

CR-Analyze

A lightweight python package for analysis of CounterRotating components in TNG50



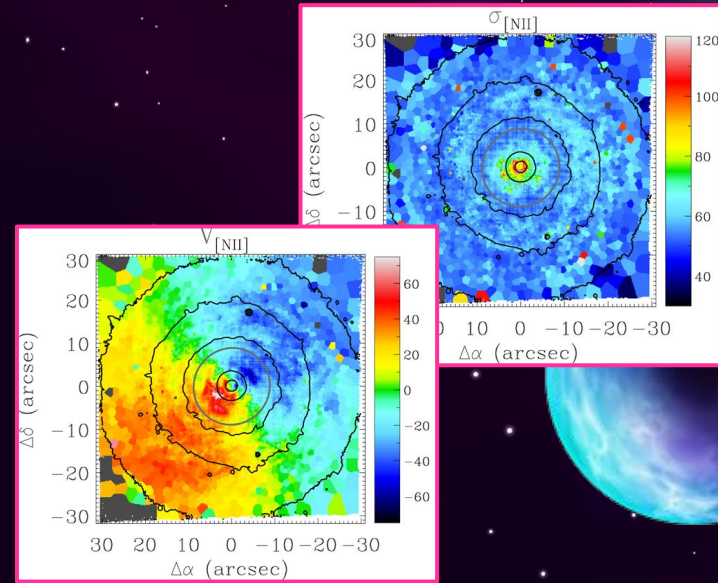
-By Marcos Bugueño

Estructura y concepto de Contra-rotacion

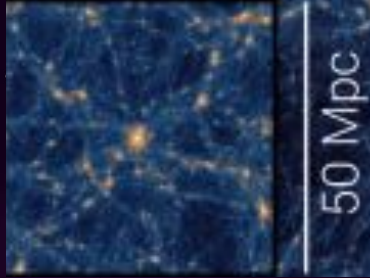
Lo intuitivo es esperar que una galaxia gire o rote con una dada dirección, especialmente en espirales, en donde tenemos: Trailing arms y leading arms.

Sin embargo, se ve en galaxias particulares que existe una componente que no sigue la regla.

> ¿El fenómeno entrega información sobre el pasado de una galaxia?



El laboratorio: TNG50



TNG50

La simulación de
menor volumen de
IllustrisTNG

Volumen de la caja cosmológica

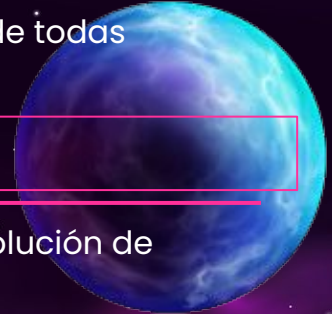
Es la simulación con el menor volumen (50 mpc^3) de todas

Resolución de masa

Sin embargo es la simulación con la más alta resolución de todas ($8.5 \times 10^4 M_\odot$)

Ideal para el análisis de las estructuras individuales

Esta última característica la vuelve en la simulación ideal para analizar las estructuras de los subhalos en sí mismas



CR-Analyze circularity.py

El código carece de:
Modularidad
Planteamiento
Y por ende transparencia en sus métodos

CR-Analyze particle_follower.py

El diagrama núcleo

CR_Analyze

rotator.py



spherical_coords_from_vector(vector)

matrix_from_spherical(r, theta, phi)

table_rotate(tabla, matriz)



star_particles_rotated_once_euler_method(tabla)

star_particles_rotated_n_euler_method(tabla)



>tabla rotada
>matriz de rotacion

circularity.py

circularities_euler_method

generate_cr_table



>tabla, con circularidades y
CR true{1} o false{0}

statistician.py



calculate_r_half(tables)



distribution_of_r_half(table)



distribution_of_metallicity(table)



detect_origin(particle_table)



>tabla de distribuciones

moviemaker.py

generate_time_table_full

generate_time_tables_full

generate_time_table



generate_time_tables



record



>tablas por snap
>gif/avi de la galaxia

Comentarios/documentacion?

Que es un codigo sin comentarios claros mas que una sarten de instrucciones a un pastel que no conoces?

