Rapport d'EA : résolution numérique du problème à N-corps

Romain Lenoble - Chloé Vernière

5 décembre 2019

Table des matières

1 Introduction

Déterminer le mouvements de N points en interaction gravitationnelle les uns avec les autres est un problème physique fondamental en astronomie. Des des méthodes numériques permettent de résoudre ce problème pour un grand nombre de particules. En particulier, l'algorithme de Barnes-Hut, que nous avons implémenté, permet, moyennant quelques approximations quantifiables, d'obtenir une complexité de $N \log(N)$ grâce à une structure en arbre récursive, alors qu'un algorithme exact serait en $O(N^2)$.

Dans un premier temps, nous allons présenter la méthode générale employée pour résoudre ce problème. Puis dans un second temps, nous reviendrons sur les résultats que nous avons obtenus.

2 Méthodes pour résoudre le problème à N-corps

2.1 Équation

Dans un problème gravitationnel à N corps, chaque point i de masse $m_i$est soumis à une force gravitationnelle et donc à un potentiel gravitationnel par tous les autres points j tels que :

— $F_{j\to i} = -G * \frac{m_i * m_j}{|\vec{r_j} - \vec{r_i}|^3} * (\vec{r_j} - \vec{r_i})$ où $G = 6,67.10^{-11} m^3.kg^{-1}.t^{-2}$ est la constante gravitationnelle, r_i et $r_j$les positions des masses $m_i$et m_j ;

— $E_{ji} = -G * \frac{m_i * m_j}{(r_j - r_i)}$

Cependant, un important problème des simulations à N-corps arrive lorsque deux objets se trouvent très proches. La trajectoire de chaque particule n'est alors pas une conique classique comme dans le problème à deux corps. Lorsque la distance qu'une particule parcourt en une seule itération dépasse la distance entre elle-même et une autre particule, une collision s'est produite. Ce genre d'évènement peut alors modifier énormément l'énergie cinétique totale : c'est ce phénomène qui se passe lorsque des particules sont éjectées très violemment et sortent du potentiel d'attraction de la galaxie dans les simulations. Une solution pour limiter ce problème est d'implanter une force modifiée à courte portée, de sorte que la force de gravitation soit réduite à courte distance. Cela peut se traduite par l'ajout d'un paramètre d'atténuation que l'on a choisit $\epsilon = 5$USI. [4]

Pour chaque point i, on a :

$$m_i * \ddot{\vec{r}}_i = -G * m_i \sum \frac{m_j}{|\vec{r_j - r_i}|^2 + \epsilon^2} * \frac{(\vec{r_j - r_i})}{|\vec{r_j - r_i}|} \tag{1}$$

2.2 Calcul numérique

L'algorithme de Barnes-Hut [2] permet de résoudre les N équations du mouvement associées aux $N(N-1)$ interactions en un temps en $O(N \log N)$ au lieu de $O(N^2)$, moyennant des approximations contrôlables. Il repose sur un arbre hiérarchique dans lequel sont placés les points du système. Il fonctionne de la manière suivante : le plan contenant les centres de masse est divisé en quatre quadrants, qui sont représentés chacun par une branche de l'arbre. Chaque quadrant est ensuite divisé en quatre tant qu'il renferme strictement plus d'une particule. Un arbre résultant avec 1000 particules est présenté en figure 2.2. Plus les particules sont réparties de façon non homogène dans l'espace, plus l'arbre est donc profond, car il faut plus diviser l'espace pour isoler les particules. L'apport de l'algorithme pour le calcul d'une force d'interaction exercée sur une particule est de ne considérer que le centre de masse du quadrant considéré si la taille de celui-ci est plus grande que la distance entre la particule et le centre de masse multiplié par un certain angle θ. Nous avons pris $\theta = 0.7$ conformément à la littérature. Ainsi, si une particule est suffisamment éloignée d'un groupe de particules, seule l'interaction avec le centre de masse du groupe

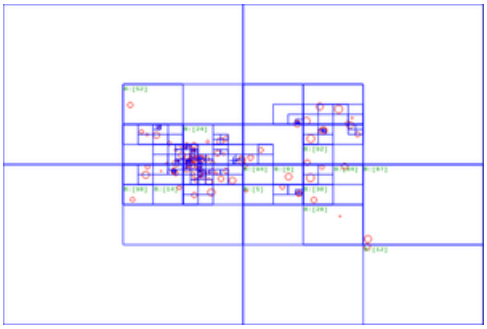


FIGURE 1 – Illustration de la division du plan selon l'algorithme de Barnes-Hut avec 1000 particules

de particules sera calculée, au lieu de la calculer avec chacune des particules du groupe :

$$Si\frac{l}{d} \leq \theta, \sum F_{j\to i} = F_{COM\to i}$$

Les feuilles externes de l'arbre contiennent ainsi au plus une particule tandis que les nœuds internes contiennent les information des centres de masse de leurs enfants.

