Dinámica de galaxias, una simulación con $N \log N$ iteraciones.

Juan Barbosa

Introducción

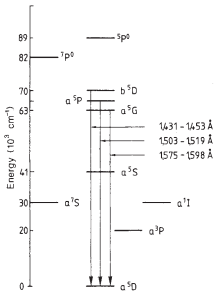


Fig. 2 A partial term scheme for Fe III, showing the observed transitions, some lower-lying terms of the same even parity and the first two terms of odd parity. Term energies are given in cm^{-1}.

abilities found by Garstang[8] for lower-lying levels are in each case ~1. Thus it is likely that the upper levels have populations close to their Boltzmann values with collisional de-excitation rates exceeding radiative decay rates. Then the ratio of the [Fe III] line fluxes to those of permitted transitions would be sensitive to the electron density. Detailed calculations of the atomic data are required to establish the collisional–radiative regime for the individual lines. The upper levels may attain only a pseudo-Boltzmann population if collisional excitation to higher states of odd parity exceeds the rate for collisional de-excitation to lower levels. In any case the new [Fe III] identifications will provide more information on the structure of the solar chromosphere–corona transition region.

If the a^{5}G, a^{5}P and b^{5}D levels are collisionally de-excited in the solar atmosphere, they could become stronger, relative to permitted transitions of species of similar excitation, in astrophysical sources of lower electron density. For this reason their presence is being investigated in such sources, including the Seyfert galaxy NGC 4151, which has unidentified emission features around 1,575, 1,581 and 1,518 Å (refs 13 and 14). [Fe III] emission is observed in the optical spectrum of NGC 4151[15,16]. The relative intensities of the quintet transitions in NGC 4151 and other sources cannot be predicted until collision cross-sections and transition probabilities are known, and these are urgently required to establish whether or not the [Fe III] lines are of wider astrophysical significance.

Received 3 September; accepted 23 October 1986.

1. Bartoe, J.-D. F., Brueckner, G. E., Purcell, J. D. & Tousey, R. *Appl. Opt.* **16**, 879–885 (1977).
2. Sandlin, G. D., Bartoe, J.-D. F., Brueckner, G. E., Tousey, R. & VanHoosier, M. E. *Astrophys. J. suppl. Ser.* **61**, 801–898 (1986).
3. Jordan, C., Brueckner, G. E., Bartoe, J.-D. F., Sandlin, G. D. & VanHoosier, M. E. *Astrophys. J.* **226**, 687–697 (1978).
4. Bartoe, J.-D. F. *et al.* in *AIAA 24th Aerospace Sciences Meeting, Reno*, AIAA-86-0225 (in the press).
5. Moore, C. E. in *Selected Tables of Atomic Spectra: Atomic Energy Levels and Multiplet Tables, Si I.* NSRDS-NBS-3, Sect. 2 (1967).
6. Edlén, B. & Swings, P. *Astrophys. J.* **95**, 532–554 (1942).
7. Reader, J. & Sugar, J. *J. phys. chem. Ref. Data* **4**, 353–440 (1975).
8. Garstang, R. H. *Mon. Not. R. astr. Soc.* **117**, 393–405 (1957).
9. Garstang, R. H., Robb, W. D. & Rountree, S. P. *Astrophys. J.* **222**, 384–397 (1978).
10. Abbott, D. C. *J. Phys.* **B11**, 3479–3497 (1978).
11. House, L. L. *Astrophys. J. suppl. Ser.* **8**, 307–328 (1964).
12. Munro, R. H. & Withbroe, G. L. *Astrophys. J.* **176**, 511–520 (1972).
13. Ulrich, M. H. *et al. Nature* **313**, 747–751 (1985).
14. Cassatella, A. & Jordan, C. (in preparation).
15. Boksenberg, A. *et al. Mon. Not. R. astr. Soc.* **173**, 381–386 (1975).
16. Boksenberg, A. & Penston, M. V. *Mon. Not. R. astr. Soc.* **177**, 127*P*–131*P* (1976).

A hierarchical $O(N \log N)$ force-calculation algorithm

Josh Barnes & Piet Hut

The Institute for Advanced Study, School of Natural Sciences, Princeton, New Jersey 08540, USA

Until recently the gravitational *N*-body problem has been modelled numerically either by direct integration, in which the computation needed increases as N^2, or by an iterative potential method in which the number of operations grows as $N \log N$. Here we describe a novel method of directly calculating the force on N bodies that grows only as $N \log N$. The technique uses a tree-structured hierarchical subdivision of space into cubic cells, each of which is recursively divided into eight subcells whenever more than one particle is found to occupy the same cell. This tree is constructed anew at every time step, avoiding ambiguity and tangling. Advantages over potential-solving codes are: accurate local interactions; freedom from geometrical assumptions and restrictions; and applicability to a wide class of systems, including (proto-)planetary, stellar, galactic and cosmological ones. Advantages over previous hierarchical tree-codes include simplicity and the possibility of rigorous analysis of error. Although we concentrate here on stellar dynamical applications, our techniques of efficiently handling a large number of long-range interactions and concentrating computational effort where most needed have potential applications in other areas of astrophysics as well.

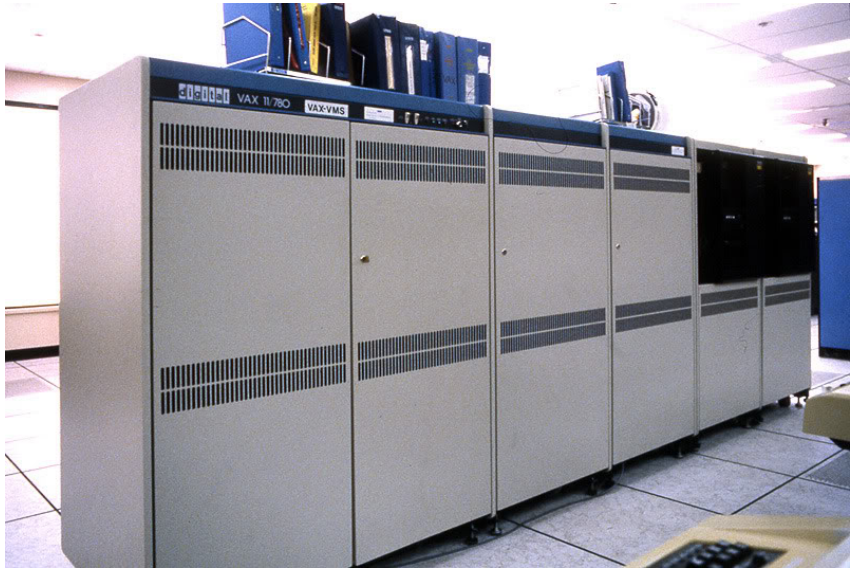
Until recently, the dynamics of a system of self-gravitating bodies (the gravitational *N*-body problem) has been modelled numerically in two fundamentally different ways. The first one, direct *N*-body integration, involves the computation of all $\frac{1}{2}N(N-1)$ forces between all pairs of particles. This allows an accurate description of the dynamical evolution but at a price that grows rapidly for increasing N[1]. The second way involves a two-step approach: after fitting the global potential field to a special model with a number of free parameters, each particle is propagated in this background field for a short time before the same procedure is reiterated. The potential method involves a number of operations that grow only as N log N. Thus calculations can be performed more quickly, but with a loss of accuracy and generality. The special nature of each potential-solving code is caused by the need to use some technique that is tuned to the geometry of the problem being considered (such as Fourier transforms or spherical or bispherical harmonics[2]).

