

Projet AN2 - Modélisation de la propagation d'une épidémie

Matthew Sanchez - Célia Roess - Adam Sekkat

Mars/Avril 2024

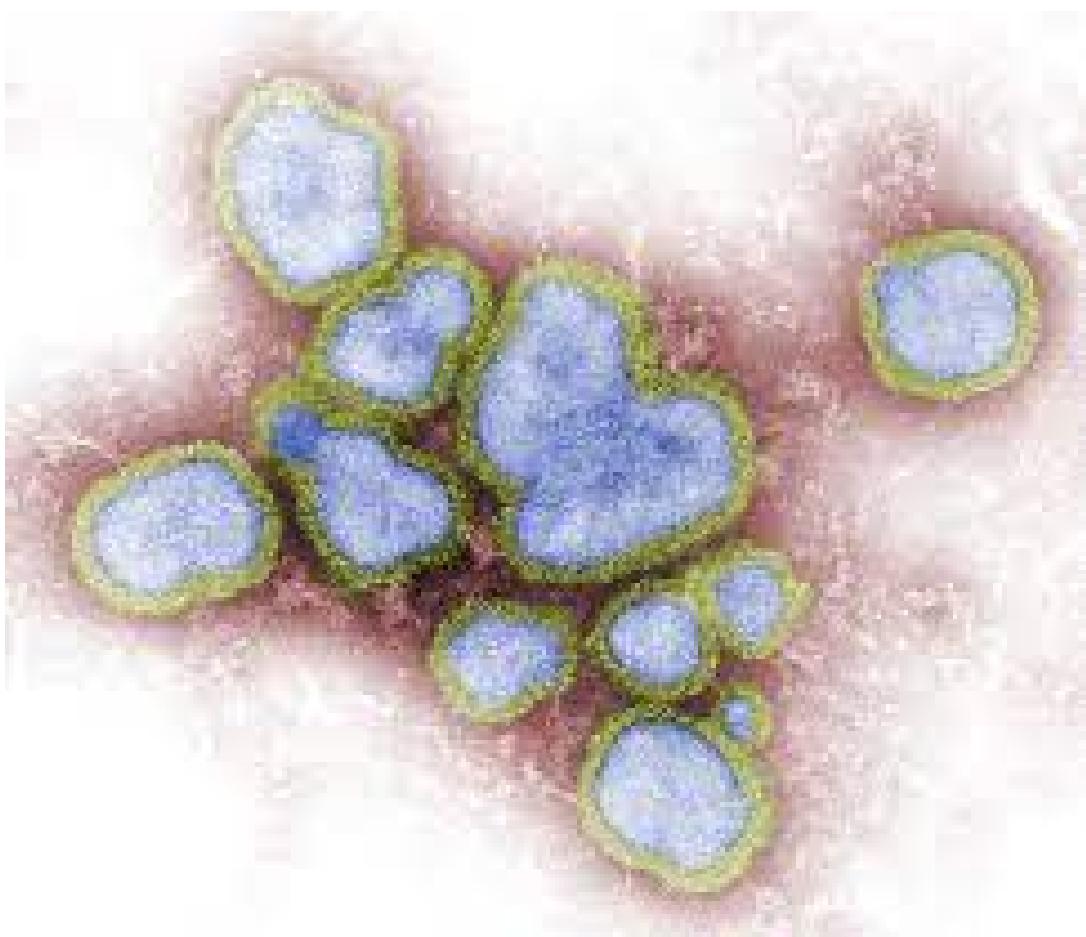


Table des matières

1	Introduction	3
2	Analyse du modèle et explications	4
2.1	La fonction associée à T'	4
2.2	La fonction associée à R'	4
2.3	La fonction associée à S'	4
2.4	La fonction associée à I'	5
2.5	Système final d'équations différentielles	5
3	Méthodes de résolution numérique utilisées	6
3.1	Euler explicite	6
3.2	Euler implicite	6
3.3	Runge-Kutta d'ordre 4	7
3.4	Adams-Bashforth	7
4	Résultats obtenus	8
5	Conclusion	9
	Annexe n°1 : Plan de travail	10
	Annexe n°2 : Carnet de bord	11

1 Introduction

La diffusion d'un agent infectieux au sein d'une population est un processus dynamique.

