Introduction

Le but de cette session est de se familiariser avec les bases d'un environnement `C++` pour le calcul scientifique. Tous les outils utilisés sont libres et peuvent être installés sur des machines personnelles.

Vous êtes libres d'utiliser l'environnement de développement de votre choix. Je recommande d'utiliser `Visual Studio Code`.

1 Débogage

Le programme fourni se compose de quatre fonctions :

- `initialization` : construit une matrice carrée à partir de sa taille
- `fill_vectors` : remplie un vecteur de dimension n, donné en entrée, avec des nombres aléatoires compris entre -10 et 10.
- `print_matrix` : affiche la matrice à partir de sa taille.
- `trace` : réalise la somme des éléments diagonaux de la matrice.

Le code proposé utilise les bases du `C` et quelques éléments de `C++` vus lors du premier cours. Le `CMakeLists.txt` proposé permet de compiler et faire l'édition de lien du fichier `trace.cxx`.

Pour l'utiliser

1. Créer une répertoire `build`.
2. Placer vous dans le répertoire `build`.
3. Générer les `Makefile` avec la commande `cmake ..`.
4. Créer l'exécutable avec la commande `make`.

Question 1.
Identifier les problèmes de ce code.

Question 2.
Corriger les différents problèmes.

Question 3.
Faire un profiling du code.

Question 4.
Proposer une modification du code qui peut améliorer les performances.

2 Résolution d'EDO

Dans ce premier exercice, nous allons écrire une méthode de résolution d'une équation différentielle du premier ordre :

$$\frac{du}{dx} = \varphi(x, u(x)) \text{ sur } I \tag{1}$$

$$u(a) = u_a \tag{2}$$

avec $\varphi \in C^0(I \times \mathbb{R}, \mathbb{R})$, $I \subset \mathbb{R}$.

On support l'existence et l'unicité d'une solution u pour $x \in I$.

Pour $h > 0$, on définit $x_n = a + nh$, $h = 0 \dots N_h$ une suite de point de noeuds de I qui forme une discrétisation de l'intervalle en sous intervalles $I_n = [x_n; x_{n+1}]$ de longueur h.

Pour chaque noeud x_n, nous souhaitons à approcher la valeur inconnue $u(x_n)$.

2.1 Euler explicite

On construit une solution numérique de l'équation différentielle en chaque noeud x_n en utilisant le taux d'accroissement

$$u'(x_n) = \frac{u(x_{n+1}) - u(x_n)}{h}. \tag{3}$$

On considère la fonction $\varphi(x, u(x)) = 2xu(x)$ définie sur $I = [0, 1]$ avec $u(0) = 1$.

Écrire, compiler et exécuter un programme écrit en `C++` qui calcule la solution de l'équation d'Euler explicite. La fonction utilise en paramètre d'entrée le nombre d'itération, la condition initiale et la fonction φ.

2.2 Euler implicite

Cette fois, on construit une solution numérique de l'équation différentielle en chaque noeud x_n en utilisant le taux d'accroissement

$$u'(x_{n+1}) = \frac{u(x_{n+1}) - u(x_n)}{h}. \tag{4}$$

On considère la fonction $\varphi(x, u(x)) = 2xu(x)$ définie sur $I = [0, 1]$ avec $u(0) = 1$.

Écrire, compiler et exécuter un programme écrit en `C++` qui calcule la solution de l'équation en utilisant la méthode d'Euler implicite. La fonction utilise en paramètre d'entrée le nombre d'itération, la condition initiale et la fonction φ.

2.3 Autres EDO

Appliquer la méthode d'Euler explicite et la méthode d'Euler implicite à l'équation

$$\frac{du(x)}{dx} = 50(u(x)cos(x)), x \in [0; 2] \tag{5}$$

$$u(0) = 0, \tag{6}$$