AshDisperse

Release 0.1.2

Mark J. Woodhouse

Feb 07, 2022

CONTENTS:

AshDisperse is a numerical tool that solves the *steady-state* Advection-Diffusion-Sedimentation (ADS) Equation for a model volcanic eruption emission profile in a real wind field.

The solver is designed to be computationally efficient while solving the ADS equation accurately.

Note: This project is under active development.

See the *Usage* section for further information, including how to *install* the project.

CHAPTER

ONE

USAGE

1.1 Installation

Note: To use *AshDisperse*, it is recommended to use a virtual environment (here called `ash`).

First, install AshDisperse using pip:

```
(ash) $ pip install ashdisperse
```

This will install the package along with dependencies.

1.2 Command-line invocation

Once installed, you can run *AshDisperse* from the shell command-line using the command:

```
(ash) $ python -m ashdisperse
```

Note: *AshDisperse* uses the Numba package to accelerate computations. Numba compiles the Python code into fast machine code. The initial compilation step can take several seconds to complete, but results in substantial reduction in run times.

Following compilation, the *AshDisperse* command-line interface will appear, which allows the model to be set up by specifying parameters and meteorological data.

1.3 Interactive python session

AshDisperse can be used in an interactive Python session (e.g. in iPython) by importing the package.

First, launch an interactive Python session, e.g.

```
(ash) $ ipython
```

and then import *AshDisperse* (and some other packages that will be useful too):

[1]:
```
import ashdisperse as ad
import matplotlib.pyplot as plt
import numpy as np
import datetime  # loaded to report compilation is complete
print('AshDisperse compilation completed {}'.format(datetime.datetime.now()))
```

with the last line indicating that the Numba compilation is complete. Here we have import matplotlib for plotting, numpy for numerical data, and datetime simply for reporting compilation.

The compiled *AshDisperse* package is now available with the alias `ad`.

The command-line interface for setting the parameters and meteorological conditions can be accessed using the `setup` function, returning `params` and `Met` objects:

[2]:
```
params, Met = ad.setup(gui=False)
```

Further details of the *parameters* and meteorology are available.

Once the parameters and meteorology is set, the solver can be started using the `solve` function which returns a `results` object:

[3]:
```
result = ad.solve(params, Met, timer=True)
```

Here we have included the optional `timer` keyword argument that indicates timing of elements of the calculation will be printed.

Further details of the results are available.

CHAPTER

TWO

PARAMETERS

The *AshDisperse* package contains the `setup()` function that launches the command-line interface to guide the user through the setup of the model, producing a `parameters` object that contains the parameter settings needed to run a model simulation and a `Met` object that contains meteorological data (see meteorology for more details of the meteorological data inputs).

The parameter setup involves setting a number of model inputs split into categories that collect their purpose:

1. *Source settings*
2. *Grain parameters*
3. *Solver settings*
4. *Physical parameters*
5. *Output settings*
6. *Model parameters*
7. *Meteorological data parameters*

2.1 Source settings

The source settings characterise the volcanic eruption column and umbrella cloud from which ash is delivered into the atmosphere and dispersed.

We begin by setting the volcano source location, either as a named volcano (with location taken from the GVP Volcanoes of the World Database) or as a WGS84 latitude and longitude in decimal degrees. The location provides for georeferenced outputs and is used to extract meteorological data (see meteorology).

The emission model adopted here is a Gaussian distribution in the horizontal (x & y) and a Suzuki-style distribution in the vertical (z). These produce a three-dimensional spatial distribution over which the ash emission occurs.

The parameters set in the source settings are contained in `parameters.source` and are listed below, with their query in the CLI:

- **Enter Volcano name or give latitude and longitude of source (as decimal degrees in format lat, lon)**
  - The location of the volcanic source, either as named location from GVP dataset, or as a latitude and longitude.
  - Tnput either non-numeric text with name from GVP dataset (e.g. Etna), or comma-separated numeric values (e.g. 30.0, 15.5).
  - The latitude and longitude can be accessed as `parameters.source.latitude` and `parameters.source.longitude`, respectively. If the name is given, it can be accessed as `parameters.source.name`.

