Université de Perpignan Via Domitia (UPVD)
UFR Sciences Exactes et Expérimentales - SEE
Département d'Informatique

Projet de synthèse

Simulation épidémiologique SEIRS

Réalisé par

- Bouzidi Adel

2025/2026

Contents

List of Figures

1 Introduction

La modélisation mathématique et numérique s'impose aujourd'hui comme un outil essentiel pour décrypter et anticiper la dynamique des maladies infectieuses. Selon la finesse d'analyse souhaitée, la littérature propose diverses échelles de description : des modèles continus, régis par des équations différentielles, jusqu'aux approches discrètes qui simulent précisément les interactions à l'échelle individuelle.

Ce projet se propose d'explorer un modèle épidémiologique de type SEIRS à travers deux regards complémentaires. Dans un premier temps, nous privilégierons une approche macroscopique basée sur un système d'équations différentielles ordinaires (EDO), afin de retracer l'évolution globale des différentes strates de la population. Dans un second temps, nous développerons une simulation multi-agent. Ce changement d'échelle permettra d'intégrer la complexité des échanges individuels et d'analyser l'influence de la stochasticité sur la propagation.

2 Partie 1 – Modèle SEIRS par équations différentielles

2.1 Présentation du modèle SEIRS

Le modèle SEIRS s'inscrit parmi les structures compartimentales fondamentales de l'épidémiologie. Il permet de segmenter une population en quatre groupes distincts, définis par l'état de santé des individus et la progression de la pathologie.

Le cycle débute avec le compartiment S (Sains), regroupant les sujets vulnérables à l'infection. Vient ensuite la phase E (Exposés), qui isole les individus en période d'incubation : ils portent le pathogène mais ne le transmettent pas encore. La phase de contagiosité est représentée par le compartiment I (Infectieux), tandis que le groupe R (Rétablis) rassemble les individus ayant surmonté la maladie et bénéficiant d'une immunité temporaire.

La spécificité du modèle SEIRS réside dans ce dernier point : à l'inverse des schémas SIR ou SEIR, il intègre une perte d'immunité progressive. Ce retour possible vers l'état "Susceptible" rend ce modèle particulièrement pertinent pour les maladies où la protection immunitaire s'estompe avec le temps.

L'étude s'appuie toutefois sur quelques hypothèses simplificatrices nécessaires. La population est considérée comme "bien mélangée" (homogène), postulant que chaque individu peut interagir de manière équivalente avec ses pairs. Enfin, dans sa forme standard, le modèle demeure déterministe : il retrace une trajectoire moyenne sans intégrer les aléas ou les fluctuations propres aux interactions individuelles.

2.2 Équations différentielles et paramètres

Le modèle SEIRS est décrit par un système de quatre équations différentielles ordinaires, représentant l'évolution temporelle des proportions $S(t)$, $E(t)$, $I(t)$ et $R(t)$ dans la population :

$$\frac{dS}{dt} = \rho R - \beta SI, \tag{1}$$

$$\frac{dE}{dt} = \beta SI - \sigma E, \tag{2}$$

$$\frac{dI}{dt} = \sigma E - \gamma I, \tag{3}$$

$$\frac{dR}{dt} = \gamma I - \rho R. \tag{4}$$

Le paramètre β représente le taux de transmission de la maladie.

Le paramètre σ correspond à l'inverse de la durée moyenne de la période d'incubation.

Le paramètre γ est l'inverse de la durée moyenne de la période infectieuse, tandis que ρ modélise la perte d'immunité.

Dans ce projet, les valeurs numériques des paramètres sont imposées et données par :

$$\rho = \frac{1}{365}, \quad \beta = 0.5, \quad \sigma = \frac{1}{3}, \quad \gamma = \frac{1}{7}.$$

Les conditions initiales correspondent à une population majoritairement susceptible, avec une faible proportion d'individus infectieux :

$$S(0) = 0.999, \quad E(0) = 0, \quad I(0) = 0.001, \quad R(0) = 0.$$

À tout instant, la somme des proportions est conservée :

$$S(t) + E(t) + I(t) + R(t) = 1.$$

Cette propriété constitue un critère important de cohérence et de stabilité numérique lors des simulations.

2.3 Méthodes numériques de résolution

Le système d'équations différentielles du modèle SEIRS ne possède pas de solution analytique explicite. Il est alors nécessaire de recourir à des méthodes numériques pour approximer la solution

Dans ce projet deux méthodes de résolution numérique sont utilisées et comparées. La première est la méthode d'Euler explicite, qui est une méthode d'ordre 1. Elle consiste à approximer la solution à l'instant suivant à partir de la valeur courante et de la dérivée du système, Cette méthode est simple à implémenter mais peut présenter des problèmes de précision et de stabilité si le pas de temps est trop grand.

La seconde est la méthode de Runge–Kutta d'ordre 4. Cette méthode utilise plusieurs évaluations intermédiaires de la dérivée afin d'obtenir une approximation plus précise de la solution. Elle est d'ordre plus élevé que la méthode d'Euler et offre une meilleure stabilité numérique, au prix d'un coût de calcul plus important.

Les simulations sont réalisées sur une durée totale de 730 jours, avec un pas de temps constant. La comparaison entre ces deux méthodes permet d'analyser l'influence du schéma numérique sur la précision des résultats et la stabilité de la solution obtenue.

2.4 Implémentation dans les différents langages

2.4.1 Environnement et organisation

L'ensemble du projet a été développé et exécuté sur un système GNU/Linux (Ubuntu), en utilisant uniquement des outils standards disponibles sur la plateforme. Cette approche garantit la portabilité et la reproductibilité des résultats et la compatibilité avec les exigences du sujet.

Pour la Partie 1 du projet un environnement Python isolé a été mis en place à l'aide de `venv`. Les bibliothèques utilisées sont limitées à `NumPy` pour le calcul numérique et `Matplotlib` pour la visualisation des résultats et `Pandas` pour la manipulation des données.

Le projet est structuré de manière modulaire afin de séparer clairement le code, les données et les figures et le rapport. L'arborescence principale est organisée comme ça :

- `src/` : contient l'ensemble des codes sources du projet.
  - `part1_seirs_ode/python/` : implémentation en Python du modèle SEIRS continu, incluant les méthodes numériques d'Euler explicite et de Runge–Kutta d'ordre 4.
  - `part1_seirs_ode/c/` : répertoire destiné à l'implémentation équivalente en langage C.

- `data/` : regroupe les données produites par les simulations.
  - `part1_seirs_ode/` : fichiers CSV contenant l'évolution temporelle des compartiments S, E, I et R pour les différentes méthodes numériques.
- `figures/` : contient les figures générées à partir des résultats.
  - `part1/` : graphiques illustrant l'évolution temporelle des variables du modèle SEIRS pour la Partie 1.
- `notebooks/` : répertoire réservé aux notebooks Python utilisés pour l'analyse et la comparaison des résultats numériques.
- `rapport/` : contient le rapport LaTeX du projet.

