

SIMULATEUR D'INTERACTIONS ÉLECTRIQUE



Louis Naili

Abstract—Cette publication est un résumé détaillé de nos recherches sur le sujet du simulateur d'interactions électrique. Il nous permet d'y exposer nos recherches ainsi que nos résultats de simulation.

I. INTRODUCTION

Dans le cadre de la 4ème année de notre formation en apprentissage, nous avons été amenés à conduire un projet de recherche.

Il nous a été demandé de concevoir un logiciel de simulation d'interactions électriques de charges isolées. L'objectif étant de calculer numériquement les interactions électriques entre plusieurs particules chargées afin de déterminer les grandeurs physiques associées. Ce programme a pour vocation, à terme, de permettre d'illustrer le cours d'électromagnétisme.

Ce projet est divisé en trois parties. Dans un premier temps il a fallu établir l'état de l'art concernant les méthodes de simulations existantes ainsi que leur mise en œuvre. Par la suite il a fallu éprouver le modèle physique retenu en le comparant avec des résultats théoriques connus. Enfin il a fallu réaliser le logiciel constituant le livrable du projet.

Ce document propose une synthèse de nos recherches bibliographiques concernant les différentes méthodes de simulation ainsi que les intégrations de Verlet. Suite à cela, sera proposé, une analyse du sujet ainsi qu'une liste non exhaustive des améliorations et modifications à apporter au sujet initial. Ce document se termine sur une présentation de nos perspectives pour l'avenir ainsi que les éventuels défis associés.

II. ÉTAT DE L'ART

Comme indiqué précédemment, la première partie de ce projet a été dédiée à effectuer un état de l'art. L'objectif a été de prendre connaissance des travaux existants sur le sujet. Nos recherches ont été axées sur le phénomène de discrétisation du temps ainsi que sur les méthodes de simulation.

A. Forces électrostatiques

Ce principe étant la base de toute nos recherches et de notre projet, nous avons d'abord commencé par travailler dessus. Il faut savoir que l'intensité de la force électrostatique entre deux charges électriques est proportionnelle au produit des deux charges et est inversement proportionnelle au carré de la distance entre les deux charges. La force est portée par la droite passant par les deux charges comme l'illustre la figure 1 ci-dessous.

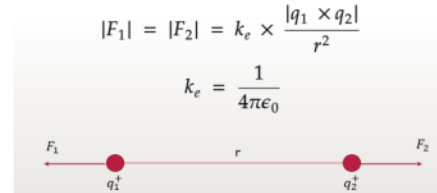


Fig. 1. Schéma du calcul de la force électrostatique ainsi que de l'application de cette force sur deux charges.

B. Discrétisation du temps

Comme dans toute simulation numérique, le temps évolue de manière discrète : le temps est découpé en une suite d'instants séparés par un intervalle de temps très court appelé "pas-de-temps" ou "time-step" h (n'étant pas nécessairement constant) et en déterminant les valeurs des variables au temps $t + h$ en fonction des variables au temps t , de proche en proche, sans avoir besoin de conserver en mémoire les valeurs calculées précédemment. La complexité associée à ce type de problèmes est d'assurer une précision contrôlée aux prédictions, et cela en dépit de la valeur finie du pas temporel h .

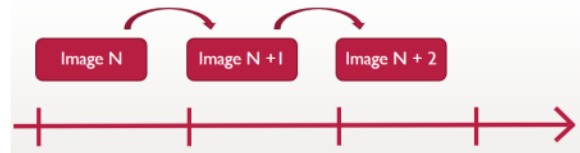


Fig. 2. Schéma de la discrétisation du temps avec le calcul de la position des charges entre chaque image.

C. Méthode de simulation

Au cours des recherches, il a été noté que de multiples méthodes de simulation existaient. Celles-ci varient en efficacité et en performance. La méthode la plus connue est l'algorithme d'Euler. Cependant cette méthode a le défaut d'être peu performante tout en étant imprécise. C'est la raison pour laquelle elle est quasi-exclusivement utilisée dans le cadre de démonstrations et d'apprentissage.

