rsas Documentation

Release 0.1.1

Ciaran J. Harman

January 17, 2015

CONTENTS

This library allows you to model transport through arbitrary control volumes using rank StorAge Selection (rSAS) functions.

CHAPTER

ONE

FURTHER READING

The rSAS theory is described in:

Harman, C. J. (2014), Time-variable transit time distributions and transport: Theory and application to storage-dependent transport of chloride in a watershed, Water Resour. Res., 51, doi:10.1002/2014WR015707.

CHAPTER

TWO

DOCUMENTATION FOR THE CODE

rsas.**solve**()

Solve the rSAS model for given fluxes

Args:

J [n x 1 float64 ndarray] Timestep-averaged inflow timeseries

Q [n x 2 float64 ndarray or list of n x 1 float64 ndarray] Timestep-averaged outflow timeseries. Must have same units and length as J. For multiple outflows, each column represents one outflow

rSAS_fun [rSASFunctionClass or list of rSASFunctionClass generated by rsas.create_function] The number of rSASFunctionClass in this list must be the same as the number of columns in Q if Q is an ndarray, or elements in Q if it is a list.

Kwargs:

mode ['age' or 'time' (default)] Numerical solution step order. 'mode' refers to which variable is in the outer loop of the numerical solution

mode='age' This is the original implementation used to generate the results in the paper. It is slightly faster than the 'time' implementation, but doesn't have the memory-saving "full_outputs=False" option. Only implemented for two outfluxes, and there is no option to calculate output concentration timeseries inline. The calculated transit time distributions must convolved with an imput concentration timeseries after the code has completed.

mode='time' Slower, but easier to understand and build on than the 'age' mode. Memory savings come with the option to determine output concentrations from a given input concentration progressively, and not retain the full age-ranked storage and transit time distributions in memory (set full_outputs=False to take advantage of this).

ST_init [m x 1 float64 ndarray] Initial condition for the age-ranked storage. The length of ST_init determines the maximum age calculated. The first entry must be 0 (corresponding to zero age). To calculate transit time dsitributions up to N timesteps in age, ST_init should have length m = M + 1. The default initial condition is ST_init=np.zeros(len(J) + 1).

dt [float (default 1)] Timestep, assuming same units as J

n_substeps [int (default 1)] (mode='age' only) If n_substeps>1, the timesteps are subdivided to allow a more accurate solution. Default is 1, which is also the value used in Harman (2015)

n_iterations [int (default 3)] Number of iterations to converge on a consistent solution. Convergence in Harman (2015) was very fast, and n_iterations=3 was adequate (also the default value here)

full_outputs [bool (default True)] (mode='time' only) Option to return the full state variables array ST the cumulative transit time distributions PQ1, PQ2, and other variables

C_in [n x 1 float64 ndarray (default None)] (mode='time' only) Optional timeseries of inflow concentrations to convolved progressively with the computed transit time distribution for the first flux in Q

C_old [float (default None)] (mode='time' only) Optional concentration of the 'unobserved fraction' of Q (from inflows prior to the start of the simulation) for correcting C_out

evapoconcentration [bool (default False)] (mode='time' only) If True, it will be assumed that species in C_in are not removed by the second flux, and instead become increasingly concentrated in storage.

Returns:

A dict with the following keys:

'ST' [numpy float64 2D array] Array of age-ranked storage for all ages and times. (full_outputs=True only)

'PQ' [numpy float64 2D array] List of time-varying cumulative transit time distributions. (full_outputs=True only)

'Qout' [numpy float64 2D array] List of age-based outflow timeseries. Useful for visualization. (full_outputs=True only)

'theta' [numpy float64 2D array] List of partial partition functions for each outflux. Keeps track of the fraction of inputs that leave by each flux. This is needed to do transport with evapoconcentration. (full_outputs=True only)

'thetaS' [numpy float64 2D array] Storage partial partition function fr each outflux. Keeps track of the fraction of inputs that remain in storage. This is needed to do transport with evapoconcentration. (full_outputs=True only)

'MassBalance' [numpy float64 2D array] Should always be within tolerances of zero, unless something is very wrong. (full_outputs=True only)

'C_out' [list of numpy float64 1D array] If C_in is supplied, C_out is the timeseries of outflow concentration in Q1. (mode='time' only)

For each of the arrays in the full outputs each row represents an age, and each column is a timestep. For N timesteps and M ages, ST will have dimensions (M+1) x (N+1), with the first row representing age T = 0 and the first column derived from the initial condition.

