Vincent Chouinard
111 161 797
Justin Hamel
111 239 203
Gabrielle Jess
111 184 460

Travail présenté à :
Louis Archambault
Antoine Allard
Daniel Côté

Travail pratique #3

PHY-3500 : Physique numérique

22 mars 2020

1 Oscillateurs

a) Pour commencer, voici l'équation à traiter :

$$\ddot{x} \equiv \frac{d^2x}{dt^2} = -\omega^2 x$$
$$\ddot{x} + \omega^2 x = 0$$

Nous devons séparer l'équation en deux EDO linéaires. Posons $u = x$ et $v = \dot{x}$. Nous obtenons 2 équations. Pour u :

$$\dot{u} = \dot{x}$$
$$\dot{u} = v$$

Pour v :

$$\dot{v} = \ddot{x}$$
$$\dot{v} = -\omega^2 x$$
$$\dot{v} = -\omega^2 u$$

Nos deux équations différentielles ordinaires sont :

$$\dot{u} = v \tag{1.1}$$
$$\dot{v} = -\omega^2 u \tag{1.2}$$

Le graphique produit par la technique numérique RK4 utilisant les équations différentielles ordinaires (1.1) et (1.2) se retrouve dans le fichier python **No1_a.py**. Il est important de noter ici que la figure affichée après traitement par RK4 n'a pas une amplitude constante. Ici, la conservation de l'énergie ne semble pas respectée, car l'amplitude augmente.

b) Nous retrouvons dans **No1_b.py** un graphique semblable, mais avec les conditions initiales changées. Sachant que $x(0)$ est la position initiale de notre oscillateur et $\dot{x}(0)$ est sa vitesse initiale, l'amplitude devrait être modifiée.

Nous pouvons voir dans le graphique produit par **No1_b.py** qu'il y a modification de l'amplitude et de la phase lorsque nous changeons les conditions initiales de l'oscillateur harmonique. L'amplitude n'est pas retrouvable par inspection, mais elle semble être fonction des deux conditions initiales. Nous voyons sur le graphique que la position initiale ($x(0)$) est bien celle que nous retrouvons sur la figure. L'amplitude du graphique produit semble être $A_{insp} \approx 11$ avec des conditions initiales $x(0) = 5$ et $\dot{x}(0) = 10$.

Résolvons analytiquement l'équation de l'oscillateur harmonique :

$$\ddot{x} = -\omega^2 x$$

posons $x = e^{rt}$ pour obtenir :

$$r^2 e^{rt} + \omega^2 e^{rt} = 0$$

Nous savons que $e^{rt} = 0$ uniquement lorsque t est l'infini négatif, et nous ne nous intéressons pas à cette limite. Donc,

$$r^2 = -\omega^2$$
$$r = \pm i\omega$$

En remettant r dans x, nous obtenons :

$$x = e^{\pm i\omega t}$$

Après un peu d'algèbre, ($e^{\pm i\omega t} = \cos\omega t \pm i \sin\omega t$) Nous obtenons une solution générale suivante :

$$x = x(0)\cos\omega t + \dot{x}(0)\sin\omega t \tag{1.3}$$

L'équation générale peut être modifiée pour

$$x = A\cos(\omega t + \phi)$$

où $A = \sqrt{x(0)^2 + \dot{x}(0)^2}$ et $\phi = \arctan\frac{\dot{x}(0)}{x(0)}$. La fréquence angulaire ω ne dépend pas des conditions initiales.

Vérifions si notre modèle est justifié par la théorie en retrouvant l'amplitude de notre figure du **No1_b.py** de $A_{insp} = 11$:

$$A_{theo} = \sqrt{5^2 + 10^2}$$
$$A_{theo} = 11.1803$$

Nous pouvons donc justifier qu'en modifiant les conditions initiales, soit la position ou la vitesse, nous modifions l'amplitude et la phase de l'équation sans modifier la période ($\omega = \frac{2\pi}{T}$).