- La importancia de colocar comentarios claros que declaren lo que se realiza en el proceso, y lo que se obtiene es clave

- De estos mismos surge la "documentacion del codigo", buenos docstring pueden ser usados con facilidad por el usuario con "paquete.funcion?"



Comentarios/documentacion?

- Se selecciono finalmente solo utilizar docstrings
- No se hace uso de las herramientas Doxygen o Sphinx.
- ambas requieren de un formato específico del docstring
- se interpone a la funcionalidad mencionada anteriormente.



Descripción de códigos y métodos.

Se realiza también comentarios completos de las funciones.

- Se realizó para todas las funciones desarrolladas

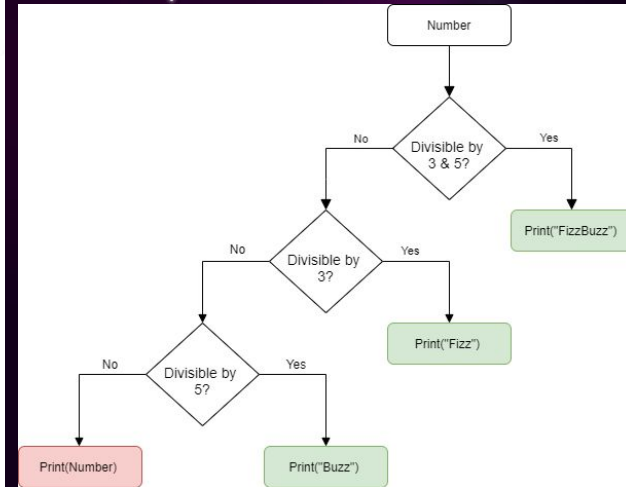
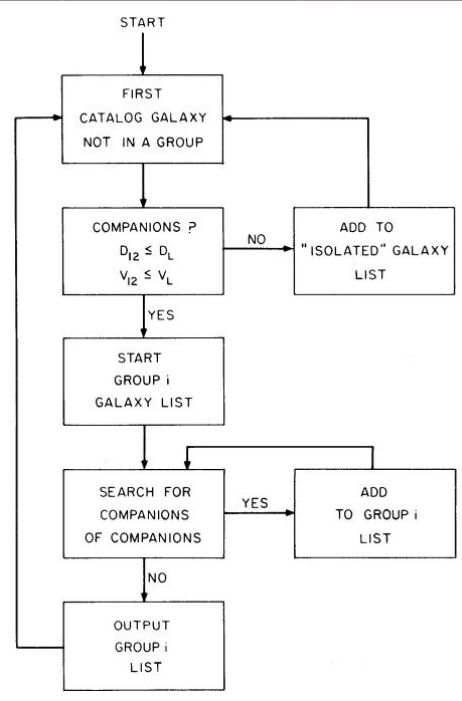
- Se comienza inicialmente comentando los métodos antiguos.. Sin embargo...

```
def table_rotates_once_angularmomenta(tabla, reference_tabla, debug=False):  
    """ 8/1000  
    Rotates a table of particles by using a given reference table. Both must be in dict form with values of "  
    Parameters:  
    - tabla (dict) a dictionary with the form {"Coordinates":numpy.array(N,3),"Velocities":numpy.array(3,N)}  
    - reference_tabla (dict) a dictionary with the form {"Coordinates":numpy.array(N,3),"Velocities":numpy.ar  
    Returns:  
    - tabla (dict), the same dictionary but with its coordinates and velocities rotated  
    - M (np.array(3,3)) the rotation matrix calculated and used. ideally for later saving.  
    """  
    # We assume that the conversion to physical coordinates has been done already, as such we calculate the a  
    # we assume as well that {0,0,0} would be the center of the subhalo or galaxy  
    reference_tabla["Angular_Momentum"] = np.cross(reference_tabla["Coordinates"],reference_tabla["Velocities"]  
    reference_J = sum(reference_tabla["Angular_Momentum"]) #Referential angular momentum  
    # this is the actual angle to rotate from the reference J  
    def frame_of_reference_change(r, v, central_pos, central_vel, L_box  
        """  
        A function that changes the frame of reference  
        in a periodic box.  
        Parameters  
        - r np.array(N,3) coordinates of particles  
        - v np.array(N,3) velocities of particles  
        - central_pos np.array(1,3) coordinate of center  
        - central_vel np.array(1,3) velocity of center  
        - L_box float the length of the periodic box  
        Returns  
        - r np.array(N, 3) the coordinates of the particles  
        - v np.array(N, 3) the velocities of the particles  
        """  
    def  
    - N_rotations (int) number of rotations, 3 by default.  
    - R_bulge (float) some definition of the radius of the bulge of the galaxy, by default is 0.  
    - Zmin (float) minimum metallicity to consider for reference particles.  
    - Zmax (float) maximum metallicity to consider for reference particles.  
    - debug (bool) whether to print debugging messages, like obtained angular momentum or rotation matrix obt  
    """
```