La hauteur h de l'arbre ainsi construit est en $O(ln(N))$. En effet, soit une feuille externe contenant une particule. En moyenne, pour une répartition homogène des particules, elle contient $\frac{N}{4^h}$ particules. Donc :

$$1 = \frac{N}{4^h} \Rightarrow h = O(ln(N))$$

Lors du calcul de la force exercée sur sur une particule, l'arbre est parcouru en profondeur jusqu'à remplir le critère d'approximation ou tomber sur une feuille externe. Comme il y a N particules pour une profondeur moyenne de l'arbre en $O(ln(N))$, le calcul de l'ensemble des forces s'effectue donc en $O(Nln(N))$.

2.3 Intégration temporelle

Une fois l'arbre construit et les interactions entre les N particules calculées, il suffit d'intégrer 1 pour chaque particule afin d'obtenir sa nouvelle position. Pour ce faire, nous avons utilisé plusieurs schémas d'intégration d'ordre différents, même si la force modifiée permet une plus grande liberté sur le pas de temps. [3]

Dans la suite, nous notons $v_i(t)$ la vitesse de la particule au temps t, $r_i(t)$ la position de la particule en t, Δt le pas de temps et $a_i(t)$ l'accélération en t.

— Tout d'abord, un schéma d'intégration naïf au premier ordre (ou schéma d'Euler) :

$$v_i(t + \Delta t) = \ddot{r_i}(t) * \Delta t$$

$$r_i(t + \Delta t) = v_i(t) * \Delta t$$

Ce schéma au premier ordre est le plus simple qu'il existe, mais ses erreurs sont assez importantes, notamment pour la conservation de l'énergie.

— Puis la méthode Leapfrog au second ordre :

$$v_i(t + \Delta t) = v_i(t) + \frac{a_i(t) + a_i(t + \Delta t)}{2} * \Delta t$$

$$r_i(t + \Delta t) = r_i(t) + v_i(t) * \Delta t + \frac{1}{2} a_i(t) * \Delta t^2$$

Nous avons beaucoup utilisé cette intégrateur car il était très rapide, tout en permettant une bonne conservation de l'énergie.

— Enfin un schéma de Hermite au quatrième ordre :

$$v_i(t + \Delta t) = a_i(t) * \Delta t + \frac{1}{2} \dot{a_i}(t) * \Delta t^2$$

$$r_i(t + \Delta t) = v_i(t) * \Delta t + \frac{1}{2} a_i(t) * \Delta t^2 + \frac{1}{6} \dot{a_i}(t) * \Delta t^3$$

où $\dot{a_i} = -G \sum_{j \neq i} m_j \frac{x_{ij}^2 \dot{x_{ij}} - 3x_{ij}(\dot{x_{ij}}.x_{ij})}{|x_{ij}|^5}$ correspond à la la dérivée de l'accélération

Cet intégrateur est le plus précis que nous ayons utilisé. Ces équations sont symétriques en temps et donnent une excellente conservation de l'énergie. Il nécessite cependant de passer par un schéma implicite qui nécessite de calculer la dérivée de la force, ce qui a un cout de calcul non négligeable.

Pour la stabilité de chacun des schémas, nous avons choisi pour un pas de temps constant.

3 Simulations réalisées

Nous avons codé l'algorithme dans le fichier /code du git.

Pendant les deux premières semaines du projet, nous avons construit notre code autour de différentes classes : c'est ce qui nous semblait le plus simple au vu de la complexité de l'algorithme. Nous avons ainsi créé une classe pour les particules, une pour les nœuds de l'arbre et une pour l'arbre en entier. La classe particule était particulièrement lourde car elle contenait la vitesse, la position, la masse, l'énergie, l'accélération et la force qui s'exerçait sur la particule notamment. Lorsque nous avons voulu optimiser le code en augmentant le nombre de particules, nous sommes rapidement arrivés à des problèmes liés à cette structure : le code était très lent et nous étions limités en mémoire. De plus, les outils vectoriels de Numpy n'étaient pas utilisables avec une telle architecture.

