Astronomy & Astrophysics manuscript no. output
January 16, 2021

Simulação de N-Corpos: Formação de estrutura em larga escala

Pedro Fanha[1]

Departamento de Física e Astronomia da Faculdade de Ciências da Universidade do Porto
e-mail: pedro.fanha@fc.up.pt

January 16, 2021

ABSTRACT

Aims. Quis-se desenvolver um código para simular formação de estrutura em larga escala no Universo.
Methods. Escolheu-se implementar o método *particle-mesh*, que é relativamente fácil de implementar e suficientemente rápido para uso num computador pessoal. Utilizou-se um modelo matemático para as equações de movimento Newtoniano com expansão do Universo. Implementaram-se condições de fronteira periódicas. Utilizou-se um integrador *leapfrog* com *timesteps* globais constantes. Comparou-se os resultados deste código com os resultados do código GADGET-4.
Results. Fez-se uma simulação com 64^3 partículas numa grelha de tamanho 64^3, de $z_{inicial} = 23$ até $z_{final} = 0$, num cubo de comprimento $50\,h^{-1}$ Mpc, com parâmetros $\Omega_0 = 0.308$, $\Omega_\Lambda = 0.692$, $\Omega_k = 0.0$, $H_0 = 100\,h^{-1}$ Mpc e *timestep* constante igual a $1 \times 10^{-5}\ \mathrm{H_0^{-1}}$. Obtiveram-se resultados semelhantes ao do código GADGET-4 com as mesmas condições iniciais, mas verificou-se que a distribuição de matéria em $z = 0$ era mais uniforme na simulação de teste do que na simulação com o código GADGET-4.
Conclusions. O objetivo pretendido era demasiado ambicioso para o tempo que se tinha. Conseguiu-se desenvolver um código e correr uma simulação, como pretendido, no entanto, não foi possível fazer uma análise muito aprofundada. Quer-se continuar a trabalhar neste projeto, fazendo a tal análise, e melhorando o código, com o objetivo de aprender mais sobre a formação de estrutura e sobre os métodos existentes para correr este tipo de simulações.

1. Introdução

De acordo com os dados observacionais obtidos até ao momento, o Universo evoluiu a partir de pequenas perturbações aleatórias num espaço com densidade quase uniforme. É, portanto, interessante, e necessário, simular a evolução de um sistema com tais condições iniciais para compreender como se poderiam ter formado os *clusters* de galáxias, as galáxias, as estrelas e os planetas que vemos hoje.

A partir da era da inflação, entende-se que o Universo é dominado pelo que chamamos "matéria negra": matéria que só parece interagir através da força da gravidade, não colidindo nem emitindo radiação. Ao longo do tempo, a distribuição aproximadamente uniforme de matéria negra colapsou numa teia de filamentos, formando *halos* de matéria negra no processo - regiões de alta densidade onde se formam as galáxias, sistemas estelares, buracos negros, entre outros, em que a força da gravidade já não é o único fator envolvido, sendo necessário ter em conta outros efeitos relativos à matéria bariónica. A matéria bariónica (digase, normal) parece comportar-se da mesma forma em larga escala, sobrepondo-se a sua distribuição sobre a distribuição de matéria negra.

O modelo mais aceite para a descrição da evolução do Universo, ΛCDM, tem em conta, entre outros, a radiação cósmica de fundo, a estrutura em larga escala observada, as abundâncias dos vários elementos, e a expansão do Universo, que é caracterizada pela constante cosmológica Λ.

Para o estudo da formação de estrutura em larga escala, simulações de N-Corpos são normalmente utilizadas. A menos que se queira descrever a formação de estruturas mais pequenas, como galáxias, uma simulação de N-Corpos utilizando apenas a força da gravidade é suficiente. No modelo ΛCDM, assume-se como teoria da gravidade a teoria da relatividade geral, no entanto, a menos da expansão do Universo, é possível obter resultados suficientemente corretos utilizando as leis de Newton [Rigopoulos & Valkenburg (2014)].

Neste tipo de simulações, um sistema de corpos é evoluído no tempo de acordo com as leis de força, começando com uma distribuição normal aleatória semelhante à que se pensa ter existido na formação do nosso Universo. Para este fim, costuma-se aplicar a aproximação de Zel'dovich ou a teoria de perturbação Lagrangiana de 2ª ordem. Note-se que os "corpos" não representam partículas individuais de matéria negra, mas sim blocos de matéria agregada num só corpo, por razões de desempenho e porque tanto detalhe seria inútil.

O método mais simples para determinar as forças entre partículas é o método de soma direta ou partícula-partícula, onde as forças são calculadas entre cada par de partículas utilizando a 2ª lei de Newton. Este método tem a vantagem de ser muito simples de implementar, além de não incluir quaisquer aproximações, no entanto, tem a grande desvantagem de ter complexidade $O(N^2)$, o que o torna inviável para uma simulação deste tipo em que se costumam utilizar acima de um milhão de corpos, e para o caso em particular de correr num computador pessoal.