Cette organisation assure une séparation entre les différentes composantes du projet et facilite l'ajout progressif de nouvelles implémentations et analyses au cours de l'avancement du travail.

2.4.2 Implémentation actuelle en Python

Dans l'état actuel du projet, la Partie 1 a été implémentée en langage Python afin de valider le modèle SEIRS continu et les méthodes numériques associées. Cette implémentation sert de référence pour la suite du travail.

Un script Python unique a été développé pour cette étape :

- `src/part1_seirs_ode/python/seirs_part1.py`

Ce script implémente explicitement le système d'équations différentielles du modèle SEIRS, ainsi que 2 méthodes de résolution numérique : la méthode d'Euler explicite et la méthode de Runge–Kutta d'ordre 4. Les paramètres du modèle et les conditions initiales sont ceux imposés sur le sujet.

La simulation est réalisée sur une durée totale de 730 jours, avec un pas de temps constant de un jour. À chaque pas de temps, les valeurs des compartiments S, E, I et R sont calculées et stockées afin de permettre une analyse ultérieure.

2.4.3 Résultats numériques obtenus en Python

Les simulations réalisées en Python produisent, pour chaque méthode numérique un fichier de sortie au format CSV contenant l'évolution temporelle des proportions des compartiments S, E, I et R.

Les fichiers suivants ont été générés :

- `data/part1_seirs_ode/python_euler.csv`
- `data/part1_seirs_ode/python_rk4.csv`

À partir de ces fichiers, des figures illustrant l'évolution temporelle des différents compartiments ont été produites. ces figures montrent la dynamique caractéristique du modèle SEIRS avec une montée rapide du nombre d'individus infectieux au début de la simulation, suivie d'une phase de guérison et d'une perte progressive de l'immunité, entraînant l'apparition de vagues épidémiques successives.

La Figure 1 présente les résultats obtenus avec la méthode d'Euler explicite, tandis que la Figure 2 montre les résultats correspondants à la méthode de Runge–Kutta d'ordre 4.

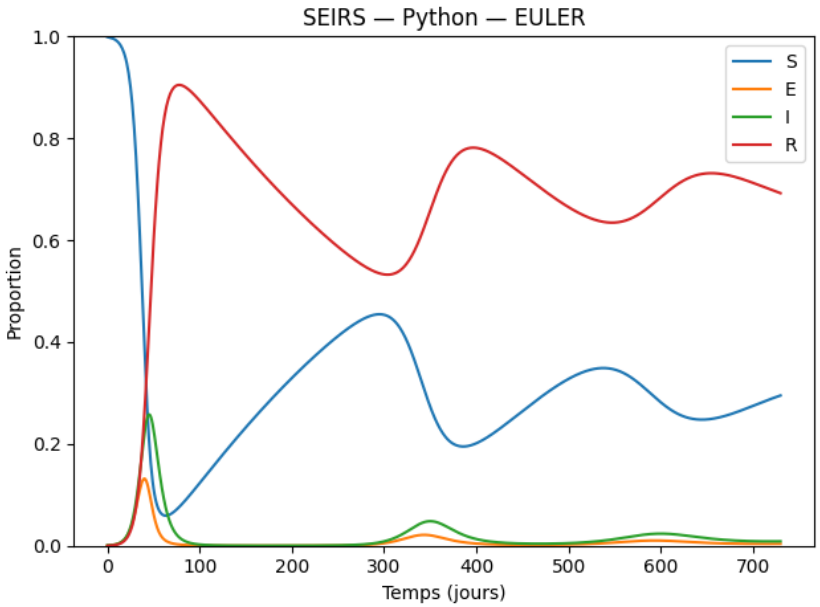

Figure 1: Évolution temporelle des compartiments S, E, I et R obtenue avec la méthode d'Euler explicite en Python.

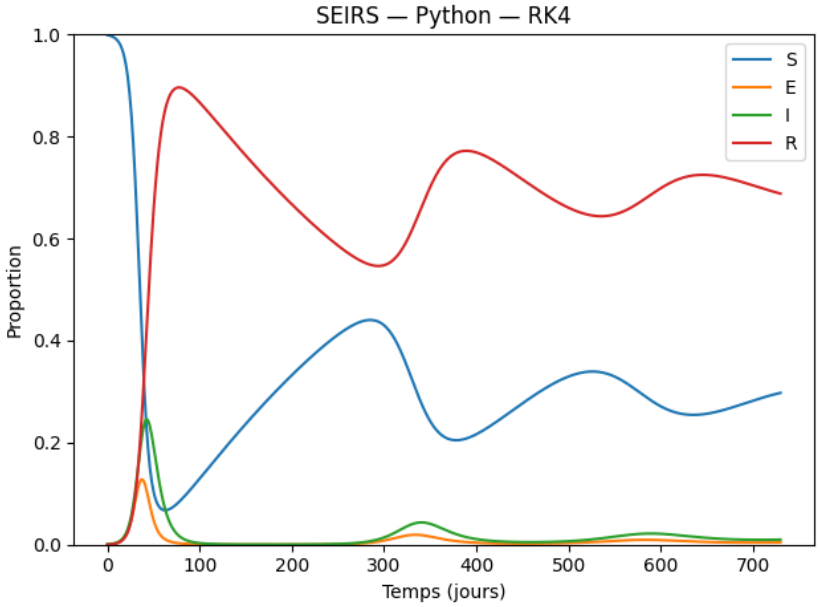

Figure 2: Évolution temporelle des compartiments S, E, I et R obtenue avec la méthode de Runge–Kutta d'ordre 4 en Python.

Les deux méthodes numériques conduisent à des dynamiques qualitativement similaires, tout en présentant de légères différences quantitatives, surtout au niveau de la hauteur et de la position temporelle des pics infectieux. Ces différences seront analysées plus en détail lors de la comparaison entre méthodes et entre langages.

2.4.4 Implémentation en langage C

Après la validation du modèle SEIRS continu en Python, une implémentation équivalente a été réalisée en langage C afin d'évaluer l'influence du langage de programmation sur les résultats

numériques et les performances de calcul.

L'implémentation en C reprend strictement le même modèle mathématique, les mêmes paramètres et les mêmes conditions initiales que la version Python. Les deux méthodes numériques utilisées précédemment, à savoir la méthode d'Euler explicite et la méthode de Runge–Kutta d'ordre 4, ont été implémentées manuellement sans recours à des bibliothèques externes de calcul scientifique.