Recently, some of the advantages of both approaches have been combined by using direct integrations of force while grouping together increasingly large groups of particles at increasingly large distances. This corresponds to the way humans interact with neighbouring individuals, further villages and increasingly further and larger states and countries—driven by increasing cost and decreasing need to deal with more removed groups on an individual basis. The first implementation of such a hierarchical grouping of interactions was given by Appel[3], who used a tree structure to represent an *N*-body system, with the particles stored in the leaves of the tree. An independent implementation by Jernigan[4] and Porter[5] incorporated regularization of close encounters. However, in both codes the logarithmic-growth gain in efficiency comes at the price of introducing additional errors that are hard to analyse because of the arbitrary structure of the tree. Nearby particles may be grouped as leaves of nearby branches, but the phase-space flow of realistic self-gravitating systems demands a continuous updating of the tree structure to avoid tangling and unphysical grouping, requiring complicated book-keeping. It is not at all clear how to understand and estimate the errors caused by the process of approximating lumps of particles together as single pseudo-particles, because individual lumps can take more or less arbitrary shapes and sizes.

Josh Barnes

Piet Hut

Introducción



VAX 11/780

Barnes y Hut realizaron una simulación con su método.

- CPU 5 MHz, 2 kB cache.
- Memoria 8 MB.

Introducción

VAX 11/780

- CPU 5 MHz, 2 kB cache.
- Memoria 8 MB.

Barnes y Hut realizaron una simulación con su método.

- 4096 cuerpos.

Introducción

VAX 11/780

- CPU 5 MHz, 2 kB cache.
- Memoria 8 MB.

Barnes y Hut realizaron una simulación con su método.

- 4096 cuerpos.
- 10 horas de CPU.

Funcionamiento

El algoritmo propuesto por Barnes y Hut consta de dos pasos fundamentales, la división recursiva del espacio y aproximaciones en la fuerza sobre un cuerpo.

Funcionamiento

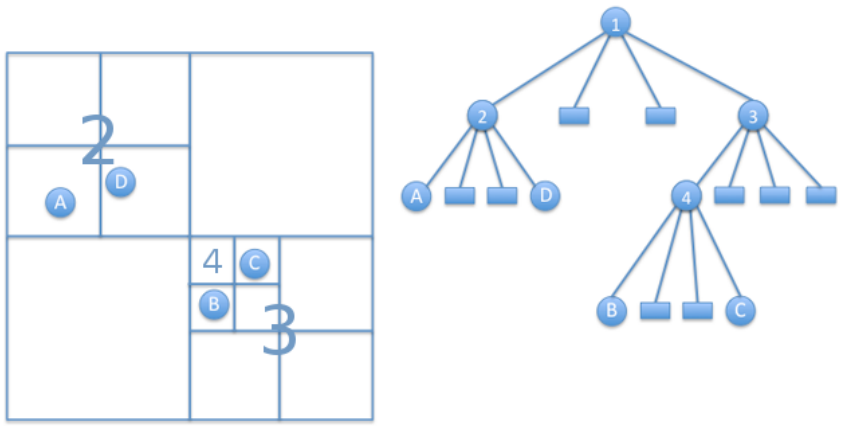
El algoritmo propuesto por Barnes y Hut consta de dos pasos fundamentales, la división recursiva del espacio y aproximaciones en la fuerza sobre un cuerpo.

1. División jerárquica del espacio. $\rightarrow$ Construcción de un árbol.

Funcionamiento

El algoritmo propuesto por Barnes y Hut consta de dos pasos fundamentales, la división recursiva del espacio y aproximaciones en la fuerza sobre un cuerpo.

1. División jerárquica del espacio. $\rightarrow$ Construcción de un árbol.

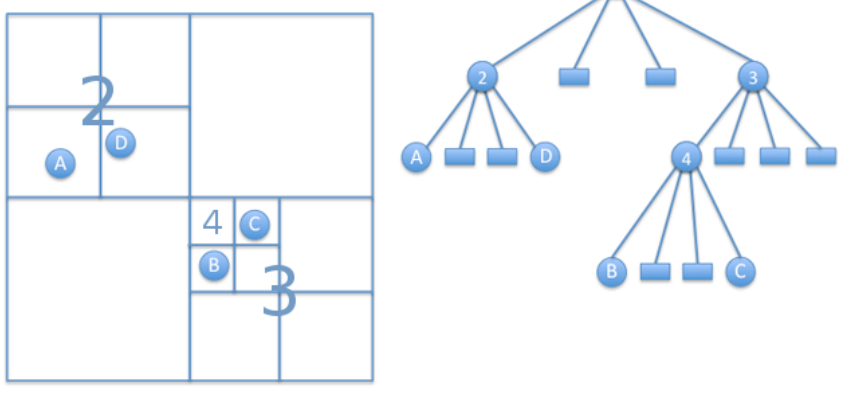


Árbol dos dimensional.

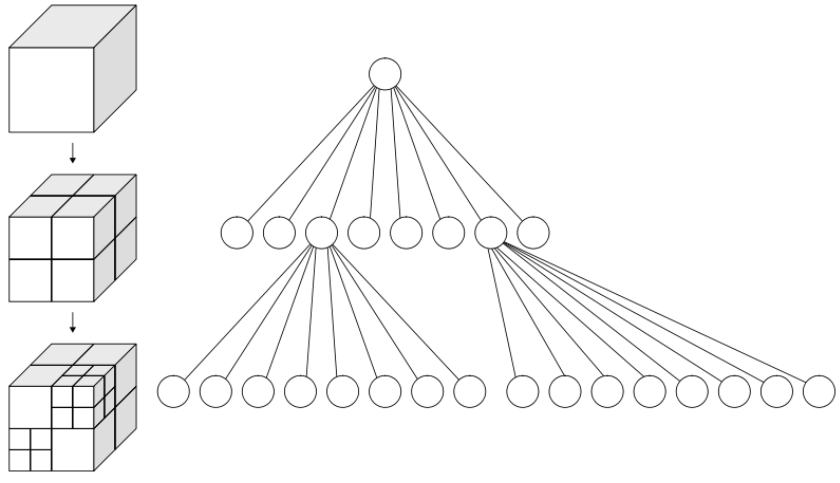
Funcionamiento

El algoritmo propuesto por Barnes y Hut consta de dos pasos fundamentales, la división recursiva del espacio y aproximaciones en la fuerza sobre un cuerpo.

1. División jerárquica del espacio. $\rightarrow$ Construcción de un árbol.



Árbol dos dimensional.



Árbol tridimensional.

2. Fuerza sobre un cuerpo.

El número de iteraciones se reduce al considerar centros de masa, y un coeficiente de precisión τ.

Funcionamiento

2. Fuerza sobre un cuerpo.

El número de iteraciones se reduce al considerar centros de masa, y un coeficiente de precisión τ.

- Cada caja tiene una longitud específica

Funcionamiento

2. Fuerza sobre un cuerpo.

El número de iteraciones se reduce al considerar centros de masa, y un coeficiente de precisión τ.

- Cada caja tiene una longitud específica
- Un centro de masa

2. Fuerza sobre un cuerpo.

El número de iteraciones se reduce al considerar centros de masa, y un coeficiente de precisión τ.

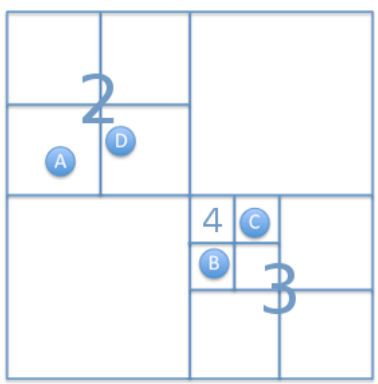
- Cada caja tiene una longitud específica
- Un centro de masa
- Y la masa contenida

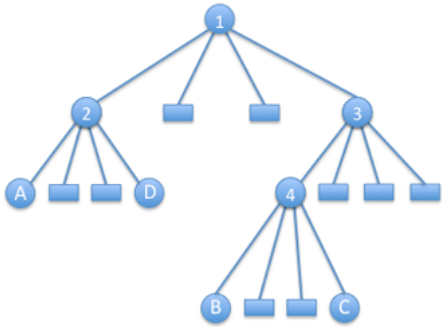
Funcionamiento

2. Fuerza sobre un cuerpo.

El número de iteraciones se reduce al considerar centros de masa, y un coeficiente de precisión τ.

