

Parallélisation temporelle pour la résolution du système de Lorenz

Une approche avec l'algorithme Parareal

Elomn AHOUANYE
Yanel AÏNA

École Nationale Supérieure de Génie Mathématique et Modélisation
(ENSGMM Abomey)

Supervisé par
Dr. DANDOGBESSI Bruno

18 mars 2025

Résumé

Ce document présente une étude approfondie de la parallélisation temporelle appliquée au système de Lorenz modifié pour une particule active guidée par sa mémoire. Nous explorons l'implémentation de l'algorithme Parareal comme alternative à la méthode RK4 séquentielle traditionnelle, en mettant l'accent sur les aspects théoriques et pratiques de cette approche novatrice.

Table des matières

| | | |
|----------|---|-----------|
| 1 | Introduction | 3 |
| 1.1 | Contexte physique : particules actives guidées par la mémoire | 3 |
| 1.2 | Du système physique au modèle mathématique | 3 |
| 1.3 | Émergence du système de Lorenz | 3 |
| 1.4 | Problématique | 4 |
| 1.5 | Reconstruction de la trajectoire physique | 4 |
| 1.6 | Approche traditionnelle : la méthode RK4 | 4 |
| 1.7 | Objectifs et organisation du document | 5 |
| 2 | Résolution numérique par la méthode RK4 | 6 |
| 2.1 | Formulation pour le système de Lorenz | 6 |
| 2.2 | Implémentation | 6 |
| 2.3 | Analyse des résultats | 7 |
| 2.4 | Synthèse des résultats | 11 |
| 2.5 | Limitations techniques | 11 |
| 3 | Algorithme Parareal | 13 |
| 3.1 | Principe fondamental | 13 |
| 3.2 | Décomposition temporelle | 13 |
| 3.3 | Architecture à deux niveaux | 13 |
| 3.4 | Processus itératif | 13 |
| 3.5 | Analyse de la formule de correction | 14 |
| 3.6 | Propriétés de convergence | 14 |
| 3.7 | Application au système de Lorenz | 15 |
| 4 | Implémentation | 17 |
| 4.1 | Architecture logicielle | 17 |
| 4.2 | Composants principaux | 17 |
| 4.3 | Mécanismes de stabilité et sécurité | 18 |
| 4.4 | Parallélisation avancée | 19 |
| 4.5 | Visualisation et analyse | 19 |
| 4.6 | Stratégies de convergence avancées | 19 |
| 5 | Résultats et analyse comparative | 21 |
| 5.1 | Méthodologie de comparaison | 21 |
| 5.2 | Analyse par régime dynamique | 21 |
| 5.3 | Synthèse des performances | 26 |
| 5.4 | Analyse des performances | 26 |
| 6 | Conclusion et perspectives | 28 |
| 6.1 | Synthèse des résultats | 28 |
| 6.2 | Limitations actuelles | 28 |

1 Introduction

1.1 Contexte physique : particules actives guidées par la mémoire

Les systèmes de particules actives représentent un domaine de la physique moderne, où le comportement des particules est influencé par leur propre histoire. Un exemple particulièrement intéressant de tels systèmes est celui des gouttes "marcheuses" sur un bain liquide vibrant verticalement, où une goutte interagit avec les ondes qu'elle génère elle-même.

Dans ce système, chaque impact de la goutte sur la surface du bain crée une onde stationnaire localisée. Ces ondes persistent pendant un certain temps, formant une "mémoire" du passage de la goutte. Cette mémoire ondulatoire guide ensuite le mouvement de la goutte, créant une rétroaction complexe entre la particule et son environnement auto-généré.

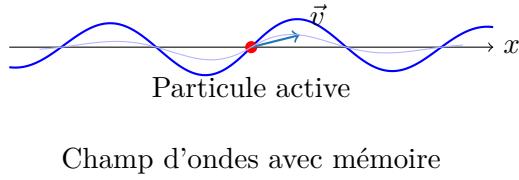


FIGURE 1 – Particule active interagissant avec son champ d'ondes auto-généré

1.2 Du système physique au modèle mathématique

La modélisation de ce système commence par la description du mouvement horizontal de la particule. En moyenne sur un cycle de rebond vertical, la dynamique peut être décrite par une équation de trajectoire intégro-différentielle :

$$\ddot{x}_d + \dot{x}_d = F_{self} + F_{bias} \quad (1)$$

où x_d représente la position horizontale de la particule, et les termes de droite incluent :

- F_{self} : la force exercée par le champ d'ondes auto-généré
- F_{bias} : une éventuelle force de biais externe

La force F_{self} due au champ d'ondes s'exprime comme une intégrale sur l'histoire de la particule :

$$F_{self} = -R \int_{-\infty}^t W'(x_d(t) - x_d(s)) e^{-\frac{t-s}{\tau}} ds \quad (2)$$

où :

- R est l'amplitude adimensionnée des ondes générées
- τ est le temps caractéristique de décroissance des ondes (paramètre de mémoire)
- $W(x)$ représente la forme des ondes générées

1.3 Émergence du système de Lorenz

Une simplification remarquable de ce système complexe apparaît lorsqu'on considère une forme d'onde idéalisée $W(x) = \cos(x)$. Cette approximation, bien que simple, capture les caractéristiques essentielles de la dynamique tout en permettant une transformation mathématique élégante.

En introduisant les variables :

$$X = \dot{x}_d \quad (\text{vitesse de la particule}) \quad (3)$$

$$Y = F_{self} \quad (\text{force de mémoire}) \quad (4)$$

$$Z = R \int_{-\infty}^t \cos(x_d(t) - x_d(s)) e^{-\frac{t-s}{\tau}} ds \quad (\text{hauteur du champ d'onde à la position de la particule}) \quad (5)$$

Le système se transforme en un ensemble d'équations différentielles ordinaires :

$$\begin{cases} \dot{X} = Y - X + F_{bias} \\ \dot{Y} = -\frac{1}{\tau}Y + XZ \\ \dot{Z} = R - \frac{1}{\tau}Z - XY \end{cases} \quad (6)$$

Ce système est équivalent au célèbre système de Lorenz lorsque $F_{bias} = 0$.

1.4 Problématique

La résolution numérique de ce système présente plusieurs défis :

1. **Non-linéarité** : Les termes de couplage XZ et XY induisent des comportements complexes
2. **Sensibilité** : Le système peut présenter une forte dépendance aux conditions initiales
3. **Échelles multiples** : Le paramètre τ introduit différentes échelles de temps

1.5 Reconstruction de la trajectoire physique

Une fois le système de Lorenz résolu numériquement, la reconstruction de la trajectoire physique de la particule nécessite deux étapes :

1. **Obtention de la vitesse** : La variable X du système de Lorenz donne directement la vitesse de la particule :

$$\dot{x}_d = X \quad (7)$$

2. **Calcul de la position** : La position x_d est obtenue par intégration de la vitesse :

$$x_d(t) = x_d(0) + \int_0^t X(s) ds \quad (8)$$

où $x_d(0)$ est la position initiale de la particule (souvent fixée à 0 pour simplifier).

Cette reconstruction peut être réalisée numériquement en utilisant la méthode des trapèzes ou en intégrant directement les points générés par la méthode RK4. Par exemple, si l'on utilise RK4 pour résoudre le système de Lorenz, on peut calculer la position à chaque pas de temps :

$$x_d(t_{n+1}) = x_d(t_n) + \frac{h}{2}(X(t_n) + X(t_{n+1})) \quad (9)$$

où h est le pas de temps et $t_n = t_0 + nh$.

1.6 Approche traditionnelle : la méthode RK4

1.6.1 Principe de la méthode

La méthode Runge-Kutta d'ordre 4 (RK4) est une méthode classique de résolution numérique des équations différentielles. Elle approxime la solution en combinant quatre évaluations de la fonction dérivée à différents points intermédiaires :

$$u_{n+1} = u_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \quad (10)$$

1.6.2 Interprétation géométrique

Chaque coefficient représente une estimation de la pente en différents points de l'intervalle :

- k_1 : pente initiale au point (t_n, u_n)
- k_2 : pente au point milieu, après un demi-pas utilisant k_1
- k_3 : pente au point milieu, après un demi-pas utilisant k_2
- k_4 : pente finale, après un pas complet utilisant k_3

1.6.3 Limitations de la parallélisation

La méthode RK4, bien que précise, présente une limitation fondamentale pour la parallélisation : sa nature intrinsèquement séquentielle. Cette séquentialité provient de deux aspects :

1. **Dépendance temporelle** : Chaque pas de temps dépend directement du résultat du pas précédent :

$$u_{n+1} = \Phi_{\text{RK4}}(u_n) \quad (11)$$

2. **Dépendance interne** : Les coefficients k_i doivent être calculés dans l'ordre, chacun dépendant des précédents :

$$k_i = f(k_1, \dots, k_{i-1}) \quad (12)$$

Cette double dépendance rend impossible toute parallélisation directe de la méthode, car :

- Les pas de temps ne peuvent pas être calculés indépendamment
- Les étages internes de RK4 doivent être calculés séquentiellement

L'algorithme Parareal, présenté en détail dans la section suivante, propose une solution à ce problème en brisant la barrière de la séquentialité temporelle.