Le code source principal est contenu dans le fichier suivant :

- `src/part1_seirs_ode/c/seirs_part1.c`

Le programme calcule l'évolution temporelle des compartiments S, E, I et R sur une durée de 730 jours, avec un pas de temps constant de un jour. À chaque itération, les valeurs calculées sont écrites dans un fichier de sortie au format CSV, de manière strictement identique à l'implémentation Python, afin de permettre une comparaison directe des résultats.

2.4.5 Résultats numériques obtenus en langage C

L'exécution du programme en langage C produit, pour chaque méthode numérique, un fichier CSV contenant l'évolution temporelle des compartiments du modèle SEIRS. Les fichiers suivants ont été générés :

- `data/part1_seirs_ode/c_euler.csv`
- `data/part1_seirs_ode/c_rk4.csv`

Les résultats obtenus en langage C présentent une dynamique globale similaire à celle observée avec l'implémentation Python. On observe une phase initiale de croissance rapide du nombre d'individus infectieux, suivie d'une phase de guérison et d'une diminution progressive de l'immunité, conduisant à l'apparition de vagues épidémiques successives.

Les figures correspondantes montrent que les courbes des compartiments S, E, I et R obtenues en C sont visuellement indiscernables de celles obtenues en Python pour une même méthode numérique. Cette similarité visuelle sera quantifiée et analysée plus précisément dans la section dédiée à la comparaison entre langages.

2.4.6 Méthodologie de comparaison entre langages et méthodes

Afin de bien comparer les implémentations du modèle SEIRS et les méthodes numériques utilisées, une méthodologie commune a été mise en place pour l'ensemble des simulations.

Les implémentations en Python et en langage C produisent des fichiers de sortie au format CSV contenant, pour chaque pas de temps, les valeurs des compartiments S, E, I et R. Ces fichiers ont été générés avec des paramètres strictement identiques (durée de 730 jours, pas de temps de un jour, mêmes conditions initiales et mêmes coefficients du modèle).

Un notebook Python dédié a ensuite été utilisé pour charger et analyser les quatre fichiers CSV produits (`python_euler`, `python_rk4`, `c_euler` et `c_rk4`). La comparaison repose sur trois axes principaux :

- la superposition des courbes temporelles des compartiments afin d'évaluer la similarité visuelle des dynamiques obtenues ;
- le calcul des écarts numériques maximaux absolus entre les solutions Python et C pour chaque compartiment ;
- l'extraction d'indicateurs caractéristiques, tels que la hauteur et la position temporelle du pic infectieux.

Cette méthodologie permet de distinguer clairement les différences liées au langage de programmation de celles induites par le choix du schéma numérique.

2.5 Comparaison entre langages

La comparaison entre les implémentations Python et C a été réalisée en superposant les courbes temporelles des compartiments S, E, I et R obtenues pour une même méthode numérique.

La Figure 3 présente la comparaison entre Python et C pour la méthode d'Euler explicite. Les courbes sont quasi parfaitement superposées sur l'ensemble de la simulation, ce qui indique une dynamique strictement équivalente entre les deux langages.

La Figure 4 montre la meme comparaison pour la méthode de Runge–Kutta d'ordre 4. Là encore, aucune différence visuelle significative n'est observable entre les résultats Python et C.

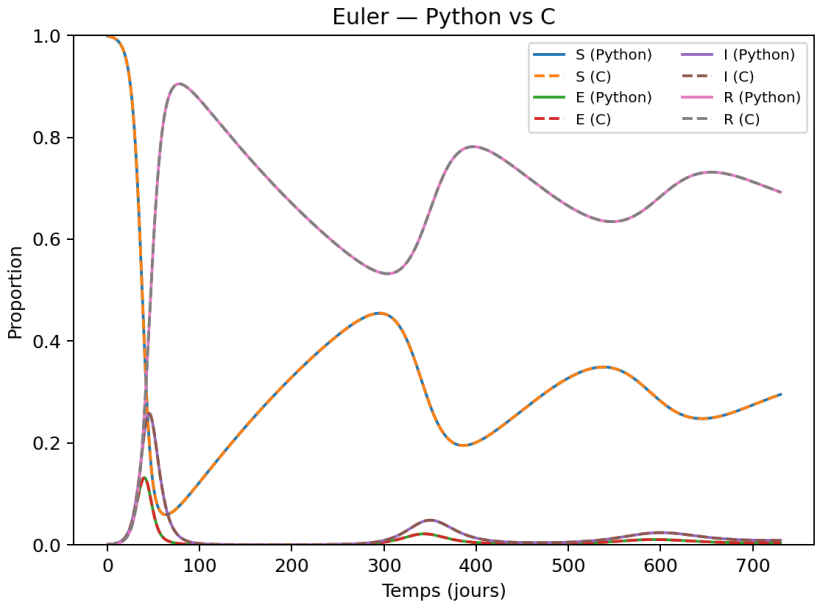

Figure 3: Comparaison des résultats Python et C pour la méthode d'Euler explicite.

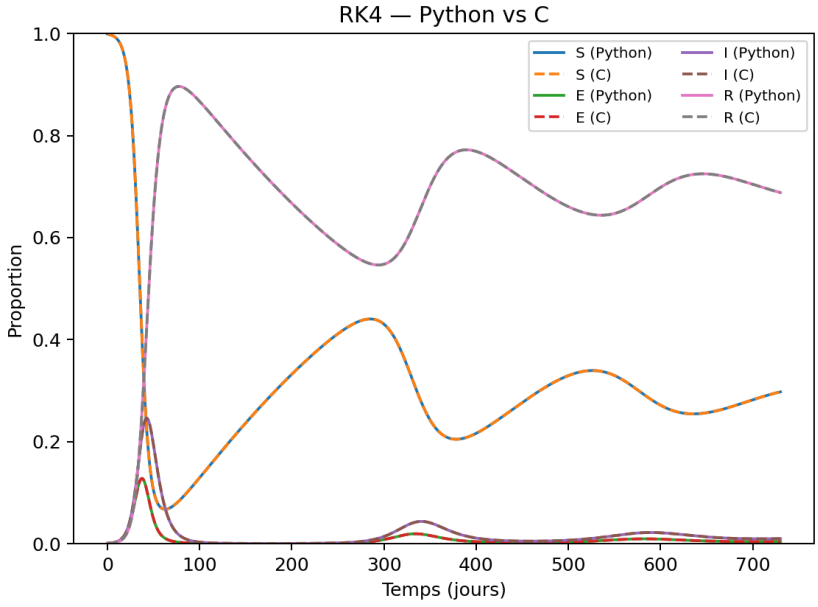

Figure 4: Comparaison des résultats Python et C pour la méthode de Runge–Kutta d'ordre 4.