Pour palier à ces problèmes, nous avons réécrit le code en nous affranchissant au maximum de la structure de classe et en utilisant des tableaux numpy qui sont beaucoup plus rapides à traiter par l'ordinateur. Dans la dernière version du code (fichier array_node), nous n'avons plus qu'une classe : la classe Node pour coder l'arbre. Le reste des données est stocké dans des tableaux où l'indice correspond à une particule en particulier. Cette nouvelle architecture a permis un gain de temps important, le code est près de 10 fois plus rapide, et nous a permis de lancer des calculs avec beaucoup plus de particules.

Les calculs restent tout de même très gourmands en temps, même sur les ordinateurs des salles d'informatique qui étaient environ dix fois plus rapides que nos ordinateurs personnels. Nous avons essayé d'optimiser la rapidité du code avec du multiprocessing mais l'architecture du code est assez complexe et nécessite d'avoir accès à la même information en même temps par tout les processeurs. La partie la plus chronophage du code est le calcul des interactions à chaque pas de temps entre toutes les particules (en $O(Nlog(N))$). Pour chaque particule, nous calculions les interactions avec ses voisines selon la descente de l'arbre, comme expliqué plus haut. C'est ici que nous aurions voulu utiliser du multiprocessing pour que chaque force soit calculée en même temps mais chaque calcul nécessitait l'accès aux mêmes données, ce qui ne facilitait pas cette pratique.

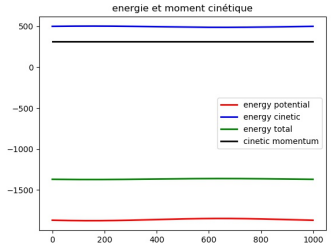
Nous avons uniquement fait des modélisation en 2D mais le code est très facilement extensible à la troisième dimension : il suffit d'ajouter une dernière dimension dans le choix du huitième de cube dans lequel appartient chaque particule au moment de la descente de l'arbre. Nous avons également toujours pris la constante gravitationnelle G=1. Toute les autres unités sont redéfinie par rapport à cette valeur.

3.1 Vérification de la validité de l'algorithme

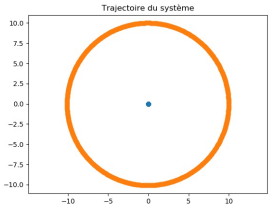
Premièrement, nous avons vérifié si notre algorithme avait une bonne précision. Pour cela, nous avons utilisé le problème à deux corps dont la solution analytique est connue. Nous avons notamment simulé un système Terre-Soleil

où l'un des deux corps a une masse bien supérieure à l'autre.

Voilà les résultats obtenus avec ce système, avec les conditions initiales du mouvement circulaire uniforme (vitesse nulle pour le soleil et $v = \sqrt{G\frac{M}{R}}$ pour la Terre), un rapport10^4 entre les masses (la masse de l'objet central vaut10^4celle du satellite) et une distance R=1 entre les deux particules :



Énergie et moment cinétique



Trajectoire (bleu : soleil, orange : Terre)

Dans cette simulation, nous avons utilisé un pas de temps de Δt =0.002 et un schéma d'intégration du premier ordre (d'Euler). On obtient bien une trajectoire circulaire et l'énergie est bien conservée. On peut vérifier les valeurs obtenues par le calcul.

Grandeur	Valeur
Énergie potentielle	$E_g = -2G\frac{Mm}{R} = -2.10^3[G]J$
Énergie cinétique	$E_c = \frac{1}{2}mv^2 = \frac{GMm}{2R} = 500[G]J$
Moment cinétique	$M = \lvert\vec{p} \wedge \vec{r}\rvert = mvR = 316[G]Jm$

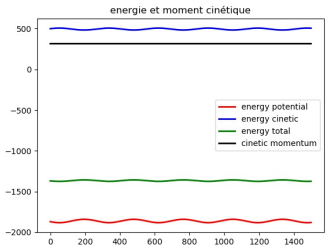
Ainsi, l'algorithme est fiable est donne des résultats cohérents pour le système à deux corps.

3.2 Étude des différents schéma d'intégration et des pas de temps

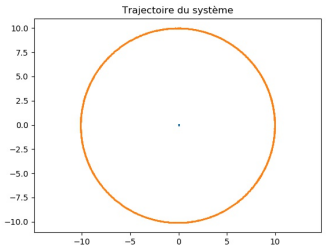
Comme présenté dans la première partie, différents schéma d'intégrations permettent d'intégrer les équations. Ils donnent des résultats différents selon le pas de temps choisi. La principale différence observée est sur la conservation de l'énergie.

