UNIVERSITÉ DE RENNES 1

Présentation projet final CPU/ GPU

Matthieu PETIT
Djemsay MORVAN
Ulysse CAROMEL

January 12, 2024

Table des matières

Méthodes d'intégration

1. Les différentes méthodes d'intégration
   - Méthode de Simpson
   - Méthode de Gauss 2D
   - Méthode de Runge-Kutta
   - Méthode de Monte-Carlo
   - Comparaison des méthodes
2. Outils de paralélisation
   - Open MP
   - CUDA
3. Résultats
   - Résultats pour MPI
   - Résultats pour CUDA

Méthode de Simpson

- Division de l'intervalle $[a, b]$ en sous-intervalles de largeur égale $h = \frac{b-a}{n}$, n étant pair.
- Approximation sur chaque sous-intervalle $[x_i, x_{i+2}]$:

$$\int_{x_i}^{x_{i+2}} f(x)\,dx \approx \frac{h}{3}\left[f(x_i) + 4f(x_{i+1}) + f(x_{i+2})\right]$$

- Estimation finale de l'intégrale :

$$\int_a^b f(x)\,dx \approx \frac{h}{3}\Big[f(a) + 4f(x_1) + 2f(x_2) + \ldots + 2f(x_{n-2}) + 4f(x_{n-1}) + f(b)\Big]$$

Quadrature Gaussienne 2D

- Estimation numérique de l'intégrale d'une fonction $f(x, y)$ sur une région bidimensionnelle définie par $a \leq x \leq b$ et $c \leq y \leq d$.
- Utilise un ensemble de points et de poids associés pour approximer l'intégrale.
- Formule générale :

$$\iint_R f(x, y)\, dx\, dy \approx \sum_{i=1}^{n} \sum_{j=1}^{m} w_{ij} \cdot f(x_i, y_j)$$

- Formule spécifique pour un quadrilatère :

$$\iint_R f(x, y)\, dx\, dy \approx \sum_{i=1}^{n} \sum_{j=1}^{m} w_{ij} \cdot f\Bigg(\frac{1}{2}(1+\xi_i)x + \frac{1}{2}(1-\xi_i)y, \frac{1}{2}(1+\eta_j)x + \frac{1}{2}(1-\eta_j)y\Bigg)$$

- Offre une précision supérieure à la quadrature de Gauss unidimensionnelle.

Méthode de Runge-Kutta (RK4) pour les EDO

- Technique numérique pour résoudre des équations différentielles ordinaires (EDO).
- Méthode de Runge-Kutta d'ordre 4 (RK4) : équilibre entre précision et complexité.
- Forme générale d'une EDO du premier ordre :

$$\frac{dy}{dt} = f(t, y)$$

- Étapes de la méthode RK4 :

$$k_1 = h \cdot f(t_n, y_n)$$
$$k_2 = h \cdot f(t_n + \frac{h}{2}, y_n + \frac{k_1}{2})$$
$$k_3 = h \cdot f(t_n + \frac{h}{2}, y_n + \frac{k_2}{2})$$
$$k_4 = h \cdot f(t_n + h, y_n + k_3)$$

- Mise à jour de la solution à chaque pas :

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

- Offre une meilleure précision que des méthodes de pas fixe plus simples, largement utilisée pour sa robustesse et polyvalence.

Méthode de Monte Carlo pour l'intégration

- Utilisée pour estimer numériquement des intégrales complexes, notamment dans des espaces multidimensionnels.
- Fondée sur des principes probabilistes, associant le hasard à des calculs numériques.
- Estimation d'une valeur en utilisant des échantillons aléatoires dans un domaine donné.
- Génération de points aléatoires dans le domaine D.
- Évaluation de la fonction $f(x, y)$ pour chaque point généré.
- Calcul de la moyenne pondérée des valeurs de $f(x, y)$ pour les points générés, multipliée par la mesure de D.

Formule d'estimation de Monte Carlo pour l'intégrale :

$$I \approx A \cdot \frac{1}{N} \sum_{i=1}^{N} f(x_i, y_i)$$

Avantages : Flexibilité, capacité à traiter des problèmes complexes en dimensions élevées. Précision dépend du nombre de points aléatoires générés.

Comparaison des méthodes numériques

Runge-Kutta (RK4)

+ Précision élevée.
+ Adaptée aux EDO.
- Plus de calculs.

Gaussienne 2D

+ Haute précision.
+ Adaptée aux intégrales bidimensionnelles.
- Plus complexe.

Simpson

+ Simple à mettre en œuvre.
+ Bonne précision.
- Limité aux formes simples.

Monte Carlo

+ Grande flexibilité.
+ Adaptée aux problèmes complexes.
- Précision dépend du nombre d'échantillons.