Les chiffres des individus en bonne santé et des malades évoluent dans le temps en fonction des interactions où l'agent infectieux est transmis d'un individu infecté à un individu sain non immunisé, qui devient alors infecté à son tour. Cette propagation infectieuse peut être représentée mathématiquement par des équations différentielles.

Ce projet vise à résoudre le système d'équations différentielles associées à l'aide de méthodes de résolution numérique, puis à visualiser graphiquement les résultats.

Il est important de mettre en relief l'importance du paramètre R_0 qui décrit le nombre moyen de nouvelles infections dues à un individu malade. Il joue un rôle crucial dans la lutte contre une épidémie.

En effet :

- si $R_0 < 1$, l'épidémie tend à disparaître
- si $R_0 > 1$, l'épidémie tend à se propager

On utilise pour ce faire le **modèle SIR** qui correspond à un des modèles compartimentaux les plus simples.

On a :

- S (Susceptibles) : le nombre de sujets susceptibles d'être infectés
- I (Infectieux) : le nombre de personnes infectieuses
- R (Rétablis) : le nombre de personnes ne pouvant plus contracter la maladie

Afin d'avoir un modèle plus précis, nous avons introduit :

- T (Traités) : le nombre de personnes qui ont reçu un traitement

Ce modèle est prédictif pour les maladies infectieuses se transmettant d'homme à homme et pour lesquelles le rétablissement confère une résistance durable face à la même maladie. On peut citer comme exemple de maladies qui valident ces conditions : la rougeole, les oreillons, la rubéole, etc...

On introduit également plusieurs paramètres pour le modèle.

- α : fraction d'individus infectés sélectionnés pour être traitée par unité de temps;
- β : nombre moyen d'individus rencontrés par un individu par unité de temps;
- γ : le taux de guérison;
- η : taux d'individus traités qui deviennent immunisés ou décédés;
- δ : facteur de réduction de l'infectivité grâce à un traitement.

2 Analyse du modèle et explications

On va, dans cette section, chercher à expliquer et valider le modèle proposé dans le sujet grâce aux hypothèses données préalablement.

2.1 La fonction associée à T'

On cherche ici à modéliser l'évolution des personnes traitées.

On ajoute les individus infectés sélectionnés pour traitement.

$$\alpha * I$$

On soustrait les personnes traitées qui deviennent immunisées.

$$-\eta * T$$

La dérivée est initialement positive car il n'y a aucun individu T (traité) au début, puis devient négative lorsque le nombre d'individus infectés est assez bas.

Donc, $T_0 = 0$. Ensuite, T augmente puis diminue.

2.2 La fonction associée à R'

On cherche ici à modéliser l'évolution des personnes rétablies.

On ajoute les individus qui guérissent naturellement.

$$\gamma * I$$

On ajoute les individus qui guérissent grâce au traitement.

$$\eta * T$$

On va regrouper ces 2 groupes dans la catégorie des rétablis, car ces personnes ne peuvent plus attraper la maladie.

La dérivée est toujours positive, donc si on suppose qu'il y a des individus infectués au départ.

Donc, $R_0 = 0$. Ensuite, R augmente jusqu'à atteindre la valeur N (avec $N = S+I+R$).

On remarque qu'il est important de ne pas le confondre avec le paramètre R0.

2.3 La fonction associée à S'

On cherche ici à modéliser l'évolution des personnes saines.