- Cada caja tiene una longitud específica
- Un centro de masa
- Y la masa contenida





Funcionamiento

Pasos en el cálculo de la fuerza sobre un cuerpo. El proceso empieza desde el nodo superior.

1. Se calcula la distancia entre el centro de masa del nodo y el cuerpo.
2. Si la distancia es lo suficientemente grande, los cuerpos contenidos en el nodo se toman como un solo cuerpo.
3. De lo contrario el proceso se repite con los nodos inferiores.

Funcionamiento

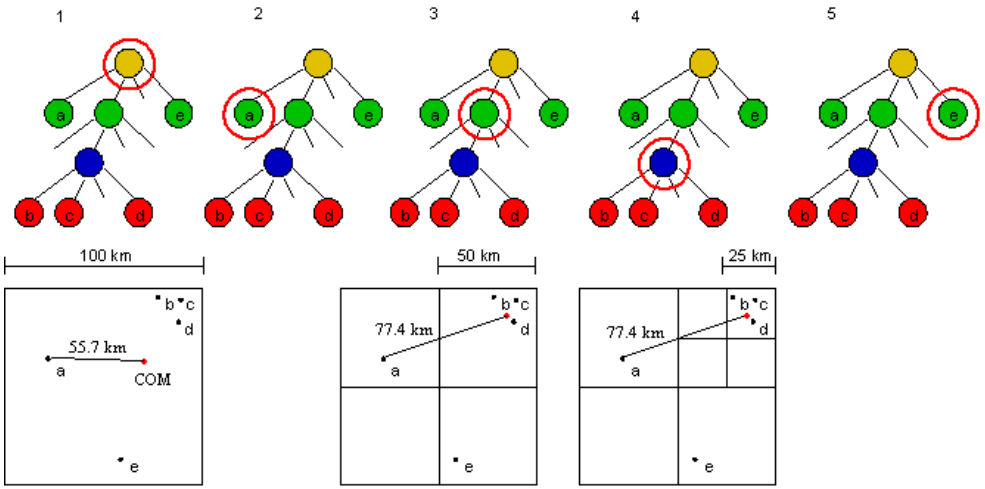
Para determinar si un nodo se encuentra lo suficientemente lejos de un cuerpo.

$$\tau' = \frac{s}{d} \tag{1}$$

Donde s corresponde con la longitud de la caja, y d con la distancia.
Se define un coeficiente de precisión τ para la simulación tal que:

- Si $\tau' > \tau$, el nodo es cercano.
- Si $\tau' \leq \tau$, el nodo es distante.

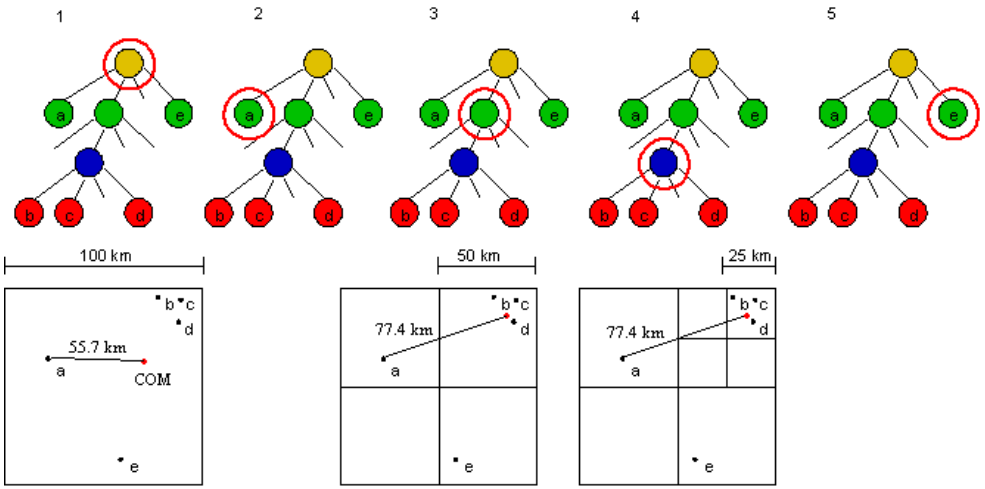
Funcionamiento



- Coeficiente de precisión.

$$\tau = \frac{size}{distance} = 0{,}5$$

Funcionamiento

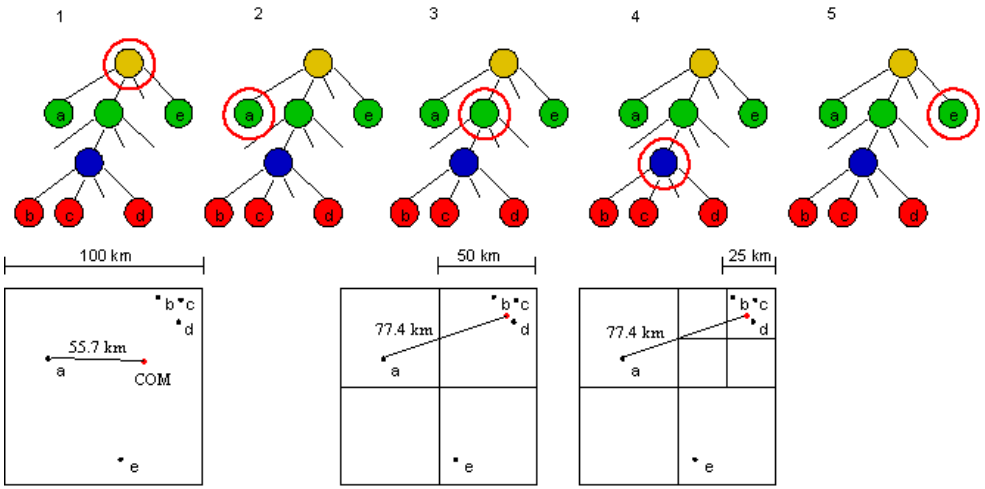


1. Nodo principal

- Coeficiente de precisión.

$$\tau = \frac{size}{distance} = 0{,}5$$

Funcionamiento



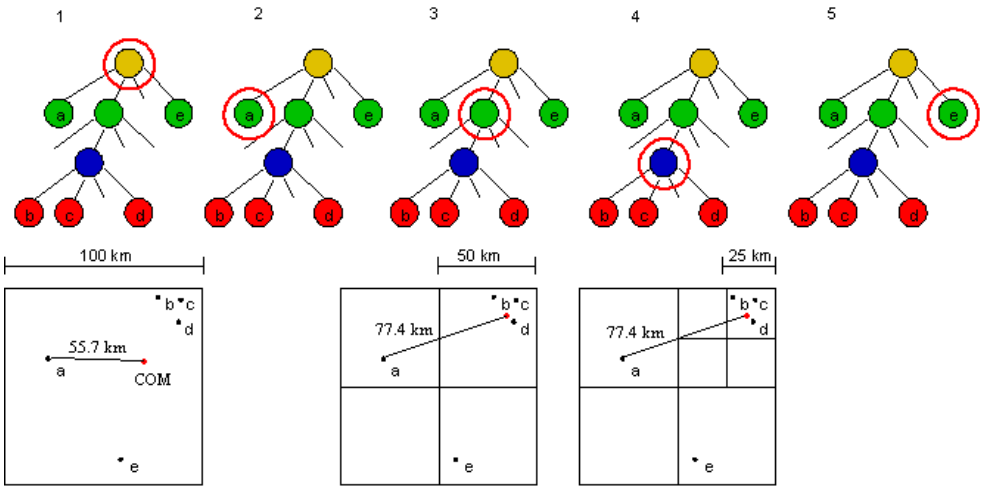
1. Nodo principal

$$\frac{s}{d} = \frac{100}{55{,}7} \approx 1{,}8 > \tau$$

- Coeficiente de precisión.

$$\tau = \frac{size}{distance} = 0{,}5$$

Funcionamiento



- Coeficiente de precisión.