Outros métodos têm sido desenvolvidos como alternativa. Os mais comuns são o *particle-mesh* [Hockney & Eastwood (1981)], que tem complexidade $O(N) + O(G \log G)$, onde N é o número de corpos e G é o número de células da grelha; o método da árvore de Barnes-Hut [Barnes & Hut (1986)], que tem uma complexidade $O(N \log N)$, onde N é o número de corpos do sistema; e combinações destes dois métodos, como o *particle-particle particle-mesh* (P^3M) [Hockney & Eastwood (1981)] e o *tree particle-mesh* ($TreePM$) [Bagla (2002)].

Para comparação, considere-se $N = 10^6$. Para o método da soma direta, necessitar-se-ia de $(10^6)^2 = 10^{12}$ operações por *timestep* para calcular as forças entre as partículas. Para o método da árvore de Barnes-Hut, é necessário apenas $10^6 \times \log_{10} 10^6 = 6 \times 10^6$ operações, ou cerca de 1.6×10^5 vezes menos operações. Para o método *particle-mesh*, tipicamente escolhe-se

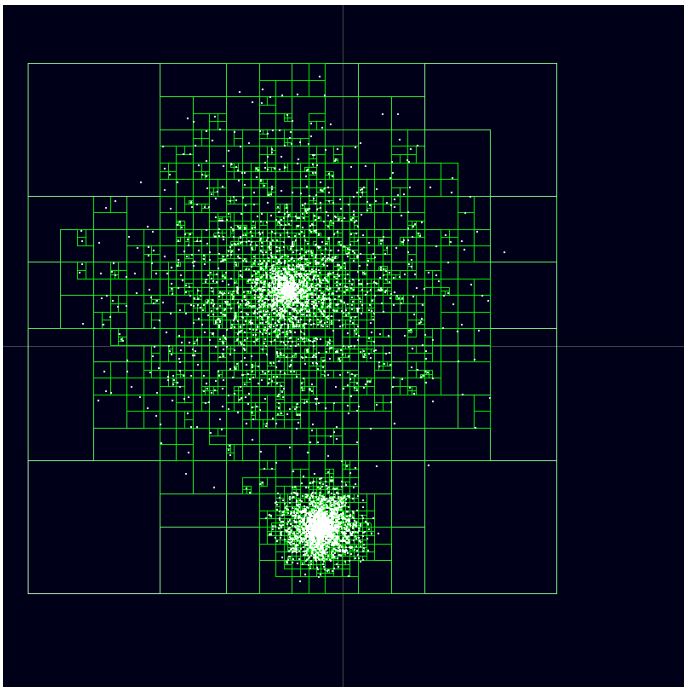

Fig. 1. Método de Barnes-Hut em 2D. O quadrado maior é sucessivamente dividido em quatro até cada quadrado conter no máximo uma partícula. Eclipse.sx, CC BY-SA 3.0 <https://creativecommons.org/licenses/by-sa/3.0>, via Wikimedia Commons (2009)

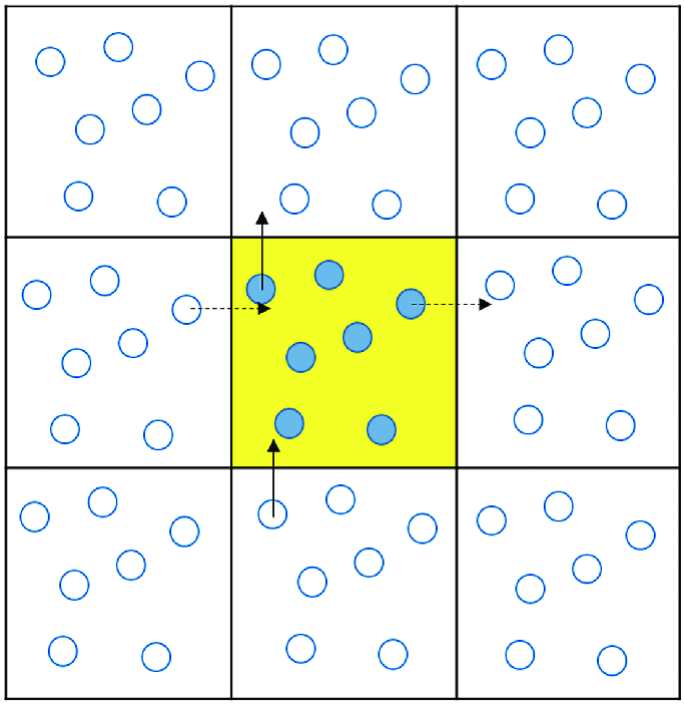

Fig. 2. Representação 2D de condições de fronteira periódicas. A célula a amarelo representa o cubo da nossa simulação. Os círculos representam partículas. Os cubos que envolvem o cubo central são as imagens periódicas dele. As setas ilustram o movimento de uma partícula ao sair do cubo de simulação. Molecular simulations in drug delivery: Opportunities & challenges Scientific Figure on ResearchGate (2018)

$G \geq N$, pelo que não podemos fazer o mesmo tipo de comparação, mas no melhor cenário (e pior resolução) obtém-se o mesmo número de operações por *timestep* do que no método da árvore de Barnes-Hut, apesar de ser possível melhorar o algoritmo através de esquemas de adequação do tamanho da grelha à densidade das diferentes regiões.

