

Dossier modélisation 2018

# Le sujet

Nous avons décidé de mener ce projet de modélisation sur le thème du pendule. Le projet comportera donc trois parties distinctes, la première étant la modélisation physique fidèle à la réalité (dans les limites du possible), la seconde étant l’animation de pendule en utilisant des matrices de coordonnées homogène. A terme nous souhaitions pouvoir choisir indépendamment chaque paramètre (gravité, origine, poids…).

## Les problèmes rencontrés

Changement de type :

Les premiers problèmes survenus sont dus à un changement de type. Lorsque nous voulions multiplier une liste de tuples par une matrice et ensuite remettre tout ça dans une liste le calcul modifiait le type dans la liste et nous ne pouvions plus utiliser cette liste comme nous le voulions.

Exemple : [(x, y)] -> [(z, u)] Ici x et y sont d’un certain type et après multiplication ils sont convertis en un autre type et deviennent donc z et u

Pour régler ce problème nous aurions pu reprendre notre code et passé tous nos coordonnés en matrice ou refaire une fonction de multiplication de matrice mais avec des listes, nous avons choisis cette dernière option.

Cocalc

Le prochain problème provient de COCALC lui-même, l’interpréteur ne calcule pas les valeurs pour ensuite les remplacées dans chaque formule mais garde à chaque fois le résultat exact :

Exemple : pi \* pi n’est pas simplifié en 9.86960 mais reste pi \* pi

Or notre application reprends en permanence le résultat précédent donc il est arrivé un moment où la variable stockant l’angle actuel du pendule est devenu trop grande pour html cocalc renvoyait une erreur directement au niveau du HTML (« html max\_size… »).

En plus de cela même si l’on n’avait pas l’erreur le programme mettait 15 minutes faire les calculs même avec la fenêtre sage et ralentissait donc le déroulement du projet.

Message cocalc :  WARNING: This project is running low on memory. Upgrade Shared RAM memory in settings, restart your project or kill some processes. ([more information](https://github.com/sagemathinc/cocalc/wiki/My-Project-Is-Running-Out-of-Memory); memory usage is updated about once per minute.)

Pour régler ces problèmes nous sommes simplement passés sur un interpréteur en local (PyCharm) qui gère mieux les calculs et est beaucoup plus rapide. Nous avons juste dû refaire l’animation avec Tkinter puisque la fonction animation n’existe pas sur le Python de base

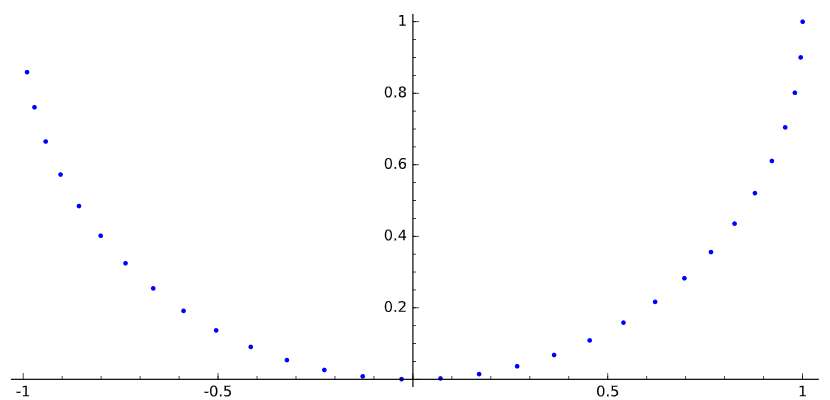
## Réalisations par ordre chronologique

Les graphiques suivants représentent la position du pendule avec un temps entre points contant. Les

Graphique suivant ne sont pas orthonormé, les distances sont plus longues en abscisse que celle en ordonné ce qui déforme un peu la trajectoire du pendule

Vitesse constante

Version 1 : un seul aller du pendule avec une vitesse constante

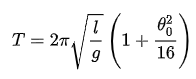


Introduction accélération linéaire

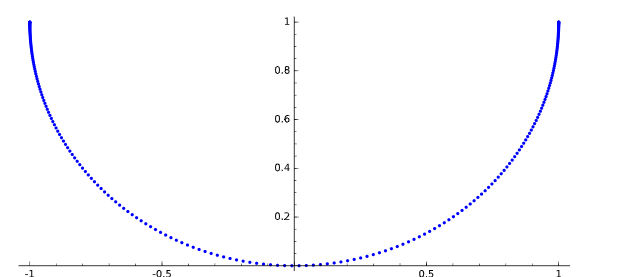
V2 : un seul aller accélération linéaire

Vitesse en t0 vaut 0

Vitesse en t = T/4 vaut Vmax

T étant la période du pendule (On divise T par quatre pour avoir la première fois où le pendule est en bas). Le calcul de T se fait avec la formule de Borda, cette formule fonctionne bien dans des angles compris entre -pi/2 et pi/2. Cette formule est plus performante que la formule utilisée au lycée qui ne marche que pour des petits angles car elle approxime sin(thêta) = thêta pour simplifier la résolution de l’équation. On appelle cela l’approximation des petits angles ou approximation de Gauss très utile pour résoudre des équations à la main.