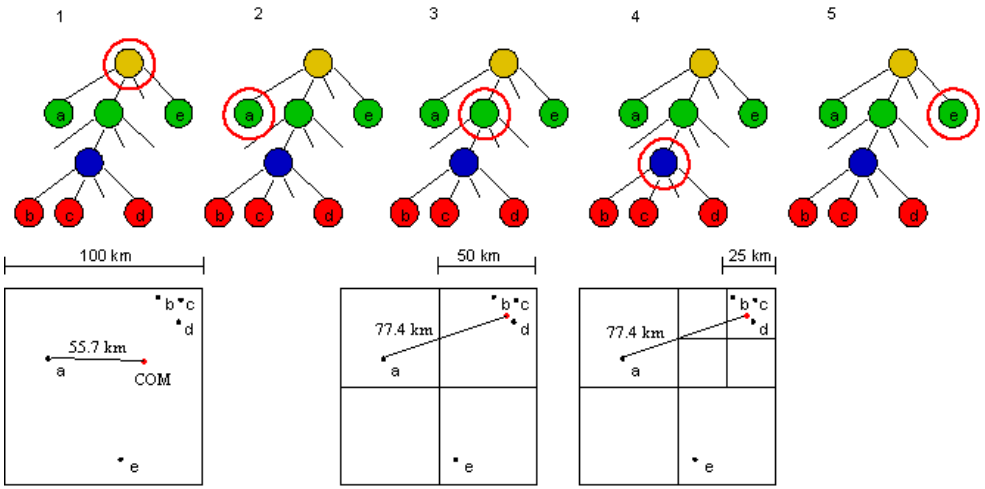
$$\tau = \frac{size}{distance} = 0{,}5$$

1. Nodo principal

$$\frac{s}{d} = \frac{100}{55{,}7} \approx 1{,}8 > \tau$$

2. Primer nodo

Funcionamiento



- Coeficiente de precisión.

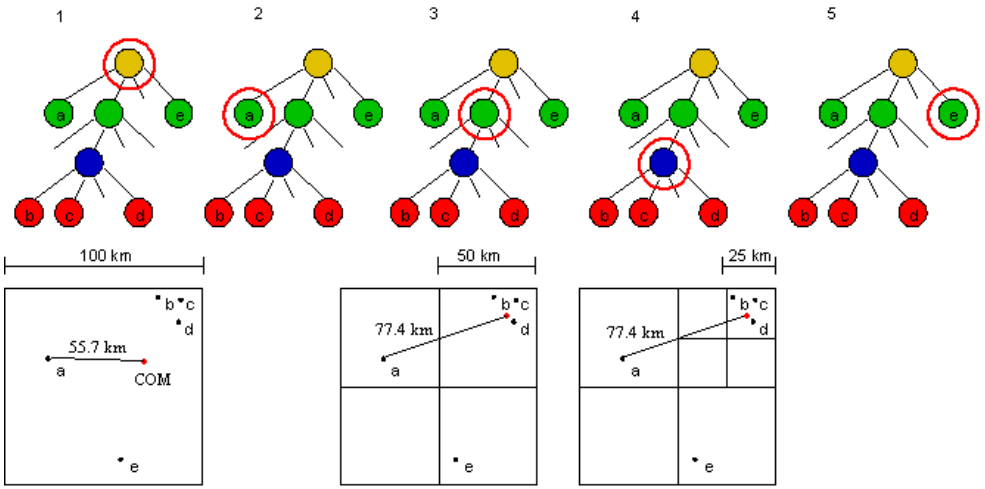
$$\tau = \frac{size}{distance} = 0{,}5$$

1. Nodo principal

$$\frac{s}{d} = \frac{100}{55{,}7} \approx 1{,}8 > \tau$$

2. Primer nodo
3. Segundo nodo

Funcionamiento



- Coeficiente de precisión.

$$\tau = \frac{size}{distance} = 0{,}5$$

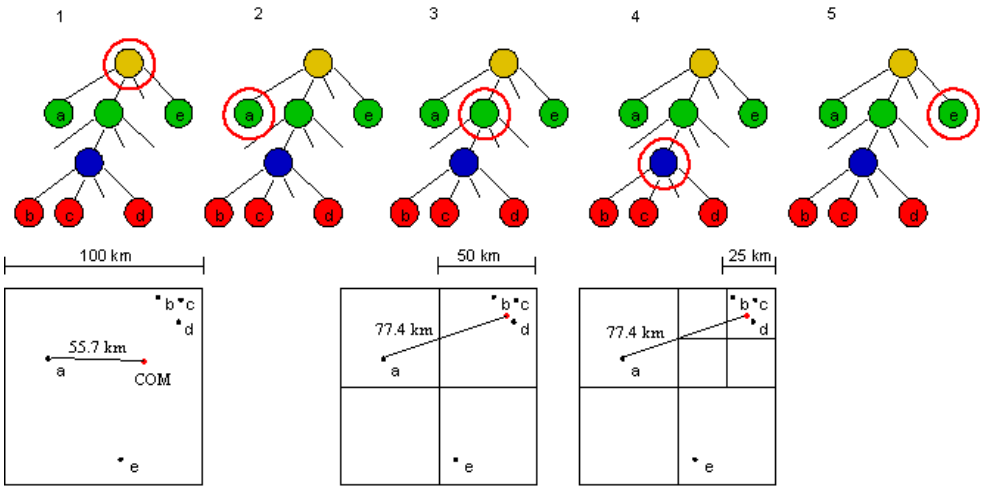
1. Nodo principal

$$\frac{s}{d} = \frac{100}{55{,}7} \approx 1{,}8 > \tau$$

2. Primer nodo
3. Segundo nodo

$$\frac{s}{d} = \frac{50}{77{,}4} \approx 0{,}6 > \tau$$

Funcionamiento



- Coeficiente de precisión.

$$\tau = \frac{size}{distance} = 0{,}5$$

1. Nodo principal

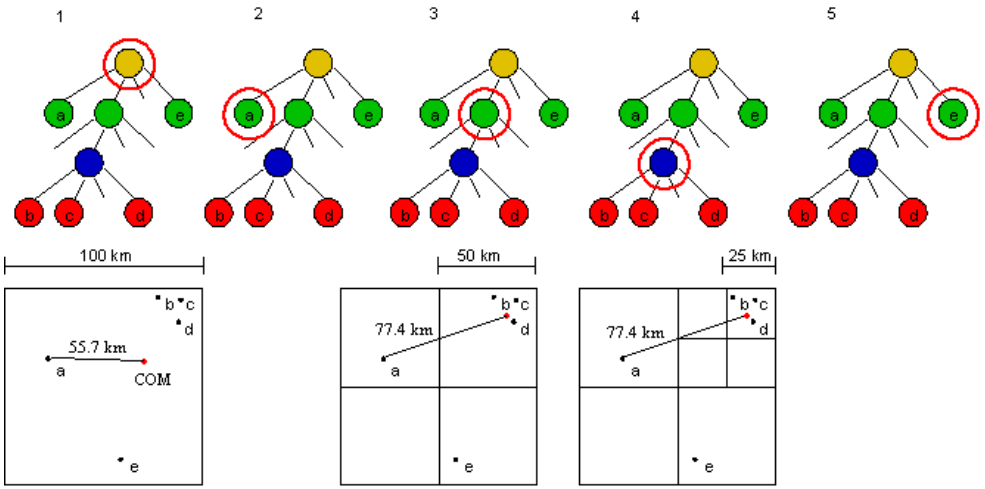
$$\frac{s}{d} = \frac{100}{55{,}7} \approx 1{,}8 > \tau$$

2. Primer nodo
3. Segundo nodo

$$\frac{s}{d} = \frac{50}{77{,}4} \approx 0{,}6 > \tau$$

4. Segundo nodo

Funcionamiento



- Coeficiente de precisión.

