Devoir 1 (H26)

MAT-2910 Analyse numérique pour l'ingénierie

Département de mathématiques et statistique de l'Université Laval

Directives (Python)

0 Consignes

- Vous pouvez former des équipes d'**au plus** 2 personnes. Il faut créer l'équipe sur monPortail avant la date limite indiquée (**même si on est seul!**).
- Pour pouvez utiliser Matlab ou Python (prendre l'énoncé correspondant)
- Plusieurs fonctions de Python vous sont proposées dans l'énoncé. N'hésitez pas à utiliser Google pour consulter la documentation.
- Vous n'avez pas droit à l'aide du CDA pour ces exercices. Vous pouvez utiliser le forum, demander à l'auxiliaire du cours ou à l'enseignant.
- Vous devez remettre un script `equipeX.py` (X étant votre numéro d'équipe) qui affiche toutes les figures demandées lors de son exécution, et tous les autres fichiers de codes demandés.
- Vous devez remettre un dossier compressé **en format .zip** par équipe nommé `EquipeX.zip`, où X est votre numéro d'équipe. Ce dossier devra contenir un **rapport écrit au traitement de texte** (*e.g.* MS Word, LaTeX) **converti en format .pdf** contenant tout ce qui est demandé pour les exercices (calculs, figures, explications, valeurs numériques, etc.). Le dossier contiendra aussi votre code **au format .py**.
- Des points sont attribués à la présentation (propreté, qualité des figures, identification des axes, titres, **légendes**, couleurs). N'oubliez donc pas d'en tenir compte lorsque vous incluez des figures au rapport.
- Votre travail doit être remis sur la boîte de dépôt monPortail avant la date limite indiquée sur le site de cours.
- En cas de retard, une pénalité de 5%/heure s'applique lors des premières 24h, jusqu'à concurrence de 20%. Après la première journée de retard, 20%/jour supplémentaire sera enlevé.

1 Questions

1.1 Prérequis

Vous aurez besoin des lignes suivants au début de votre fichier Python `equipeX.py` :

```
import numpy as np
import matplotlib.pyplot as plt
from suiteSn import suiteSn
```

La dernière ligne vous permettra d'utiliser dans votre fichier script la fonction que vous définirez dans la question 2. Par ailleurs, le fichier `suiteSn.py` que vous créérez à la question 2 aura les importations suivantes :

```
import numpy as np
from math import e
```

1.2 Exercices

La première question permet de se familiariser avec les principales commandes Python qui seront utiles pour tous les devoirs à venir (incluant celui-ci). On pourra consulter le (court) guide de M. LeMesurier et M. Roberts (contributions) vers lequel un lien a été placé sur le site web du cours, dans la page d'introduction (c'est le premier guide Python suggéré dans la liste).

1. Créer un fichier script Python dont le nom contiendra le numéro de votre équipe , `equipeX.py`. Dans ce fichier, on commencera par écrire les lignes de commandes qui définiront les nombres, vecteurs et matrices suivants et qui les afficheront sur la fenêtre de commandes de l'IDE Python. Attention, sauf indication contraire, il faut 1 seule ligne de commandes pour chaque élément demandé.

    - a, le vecteur colonne de longueur 6 dont toutes les composantes valent 1. Utiliser pour cela la commande Python *np.full*.
    - b, le vecteur ligne de longueur 6 dont les composantes valent 1, 2, ..., 6 (en une seule ligne). Utiliser pour cela la commande Python *np.arange*.
    - c, le vecteur ligne de longueur 6 dont toutes les composantes valent 1. On obtiendra c à partir de a grâce à la transposition (fonction *np.reshape* ou méthode *reshape*).
    - d, le vecteur ligne de longueur 6 dont toutes les composantes valent le numéro de votre équipe. On obtiendra d en multipliant c par le numéro de votre équipe.
    - I, la matrice identité de taille 6×6. On utilisera la commande *np.identity*.
    - J, la matrice de taille 6×6 dont toutes les composantes valent 1. Utiliser pour cela la commande *np.full*.
    - K, la matrice de taille 6×6 dont la diagonale est le vecteur b. Utiliser pour cela la commande *np.diag*.

- L, la matrice de taille 6×6, définie par $L = 55\,I - J + 2\,a * c$. On obtiendra L à partir de I, J, c et a, bien sûr.
- M, la matrice obtenue en remplaçant la 1ère colonne de K par a. On obtiendra M à partir de K et a, bien sûr, en commençant votre ligne de commande par : *M=K; M[:,0]=...* (à compléter, utiliser *reshape*).
- *dd*, le déterminant de M (utiliser la commande *np.linalg.det*).
- x, la solution de $Mx = a$, sans calculer l'inverse de L (utiliser la commande *np.linalg.solve*).
- N, la matrice solution de $M * N = M'$, où M' est la matrice transposée de M (utiliser la commande *np.linalg.solve* pour résoudre et la méthode `.T` pour transposer).

Sur la fenêtre de commandes, on verra donc s'afficher (utiliser des *print*):
a =

...

b =

...

et ainsi de suite ....

À la suite, dans ce fichier script, créer une figure (Figure 1) visualisant la matrice N. On utilisera la commande *plt.matshow*. On écrira en titre, grâce à la commande *plt.title*, qu'il s'agit de la matrice N.
Mettre la figure dans le rapport (*plt.show* pour afficher dans l'IDE Python).