Il existe différents algorithmes de Verlet. De ceux-ci, l'algorithme "Verlet Positions" proposé par Loup Verlet en 1967, présente les meilleures performances. Cependant, ce dernier donne les positions à l'instant $t + t$ et les vitesses à l'instant t . Un autre de ces algorithmes est nommé "Verlet

Vitesses", il est dérivé du précédent et donne les positions et vitesses à $t + \delta t$. En dehors de l'algorithme de Verlet, il existe un algorithme nommé "leap frog", qui donne les vitesses à $t + \frac{\delta t}{2}$ et les positions à $t + \delta t$. Une autre méthode est d'utiliser les algorithmes de type "prédicteur correcteur" qui font une approximation de la nouvelle position, vitesse et accélération à $t + \delta t$ à l'aide d'un développement de Taylor.

D. Intégration de Verlet

L'intégration de Verlet est un schéma d'intégrations permettant de calculer la trajectoire de particules en simulation de dynamique moléculaire. Malgré la simplicité de la méthode d'Euler, les intégrations de Verlet permettent une meilleure stabilité ainsi que la validation d'importantes propriétés associées aux systèmes physiques telles que réversibilité dans le temps et la conservation des propriétés.

E. Algorithme de Verlet à un pas

- Initialisation du pas h et de la durée T .
- Initialisation des conditions initiales: $t = 0$, $\vec{r} = \vec{r}_0$ et $\vec{v} = \vec{v}_0$
- Définition de la fonction $\vec{a}(\vec{r})$
- Tant que $t \leq T$
 - Calcul de $\vec{a}_0 = \vec{a}(\vec{r})$
 - Nouvelle position: $\vec{r} = \vec{r} + h\vec{v} + \frac{h^2}{2}\vec{a}_0$
 - Calcul de $\vec{a}_1 = \vec{a}(\vec{r})$
 - Nouvelle vitesse: $\vec{v} = \vec{v} + \frac{h}{2}(\vec{a}_0 + \vec{a}_1)$
 - $t = t + h$
 - Enregistrement des données

III. SPÉCIFICATIONS DU LOGICIEL

A. Choix des technologies

Avant de commencer le développement de l'application, nous avons effectué un comparatif des technologies et langages de programmation les plus adaptés au besoin.

Le livrable se devait de répondre à plusieurs critères simples. L'application de simulation doit être multiplateforme, simple d'installation et d'utilisation, tout en restant modulaire et en conservant des performances suffisantes. Les critères précédents étant très dépendants de la question du langage de programmation, il a donc été choisi de prendre le temps d'étudier la question en effectuant des tests de performance au travers de "benchmarks". Ces tests ont été croisés avec la capacité de ces langages à gérer simplement les dépendances tout en étant aisément testable et simplement compilable pour tous les systèmes.

Voici la liste des langages initialement choisie pour effectuer nos tests :

- C avec GTK3
- C++ avec GTKMM
- NodeJS avec Front VueJS
- Golang avec Front VueJS

Afin de comparer le temps de calcul pour chaque langage, nous avons étudié les résultats de "The Computer Language Benchmarks Game" sur l'algorithme 'regex-redux'.

source	secs	mem	cpu load			
C gcc #5	0.80	148,844	53%	100%	54%	48%
C++ g++ #6	1.03	275,752	36%	37%	100%	40%
Go #5	3.55	375,092	30%	46%	21%	68%
Node js #3	4.67	1,115,744	2%	2%	29%	91%

Fig. 3. Tableau comparatif des performances des différentes langages.

Sur le plan performance le C représente la meilleur option afin de garantir un système performant même dans le cas de lourdes opération. Néanmoins le C a de nombreux défaut comme la portabilité difficile du code, l'obscurité des sources lorsque le projet prend de l'envergure, la mauvaise gestion des dépendances et la très faible quantité d'outils intégrés built-in. En terme de gestion des dépendances et de portabilité, le NodeJS offre la meilleur solution. En effet NPM assure tout le travail de gestion des dépendances. Le NodeJS étant un langage interprété, il est très simple de distribuer le code. Cependant les scripts JavaScript ont besoins d'avoir NodeJS installé sur la machine hôte. De plus il offre la moins bonne performance de calculs et d'utilisation de la mémoire. NodeJS offre un faible nombre d'outils intégré.