c) Examinons l'oscillateur anharmonique suivant :

$$\ddot{x} = -\omega^2 x^3 \tag{1.4}$$

Retrouvons les deux équations différentielles ordinaires liées à l'équation (1.4). Posons $u = x$, $v = \dot{x}$ et $\omega = 1$. Pour $\dot{u}$:

$$\dot{u} = \dot{x}$$
$$\dot{u} = v$$

Pour v :

$$\dot{v} = \ddot{x}$$
$$\dot{v} = -\omega^2 x^3$$
$$\dot{v} = -u^3$$

Le système d'EDO à évaluer est :

$$\dot{u} = v \tag{1.5}$$
$$\dot{v} = -u^3 \tag{1.6}$$

Dans le fichier python **No1_c.py**, nous réutilisons la fonction programmée au début du numéro pour retrouver un graphique de la position en fonction du temps pour l'équation (1.4).

En testant la fonction avec une position initiale de 1.0, 3.0 et 5.0, nous avons constaté que certaines données traitées selon la résolution numérique RK4 surpassait la limite possible de traitement de python. Le fait que la technique RK4 rajoute de l'énergie dans le système pour chaque itération rendait exponentiel nos données à traiter. Nous avons dû réajuster le nombre d'itérations pour diminuer cette augmentation d'énergie et rendre traitable nos données avec RK4.

Nous pouvons constater avec les trois graphiques produits par le fichier **No1_c.py** qu'en effet, la fréquence angulaire ω change avec l'amplitude, ce qui n'était pas le cas pour l'oscillateur harmonique. Nous constatons donc les effets d'un oscillateur anharmonique dans une situation où le système ne s'approxime pas à un système harmonique (comme un pendule que nous rendons harmonique en faisant l'approximation des petits angles).

Le fichier **No1_c.py** présente aussi la période d'oscillation de chaque système. Nous observons que plus la position initiale est grande, plus la période est courte.

d) Dans le fichier **No1_d.py**, nous produisons 2 graphiques qui correspondent à l'espace de phase de chaque système (soit celui où $x(0) = 1.0$ et l'autre $x(0) = 5.0$). Nous voyons que la forme de la figure est ellipsoïdale et que les côtés s'éloignent de l'origine à chaque rotation.

L'éloignement de l'origine est causé par l'énergie du système qui n'est pas constante (mais qui augmente plutôt avec le temps). Ainsi, l'amplitude du système augmente et la vitesse maximale aussi. La vitesse maximale est atteinte lorsque la position de l'oscillateur est à $x = 0$, car l'énergie potentielle est alors entièrement transformée en énergie cinétique.

La forme ellipsoïdale du système est causée par la non-linéarité entre la vitesse et la position. C'est causé par le côté anharmonique de notre oscillateur. Aussi, la distance entre deux points situés à une période de différence est plus courte pour le graphique de condition initiale $x(0) = 5.0$ (Les lignes sont beaucoup plus "Collées entre eux"). Cela résulte en une plus grande distance parcourue par la ligne tracée en un même temps $t = 50$, et donc une plus grande fréquence angulaire et une plus petite période. L'amplitude change donc en effet la période d'oscillation de notre système anharmonique.

e) Dans les parties suivantes, nous utiliserons la fonction *solve_ivp* de *scipy.integrate*. Cette fonction permet d'utiliser RK45, DOS853 et plusieurs autres méthodes de résolution d'EDO. Nous prendrons RK45 (qui est essentiellement Runge Kutta de 5e ordre). Nous devons insérer notre système d'équations différentielles de premier ordre dans *solve_ivp*, comme nous avons fait dans le fichier python **No1_e.py**. Une figure présente le graphique de la position et de vitesse en fonction du temps du système d'EDO.

Nous constatons qu'après un certain nombre de périodes, le système semble entrer dans un équilibre et les courbes x(t) et v(t) ne semblent plus aléatoires.

f) À l'aide du fichier **No1_f.py**, nous remarquons la présence du cycle limite, qui est identique sur chaque graphique de condition initiale différente. Il n'y a en effet aucune interdépendance entre le cycle et les conditions initiales. Nous avons fait le test avec une vitesse initiale non-nulle ($v(0) = 3$) pour observer ce qui se produit sur le système, mais l'équilibre se fait quand même et nous obtenons à nouveau le cycle limite. Les conditions initiales ne produisent donc aucune dépendance sur le cycle limite.

g) Nous pouvons voir une figure en 3D au fichier **No1_g.py**. Ce n'est pas le bon modèle de graphique, mais les informations sont présentes quand même. Le graphique présenté est créé avec des conditions initiales $x(0) = 0.5$ et $v(0) = 0.0$. Nous pouvons voir et même bouger le graphique en 3D pour voir de haut que la forme se place selon le cycle limite.