$$\tau = \frac{size}{distance} = 0{,}5$$

1. Nodo principal

$$\frac{s}{d} = \frac{100}{55{,}7} \approx 1{,}8 > \tau$$

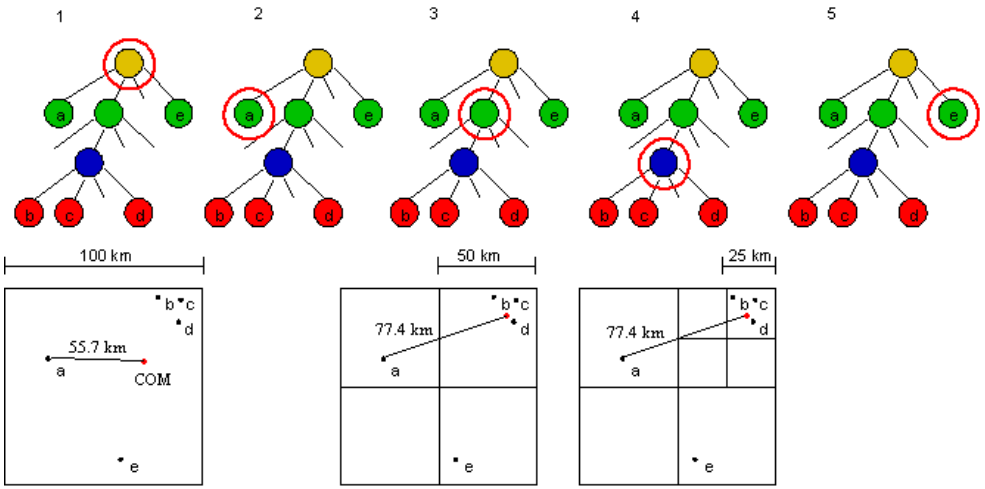
2. Primer nodo
3. Segundo nodo

$$\frac{s}{d} = \frac{50}{77{,}4} \approx 0{,}6 > \tau$$

4. Segundo nodo

$$\boxed{\frac{s}{d} = \frac{25}{77{,}4} \approx 0{,}3 < \tau}$$

Funcionamiento



- Coeficiente de precisión.

$$\tau = \frac{size}{distance} = 0{,}5$$

1. Nodo principal

$$\frac{s}{d} = \frac{100}{55{,}7} \approx 1{,}8 > \tau$$

2. Primer nodo
3. Segundo nodo

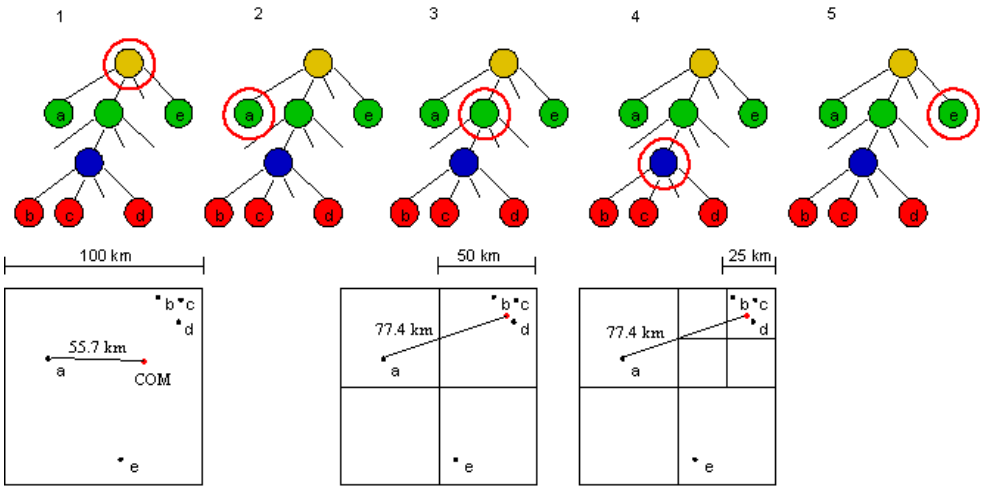
$$\frac{s}{d} = \frac{50}{77{,}4} \approx 0{,}6 > \tau$$

4. Segundo nodo

$$\boxed{\frac{s}{d} = \frac{25}{77{,}4} \approx 0{,}3 < \tau}$$

5. Cuarto nodo

Funcionamiento



- Coeficiente de precisión.

$$\tau = \frac{size}{distance} = 0{,}5$$

1. Nodo principal

$$\frac{s}{d} = \frac{100}{55{,}7} \approx 1{,}8 > \tau$$

2. Primer nodo
3. Segundo nodo

$$\frac{s}{d} = \frac{50}{77{,}4} \approx 0{,}6 > \tau$$

4. Segundo nodo

$$\boxed{\frac{s}{d} = \frac{25}{77{,}4} \approx 0{,}3 < \tau}$$

5. Cuarto nodo

Nodo externo, contribuye

Funcionamiento

La construcción del árbol se realiza para cada instante de tiempo.

Observación

Funcionamiento

La construcción del árbol se realiza para cada instante de tiempo.

Todas las cajas

Funcionamiento

La construcción del árbol se realiza para cada instante de tiempo.

Cajas con una partícula

Construcción de una simulación

1. Descripción del sistema.

Construcción de una simulación

1. Descripción del sistema.
2. Condiciones iniciales.

Construcción de una simulación

1. Descripción del sistema.
2. Condiciones iniciales.
3. Solución de las ecuaciones.

Construcción de una simulación

1. Descripción del sistema.
2. Condiciones iniciales.
3. Solución de las ecuaciones.
4. Visualización.

Usando la ley de gravitación universal:

$$\vec{F}_i = m_i \vec{a}_i = -\sum_{j \neq i}^{N} G \frac{m_i m_j}{|\vec{r}_i - \vec{r}_j|^3} (\vec{r}_i - \vec{r}_j) \tag{2}$$

Descripción del sistema

Usando la ley de gravitación universal:

$$\vec{F}_i = m_i \vec{a}_i = -\sum_{j \neq i}^{N} G \frac{m_i m_j}{|\vec{r}_i - \vec{r}_j|^3} (\vec{r}_i - \vec{r}_j) \tag{2}$$

es posible obtener las ecuaciones que describen la dinámica del sistema.

Descripción del sistema

Usando la ley de gravitación universal:

$$\vec{F}_i = m_i \vec{a}_i = -\sum_{j \neq i}^{N} G \frac{m_i m_j}{|\vec{r}_i - \vec{r}_j|^3} (\vec{r}_i - \vec{r}_j) \tag{2}$$

es posible obtener las ecuaciones que describen la dinámica del sistema.

$$\ddot{\vec{r}}_i = -\sum_{j \neq i}^{N} G \frac{m_j}{(|\vec{r}_i - \vec{r}_j|^2 + \epsilon^2)^{3/2}} (\vec{r}_i - \vec{r}_j) \tag{3}$$

donde ϵ corresponde con el *softening length*.

