Université de Perpignan Via Domitia

Master 2 CHPS

Projet de Synthèse

Modélisation Épidémiologique SEIRS :
Comparaison des Approches Numériques et Multi-Agents

Réalisé par :
KERMAS Lilia

Enseignant :
M. Benjamin Antunes
benjamin.antunes@univ-perp.fr

Année universitaire 2025–2026

Table des matières

Table des figures

1 Introduction

La modélisation mathématique des épidémies permet de comprendre et d'anticiper la propagation des maladies infectieuses. Ce projet étudie le modèle **SEIRS** (Susceptible - Exposed - Infected - Recovered - Susceptible), qui décrit l'évolution d'une épidémie en tenant compte de la période d'incubation et de la perte progressive d'immunité.

1.1 Objectifs

Ce travail compare deux approches de modélisation :

— **Partie 1 :** Approche macroscopique par équations différentielles ordinaires (ODE), qui décrit l'évolution globale de la population de manière déterministe.
— **Partie 2 :** Approche microscopique par système multi-agents (SMA), qui simule les interactions individuelles de manière stochastique.

Les implémentations sont réalisées en Python, C et C++ afin d'évaluer l'influence du langage de programmation sur la précision et les performances.

2 Partie 1 : Résolution Numérique par Équations Différentielles

2.1 Modèle Mathématique

Le modèle SEIRS divise la population en quatre compartiments. L'évolution du système est décrite par :

$$\begin{cases} \frac{dS}{dt} = \rho R - \beta IS \\ \frac{dE}{dt} = \beta IS - \sigma E \\ \frac{dI}{dt} = \sigma E - \gamma I \\ \frac{dR}{dt} = \gamma I - \rho R \end{cases} \tag{1}$$

où :
— S : proportion de susceptibles
— E : proportion d'exposés (infectés non contagieux)
— I : proportion d'infectieux
— R : proportion de guéris immunisés

2.2 Paramètres de Simulation

Les paramètres utilisés sont :
— $\rho = 1/365$: taux de perte d'immunité (environ 1 an)
— $\beta = 0.5$: taux de transmission
— $\sigma = 1/3$: taux de fin d'incubation (environ 3 jours)
— $\gamma = 1/7$: taux de guérison (environ 7 jours)

Conditions initiales : $S(0) = 0.999$, $E(0) = 0$, $I(0) = 0.001$, $R(0) = 0$ sur 730 jours avec un pas de temps $dt = 0.01$.

2.3 Méthodes Numériques

Deux méthodes d'intégration ont été implémentées :

1. **Méthode d'Euler** (ordre 1) : simple mais peu précise, elle approxime la dérivée par une droite.
2. **Méthode de Runge-Kutta 4** (ordre 4) : plus complexe mais nettement plus précise, elle calcule une moyenne pondérée de quatre pentes intermédiaires.