2 Mécanique céleste

Il sera tout d'abord important de considérer que l'on étudie un problème de mécanique qui nécessite que l'on utilise une méthode qui conserve l'énergie du système. On utilisera donc la méthode saute-mouton pour résoudre le problème puisque celle-ci permet la conservation de l'énergie.

De plus, on trouvera les positions au fil du temps des trois corps en solutionnant l'équation différentielle trouvée à partir de la force gravitationnelle appliquée sur chacun des corps :

$$\mathbf{F}_i = m_i \mathbf{a}_i = -G \sum_{j \neq i} m_i m_j \frac{\mathbf{r}_i - \mathbf{r}_j}{|\mathbf{r}_i - \mathbf{r}_j|^3}$$

$$\mathbf{a}_i = -G \sum_{j \neq i} m_j \frac{\mathbf{r}_i - \mathbf{r}_j}{|\mathbf{r}_i - \mathbf{r}_j|^3}$$

$$\dot{\mathbf{v}}_i = -G \sum_{j \neq i} m_j \frac{\mathbf{r}_i - \mathbf{r}_j}{|\mathbf{r}_i - \mathbf{r}_j|^3}$$

a) Dans le fichier *TP3_No2_a.py*, on commence par définir les constantes du problème puis on définit le membre droit des équations différentielles dans la fonction F.

En appelant la fonction F, on spécifie le corps sur lequel la force est appliquée à l'aide d'une chaîne de caractères définissant la lettre représentant le corps (A, B, ou C).

En solutionnant cette équation différentielle, on retrouvera la vitesse des corps. Avec laquelle on pourra trouver la position. On solutionnera l'équation différentielle à l'aide de la méthode saute-mouton. Celle-ci consiste à trouver une valeur de

position ou de vitesse à l'aide des valeurs situées à un pas complet et à un demi pas en arrière de la valeur voulue. Mathématiquement, on a :

$$\mathbf{v}(t+h) = \mathbf{v}(t) + h\mathbf{F}\left(\mathbf{v}\left(t+\frac{1}{2}h\right)\right)$$
$$\mathbf{v}\left(t+\frac{3}{2}h\right) = \mathbf{v}\left(t+\frac{1}{2}h\right) + h\mathbf{F}(\mathbf{v}(t+h))$$
$$\ldots$$

Où h représente une tranche de temps :

$$h = \frac{t_f - t_i}{N}$$

On répète ces étapes pour trouver la vitesse pour toutes les tranches de temps. Pour la position des corps il faudrait utiliser le fait que $\mathbf{r} = t\mathbf{v}$. Dans notre cas, on aura donc :

$$\mathbf{r}(t+h) = \mathbf{r} + h\mathbf{v}\left(t+\frac{1}{2}\right)$$
$$\mathbf{r}\left(t+\frac{3}{2}h\right) = \mathbf{r}\left(t+\frac{1}{2}h\right) + h\mathbf{v}(t+h)$$

Par contre, pour commencer la méthode, il est nécessaire de connaître $\mathbf{v}(t)$, $\mathbf{F}\big(\mathbf{v}\big(t+\frac{1}{2}h\big)\big)$, $\mathbf{r}(t)$ et $\mathbf{F}\big(\mathbf{r}\big(t+\frac{1}{2}h\big)\big)$. Les conditions initiales nous donneront :

$$\mathbf{v}(t) = \mathbf{v}_i \qquad\qquad \mathbf{r}(t) = \mathbf{r}_i$$

On trouvera $\mathbf{v}\big(t+\frac{1}{2}h\big)$ à l'aide de la méthode de Runge-Kutta de quatrième ordre :