Une analyse quantitative confirme ces observations. Les différences maximales absolues entre

les résultats Python et C sont de l'ordre de 5×10^{-13} pour tous les compartiments et pour les deux méthodes numériques. Ces écarts sont dus uniquement aux arrondis en arithmétique flottante et peuvent être considérés comme négligeables.

Ces résultats montrent que le choix du langage de programmation n'influence pas la solution numérique obtenue à précision machine près.

2.6 Comparaison entre méthodes numériques

La comparaison entre les méthodes numériques d'Euler explicite et de Runge–Kutta d'ordre 4 a été réalisée séparément pour chaque langage afin d'évaluer l'influence du schéma numérique sur la solution du modèle SEIRS.

La Figure 5 montre la comparaison entre Euler et RK4 pour l'implémentation Python. Les deux méthodes conduisent à une dynamique globale similaire mais des différences quantitatives apparaissent, surtout au niveau de l'amplitude et de la position temporelle du pic infectieux.

Des comportements similaires sont observés pour l'implémentation en langage C, comme illustré par la Figure 6. Les écarts constatés sont du même ordre de grandeur que ceux obtenus en Python, ce qui confirme que ces différences sont dues au schéma numérique et non au langage de programmation.

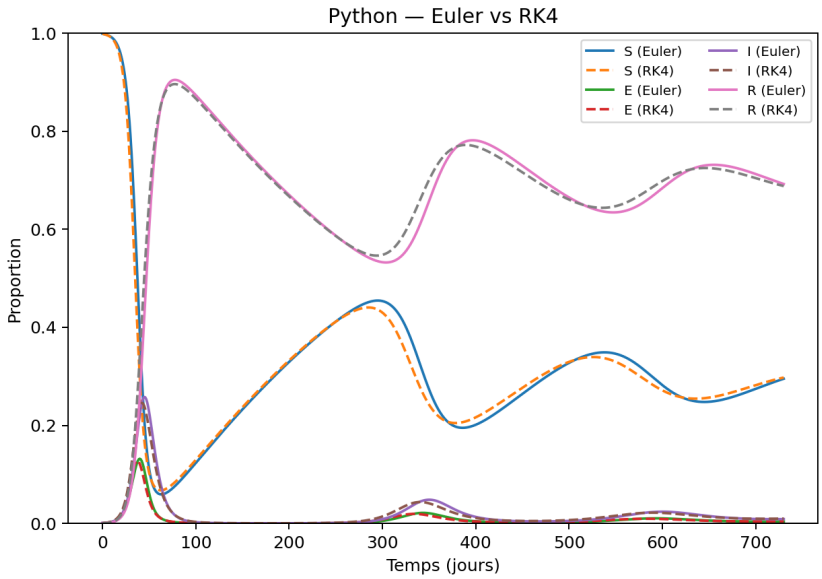

Figure 5: Comparaison entre les méthodes d'Euler et de Runge–Kutta d'ordre 4 pour l'implémentation Python.

L'analyse du pic infectieux souligne l'influence déterminante du schéma numérique sur les résultats obtenus. Avec un pas de temps fixé à un jour, la méthode d'Euler estime un pic d'environ 0,258 au 45^{e} jour. À l'inverse, l'algorithme de Runge–Kutta d'ordre 4 (RK4) aboutit à une prévision plus modérée, avec un pic de 0,246 dès le 43^{e} jour. Il est à noter que ces résultats sont strictement identiques entre les implémentations en Python et en C, confirmant la fiabilité du code.

Ces écarts mettent en exergue la sensibilité des systèmes dynamiques au choix du solveur : la précision de la simulation dépend directement de l'aptitude de la méthode à minimiser l'erreur numérique accumulée.

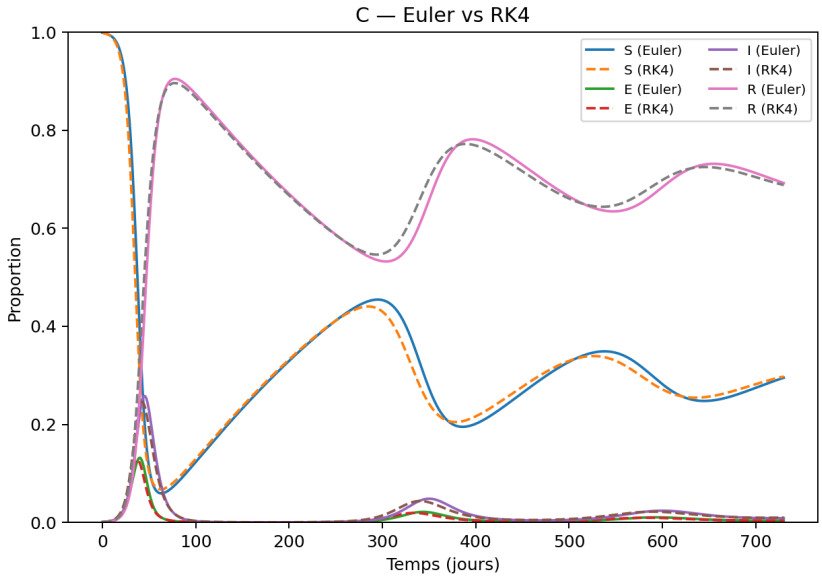

Figure 6: Comparaison entre les méthodes d'Euler et de Runge–Kutta d'ordre 4 pour l'implémentation en langage C.

2.7 Discussion

Cette première partie du projet a permis d'analyser le modèle épidémiologique SEIRS à l'aide d'un cadre macroscopique basé sur des équations différentielles ordinaires. Les simulations montrent une dynamique réaliste caractérisée par l'apparition de vagues épidémiques successives dues à la perte progressive de l'immunité.

La comparaison des méthodes numériques montre que la méthode d'Euler explicite et la méthode de Runge–Kutta d'ordre 4 conduisent à des dynamiques globalement similaires tout en présentant des différences quantitatives, surtout au niveau de l'amplitude et de la position temporelle du pic infectieux. Ces écarts sont cohérents avec la différence d'ordre de précision entre les deux schémas.

Par ailleurs, les implémentations en Python et en langage C produisent des résultats numériquement équivalents, les écarts observés étant limités à la précision machine. Cela confirme que le choix du langage n'influence pas la solution numérique obtenue.

Enfin, les hypothèses du modèle macroscopique, telles que l'homogénéité de la population, constituent une limite de cette approche et motivent l'étude, dans la seconde partie du projet, d'un modèle multi-agent permettant de prendre en compte les interactions individuelles et la stochasticité.

3 Partie 2 – Modèle multi-agent

3.1 Principe général du modèle multi-agent

Contrairement à l'approche macroscopique utilisée dans la première partie du projet, le modèle multi-agent repose sur une description microscopique de la population. Chaque individu est représenté explicitement par un agent autonome, doté de son propre état épidémiologique et évoluant dans un environnement spatial discret.

