Python notes

Contents

[Standards 1](#_Toc529087660)

[Functions 2](#_Toc529087661)

[Inputs 2](#_Toc529087662)

[accretion\_disk 2](#_Toc529087663)

[viscous\_timescale 2](#_Toc529087664)

[phase\_shift 2](#_Toc529087665)

[viscous\_frequency 3](#_Toc529087666)

[time\_division 3](#_Toc529087667)

[create\_variable\_array 3](#_Toc529087668)

[radius\_to\_annulus 4](#_Toc529087669)

[read\_variable\_array 4](#_Toc529087670)

[update\_variable\_array 4](#_Toc529087671)

[m\_dot 4](#_Toc529087672)

[M\_dot 4](#_Toc529087673)

# Standards

Use the Southampton University naming standards in:

<http://www.southampton.ac.uk/~feeg1001/pdfs/Python-for-Computational-Science-and-Engineering-slides.pdf>

Which in turn points to the Python.org style guide:

<http://www.python.org/dev/peps/pep-0008/>

# Functions

## Inputs

There are some common inputs across the functions for consistency:

annuli 1D array of the radii of the annuli

times 1D array of the number of time divisions of the annuli

array array of variable number of values per array (either m\_dot or M\_dot so far)

t time

r radius

value when updating a variable array the value to be placed at r,t

## accretion\_disk

Creates an array of the radii of all the. Annuli are geometrically distributed so that r/(r+1) is a constant

Inputs: Number of Annuli (N), inner radius (R\_in), outer radius (R\_out)

Outputs: Array R of the radii of that annulus

## viscous\_timescale

Function to calculate the viscous velocity at a particular radius using the formula given in (enter ref)

Inputs: radius

Output: viscous velocity at that radius

## phase\_shift

Function to calculate the delta viscous velocity at a given annulus (from the next one out)

Inputs: Annulus (radius is calculated from this)

Output: Delta Timescale

## viscous\_frequency

This creates an individual frequency value for each annulus

## time\_division

This calculates the number of time divisions required for each annulus from the formula:

sample ratio \* viscous timescale for that annulus

## create\_variable\_array

Creates a 1D array as a projection of a 2D array that has variable number of the 2nd dimension for each of the 1st dimension

**2D**

A1 A2 A2

|  |  |  |
| --- | --- | --- |
| t1 | t1 | t1 |
| t2 | t2 | t2 |
| t3 | t3 | t3 |
|  | t4 | t4 |
|  |  | t5 |

**1D**

A1 A2 A3

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| t1 | t2 | t3 | t1 | t2 | t3 | t4 | t1 | t2 | t3 | t4 | t5 |

To keep track an array is created of the number of timeslots per annulus

Number of entries (3,4,5…………)

Start of annulus n is at position sum(number of entries(1…n))

## radius\_to\_annulus

Returns the annulus number for a given radius

## read\_variable\_array

Reads a 1D projection of a 2D array given the 2 dimensions (radius and time). Radius is used to find an annulus and the time is used to give a linear interpolation of the value between the two nearest time points in the 1D array

## update\_variable\_array

Updates 1D projection of a 2D array given the 2 dimensions (radius and time). Radius is used to find an annulus and the time is the timeslot that is smaller or equal to the time requested (it is assumed that generally the time will be the same as the code will know how the array works)

## m\_dot

Function to calculate variable rate of change of mass in a particular annulus. For each annulus calculate the radius and viscous timescale then for each time division use the timescale as the frequency of a sine wave to calculate the small periodic changes in each annulus.

**\*\* Next step to replace the sine wave function with a red noise generator \*\***

Inputs: array of annuli,,array of time divisions (how many per annulus)

Output: Array of lowercase m\_dot changes

## M\_dot

Function to calculate the overall rate of change of mass in a particular annulus. For inner annuli the rate of flow is determined by the rate from the next annuli and the small change m\_dot. The rate from the outer annuli is time delayed by the time it takes the mass to cross that annuli (t\_offset). The code cycles through the annuli from the outside (penultimate annulus) to the inside. As the pattern is cyclical I assume that is the time is smaller than the offset we can start the cycle again

Inputs: array of annuli,,array of time divisions (how many per annulus)

Output: Array of Capital M\_dot changes