Condiciones iniciales

Suponiendo órbitas circulares y teniendo en cuenta la masa encerrada en las órbitas de menor tamaño:

$$v \approx \sqrt{\frac{GM(r)}{r}} \tag{4}$$

Condiciones iniciales

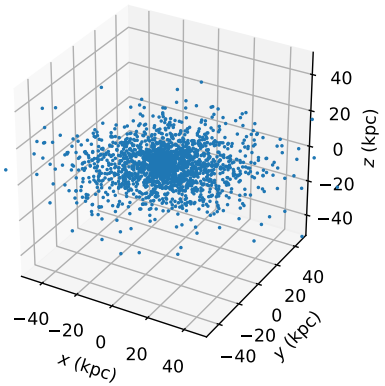
Suponiendo órbitas circulares y teniendo en cuenta la masa encerrada en las órbitas de menor tamaño:

$$v \approx \sqrt{\frac{GM(r)}{r}} \tag{4}$$

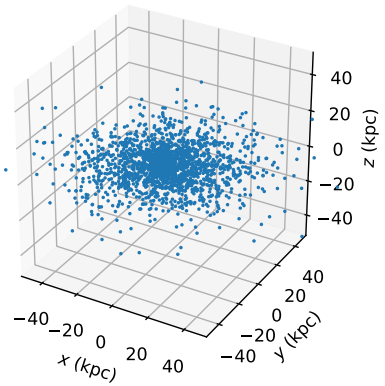
En coordenadas polares:

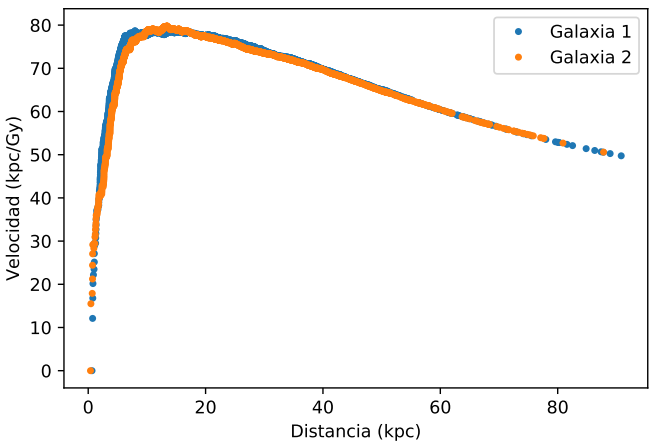
$$\begin{aligned} x = r\cos(\theta) \quad &\longrightarrow \quad \dot{x} = -r\sin(\theta) = -y \\ y = r\sin(\theta) \quad &\longrightarrow \quad \dot{y} = r\cos(\theta) = x \\ z = z \longrightarrow \dot{z} = \dot{z} \quad &??? \end{aligned} \tag{5}$$

Condiciones iniciales

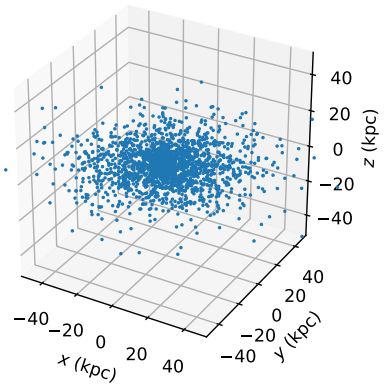


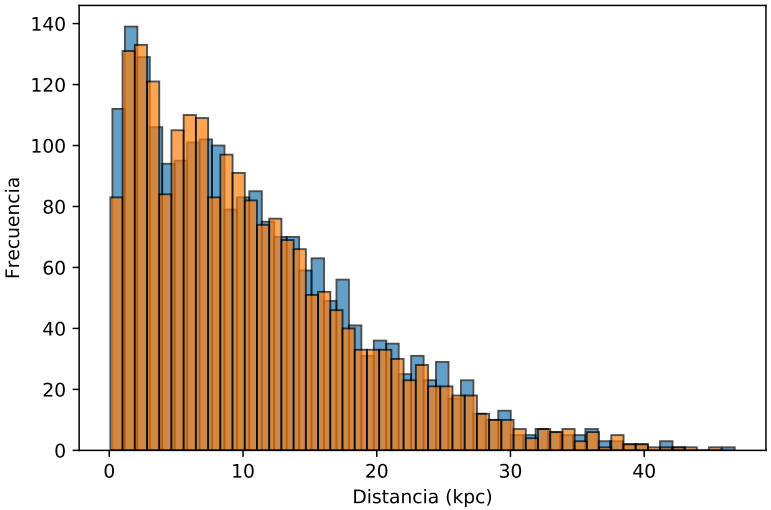
Condiciones iniciales

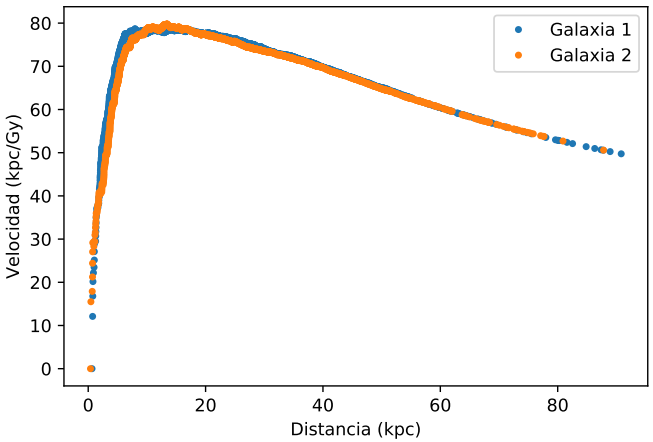


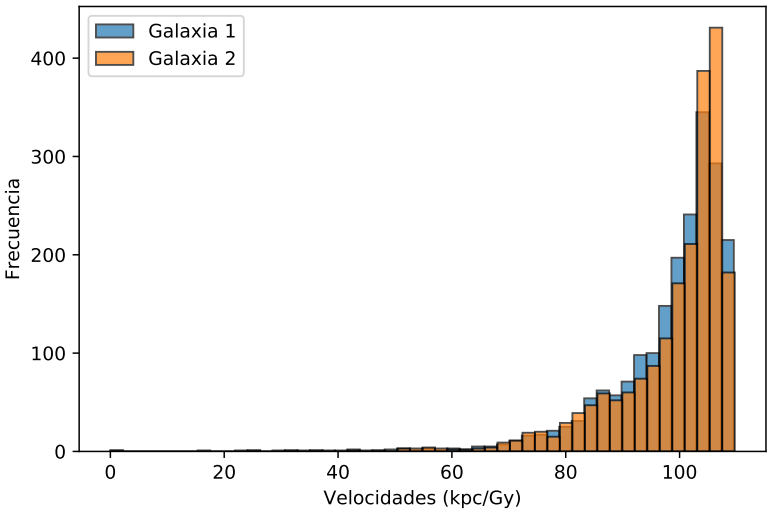


Condiciones iniciales

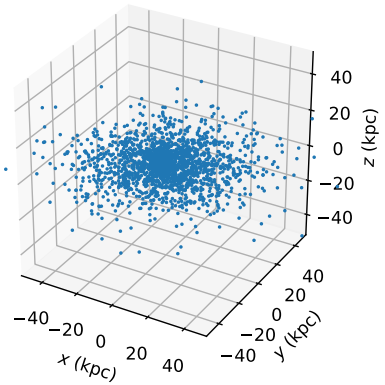


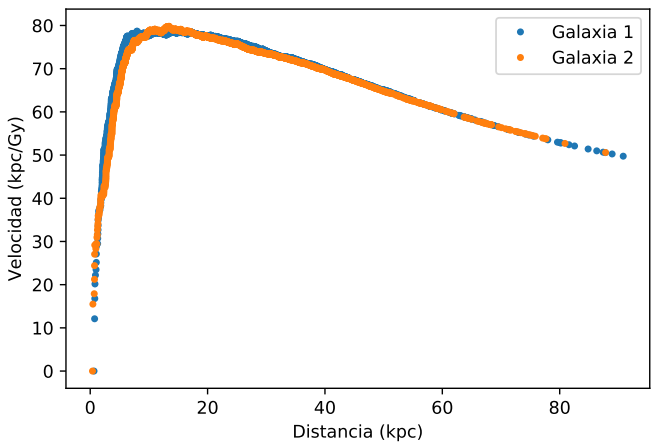






Condiciones iniciales





- $G = 44.97\ (10^7\ \mathrm{M}_\odot^{-1}\mathrm{kpc}^3\mathrm{Gy}^{-2})$
- Tamaño ≈ 80 (kpc)
- $m = 2.44\ (10^7 \mathrm{M}_\odot)$
- $N = 4096$
- $\epsilon = 0.055$

La solución numérica a las ecuaciones diferenciales se obtiene usando el método de *Leapfrog*, el cual avanza asincrónicamente en la posición y la velocidad.

$$\begin{aligned} v_{i+1/2} &= v_i + a_i \frac{\Delta t}{2} \\ x_{i+1} &= x_i + v_{i+1/2} \Delta t \\ v_{i+1} &= v_{i+1/2} + a_{i+1} \frac{\Delta t}{2} \end{aligned} \tag{6}$$

Visualización

La visualización de los resultados se realiza usando animaciones de las posiciones en función del tiempo. También se obtiene información de las curvas de rotación, distribución de masa y velocidades.