Dans ce cadre l'état global du système n'est plus décrit par des proportions moyennes de population, mais résulte de l'ensemble des interactions locales entre agents. Chaque agent peut se trouver dans l'un des quatre états du modèle SEIRS : susceptible (S), exposé (E), infectieux (I) ou retiré (R).

L'évolution du système est gouvernée par des règles locales appliquées à chaque agent individuellement en fonction de son état courant, de son environnement immédiat et de paramètres probabilistes imposés par le modèle. Les transitions entre états ne sont plus déterministes, mais dépendent de processus stochastiques qui introduisent une variabilité naturelle dans les dynamiques épidémiques.

Cette approche permet de prendre en compte explicitement l'hétérogénéité des individus et les fluctuations statistiques, qui sont absentes des modèles basés sur des équations différentielles moyennes.

3.2 Environnement spatial et temporel

Le modèle multi-agent évolue dans un environnement spatial discret représenté par une grille bidimensionnelle régulière. Chaque cellule de la grille peut accueillir au plus un agent et l'ensemble des agents est réparti sur cette grille au cours de la simulation.

L'espace est muni de conditions aux limites périodiques ce qui signifie que la grille est considérée comme toroïdale. Ainsi, un agent qui sort par un bord de la grille réapparaît automatiquement sur le bord opposé. Ce choix permet d'éviter les effets de bord et d'assurer une homogénéité spatiale du domaine de simulation.

Le temps est discret et évolue par pas successifs. À chaque pas de temps tous les agents sont susceptibles de se déplacer et de mettre à jour leur état épidémiologique selon les règles définies par le modèle. Les déplacements des agents sont aléatoires et limités à l'environnement spatial local, de manière à simuler des interactions de proximité entre individus.

Cette discrétisation conjointe de l'espace et du temps constitue le cadre général dans lequel s'appliquent les règles de transition d'état et les mécanismes de transmission de l'infection.

3.3 Règles de transition d'état

Dans le modèle multi-agent, chaque agent possède un état épidémiologique individuel appartenant à l'ensemble $\{S, E, I, R\}$. L'évolution de l'état d'un agent au cours du temps est gouvernée par des règles de transition qui reproduisent le cycle épidémiologique du modèle SEIRS.

Un agent susceptible (S) peut devenir exposé (E à la suite d'un contact avec des agents infectieux présents dans son voisinage. Cette transition représente la contamination initiale de l'individu, sans manifestation immédiate de la capacité à infecter d'autres agents.

Un agent exposé (E) devient infectieux (I après une période d'incubation individuelle, exprimée en nombre de pas de temps. Durant cette phase, l'agent n'est pas encore capable de transmettre l'infection.

Un agent infectieux (I passe à l'état retiré (R) après une durée d'infection individuelle, correspondant à la guérison ou à l'isolement de l'individu. Dans cet état, l'agent est considéré comme temporairement immunisé.

Enfin, un agent retiré (R) redevient susceptible (S) après une durée d'immunité finie, modélisant la perte progressive de l'immunité acquise. Cette transition permet la réapparition de vagues épidémiques successives au cours du temps.

Les durées associées aux états E, I et R sont propres à chaque agent et peuvent varier d'un individu à l'autre, introduisant une hétérogénéité temporelle conforme aux hypothèses du modèle multi-agent.

3.4 Modélisation probabiliste de l'infection

La transmission de l'infection dans le modèle multi-agent est décrite à l'aide d'un mécanisme probabiliste reposant sur des interactions locales entre agents. Pour chaque agent susceptible (S), le risque de contamination dépend de la présence d'agents infectieux dans son voisinage spatial immédiat.

Le voisinage considéré est le voisinage de Moore, constitué des huit cellules adjacentes à la cellule occupée par l'agent. À chaque pas de temps, on dénombre le nombre N_I d'agents infectieux présents dans ce voisinage.

La probabilité qu'un agent susceptible devienne exposé au cours d'un pas de temps est alors donnée par la relation :

$$p = 1 - \exp(-0.5\, N_I).$$

Cette expression garantit que la probabilité de contamination est nulle en l'absence de voisins infectieux et qu'elle augmente de manière croissante avec le nombre d'agents infectieux à proximité, tout en restant strictement comprise entre 0 et 1.

La transition $S \to E$ est ensuite réalisée par tirage aléatoire, en comparant la probabilité p à une variable aléatoire uniforme. Ce mécanisme introduit une stochasticité essentielle dans la dynamique du modèle et permet de reproduire des trajectoires épidémiques différentes pour des conditions initiales identiques.

3.5 Asynchronisme et ordre de mise à jour

Le modèle multi-agent repose sur un schéma de mise à jour asynchrone des agents. Contrairement à une mise à jour synchrone, où tous les agents changeraient d'état simultanément, chaque agent est mis à jour individuellement au cours d'un pas de temps.

À chaque pas de temps les agents sont parcourus selon un ordre aléatoire, renouvelé à chaque itération. cet ordre aléatoire évite l'introduction de biais systématiques liés à la position des agents dans la grille ou à un ordre de parcours fixe.

Lorsqu'un agent est mis à jour, son état épidémiologique et sa position spatiale sont immédiatement modifiés,et ces changements sont pris en compte pour les mises à jour des agents suivants au sein du même pas de temps. Ce mécanisme correspond à une mise à jour immédiate, caractéristique des modèles multi-agents asynchrones.

L'asynchronisme et l'ordre de mise à jour aléatoire jouent un rôle essentiel dans la dynamique du modèle, en introduisant une variabilité supplémentaire et en garantissant une représentation plus réaliste des interactions entre individus.

3.6 Implémentation et générateurs aléatoires

Le modèle multi-agent SEIRS a été implémenté dans plusieurs langages de programmation afin de comparer les résultats obtenus et d'évaluer l'influence du langage sur les performances de calcul. Trois implémentations distinctes ont été réalisées au cours du projet : en Python, en C++ et en langage C.

L'implémentation en Python a servi de référence pour le développement et la validation du modèle. Elle a permis de vérifier la cohérence des règles de transition, du voisinage spatial, du schéma de mise à jour asynchrone et de la dynamique globale du système. Une implémentation équivalente a ensuite été réalisée en C++, afin de comparer les résultats obtenus dans un langage compilé tout en conservant une structure de code proche de celle du modèle Python.

Enfin, une implémentation complète en langage C a été développée afin de répondre aux exigences du sujet concernant la réalisation d'un grand nombre de réplications indépendantes. Cette implémentation permet d'effectuer des simulations massives dans un temps de calcul réduit et constitue la base de l'analyse statistique présentée par la suite.