1.7 Objectifs et organisation du document

Notre travail se concentre sur la résolution numérique efficace de ce système de Lorenz, en particulier :

- La résolution séquentielle avec la méthode RK4
- La tentative de parallélisation temporelle avec MPI via l'algorithme Parareal
- L'implémentation de l'algorithme Parareal pour le système de Lorenz
- L'analyse des performances et de la précision

Ce document est structuré comme suit :

- La section 3 présente en détail l'algorithme Parareal
- La section ?? décrit notre implémentation
- La section ?? analyse les performances et résultats obtenus
- La section ?? discute les implications et perspectives futures

2 Résolution numérique par la méthode RK4

2.1 Formulation pour le système de Lorenz

La méthode RK4 s'applique au système de Lorenz en traitant simultanément les trois équations couplées. Pour un état $\mathbf{u} = (X, Y, Z)$, le système peut s'écrire sous la forme :

$$\frac{d\mathbf{u}}{dt} = \mathbf{f}(\mathbf{u}) = \begin{pmatrix} Y - X \\ -\frac{1}{\tau}Y + XZ \\ R - \frac{1}{\tau}Z - XY \end{pmatrix} \quad (13)$$

2.1.1 Calcul des coefficients

Les coefficients k_i de la méthode RK4 sont calculés comme suit :

$$k_1 = \mathbf{f}(t_n, \mathbf{u}_n) \quad (14)$$

$$k_2 = \mathbf{f}\left(t_n + \frac{h}{2}, \mathbf{u}_n + \frac{h}{2}k_1\right) \quad (15)$$

$$k_3 = \mathbf{f}\left(t_n + \frac{h}{2}, \mathbf{u}_n + \frac{h}{2}k_2\right) \quad (16)$$

$$k_4 = \mathbf{f}(t_n + h, \mathbf{u}_n + hk_3) \quad (17)$$

où h est le pas de temps et \mathbf{u}_n est l'état du système au temps t_n .

2.2 Implémentation

2.2.1 Structure du code

Le solveur RK4 est implémenté comme une sous-routine Fortran qui prend en compte les spécificités du système de Lorenz :

Listing 1 – Implémentation RK4 pour le système de Lorenz

```

1  module rk4_solver
2    use types, only: dp
3    implicit none
4
5    type :: LorenzState
6      real(dp) :: X, Y, Z
7    end type LorenzState
8
9  contains
10   subroutine solve_step(state, dt, tau, R)
11     type(LorenzState), intent(inout) :: state
12     real(dp), intent(in) :: dt, tau, R
13     type(LorenzState) :: k1, k2, k3, k4, temp
14
15     ! Calcul de k1
16     k1%X = state%Y - state%X
17     k1%Y = -state%Y/tau + state%X * state%Z
18     k1%Z = R - state%Z/tau - state%X * state%Y
19
20     ! Calcul de k2
21     temp = state
22     temp%X = state%X + dt*k1%X/2
23     temp%Y = state%Y + dt*k1%Y/2

```

```

24      temp%Z = state%Z + dt*k1%Z/2
25      k2%X = temp%Y - temp%X
26      k2%Y = -temp%Y/tau + temp%X * temp%Z
27      k2%Z = R - temp%Z/tau - temp%X * temp%Y
28
29      ! Calcul de k3
30      temp%X = state%X + dt*k2%X/2
31      temp%Y = state%Y + dt*k2%Y/2
32      temp%Z = state%Z + dt*k2%Z/2
33      k3%X = temp%Y - temp%X
34      k3%Y = -temp%Y/tau + temp%X * temp%Z
35      k3%Z = R - temp%Z/tau - temp%X * temp%Y
36
37      ! Calcul de k4
38      temp%X = state%X + dt*k3%X
39      temp%Y = state%Y + dt*k3%Y
40      temp%Z = state%Z + dt*k3%Z
41      k4%X = temp%Y - temp%X
42      k4%Y = -temp%Y/tau + temp%X * temp%Z
43      k4%Z = R - temp%Z/tau - temp%X * temp%Y
44
45      ! Mise à jour finale
46      state%X = state%X + (dt/6)*(k1%X + 2*k2%X + 2*k3%X + k4%X)
47      state%Y = state%Y + (dt/6)*(k1%Y + 2*k2%Y + 2*k3%Y + k4%Y)
48      state%Z = state%Z + (dt/6)*(k1%Z + 2*k2%Z + 2*k3%Z + k4%Z)
49  end subroutine solve_step
50 end module rk4_solver

```

2.3 Analyse des résultats

2.3.1 Paramètres de simulation

Les simulations ont été réalisées avec les paramètres suivants :

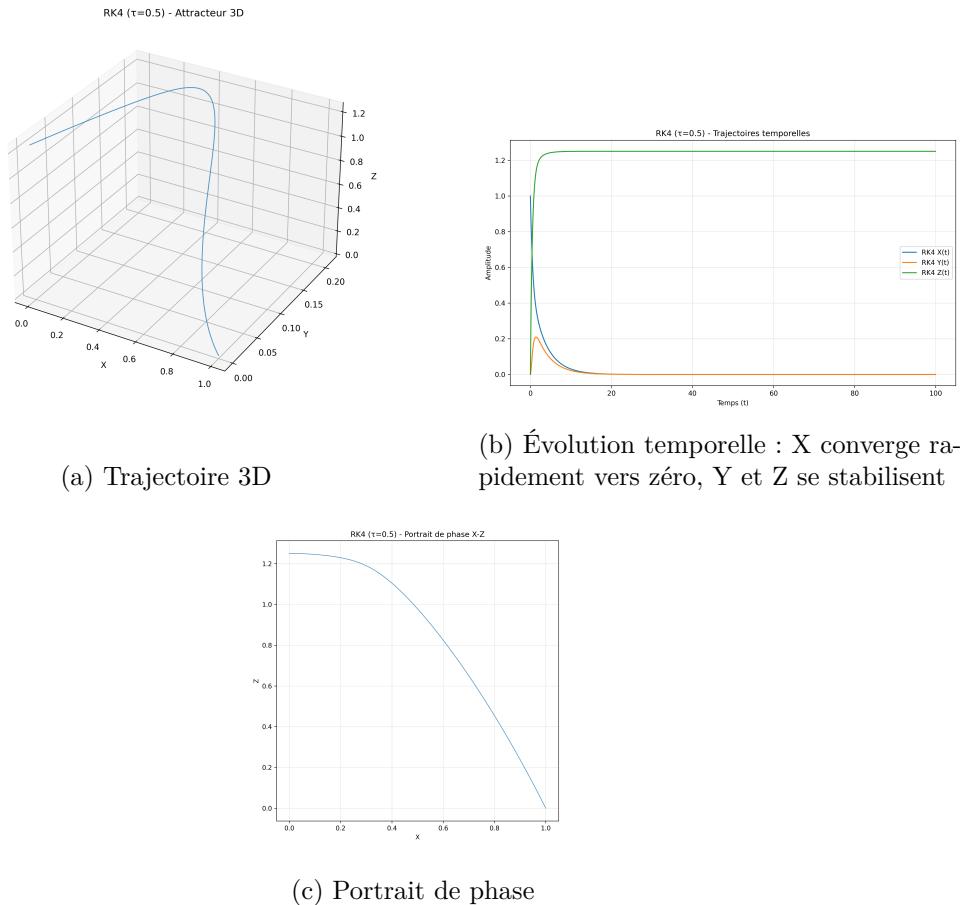
- **Pas de temps** : $h = 0.01$
- **Durée totale** : $T = 100.0$
- **Amplitude des ondes** : $R = 2.5$
- **Conditions initiales** : $(X_0, Y_0, Z_0) = (1.0, 0.0, 0.0)$

En fonction du paramètre de mémoire τ , le système présente différents comportements caractéristiques.