Implementación

Binding de `bruteforce.c` para Python.

C Programming Language

- `init.c`: configura las variables globales de la simulación (N, m, G, ϵ, τ), los arrays de posiciones y velocidades.
- `box.c`: contiene las funciones propias del árbol y sus cajas.
- `bruteforce.c`: resuelve las ecuaciones diferenciales, y genera archivos de datos.

Implementación

Python

- `core.py`: contiene la clase `Galaxy` y `Simulation`, las cuales generan condiciones iniciales y realizan la interfaz con C.

Ejecución de la simulación

```python
from core import *

N = 1096
M_T = 5e10/1e7
M = 2.0*M_T/N #two galaxies
G = 44.97
epsilon = 0.1

system, speeds = example(N, M, G, epsilon)

sim = simulation(M, G, epsilon, tolerance = 1.0, pos = system, speeds = speeds)
sim.start(0, 1.0, 0.01)

data = read_output()

"""
plotting
"""
import numpy as np
from glob import glob
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.animation import FuncAnimation


fig = plt.figure()
ax = fig.gca(projection='3d')
ax.set_aspect("equal")
# plot1 = ax.plot(system[0], system[1], system[2], "o", ms=0.5, c="g", alpha = 0.5)[0]
plot1 = ax.plot([], [], [], "o", ms=0.5, c="g", alpha = 0.5)[0]
plot2 = ax.plot([],[],[], "o", ms=0.5, c="b", alpha = 0.5)[0]
ax.set_xlabel("$x$ kpc")
ax.set_ylabel("$y$ kpc")
ax.set_zlabel("$z$ kpc")
N_first = int(0.5*N)

def update(i):
    temp = data[i]
    plot1.set_data(temp[:N_first,0], temp[:N_first,1])
    plot1.set_3d_properties(temp[:N_first, 2])
    plot2.set_data(temp[N_first:,0], temp[N_first:,1])
    plot2.set_3d_properties(temp[N_first:, 2])
```

Python script

1. `galaxy1, galaxy2, system, speeds = example(N, M, G)`

Ejecución de la simulación

```python
from core import *

N = 1096
M_T = 5e10/1e7
M = 2.0*M_T/N #two galaxies
G = 44.97
epsilon = 0.1

system, speeds = example(N, M, G, epsilon)

sim = simulation(M, G, epsilon, tolerance = 1.0, pos = system, speeds = speeds)
sim.start(0, 1.0, 0.01)

data = read_output()

"""
plotting
"""
import numpy as np
from glob import glob
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.animation import FuncAnimation


fig = plt.figure()
ax = fig.gca(projection='3d')
ax.set_aspect("equal")
# plot1 = ax.plot(system[0], system[1], system[2], "o", ms=0.5, c="g", alpha = 0.5)[0]
plot1 = ax.plot([], [], [], "o", ms=0.5, c="g", alpha = 0.5)[0]
plot2 = ax.plot([],[],[], "o", ms=0.5, c="b", alpha = 0.5)[0]
ax.set_xlabel("$x$ kpc")
ax.set_ylabel("$y$ kpc")
ax.set_zlabel("$z$ kpc")
N_first = int(0.5*N)

def update(i):
    temp = data[i]
    plot1.set_data(temp[:N_first,0], temp[:N_first,1])
    plot1.set_3d_properties(temp[:N_first, 2])
    plot2.set_data(temp[N_first:,0], temp[N_first:,1])
    plot2.set_3d_properties(temp[N_first:, 2])
```

Python script

1. `galaxy1, galaxy2, system, speeds = example(N, M, G)`
2. `sim = Simulation(M, G, system, speeds, epsilon, tolerance = 1.0, threads = -1)`

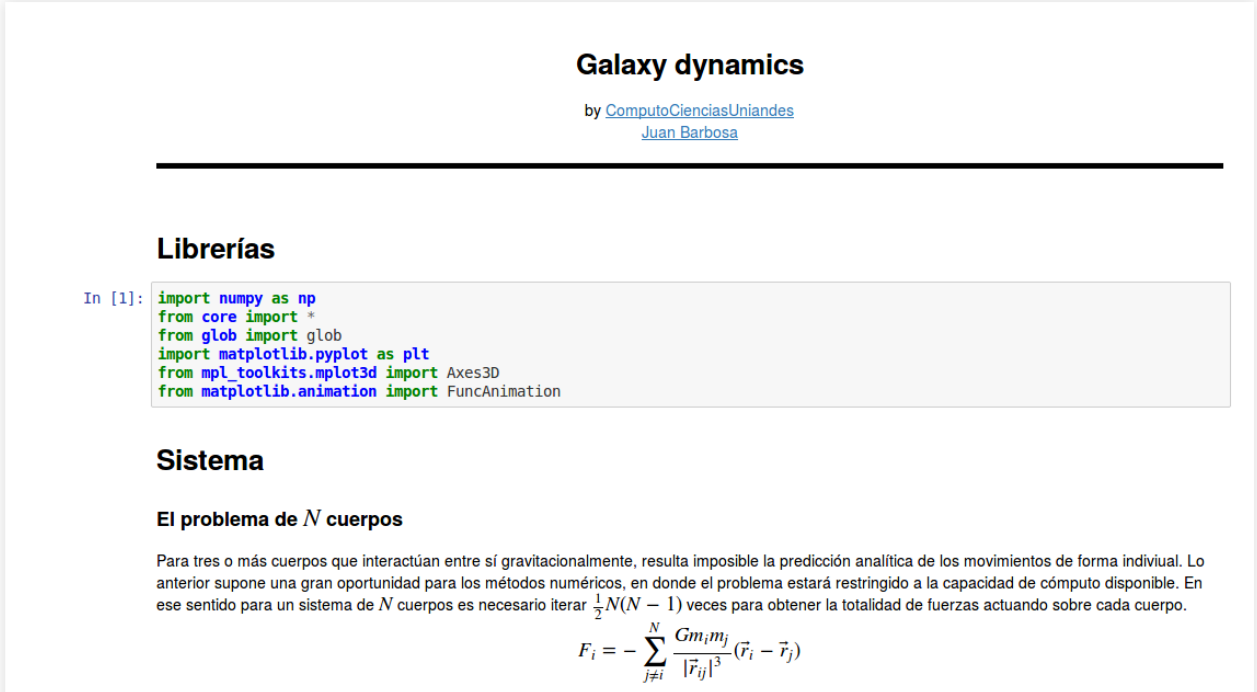
Ejecución de la simulación

```python
from core import *

N = 1096
M_T = 5e10/1e7
M = 2.0*M_T/N #two galaxies
G = 44.97
epsilon = 0.1

system, speeds = example(N, M, G, epsilon)

sim = simulation(M, G, epsilon, tolerance = 1.0, pos = system, speeds = speeds)
sim.start(0, 1.0, 0.01)

data = read_output()

"""
plotting
"""
import numpy as np
from glob import glob
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.animation import FuncAnimation


fig = plt.figure()
ax = fig.gca(projection='3d')
ax.set_aspect("equal")
# plot1 = ax.plot(system[0], system[1], system[2], "o", ms=0.5, c="g", alpha = 0.5)[0]
plot1 = ax.plot([], [], [], "o", ms=0.5, c="g", alpha = 0.5)[0]
plot2 = ax.plot([],[],[], "o", ms=0.5, c="b", alpha = 0.5)[0]
ax.set_xlabel("$x$ kpc")
ax.set_ylabel("$y$ kpc")
ax.set_zlabel("$z$ kpc")
N_first = int(0.5*N)

def update(i):
    temp = data[i]
    plot1.set_data(temp[:N_first,0], temp[:N_first,1])
    plot1.set_3d_properties(temp[:N_first, 2])
    plot2.set_data(temp[N_first:,0], temp[N_first:,1])
    plot2.set_3d_properties(temp[N_first:, 2])
```