- **Mass eruption rate**
  - The intensity of the eruption, which characterized using a Mass Eruption Rate (MER, in kg/s). Note, here we required the MER of tephra (so excluding gases).
  - Enter as numeric value, including scientific notation (e.g. as 1000 or 1e3)
  - Accessed as `parameters.source.MER`.
- **Eruption duration**
  - The duration of the eruption (in seconds), which is used to determine the mass loading.
  - Enter as numeric value, including scientific notation.
  - Accessed as `parameters.source.duration`.
- **Plume height**
  - A plume height (in metres) is input to set the vertical height over which the emission occurs, denoted by H.
  - Enter as numeric value, including scientific notation.
  - Accessed as `parameters.source.PlumeHeight`.
- **Gaussian source radius**
  - The radius (in metres) of the Gaussian source distribution in the horizontal plane, denoted by R.
  - Enter as numeric value, including scientific notation.
  - Accessed as `parameters.source.radius`.
- **Select Suzuki emission profile parameter**
  - Choice of parameterization for the Suzuki-style distribution, either specifying the Suzuki k-parameter is input, or the altitude of the peak emission rate is specified
  - Enter either `k` or `peak`
- **Suzuki emission profile k-parameter**
  - If Suzuki emission profile parameter is specified as `k`, the Suzuki k-parameter is required and is denoted by k.
  - Enter a positive numeric value.
  - Accessed as `parameters.source.Suzuki_k`.
- **Suzuki emission profile peak-parameter**
  - If Suzuki emission profile parameter is specified as `peak`, the emission profile is specified using the altitude of the peak of the emission rate, input here as a proportion of the plume height (e.g. a value of 0.9 corresponds to a peak in the emission rate at 90% of the plume height).
  - Enter numeric value in range [0, 1].
  - Accessed as `parameters.source.Suzuki_peak`.

Note, there are two methods in `parameters.source` to compute the `Suzuki_peak` value from `Suzuki_k` (`parameters.source.peak_from_k()`) and vice versa (`parameters.source.k_from_peak()`).

2.2 Grain parameters

The grain parameters set characteristics of the tephra particles emitted during the eruption and are contained in the `parameters.grains` object.

We can set multiple grain classes, and in the CLI we must add grain classes until the sum of their proportions is equal to unity.

For each grain class, we specify:

- **Grain diameter**
  - The diameter of the grain (in metres), denoted by d, which is a major factor in determining the settling speed.
  - Enter as a positive numeric value, including scientific notation.
  - Accessed as `parameters.grains.diameter[j]` for grain class `j`.
- **Grain density**
  - The density of the grain (in kg/m^3), denoted by ρ, which also contributes to the settling speed.
  - Enter as a positive numeric value, including scientific notation
  - Accessed as `parameters.grains.density[j]` for grain class `j`.
- **Grain class proportion**
  - The proportion (mass fraction) of the grain class in the total grain size distribution.
  - Enter as a positive numeric value, including scientific notation.
  - Accessed as `parameters.grains.proportion[j]` for grain class `j`.

2.3 Solver settings

There are settings for the numerical solver that are specified in the solver parameters object, `parameters.solver`.

Each of the solver parameters has a default value.

The solver parameters are:

- **Dimensionless domain size in x**
  - Denoted by `domX` and accessed as `parameters.solver.domX`.
  - The advective distance in the x-direction (given by UH/W_s, where U is the maximum wind speed in x, H is the column height and W_s is the particle settling speed) is used to non-dimensionalize the x-coordinate.
  - Diffusion can carry particles further than the advective distance, so the domain is taken to be larger than the advective distance by a factor `domX`.
  - Default value `domX = 1.5`.
  - Require `domX` > 1.
- **Dimensionless domain size in y**
  - Denoted by `domY` and accessed as `parameters.solver.domY`.
  - The advective distance in the y-direction (given by VH/W_s, where V is the maximum wind speed in y, H is the column height and W_s is the particle settling speed) is used to non-dimensionalize the y-coordinate.

