TE 1-B : Dérivation numérique

1 Objectif de l'étude

1.1 Dérivation par différences finies

On s'intéresse à la dérivation numérique d'une fonction f **en un point** t_0 **fixé** via différents schémas aux différences finies de pas h, et au **choix du pas** h permettant de minimiser les erreurs.
Plus précisément, on étudie des schémas à deux termes pour la dérivée première :

— Estimation décentrée à droite

$$f'(t_0) \approx f'_{dd}(t_0) = \frac{f(t_0+h) - f(t_0)}{h} \tag{1}$$

— Estimation décentrée à gauche

$$f'(t_0) \approx f'_{dg}(t_0) = \frac{f(t_0) - f(t_0-h)}{h} \tag{2}$$

— Estimation centrée

$$f'(t_0) \approx f'_{c}(t_0) = \frac{f(t_0+h) - f(t_0-h)}{2h} \tag{3}$$

1.2 Erreurs associées aux estimateurs

Deux types d'erreurs indépendantes interviennent dans ces approximations et s'ajoutent en valeur absolue :

— L'**erreur de troncature** déterministe, liée au nombre limité de termes du schéma aux différences finies. On peut l'estimer à partir d'un développement en série de Taylor avec reste : $h|f^{(2)}(t_0)|/2$ pour les estimateurs décentrés (1 et 2), $h^2|f^{(3)}(t_0)|/6$ pour l'estimateur centré (3).

— L'**erreur d'arrondi** pseudo-aléatoire, qui dépend de la précision relative ε des `float`, que l'on choisira pour le calcul. Sa borne supérieure décroît avec le pas en $1/h$: $2|f(t_0)|\varepsilon/h$ pour les estimateurs décentrés (1 et 2) et $|f(t_0)|\varepsilon/h$ pour l'estimateur centré (3).

1.3 Recherche graphique du pas minimisant l'erreur totale

Les valeurs absolues de ces erreurs dépendent du pas selon des lois en puissance h^p : $p < 0$ pour l'arrondi et $p > 0$ pour la troncature. L'erreur totale est donc dominée par l'erreur d'arrondi à pas très faible et par l'erreur de troncature à grand pas. Le tracé de l'erreur totale en fonction du pas permet de mettre en évidence ces domaines respectifs et de vérifier graphiquement les lois en puissance.
À cet effet, on choisit une représentation en échelles log-log ; pour répartir régulièrement les points sur l'axe des pas, les valeurs du pas sont choisies selon une **progression géométrique** de raison r : si n est le nombre de valeurs du pas, r est donné par $r^{n-1} = h_{\max}/h_{\min}$.
L'objectif final est d'estimer graphiquement la valeur du pas minimisant l'erreur totale d'estimation.

1.4 Fonctions test

Estimer, en simple précision, la dérivée de $f(t) = \sin(kt)$ en $t_0 = \pi/4$, pour, par exemple, $k = 1$ avec $10^{-5} \leqslant h \leqslant 1$. (+)Dans la mesure du temps disponible, reprendre le calcul avec $f(t) = \exp(kt)$ en $t_0 = 0$.

2 Partie calcul en C++

2.1 Organisation des codes

La structure générale doit permettre la compilation séparée des quatre fichiers via `make` :

1. Un programme principal (fichier `main.cpp`) auquel est confié le dialogue avec l'utilisateur pour choisir le nombre `n` de valeurs du pas, le plus petit pas `hmin` et le plus grand pas `hmax`, l'estimateur de dérivation numérique utilisé, (+)et enfin la fonction à étudier. Ce programme calcule la dérivée $f'(t_0)$ avec l'expression analytique. Il alloue ensuite le tableau `h` des pas et celui `delta` des erreurs. Pour tabuler l'erreur en fonction du pas, il lance alors une boucle sur les pas dans laquelle :
   — il calcule le pas et le stocke dans `h`
   — il appelle l'estimateur numérique (+)choisi par l'utilisateur

— il calcule la valeur absolue de l'écart avec la valeur de l'expression analytique $f'(t_0)$ de la dérivée et la stocke dans le tableau des erreurs `delta`.

Il fournit enfin les tableaux de résultats à la procédure d'écriture sur fichier.

2. Un fichier (`fcts.cpp`) définissant les différentes fonctions à tester ainsi que leurs dérivées analytiques.
3. Un fichier (`methode.cpp`) hébergeant les fonctions `deriv2tdd`, `derriv2tdg` et `deriv2tc` qui évaluent la dérivée de la fonction test `f` à l'abscisse passée en argument. Comme ces opérateurs évaluent f en plusieurs points autour de t_0, il est nécessaire de leur passer un argument de type fonction.
4. Une fonction `ecrit` chargée d'écrire les résultats dans un fichier formaté.

Fichiers des fonctions à tester
`fcts.cpp`

Définition des différentes fonctions à tester.
Un seul argument pour chacune des fonctions ;
les paramètres éventuels sont des variables globales du fichier.

Fichier de la méthode à appliquer
`methode.cpp`

Arguments des estimateurs numériques :

— la fonction à tester
— l'abscisse du point où on calcule les dérivées
— le pas h

Programme principal
`main.cpp`

— Saisie des paramètres
— Construction des vecteurs
— (+)Choix de la fonction à tabuler et de l'estimateur numérique
— Calcul de la dérivée analytique en t_0
— Boucle sur les pas avec remplissage du tableau des pas, appel de la méthode d'estimation numérique de la dérivée et remplissage du tableau des écarts à la dérivée analytique
— Appel de la procédure d'écriture des résultats

Fichier d'écriture du fichier de résultats
`util.cpp`

— Ouverture du fichier
— (+)Écriture de l'entête avec valeurs extrêmes
— Boucle d'écriture des h et des écarts correspondants, à raison d'un couple par ligne
— Fermeture du fichier

2.2 Structure du fichier de résultats

Le **corps du fichier** comportera une ligne par valeur du pas, ligne constituée de la valeur du pas suivie de la valeur absolue de l'erreur. On spécifiera un format compatible avec la dynamique des valeurs et assurant une précision relative de l'ordre de 10^{-7}.
(+)Le fichier de résultats comportera également deux lignes d'**entête** indiquant respectivement :
— le nombre de valeurs du pas et les valeurs minimale et maximale du pas ;
— les bornes des valeurs absolues des erreurs (valeurs minimale et maximale) ;
Cet entête ne sera implémenté qu'après un premier test et une visualisation.

2.3 Exemple d'interface des méthodes de dérivation

extrait de methode.h

```
#ifndef METHODE_H
#define METHODE_H
#include <functional>
float deriv2tdd(
  std::function<float(const float)>,
  const float, const float);
#endif
```

extrait de util.h

```
#ifndef UTIL_H
#define UTIL_H
#include <Eigen/Dense>
#include <string>
void ecrit(const std::string &,
  const Eigen::VectorXf &, const Eigen::VectorXf &);
#endif
```

3 Affichage en Python

On adaptera le script Python fourni au précédent TE pour traiter les fichiers de résultats produits pendant ce TE. En particulier, pour tracer en échelles log sur les deux axes, on utilisera la fonction `loglog` de `matplotlib.pyplot` à la place de `plot`.