

```
solve_ivp(fun=deriv, t_span=[0, sim_time], t_eval=t, method='RK45', rtol=1e-05, atol=1e-08, R = ret.y)

des courbes d'évolution avec leurs labels
= plt.plot(S, label="Susceptible")
= plt.plot(I, label="Infected")
= plt.plot(R, label="Recovered with immunity")
= plt.plot(E, label="Exposed")
= plt.plot(N, label="Population")

plt.subplots_adjust(left=0.1, right=0.9, bottom=0.1, top=0.9)
plt.xlabel('Time (days)')
plt.legend()

# Create a horizontal slider to control alpha
a_slider = Slider(
    ax = plt.axes([0.1, 0.25, 0.8, 0.05]), facecolor="lightgoldenrodyellow"),
    label='alpha (Incubation)', valmin=0,
    valmax=i,
    valinit=init_alpha,
    color="green")
```

PROJET MODÉLISATION

MODULE M3202C - MODÉLISATIONS MATHÉMATIQUES

BILLY MAXIME // JUBERT KILLIAN // MARGAILLAN KYLIAN // PINEL ANTHONY

Sommaire

- 01** Introduction au projet
- 02** Recherches sur le sujet
- 03** Modélisation de l'épidémie
 - Calcul des équations différentielles*
 - Modèles SIR et SEIR*
- 10** Génération de l'épidémie
 - Les fonctions utilisées*
 - Implémentation et simulation en code*
- 14** Analyse des résultats
 - Impact des différents paramètres*
 - Conjecture globale*
- 17** "Pour aller plus loin"
 - Modèles SEIGV et SEIGVM*
 - Observations des rendus*
 - Simulation 2D*
- 23** Conclusion
- 24** Bibliographie

MODULE M3202C - MODÉLISATIONS MATHÉMATIQUES

BILLY MAXIME // JUBERT KILLIAN // MARGAILLAN KYLIAN // PINEL ANTHONY

MODÉLISER & ANALYSER LA PROPAGATION D'UNE ÉPIDÉMIE SUR UNE POPULATION

La crise sanitaire actuelle a permis de faire comprendre à tous l'utilité des prédictions épidémiologiques.

La modélisation du comportement d'un virus dans une population donnée étant nécessaire pour se préparer et s'adapter, nous avons trouvé pertinent de s'interroger sur cette évolution.

Pour ce qui est de la démonstration, nous avons choisi des graphiques complets avec des sliders pour modifier en temps réel les valeurs.

Afin de compléter notre analyse, nous avons également mis en place une simulation 2D où des points s'infectent lorsqu'ils se touchent.

En clair, l'objectif est de modéliser et d'analyser la propagation d'une épidémie ou d'un virus en fonction de différents paramètres.

La dynamique des populations

CRÉER DES GRAPHIQUES UTILISANT 2 MODÈLES DE MODÉLISATION D'ÉPIDÉMIE

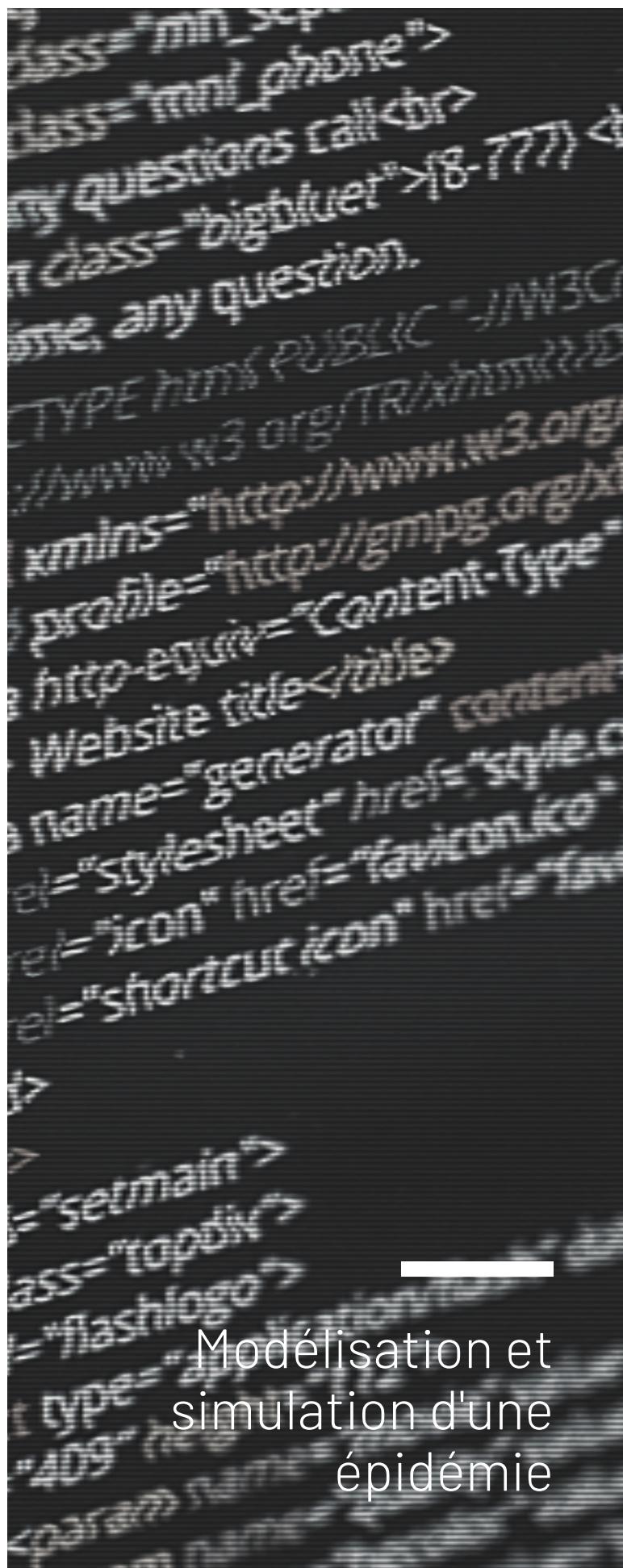
Nous savions déjà que l'utilisation du langage Python était obligatoire pour la réalisation, l'implémentation ainsi que la modélisation de notre projet.