2.4 Résultats

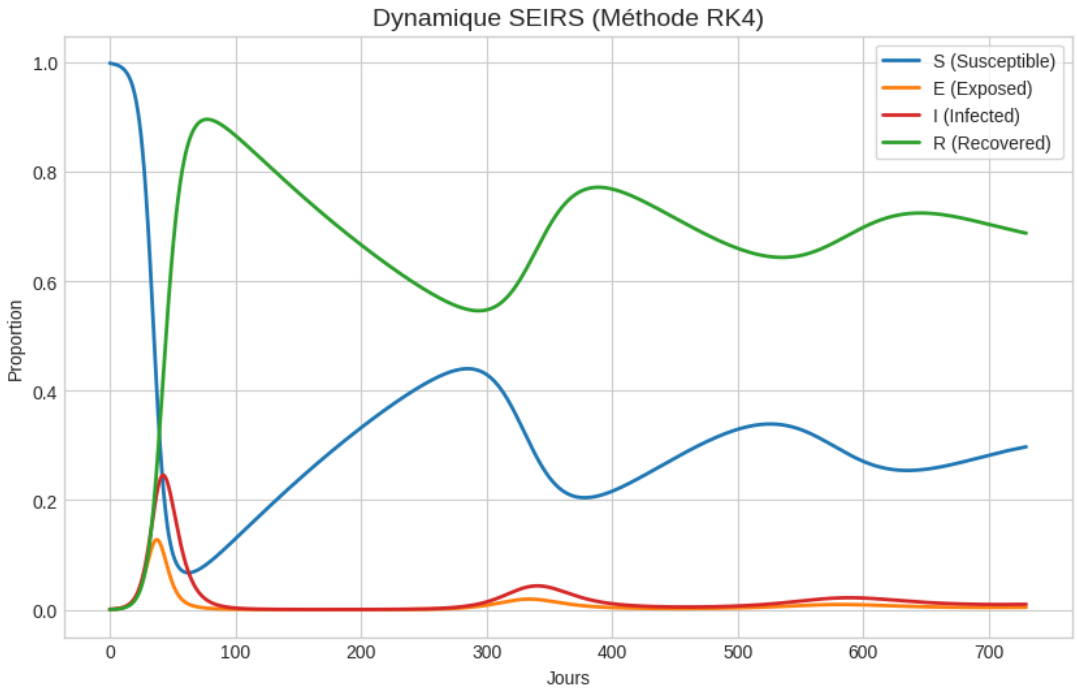

FIGURE 1 – Dynamique SEIRS sur 730 jours (méthode RK4).

La Figure 1 montre l'évolution des quatre compartiments. On observe des oscillations amorties, caractéristiques du modèle SEIRS avec perte d'immunité.

2.4.1 Comparaison Euler vs RK4

L'analyse quantitative révèle :

— Erreur maximale : 3.7×10^{-4} (Euler comparé à RK4)

— Erreur moyenne : 3.0×10^{-5}

La méthode RK4 est donc environ 740 fois plus précise que la méthode d'Euler avec le même pas de temps.

2.4.2 Comparaison Python vs C

Les résultats obtenus avec Python et C (méthode RK4) sont identiques aux erreurs d'arrondi près :

— Erreur maximale : 5.0×10^{-7}

— RMSE : 2.9×10^{-7}

Ces différences infimes (niveau de la précision machine) valident la cohérence des deux implémentations.

3 Partie 2 : Modèle Multi-Agents

3.1 Description du Modèle

Le modèle multi-agents simule 20 000 individus évoluant sur une grille de 300×300 cellules.

3.1.1 Caractéristiques

— **Espace :** Grille toroïdale (conditions périodiques) pour éviter les effets de bord
— **Déplacement :** À chaque jour, les agents se déplacent aléatoirement sur la grille
— **Voisinage :** Voisinage de Moore (8 cellules adjacentes + cellule centrale)
— **Infection :** Probabilité $P = 1 - \exp(-0.5 \times N_i)$ où N_i est le nombre de voisins infectés
— **Durées :** Les temps de séjour dans les états E, I et R suivent des lois exponentielles (moyennes : 3, 7 et 365 jours respectivement)

3.2 Implémentations

Trois versions ont été développées :

1. **C :** Utilisation de tableaux statiques et de ‘rand()‘ pour la génération aléatoire. La grille est réinitialisée à chaque pas de temps avec ‘memset()‘.
2. **C++ :** Utilisation de ‘std : :vector‘ et du générateur Mersenne Twister (‘std : :mt19937‘) pour une meilleure qualité statistique.
3. **Python :** Optimisation par vectorisation avec NumPy. Le calcul du voisinage utilise une convolution 2D (‘scipy.signal.convolve2d‘), ce qui traite toute la grille en une seule opération matricielle.