Les trois implémentations respectent strictement les spécifications du sujet : population de $N = 20\,000$ agents, grille toroïdale de taille 300×300, temps discret sur 730 jours, déplacements aléatoires globaux, voisinage de Moore incluant la cellule centrale, probabilité d'infection $p = 1 - \exp(-0.5\, N_I)$, et durées individuelles tirées selon des lois exponentielles.

Chaque langage est basé sur un générateur pseudo-aléatoire adapté à son environnement. L'implémentation Python utilise le générateur `numpy.random.default_rng`, tandis que l'implémentation C++ repose sur le générateur `std::mt19937` (Mersenne Twister). L'implémentation en langage

C utilise un générateur pseudo-aléatoire basé sur la fonction `rand()`, dont la qualité est suffisante dans le cadre du projet compte tenu du nombre élevé de réplications réalisées.

Dans les trois cas, la reproductibilité des expériences est assurée en fixant explicitement la graine du générateur pseudo-aléatoire. Des graines différentes sont utilisées afin de produire des réplications indépendantes du modèle multi-agent.

3.7 Protocole expérimental

Les simulations multi-agents sont réalisées avec des paramètres communs imposés par le sujet. La population initiale est composée de $N = 20\,000$ agents, répartis aléatoirement sur une grille toroïdale de taille 300×300. Les conditions initiales sont fixées à $S(0) = 19\,980$, $E(0) = 0$, $I(0) = 20$ et $R(0) = 0$.

Le temps est discret et la simulation est menée sur une durée totale de 730 jours. Les durées individuelles passées dans les états E, I et R sont tirées une fois au début de la simulation selon des lois exponentielles de moyenne respectivement 3, 7 et 365 jours, puis conservées fixes pour chaque agent.

À chaque pas de temps les agents sont mis à jour de manière asynchrone selon un ordre aléatoire et leurs déplacements sont globaux, la nouvelle cellule étant choisie uniformément sur l'ensemble de la grille.

Conformément au protocole de comparaison entre langages, trois réplications indépendantes ont été réalisées en Python et trois réplications indépendantes en C++, en faisant varier la graine du générateur pseudo-aléatoire. Pour chaque langage, les résultats sont ensuite moyennés afin de réduire l'effet des fluctuations stochastiques inhérentes au modèle multi-agent.

3.8 Résultats des simulations

Les simulations multi-agents produisent des trajectoires temporelles des effectifs $S(t)$, $E(t)$, $I(t)$ et $R(t)$ sur l'ensemble de la période simulée. En raison du caractère stochastique du modèle, les résultats sont présentés sous forme de moyennes calculées sur trois réplications indépendantes.

La Figure 7 montre l'évolution moyenne des populations obtenue à partir des trois réplications Python. La Figure 8 présente les résultats correspondants pour l'implémentation en C++.

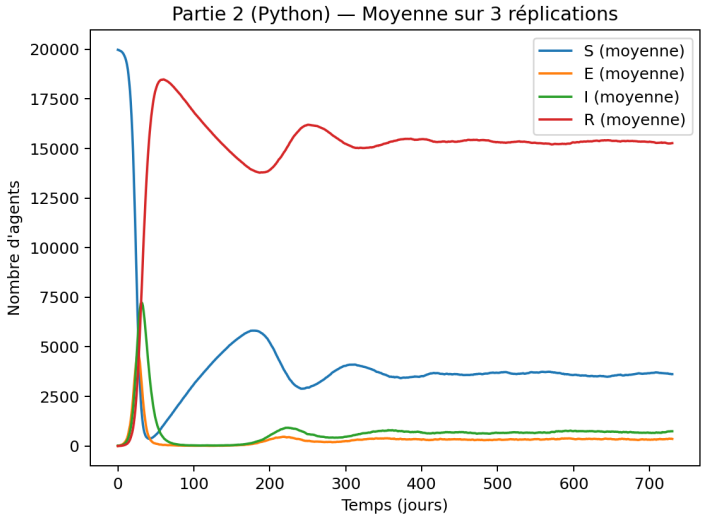

Figure 7: Modèle multi-agent : évolution moyenne des compartiments S, E, I et R obtenue en Python à partir de trois réplications indépendantes.

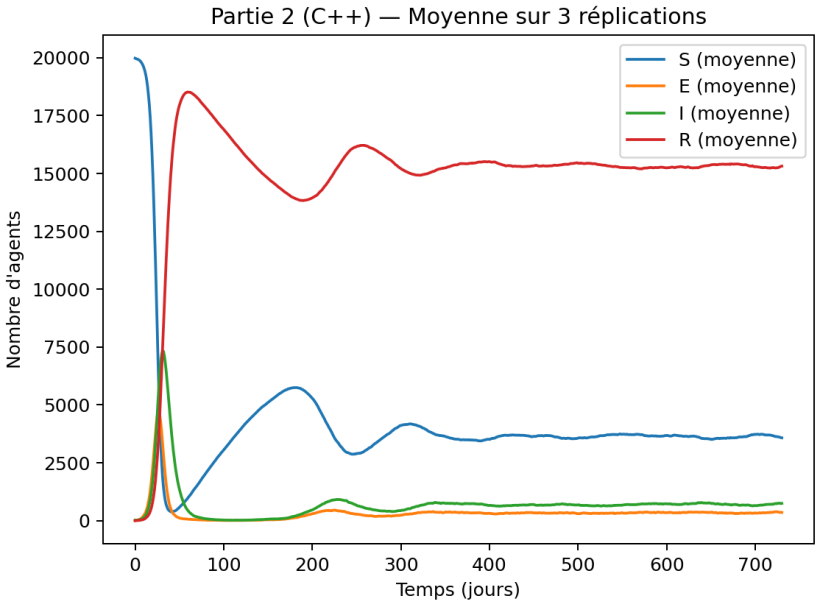

Figure 8: Modèle multi-agent : évolution moyenne des compartiments S, E, I et R obtenue en C++ à partir de trois réplications indépendantes.

Afin de comparer directement les deux langages la Figure 9 superpose les courbes moyennes du nombre d'agents infectieux $I(t)$ obtenues en Python et en C++. Les deux courbes présentent des dynamiques très proches, ce qui indique que les implémentations Python et C++ reproduisent le même comportement épidémique global, à fluctuations stochastiques près.

