Université Paris Est - Créteil Val de Marne
Faculté des Sciences et Technologie
M1 Mécanique
Année 2024-2025, 1er semestre
Intervenants: Felipe ROCHA et Sara TOUHAMI
Emails: `felipe.figuereredo-rocha@u-pec.fr` et `sara.touhami@u-pec.fr`

Analyse numérique et calcul scientifique - 1

TP 1 (3h)

Interpolation de données

Objectifs

- Lire, comprendre et utiliser un code Python complexe (script et fonctions).
- Se familiariser avec les méthodes d'interpolation par polynôme de Lagrange et spline du 1er ordre, en appréhender les potentialités et les limites.
- Communiquer des résultats scientifiques.

Introduction

Les méthodes d'interpolation permettent de construire des fonctions interpolant un ensemble de points dits nœuds de l'interpolation. Notant $\{(x_0, y_0), (x_1, y_1), \ldots, (x_N, y_N)\}$ ces nœuds, la fonction d'interpolation $\varphi(x)$ peut alors s'écrire comme une combinaison linéaire de $N+1$ fonctions de bases $\Phi_i(x)$ $(i = 0..N)$:

$$\varphi(x) = y_0\,\Phi_0(x) + y_1\,\Phi_1(x) + \ldots + y_N\,\Phi_N(x) = \sum_{i=0\ldots N} y_i\,\Phi_i(x)\,, \qquad (1)$$

Chaque fonction de bases $\Phi_i(x)$ a la propriété suivante:

$$\Phi_i(x_j) = \delta_{ij}\,, \quad i, j \in 0\ldots N. \qquad (2)$$

La fonction $\varphi(x)$ aura alors la propriété de passer par les nœuds de l'interpolation, soit:

$$\varphi(x_j) = y_j\,, \quad j \in 0 \ldots N. \tag{3}$$

L'expression des fonctions de base change d'une méthode d'interpolation à l'autre.

Interpolation par polynôme de Lagrange. Les fonction de bases de Lagrange sont des polynômes d'ordre N s'écrivant:

$$\Phi_i^{\text{Lagrange}}(x) \quad \equiv \quad L_i(x) = \prod_{\substack{j=0..N \\ j\neq i}} \frac{x - x_j}{x_i - x_j}\,. \tag{4}$$

La fonction d'interpolation est un polynôme d'ordre N dit polynôme de Lagrange:

$$\varphi^{\text{Lagrange}}(x) \quad \equiv \quad p(x) = \sum_{i=0\ldots N} y_i\, L_i(x)\,. \tag{5}$$

Interpolation par fonctions spline d'ordre 1. Les fonction de bases spline d'ordre 1 sont des polynômes d'ordre 1 par morceaux s'écrivant:

$$\Phi_i^{\text{Spline-1}}(x) \quad \equiv \quad S_i(x) = \begin{cases} \dfrac{x - x_{i-1}}{x_i - x_{i-1}}, & \text{si } x \in [x_{i-1}, x]; \\ \dfrac{x - x_{i+1}}{x_i - x_{i+1}}, & \text{si } x \in [x, x_{i+1}]; \\ 0, & \text{sinon.} \end{cases} \tag{6}$$

La fonction d'interpolation est un polynôme d'ordre 1 par morceaux:

$$\varphi^{\text{Spline-1}}(x) \quad \equiv \quad s(x) = \sum_{i=0\ldots N} y_i\, S_i(x)\,. \tag{7}$$

Description du problème

Les résultats d'une expérience sont enregistrés dans le fichier texte `Data0.txt` sous forme de deux colonnes des valeurs, correspondant au temps et à la valeur mesurée (chaque ligne correspond à un point de mesure). Des résultats typiques sont montrés sur la figure **??**.

On vous demande d'utiliser deux méthodes d'interpolation, celle des polynômes de Lagrange et celle des fonctions spline d'ordre 1, pour interpoler ces données en se basant sur un nombre réduit de points dits *nœuds* de l'interpolation. Pour ce faire, vous utiliserez les fonctions `Python` qui vous sont fournies, notamment les fonctions `fStruct_LagrPoly.py` et `fStruct_Spline1.py`, en les pilotant *via* le fichier script `mainTP1.py`.