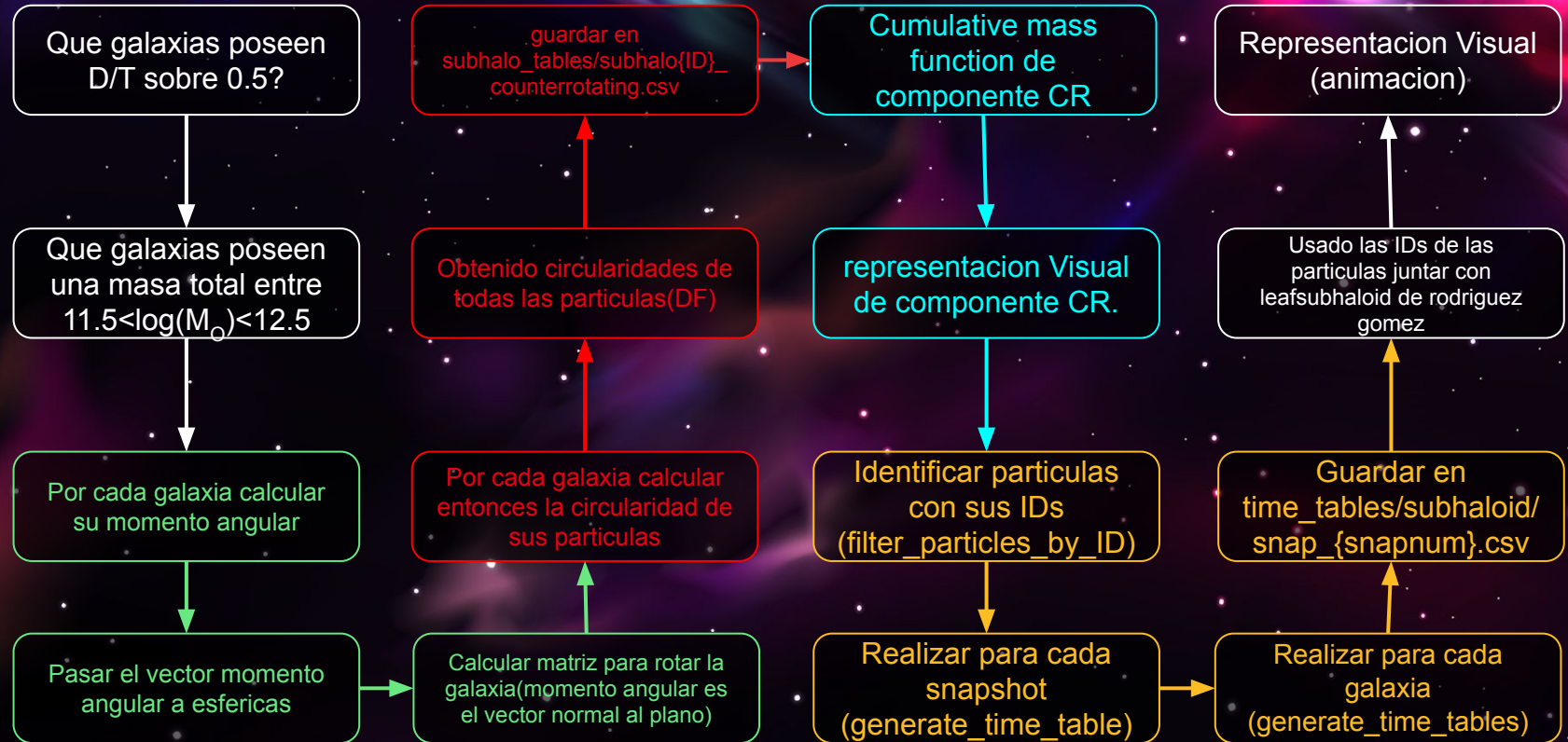

Diagramas de flujo in a nutshell?