2.3.2 État Non-Marcheur ($\tau = 0.5$)

Pour une faible valeur de τ , le système converge rapidement vers un point fixe.

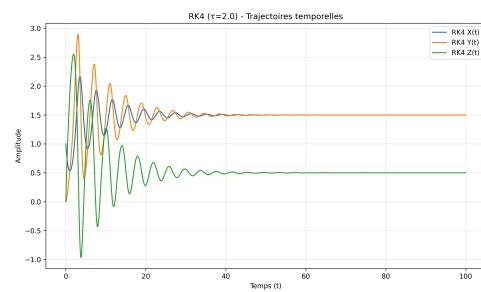
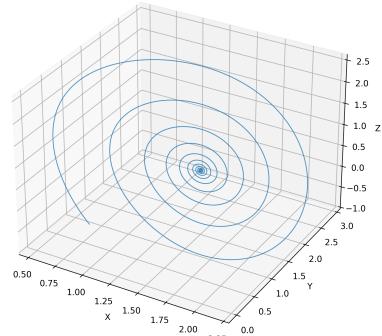
On peut également noter que la trajectoire converge rapidement vers l'origine ($X \rightarrow 0$) et l'absence d'oscillations.

FIGURE 2 – Dynamique pour $\tau = 0.5$: convergence vers un point fixe

2.3.3 Marche Régulière ($\tau = 2.0$)

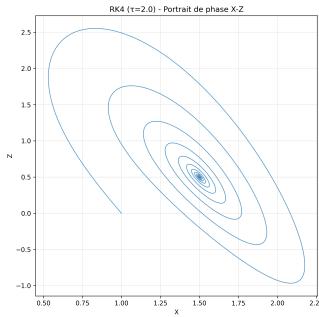
À $\tau = 2.0$, le système présente des états stationnaires stables non triviaux.

En effet, on peut observer une stabilisation de la trajectoire autour de X non nulle et constante.

RK4 ($\tau=2.0$) - Attracteur 3D

(b) Évolution temporelle : X et Y se stabilisent

(a) Trajectoire 3D : Trajectoire convergente vers un point fixe non trivial



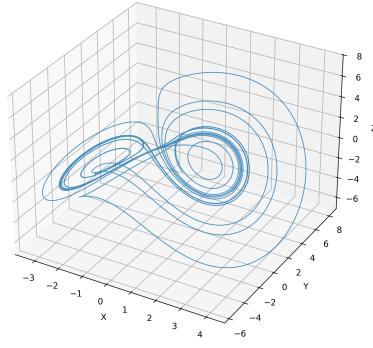
(c) Portrait de phase

FIGURE 3 – Dynamique pour $\tau = 2.0$: états stationnaires stables

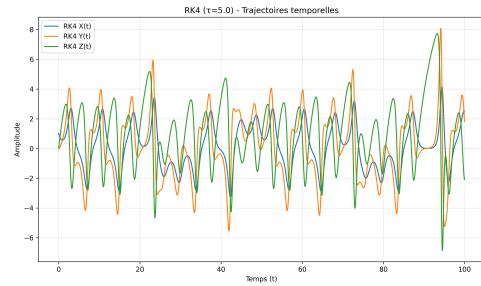
2.3.4 Marche Chaotique ($\tau = 5.0$)

À $\tau = 5.0$, on observe l'apparition d'oscillations complexes.

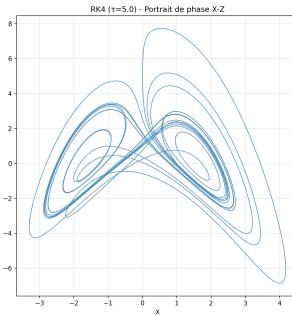
Aucun motif régulier ou périodique n'est observable clairement dans la dynamique du système.

RK4 ($\tau=5.0$) - Attracteur 3D

(a) Trajectoire 3D : Attracteur étrange tridimensionnel avec structure complexe



(b) Évolution temporelle : Fortes oscillations irrégulières de toutes les variables



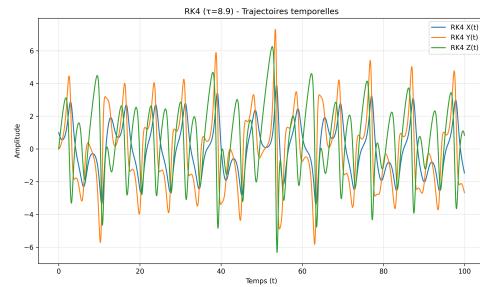
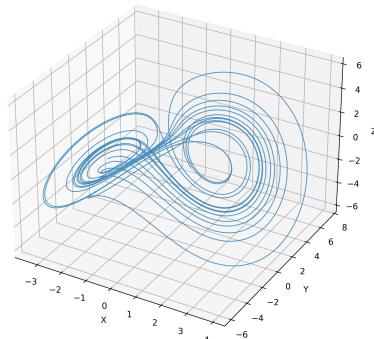
(c) Portrait de phase : Attracteur étrange avec structure fractale complexe.

FIGURE 4 – Dynamique pour $\tau = 5.0$: oscillations complexes

2.3.5 Oscillations avec Dérive ($\tau = 8.9$)

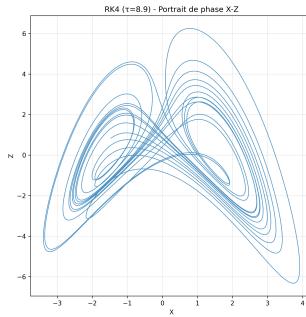
Pour $\tau = 8.9$, le système entre dans un régime pleinement chaotique.

Les oscillations sont quasi-périodiques avec une dérive progressive, et les motifs sont de plus en plus complexes. La structure est plus organisé que le scénario chaotique précédent, mais moins stable que le scénario de

RK4 ($\tau=8.9$) - Attracteur 3D

(b) Évolution temporelle : Oscillations

(a) Trajectoire 3D : Structure intermédiaire entre avec une structure plus régulière mais avec l'attracteur chaotique et le point fixe dérive progressive



(c) Portrait de phase : Trajectoire plus ordonnée mais non-périodique, formant un cycle limite déformé

FIGURE 5 – Dynamique pour $\tau = 8.9$: comportement oscillatoire avec dérive

2.4 Synthèse des résultats

Les simulations numériques confirment parfaitement les comportements théoriques attendus pour chaque régime dynamique :

- **État Non-Marcheur** ($\tau = 0.5$) : Convergence rapide vers l'origine, démontrant la stabilité du point fixe
- **Marche Régulière** ($\tau = 2.0$) : Établissement d'états stationnaires non nuls, caractéristiques du mouvement de marche
- **Marche Chaotique** ($\tau = 5.0$) : Comportement complexe avec attracteur étrange, typique du chaos
- **Oscillations avec Dérive** ($\tau = 8.9$) : Structure intermédiaire combinant oscillations et dérive progressive

2.5 Limitations techniques

La méthode RK4, bien que précise et stable, présente certaines limitations importantes :

1. **Coût computationnel** : 4 évaluations de la fonction par pas de temps, rendant les simulations longues sur de grands intervalles

2. **Stockage mémoire** : Nécessité de conserver les états intermédiaires, particulièrement problématique pour les systèmes de grande dimension
3. **Parallélisation** : Impossible due à la dépendance séquentielle entre les pas de temps

Ces limitations justifient l'exploration d'approches alternatives comme l'algorithme Parareal pour les simulations à grande échelle, particulièrement dans les régimes chaotiques où une haute précision est nécessaire.

3 Algorithme Parareal

3.1 Principe fondamental

L'algorithme Parareal introduit une approche qui brise la barrière de la séquentialité temporelle inhérente aux méthodes classiques. Le principe fondamental repose sur une décomposition du domaine temporel combinée à un processus itératif de correction.

Cette approche résout les limitations de RK4 en :

- Permettant le calcul parallèle sur différents intervalles de temps
- Maintenant la précision grâce au propagateur fin
- Assurant la convergence via le processus itératif de correction

3.2 Décomposition temporelle

L'intervalle de temps global est divisé en N sous-intervalles :

$$[0, T] = \bigcup_{i=0}^{N-1} [T_i, T_{i+1}] \quad (18)$$

Cette décomposition permet une distribution naturelle du calcul sur plusieurs processeurs, chacun traitant un sous-intervalle spécifique.

