Rapport d'Analyse Numérique : Comparaison des Méthodes de Résolution d'Équations Différentielles et d'Intégration Numérique

Fofana Adama
Master 2 Génie Informatique
Université Nangui Abrogoua

30 décembre 2024

Abstract

Ce rapport présente une étude comparative des méthodes numériques pour la résolution d'équations différentielles ordinaires et le calcul d'intégrales. Trois méthodes de résolution d'EDO (Euler explicite, Heun et Runge-Kutta d'ordre 4) sont comparées sur trois équations différentielles test. Parallèlement, cinq méthodes d'intégration numérique (Gauss-Legendre, Gauss-Laguerre, Gauss-Chebyshev, Simpson composite et spline cubique) sont évaluées sur quatre fonctions tests. Les critères de comparaison incluent la précision, le temps d'exécution et le taux de convergence.

Introduction

Contexte académique

Ce travail s'inscrit dans le cadre du cours de Calcul Numérique (Analyse Numérique) du Master 2 Génie Informatique de l'Université Nangui Abrogoua. L'objectif est d'implémenter et de comparer différentes méthodes numériques fondamentales.

Objectifs

- Implémenter les méthodes de résolution d'équations différentielles ordinaires
- Implémenter les méthodes d'intégration numérique par quadrature
- Comparer les performances en termes de précision et de temps d'exécution
- Analyser la convergence des différentes méthodes
- Identifier les méthodes les plus adaptées à différents types de problèmes

Théorie Mathématique

Équations Différentielles Ordinaires (EDO)

Une équation différentielle ordinaire du premier ordre s'écrit :

$$\frac{dy}{dx} = f(x, y), \quad y(x_0) = y_0$$

Méthode d'Euler explicite

La méthode d'Euler explicite est donnée par :

$$y_{n+1} = y_n + hf(x_n, y_n)$$

où h est le pas d'intégration.

Méthode de Heun (Euler amélioré)

La méthode de Heun utilise une prédiction-correction :

$$\begin{aligned} k_1 &= f(x_n, y_n) \\ k_2 &= f(x_n + h, y_n + h k_1) \\ y_{n+1} &= y_n + \frac{h}{2}(k_1 + k_2) \end{aligned}$$

Méthode de Runge-Kutta d'ordre 4 (RK4)

La méthode RK4 est donnée par :

$$\begin{aligned} k_1 &= f(x_n, y_n) \\ k_2 &= f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2} k_1\right) \\ k_3 &= f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2} k_2\right) \\ k_4 &= f(x_n + h, y_n + h k_3) \\ y_{n+1} &= y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{aligned}$$

Méthodes d'Intégration Numérique

Quadrature de Gauss-Legendre

Pour l'intégrale $\int_a^b f(x)\,dx$, on utilise la formule :

$$\int_a^b f(x)\, dx \approx \frac{b-a}{2} \sum_{i=1}^n w_i \, f\left(\frac{b-a}{2}x_i + \frac{a+b}{2}\right)$$

où x_i et w_i sont les points et poids de Legendre sur $[-1,1]$.

Quadrature de Gauss-Laguerre

Pour les intégrales sur $[0,\infty)$ avec poids e^{-x} :

$$\int_0^\infty e^{-x} f(x)\, dx \approx \sum_{i=1}^n w_i \, f(x_i)$$

Quadrature de Gauss-Chebyshev

Pour les intégrales sur $[-1,1]$ avec poids $\dfrac{1}{\sqrt{1-x^2}}$:

$$\int_{-1}^1 \frac{f(x)}{\sqrt{1-x^2}}\, dx \approx \sum_{i=1}^n w_i \, f(x_i)$$

Méthode composite de Simpson

$$\int_a^b f(x)\, dx \approx \frac{h}{3} \left[f(x_0) + f(x_n) + 4\sum_{i=1}^{n/2} f(x_{2i-1}) + 2\sum_{i=1}^{n/2-1} f(x_{2i}) \right]$$

avec $\displaystyle h = \frac{b-a}{n}$ et n pair.

