Mécanique lagrangienne et différences finies : Implémentation de la dynamique du pendule non-linéaire en `python`

Florence Bertails-Descoubes [*], Thibaut Métivet [†], Jean Jouve [‡]

Préambule
L'objectif primaire de ce premier TP (Exercice 1) est de simuler la dynamique d'un pendule simple en 2d, en s'appuyant sur les résultats et notions vus en cours :

- les équations (non-linéaires) du pendule simple, démontrées en cours;
- le schéma d'intégration Euler explicite, qui est rappelé ci-dessous.

Une analyse critique des résultats obtenus est également demandée.

Pour aller plus loin, l'Exercice 2 (en amorce du Projet à effectuer par les étudiants) propose d'explorer différents schémas d'intégration (Euler semi-implicite, Euler implicite), et les questions optionnelles pour chaque exercice proposent d'implémenter la dynamique du pendule double.

Code et bibliothèque `PyGLViewer`
Nous proposons aux étudiants d'implémenter ce TP en `python`. Ce choix n'est pas obligatoire mais fortement recommandé. Pour aider le travail de développement, nous fournissons un code de visualisation 3d, `PyGLViewer`, développé par Mickaël Ly[1] et Thibaut Métivet, et distribué librement sur la page https://gitlab.inria.fr/elan-public-code/pyglviewer.

Pour récupérer cette bibliothèque, il suffit de cloner son dépôt :

```
> git clone https://gitlab.inria.fr/elan-public-code/pyglviewer.git
```

Dans le répertoire principal, ouvrir le fichier `README.md` pour des explications sur une utilisation rapide de la bibliothèque (fichier d'exemple). Notamment, dans le répertoire `code`, l'exécution de:

```
> python main.py
```

devrait lancer une fenêtre de visualisation, avec une petite animation 2d.

Par défaut, la branche principale (`master`) de `PyGLViewer` pointe sur un affichage 2d. Pour passer à un affichage 3d (pas indispensable pour ce TP, mais nécessaire pour la prochaine séance sur les rigides 3d), il suffit de basculer sur la branche `feature/3d`:

```
> git checkout -b feature/3d -track origin/feature/3d
```

Lorsque la fenêtre de visualisation se lance, il est alors possible de faire tourner la caméra avec la souris (clic gauche + mouvement souris).

[*]florence.descoubes@inria.fr
[†]thibaut.metivet@inria.fr
[‡]jean.jouve@inria.fr

[1]Ce TP et le code qui correspond ont été initialement développés par Mickaël Ly dans le contexte d'un cours spécial donné avec F. Bertails-Descoubes et M. Skouras à l'ENS Lyon, en 2019 et 2020. Mickaël est un ancien étudiant de l'Ensimag (2014-2017) et ancien doctorant (2017-2021) de l'équipe Elan de l'Inria.

Rappel: le schéma d'Euler explicite
Soit f une fonction de $\mathbb{R}^+ \times \mathbb{R}^n$ dans $\mathbb{R}^n$ et soit le problème de Cauchy du premier ordre

$$\begin{cases} \forall t \in \mathbb{R}^+ \ : \ \dot{x}(t) = f(t, x(t)) \\ x(0) = x_0 \in \mathbb{R}^n \qquad\qquad\qquad\qquad \text{(condition initiale)} \end{cases}$$

Le but de l'intégration numérique est de calculer une solution approchée de ce problème (dont la solution exacte n'est a priori analytique) en découpant l'intervalle d'intégration en sous-intervalles. Plus précisément, si $x^* : \mathbb{R}^+ \to \mathbb{R}^n$ est la solution exacte du problème, nous cherchons une solution $x_n \approx x^*(t_n)$, avec $t_n = t_0 + n\,h$ et $h \in \mathbb{R}^+$ le pas d'intégration.

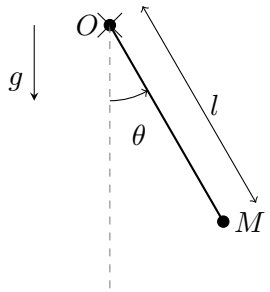
Le **schéma d'Euler explicite** approche la solution sur chaque sous-intervalle en utilisant un développement de Taylor à l'ordre un,

$$x(t_{n+1}) = x(t_n) + h\dot{x}(t_n) + o(h)$$

et donc, se construit par la formule itérative suivante

$$\boxed{x_{n+1} = x_n + hf(t_n, x_n)}.$$

Exercice 1 - Pendule simple - Schéma d'Euler explicite



Dans cet exercice, on se propose de simuler la dynamique d'un système mécanique, en utilisant le schéma d'integration d'Euler explicite.