On remarque que :

$$\frac{\beta}{N} \leq 1$$

On a le nombre de personnes infectées rencontrées en moyenne par unité de temps

$$\frac{\beta}{N} * I$$

On a également le nombre de personnes traitées mais toujours contagieuses rencontrées en moyenne par unité de temps (l'infectivité est réduite par le traitement)

$$\frac{\beta}{N} * \delta * T$$

S' est négative, donc S diminue dans le temps, en effet, les personnes susceptibles à la maladie deviennent graduellement des Infectés, des Rétablis ou des Traités.

Plus on avance dans le temps, moins S diminue (avec un facteur $\frac{\beta}{N}$) car les personnes Susceptibles vont rencontrer moins d'Infectés et de $\delta * T$ qui correspondent aux personnes traitées mais toujours contagieuses.

2.4 La fonction associée à I'

On cherche ici à modéliser l'évolution des personnes infectées.

Ici, le nombre de personnes susceptibles rencontrées en moyenne par unité de temps est modélisée par le terme suivant :

$$\frac{\beta}{N} * S$$

Afin d'avoir le nombre de personnes susceptibles rencontrées en moyenne par unité de temps par les personnes infectées et les personnes traitées qui restent contagieuses, on multiplie le facteur précédent par :

$$I + \delta * T$$

On soustrait aux personnes infectées ceux qui ont été sélectionnés pour un traitement dont le taux est égal à α ou ceux qui ont guéri naturellement dont le taux est égal à γ

I' reste positive tant que les infectés rencontrent assez de personnes susceptibles à la maladie.

A un stade plus avancé de l'épidémie, lorsqu'il n'y aura plus assez de S , lorsqu'on sélectionne plus de personnes pour un traitement, ou lorsque plus de personnes vont guérir grâce à l'immunité collective, alors I' deviendra négative, et I tendra vers 0.

2.5 Système final d'équations différentielles

Finalement, on obtient :

$$\begin{cases} S' = -\frac{\beta}{N}(I + \delta T)S \\ I' = \frac{\beta}{N}S(I + \delta T) - (\alpha + \gamma)I \\ T' = \alpha I - \eta T \\ R' = \gamma I + \eta T \end{cases}$$

On définit alors les fonctions Python associées à S' , I' , T' et R' .

```
def phiS(S,I,T,beta,delta,N):
    return (-beta/N) * (I + delta*T) * S

def phiI(S,I,T,alpha,beta,delta,gamma,N):
    return (beta/N) * S * (I + delta*T) - (alpha + gamma) * I

def phiT(I,T,alpha,eta):
    return alpha*I - eta*T

def phiR(I,T,gamma,eta):
    return gamma*I + eta*T
```

3 Méthodes de résolution numérique utilisées

On part du système d'équations différentielles suivant :

$$\begin{cases} S' = -\frac{\beta}{N}(I + \delta T)S \\ I' = \frac{\beta}{N}S(I + \delta T) - (\alpha + \gamma)I \\ T' = \alpha I - \eta T \\ R' = \gamma I + \eta T \end{cases} \quad (1)$$

3.1 Euler explicite

En appliquant la **méthode d'Euler Explicite** sur le système (1), on obtient :

$$\begin{cases} S_{n+1} = S_n + h \times \left(-\frac{\beta}{N}(I_n + \delta T_n)S_n\right) \\ I_{n+1} = I_n + h \times \left(\frac{\beta}{N}S_n(I_n + \delta T_n) - (\alpha + \gamma)I_n\right) \\ T_{n+1} = T_n + h \times (\alpha I_n - \eta T_n) \\ R_{n+1} = R_n + h \times (\gamma I_n + \eta T_n) \end{cases}$$

3.2 Euler implicite

En appliquant la **méthode d'Euler Implicit** sur le système (1), on obtient :

$$\begin{cases} S_{n+1} = S_n - h \frac{\beta}{N}(I_{n+1} + \delta T_{n+1})S_{n+1} \\ I_{n+1} = I_n + h \frac{\beta}{N}S_{n+1}(I_{n+1} + \delta T_{n+1}) - h(\alpha + \gamma)I_{n+1} \\ T_{n+1} = T_n + h\alpha I_{n+1} - h\eta T_{n+1} \\ R_{n+1} = R_n + h\gamma I_{n+1} + h\eta T_{n+1} \end{cases}$$