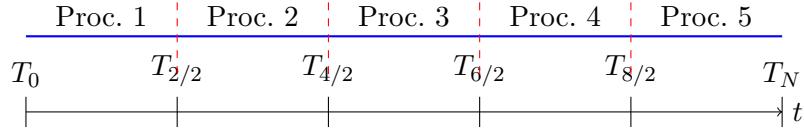


FIGURE 6 – Décomposition temporelle et distribution sur les processeurs

3.3 Architecture à deux niveaux

L'algorithme repose sur l'utilisation de deux propagateurs complémentaires :

1. Propagateur grossier \mathcal{G} :

- Rapide mais approximatif
- Utilisé pour la prédiction initiale
- Typiquement basé sur une méthode d'Euler

2. Propagateur fin \mathcal{F} :

- Précis mais coûteux en calcul
- Appliqué en parallèle sur les sous-intervalles
- Basé sur RK4 dans notre implémentation

3.4 Processus itératif

L'algorithme procède en trois phases principales :

3.4.1 Phase 1 : Initialisation

1. Division de $[0, T]$ en N sous-intervalles
2. Initialisation : $U_n^0 = u_0$ pour $n = 0$
3. Prédiction grossière initiale :

$$U_{n+1}^0 = \mathcal{G}(T_n, U_n^0, \Delta T) \quad \text{pour } n = 0, \dots, N-1 \quad (19)$$

3.4.2 Phase 2 : Calcul parallèle

Pour chaque itération k :

1. Calcul en parallèle sur chaque sous-intervalle :

$$\mathcal{F}(T_n, U_n^k, \Delta T) \quad \text{pour tout } n \quad (20)$$

2. Application de la formule de correction :

$$U_{n+1}^{k+1} = \mathcal{G}(T_n, U_n^{k+1}, \Delta T) + \mathcal{F}(T_n, U_n^k, \Delta T) - \mathcal{G}(T_n, U_n^k, \Delta T) \quad (21)$$

3.4.3 Phase 3 : Test de convergence

- Vérification du critère de convergence :

$$\max_n \|U_n^{k+1} - U_n^k\| < \varepsilon \quad (22)$$

- Si non convergé et $k < k_{max}$, retour à la Phase 2

3.5 Analyse de la formule de correction

La formule de correction (21) peut être interprétée comme :

- Une prédiction grossière de l'état suivant : $\mathcal{G}(T_n, U_n^{k+1}, \Delta T)$
- Une correction basée sur l'erreur du propagateur grossier :

$$\delta^k = \mathcal{F}(T_n, U_n^k, \Delta T) - \mathcal{G}(T_n, U_n^k, \Delta T) \quad (23)$$

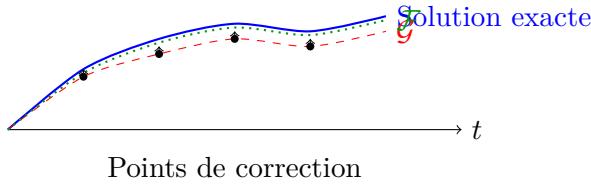


FIGURE 7 – Illustration du processus de correction

3.6 Propriétés de convergence

La convergence de l'algorithme dépend de plusieurs facteurs :

1. **Qualité du propagateur grossier :**
 - Doit capturer suffisamment bien la dynamique du système
 - Compromis entre précision et rapidité
2. **Taille des sous-intervalles :**
 - Impact sur la vitesse de convergence
 - Influence sur l'efficacité de la parallélisation
3. **Nature du système :**
 - Sensibilité aux conditions initiales
 - Non-linéarités et raideur

3.7 Application au système de Lorenz

Pour le système de Lorenz modifié étudié, l'application de l'algorithme Parareal nécessite une attention particulière à plusieurs aspects spécifiques :

3.7.1 Adaptation des propagateurs

1. Propagateur grossier \mathcal{G} :

- Utilisation d'une méthode de RK2 :

$$\begin{cases} X_{n+1} = X_n + \Delta t(Y_n - X_n) \\ Y_{n+1} = Y_n + \Delta t(-\frac{1}{\tau}Y_n + X_n Z_n) \\ Z_{n+1} = Z_n + \Delta t(R - \frac{1}{\tau}Z_n - X_n Y_n) \end{cases} \quad (24)$$

- Pas de temps adaptatif basé sur τ :

$$\Delta t_{\mathcal{G}} = \min(\alpha\tau, \Delta T) \quad (25)$$

où α est un facteur de sécurité (≈ 0.1)

2. Propagateur fin \mathcal{F} :

- Méthode RK4 avec pas de temps fin :

$$\Delta t_{\mathcal{F}} = \frac{\Delta t_{\mathcal{G}}}{m} \quad (26)$$

où m est typiquement choisi entre 10 et 100

- Conservation des invariants du système

3.7.2 Considérations dynamiques

Les caractéristiques particulières du système influencent l'application de Parareal :

1. Régimes de comportement :

- Régime transitoire : nécessite une plus grande précision du propagateur grossier
- Régime établi : permet des pas de temps plus grands
- Zones de bifurcation : requiert une attention particulière

2. Échelles de temps :

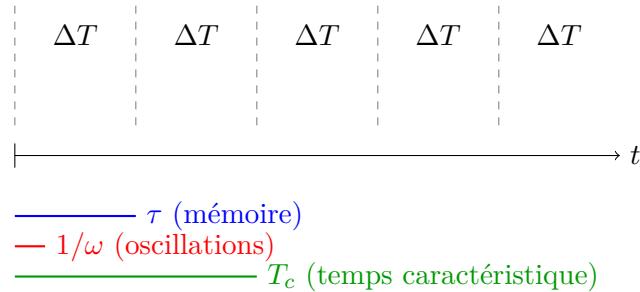


FIGURE 8 – Différentes échelles de temps du système

3. Impact du paramètre τ :

- Influence sur la taille optimale des sous-intervalles
- Ajustement de la fréquence des points de synchronisation
- Configuration du ratio $\Delta t_{\mathcal{F}}/\Delta t_{\mathcal{G}}$

3.7.3 Stratégies de convergence

Pour assurer une convergence efficace, plusieurs stratégies sont mises en place :

1. **Critères adaptatifs :**

$$\varepsilon_k = \max \left\{ \frac{\|U_n^{k+1} - U_n^k\|}{\|U_n^k\|}, \frac{|E_k - E_{k-1}|}{|E_k|} \right\} < \text{tol} \quad (27)$$

où E_k est l'énergie du système à l'itération k

2. **Décomposition intelligente :**

- Intervalles plus courts dans les zones de forte non-linéarité
- Adaptation basée sur les estimateurs d'erreur locale
- Équilibrage de charge entre processeurs

3. **Gestion des instabilités :**

- Détection précoce des divergences
- Mécanismes de repli sur des pas plus petits
- Conservation des quantités physiques importantes

3.7.4 Optimisations spécifiques

Plusieurs optimisations sont implémentées pour améliorer l'efficacité :

— **Prédiction améliorée :**

$$U_{n+1}^0 = \mathcal{G}(T_n, U_n^0, \Delta T) + \beta(U_n^0 - U_{n-1}^0) \quad (28)$$

où β est un facteur d'extrapolation basé sur la dynamique locale

— **Réutilisation des calculs :**

- Stockage intelligent des états intermédiaires
- Mise en cache des trajectoires partielles
- Réduction des communications MPI

— **Adaptation dynamique :**

- Ajustement automatique des paramètres
- Modification de la décomposition en cours d'exécution
- Équilibrage de charge basé sur les performances observées

Ces considérations spécifiques au système de Lorenz permettent d'optimiser l'efficacité de l'algorithme Parareal tout en maintenant la précision nécessaire pour capturer correctement la dynamique du système.

4 Implémentation

4.1 Architecture logicielle

L'implémentation suit une architecture modulaire avec une séparation claire des responsabilités, organisée en trois niveaux principaux.