Suite à cela, nous nous sommes penchés et questionnés sur la partie des mathématiques et les différentes méthodes disponibles pour réaliser nos graphiques et nos simulations.

Nous avons trouvé la réponse dans les deux modèles à compartiments : SIR et SEIR.

Ces modèles utilisent des équations différentielles avec de nombreux paramètres naturels pour modéliser des graphiques.

Nous avons ensuite utilisé nos données pour simuler les modèles par le biais d'un rendu interactif 2D.



Modélisation et simulation d'une épidémie

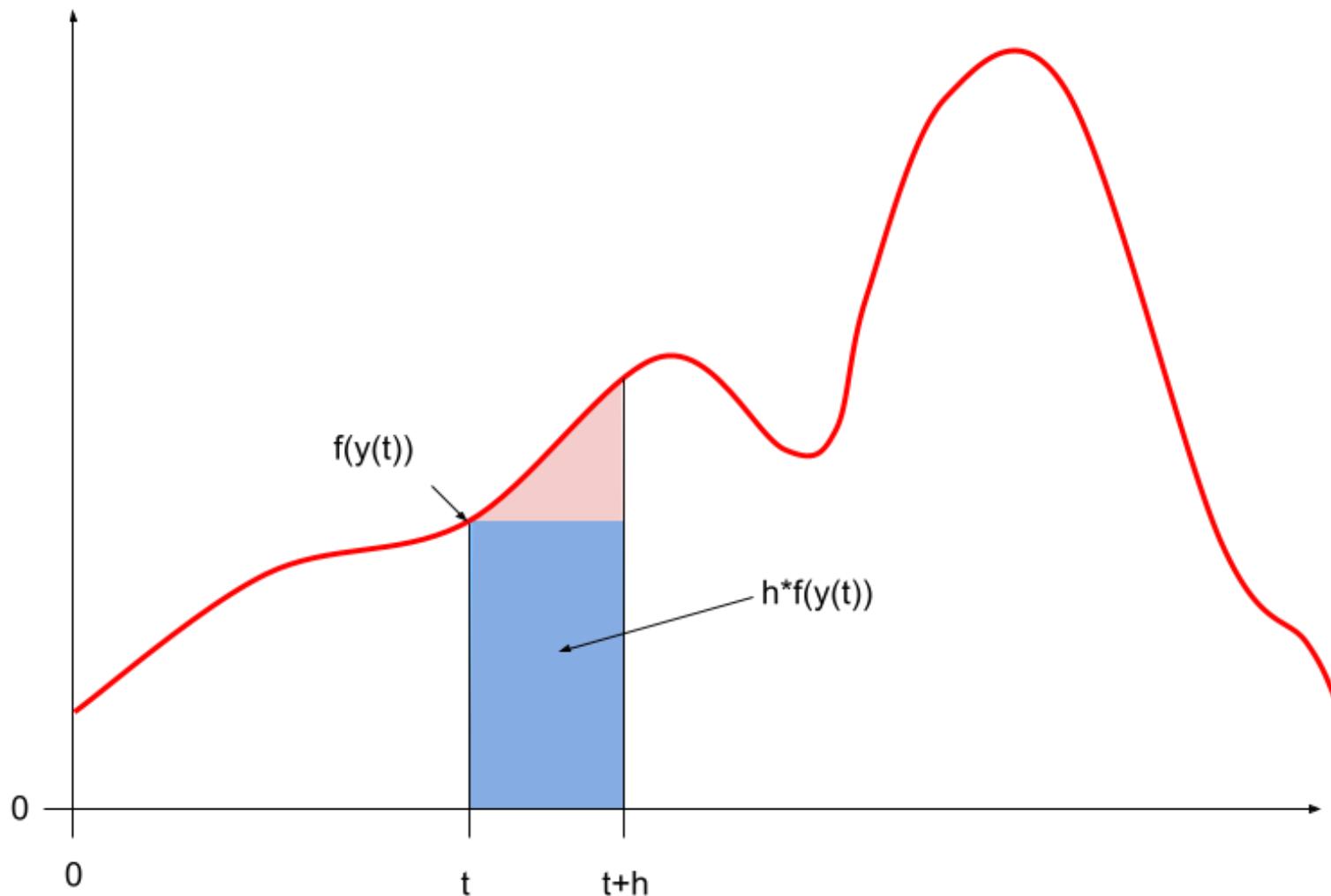
Pour réaliser ce projet, nous avions besoin de modéliser une population globale composée de sous-populations ayant elles-mêmes leur propre fonction et paramètres. C'est pour cette raison que nous avons utilisé des équations différentielles.