On fait passer le membre de droite de l'égalité vers la gauche pour obtenir le système $F(S_{n+1}, I_{n+1}, T_{n+1}, R_{n+1}) = 0$ suivant :

$$\begin{cases} S_{n+1} + h \frac{\beta}{N}(I_{n+1} + \delta T_{n+1})S_{n+1} - S_n = 0 \\ I_{n+1} + h(\alpha + \gamma)I_{n+1} - h \frac{\beta}{N}S_{n+1}I_{n+1} - h \frac{\beta}{N}\delta T_{n+1} - I_n = 0 \\ T_{n+1} - h\alpha I_{n+1} + h\eta T_{n+1} - T_n = 0 \\ R_{n+1} - h\gamma I_{n+1} - h\eta T_{n+1} - R_n = 0 \end{cases}$$

On dérive chaque ligne par $S_{n+1}, I_{n+1}, T_{n+1}, R_{n+1}$, on obtient alors la matrice jacobienne J suivante :

$$\begin{pmatrix} 1 + h \frac{\beta}{N}(I_{n+1} + \delta T_{n+1}) & h \frac{\beta}{N}S_{n+1} & h \frac{\beta}{N}\delta S_{n+1} & 0 \\ -h \frac{\beta}{N}I_{n+1} & 1 + h(\alpha + \gamma) - h \frac{\beta}{N}S_{n+1} & -h \frac{\beta}{N}\delta S_{n+1} & 0 \\ 0 & -h\alpha & 1 + h\eta & 0 \\ 0 & -h\gamma & -h\eta & 1 \end{pmatrix}$$

Pour obtenir les $S_{n+1}, I_{n+1}, T_{n+1}, R_{n+1}$, on applique à chaque pas de temps une itération de Newton :

$$\begin{pmatrix} S_{n+1} \\ I_{n+1} \\ T_{n+1} \\ R_{n+1} \end{pmatrix} = \begin{pmatrix} S_n \\ I_n \\ T_n \\ R_n \end{pmatrix} - F(S_{n+1}, I_{n+1}, T_{n+1}, R_{n+1}) = J^{-1}(S_n, I_n, T_n, R_n)$$

3.3 Runge-Kutta d'ordre 4

En appliquant la **méthode de Runge-Kutta d'ordre 4**, on trouve :

$$\begin{cases} S_{n+1} = S_n + \frac{h}{6}(K_{1,S} + 2K_{2,S} + 2K_{3,S} + K_{4,S}) \\ I_{n+1} = I_n + \frac{h}{6}(K_{1,I} + 2K_{2,I} + 2K_{3,I} + K_{4,I}) \\ T_{n+1} = T_n + \frac{h}{6}(K_{1,T} + 2K_{2,T} + 2K_{3,T} + K_{4,T}) \\ R_{n+1} = R_n + \frac{h}{6}(K_{1,R} + 2K_{2,R} + 2K_{3,R} + K_{4,R}) \end{cases}$$

avec :

