Disco Output Notes

Geoffrey Ryan

July 21, 2023

1 Output Types

`Disco` has 3 forms of output: *checkpoints*, *snapshots*, and *reports* (the `report.dat`).

	Checkpoints	→	Snapshots	→	`report.dat`
Size-per-write:	Large ($10 \to 10^3$MB)		Medium ($10 \to 10^3$kB)		Small ($1 - 10$kB)
Cadence:	Slow		Medium		Fast
Cadence Parameter:	`Num_Checkpoints`		`Num_Snapshots`		`Num_Reports`
Dimensionality:	2D/3D (full grid)		2D/1D (arrays)		0D (Scalar)

2 Checkpoints

Checkpoints are dumps of the full `Disco` grid and all its associated data. They are large, and contain enough information to restart a run. If you have a checkpoint, you have *everything*.

Because they are large, they cannot be output too often: writing them is slow, and you will quickly run out of disk space.

2.1 Diagnostics

Diagnostics are values that are computed (and time-averaged) over every timestep of the run, and then output with the checkpoints. These include ϕ-integrated fluxes and source terms for all conserved quantities, as well as a number of user-defined values.

3 Snapshots

Snapshots are a form of Disco output meant to be at a faster cadence than checkpoints, but with more information than the `report.dat`. They are similar to Diagnostics, but are not time averaged. Values in the snapshots are computed only when a snapshot is about to be output, this makes them a useful place for high-cost analysis (that cannot be run every timestep) that must be done on a faster cadence than the checkpoints. For example: computing the first 100 fourier moments of the mass distribution. There are two arrays inside the snapshots:

`Qrz` - shape: [Nz, Nr, Nq]. Each of the `Nq` quantities are ϕ-averaged, and output in the array according to their z and r.

`Qarr` - shape: [Narr]. Each of the `Narr` quantities are volume-integrated (NOT averaged). This array is otherwise arbitrary.

3.1 Reading with Python

There are two relevant functions in `discopy.util`: `loadSnapshotRZ()` and `loadSnapshotArr()`:

`loadSnapshotRZ(filename):`

- input: `filename`, the filename of the snapshot to load
- output: `(t, r, z, Qrz, rf, zf, planetDat)`
  - `t`: `(double(` the time
  - `r`: `(array` [Nz, Nr]`, double)` the radial positions of each entry in `Qrz`
  - `z`: `(array` [Nz, Nr]`, double)` the vertical positions of each entry in `Qrz`
  - `Qrz`: `(array` [Nz, Nr, Nq]`, double)` the snapshot data!
  - `rf`: `(array` [Nr + 1]`, double)` the radial positions of the intercell faces
  - `zf`: `(array` [Nz + 1]`, double)` the vertical positions of the intercell faces
  - `planetDat`: `(array, double)` planet stuff, as in the checkpoints

`loadSnapshotArr(filename):`

- input: `filename`, the filename of the snapshot to load
- output: `(t, r, z, Qrz, rf, zf, planetDat)`
  - `t`: `(double(` the time
  - `Qarr`: `(array` [Narr]`, double)` the snapshot data!
  - `rf`: `(array` [Nr + 1]`, double)` the radial positions of the intercell faces
  - `zf`: `(array` [Nz + 1]`, double)` the vertical positions of the intercell faces
  - `planetDat`: `(array, double)` planet stuff, as in the checkpoints

3.2 `diag_cb`

This setup outputs snapshot data for circumbinary accretion runs. There are `Nq` = 19 quantities computed.

0. Σ (surface density)
1. Σv^r (radial mass flux)
2. $\Sigma r^2 \Omega$ (angular momentum)
3. $\Sigma r^2 \Omega v^r$ (advective angular momentum flux)
4. P (pressure)

5. Σe_x (mass-weighted eccentricity vector x-component)
6. Σe_y (mass-weighted eccentricity vector y-component)
7. $\Sigma(\mathbf{r} \times \mathbf{g}_0)_z$ (gravitational torque from planet 0)
8. $\Sigma g_{0,x}$ (gravitational force from planet 0 x-component)
9. $\Sigma g_{0,y}$ (gravitational force from planet 0 y-component)
10. $\Sigma g_{0,z}$ (gravitational force from planet 0 z-component)
11. $\Sigma(\mathbf{r} \times \mathbf{g}_1)_z$ (gravitational torque from planet 1)
12. $\Sigma g_{1,x}$ (gravitational force from planet 1 x-component)
13. $\Sigma g_{1,y}$ (gravitational force from planet 1 y-component)
14. $\Sigma g_{1,z}$ (gravitational force from planet 1 z-component)
15. $\Sigma \cos\phi$ (1st cosine Fourier component of surface density)
16. $\Sigma \sin\phi$ (1st sine Fourier component of surface density)
17. $\Sigma \cos 2\phi$ (2nd cosine Fourier component of surface density)
18. $\Sigma \sin 2\phi$ (2nd sine Fourier component of surface density)