Travail à réaliser

Vous devez étudier les fonctions et le script `Python` fournis pour vous les approprier, puis les utiliser pour étudier les performances des deux méthodes d'interpolation en faisant varier le nombre et la position des nœuds.

Objectif: Lire, comprendre et utiliser un code Python complexe (script et fonctions).

Vous commencerez par renommer le script comme `<CODE>_mainTP1.py`, où `<CODE>` est l'identifiant de votre groupe de TP (exemples: `<CODE>` = `GrA1` pour le groupe $A/1$, `<CODE>` = `GrB2` pour le groupe $B/2$ ...).

Dans le script, tout d'abord vous lirez les valeurs de toutes les mesures effectuées à partir du fichier `Data0.txt` en les enregistrant dans les vecteurs `X` et `Y`. À partir de ces vecteurs, vous extrairez deux vecteurs `Xi` et `Yi` contenant les coordonnées des nœuds à utiliser pour le calcul des fonctions d'interpolation.

Ces calculs sont effectués par les fonctions `fStruct_LagrPoly.py` et `fStruct_Spline1.py`. Celles-ci prendront comme arguments `Xi`, `Yi` et `X` pour restituer comme résultat une variable `Python` de type *structure*[1]. Par exemple, elles pourront être utilisées comme suit:

```
L = fStruct_LagrPoly(Xi,Yi,X)
S = fStruct_Spline1(Xi,Yi,X)
```

Les *structures* `L` et `S` contiennent toutes les informations utiles pour analyser les résultats d'un calcul, chaque information étant enregistrée dans un *champ* différent. À titre d'exemple, la fonction `fStruct_LagrPoly` est organisée

[1] Une *structure* `Python` est une variable contenant un ou plusieurs *champs*. Chaque champ est une variable de n'importe quel type (réel, vecteur, matrice, chaîne de caractères, structure ...). On accède à un champ en utilisant la notation `<NOM_STRUCTURE>.<NOM_CHAMP>`.

comme ci-dessous:

```
def L = fStruct_LagrPoly(Xi,Yi,X):
    ⋮
    L.Xi = Xi
    L.Yi = Yi
    ⋮
    L.X = X
    <Calcul de L.Y>
    ⋮
```

On peut remarquer que les champs `Xi`, `Yi` et `X` sont des vecteurs identiques aux arguments passés à la fonction et que le champ `Y` est un vecteur contenant les valeurs de la fonction d'interpolation (polynôme de Lagrange ou spline) sur l'ensemble des points contenus dans le vecteur `X`. Vous prendrez le temps d'étudier les fonctions qui effectuent le calcul de `L.Y` selon les deux méthodes proposées.

Remarque: L'étude de ces dernières fonctions pourra être réalisée hors présentiel.

Vous étudierez la précision des méthodes en calculant les erreurs d'interpolation d'une part sur l'ensemble des points `X` (variables `L_errX` et `S_errX`) et d'autre part sur les nœuds `Xi` (variables `L_errXi` et `S_errXi`). Pour cela, vous pourrez utiliser la fonction `norm` de `Python`, qui prend comme argument un vecteur `U` et rend la valeur de sa norme. Ainsi, vous aurez par exemple:

```
L_errX  = norm(Y -L.Y)
L_errXi = norm(Yi-L.Y(ii))
```

où `ii` est un vecteur d'entiers positifs contenant les indices des nœuds de l'interpolation.

Objectif: Se familiariser avec les méthodes d'interpolation par polynôme de Lagrange et spline du 1er ordre, en appréhender les potentialités et les limites.

Vous testerez les performances de chacune des deux méthodes d'interpolation en faisant varier *le nombre* et *la position* des nœuds. Il est conseillé d'utiliser au moins 5 ensembles de nœuds (`Xi`,`Yi`) différents. Pour chaque ensemble de nœuds, vous calculerez à la fois le polynôme de Lagrange et la fonction spline d'ordre 1 associés, ainsi que les temps de calcul et les erreurs correspondantes. Les variables `L`, `S`, `L_errX`, `S_errX`, `L_errXi` et `S_errXi` seront alors des vecteurs à n éléments.