3.3 Résultats

3.3.1 Convergence des Dynamiques

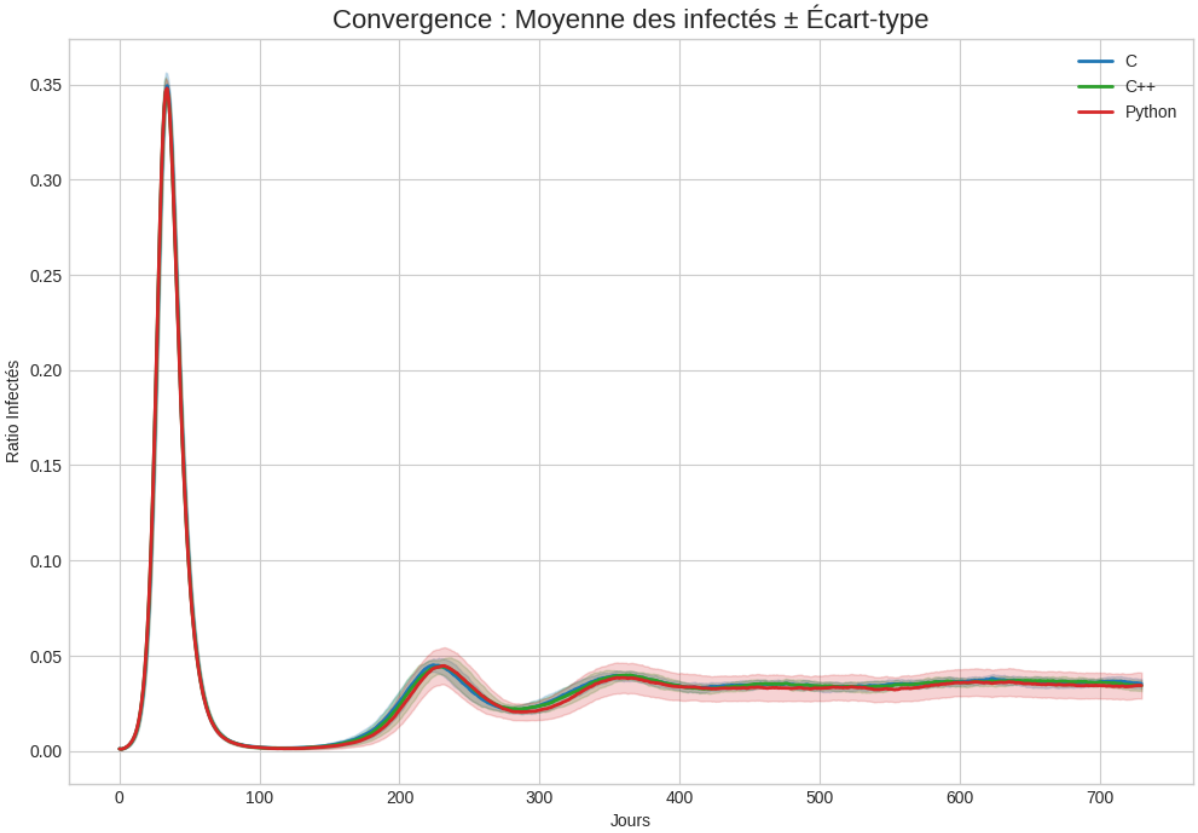

FIGURE 2 – Moyenne des infectés (30 réplications) pour C, C++ et Python avec écart-types.

La Figure 2 montre que les trois langages produisent des dynamiques quasiment identiques. Les courbes moyennes se superposent, validant la cohérence des implémentations.