3.2.1 `diag_cb` - `Qrz`

Each of the 19 (=`Nq`) quantities are computed on the entire grid and then averaged in ϕ. The `Qrz` will have shape `[Nz, Nr, Nq]`.

So, for instance, `Qrz[:, :, 0]`= $(1/2\pi)\int_0^{2\pi} d\phi\Sigma$ and `Qrz[:, :, 1]`= $(1/2\pi)\int_0^{2\pi} d\phi\Sigma v^r$. The local accretion rate is $\dot{M} = -\int_0^{2\pi} d\phi r \Sigma v^r = 2\pi r$ `Qrz`$[:, :, 1]$.

3.2.2 `diag_cb` - `Qarr`

`Qarr` contains the same 19 quantities, but computed in the frame centered on each (moving) planet, on a radial grid centered on each planet. That is, they are the same 19 quantities, but calculated for the minidisks!

This is all stored in a large 1D array. There are 2 planets, `SNAP_NUM_R` (=30) radial bins, and `Nq` + 1 (=20) quantities computed for each, so the array is `2 x SNAP_NUM_R x (Nq+1) = 1200` elements long. This array is stored in memory (in the C-fashion) so it can be trivially `reshape`'d to a 3D array of shape `[2, SNAP_NUM_R, Nq+1]` = `[2, 30, 20]`.

The radial grid has `SNAP_NUM_R` (=30) linearly spaced bins between 0 and `SNAP_MAX_R` (=1.0).

Each entry in the array is the volume integral (really: sum) over cells whose center is in that radial bin. The first (index 0) entry is just the total volume integrated over. The rest are the other quantities, but shifted up by one index. So ΔV is index 0, $\Sigma\Delta V$ is index 1, etc:

$$\texttt{Qarr}[0, \texttt{idx}, 0] = \int_{r_{0,\texttt{idx}} < r_0 < r_{0,\texttt{idx}+1}} dV \equiv \Delta V \tag{1}$$

$$\texttt{Qarr}[0, \texttt{idx}, 1] = \int_{r_{0,\texttt{idx}} < r_0 < r_{0,\texttt{idx}+1}} dV \ \Sigma = \Sigma \ \Delta V \tag{2}$$

$$\texttt{Qarr}[0, \texttt{idx}, 2] = \int_{r_{0,\texttt{idx}} < r_0 < r_{0,\texttt{idx}+1}} dV \ \Sigma v^r = \Sigma v^r \ \Delta V \tag{3}$$

Here r_0 is the radial distance from the 0'th planet and $r_{0,\texttt{idx}}$ is the `Qarr` radial grid (for planet 0).

To compute the ϕ-averaged (about planet 1) surface density you need to divide by the ΔV of each bin:

$$\langle \Sigma \rangle_1 = \texttt{Qarr}[1, :, 1]/\texttt{Qarr}[1, :, 0] \tag{4}$$

To compute the ϕ-averaged (about planet 0) eccentricity vector you need to start with the mass-multiplied integral of $\Sigma\mathbf{e}$ and divide by the *total mass* $\Delta M \equiv \Sigma\Delta V$ of each bin:

$$\langle e_x \rangle_{\Sigma,0} = \texttt{Qarr}[0, :, 6]/\texttt{Qarr}[0, :, 1] \tag{5}$$

$$\langle e_y \rangle_{\Sigma,0} = \texttt{Qarr}[0, :, 7]/\texttt{Qarr}[0, :, 1] \tag{6}$$

4 Reports

Reports are small, each is a written as a single line to the `report.dat` text file. They are also fast, and can be output almost every time step. Each report consists of several (up to a few dozen) values, specified in the chosen setup from `Reports`. They are typically either point data (location of planets) or volume integrals (total mass on the grid, total gravitational torque).