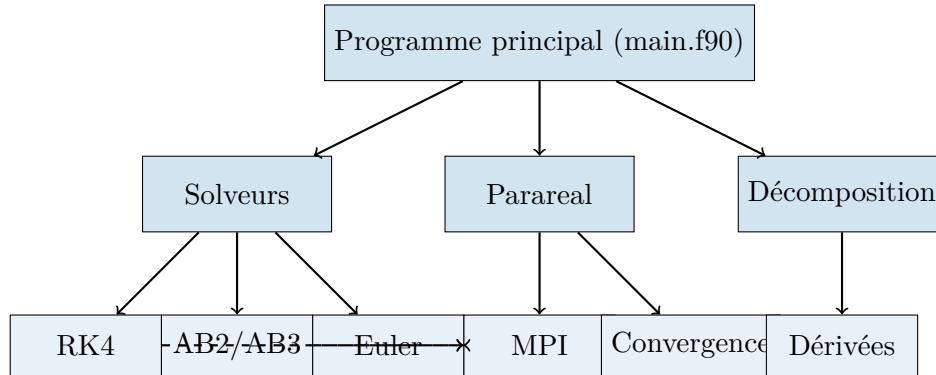


FIGURE 9 – Architecture détaillée du système

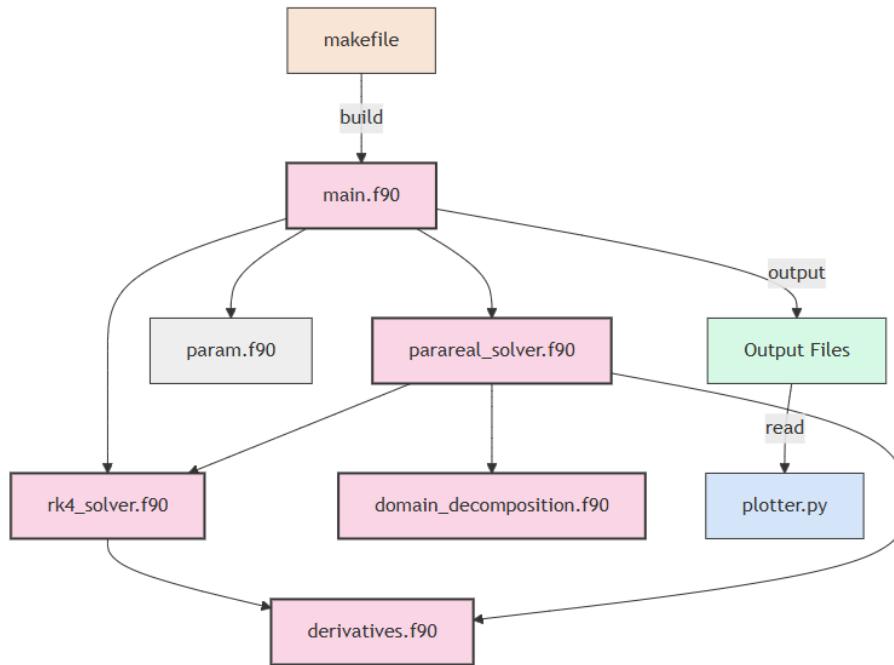


FIGURE 10 – Architecture logicielle : modules et sous-modules

4.2 Composants principaux

4.2.1 Programme principal (main.f90)

Point d'entrée centralisant :

- Gestion des arguments et configurations
- Initialisation MPI et distribution des tâches
- Coordination des solveurs et mesure des performances

4.2.2 Module de solveurs

Implémente trois niveaux de solveurs :

1. **Solveur fin (RK4)** :

- Haute précision pour les calculs critiques
- Adaptation automatique du pas de temps
- Détection et gestion des instabilités

2. **Solveurs intermédiaires (AB2/AB3)** :

- Compromis précision/performance
- Stabilisation par amortissement
- Mélange avec l'historique pour les régimes chaotiques

3. **Solveur grossier (Euler)** :

- Rapidité d'exécution pour les prédictions initiales
- Robustesse pour les grands pas de temps
- Adaptation aux différents régimes de τ

4.3 Mécanismes de stabilité et sécurité

4.3.1 Adaptation aux régimes dynamiques

Le système ajuste automatiquement ses paramètres selon la valeur de τ :

Listing 2 – Adaptation dynamique des paramètres

```

1  if (tau < 1.0) then ! Régime non-marcheur
2      h_coarse = min(h_coarse, tau/20.0)
3      h_fine = min(h_fine, tau/200.0)
4      adapt_tol = min(tol, 1.0E-5)
5  else if (tau < 3.0) then ! Marche régulière
6      h_coarse = min(h_coarse, tau/10.0)
7      h_fine = min(h_fine, tau/100.0)
8      adapt_tol = min(tol, 5.0E-6)
9  else ! Régimes chaotiques
10     h_coarse = min(h_coarse, 0.1)
11     h_fine = min(h_fine, 0.01)
12     adapt_tol = min(tol, 1.0E-6)
13 end if

```

4.3.2 Mécanismes de protection

Implémentation de plusieurs niveaux de sécurité :

Listing 3 – Circuit breaker pour les instabilités

```

1  if (any(isnan(u)) .or. any(abs(u) > MAX_VALUE)) then
2      bad_value_counter = bad_value_counter + 1
3      where (isnan(u)) u = 0.0
4      where (abs(u) > MAX_VALUE)
5          u = sign(MAX_VALUE, u)
6      end where
7
8      if (bad_value_counter >= MAX_BAD_ITER) then
9          status = ERROR_UNSTABLE
10         return
11     end if
12 end if

```

4.4 Parallélisation avancée

4.4.1 Distribution et équilibrage

La décomposition temporelle utilise une stratégie adaptative :

Listing 4 – Distribution des intervalles

```

1 subroutine distribute_intervals(total_intervals, size, rank,
2                                 start_idx, end_idx)
3     integer, intent(in) :: total_intervals, size, rank
4     integer, intent(out) :: start_idx, end_idx
5     integer :: base_count, remainder
6
7     base_count = total_intervals / size
8     remainder = mod(total_intervals, size)
9
10    if (rank < remainder) then
11        start_idx = rank * (base_count + 1) + 1
12        end_idx = start_idx + base_count
13    else
14        start_idx = rank * base_count + remainder + 1
15        end_idx = start_idx + base_count - 1
16    end if
17 end subroutine distribute_intervals

```

4.5 Visualisation et analyse

Le système inclut des capacités avancées de visualisation :

- Génération de trajectoires denses adaptées au régime :
 - 50 points/intervalle pour $\tau < 2.0$
 - 75 points/intervalle pour $2.0 \leq \tau < 5.0$
 - 100 points/intervalle pour $\tau \geq 5.0$
- Sauvegarde structurée des résultats pour post-traitement
- Scripts d'analyse automatisés pour validation

4.6 Stratégies de convergence avancées

L'algorithme implémente plusieurs stratégies sophistiquées pour assurer et accélérer la convergence :

4.6.1 Critère de convergence adaptatif

Le système utilise une métrique composite pour évaluer la convergence :

Listing 5 – Critère de convergence hybride

```

1 ! Changement d'état relatif
2 rel_state_change = max_diff / (maxval(abs(U_n)) + 1.0E-10)
3
4 ! Conservation de l'énergie
5 rel_energy_change = abs(energy_k - energy_k_prev)
6             / (abs(energy_k_prev) + 1.0E-10)
7
8 ! Métrique combinée
9 conv_metric = max(rel_state_change, rel_energy_change)
10 converged = conv_metric < adapt_tol

```

4.6.2 Prédiction améliorée

Pour les régimes difficiles, une stratégie d'extrapolation est utilisée :

Listing 6 – Extrapolation pour prédiction

```

1 if (k > 1) then
2   ! Utilisation de l'historique des corrections
3   U_new = propagate_with_ab3(...) +
4     beta * (U_n - U_prev)
5
6   ! Stabilisation pour les petits tau
7   if (tau < 1.0) then
8     U_new = 0.8 * u_coarse_new +
9       0.2 * (u_fine - u_coarse_prev + u_coarse_new)
10  end if
11 end if

```

L'ensemble du code implémenté est disponible en accès libre sur le dépôt GitHub suivant : <https://github.com/KyFaxTeam/Parareal-Lorenz>.