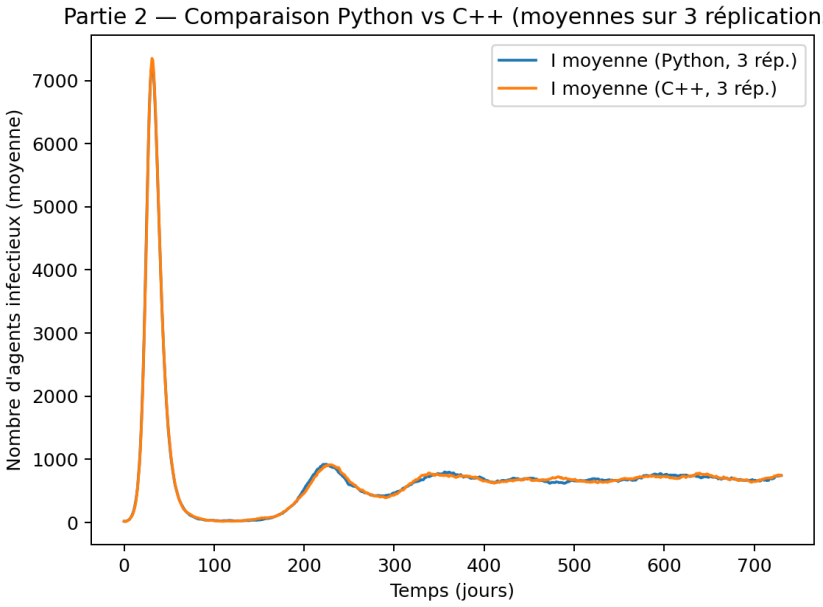

Figure 9: Comparaison Python vs C++ : évolution moyenne du nombre d'agents infectieux $I(t)$ calculée à partir de trois réplications indépendantes.

Cette figure suivante permet de visualiser directement les similitudes et légers décalages temporels entre les implémentations, de manière analogue à l'exemple présenté dans l'énoncé du sujet.

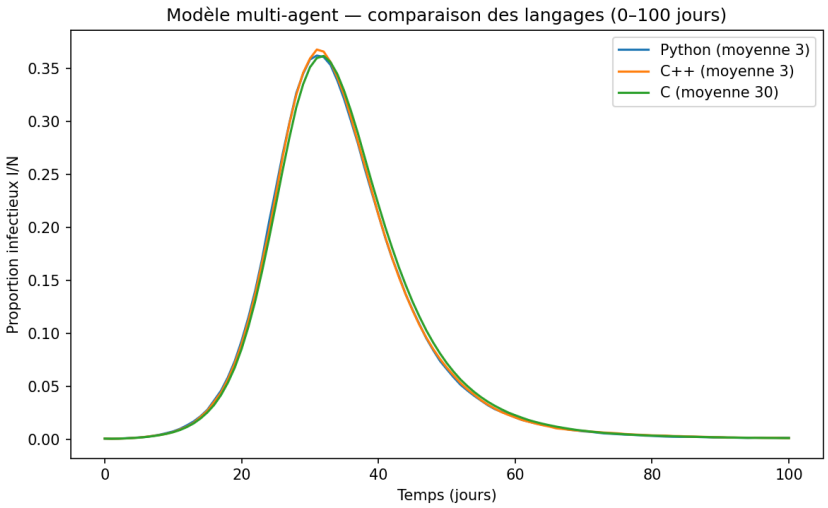

Figure 10: Modèle multi-agent : comparaison de l'évolution du nombre d'agents infectieux normalisé $I(t)/N$ sur les 100 premiers jours pour les implémentations Python (moyenne sur 3 réplications), C++ (moyenne sur 3 réplications) et C (moyenne sur 30 réplications).

3.9 Analyse statistique

En raison du caractère stochastique du modèle multi-agent, une analyse statistique a été menée afin de comparer quantitativement les résultats obtenus avec les différentes implémentations du modèle (Python, C++ et C). L'analyse porte sur deux indicateurs caractéristiques de la dynamique épidémique :

- la hauteur du premier pic infectieux, notée $\max(I)$;
- le jour d'apparition de ce premier pic.

Pour chaque réplication, ces indicateurs ont été extraits automatiquement à partir des fichiers CSV produits par les simulations. Trois réplications ont été réalisées pour les implémentations Python et C++, tandis que trente réplications indépendantes ont été effectuées pour l'implémentation en langage C, conformément aux exigences du sujet.

Les statistiques descriptives (moyenne et écart-type) montrent que les valeurs moyennes du pic infectieux et de sa date d'apparition sont comparables entre les trois langages, malgré une variabilité plus importante observée pour l'implémentation en C, liée au nombre plus élevé de réplications.

Afin d'évaluer formellement l'existence de différences significatives entre les langages, un test non paramétrique de Kruskal–Wallis a été appliqué aux deux indicateurs. Ce test est particulièrement adapté dans ce contexte, les distributions n'étant pas supposées gaussiennes et les tailles d'échantillons étant différentes selon les langages.

Les résultats du test indiquent que les valeurs de p associées à la hauteur du pic infectieux et à la date du pic sont toutes supérieures au seuil de significativité de 5%. Il n'est donc pas possible de rejeter l'hypothèse nulle d'égalité des distributions entre les trois implémentations.

Ces résultats montrent qu'aucune différence statistiquement significative n'est mise en évidence entre les implémentations Python, C++ et C du modèle multi-agent, et que les écarts observés sont principalement dus aux fluctuations stochastiques inhérentes au modèle.

La Figure 11 illustre cette variabilité à l'aide de diagrammes en boîte pour les deux indicateurs étudiés.

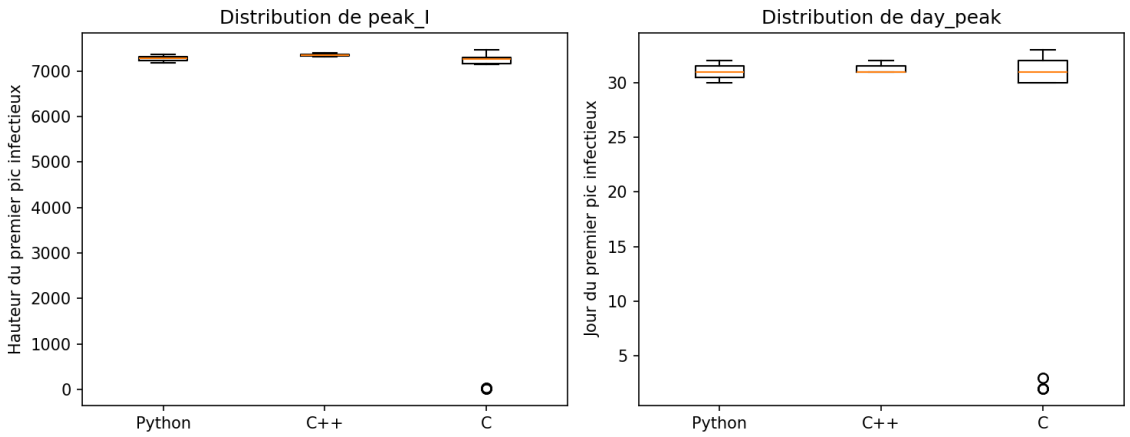

Figure 11: Diagrammes en boîte de la hauteur du premier pic infectieux (gauche) et du jour d'apparition du pic (droite) pour les implémentations Python, C++ et C du modèle multi-agent.