Vous passerez ensuite à la représentation graphique des résultats de vos calculs :

- Vous tracerez les courbes correspondant aux données initiales (`X`,`Y`) et à toutes les fonctions d'interpolation calculées (soit `L(k).X`,`L(k).Y` d'une part et `S(k).X`,`S(k).Y` d'autre part, avec $k = 1 \dots n$) en prenant soin d'utiliser des styles de ligne différents pour chaque courbe et d'ajouter une légende à la figure.
- Vous tracerez les erreurs commises sur les points `X` en fonctions de k.
- Vous tracerez les erreurs commises sur les nœuds `Xi` en fonctions de k.

Pour ces trois dernières figures, vous pourrez utiliser une représentation par courbes (fonction `plot`) ou par histogrammes (fonction `bar`). En outre, le cas échéant, vous pourrez utiliser une échelle logarithmique pour l'axe vertical (notamment pour la représentation des erreurs d'interpolation).

Le code `Python` nécessaire pour effectuer les calculs, tracer les courbes et enregistrer les figures sera contenu dans le script `<CODE>_mainTP1.py`. Il devra être possible d'exécuter ce script sur un ordinateur avec une installation standard de `Python` (il est déconseillé d'utiliser des fonctions `Python` liées à des Toolbox).

Vous pourrez répéter vos expériences numériques en utilisant d'autres fichiers de données ou en utilisant vos propres données:

- Fichier `Data0.txt`: courbe de relaxation
- Fichier `Data1.txt`: problème de Runge
- Fichier `Data2.txt`: vibration amorties
- Fichier `Data3.txt`: vibration non-amorties
- Fichier `Data4.txt`: droite
- Fichier `mainCreateDataFile.py`: pour générer vos propres données

Remarque: L'étude d'autres fichiers de données pourra être réalisée hors présentiel.

Objectif: Communiquer des résultats scientifiques.

Vous rédigerez un compte-rendu scientifique faisant état de vos expériences numériques et de vos acquis. Le compte-rendu doit être rédigé avec soin, de façon à la fois claire et synthétique, en incluant dans le corps du rapport les informations principales et en renvoyant aux annexes pour celles secondaires. Idéalement, la structure du rapport sera la suivante:

- Page de garde

- Table des matières
- Introduction
- Matériels et méthodes
- Résultats de la méthode d'interpolation par polynômes de Lagrange
- Résultats de la méthode d'interpolation par fonctions spline du 1er ordre
- Discussion
- Conclusion
- Annexes

Dans l'introduction vous présenterez brièvement le ou les problèmes étudiés. Dans la section "Matériels et méthodes" vous présenterez les deux méthodes étudiées (une recherche bibliographique sera la bienvenue) ainsi que les paramétrages utilisés dans vos calculs (en prenant soin de justifier vos choix). Dans chaque section "Résultats ..." vous présenterez (sans discuter) les résultats obtenus par la méthode considérée en fonction des paramétrages effectués. Dans la section "Discussion", vous discuterez ces résultats pour tracer un bilan des points forts et faibles des deux méthodes. Dans la section "Conclusion", vous dresserez un bilan synthétique de ce TP en termes d'acquis personnels par rapport aux objectifs prévus. Vous fournirez en annexe tout matériel complémentaire qui pourra faciliter l'appréciation de votre travail.

Travail à rendre

Le travail de ce TP devra être envoyé par email (aux deux intervenants!) au plus tard **deux semaines après la fin du TP**. Le travail à rendre est un dossier comprimé (archive `.zip`, `.rar`, ...) nommé `NOM1_NOM2_GPX_TP1` et contenant :

- Un rapport de TP en format `.pdf` : pour générer ce pdf nous vous proposons trois options de templates:
    1. **(recommandé)** Latex nommé `template_rapport_latex.tex`: Si vous n'avez jamais installé sur votre ordinateur, nous vous conseillons la version en ligne `https://www.overleaf.com/`, très simple à utiliser.
    2. Notebook Jupyter nommé `template_rapport_jupyter.ipynb`: Sur Jupyter même vous pouvez écrire des textes et formules à l'aide de la cellule de texte. Pour les formules il suffit d'écrire

les commandes Latex entre `$ ... $` ou `$$ ... $$`. Une bonne référence pour trouver ces commandes est le site `https://editor.codecogs.com/`. Finallement, il vous suffit d'exporter ce fichier en format `.pdf`. L'avantage de cette approche

- Un sous-dossier nommé `Numerics` avec tous les fichiers utilisés et éventuellement créés lors du TP (données, script et fonctions `Python`, figures, ...).