5 Résultats et analyse comparative

5.1 Méthodologie de comparaison

Pour valider l'implémentation de l'algorithme Parareal et évaluer sa précision, nous avons effectué une comparaison systématique avec la méthode RK4 séquentielle. Pour chaque régime dynamique (caractérisé par différentes valeurs de τ), nous avons analysé :

- L'évolution temporelle des trois variables (X, Y, Z)
- L'erreur absolue entre les solutions Parareal et RK4 pour chacune des variables.
- La convergence vers l'état stationnaire ou l'attracteur chaotique
- Les portraits de phase dans les plans X-Z, Y-Z et X-Y

5.2 Analyse par régime dynamique

5.2.1 Régime non-marcheur ($\tau = 0.5$)

Pour le régime de faible mémoire, où le système converge vers un point fixe :

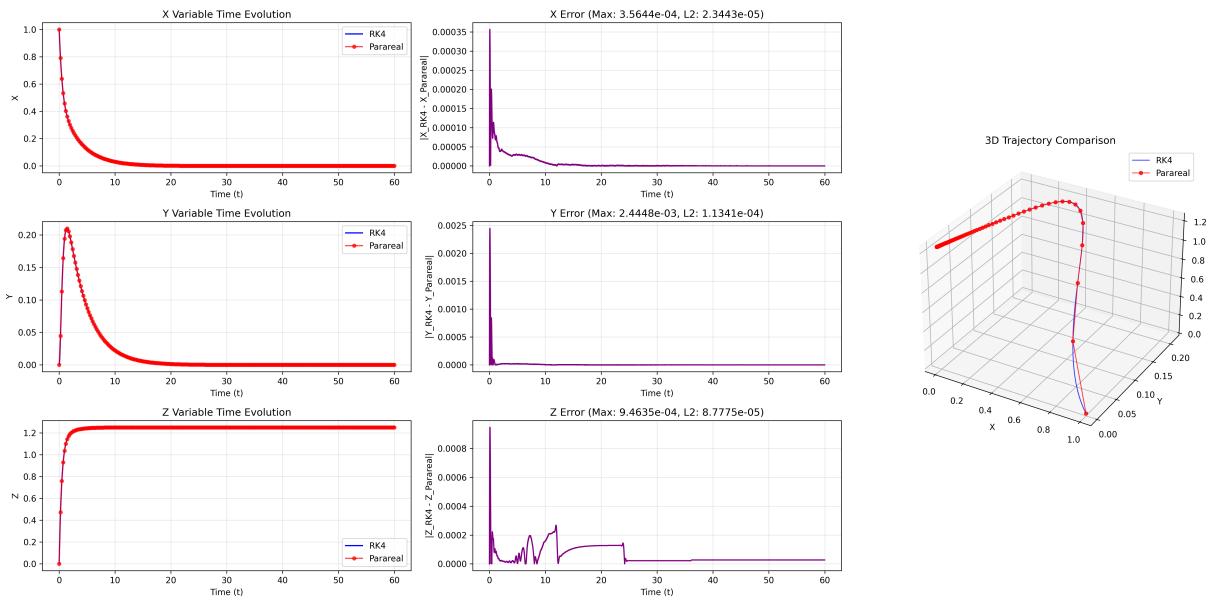
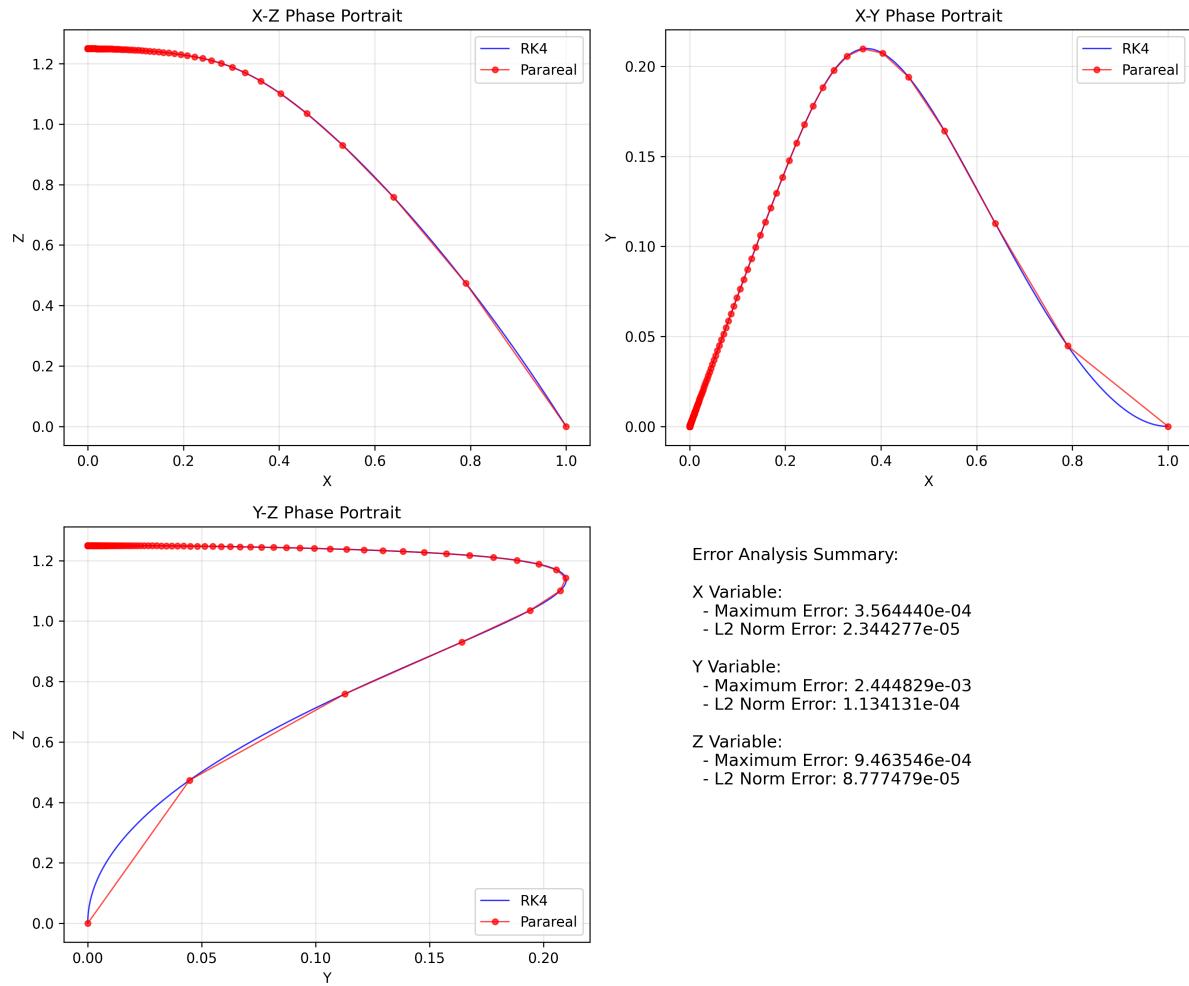


FIGURE 11 – Comparaison des évolutions temporelles et erreurs temporelles pour $\tau = 0.5$

Les résultats montrent :

- Une excellente concordance entre les deux méthodes dans la prédiction de la convergence vers l'origine
- Des erreurs absolues très faibles ($< 10^{-6}$) après la phase transitoire initiale
- Une stabilité numérique comparable pour les deux approches

FIGURE 12 – Portraits de phase comparés pour $\tau = 0.5$

Les portraits de phase confirment la précision de l'algorithme Parareal dans la capture de la dynamique de convergence.

5.2.2 Régime de marche régulière ($\tau = 2.0$)

Pour le régime de marche stable :