$$\begin{cases} K_{1,S} = -\frac{\beta}{N}(I_n + \delta T_n)S_n \\ K_{1,I} = \frac{\beta}{N}S_n(I_n + \delta I_n) - (\alpha + \gamma)I_n \\ K_{1,T} = \alpha I_n - \eta T_n \\ K_{1,R} = \gamma I_n + \eta T_n \\ K_{2,S} = -\frac{\beta}{N}((I_n + \frac{h}{2}K_{1,I}) + \delta(T_n + \frac{h}{2}K_{1,T}))\left(S_n + \frac{h}{2}K_{1,S}\right) \\ K_{2,I} = \frac{\beta}{N}((I_n + \frac{h}{2}K_{1,I}) + \delta(T_n + \frac{h}{2}K_{1,T})) - (\alpha + \gamma)(I_n + \frac{h}{2}K_{1,I}) \\ K_{2,T} = \alpha(I_n + \frac{h}{2}K_{1,I}) - \eta(T_n + \frac{h}{2}K_{1,T}) \\ K_{2,R} = \gamma(I_n + \frac{h}{2}K_{1,I}) + \eta(T_n + \frac{h}{2}K_{1,T}) \\ K_{3,S} = -\frac{\beta}{N}((I_n + \frac{h}{2}K_{2,I}) + \delta(T_n + \frac{h}{2}K_{2,T}))\left(S_n + \frac{h}{2}K_{2,S}\right) \\ K_{3,I} = \frac{\beta}{N}((I_n + \frac{h}{2}K_{2,I}) + \delta(T_n + \frac{h}{2}K_{2,T})) - (\alpha + \gamma)(I_n + \frac{h}{2}K_{2,I}) \\ K_{3,T} = \alpha(I_n + \frac{h}{2}K_{2,I}) - \eta(T_n + \frac{h}{2}K_{2,T}) \\ K_{3,R} = \gamma(I_n + \frac{h}{2}K_{2,I}) + \eta(T_n + \frac{h}{2}K_{2,T}) \\ K_{4,S} = -\frac{\beta}{N}((I_n + \frac{h}{2}K_{3,I}) + \delta(T_n + \frac{h}{2}K_{3,T}))\left(S_n + \frac{h}{2}K_{3,S}\right) \\ K_{4,I} = \frac{\beta}{N}((I_n + \frac{h}{2}K_{3,I}) + \delta(T_n + \frac{h}{2}K_{3,T})) - (\alpha + \gamma)(I_n + \frac{h}{2}K_{3,I}) \\ K_{4,T} = \alpha(I_n + \frac{h}{2}K_{3,I}) - \eta(T_n + \frac{h}{2}K_{3,T}) \\ K_{4,R} = \gamma(I_n + \frac{h}{2}K_{3,I}) + \eta(T_n + \frac{h}{2}K_{3,T}) \end{cases}$$

3.4 Adams-Bashforth

En appliquant la **méthode d'Adams-Bashforth d'ordre 2**, on obtient :

$$\begin{cases} S_{n+1} = S_n + \frac{h}{2}(3(-\frac{\beta}{N}(I_n + \delta T_n)S_n) - (-\frac{\beta}{N}(I_{n+1} + \delta T_{n+1})S_{n+1})) \\ I_{n+1} = I_n + \frac{h}{2}(3(\frac{\beta}{N}S_n(I_n + \delta T_n) - (\alpha + \gamma)I_n) - (\frac{\beta}{N}S_{n+1}(I_{n+1} + \delta T_{n+1}) - (\alpha + \gamma)I_{n+1})) \\ T_{n+1} = T_n + \frac{h}{2}(3(\alpha I_n - h\eta T_n) - (\alpha I_{n+1} - h\eta T_{n+1})) \\ R_{n+1} = R_n + \frac{h}{2}(3(\gamma I_n + h\eta T_n) - (\gamma I_{n+1} + h\eta T_{n+1})) \end{cases}$$

4 Résultats obtenus

Afin que l'utilisateur puisse facilement choisir la méthode de résolution qu'il souhaite, nous avons créé une interface Tkinter où les boutons donnent accès aux graphiques de chaque méthode.

⚠ Le programme MAIN.PY qui lance l'interface Tkinter doit être exécuté avec Visual Studio Code pour pouvoir utiliser les sliders.

Nous avons également fait le choix de rajouter un bouton d'aide qui rappelle à l'utilisateur à quoi correspond chaque paramètre : alpha, beta, gamma, eta, delta.



FIGURE 1 – Interface graphique Tkinter

Toujours dans l'optique de rendre l'utilisation de notre interface simple et efficace, nous avons utilisé des sliders qui permettent de gérer en temps réel les paramètres de nos modèles.