$$\mathbf{k}_1 = \frac{1}{2}h\mathbf{F}(\mathbf{r}_i)$$
$$\mathbf{k}_2 = \frac{1}{2}h\mathbf{F}\left(\mathbf{v}_i + \frac{1}{2}\mathbf{k}_1\right)$$
$$\mathbf{k}_3 = \frac{1}{2}h\mathbf{F}\left(\mathbf{v}_i + \frac{1}{2}\mathbf{k}_2\right)$$
$$\mathbf{k}_4 = \frac{1}{2}h\mathbf{F}(\mathbf{v}_i + \mathbf{k}_3)$$
$$\mathbf{v}\left(t+\frac{1}{2}h\right) = \mathbf{v}(t) + \frac{1}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4)$$

Puis on aura finalement :

$$\mathbf{r}\left(t+\frac{1}{2}h\right) = \mathbf{r}_i + \frac{1}{2}h\mathbf{v}\left(t+\frac{1}{2}h\right)$$

C'est ce que la fonction *mouton_3_corps* fait après avoir initialisée des *array numpy* vides pour enregistrer les positions de chacun des corps pour chaque tranche temporelle. Après avoir trouvé les valeurs nécessaires comme montré plus haut, la fonction fait une boucle à partir de la deuxième tranche temporelle pour trouver les positions.

De cette manière, on solutionne 12 équations différentielles simultanément, et ensuite on affiche les trajectoires parcourues par les corps à l'aide de la fonction *graph_3_corps*.

On remarque qu'on a besoin de beaucoup de tranches pour obtenir de bons résultats. En effet, les résultats fiables nécessitent au moins $N = 50000$ pour $t = 1$. Ceci est dû au fait que les corps se rapprochent beaucoup. Donc si le nombre de tranche est trop faible, l'algorithme n'est pas assez précis et voit une l'équivalent d'une singularité. On remarque alors que les corps sont envoyés extrêmement loin et n'interagissent plus.

Une fonction *anim_3_corps* permet de créer des animations lorsqu'on lance le programme *Python*. Cette fonction prend en argument le temps de départ *t_i* (0 dans notre cas), le temps d'arrêt *t_f*, le nombre de tranche N et la quantité *slice* qui

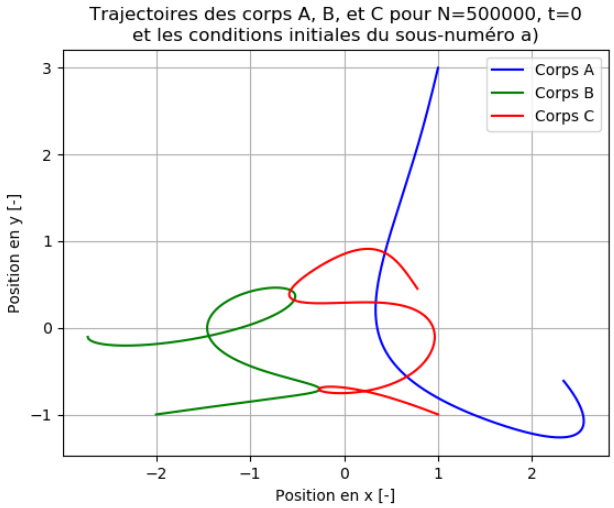


FIGURE 2.1 – Trajectoires des corps A, B et C pour $t = 1$ et $N = 500000$

définit le nombre de fois que l'on veut couper de moitié la grosseur des *array* de résultats. Le dernier argument permet d'observer les animations dans un délai raisonnable, car le minimum de temps qu'il peut y avoir entre deux images dans une animation de *matplotlib* est de 1 ms. Par exemple, dans le cas où on observe une animation d'un cas avec $N = 50000$, l'animation durera 50 secondes. Les animations nous permettent d'analyser plus précisément les trajectoires des corps au fil du temps.

b) Dans ce cas, il faut porter un regard particulier au nombre de tranche. En effet, entre $t = 2$ et $t = 3$, les corps passent extrêmement proche l'un de l'autre. Il faut donc plus de $N = 15 \times 10^6$ pour ne pas que les corps B et C soient éjectés déraisonnablement loin par la simulation. Évidemment, il ne faut pas monter le nombre de tranches trop haut pour ne pas avoir un temps de calcul trop haut.