`rsas.`**`create_function`**`()`

Initialize an rSAS function

Args:

rSAS_type [str] A string indicating the requested rSAS functional form.

params [n x k float64 ndarray] Parameters for the rSAS function. The number of columns and their meaning depends on which rSAS type is chosen. For all the rSAS functions implemented so far, each row corresponds with a timestep.

Returns:

rSAS_fun [rSASFunctionClass] An rSAS function of the chosen type

The created function object will have methods that vary between types. All must have a constructor ("__init__") and two methods cdf_all and cdf_i. See the documentation for rSASFunctionClass for more information.

Available choices for rSAS_type, and a description of parameter array, are below. These all take one parameter set (row) per timestep:

'uniform'

Uniform distribution over the range [a, b].

- `Q_params[:, 0]` = a
- `Q_params[:, 1]` = b

'gamma'

Gamma distribution

- `Q_params[:, 0]` = shift parameter
- `Q_params[:, 1]` = scale parameter
- `Q_params[:, 2]` = shape parameter

'gamma_trunc'

Gamma distribution, truncated at a maximum value

- `Q_params[:, 0]` = shift parameter
- `Q_params[:, 1]` = scale parameter
- `Q_params[:, 2]` = shape parameter
- `Q_params[:, 3]` = maximum value

'SS_invgauss' Produces analytical solution to the advection-dispersion equation (inverse Gaussian distribution) under steady-state flow.

- `Q_params[:, 0]` = scale parameter
- `Q_params[:, 1]` = Peclet number

'SS_mobileimmobile' Produces analytical solution to the advection-dispersion equation with linear mobile-immobile zone exchange under steady-state flow.

- `Q_params[:, 0]` = scale parameter
- `Q_params[:, 1]` = Peclet number
- `Q_params[:, 2]` = beta parameter

'from_steady_state_TTD' rSAS function constructed to reproduce a given transit time distribution at steady flow. Assumes timestep is dt=1.

- `Q_params[0, 0]` = Steady flow rate
- `Q_params[1:, 0]` = Steady-state transit time distribution, from T = dt.

`rsas.`**`transport`**(*PQ*, *C_in*, *C_old*)

Apply a time-varying transit time distribution to an input concentration timseries

Args:

PQ [numpy float64 2D array, size N x N] The CDF of the backwards transit time distribution P_Q1(T,t)

C_in [numpy float64 1D array, length N.] Timestep-averaged inflow concentration.

C_old [numpy float64 1D array, length N.] Concentration to be assumed for portion of outflows older than initial timestep

Returns:

C_out [numpy float64 1D array, length N.] Timestep-averaged outflow concentration.

C_mod_raw [numpy float64 1D array, length N.] Timestep-averaged outflow concentration, prior to correction with C_old.

observed_fraction [numpy float64 1D array, length N.] Fraction of outflow older than the first timestep

rsas.**transport_with_evapoconcentration**(*PQ*, *thetaQ*, *thetaS*, *C_in*, *C_old*)

Apply a time-varying transit time distribution to an input concentration timseries

Args:

PQ [numpy float64 2D array, size N+1 x N+1] The CDF of the backwards transit time distribution P_Q1(T,t)

thetaQ, thetaS [numpy float64 2D array, size N+1 x N+1] Partial partition functions for discharge and storage

C_in [numpy float64 1D array, length N.] Timestep-averaged inflow concentration.

C_old [numpy float64 1D array, length N.] Concentration to be assumed for portion of outflows older than initial timestep

Returns:

C_out [numpy float64 1D array, length N.] Timestep-averaged outflow concentration.

C_mod_raw [numpy float64 1D array, length N.] Timestep-averaged outflow concentration, prior to correction with C_old.

observed_fraction [numpy float64 1D array, length N.] Fraction of outflow older than the first timestep

CHAPTER

THREE

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

r