Implémentation

Structure du code Python

Le code est organisé en deux classes principales :

- `SolveurEDO` pour les méthodes de résolution d'EDO
- `IntegrationNumerique` pour les méthodes d'intégration

Équations différentielles testées

Équation 1

$$z'(x) = 0.1xz(x), \quad z(0) = 1$$

Solution exacte : $z(x) = e^{0.05x^2}$

Équation 2

$$z'(x) = \frac{1 - 30x}{2\sqrt{x}} + 15z(x), \quad z(0) = 0$$

Solution exacte : $\displaystyle z(x) = \sqrt{x}$

Équation 3

$$z'(x) = \pi\cos(\pi x)z(x), \quad z(0) = 1$$

Solution exacte : $z(x) = e^{\sin(\pi x)}$

Fonctions d'intégration testées

1. **Fonction Chebyshev** : $f(x) = \cos(10x)$ sur $[-1,1]$ avec poids $\dfrac{1}{\sqrt{1-x^2}}$
2. **Fonction Laguerre** : $\displaystyle f(x) = \frac{1}{1+x^2}$ sur $[0,\infty)$ avec poids e^{-x}
3. **Fonction combinée** : $f(x) = \cos(x)$ sur $[0,1]$ avec poids $\dfrac{1}{\sqrt{1-x^2}}$
4. **Fonction neutre** : $\displaystyle f(x) = \frac{1}{1+25x^2}$ sur $[-1,1]$

Résultats et Analyse

Comparaison des méthodes pour les EDO

Temps d'exécution

Les temps d'exécution pour chaque méthode et chaque équation sont présentés dans le tableau suivant :

Temps d'exécution des méthodes pour les trois équations différentielles

Méthode	Équation 1 (s)	Équation 2 (s)	Équation 3 (s)
Euler	t_{e1}	t_{e2}	t_{e3}
Heun	t_{h1}	t_{h2}	t_{h3}
Runge-Kutta 4	t_{rk1}	t_{rk2}	t_{rk3}

Erreurs absolues maximales

Erreurs absolues maximales (échelle logarithmique)

Méthode	Équation 1	Équation 2	Équation 3
Euler	e_{e1}	e_{e2}	e_{e3}
Heun	e_{h1}	e_{h2}	e_{h3}
Runge-Kutta 4	e_{rk1}	e_{rk2}	e_{rk3}

Comparaison des méthodes d'intégration

Précision en fonction du nombre de points

Évolution de l'erreur en fonction du nombre de points pour les quatre fonctions tests

Temps d'exécution en fonction du nombre de points

Évolution du temps d'exécution en fonction du nombre de points

Discussion

Performance des méthodes EDO

- **Méthode d'Euler** : La plus rapide mais la moins précise. Erreur de l'ordre de $\square(h)$.
- **Méthode de Heun** : Compromis entre précision et temps. Erreur de l'ordre de $\square(h^2)$.
- **Runge-Kutta 4** : La plus précise mais la plus coûteuse en temps. Erreur de l'ordre de $\square(h^4)$.

Performance des méthodes d'intégration

- **Gauss-Chebyshev** : Excellente pour les intégrales avec poids $\dfrac{1}{\sqrt{1-x^2}}$.
- **Gauss-Laguerre** : Optimale pour les intégrales sur $[0, \infty)$ avec poids e^{-x}.
- **Gauss-Legendre** : Robuste pour les intégrales standards sur intervalles finis.
- **Simpson composite** : Simple à implémenter mais nécessite plus de points.
- **Spline cubique** : Coûteuse en temps mais fournit aussi une interpolation.

Analyse de convergence

Les méthodes de Gauss présentent une convergence exponentielle pour les fonctions analytiques, tandis que Simpson et les splines présentent une convergence algébrique.