O método da árvore de Barnes-Hut reduz a complexidade relativamente ao método partícula-partícula gerando uma *octree*, uma árvore binária tridimensional, onde as partículas, ou corpos, são agrupadas em cubos de tamanho cada vez menor até que cada partícula esteja colocada sozinha no seu cubo. As forças são então calculadas entre esses grupos de partículas, começando no topo da árvore, e baseando-se num critério de abertura que decide se um grupo de partículas está longe o suficiente para ser considerado um corpo único ou se se deve calcular a força para cada uma das partículas do grupo. A figura 1 exemplifica o método.

O método *particle-mesh*, por outro lado, começa por gerar uma grelha fixa no espaço. A cada *timestep*, a densidade do campo é calculada apenas nos pontos fixos da grelha através de um esquema de interpolação e da ponderação das massas. Estas densidades são então utilizadas para resolver a forma diferencial da equação de Poisson para o potencial gravítico, donde se obtém o potencial em toda a grelha. Utilizando o mesmo esquema de interpolação, pode-se então aproximar a força sentida por cada corpo em todo o espaço. Este método, infelizmente, tem a desvantagem de ter resolução baixa em regiões de alta densidade, a menos que se use uma grelha de tamanho ajustável à região.

Todos os três métodos sofrem do mesmo problema de *softening* da força. De modo a evitar erros numéricos no cálculo da força entre dois corpos próximos causados pela discretização do tempo, introduz-se uma constante ϵ na distância r que aparece na 2ª lei de Newton. No entanto, isso significa perder alguma exatidão nos resultados obtidos. No método *particle-mesh*, não é utilizada a 2ª lei de Newton, mas este efeito de softening aparece na mesma, sendo ϵ igual ao tamanho de uma célula da grelha.

Finalmente, as forças são utilizadas para avançar as partículas no espaço e no tempo utilizando diferentes esquemas de integração. O integrador *leapfrog* de 2ª ordem é o mais usado, sendo preferido a algoritmos de ordem mais elevada, e.g. Runge-Kutta, visto ser extremamente simples de implementar e ser *time-reversible* para *timesteps* fixos. Para *timesteps* variáveis, perde-se essa propriedade, mas é possível torná-lo suficientemente preciso [Quinn et al. (1997)]. Esta propriedade é importante porque garante que quantidades conservadas mantêm-se conservadas ao longo de uma simulação longa, minimizando-se assim erros numéricos que se vão acumulando devido ao esquema de integração.

Além disto, é necessário ter ainda em conta o facto de, numa simulação, termos um número finito de corpos. Na realidade, qualquer partícula está rodeada por outras partículas, e isso afeta o seu movimento. De modo a remediar este problema, costumam-se implementar condições de fronteira periódicas colocando todos os corpos num cubo, replicando esse cubo em todo o espaço envolvente. Assim, garante-se que todas as partículas estão sobre um mesmo efeito da gravidade. No caso de as partículas saírem do cubo, voltam a entrar pelo lado oposto. A figura 2 ilustra o método.

O objetivo deste projeto foi o de correr uma simulação da formação de estrutura em larga escala com matéria negra e analisá-la, no entanto, isso provou demorar muito mais tempo e requerer muito mais conhecimento do que esperava. Foquei-me principalmente em tentar compreender e implementar os métodos para obter uma simulação o mais fiél possível ao real (comparando

com o resultado do código GADGET-4) em tempo útil, mas mesmo isso levou demasiado tempo.

Na secção 2, descreve-se o modelo matemático utilizado, assim como os métodos utilizados para calcular as forças e integrar as equações de movimento.

Na secção 3, faz-se uma avaliação muito breve da qualidade dos resultados.

Na secção 4, conclui-se sobre o trabalho feito e sobre o que pretendo fazer no futuro.

2. Método

Estando a aprender novos métodos e a tentar criar uma simulação correta, ajuda bastante ter algo com que comparar, algo de qual eu esteja confiante estar certo. O código GADGET-4 [Springel et al. (2020)] tem muitos anos de vida, é robusto o suficiente para gerar muitos tipos de simulações astrofísicas diferentes e está bem documentado. Uma versão anterior deste código foi utilizada, por exemplo, para correr a primeira simulação Millenium [Springel et al. (2005)]. Utilizei este código para gerar as condições iniciais pela teoria de perturbação Lagrangiana de 2ª ordem, para validar os resultados do meu código e para tirar partido de ferramentas *third-party* para o GADGET, como o Gadget Viewer[1], um programa para visualizar os *snapshots* da simulação em 3D, guardando os resultados da minha própria simulação em formato HDF5 compatível com o GADGET.

Na introdução, tornei claro que tentar correr uma simulação com um número minimamente razoável de partículas seria inviável com o método de soma direta da força. Isto significa que eu não segui de todo o método descrito em Mário João P. F. G. Monteiro (2019).

