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.

1 Débogage

Le programme fourni se compose de quatre fonctions :

- `initialization` : construit une matrice carrée à partir de sa taille
- `fill_vectors` : remplit 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 un répertoire `build`.
2. Se placer 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 cette partie, nous allons implémenter des méthodes de résolution d'équations différentielles ordinaires (EDO) du premier ordre. C'est un problème fondamental en calcul scientifique avec de nombreuses applications (dynamique, physique, biologie, finance, etc.).

2.0.1 Formulation du problème

On considère le problème de Cauchy :

$$\frac{du}{dx} = \varphi(x, u(x)) \quad \text{sur } I = [a, b] \tag{1}$$

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

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

On suppose l'existence et l'unicité d'une solution u pour $x \in I$ (conditions de Cauchy-Lipschitz vérifiées).

2.0.2 Discrétisation

Pour $h > 0$ (le *pas de temps*), on définit :

- $x_n = a + nh$, pour $n = 0, \ldots, N_h$: les *nuds* de discrétisation
- $I_n = [x_n, x_{n+1}]$: les sous-intervalles de longueur h
- $u_n \approx u(x_n)$: l'approximation numérique de la solution au nud x_n

L'objectif est de calculer $(u_n)_{n=0,\ldots,N_h}$ qui approche la solution exacte $(u(x_n))_{n=0,\ldots,N_h}$.

2.1 Méthode d'Euler explicite

2.1.1 Principe

La méthode d'Euler explicite utilise l'approximation par différences finies :

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

En remplaçant dans l'équation (1), on obtient le schéma numérique :

$$\boxed{u_{n+1} = u_n + h \cdot \varphi(x_n, u_n)} \tag{4}$$

2.1.2 Cas test

On considère l'EDO :

$$\frac{du}{dx} = 2xu(x), \quad x \in I = [0, 1] \tag{5}$$

$$u(0) = 1 \tag{6}$$

Solution analytique : Cette EDO a une solution analytique connue : $u(x) = e^{x^2}$

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'itérations, la condition initiale et la fonction φ.

2.2 Méthode d'Euler implicite

2.2.1 Principe

La méthode d'Euler implicite utilise l'approximation décalée :

$$u'(x_{n+1}) \approx \frac{u(x_{n+1}) - u(x_n)}{h} \tag{7}$$

Le schéma numérique devient :

$$\boxed{u_{n+1} = u_n + h \cdot \varphi(x_{n+1}, u_{n+1})} \tag{8}$$

Propriétés :

- *Implicite* : u_{n+1} apparaît des deux côtés $\Rightarrow$ nécessite la résolution d'une équation (non-linéaire en général)
- *Ordre 1* : même ordre que Euler explicite
- *Stabilité* : inconditionnellement stable (A-stable) $\Rightarrow$ meilleur pour les problèmes raides

2.3 Application à un problème raide

Un problème est dit *raide* quand il contient des phénomènes à échelles de temps très différentes, nécessitant un pas h très petit pour la stabilité.

2.3.1 EDO raide

On considère maintenant :

$$\frac{du(x)}{dx} = -50(u(x) - \cos(x)), \quad x \in [0, 2] \tag{9}$$

$$u(0) = 0 \tag{10}$$

d

2.4 Résolution

Appliquer la méthode d'Euler explicite et la méthode d'Euler implicite à l'équation (9).