Nous allons comparer ici le schéma d'Euler du premier ordre, le schéma leapfrog (deuxième ordre) et le schéma de Hermite (quatrième ordre) pour le même pas de temps dans la situation précédente. Le pas de temps vaut 0.007.

Schéma d'Euler :

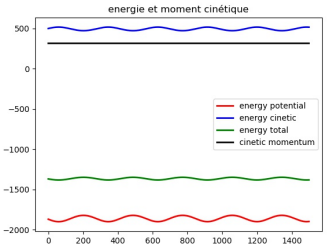


Énergie et moment cinétique

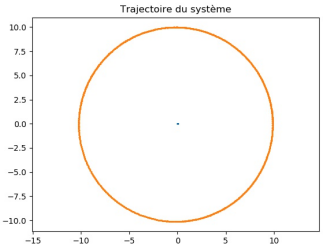


Trajectoire (bleu : soleil, orange : Terre)

Schéma Leapfrog :

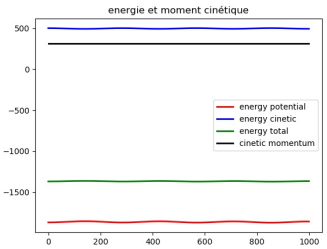


Énergie et moment cinétique

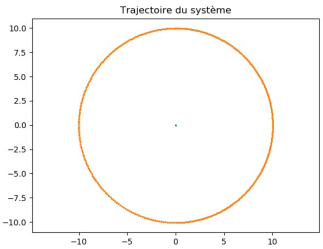


Trajectoire (bleu : soleil, orange : Terre)

Intégrateur de Hermite développé au 2ème ordre :



Énergie et moment cinétique



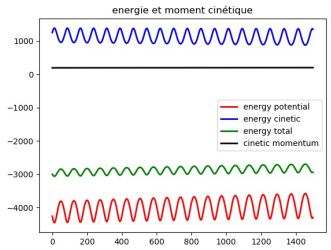
Trajectoire (bleu : soleil, orange : Terre)

Ainsi, on remarque une bien meilleure conservation de l'énergie pour le schéma du quatrième ordre, ce qui était attendu : c'est le schéma à l'ordre le plus élevé et symétrique en temps. La trajectoire semble cependant identique. Pour des simulations beaucoup plus longues, nous avons remarqué que les deux premiers schémas présentaient une plus mauvaise conservation de l'énergie et

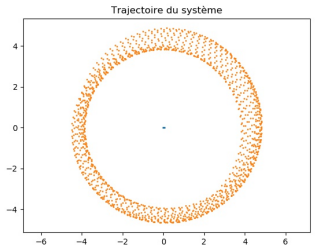
étaient très dépendant du pas de temps. Cependant, le schéma de Hermite au 4ème ordre nécessite de calculer de taux de variation de l'accélération. Ce calcul est coûteux, ce qui rend le code utilisant le schéma de Hermite près de 3 fois plus lent que le schéma de Leapfrog. C'est pour cela que nous avons utilisé ce dernier pour la suite de nos simulations.

On ne peut toutefois pas déduire de tels calcul le pas minimal qu'il faut adopter pour une simulation. En effet, le pas de temps est uniquement utilisé pour déduire l'accélération et la vitesse à partir de la force calculée. Si la force est très importante, alors l'accélération le sera aussi et la vitesse variera beaucoup. Le pas de temps devra alors être assez faible pour conserver l'énergie. Cette situation est observée lorsque le satellite se rapproche du soleil. La force varie de plus considérablement avec la position dans ce cas.

Exemple de simulation avec un pas de temps trop grand :



Énergie et moment cinétique



Trajectoire (bleu : soleil, orange : Terre)

Simulation obtenue avec les mêmes paramètres que précédemment mais avec une distance de R = 5 du satellite au soleil. Le schéma d'intégration est celui de Leapfrog.

Ainsi, lors d'une simulation avec de nombreuses particules et un point central très massif (comme c'est le cas dans une galaxie ou pour le système solaire), les particules situées proches du centre seront plus sensibles au pas de temps. Dans nos simulation, nous n'avons pas implémenté de pas de temps variable selon la valeur de la force, mais cela aurait pu être un bon prolongement.

Lors des simulation, nous tirons les particules au hasard. Il se peut alors qu'une particule se trouve très proche de l'objet massif et perturbe l'énergie totale du système très fortement. Pour cela, nous avons imposé un périmètre d'environ 1/10 du rayon de la distribution de masse sans particule.