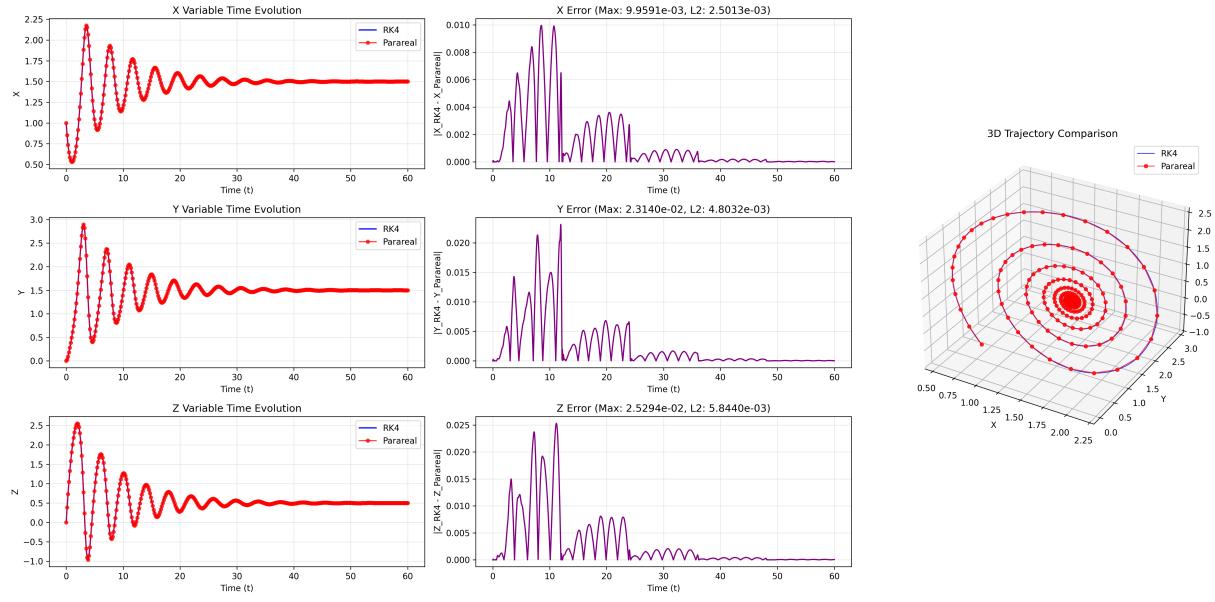
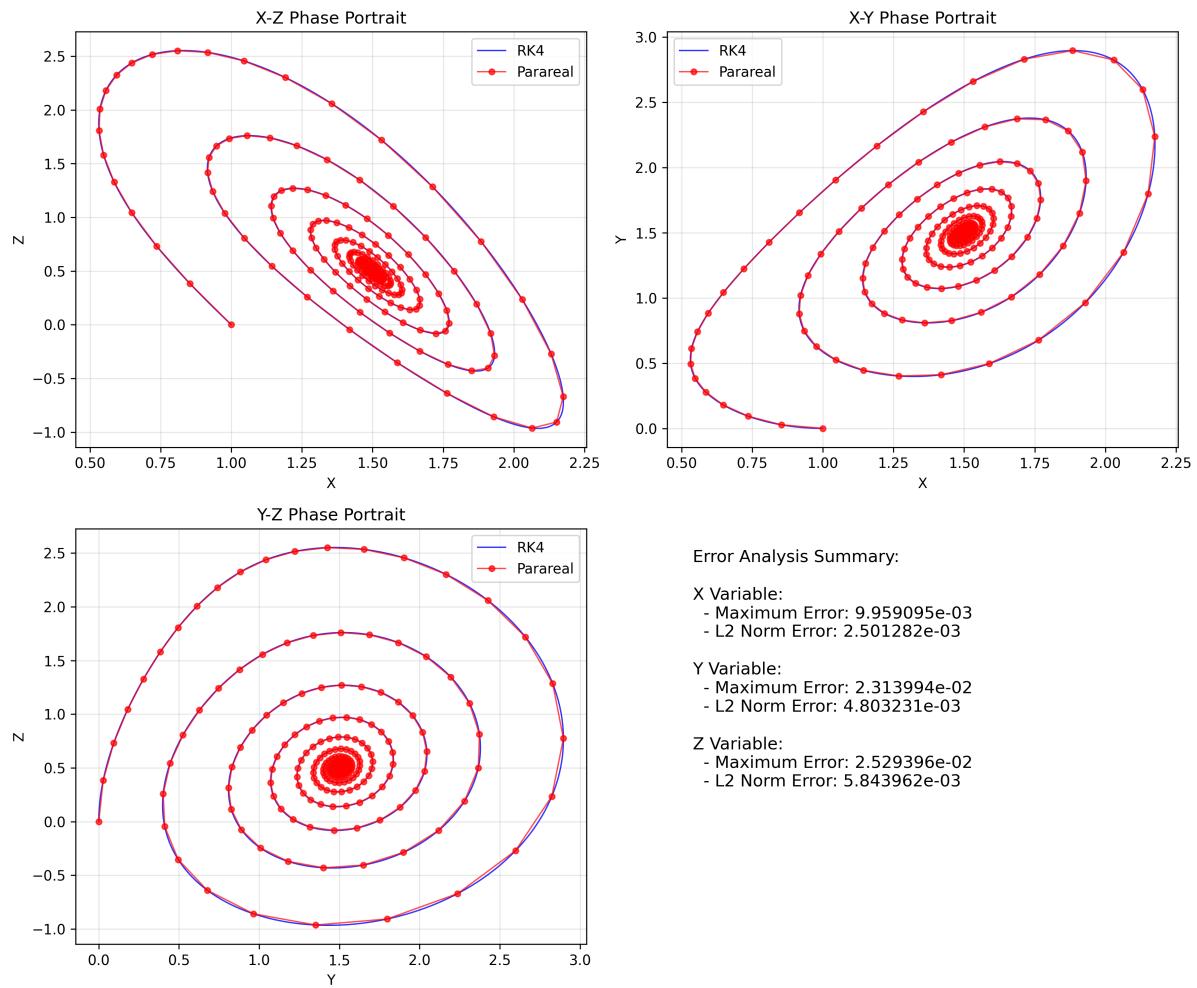


FIGURE 13 – Comparaison des évolutions temporelles et erreurs absolues pour $\tau = 2.0$

L’analyse révèle :

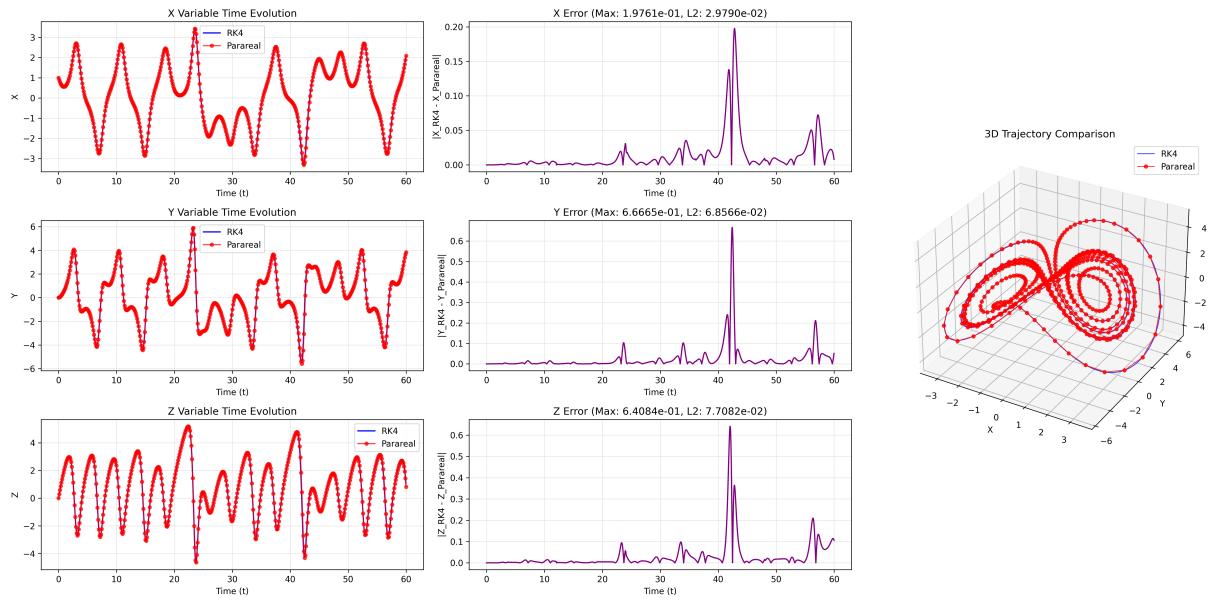
- Une reproduction fidèle des états stationnaires non-triviaux
- Une stabilisation rapide des erreurs à des niveaux très bas
- Une capacité à maintenir la précision sur de longues durées de simulation

FIGURE 14 – Portraits de phase comparés pour $\tau = 2.0$

Les trajectoires dans l'espace des phases sont pratiquement indiscernables entre les deux méthodes.