3.10 Discussion

Les résultats obtenus avec le modèle multi-agent mettent en évidence le role central de la stochasticité et des interactions locales dans la dynamique épidémiologique. Contrairement au modèle déterministe étudié dans la première partie du projet, les trajectoires issues du modèle multi-agent présentent une variabilité significative d'une réplication à l'autre meme lorsque les conditions initiales et les paramètres du modèle sont strictement identiques.

L'analyse des trajectoires moyennes montre toutefois que les implémentations en Python, C++ et C conduisent à des dynamiques globales très similaires. Les courbes moyennes des populations $S(t)$, $E(t)$, $I(t)$ et $R(t)$ présentent les mêmes tendances générales, notamment l'apparition rapide d'un premier pic infectieux suivi de phases de décroissance et de résurgence liées à la perte progressive de l'immunité.

L'analyse statistique des indicateurs caractéristiques, en particulier la hauteur et la date du premier pic infectieux, confirme que les différences observées entre les langages ne sont pas statistiquement significatives. Les tests de Kruskal–Wallis indiquent que les écarts mesurés sont compatibles avec des fluctuations stochastiques naturelles du modèle, et non avec des différences liées aux implémentations numériques.

La variabilité plus marquée observée pour l'implémentation en langage C s'explique principalement par le nombre plus élevé de réplications réalisées, conformément aux exigences du sujet. Cette augmentation du nombre de réplications permet une meilleure estimation des distributions des indicateurs, au prix d'un coût de calcul plus important.

Ces résultats soulignent l'intérêt du langage C pour la réalisation de simulations massives et d'analyses statistiques approfondies, tandis que les implémentations Python et C++ offrent une plus grande facilité de développement et de mise au point du modèle. Le modèle multi-agent apparaît ainsi comme un outil puissant pour l'étude des dynamiques épidémiques complexes, à condition de disposer de ressources de calcul suffisantes pour explorer correctement la variabilité statistique du système.

Influence du générateur pseudo-aléatoire

Les implémentations du modèle multi-agent reposent sur des générateurs pseudo-aléatoires différents selon le langage utilisé : PCG64 en Python, Mersenne Twister en C++ et un générateur standard de type LCG en C. Bien que ces générateurs présentent des propriétés statistiques distinctes, aucune différence statistiquement significative n'a été observée sur les indicateurs étudiés (hauteur et date du pic infectieux).

Ces résultats suggèrent que, dans le cadre de ce modèle et pour les indicateurs considérés, le choix du générateur pseudo-aléatoire n'a pas d'influence détectable sur la dynamique épidémique globale, les variations observées étant principalement dues à la stochasticité intrinsèque du modèle.

4 Discussion générale

Les deux approches étudiées dans ce projet, le modèle SEIRS basé sur des équations différentielles ordinaires et le modèle multi-agent, offrent des points de vue complémentaires sur la dynamique épidémique.

Le modèle SEIRS continu, étudié dans la première partie, repose sur une description macroscopique de la population. Il fournit une évolution déterministe des proportions $S(t)$, $E(t)$, $I(t)$ et $R(t)$, gouvernée par un système d'équations différentielles. Cette approche présente l'avantage d'être rapide à simuler, facile à analyser mathématiquement et bien adaptée à l'étude de comportements moyens à grande échelle. Elle permet notamment d'évaluer l'influence des paramètres du modèle et de comparer efficacement différentes méthodes numériques.

En revanche, le modèle multi-agent adopté dans la seconde partie propose une description microscopique de la population, dans laquelle chaque individu est représenté explicitement. Cette approche intègre naturellement l'hétérogénéité des individus, les interactions locales et la stochasticité des processus de contamination. Elle met en évidence des fluctuations statistiques absentes du modèle déterministe, ainsi qu'une variabilité inter-réplications significative, même pour des conditions initiales identiques.

Les résultats obtenus montrent que, bien que les deux approches produisent des dynamiques globales comparables (apparition d'un pic infectieux, phases de déclin et de résurgence), le modèle multi-agent permet d'accéder à des informations supplémentaires, telles que la distribution des pics infectieux ou la dispersion temporelle des événements épidémiques. Ces informations sont essentielles pour une analyse statistique fine et pour l'évaluation de scénarios réalistes.

D'un point de vue numérique et informatique, le modèle ODE est peu coûteux en temps de calcul et se prête bien à des explorations paramétriques rapides. À l'inverse, le modèle multi-agent est nettement plus exigeant en ressources de calcul, en particulier lorsqu'un grand nombre de réplications est nécessaire, mais il bénéficie pleinement de l'utilisation de langages compilés tels que le C et le C++.

Ainsi, les deux approches ne s'opposent pas mais se complètent. Le modèle SEIRS continu constitue un outil efficace pour l'analyse globale et la compréhension des mécanismes moyens, tandis que le modèle multi-agent offre une représentation plus fine et réaliste des dynamiques épidémiques, au prix d'une complexité numérique accrue.

5 Conclusion

Ce projet a permis d'étudier le modèle épidémiologique SEIRS à travers deux approches complémentaires : une modélisation déterministe basée sur des équations différentielles ordinaires et une modélisation stochastique fondée sur un modèle multi-agent.

La première approche a mis en évidence la dynamique moyenne du système et a permis de comparer l'influence des méthodes numériques sur la précision des résultats. Les simulations ont montré que, pour un pas de temps adapté, les méthodes d'Euler explicite et de Runge–Kutta d'ordre 4 conduisent à des dynamiques qualitativement similaires, avec des différences quantitatives liées au schéma numérique utilisé.

La seconde approche a permis d'analyser plus finement la propagation de l'infection en intégrant les interactions locales et la variabilité individuelle. Les résultats obtenus montrent que le langage de programmation n'influence pas la dynamique épidémiologique lorsque les règles

du modèle sont strictement identiques, mais qu'un nombre élevé de réplications est nécessaire pour caractériser correctement la variabilité statistique des trajectoires.

Ces travaux soulignent la complémentarité des modèles continus et multi-agents pour l'étude des phénomènes épidémiques. Alors que les modèles déterministes offrent un cadre simple et efficace pour l'analyse globale, les modèles multi-agents permettent une description plus réaliste des dynamiques au prix d'un coût de calcul plus important. Des perspectives d'extension incluent l'introduction de structures de contact plus complexes, de paramètres dépendants du temps ou de l'espace, ou encore l'étude de stratégies de contrôle épidémique dans un cadre multi-agent.