Malheureusement, il est en ce moment impossible de pousser le programme plus loin que que $t = 2.519$ puisque la précision nécessaire pour atteindre une précision adéquate pour surpasser la singularité nécessite un temps qui nous n'ait pas disponible à l'instant. La figure 2.2 montre donc l'instant avant que les corps B et C soient ejectés hors du puits de potentiel de chacun des autres corps

c) Il y a une infinité de direction que l'on pourrait prendre pour analyser la stabilité du système. Le système avec ses valeur initiales le rendant stable est montré à la figure 2.3.

Nous avons décidé d'analyser deux cas. [1] Le premier est affiché à la figure 2.4. Dans ce cas, nous avons augmenté la vitesse d'initiale d'une quantité minime de 0.003 unités de vitesse. On voit alors que le système reste plutôt stable entre dans une précession horaire et le bulbe qui était au départ à gauche s'agrandit, alors que celui qui était à droite devient plus petit. Le système reste plutôt assez longtemps. En effet, dans la sous-figure (d), à $t = 1996$, le système est encore stable mais les anneaux commencent à s'agrandir et s'approcher d'une grandeur critique. Malheureusement, puisque la trajectoire a traversé presque toute l'aire de la figure, on ne peut pas voir la trajectoire des deux autres corps. Par contre, on sait qu'ils sont encore là puisque les anneaux sont encore assez centrés, et implique que les deux autres corps sont encore présents. Puisque le bulbe de gauche continu de s'agrandir, le système va tendre vers l'instabilité et tranquillement devenir chaotique.

Dans le deuxième cas, montré à la figure 2.5, on a ajusté la position initiale du corps C (qui est normalement centré) de 0.1 unités de longueur. On remarque que dès la fin de la première révolution des corps, le système commence déjà à être beaucoup plus instable que le système initial. En effet, celui-ci entre dans une précession horaire, et les trajectoires des différents corps sont beaucoup plus séparées que dans le cas précédent.

À $t = 251.5$, des flèches ont été ajoutées à l'image pour faire paraître le corps A et B qui m'apprêtent à passer très proche l'un de l'autre. Dans notre analyse du cas pythagoricien, on sait deux corps qui passent aussi proche l'un de l'autre va nécessairement briser le cycle dans lequel ils étaient entrés et nécessitent une très grande précision pour pouvoir calculer

1. Les figures peuvent être obtenues avec les commandes en commentaires dans le fichier *TP3_No2_c*

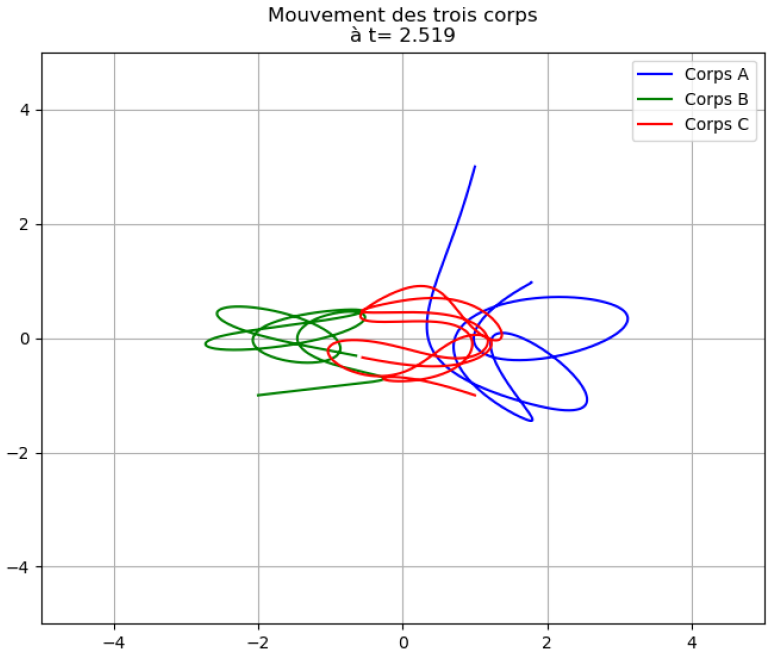


FIGURE 2.2 – Trajectoires des corps A, B, et C à $t = 2.519$ et $N = 15 \times 10^6$

la trajectoire exacte. On sait donc que c'est le moment exact où le système perd sa stabilité précaire.