  - Diffusion can carry particles further than the advective distance, so the domain is taken to be larger than the advective distance by a factor `domY`.
  - Default value `domY = 1.5`.
  - Require `domY` > 1.

- **Minimum resolution in z**
  - Denoted by `minN` and accessed as `parameters.solver.minN`.
  - Input as log_2 of `minN`, denoted by `minN_log2` and accessed as `parameters.solver.minN_log2`.
  - Default value is `minN_log2 = 4` (`minN = 16`).

- **Maximum resolution in z**
  - Denoted by `maxN` and accessed as `parameters.solver.maxN`.
  - Input as log_2 of *maxN*, denoted by `maxN_log2` and accessed as `parameters.solver.minN_log2`.
  - Default value is `maxN_log2 = 8` (`maxN = 256`).

- **Tolerance for Chebyshev series**
  - Denoted by ϵ and accessed as `parameters.solver.epsilon`.
  - The Chebyshev spectral series (with coefficients a_n) is taken of degree N so that $|a_n| <$ `epsilon` for $n > N$.
  - Default value is $\epsilon = 10^{-8}$.

- **Resolution in x**
  - Denoted by `Nx` and accessed as `parameters.solver.Nx`.
  - Input as log_2 of `Nx`, denoted by `Nx_log2` and accessed as `parameters.solver.Nx_log2`.
  - Number of Fourier modes in the x-direction.
  - Default value is `Nx_log2 = 8` (`Nx = 256`).

- **Resolution in y**
  - Denoted by `Ny` and accessed as `parameters.solver.Ny`.
  - Input as log_2 of `Ny`, denoted by `Ny_log2` and accessed as `parameters.solver.Ny_log2`.
  - Number of Fourier modes in the y-direction.
  - Default value is `Ny_log2 = 8` (`Ny = 256`).

Note: The default resolutions for the horizontal coordinates, `Nx_log2 = 8` and `Ny_log2 = 8`, are recommended as the minimum for usable output. Lower values will compute solutions with fast runtimes but that the outputs of the calculation are potentially under-resolved. Lower values could be used as an initial check of parameter settings, followed by runs with higher resolution. See Copy and update parameters for details of how to update parameters.

2.4 Physical parameters

It is possible to change the values of the physical parameters in the model, which are contained in the `parameters.physical` object.

The physical parameters, with their default values, are:

- **Horizontal diffusion coefficient**
  - Denoted by κ_h and accessed as `parameters.physical.Kappa_h`.
  - The turbulent diffusivity (m^2/s) of the atmosphere in the horizontal directions (x and y).
  - Default value is $\kappa_h = 100$ m^2/s.
- **Vertical diffusion coefficient**
  - Denoted by κ_v and accessed as `parameters.physical.kappa_v`.
  - The turbulent diffusivity (m^2/s) of the atmosphere in the vertical direction (z).
  - Default value is $\kappa_v = 10$ m^2/s.
- **Gravitational acceleration**
  - Denoted by g and accessed as `parameters.physical.g`.
  - The gravitation acceleration (m/s^2), taken to be constant with altitude.
  - Default value is $g = 9.81$ m/s^2.
- **Viscosity of air**
  - Denoted by μ and accessed as `parameters.physical.mu`.
  - The dynamic viscosity of air (kg/m/s), taken to be constant with altitude.
  - The viscosity can affect the settling speed of grains, particularly for small sizes.
  - Default value is $\mu = 1.85 \times 10^{-5}$ kg/m/s.

