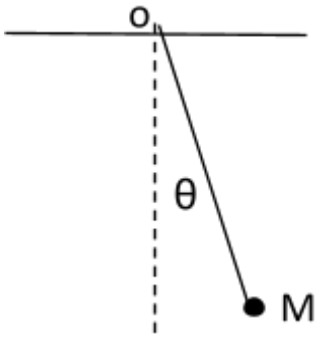
DS Informatique N°1

Classes : 2ème Année SM &SP
Durée : 1 H 30mn **Documents non autorisés**

✔ ***N.B. :Les exercices sont indépendants***

Exercice 1 : Période d'un pendule simple

Soit M un point matériel de masse m relié par une barre rigide de masse nulle et de longueur l à un point fixe o. Le système est soumis à la pesanteur et la position du point M est repérée par l'angle θ entre la verticale et la droite OM. Le vecteur d'accélération de la pesanteur est noté $\vec{g}$.



Le pendule est lâché avec une vitesse initiale nulle et avec un angle θ_0. La période du mouvement de M est donnée par la formule suivante:

$$T = 4\sqrt{\frac{l}{2g}}\int_0^{\theta_0} \frac{1}{\sqrt{coscos\,(\theta) - cos(\theta_0)}} d\theta$$

On prendra g=9.81ms^2, l=1m.

Dans ce problème, on cherche à implémenter des fonctions Python pour étudier la période du mouvement du système proposé.

Travail demandé:

1. Ecrire une fonction python **f(theta,theta0,g,l)** qui permet d'évaluer la fonction f_{θ_0} suivante:

$$f_{\theta_0} : \theta \rightarrow 4\sqrt{\frac{l}{2g}} \frac{1}{\sqrt{coscos\,(\theta) - cos(\theta_0)}}$$

2. Ecrire une fonction **PeriodRect(theta0,g,l,n)** qui permet de calculer la valeur approchée de la période du mouvement T par la méthode des rectangles.

La méthode des rectangles approche la valeur de l'intégrale par la somme des aires des n rectangles de base $[\theta_i, \theta_{i+1}]$ et d'hauteur $f_{\theta_0}(\theta_i)$ *pour i variant de* 0 à $n - 1$.

Ainsi, la période du mouvement T est approchée par la quantité R.

$$R = 4\frac{\theta_0}{n}\sqrt{\frac{l}{2g}}\sum_{i=0}^{n-1}\frac{1}{\sqrt{\cos\cos\left(\theta_i\right)-cos(\theta_0)}}\text{, avec}\theta_i = \frac{i\theta_0}{n}$$

On utilisera la fonction **f** définie précédemment.

3. Ecrire une fonction **PeriodRectquad(theta0,g,l)** qui permet de calculer la valeur approchée de la période du mouvement T en utilisant la fonction **quad** du module **scipy.integrate** de Python.

La fonction prédéfinie **quad (f,a,b,args)** du module **scipy.integrate** permet de calculer sous Python l'intégrale d'une fonction f sur l'intervalle [a,b]. L'argument **args** est un tuple composé des arguments de f à l'exception du premier.
La fonction **quad** retourne un tuple contenant la valeur de l'intégrale et une estimation de l'erreur maximale commise lors de l'intégration.

4. Représenter les courbes de variation de la période T en fonction deθ_0 ($\theta_0 \in]0,\pi[$) en utilisant les solutions approchées déterminées à partir de la question 2 ainsi que celles déterminées à partir de la question 3.

Pour cela:

- Importer le module **matplotlib.pyplot**.
- Construire une liste **Ltheta0**composée par 100 valeurs régulièrement espacées entre 0 et π (exclues).
- Construire les deux listes **Periode1** et **Periode2** composées par les résultats de calcul de la période du mouvement T pour les différentes valeurs de la liste **Ltheta0** en utilisant respectivement les fonctions **PeriodRect** et **PeriodRectquad**. On prendra **n=1000**.
- Utiliser la commande Python adéquate pour représenter les deux courbes de T.

N.B : l'utilisation des tableaux au lieu des listes est autorisée

Exercice 2 : Algorithme d'ordonnancement

En informatique, un processus est un programme en cours d'exécution auquel est associé un environnement processeur et un environnement mémoire. Un ordinateur possède forcement plusieurs processus en concurrence pour l'obtention du temps processeur. Ainsi, différents algorithmes d'ordonnancement existent pour choisir l'ordre dans lequel les différents processus seront exécutés.

Dans cet exercice, on s'intéresse à ordonner les éléments d'une file de processus. Chaque élément de la file est une liste qui contient le numéro du processus (unique), et son degré de priorité.

Le principe de cet ordonnancement se base sur la priorité. Le processus ayant la priorité la plus élevée sera placé en tête de la file.

Les fonctions nécessaires à la manipulation d'une file F sont supposées prédéfinies à savoir:

- creer_file() qui retourne une file vide.
- file_vide(F) qui retourne True si la file F est vide et False sinon.

- enfiler(F,elt) qui permet d'ajouter un élément en queue de file.
- defiler(F) qui permet de supprimer l'élément en tête de file et de le retourner. Si la file est vide cette fonction retourne 0.
- taille_fille(F)qui permet de retourner la taille de la file F.

Soit F une file de processus. L'objectif étant de créer une file FP de processus ordonnés selon l'ordre décroissant de la priorité.

Travail demandé:

1- Ecrire une fonction Python **Permutation_Circulaire*(F,n)*** qui permet de réaliser n permutations circulaires sur les éléments de la file F.Une permutation circulaire agit comme un décalage de telle manière que chaque élément occupera la position suivante dans la file (l'élément en tête devient en queue, le deuxième devient en tête etc.…)

2- Ecrire une fonction Python**Inserer_Elt(F,L)**qui permet d'insérer un élément L (L est une liste qui contient le numéro du processus et son degré de priorité) dans une file F ordonnée selon l'ordre décroissant de la priorité de ses éléments sans utiliser une file intermédiaire et en faisant appel à la fonction **Permutation_Circulaire**.

3- Ecrire une fonction Python**Ordonner_file(F)** qui permet, à partir d'une file F, de retourner une file FP ordonnée selon l'ordre décroissant de la priorité des éléments.

4- Ecrire une fonction **Afficher_file(F)** qui permet d'afficher les numéros des processus de la file F. Après l'exécution de cette fonction, la file F reste inchangée.

Bon travail