Python script

1. `galaxy1, galaxy2, system, speeds = example(N, M, G)`
2. `sim = Simulation(M, G, system, speeds, epsilon, tolerance = 1.0, threads = -1)`
3. `sim.start(0.0, 1.0, 0.01)`

Ejecución de la simulación



Galaxy dynamics

by ComputoCienciasUniandes
Juan Barbosa

Librerías

```
In [1]: import numpy as np
        from core import *
        from glob import glob
        import matplotlib.pyplot as plt
        from mpl_toolkits.mplot3d import Axes3D
        from matplotlib.animation import FuncAnimation
```

Sistema

El problema de N cuerpos

Para tres o más cuerpos que interactúan entre sí gravitacionalmente, resulta imposible la predicción analítica de los movimientos de forma indiviual. Lo anterior supone una gran oportunidad para los métodos numéricos, en donde el problema estará restringido a la capacidad de cómputo disponible. En ese sentido para un sistema de N cuerpos es necesario iterar $\frac{1}{2}N(N-1)$ veces para obtener la totalidad de fuerzas actuando sobre cada cuerpo.

$$F_i = -\sum_{j \neq i}^{N} \frac{G m_i m_j}{|\vec{r}_{ij}|^3}(\vec{r}_i - \vec{r}_j)$$

Python Notebook

Disponible en:

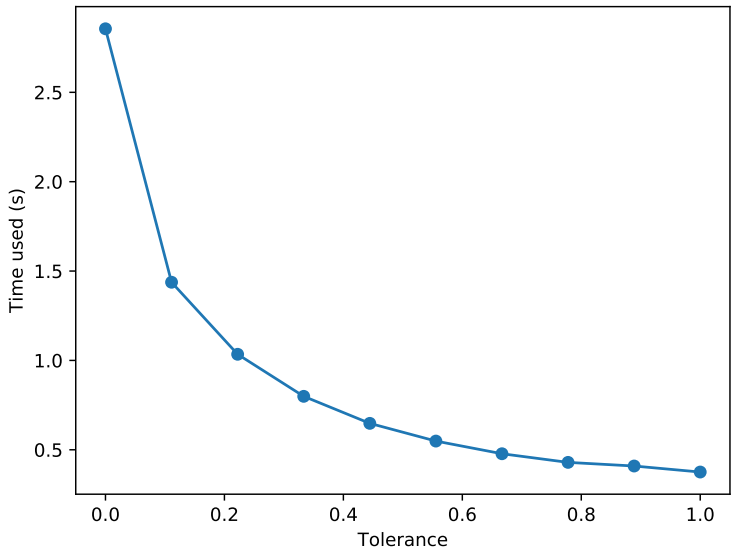
`https://github.com/ComputoCienciasUniandes/Demonstrations/tree/master/GalaxyDynamics`

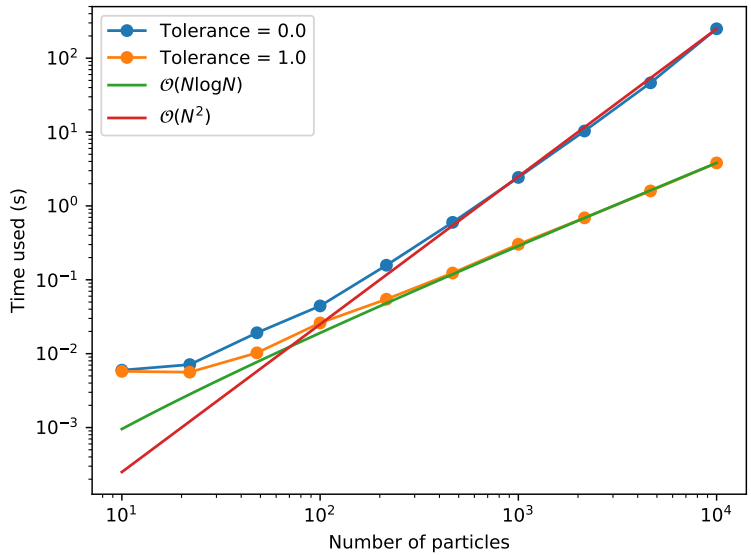
Resultados

Resultados

Resultados

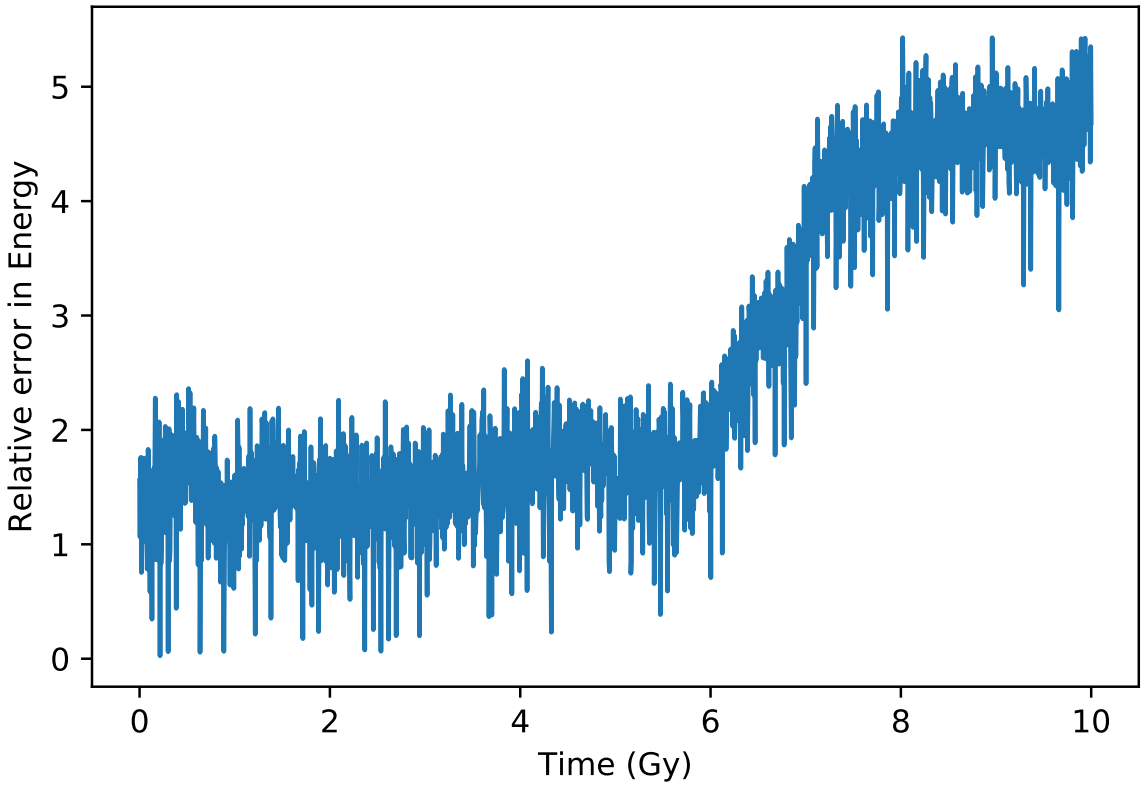
Efecto del coeficiente de precisión en el tiempo de cómputo.





Resultados

Conservación de la energía para $\tau = 1{,}0$.



Conclusiones

- El método de Barnes y Hut constituye una buena aproximación para las interacciones gravitacionales de N cuerpos.
- El órden del algoritmo es $\mathcal{O}(N \log N)$.
- El costo computacional disminuye exponencialmente con el valor de τ.