3.3 Simulations obtenues :

A l'aide des ordinateurs des salles informatiques, nous avons pu faire des calculs avec un grand nombre de particules (jusqu'à plusieurs milliers). Ces calculs mettaient plusieurs heures à tourner. Nous avions comme objectif d'obtenir des simulations similaires à [1]. Pour cela, il nous fallait tout d'abord une simulation viable pour modéliser une galaxie.

Dans toute nos simulation, nous avons pris $\theta = 0.7$, qui était la valeur utilisée dans [1]. Une valeur plus grande permettait de meilleur calcul mais la conservation de l'énergie potentielle n'était pas très bonne.

Le schéma d'intégration choisi est le schéma Leapfrog qui a l'avantage d'être beaucoup plus rapide que le schéma hermitien et d'être stable.

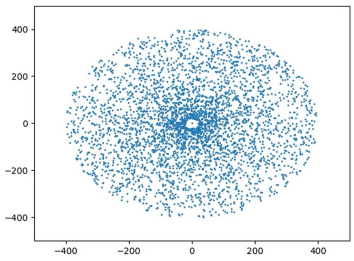
3.3.1 Simulation galactique

La simulation d'une galaxie dépend énormément des conditions initiales : masse de trou noir, distribution de masse et de vitesse.

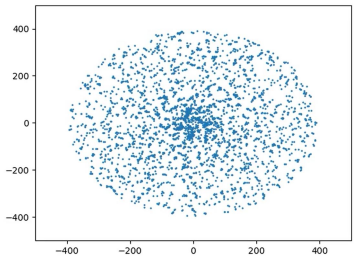
Pour commencer, nous avons pris une distribution de masse proportionnelle à $\frac{1}{r}$ et un vitesse pour chaque particule égale à $v(r) = \sqrt{G\frac{M(\leq r)}{R}}$. Cette vitesse assure une stabilité et évite que les particules s'effondrent au centre ou se libèrent du potentiel central. Nous nous attendons à voir les inhomogénéités de densité liées à l'initialisation augmenter jusqu'à former des filaments, à l'image des simulations cosmologiques actuelles. Cependant, notre simulation est très simpliste et ne prend prend en compte que la gravitation : pas de noyau actif de galaxies, de supernovae ou d'équations de mécanique des fluides. Ces phénomènes modifient énormément la morphologie des galaxies. Pour toutes ces raisons, notre modèle n'est pas très réaliste.

Enfin, notre modèle ne représente pas non plus tout un proto-système solaire car nous n'avons pas mis en place de processus d'accrétion entre les particules. Il ne prend pas en compte les collisions entre corps. Cette approximation est justifiée pour la modélisation d'une galaxie où les collisions entre étoiles sont très rares car les distances sont très grandes. En revanche, les planètes du système solaire se sont formées par accrétion successive de corps plus petits, les collisions étaient alors très fréquentes.

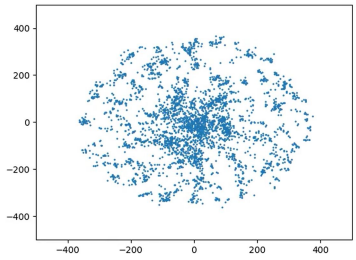
La simulation ci-dessus est la plus grande que nous ayons faite pour une galaxie seule. Elle inclue 4000 particules réparties aléatoirement suivant une distribution en $\frac{1}{r}$. Le pas de temps vaut 0.008 et il y a 10000 itérations. La masse du trou noir central vaut $10^3[G]kg$ et celle des particules élémentaire $1[G]kg$. Elle a été calculée sur les ordinateurs de l'X et a mis près de 10h à calculer.



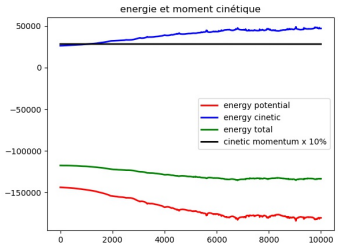
État initial



État après 4000 itérations



État final (après 10000 itérations)



Énergie et moment cinétique

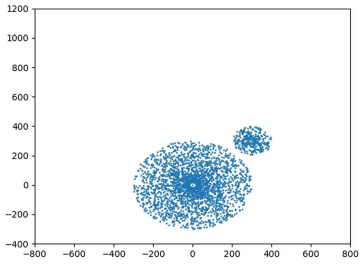
Sur ces quelques images, on remarque que à partir d'une distribution aléatoire, on observe rapidement l'émergence de grandes structures filamentaires, à l'image des bras observés sur la majorité des galaxies. L'énergie n'est pas conservée, ce qui est lié au schéma d'intégration.