La formule de borda nous permet par la suite de vérifier que l’utilisateur ne rentre pas un temps entre image supérieur au temps d’un aller (le programme n’est pas fait pour ça).



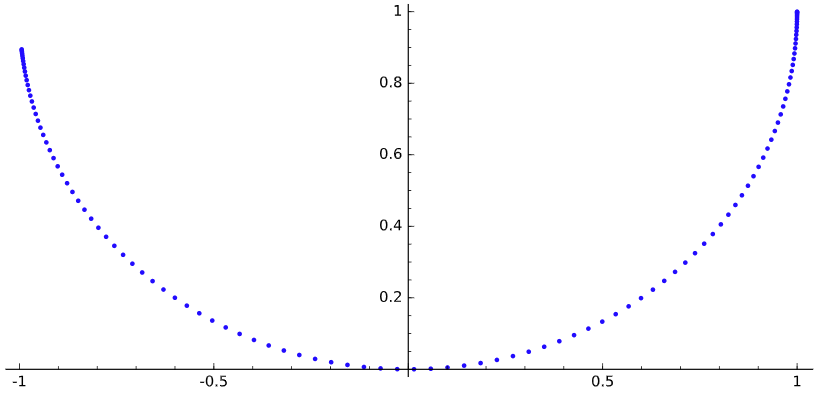
Introduction des frottements

V3 : un seul aller

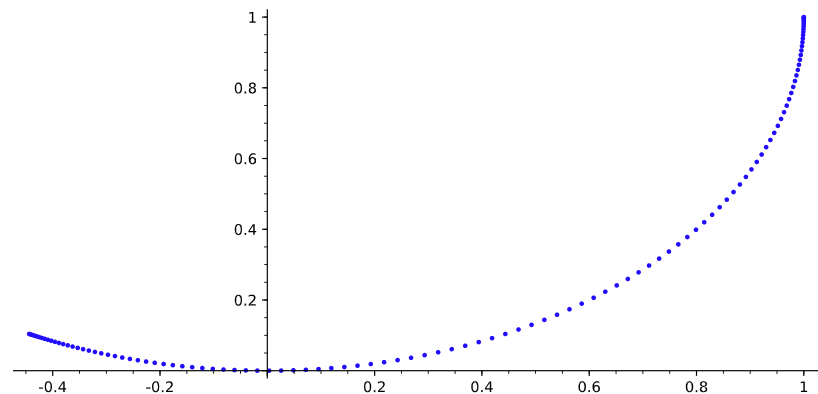
La formule utilisée pour obtenir

Nous avons introduit la programmation orienté objet lors de cette version, permettant de choisir les paramètres suivants : longueur de fil, angle de départ, masse du pendule, accélération de la pesanteur (g).

g = 9.81 (m/s) theta0 = pi/2 (radian) longueur fil = 1(m) masse = 1 (Kg) temps entre image = 0.1(s)



Voici une autre simulation avec une masse de 0.05Kg.

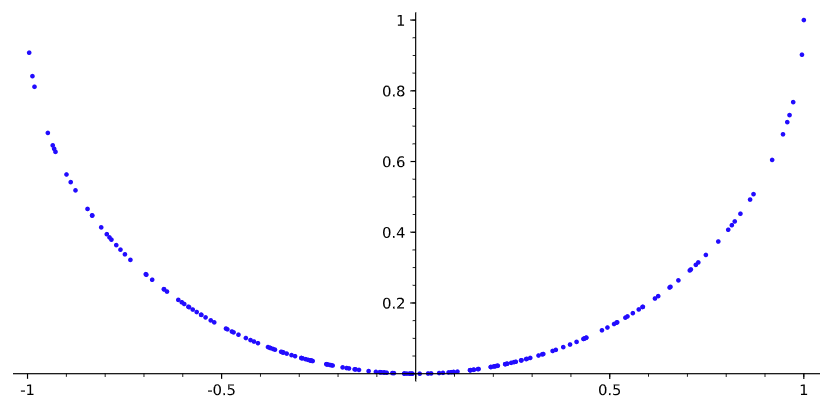


Introduction du trajet complet du pendule

V4

Ici on arrête la simulation quand le pendule atteint une vitesse trop petite, sinon celui-ci ne s’arrêterait jamais, à l’œil nu il s’arrête mais en réalité il ne s’arrête jamais et subit des perturbations qu’il est possible de négliger à notre échelle mais pas à l’échelle du nm.

g = 9.81 (m/s) theta0 = pi/2 (radian) longueur fil = 1(m) masse = 1 (Kg) temps entre image = 0.1(s)



On change g et on le met à 40 m/s

Ici le temps pour que le pendule ait une vitesse basse est très long donc beaucoup de calcul donc il faut attendre très longtemps, trop longtemps donc on ne peut pas tester cette configuration.

-------Passage sur python ------

Nous avons du refaire une partie du code notamment l’affichage en utilisant la bibliothèque TKinter

V5

A partir de là nous avons eu les problèmes du HTML, de la conversion de type et de l’exécution lente c’est pourquoi nous n’avons pas d’autres captures d’écran sur cocalc et que nous sommes sur PyCharm.