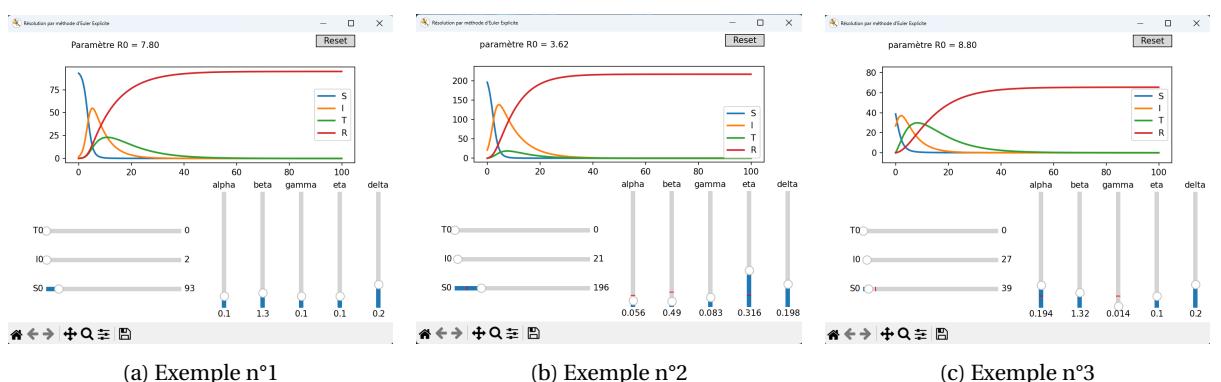


FIGURE 2 – Résultats obtenus avec la méthode d'Euler Explicite

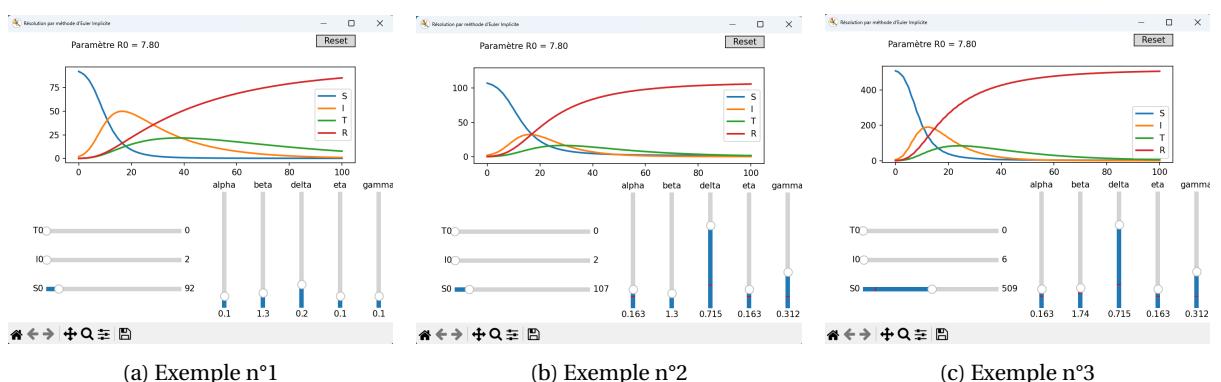


FIGURE 3 – Résultats obtenus avec la méthode d'Euler Implicite

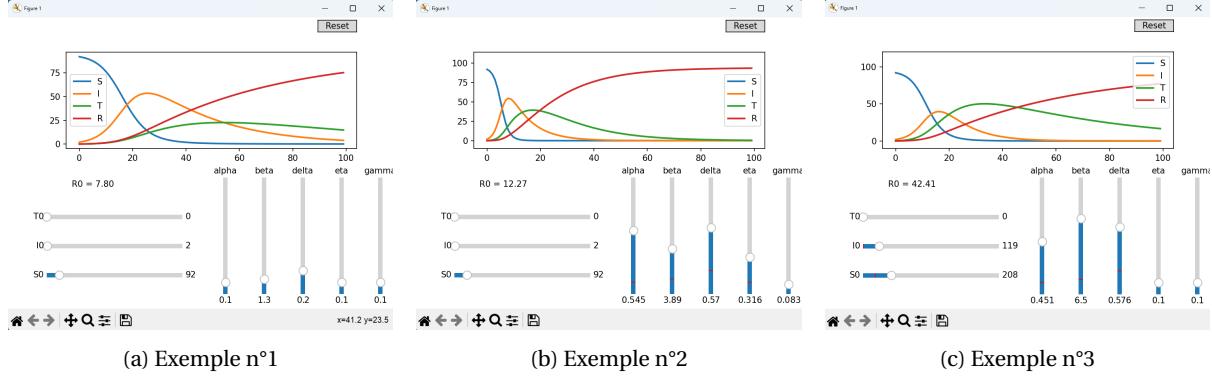


FIGURE 4 – Résultats obtenus avec la méthode de Runge-Kutta 4

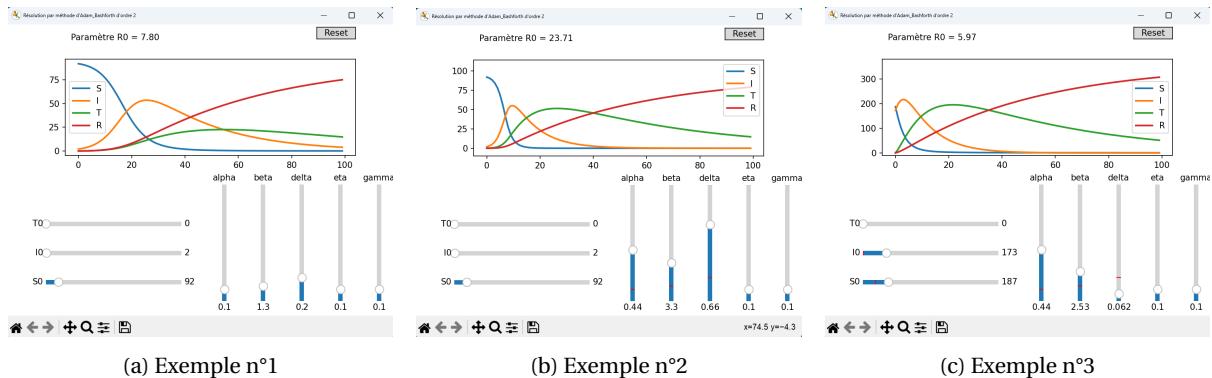


FIGURE 5 – Résultats obtenus avec la méthode d'Adams-Bashforth 2

On remarque que le système d'équations différentielles utilisé renvoie un modèle SIRT complet. Les résultats obtenus sont proches de ceux que l'on peut retrouver avec d'autres maladies déjà étudiées. Les courbes renvoyées convergent vers une solution stable.

Dans le sens physique, on peut dire que les résultats que l'on a sont cohérents avec la réalité. En effet, nous avons un pic d'infection au début de l'épidémie, le nombre de personnes saines et infectées tend vers 0 et le nombre de personnes rétablies tend vers N (la population totale de notre environnement).

5 Conclusion

Pour conclure, ce projet nous a permis de mettre en pratique différentes méthodes pour résoudre numériquement un système d'équations différentielles. Notre cas s'intéresse à la modélisation de la propagation d'une épidémie. Pour ce faire, nous avons pu utiliser les méthodes d'Euler explicite, Euler implicite, Runge-Kutta d'ordre 4 et Adams-Bashforth d'ordre 2.

Après avoir initialisé les constantes et les paramètres du modèle, nous avons défini les différentes fonctions correspondant à l'évolution en fonction du temps de chaque compartiment.