Conclusion

Principales conclusions

1. Pour les EDO, le choix de la méthode dépend du compromis précision/temps requis.
2. Runge-Kutta 4 est recommandé pour les applications nécessitant une haute précision.
3. Pour l'intégration numérique, les méthodes de Gauss adaptées au poids sont les plus efficaces.
4. Gauss-Legendre est la méthode la plus polyvalente pour les intégrales standards.

5. Simpson reste une bonne option pour des applications simples avec des fonctions régulières.

Perspectives

- Extension aux systèmes d'équations différentielles
- Implémentation de méthodes adaptatives
- Application à des problèmes physiques réels
- Étude de la stabilité numérique

9 Quarteroni, A., Saleri, F., & Gervasio, P. (2010). *Calcul scientifique*. Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (2007). *Numerical Recipes*. Atkinson, K. E. (2008). *An Introduction to Numerical Analysis*. Chapra, S. C., & Canale, R. P. (2010). *Numerical Methods for Engineers*.

Code Python complet

Code pour les EDO

Voici un extrait du code Python pour les méthodes de résolution d'EDO :

```python
class SolveurEDO:
    def __init__(self):
        pass

    # Méthode d'Euler explicite
    def euler(self, f, x0, y0, h, n):
        x = np.zeros(n+1)
        y = np.zeros(n+1)
        x[0] = x0
        y[0] = y0

        for i in range(n):
            y[i+1] = y[i] + h * f(x[i], y[i])
            x[i+1] = x[i] + h

        return x, y

    # Méthode de Heun (Euler amélioré)
    def heun(self, f, x0, y0, h, n):
        x = np.zeros(n+1)
        y = np.zeros(n+1)
        x[0] = x0
        y[0] = y0

        for i in range(n):
            k1 = f(x[i], y[i])
            k2 = f(x[i] + h, y[i] + h * k1)
            y[i+1] = y[i] + h * (k1 + k2) / 2
            x[i+1] = x[i] + h

        return x, y

    # Méthode de Runge-Kutta d'ordre 4
    def runge_kutta_4(self, f, x0, y0, h, n):
        x = np.zeros(n+1)
        y = np.zeros(n+1)
        x[0] = x0
        y[0] = y0

        for i in range(n):
            k1 = f(x[i], y[i])
            k2 = f(x[i] + h/2, y[i] + h*k1/2)
            k3 = f(x[i] + h/2, y[i] + h*k2/2)
            k4 = f(x[i] + h, y[i] + h*k3)

            y[i+1] = y[i] + h * (k1 + 2*k2 + 2*k3 + k4) / 6
            x[i+1] = x[i] + h

        return x, y
```

Code pour l’intégration numérique

```python
class IntegrationNumerique:
    def __init__(self):
        pass

    # Quadrature de Gauss-Legendre
    def gauss_legendre(self, f, a, b, n):
        if b == np.inf:
            def g(t):
                x = t / (1 - t**2) if t != 1 else np.inf
                return f(x) * (1 + t**2) / (1 - t**2)**2
            a_tr = -0.999
            b_tr = 0.999
        else:
            g = f
            a_tr = a
            b_tr = b

        x, w = roots_legendre(n)
        t = 0.5 * (b_tr - a_tr) * x + 0.5 * (a_tr + b_tr)
        integral = 0.5 * (b_tr - a_tr) * np.sum(w * g(t))

        return integral
```

Annexes techniques

Calcul des valeurs exactes

Fonction de Bessel

Pour l'intégrale de Chebyshev :

$$\int_{-1}^{1} \frac{\cos(10x)}{\sqrt{1-x^2}} \, dx = \pi J_0(10)$$

où J_0 est la fonction de Bessel de première espèce d'ordre 0.

Intégrale exponentielle

Pour l'intégrale de Laguerre :

$$\int_{0}^{\infty} \frac{e^{-x}}{1+x^2} \, dx = \text{Ci}(1)\sin(1) + \left(\frac{\pi}{2} - \text{Si}(1)\right)\cos(1)$$

où Si et Ci sont les intégrales sinus et cosinus.