2.5 Output settings

One of the outputs from *AshDisperse* is ash concentration in three-dimensions. The horizontal resolution is determined by the solver settings and grain parameters, but the vertical levels for output can be specified in the output settings, contained in the `parameters.output` object. A set of linearly spaced levels are specified using:

- **Lower altitude**
  - The lowest altitude level (in metres) to compute.
  - Zero is taken to be ground level.
  - Default is 0.0.
- **Upper altitude**
  - The highest altitude level (in metres) to compute.
  - Default is 20000.0 (i.e. 20 km).
- **Altitude step**
  - The increment in altitude (in metres) for output levels between the lower and upper altitudes.
  - Default is 2000.0 (i.e. 2 km).

2.6 Model parameters

AshDisperse computes several model parameters (typically dimensionless numbers constructed from parameters specified above) which are stored in the `parameters.model` object. There parameters cannot be set directly by the user. The model parameters are:

- **Settling speed scale**
  - An array of dimensional scales for the settling speed (m/s), with one value for each grain class.
  - Accessed as `parameters.model.SettlingScale`.
- **Velocity ratio**
  - An array of dimensionless numbers giving the ratio of the settling speed scale to the horizontal wind speed (taken at the plume top height), with one value for each grain class.
  - Accessed as `parameters.model.Velocity_ratio`.
- **Concentration scale**
  - An array of dimensional values (in kg/m^3) that are used to scale dimensionless ash concentrations computed in the model, with one value for each grain class.
  - Accessed as `parameters.model.cScale`.
- **Horizontal length scales**
  - An array of dimensional values used to scale the dimensionless horizontal distances in the model, with one value for each grain class. Note, the values are much larger than the dispersion distances, since the model requires negligable ash concentration at the edge of the grid.
  - Accessed as `parameters.model.xyScale`.
- **Dimensionless domain size in x**
  - An array of dimensionless values for the domain size in the x-direction, with one value for each grain class.
  - This is an adjustment of `domX` to account for the radius of the source. If the source radius is larger than the horizontal length scale (`parameters.model.xyScale`), then the domain must be made larger so that the source is not too close to the boundary. If the source radius is smaller than `parameters.model.xyScale`, the value is equal to `domX`.
  - Accessed as `parameters.model.Lx`.
- **Dimensionless domain size in y**
  - An array of dimensionless values for the domain size in the y-direction, with one value for each grain class.
  - This is an adjustment of `domY` to account for the radius of the source. If the source radius is larger than the horizontal length scale (`parameters.model.xyScale`), then the domain must be made larger so that the source is not too close to the boundary. If the source radius is smaller than `parameters.model.xyScale`, the value is equal to `domY`.
  - Accessed as `parameters.model.Ly`.
- **Source flux scale**
  - An array of dimensional source fluxes, with one value for each grain class.
  - The total mass eruption rate is divided according to the proportion specified for each grain class.
  - Accessed as `parameters.model.QScale`.
- **Peclet number**

  - Defined as $\mathrm{Pe} = \kappa_h/(UH)$ where κ_h is the lateral diffusivity, U is a scale for the wind speed, and H is the eruption plume height.
  - Characterizes the relative importance of diffusion to advection. If $\mathrm{Pe} \gg 1$ then diffusive transport dominates over advective transport, while advection dominates over diffusion if $\mathrm{Pe} \ll 1$.
  - Accessed as `parameters.model.Peclet_number`.
- **Diffusion ratio**
  - Defined as κ_h/κ_v, i.e. the ratio of the lateral to the vertical diffusivities of the atmosphere.
  - Accessed as `parameters.model.Diffusion_ratio`.

2.7 Meteorological parameters

The model parameters require characteristic scales from the meteorological data. These are stored in the `parameters.met` object and are:

- **Wind speed scale**
  - Denoted by U and accessed as `parameters.met.U_scale`.
  - Taken to be the maximum value of the wind speed over altitudes from ground level to the plume height H.
- **Settling speed scale**
  - Denoted by W_s and accessed as `parameters.met.Ws_scale`.
  - Each grain class has its own settling speed scale, so `parameters.met.Ws_scale` is an array.
  - Taken to be the settling speed for each grain class at ground level.

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search