Implementei ambos o método da árvore de Barnes-Hut e o método *particle-mesh*. Tentei escolher sempre os métodos mais rápidos de implementar devido à falta de tempo, tentando na mesma obter resultados minimamente bons com o código. Para o método da árvore de Barnes-Hut, implementar condições de fronteira periódicas significaria aprender novas técnicas que, a meu ver, iriam demorar muito tempo, pelo que deixei este método de parte por agora. O método *particle-mesh* é algo mais complicado, mas foi relativamente fácil de encontrar informação detalhada de como o implementar, incluindo toda a teoria para um Universo em expansão. Além disso, implementar condições de fronteira periódicas neste método é extremamente simples.

O modelo matemático utilizado é semelhante ao apresentado em Klypin & Holtzman (1997).

2.1. Modelo matemático das equações de movimento

Tenham-se $\mathbf{r}$ as coordenadas físicas e $\mathbf{u} = \frac{d\mathbf{r}}{dt} = \dot{\mathbf{r}}$. A distância entre dois corpos que se afastam devido à expansão do espaço é dada por:

$$d(t) = a(t)d_0 \tag{1}$$

donde sai que $\mathbf{x}$, as coordenadas de um corpo relativamente a um referencial que se move com o espaço, é:

$$\mathbf{x} = \frac{\mathbf{r}}{a} \tag{2}$$

onde a é o fator de escala do Universo. Adotou-se a convenção típica de $a = 1.0$ para o instante presente e $a \to 0$ para o início do Universo. Para facilitar comparações de tempo, utiliza-se o redshift, z, que se relaciona com a da seguinte forma:

$$a = \frac{1}{1+z} \tag{3}$$

A velocidade, $\mathbf{v}$, relativamente a este mesmo referencial móvel, pode ser determinada notando que:

$$\dot{\mathbf{x}} = \frac{\dot{\mathbf{r}}a - \mathbf{r}\dot{a}}{a^2} = \frac{1}{a}(\mathbf{u} - \mathbf{r}H) = \frac{1}{a}\mathbf{v} \tag{4}$$

onde, por definição, $H = \frac{\dot{a}}{a}$, o parâmetro de Hubble.

O potencial gravítico para um Universo com $\Lambda = 3H_0^2\Omega_{\Lambda,0} \neq 0$ é dado por:

$$\nabla^2\Phi = 4\pi G\rho_{tot} - \Lambda \tag{5}$$

que deriva das equações de campo de Einstein.

É razoável transformar esta equação, falando de perturbação no potencial e na densidade relativamente ao potencial e à densidade média do Universo. Através de teoria de perturbação, e incluindo a expansão do Universo, pode-se obter a equação de Poisson na forma:

$$\nabla^2\phi = 4\pi G\rho_m a^2\delta \tag{6}$$

onde:

$$\delta = \frac{\rho - \overline{\rho}}{\overline{\rho}} \tag{7}$$

é a chamada *overdensity*, e ρ_m é a densidade de massa média em a.

Daqui por diante, toma-se *subscript* 0 nas variáveis como indicação de que são valores dados para $z = 0$.

Da equação 1, é fácil verificar que:

$$\rho(a) = \frac{\rho_{m,0}}{a^3} \tag{8}$$

e, além disso, tem-se:

$$\Omega_{m,0} = \frac{\rho_{m,0}}{\rho_{c,0}} = \frac{8\pi G}{3H_0^2}\rho_{m,0} \tag{9}$$

pelo que a equação de Poisson pode ainda ser reescrita na forma:

$$\nabla^2\phi = 4\pi G\frac{\Omega_{m,0}\rho_{c,0}}{a^3}a^2\delta = \frac{3H_0^2}{2a}\Omega_{m,0}\delta \tag{10}$$

Introduz-se ainda o momento linear relativista, $\mathbf{p} = a\mathbf{v}$, como sendo a velocidade física (i.e. incluindo a expansão do Universo). Apesar de misturarmos momento relativista com coordenadas fixas relativas ao espaço na integração do movimento, esta escolha de variável simplifica as equações.

Da equação de Poisson e destas definições, sai que:

$$\frac{d\mathbf{p}}{da} = -\frac{\nabla\phi}{\dot{a}} \tag{11}$$

$$\frac{d\mathbf{x}}{da} = \frac{\mathbf{p}}{\dot{a}a^2} \tag{12}$$

É útil escolher novas unidades base, quer porque podem ajudar a simplificar as equações como também eliminam potenciais

[1] Disponível em: https://github.com/jchelly/gadgetviewer

problemas de *overflow* no código se as grandezas forem muito grandes. Escolheram-se as seguintes unidades base:

$$r_b = \frac{L}{G} \quad (13)$$
$$t_b = H_0^{-1} \quad (14)$$
$$v_b = \frac{r_b}{t_b} \quad (15)$$
$$\rho_b = \rho_m \quad (16)$$
$$\phi_b = v_b^2 \quad (17)$$

onde L é o comprimento do lado do cubo da simulação, e G o número de divisões de cada lado da grelha. Isto implica que uma unidade de distância equivale ao comprimento do lado de uma célula da grelha.