Outils de paralélisation

Open MP

- **Intégration de Gauss 2D**
  - Parallélisation de la génération de points et de poids avec `#pragma omp parallel for collapse(2)`.
  - L'utilisation de `collapse(2)` exploite le parallélisme dans les deux boucles imbriquées, améliorant l'efficacité.
- **Intégration de Monte Carlo**
  - La boucle de génération de points aléatoires est parallélisée avec `#pragma omp parallel for reduction(+:total)`.
  - Chaque thread utilise une graine différente (`rd() + thread_id`) pour éviter les séquences aléatoires identiques.
- **Runge-Kutta**
  - Boucle temporelle parallélisée avec `#pragma omp parallel for`.
  - Copie temporaire des données (`std::vector<double> tempU(u)`) utilisée pour chaque thread, évitant les dépendances de données et optimisant les calculs.
- **Intégration de Simpson**
  - Boucles `for` parallélisées avec `#pragma omp parallel for reduction(+:integral)` pour les deux versions.
  - Pour `compositeSimpsons_3_8`, utilisation d'un pas de 3 pour les indices de boucle, améliorant l'efficacité de la méthode.

MPI

- **Intégration Gauss 2D**
  - Parallélisation de la génération de points et de poids.
  - Utilisation de MPI_Sendrecv pour l'échange des bords entre les processus.
- **Intégration Monte Carlo**
  - Division équitable des points entre les processus.
  - Utilisation de MPI_Reduce pour obtenir le résultat global.
- **Runge-Kutta**
  - Division des données spatiales entre les processus.
  - Utilisation de MPI_Gather pour collecter les résultats.
- **Intégration Simpson**
  - Division équitable des points entre les processus.
  - Utilisation de MPI_Reduce pour obtenir le résultat global.

CUDA

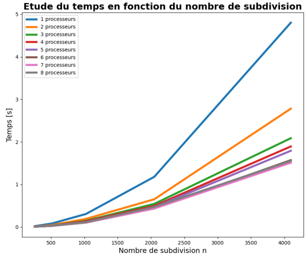
Résultats

1. Les différentes méthodes d'intégration
   - Méthode de Simpson
   - Méthode de Gauss 2D
   - Méthode de Runge-Kutta
   - Méthode de Monte-Carlo
   - Comparaison des méthodes

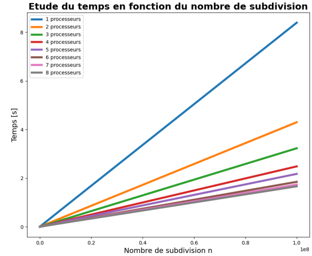
2. Outils de paralélisation
   - Open MP
   - CUDA

3. Résultats
   - Résultats pour MPI
   - Résultats pour CUDA

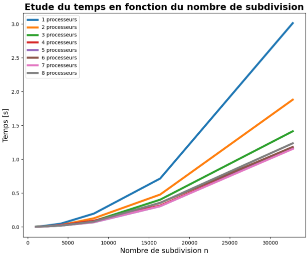
Résultats pour Open MP



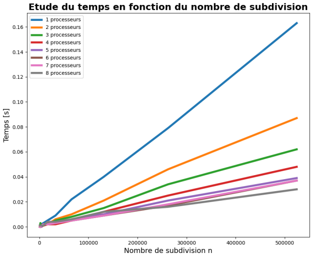
Gauss 2D



Simpson

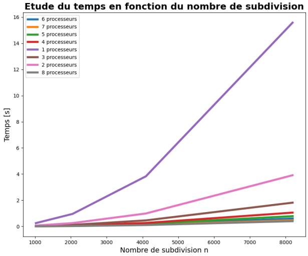


Runge Kutta

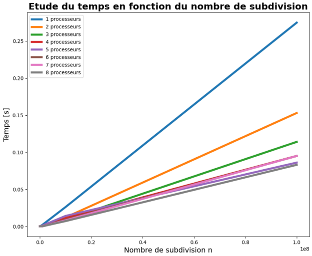


Monte Carlo

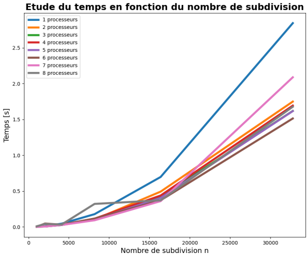
MPI



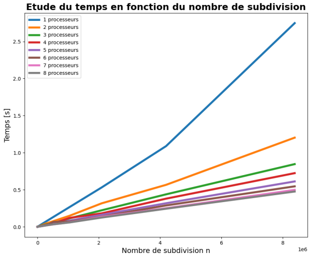
Gauss 2D



Simpson

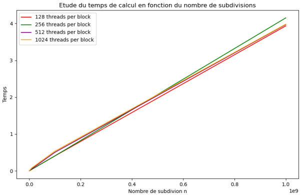


Runge Kutta

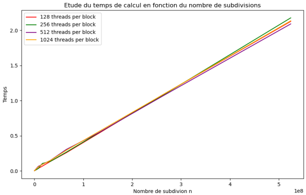


Monte Carlo

CUDA



Gauss 2D
Runge Kutta



Simpson
Monte Carlo