En moyenne le Go est la solution qui offre le plus d'avantages. Certes il n'est pas le plus performant et n'a pas la meilleur gestion des dépendances. Mais il garde de bons résultats dans les différentes catégories. Il possède également un grand nombre d'outils built-in et il se démarque grâce à sa simplicité d'utilisation. Le Go possède également un système de compilation avancé qui lui permet de distribuer le livrable très simplement.

B. Le livrable

Le livrable de ce projet est un logiciel de simulation d'interactions électroniques permettant d'observer les interactions entre différentes charges. Le but est de pouvoir effectuer une simulation et d'en extraire les données dans un fichier JSON. Ce dernier contient toutes les informations sur les particules comme leur positions, leur masses, leur vitesses ou encore leur accélérations. Ce logiciel a été construit à partir de zéro en conservant une modularité suffisamment importante dans le but de pouvoir implémenter les fonctionnalités demandées et également de pouvoir améliorer le logiciel à l'avenir.

Le logiciel fonctionne dans un navigateur web, ce choix a été fait dans l'optique de rendre l'utilisation la plus simple possible. Un des objectifs était de rendre la prise en main du logiciel la plus simple possible, de manière à ce qu'une personne extérieure au domaine de l'informatique puisse l'utiliser sans souci.

Le logiciel est doté d'un système de gestion des projets, il est donc possible de créer, sauvegarder et ouvrir des fichiers projet depuis l'interface. Le logiciel permet de simuler les interactions électroniques entre plusieurs charges dans un espace défini ou non. En effet, il est possible d'ajouter des formes géométriques sur le canevas afin de limiter l'espace de déplacement des charges et d'observer la répartition des charges au sein de la forme. Le volet de droite permet de visualiser les caractéristiques d'une charge sélectionnée.

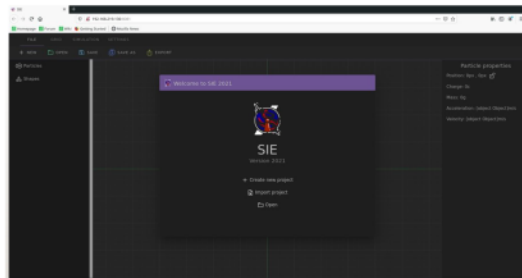


Fig. 4. Visuel de l'application à son démarrage.

IV. RÉSULTATS

Suite à notre état de l'art, nous avons effectué des calculs théoriques afin de savoir à quoi devra ressembler notre simulation une fois finie et pouvoir valider le modèle physique. Suite aux résultats théoriques, nous avons observé que lors d'une simulation avec plusieurs charges dans un cercle, ces charges se répartissent uniformément autour de ce cercle comme le montre la figure suivante.



Fig. 5. Résultat théorique de la répartition des charges.

Nous avons simulé 3 cas différents que nous allons expliquer ci-dessous. Tout d'abord, nous avons commencé par le cas classique, c'est à dire disposer différentes charges dans un cercle à des positions aléatoires pour observer leurs déplacements. Nous avons pu observer lors de cette simulation que les charges se sont réparties de manière uniforme autour du cercle. Nous avons donc pu retrouver ici l'application du théorème de Gauss.

Nous avons ensuite testé un deuxième cas qui est de disposer plusieurs charges autour d'une autre charge se trouvant au centre du cercle. On observe donc que les différentes charges se répartissent tout autour du cercle mais que la charge centrale ne se déplace pas. En effet, les forces exercées sur

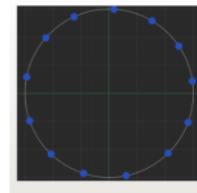


Fig. 6. Résultat simulé de la répartition des charges dans un cas classique.

cette charge ne changent pas malgré le déplacement des autres particules.

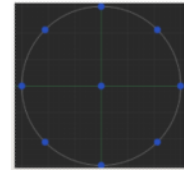


Fig. 7. Résultat simulé de la répartition des charges avec une charge centrale.