Assim, obtêm-se as variáveis:

$$\tilde{\mathbf{x}} = \frac{\mathbf{x}}{r_b} \quad (18)$$
$$\tilde{\mathbf{p}} = \frac{\mathbf{p}}{v_b} \quad (19)$$
$$\tilde{\phi} = \frac{\phi}{\phi_b} \quad (20)$$

Neste caso, tem-se, unicamente por substituição:

$$\frac{d\tilde{\mathbf{p}}}{da} = \frac{-H_0}{\dot{a}}\tilde{\nabla}\tilde{\phi} \quad (21)$$
$$\frac{d\tilde{\mathbf{x}}}{da} = \frac{H_0}{\dot{a}}\frac{\tilde{\mathbf{p}}}{a^2} \quad (22)$$
$$\tilde{\nabla}^2\tilde{\phi} = \frac{3}{2}\frac{\Omega_{m,0}}{a}\tilde{\delta} \quad (23)$$

onde $\tilde{\delta} = \tilde{\rho} - 1$, pois a unidade base é a densidade de massa média (para qualquer que seja o a), e $\dot{a}$ pode ser obtido pela definição de parâmetro de Hubble e pela equação:

$$H^2(a) = \left(\frac{\dot{a}}{a}\right)^2 = \frac{8\pi G}{3}\rho - \frac{\Omega_{k,0}}{a^2} + \frac{\Lambda}{3} \quad (24)$$

que deriva das equações de Friedmann, obtendo-se:

$$\frac{d\tilde{\mathbf{p}}}{da} = -f(a)\tilde{\nabla}\tilde{\phi} \quad (25)$$
$$\frac{d\tilde{\mathbf{x}}}{da} = f(a)\frac{\tilde{\mathbf{p}}}{a^2} \quad (26)$$
$$(27)$$

com:

$$f(a) = \left[a^{-1}\left(\Omega_{m,0} + \Omega_{k,0}a + \Omega_{\Lambda,0}a^3\right)\right]^{-1/2} \quad (28)$$

2.2. O método particle-mesh para o cálculo da força

O cálculo da aceleração de uma partícular requer resolver a equação de Poisson (23), o que, por sua vez, requer o cálculo da densidade na grelha. A densidade é determinada por um algoritmo de interpolação. Utiliza-se o método *Cloud-In-Cell* (CIC) que supõe partículas cúbicas de densidade uniforme. Toma-se o tamanho do lado da partícula como sendo o tamanho do lado de uma célula, que é igual a 1 na unidade base escolhida. A densidade na grelha é então calculada iterando sobre cada partícula e interpolando linearmente a massa da mesma para a posição de oito das células mais próximas.

A massa de cada partícula é calculada como uma fração da densidade média do Universo:

$$m = \rho_m L^3/N \quad (29)$$

onde L^3/N é a porção de volume ocupada por cada partícula.

A equação de Poisson (23) pode ser resolvida por FFT facilmente, pois no espaço de Fourier, tem-se:

$$\tilde{\phi}(\mathbf{k}) = G(\mathbf{k})\tilde{\delta}(\mathbf{k}) \quad (30)$$

em que, neste caso:

$$G(\mathbf{k}) = -\frac{3\Omega_0}{8a}\left[\sin^2\left(\frac{k_x}{2}\right) + \sin^2\left(\frac{k_y}{2}\right) + \sin^2\left(\frac{k_z}{2}\right)\right]^{-1} \quad (31)$$

e:

$$k_x = \frac{2\pi l}{\tilde{L}} \quad (32)$$
$$k_y = \frac{2\pi m}{\tilde{L}} \quad (33)$$
$$k_z = \frac{2\pi n}{\tilde{L}} \quad (34)$$

Primeiro, é feita a transformação da densidade $\tilde{\delta}(\mathbf{r})$ para o espaço de Fourier. Depois, multiplica-se pela função de Green para cada ponto na grelha, e finalmente transfoma-se para o espaço real, obtendo-se o potencial.

Na componente $l = m = n = 0$, é típico definir-se manualmente $\tilde{\phi} = 0$.

A aceleração em cada célula da grelha é obtida a partir do potencial pela aproximação:

$$\tilde{g}^x_{i,j,k} = -(\tilde{\phi}_{i+1,j,k} - \tilde{\phi}_{i-1,j,k})/2 \quad (35)$$
$$\tilde{g}^y_{i,j,k} = -(\tilde{\phi}_{i,j+1,k} - \tilde{\phi}_{i,j-1,k})/2 \quad (36)$$
$$\tilde{g}^z_{i,j,k} = -(\tilde{\phi}_{i,j,k+1} - \tilde{\phi}_{i,j,k-1})/2 \quad (37)$$

sabendo que $\tilde{g} = -\tilde{\nabla}\tilde{\phi}$, e finalmente utiliza-se o mesmo método de interpolação, CIC, mas no sentido inverso, para obter a aceleração na posição de cada partícula no espaço.

Neste método, é bastante simples implementar condições de fronteira periódicas. Basta garantir que para cada célula, ao acessar a célula com coordenadas $i+1$, $j+1$, $k+1$, etc., se faz *wrap*. Ou seja, se $i + 1 > G^{1/3}$, então $i + 1 \rightarrow 1$; se $i - 1 < 1$, então $i - 1 \rightarrow G^{1/3}$, e por assim adiante. Além disso, deve-se aplicar a mesma técnica à posição das partículas no fim de cada *timestep*.