Un diagrama de flujo es una representación visual de los datos a través de algún proceso.

Son especialmente útiles a la hora de diseñar software.



Flujo del método antiguo.



Flujo básico de la rotación



Cargar



Calcular momentum angular



Coordenadas esféricas del
momentum angular.

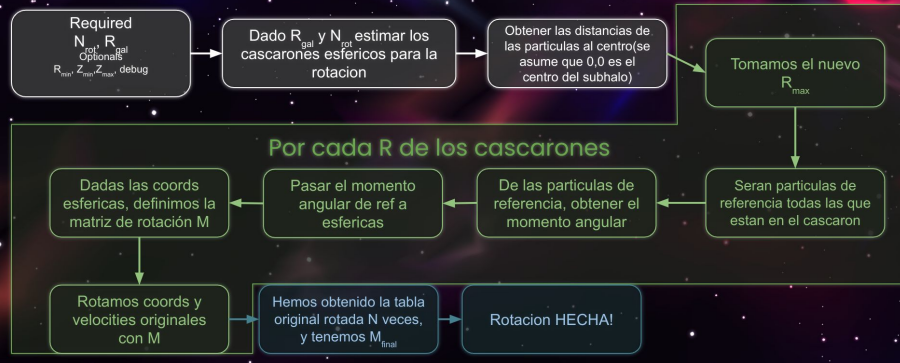


Matrices de rotación, según
 ϕ y ψ .



Rotar partículas con
matrices

Flujo de rotator.py

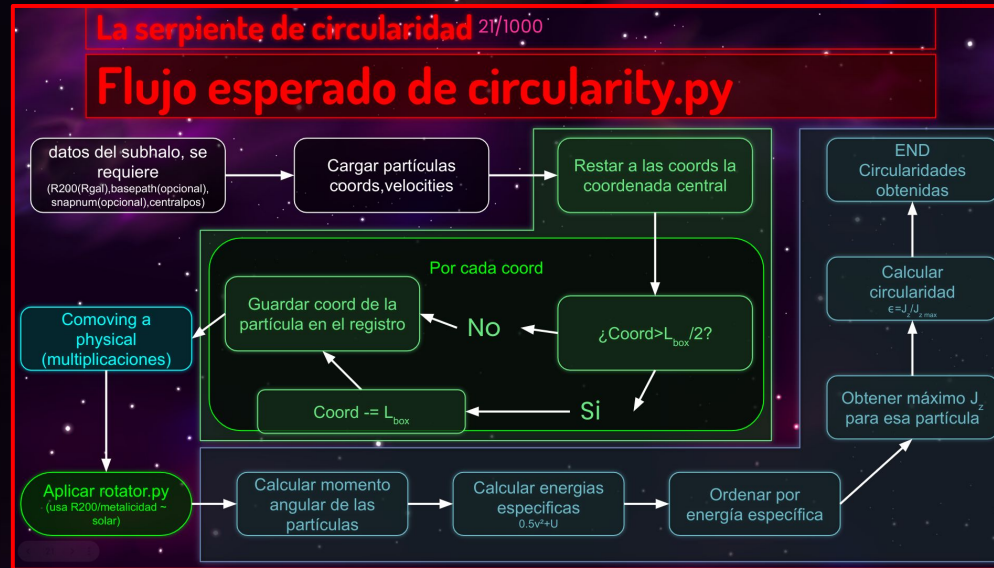


Flujo básico de la circularidad

Mover marco de referencia (tener en cuenta la periodicidad de la caja)

