not exist are not plotted by default, although this behavior can be changed. If it seems like there are no particles in the graphics, this is probably because their exited the ceiling or lateral boundary (in the case of regional data or global data with the --regionalWinds option).

Troubleshooting

In a perfect world, everything will always run just as you intended with no errors. In this world, problems will occur. Usually, puff will give you a short, perhaps cryptic, message describing the error. If that doesn't help, below is a list of possible reasons things might go wrong and how to fix the problem.

```
unknown volcano site

ERROR: Bad volcano site: "klychevskoi" ( -9999 , -9999 )

Expected (-180 < lon < 360) and (-90 < lat < 90)
Explicitly set -volcLon and -volcLat or use a predefined site with -volc
```

This volcano name is not in the file volcanos.txt (set by the environment variable PUFF_VOLCANO_LIST). Sometimes this is due to a misspelling, such as the above example where the correct spelling is Kliuchevskoi. A listing of all the volcanos is obtained using puff --showVolcs. Since this listing may be very long, piping the output to

the grep utility may be useful. Only enough letter need to be specified to make the name unambiguous.

```
WARNING: file /home/rorik/data/ncep/uwnd.1992.nc does not contain data
covering the eruption beginning at 1992 09 17 11:00
```

This error seems like a bug in the program, because obviously the 1992 NCEP reanalysis data file must contain data for 1992. However, because the original NCEP reanalysis data files use a time variable with units of "hours since 1-1-1", Puff has difficulty working with these relatively large values. The current work-around is to reference the time variable to something more managable. In this example, "hours since 1999-1-1" would be more appropriate. The script reference.pl, included in the Puff distribution, will do this for you. Note that the NCO operators (nco.sourceforge.net) are required to use this script.

```
ERROR: failed to initialize UDUnits library
```

This error may occur with a binary installation. You need to get the UDUNITS data file udunits.dat from http://puff.images.alaska.edu/download. Put it somewhere like the etc/ directory and define the environment variable UDUNITS_PATH to this location.

```
Cannot read fontfile [...]
```

This error may occur with a binary installation. You need to over ride the default font file using the --fontfile option to ashxp, or set the environment variable ASHXP_FONTFILE to this location.

Some arguments are getting ignored, like the options to gridOutput.

This often happens when the optional arguments are specified without using an equal sign. The --gridOutput option has an optional argument. When no argument is given, the gridded output is turned on with the default grid size. If you want to change the size, you must use an equal size, i.e. --gridOutput=0.25x500. Without the equal size, the argument and everything that appears after it on the line is ignored. This also happens with boolean options like --averageOutput. The following are all correct

```
puff --gridOutput=0.5x2000 --averageOutput
puff --averageOutput --gridOutput=0.5x2000
puff --averageOutput=true --gridOutput
puff --averageOutput=true --gridOutput=0.5x2000
```

To be safe, you can give every boolean option a true/false value with an equal size. You can also use an argument file which does not require an equal sign for any options, although you can still use them.

Puff License

Puff and related utilities are software for the tracking and simulation of wind-borne ash particles. They require a gridded wind data set for input, and may possibly require additional third-party software for visualization. The following text came with the source code for 'puff' and 'ashdump' from Craig Searcy, from which the PuffUAF versions originated. This license does not necessarily apply to 'ashxp', 'ashgmt', 'webpuff' and any other programs that is part of the Puff UAF suite. These programs are only restricted to the GNU General Public License in the next section.

Legal Terms

This work was produced as part of my thesis at the University of Alaska, Fairbanks. Additional support came from a COMET project with The National Weather Service, Alaska Region. Current support includes the NWS Forecast Office, Anchorage and the University of Alaska, Fairbanks.

AUTHOR
Craig Searcy

National Weather Service, Anchorage Forecast Office
6930 Sand Lake Road
Anchorage, AK 99502-1845 craig.searcy@noaa.gov
April 1999

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

    1. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

    2. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

    3. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution

of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

1. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

2. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

3. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further

restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO

WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.