2.3. Integrador

A integração das equações de movimento é feita com o integrador *leapfrog* com *timesteps* constantes:

$$\tilde{\mathbf{p}}_{n+1/2} = \tilde{\mathbf{p}}_{n-1/2} + f(a_n)\tilde{\mathbf{g}}_n\Delta a \quad (38)$$
$$\tilde{\mathbf{x}}_{n+1} = \tilde{\mathbf{x}}_n + a^{-2}_{n+1/2}f(a_{n+1/2})\tilde{\mathbf{p}}_{n+1/2}\Delta a \quad (39)$$

Estas equações derivam das equações (26), (27).

Como este integrador não é *self-starting* - é necessário calcular o momento no instante $a = \frac{1}{2}\Delta a$, - utiliza-se o método de Euler para começar a integração, passando depois a utilizar as equações acima.

Foram utilizados *timesteps* globais e constantes pois um esquema de *timesteps* por partícula requere um algoritmo mais sofisticado para sincronizar as partículas e para garantir que não há perda de precisão numérica, algo que era impossível no tempo que eu tinha. No entanto, dependendo dos resultados de testes quanto ao efeito na precisão e eficiência do código, *timesteps* variáveis poderão ser implementados no futuro.

Parâmetro	Valor
N	64^3
L	$50\,h^{-1}$ Mpc
G	64^3
z inicial	23
z final	0
Ω_0	0.308
Ω_Λ	0.692
Ω_k	0.0
H_0	$100\,h^{-1}$ Mpc
Timestep (meu código)	$1 \times 10^{-5}\ \mathrm{H}_0^{-1}$
Timestep (máx, GADGET-4)	0.01
Timestep (mín, GADGET-4)	0.0

Table 1. Parâmetros utilizados para a 1ª simulação, tanto com o meu código como com o código GADGET-4.

3. Validação do código

Como teste inicial, correu-se uma simulação com os parâmetros tabelados em 1. Não é suposto estes parâmetros representarem, de todo, um sistema físico real. Os resultados apresentam-se nas figuras 3, 4 (plano XY), 5, 6 (plano YZ) e 7, 8 (plano XZ) para $z = 0$. Por um lado, a estrutura em ambas as simulações são bastante parecidas. Por outro, nota-se que há uma pequena diferença nas distribuições de matéria. Há maior agregação de matéria na simulação com o código GADGET-4 do que com o meu. Isto torna-se claro quando se analisa o espetro de potência da matéria, que se apresenta na figura 9. Por a matéria estar mais dispersa, a quantidade de matéria que é provável encontrar em qualquer escala é sempre inferior à que se obtém com o código GADGET-4.

As razões para isto podem ser várias. A minha principal suspeita é que não esteja a ler algum parâmetro das condições iniciais corretamente. Outra possibilidade é a de ter algum erro em algum cálculo, mas suponho que as diferenças entre as simulações seriam muito maiores nesse caso. O *timestep* utilizado também é uma possível causa de desvios no resultado. Foram feitos alguns testes rápidos, comparando o *timestep* utilizado com o *timestep* sugerido por vários critérios publicados na literatura, que indicam que o *timestep* ideal durante a maior parte da simulação é da ordem de 10^{-6}, 10 vezes menor do que o utilizado. É de notar também que o algoritmo utilizado pelo GADGET-4 é o TreePM, que consegue gerar simulações com melhor resolução (mas, à partida, isso só se notará em escalas mais pequenas), e utiliza *timesteps* variáveis. No entanto, não tive tempo de fazer uma análise mais completa. Só com mais tempo poderia ter resolvido este problema e ter feito uma análise mais aprofundada, tentando validar as escolhas feitas e verificar onde posso melhorar. Além disso, o código precisa de ser bastante otimizado. Apesar de estar a correr na GPU, continua a demorar mais tempo do que o GADGET-4 na CPU. Não tendo experiência de programação na GPU, e não tendo mais tempo, certamente o problema está na má gestão de recursos.

4. Conclusões

O meu objetivo para este projeto foi, claramente, muito ambicioso para o tempo que tinha. Estou certo que conseguiria fazer tudo o que queria, com tempo, mas não deu para mais - e bem que fui avisado. No entanto, não me arrependo de o ter escolhido, pois aprendi um pouco de tudo relativamente a simulações N-Corpos. Quero continuar a trabalhar neste projeto de forma a aprender mais sobre as diferentes técnicas e sobre a formação de estrutura em larga escala.

Além de resolver o problema identificado durante a avaliação do código, quero fazer uma análise mais aprofundada, começando por utilizar o algoritmo FoF e SUBFIND do GADGET-4 para analisar os *halos* de matéria negra no meu código e a sua evolução ao longo do tempo[2], verificar o efeito de diferentes *timesteps*, verificar a necessidade de *timesteps* variáveis, tanto a nível de eficiência como a nível de precisão numérica, e outros que possa vir a achar relevantes. Otimização também será um passo muito importante, pois, a correr na GPU, o meu código levou 40 minutos a correr a simulação, enquanto que o GADGET-4 na CPU com 6 cores demorou 25, o que é inaceitável tendo em conta que a GPU, na teoria, é ideal para estas simulações.