En nous inspirant de [4], nous avons en fin de projet tenté de complexifier le modèle en simulant une galaxie composée d'un disque, d'un halos et d'un bulbe galactique. Les particules de ces différentes entités ont alors des positions, des densités et des vitesses initiales différentes. La simulation conduite plus haut est une simplification de celle-ci dans la mesure où toutes les particules sont du même type et que la densité de masse et la vitesse initiale suivent un modèle simple. Cependant, en dépit de nombreuses heures passées dessus, cette simulation n'a pas pu voir le jour car les modèles de vitesse présentés dans l'article ne pouvaient pas être déduits et calculés simplement.

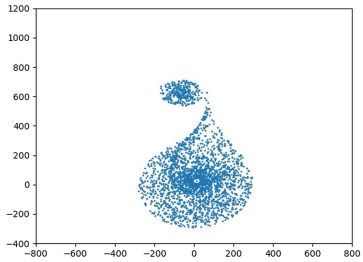
3.3.2 Simulation de collision entre galaxies

Pour reprendre la simulation de [1], nous avons utilisé la même initialisation que précédemment. Nous avons ajouté une deuxième galaxie en orbite autour de la première et nous avons observé les effets que cela produisait. Nous avons obtenu de belles simulations, que l'on peut retrouver dans le dossier /result du git.

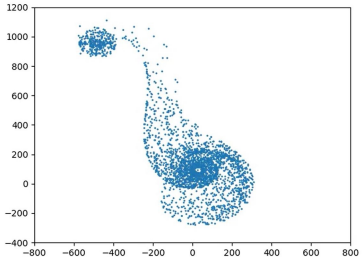
La simulation ci dessous correspond à ce genre de collision. Il y a 4000 particules et 10000 itérations. Le pas de temps vaut 0.008. La masse des deux trous noir vaut 10^6, le reste des particules a une masse de 1. Dans le référentiel des deux galaxies, chaque particule a une vitesse $v(r) = \sqrt{G\frac{M(\leq r)}{R}}$ assurant la stabilité de l'ensemble. Nous avons ajouté une vitesse de dérive de la petite galaxie : $v_{der} = \sqrt{G\frac{2M(\leq r)}{R}}$, supérieure à la vitesse de libération, ce qui fait que la deuxième galaxie n'est pas liée à la première.



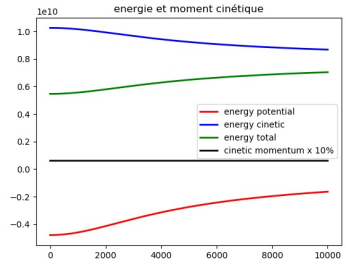
État initial



État après 3000 itérations



État final (après 8000 itérations)



Énergie et moment cinétique

Cette simulation ne conserve pas l'énergie : le pas de temps de 0.008 est trop grand.

4 Conclusion

Finalement, l'algorithme de Barn-Hut nous a permis de faire des simulations à N-corps à plusieurs milliers de particules. Cette algorithme rapide et optimisé permet de faire une infinité de simulations différentes selon les conditions initiales. Dans cette étude, nous avons étudié l'évolution d'un ensemble de particules, simulant une galaxie et la collision entre deux galaxies. Malgré sa simplicité à l'heure des super-calculateurs, il permet de rendre compte plutôt fidèlement de quelques phénomènes de l'évolution cosmologique.

Cependant, notre simulation pourrait être amélioré en implémentant différents comportements comme la collision et l'accrétion de matière pour le système solaire ou encore la dynamique des fluides pour les gaz galactiques. L'algorithme pourrait quant à lui être amélioré en prenant un pas de temps adapté à chaque particule ou en implémentant la méthode Fast Multipole dont la complexité est en $O(N)$.

Références

[1] The barnes-hut galaxy simulator. `https://beltoforion.de/article.php?a=barnes-hut-galaxy-simulator`.

[2] Josh Barnes and Pier Hut. A hierarchical o(n log n) force-calculation algorithm. *Nature*, 1986.

[3] Walter Dehnen and Justin I. Read. N-body simulation of gravitational dynamices. *Astro*, (20), May 2011.

[4] Lars Hernquist. N-body realizations of compound galaxies. *Astrophysical Journal Supplement*, June 1993.