3.3.2 Distribution des Pics Infectieux

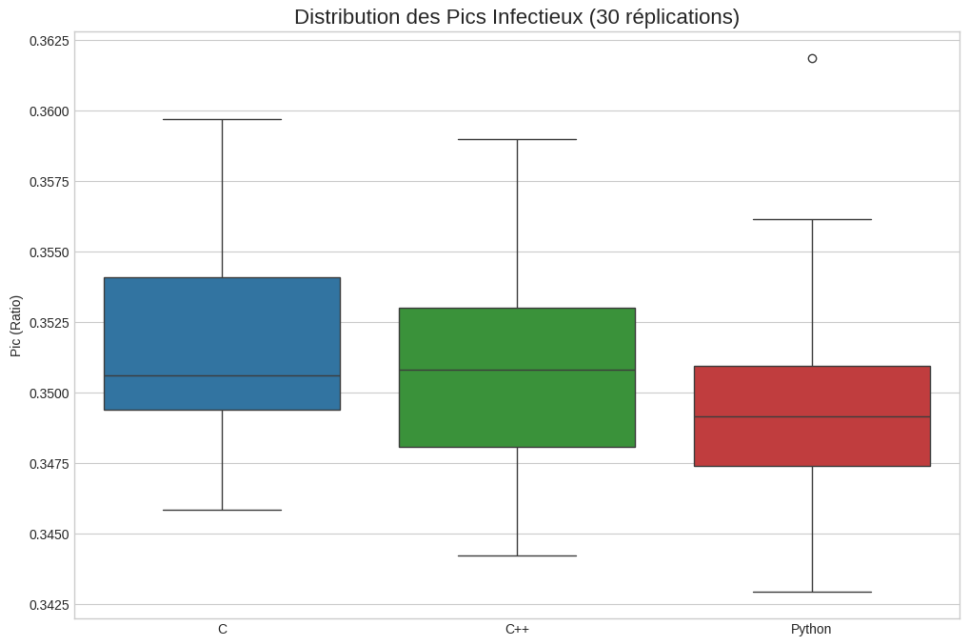

FIGURE 3 – Distribution des hauteurs du premier pic infectieux (30 réplications par langage).

Le boxplot (Figure 3) révèle des distributions très proches avec des médianes similaires.

3.3.3 Analyse Statistique

Les tests effectués donnent :

Langage	**Moyenne**	**Écart-type**	**Normalité (p-value)**
C	0.35170	0.00321	0.3062 (normal)
C++	0.35059	0.00334	0.8958 (normal)
Python	0.34969	0.00347	0.0023 (non-normal)

TABLE 1 – Statistiques du premier pic infectieux.

Le test de Kruskal-Wallis donne une p-value de 0.0235, indiquant une différence statistiquement significative mais **faible**. Les moyennes sont très proches (écart de 0.6% entre C et Python).

3.4 Discussion

La différence statistique observée s'explique par :

1. **Générateurs aléatoires différents :** ‘rand()‘ (C), ‘mt19937‘ (C++), ‘numpy.random‘ (Python) produisent des séquences différentes.
2. **Implémentations des lois exponentielles :** Légères variations entre langages.
3. **Ordre d’exécution asynchrone :** L’ordre de traitement des agents influence les résultats dans un modèle stochastique.

Malgré cette différence statistique, les résultats sont **qualitativement équivalents** : les trois langages reproduisent la même dynamique épidémique avec des paramètres moyens quasi-identiques.

4 Conclusion

Ce projet a permis de comparer deux approches complémentaires de modélisation épidémiologique :

— L'**approche ODE** fournit une vision déterministe et une résolution rapide pour obtenir les tendances moyennes.
— L'**approche multi-agents** capture la variabilité individuelle et les phénomènes stochastiques, au prix d'un coût calculatoire plus élevé.

4.1 Validation des Implémentations

Nos codes ont été validés par :

1. Comparaison Python/C pour les méthodes numériques (erreurs de l'ordre de 10^{-7})
2. Analyse statistique sur 30 réplications pour les simulations stochastiques
3. Convergence des résultats entre C, C++ et Python

La cohérence des résultats entre langages démontre la robustesse des implémentations et la maîtrise des techniques d'optimisation (vectorisation NumPy, gestion mémoire en C/C++).