Quero também comparar a resolução dos dois códigos em escalas mais pequenas e estudar o efeito de métodos como o AP^3M (*adaptive particle-particle particle-mesh*) ou simplesmente introduzir grelhas de tamanho ajustáveis, mantendo o cálculo da força unicamente através da densidade e potencial.

References

Bagla, J. S. 2002, Journal of Astrophysics and Astronomy, 23, 185–196

Barnes, J. & Hut, P. 1986, Nature, 324, 446

Eclipse.sx, CC BY-SA 3.0 <https://creativecommons.org/licenses/by-sa/3.0>, via Wikimedia Commons. 2009, Complete barnes hut tree of a distribution of 5000 particles resembling two galaxies. Each particle is located in its own node., [Online; acessado em 15 de janeiro de 2021]

Hockney, R. & Eastwood, J. 1981, Computer Simulation Using Particles, Advanced book program (McGraw-Hill International Book Company)

Klypin, A. & Holtzman, J. 1997, arXiv preprint astro-ph/9712217

Molecular simulations in drug delivery: Opportunities & challenges Scientific Figure on ResearchGate. 2018, [Online; acessado em 15 de janeiro de 2021]

Mário João P. F. G. Monteiro. 2019, Sebenta de Astronomia Computacional (2019-10-18)

Quinn, T., Katz, N., Stadel, J., & Lake, G. 1997, arXiv preprint astro-ph/9710043

Rigopoulos, G. & Valkenburg, W. 2014, Monthly Notices of the Royal Astronomical Society, 446, 677

Springel, V., Pakmor, R., Zier, O., & Reinecke, M. 2020, Simulating cosmic structure formation with the GADGET-4 code

Springel, V., White, S. D. M., Jenkins, A., et al. 2005, Nature, 435, 629–636

[2] Teria conseguido fazer isto se tivesse conhecimento para tal. Como tinha que aprender mais teoria, não consegui fazê-lo a tempo.

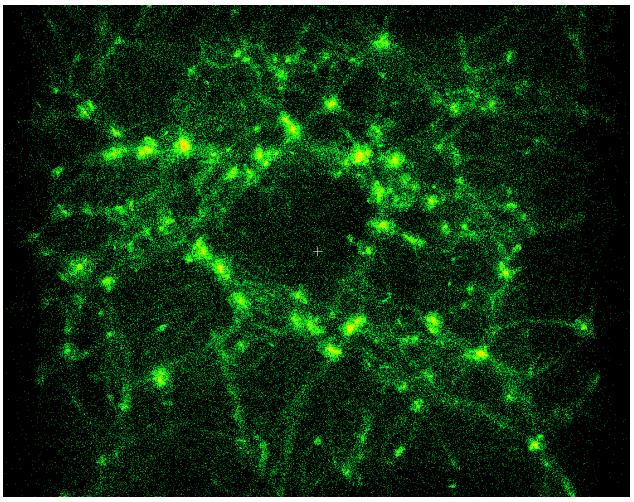

Fig. 3. Resultado da simulação para $z = 0$ com o meu código: plano *xy*.

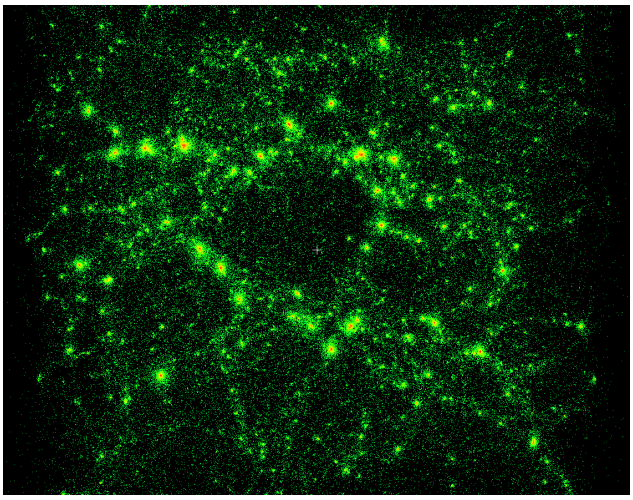

Fig. 4. Resultado da simulação para $z = 0$ com o GADGET-4: plano *xy*.

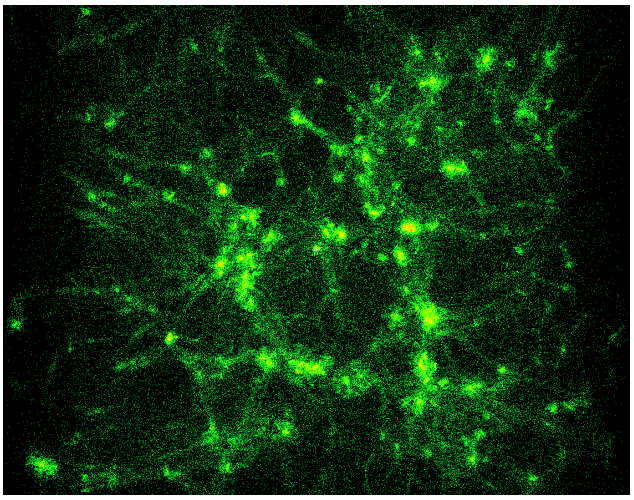

Fig. 5. Resultado da simulação para $z = 0$ com o meu código: plano *yz*.