Ensuite, toujours dans le fichier script, définir la fonction $f(x) = -\frac{x^2}{2} + e^x + \sin x$ de façon à ce qu'elle permette d'être appliquée à un vecteur x, en utilisant la syntaxe (à compléter):

```
def nom_fonction(variable_a_entrer)
    code...
    return variable_a_retourner
```

Rappel: il est possible de tout évaluer en une ligne en utilisant les fonctions *np.exp* et *np.sin* de Python.

Faire tracer le graphe de f sur l'intervalle $[0, 1]$, en évaluant la fonction aux abscisses $0, 0.01, 0.02, \cdots, 0.99, 1$. On définira pour cela le vecteur x, contenant les abscisses des points, par $x = np.linspace(0, 1, 101)$. Puis utiliser la commande *plt.plot(x,nom_fonction(x))* pour tracer le graphe.
Mettre un titre à la figure avec la commande *plt.title*. Cela constituera la Figure 2.
Mettre la figure dans le rapport (*plt.show* pour afficher dans l'IDE Python).

2. **Instabilité numérique**

Soit $S_n = \int_0^1 x^n e^x dx$.

On aimerait calculer S_n quelle que soit la valeur de n. On voit bien que $S_0 = e - 1$. Mais pour des valeurs de $n \geq 1$ il faut faire n intégrations par parties pour obtenir la valeur de S_n. Un autre moyen est d'utiliser une formule de récurrence et de programmer cet algorithme avec une boucle.

(a) On peut voir que $S_n \geq 0$ quel que soit n. On peut également montrer que $S_n \longrightarrow 0$ lorsque $n \longrightarrow +\infty$.
Dans le rapport, démontrer la formule de récurrence suivante en utilisant une intégration par parties: $S_n = e - nS_{n-1}$.

(b) Créer un fonction Python, appelée *suiteSn*, dans un nouveau fichier *suiteSn.py*, dont la seule entrée est n et la sortie est un vecteur ligne (*np.array*) S de longueur $n + 1$, dont les composantes sont les valeurs S_0, S_1, ..., S_n. On utilisera une boucle *for* et la commande *np.append* pour construire les éléments de la suite S_n en utilisant la formule de récurrence précédente. Outre les *import*, ce fichier Python commencera par les 2 lignes suivantes:

```
def suiteSn(n)
    S = np.array([e-1])
    ...
```

(c) Dans le fichier script, faire appel à la fonction *suiteSn* créée précédemment pour calculer $S_1, ..., S_{19}$. Dans une nouvelle figure (Figure 3), créer un graphique montrant les valeurs de S_n en fonction de n, pour $n = 0, 1, ..., 19$ (Figure 3). On utilisera la commande *plt.plot* pour le graphique. On utilisera également les commandes *plt.xlabel*, *plt.ylabel*, *plt.title* pour rendre la figure très explicite.
Mettre la figure dans le rapport (*plt.show* pour afficher dans l'IDE Python).

(d) Dans le rapport, expliquer pourquoi on obtient des valeurs surprenantes pour S_{18} et S_{19}. À cette fin on exprimera ΔS_n en fonction de ΔS_{n-1} en utilisant la formule de propagation de l'erreur vue en cours.

3. **Opération dangereuse**

Pour une fonction $f(x)$ donnée, on sait qu'une approximation de $f'(x)$ est donnée par la formule

$$D(x, h) = \frac{f(x + h) - f(x)}{h}$$

où h est supposé positif. Plus h est petit et plus $D(x, h)$ est proche de $f'(x)$. On a d'ailleurs $D(x, h) \longrightarrow f'(x)$ lorsque $h \longrightarrow 0$.

(a) Dans le script Python, caluler l'erreur $|f'(0) - D(0, h)|$, pour la fonction $f(x)$ de la question 1, et pour les valeurs suivantes de h: $10^{-1}, 10^{-2}, \ldots, 10^{-12}$ et faire tracer le graphique (Figure 4) de l'erreur en fonction de h avec des échelles logarithmiques (au lieu de la commande *plt.plot*, pour avoir des échelles logarithmiques à la fois en abscisse et en ordonnée, on utilisera la commande *plt.loglog*). Bien identifier les axes et mettre un titre au graphique (commandes *plt.xlabel*, *plt.ylabel*, *plt.title*).
Mettre la figure dans le rapport (*plt.show* pour afficher dans l'IDE Python).

(b) Dans le rapport, expliquer pourquoi, alors qu'on s'attend à ce que $D(x,h) \longrightarrow f'(x)$ lorsque $h \longrightarrow 0$, l'erreur devient bizarre pour les plus petites valeurs de h. (Indice: voir titre de cette question).

4. **Ordre d'un polynôme de Taylor**

Soit la fonction

$$f(x) = x^{9/4} + (1+x)^2$$

dont le domaine de définition est $[0, \infty)$.

(a) Dans le rapport, déterminer le polynôme de Taylor de degré 1, noté $p_1(h)$, de la fonction f, en $x_0 = 0$, ainsi que $R_1(h)$, le reste. Montrer que cette approximation est bien d'ordre 2 en trouvant une majoration de la forme $E_1(h) \leq C|h|^2$. Remarque: on a forcément $h > 0$.

(b) Similairement, calculer $p_2(h)$ et $R_2(h)$. Peut-on majorer le terme d'erreur par $C|h|^3$ pour une certaine constante C ? Justifier.

(c) En calculant explicitement $|p_2 - f|$ (sans utiliser l'expression du reste), montrer que p_2 est une approximation d'ordre $9/4$.