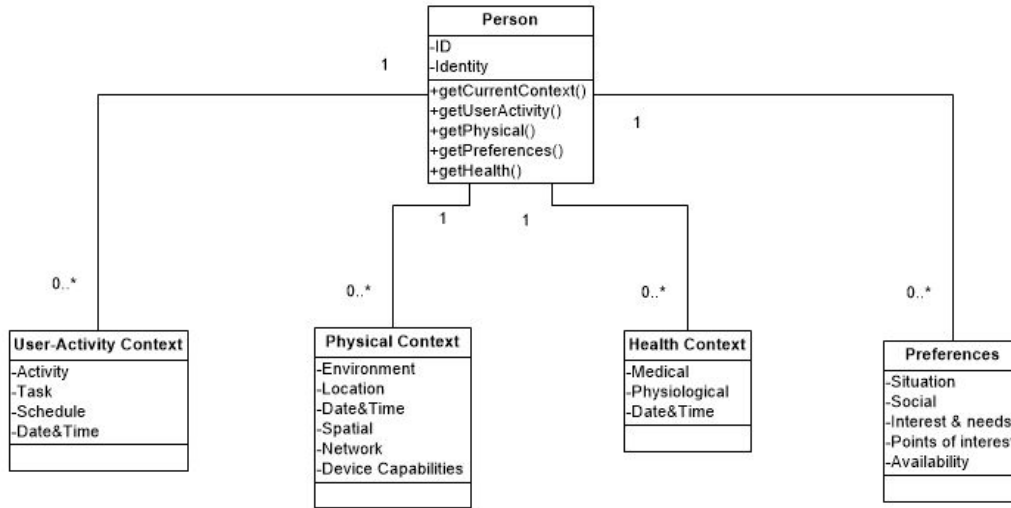
Rotar las partículas para tener la galaxia edge-on

Calcular las circularidades



UMLs?

UML Class Diagram



Los Modelos Unificados de Lenguaje, son muy utiles para el diseño de métodos y las relaciones entre ellos.

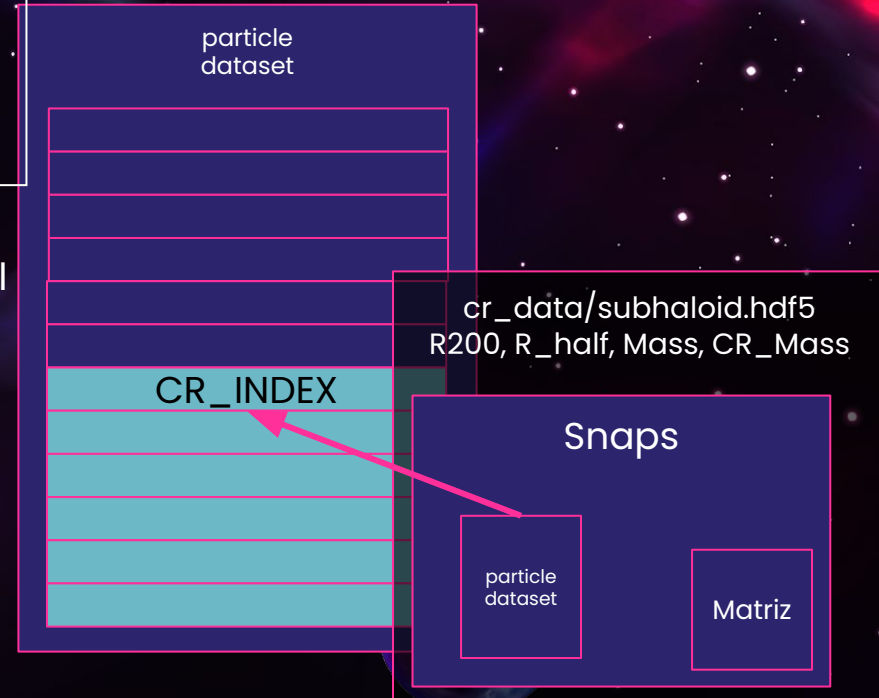
-En el proyecto se hace uso de ellos de manera exhaustiva en el desarrollo de los scripts.

UML de rotator.py



HDF5 y guardado

- Se plantea el guardado en HDF5 ya en el UML de circularity.py
- Esta planteado el formato de guardado del mismo.



¿Porcentaje de finalización del
proyecto?