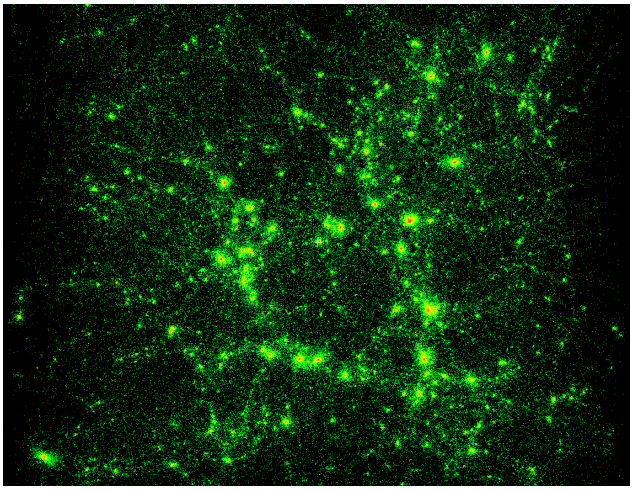

Fig. 6. Resultado da simulação para $z = 0$ com o GADGET-4: plano *yz*.

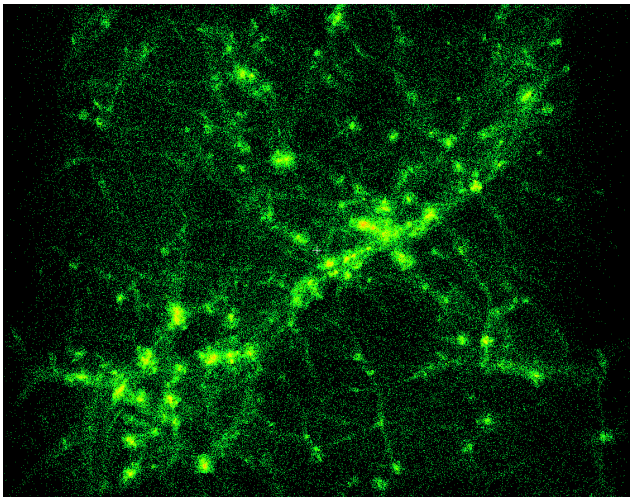

Fig. 7. Resultado da simulação para $z = 0$ com o meu código: plano *xz*.

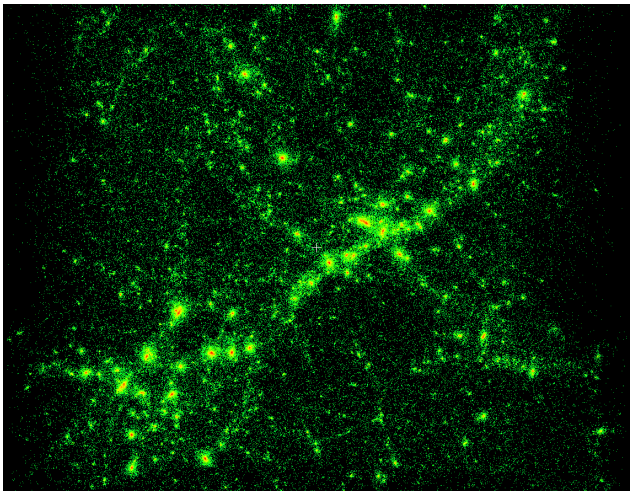

Fig. 8. Resultado da simulação para $z = 0$ com o GADGET-4: plano *xz*.

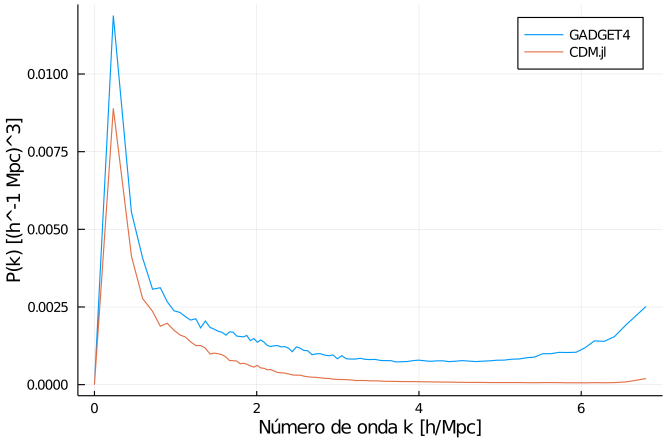

Fig. 9. Espetros de potência da matéria para o resultado do meu código e para o resultado do código GADGET-4, em $z = 0$. "CDM.jl" refere-se ao meu código.