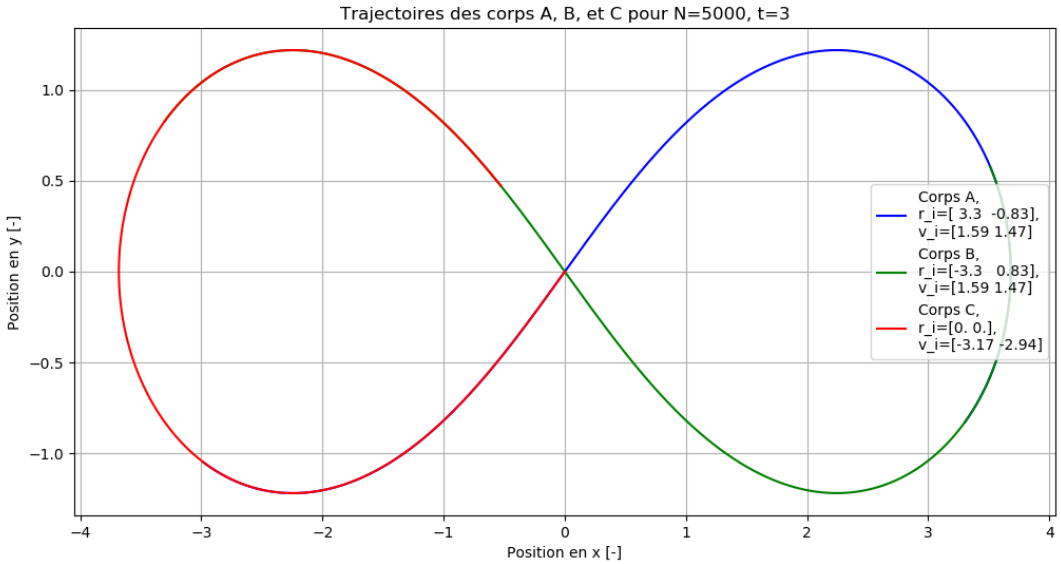
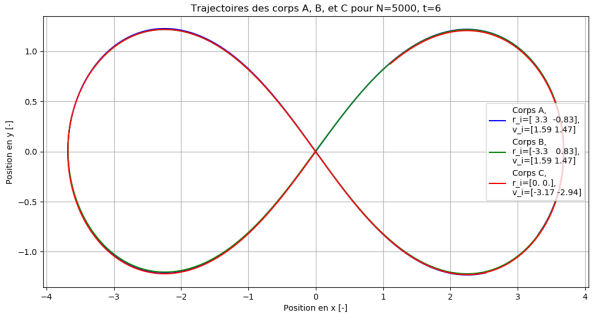
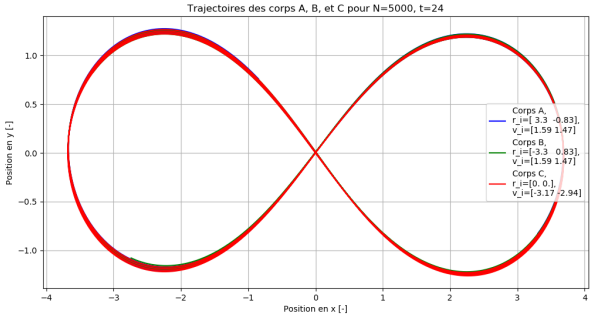


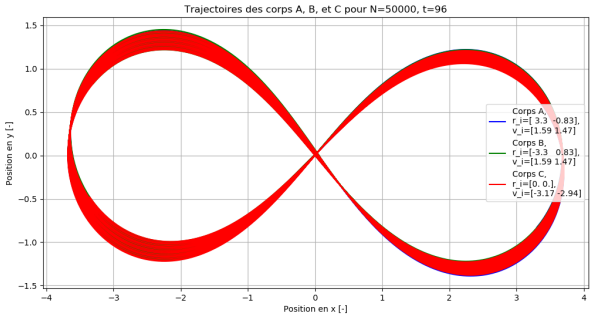
FIGURE 2.3 – Le système avec ses valeurs initiales de départ le rendant stable à t=3 et N=5000