CR_Analyze

rotator.py

`spherical_coords_from_vector(vector)``matrix_from_spherical(r, theta, phi)``table_rotate(tabla, matriz)``star_particles_rotated_once_euler_method(tabla)``star_particles_rotated_n_euler_method(tabla)`

>tabla rotada
>matriz de rotacion

circularity.py

`circularities_euler_method``generate_cr_table`

>tabla, con circularidades y
CR true{1} o false{0}

statistician.py +

`calculate_r_half(tables)` +`distribution_r_half(table)` +`distribution_of_velocity(table)` +`detect_origin(particle_table)` +

>tabla de distribuciones

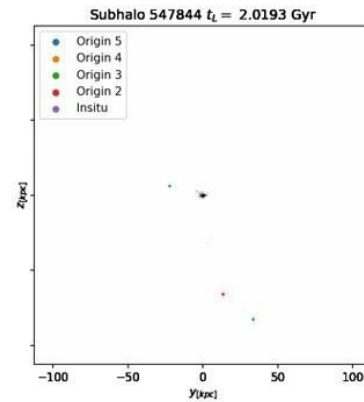
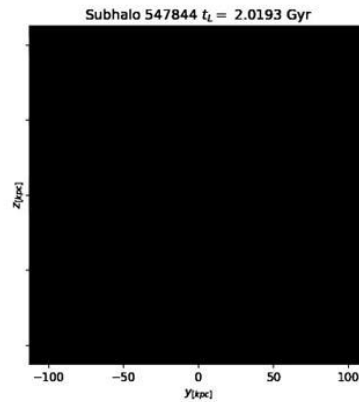
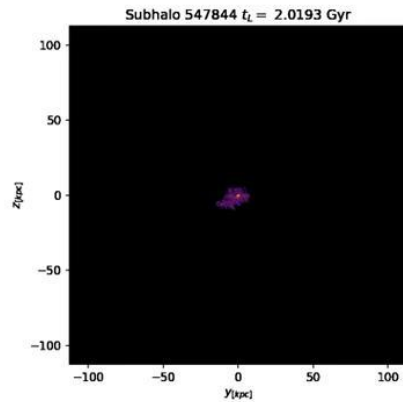
moviemaker.py

`generate_time_table_full``generate_time_tables_full``generate_time_table``generate_time_tables``record` +

>tablas por snap
>gif/avi de la galaxia

~48%

Remarcaciones finales



¿Y cual es el giro argumental?

- ¡Los nuevos métodos no son sólo aplicables a IllustrisTNG!
- Se desarrollaron los métodos con la alternativa(fijarse en los parámetros opcionales) de entregarle los datos en "crudo" de partículas(dado que estén clasificadas en subhalos) de otras fuentes.

Entonces, existe la posibilidad de aplicarlos en múltiples entornos :D

Gracias!

CR-Analyze



Area de Extras

Digamosle apendice



Flujo de rotator.py

Required
 N_{rot} , R_{gal}
 Optionals
 R_{min} , Z_{min} , Z_{max} , debug

Dado R_{gal} y N_{rot} estimar los
 cascarones esfericos para la
 rotacion

Obtener las distancias de
 las particulas al centro (se
 asume que 0,0 es el
 centro del subhalo)

