Simulating a redshift catalog with SQL

Bridget Falck

May 26, 2010

This paper investigates how to load a simulation catalog into a SQL database and create a mock redshift catalog using the Simulation evironment created by Tamás Budavári.

1 Data

We use a Gadget simulation created by Miguel Aragon-Calvo that has 64^3 particles in a 200 Mpc/h box, which is low resolution but easy to work with. We create a halo catalog by tagging particles whose deformation tensor has 3 negative eigenvalues, implying that their initial Lagrangian cube has been inverted along 3 axes. This is calculated simply by comparing the final positions of initial nearest neighbors. Not all halo particles will have 3 negative eigenvalues in a given timestep, however, since the motion becomes nonlinear after caustic crossing, so we include information from as many timesteps as possible. We then add particles that are tessellation neighbors to, and have higher densities than, designated halo particles. Note that this is not an optimal method but represents what we were trying at the time this halo catalog was created.

The catalog we create contains a row for each particle and includes the particle ID, the (xyz) position and velocity, the halo ID (which is 0 if the particle is not in a halo), and the redshift. The primary key is set to the particle ID. Halo catalogs of larger and higher-resolution simulations will likely contain a row only for each halo and more columns, such as halo mass and number of particles. The catalog is added into the database directly using `Bulk Insert`, given the filename, the character that separates columns, and the character that terminates rows.

2 Mock redshift catalog

To create a mock redshift catalog from an nbody simulation, we define a light cone and place it within a Simulation instance, then search for the objects from the catalog that fall within the light cone. The Simulation instance automatically takes care of the periodic boundary conditions and, after assigning each particle a Peano-Hilbert index, can be very fast. The light cone Shape is put into a temporary table, which is compared with the data catalog using `inner join` to create the table of objects within the light cone. This table can be used to calculate the redshift, RA, and declination that make up the mock catalog.

3 Code

```
-- Define table structure according to your data file:
create table nbody (
       partid int,
       posx float,
       posy float,
       posz float,
       velx float,
       vely float,
       velz float,
       halo int,
       z float,
       primary key(partid) /* choose a column as primary key */
)
go

/* If loaded data already and want to reload, get rid of data but keep
structure: */

truncate table nbody

/* Insert ASCII data file (can also insert binary files): */

BULK INSERT nbody
   FROM '\\skydev\c$\temp\MMF200S_021.csv'
   WITH
      (
         FIELDTERMINATOR =',',
         ROWTERMINATOR ='\n'
      )
go

/*
At this point right click on dbo.nbody in current database, go to
design, and add the row phkey as int. Or look up the modify table add
row command...
*/

update nbody

/* Use Tamas's function to create Peano-Hilbert index (currently not
documented!) - the first argument is the number of bits and sets resolution,
i.e. 2^8 = 256 is good for a 200 Mpc/h box with only 64^3 particles: */
```

```
set phkey = dbo.fPHkey(8,convert(int,(posx/200000.)*256),
   convert(int,(posy/200000.)*256),convert(int,(posz/200000.)*256))
go

create index idx_nbody_phkey on nbody (phkey,posx,posy,posz)
go

declare @sim Simulation
set @sim = Simulation::newInstance(8,0,0,0,200000.,200000.,200000.)

/* Define a light cone giving [xc,yc,zc,xdir,ydir,zdir,angle,depth]: */

declare @qshape Shape3D
set @qshape = Shape3D::newInstance('Cone
   [100000,100000,100000,1,1,1,0.1,300000]')

/* Put simulation area within defined shape into a temporary table: */

select * into #fsim
from dbo.fSimulationCoverShape(@sim,@qshape,8)

/* Join simulation shape with data to get table of objects within shape.
Also, while we're at it, calculate the distance from the cone vertex
to each particle: */

select *, sqrt(power(posx+ShiftX-100000,2)+power(posy+shifty-100000,2)+
   power(posz+shiftz-100000,2)) as d
into #obs
from nbody n inner join #fsim fs
on n.phkey between fs.keymin and fs.keymax
where fs.FullOnly=0
and @qshape.ContainsPoint(posx+Shiftx,posy+shifty,posz+shiftz)=1

/* Now define mock observables. For example, calculate redshift as fn. of
distance from vertex of cone (using linear Hubble relation for small z,
but can use cosmological functions for larger z's and potentially use earlier
timesteps of the simulation) and an RA and dec defined wrt the cone vertex: */

select d/3e6 as z, asin((posz+shiftz-100000)/d) as dec,
   atan((posy+shifty-100000)/(posx+shiftx-100000)) as ra
from #obs
order by z
```