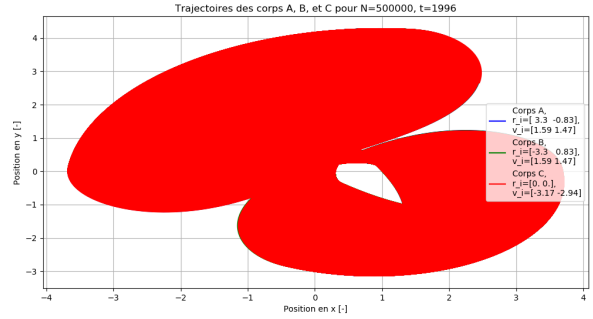
(a) t=6, N=5000



(b) t=24, N=5000

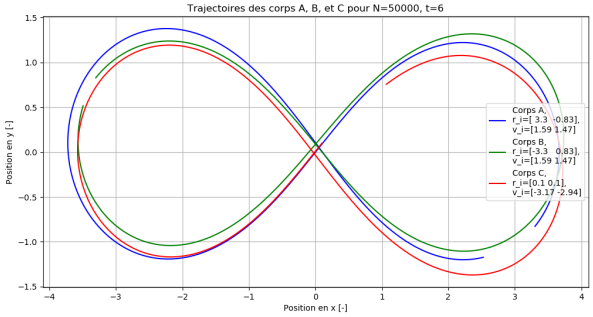


(c) t=96, N=50000

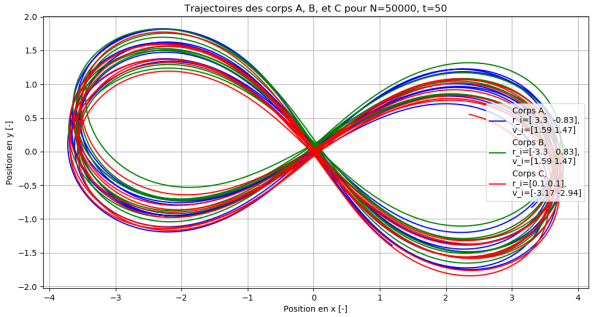


(d) t=1996, N=500000

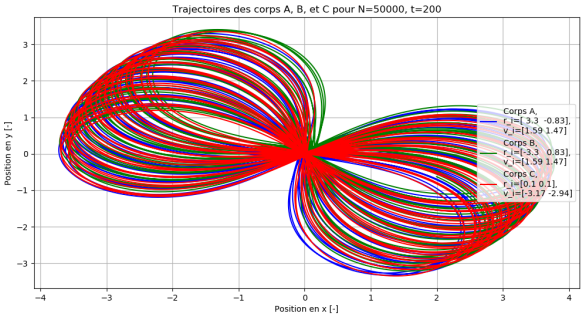
FIGURE 2.4 – Ajustement de la vitesse initiale du corps C de +0,001 unités de vitesse



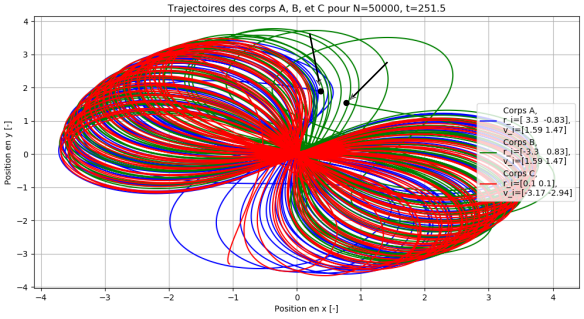
(a) t=6, N=50000



(b) t=50, N=50000



(c) t=200, N=50000



(d) t=251.5, N=50000

FIGURE 2.5 – Ajustement de la position initiale du corps C de +0,1 unités de longueur