Tomamos el nuevo
 R_{max}

Por cada R de los cascarones

Dadas las coords
 esfericas, definimos la
 matriz de rotación M

Pasar el momento
 angular de ref a
 esfericas

De las particulas de
 referencia, obtener el
 momento angular

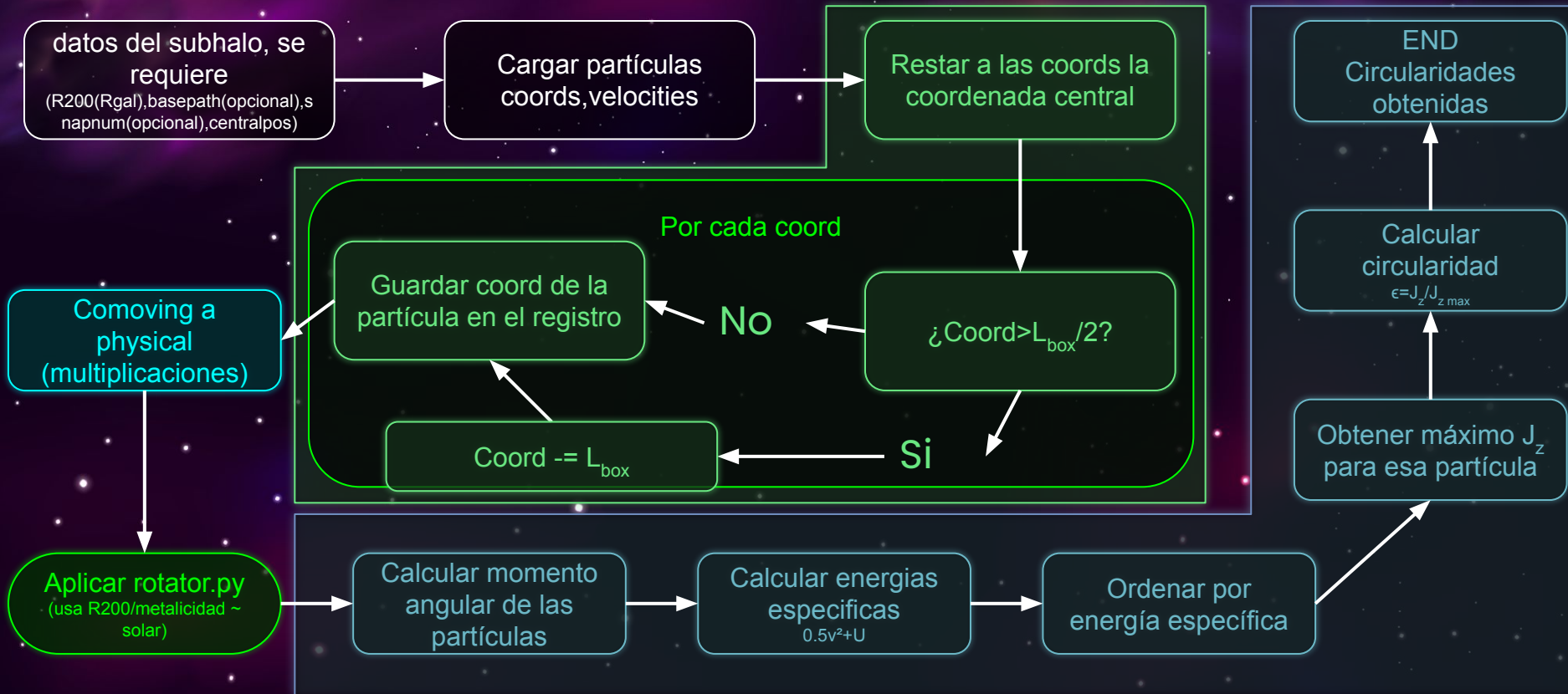
Seran particulas de
 referencia todas las que
 estan en el cascaron

Rotamos coords y
 velocities originales
 con M

Hemos obtenido la tabla
 original rotada N veces,
 y tenemos M_{final}

Rotacion HECHA!

Flujo esperado de circularity.py



¿Que paso con los objetos?

-Realmente a lo largo del diseño de los algoritmos, se fueron encontrando diversas maneras de aprovechar, la funcionalidad de objetos.

25/1000

`numpy.where(condition, [x, y,])`

Return elements chosen from *x* or *y* depending on *condition*.

Note

When only *condition* is provided, this function is a shorthand for `np.asarray(condition).nonzero()`. Using `nonzero` directly should be preferred, as it behaves correctly for subclasses. The rest of this documentation covers only the case where all three arguments are provided.

Parameters: *condition* : *array_like, bool*

Where True, yield *x*, otherwise yield *y*.

x, y : *array_like*

Values from which to choose. *x*, *y* and *condition* need to be broadcastable to some shape.

```
- v np.array(N, 3) the velocities of
"""
r -= central_pos
v -= central_vel
# now we need to check if any coordinate from the particles
# goes beyond L_box/2
for i in range(3):
    periodicity_fix = np.where(r[:,i]>L_box/2,L_box,0)
    r[:,i] -= periodicity_fix
    continue
# now the whole system should be centered around central_pos
return (r, v)
```

analyze/circularity.py

Por eso recuerden
RTFM!