On considère le pendule pesant simple : une masse M reliée à un point O par un fil inextensible et sans masse de longueur l, et soumise à la gravité.

(Q1) En cours, on a déjà démontré les équations qui régissent ce système mécanique. Les discrétiser en utilisant le schéma d'Euler explicite.

(Q2) Implémenter le modèle numérique résultant.

Remarque 1: On pourra jeter un coup d'oeil au fichier `scenes.py` ainsi qu'à la classe `DummyDynamicSystem` pour comprendre comment le code fonctionne.

Remarque 2: Lorsque l'on aura l'impression que le simulateur fonctionne, on pourra vérifier quantitativement que le simulateur se comporte de façon cohérente, en comparant son comportement aux petits angles avec la solution analytique du problème linéarisé.

(Q3) Comparer le comportement du simulateur pour différents pas de temps, avec les paramètres physiques suivants: $g = 9.81$, $l = 1$, $\theta_0(t = 0) = \frac{\pi}{5}$ et $M = 1$.

Remarque 3: On pourra utiliser `matplotlib` pour tracer l'évolution de l'angle $\theta(t)$.

(Q4) Étudier l'évolution de l'énergie mécanique du système au cours du temps (l'énergie $E_m(x_n)$ peut être calculée à chaque état du système). Que peut-on en conclure ?

(Q5) *(Optionnel)*Pour aller plus loin : implémenter et étudier la dynamique du pendule double.

Exercice 2 - Pendule simple - Schémas d'Euler semi-implicite et implicite

Le but est maintenant d'implémenter et de tester des schémas numériques plus avancés, toujours sur le cas du pendule.

Remarque 4: Rappel sur les trois schémas d'Euler :

Euler Explicite	Euler implicite	Euler semi-implicite/symplectique
$\dot{q}^{n+1} = \dot{q}_n + f(q^{\mathbf{n}}, \dot{q}^{\mathbf{n}})$	$\dot{q}^{n+1} = \dot{q}_n + f(q^{\mathbf{n+1}}, \dot{q}^{\mathbf{n+1}})$	$\dot{q}^{n+1} = \dot{q}_n + f(q^{\mathbf{n}}, \dot{q}^{\mathbf{n}})$
$q^{n+1} = q_n + h\dot{q}^{\mathbf{n}}$	$q^{n+1} = q_n + h\dot{q}^{\mathbf{n+1}}$	$q^{n+1} = q_n + h\dot{q}^{\mathbf{n+1}}$

Pour utiliser les mêmes notations qu'en cours, on peut noter $x \equiv \begin{bmatrix} q \\ \dot{q} \end{bmatrix}$.

(Q1) Faire une petite modification du code pour implémenter le schéma d'Euler semi-implicite (appelé aussi Euler symplectique).

(Q2) À nouveau, comparer le comportement du simulateur en fonction du pas de temps.

(Q3) Quelle est la condition de stabilité sur le pas de temps ?

(Q4) Implémenter le schéma d'Euler implicite **linéarisé** (voir remarque ci-dessous) et refaire les questions 2 et 3.

Remarque 5 - Linéarisation d'une équation non-linéaire: Au lieu de résoudre la vraie équation (non-linéaire) $x^{n+1} = g(x^{n+1})$, on suppose que $\delta x = x^{n+1} - x^n$ est "petit", ce qui nous permet de développer g au premier ordre autour de x^n,

$$x^{n+1} = g(x^n) + Dg(x^n)\delta x,$$

où $Dg(x^n)$ est la jacobienne de g évaluée en x^n. Ensuite, en remplaçant δx, on obtient une équation *linéaire* en x^{n+1}.

(Q5) Comment pourrait-on faire si l'on souhaite résoudre le problème non-linéaire à chaque pas de temps, et pas seulement sa version linéarisée ?

(Q6) Que peut-on dire de la stabilité de ces trois schémas ? (Tracer $\theta(t)$ ou l'évolution de l'énergie...).

(Q7) Conclure en comparant les trois schémas numériques.

(Q8) *(Optionnel)*Pour aller plus loin : implémenter et étudier la dynamique du pendule double avec Euler semi-implicite et Euler implicite linéarisé.