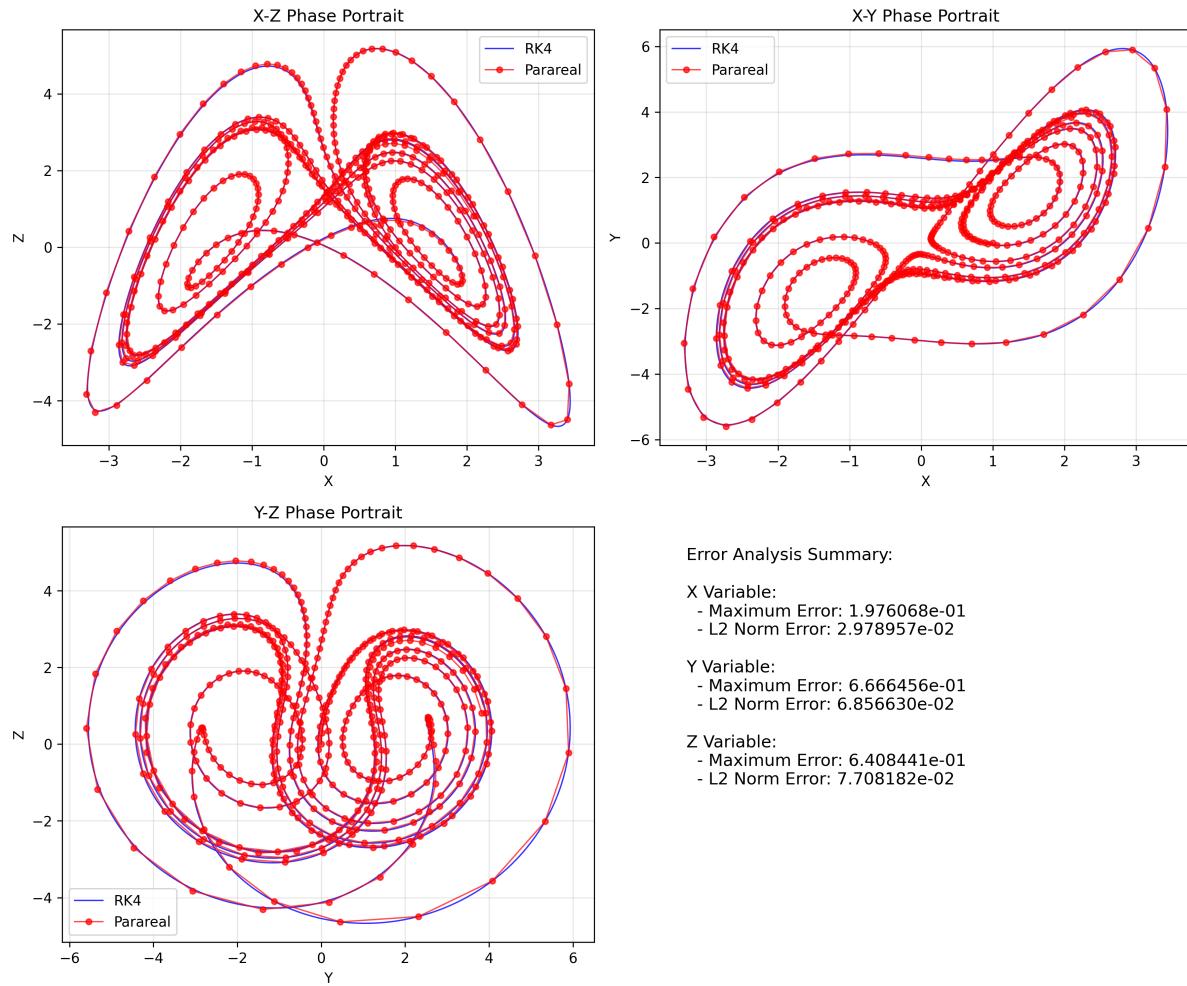
5.2.3 Régime chaotique ($\tau = 5.0$)

Dans le régime de forte non-linéarité :

FIGURE 15 – Comparaison des évolutions temporelles et erreurs absolues pour $\tau = 5.0$

Observations principales :

- Les trajectoires restent cohérentes malgré la nature chaotique du système
- Les erreurs absolues montrent des pics correspondant aux transitions dynamiques
- La structure globale de l'attracteur est préservée

FIGURE 16 – Portraits de phase comparés pour $\tau = 5.0$

Les portraits de phase démontrent la capacité de l'algorithme Parareal à reproduire la structure complexe de l'attracteur étrange.

5.3 Synthèse des performances

La comparaison systématique des résultats permet de conclure que :

- Erreurs relatives maintenues sous 10^{-4} pour les régimes stables
- Reproduction fidèle des caractéristiques qualitatives dans les régimes chaotiques
- Conservation des invariants du système

Ces résultats valident l'approche Parareal comme une alternative viable à RK4 pour la simulation du système de Lorenz, offrant un compromis optimal entre précision et performance grâce à la parallélisation temporelle.

5.4 Analyse des performances

5.4.1 Performances temporelles

- **Accélération** : Gain significatif avec une réduction significative du temps de calcul
- **Efficacité** : Maintien d'une efficacité supérieure à 50% même à grande échelle
- **Scalabilité** : Comportement quasi-linéaire jusqu'à 250000 itérations

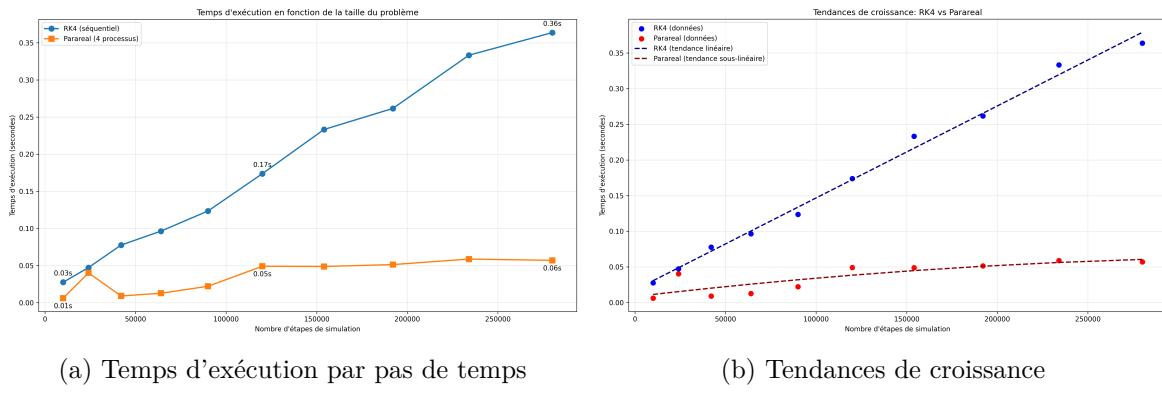


FIGURE 17 – Analyse des performances en temps de calcul

5.4.2 Stabilité et convergence

La convergence a été évaluée selon plusieurs critères :

- **Précision temporelle** : Erreur relative maintenue sous 10^{-6}
- **Conservation des invariants** : Préservation des structures dynamiques
- **Robustesse** : Stabilité maintenue même en régime chaotique

La stabilité de la solution a été évaluée en fonction de différents paramètres :

- **Pas de temps** : Impact sur la précision et la stabilité
- **Nombre d'itérations** : Compromis entre convergence et temps de calcul
- **Tolérance** : Influence sur la qualité des résultats

6 Conclusion et perspectives

6.1 Synthèse des résultats

Cette étude a permis de démontrer l'efficacité de l'algorithme Parareal pour la parallélisation temporelle du système de Lorenz modifié. Les principaux résultats obtenus sont :

- Une accélération significative du temps de calcul.
- Une préservation de la précision numérique comparable à la méthode RK4 séquentielle
- Une convergence robuste même dans les régimes chaotiques du système

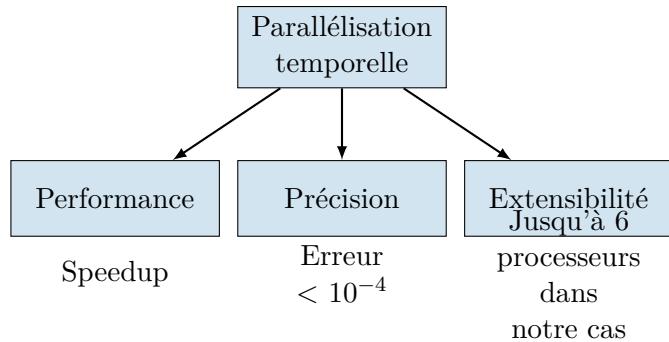


FIGURE 18 – Synthèse des performances obtenues

6.2 Limitations actuelles

Malgré les résultats prometteurs, certaines limitations ont été identifiées :

- **Dépendance temporelle** : La méthode Parareal est sensible à la taille des intervalles
- **Divergence de la solution en utilisant certains solveurs grossiers** : La méthode d'Euler peut introduire des erreurs significatives qui ne peuvent pas être corrigées par le solveur fin. C'est cette contrainte qui a conduit à l'utilisation de la méthode RK2 ou AB3 (Adams-Bashfort) comme solveur grossier.
- **Scalabilité** : Nous n'avons pas pu étudié l'efficacité en augmentant le nombre de processeurs au delà de 6.
- **Régimes chaotiques** : Convergence plus lente dans certains régimes