Grâce à une analyse des résultats obtenus, il est possible de dire que le modèle SIRT est un modèle plutôt complet. Cependant, il présente quelques limites :

- la non-prise en compte de l'hétérogénéité de la population (âge, système immunitaire...);
- l'absence de compartiments supplémentaires (vaccinés, en quarantaine...);
- les taux de transmission et de rétablissement sont supposés constants.

Annexe n°1 : Plan de travail

Titre et description du projet

Modélisation de la propagation d'une épidémie

Buts et objectifs du projet

L'objectif de ce projet est d'élaborer un code Python visant à représenter graphiquement l'évolution de la propagation d'une épidémie à l'aide d'un modèle résolu par plusieurs méthodes de résolution numérique.

On utilise les méthodes de résolution numérique suivantes :

Euler Explicite / Euler Implicite / Runge-Kutta d'ordre 4 / Adams-Bashforth

Difficultés rencontrées lors du projet

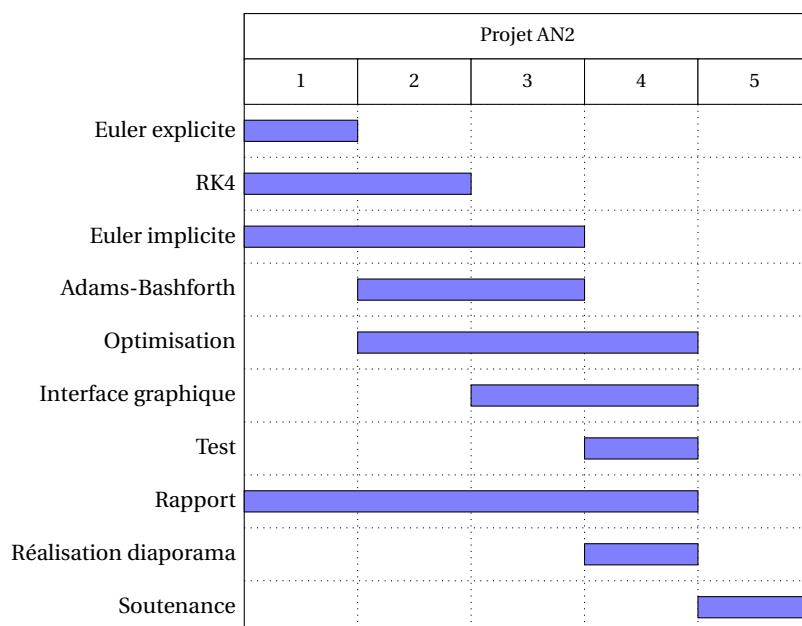
La méthode d'Euler Implicite a été complexe à optimiser. Le code initial mettait 3 secondes avant de renvoyer le nouveau graphique.

Une erreur que l'on avait rencontré au début du projet était la non-distinction entre le paramètre R0 qui correspond au nombre moyen de nouvelles infections dues à un individu malade et la valeur R0 qui est la première valeur de la liste des personnes Rétablies.

Après la réalisation de notre interface graphique avec Tkinter, on a remarqué des interférences sur nos graphiques de nos codes. Il a fallu chercher l'erreur et recoder chaque méthode.

Calendrier et échéancier du projet

Date de fin de projet : 09/04/2024



Annexe n°2 : Carnet de bord

Travail réalisé par Adam

- Rédaction des premières versions de chaque méthode

Travail réalisé par Célia

- Modification des codes pour qu'ils soient plus lisibles
- Affichage du paramètre R_0 sur l'interface graphique
- Modification des limites de l'axe y pour que les courbes ne sortent jamais du graphique
- Rédaction du rapport en \LaTeX
- Séparation du paramètre R_0 et la 1ere valeur de la population Rétablies
- Création de la fonction `MAIN.PY` pour permettre à l'utilisateur de choisir la méthode de résolution qu'il souhaite à travers une interface Tkinter

Travail réalisé par Matthew

- Analyse et compréhension du modèle
- Travail sur la méthode d'Euler implicite
- Modification de la forme du code d'Euler implicite