Nous avons fini par la simulation d'un oscillateur. Cette simulation consiste à placer différentes charges de manière parfaitement symétrique afin que certaines d'entre elles puissent osciller. Cette simulation est physiquement impossible mais ne l'est pas dans notre simulation. En effet, dans notre simulation il nous est possible de placer précisément les charges de façon à ce qu'elles aient une symétrie parfaite. Or en physique, il va toujours exister une marge d'erreur qui fera en sorte que cette symétrie parfaite ne soit pas possible à atteindre. De plus, il faut savoir que nous simulons dans un environnement idéal, c'est à dire dans le vide, il n'y a donc ni perturbation extérieure ni perturbation des matériaux.

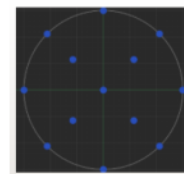


Fig. 8. Résultat simulé de la répartition des charges dans un oscillateur.

V. PERSPECTIVE POUR LE FUTUR DE L'APPLICATION

Ce projet étant un projet qui nous a fortement intéressés, nous avons pris la décision de ne pas nous arrêter après la date butoir et de continuer à améliorer notre application. En effet, actuellement l'application est totalement fonctionnelle, il est donc possible d'effectuer une simulation en plaçant plusieurs charges dans un cercle ou non en fonction du besoin. Cependant, nous souhaitons finir de développer le livrable de façon à ce qu'il puisse être utilisé comme exemple en cours d'électromagnétisme. Pour cela, nous devons encore travailler sur certains éléments visuels.

De plus, nous avons d'autres objectifs afin d'obtenir une application bien plus complète qu'à présent. Nous allons donc commencer par implémenter la visualisation des lignes de champs lors de la simulation.

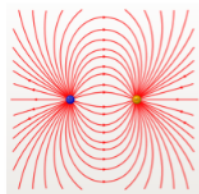


Fig. 9. Exemple de visualisation des lignes de champs.

Ensuite, dans la continuité de pouvoir simuler le déplacement des charges dans un cercle, nous allons implémenter d'autres formes comme le rectangle ou le triangle afin d'avoir des simulations d'une complexité supérieure. En effet, le cercle reste une forme basique ce qui ne complexifie par énormément la simulation. A terme, nous aimerions pouvoir simuler le déplacement des charges dans une forme quelconque.

VI. CONCLUSION

En définitive, nous avons eu la chance de travailler sur un projet très intéressant qui nous a permis de travailler d'une manière que nous ne connaissions pas en alliant développement et recherche. Nous sommes très satisfaits de l'application que nous avons conçue car celle-ci est fonctionnelle et nous avons pu valider le modèle physique que nous voulions implémenter.

REFERENCES

- [1] Charlotte BECQUART et Michel PEREZ, "Dynamique moléculaire appliquée aux matériaux", Techniques de l'ingénieur, 10 janvier 2010.
- [2] Loup Verlet, "Computer experiments on classical fluids. i. thermodynamical properties of lennard-jones molecules", Physical Review, vol 159, 1967, p. 98.
- [3] Furio Ercolessi, A Molecular Dynamics Primer.
- [4] Méthode de Verlet : <https://femto-physique.fr/omp/methode-de-verlet.php>
- [5] https://fr.wikipedia.org/wiki/int%C3%A9gration_de_verlet
- [6] Principes mathématiques de la philosophie naturelle d'après la traduction du latin en français par Emilie du Chatelet (1756), p. 17; la version originale en latin est : Corpus omne perseverare in statu quo quiescendi vel movendi uniformiter in directum, nisi quatenus a viribus impressis cogitur statum illum mutare.
- [7] Méthodes numériques et simulations sur : https://www.physique.usherbrooke.ca/pages/sites/default/files/PHQ404_0.pdf par David Sénéchal.
- [8] https://fr.wikipedia.org/wiki/M%C3%A9thode_d'Euler
- [9] Butcher, John C. (2003), Numerical Methods for Ordinary Differential Equations, New York: John Wiley & Sons, ISBN 978-0-471-96758-3.
- [10] Press, WH; Teukolsky, SA; Vetterling, Flannery, BP (2007). "Section 17.6. Multistep, Multivalue, and Predictor-Corrector Methods". Numerical Recipes: The Art of Scientific Computing (3rd ed.). New York: Cambridge University Press. ISBN 978-0-521-88068-8.