En mathématiques, ces équations comportent une ou des inconnues qui sont des fonctions. Elles se présentent sous la forme d'une relation entre ces fonctions inconnues et leurs dérivées successives. C'est un cas particulier d'équation fonctionnelle.

$$\begin{aligned}y'(t) &= f(y(t)) \\ \int_t^{t+h} y'(s) ds &= \int_t^{t+h} f(y(s)) ds \\ y(t + h) - y(t) &= \int_t^{t+h} f(y(s)) ds \\ y(t + h) &= y(t) + \int_t^{t+h} f(y(s)) ds\end{aligned}$$

Comme nous ne pouvons pas calculer de manière triviale l'intégrale de t à $t+h$ de $f(y(s))$ ds nous faisons une approximation en prenant un rectangle de largeur h et de hauteur $f(y(t))$.

L'aperçu de cette explication via un graphique est à la page suivante.



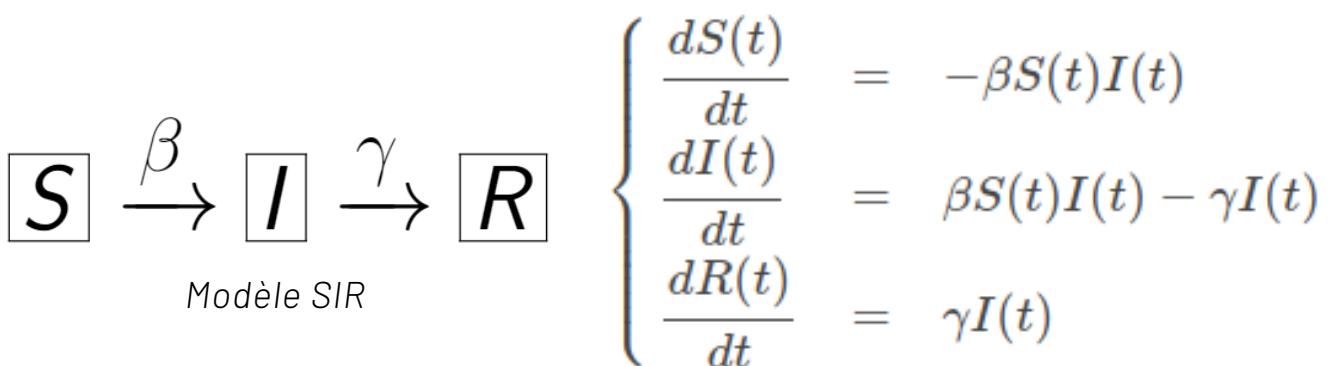
On a donc finalement:

$$y(t + h) = y(t) + h * f(y(t))$$

Maintenant que nous avons vu comment calculer des équations différentielles, comment appliquer ceci à des modèles contenant des populations ?

Les principaux modèles mathématiques traitant de maladies infectieuses ont commencé à être mis en pratique avec dans les années 1980.

On utilise désormais des modèles à compartiments pour faciliter les calculs de probabilité et produire des graphiques sur des populations d'individus. Ces modèles sont fréquemment utilisés depuis l'apparition de la grippe espagnole. La pandémie Covid 19 a permis aux modélisations mathématiques de connaître un essor lors de la prise de décision relatives aux politiques de santé publique.



Le modèle de simulation épidémique le plus simple est le modèle SIR datant de 1927, ce dernier est un modèle à compartiments, c'est-à-dire que l'on divise la population en plusieurs catégories sur un temps t . La population totale sera représentée par N qui est simplement la somme de ces 3 populations au cours du temps.

Il utilise 3 équations qui représentent l'évolution en fonction du temps sur trois populations : $S(t)$ pour saine, $I(t)$ pour infectée et $R(t)$ pour retirée.

Ce modèle prend en compte 2 paramètres :

- β qui représente le taux d'infection.
- γ qui représente le taux de guérison.

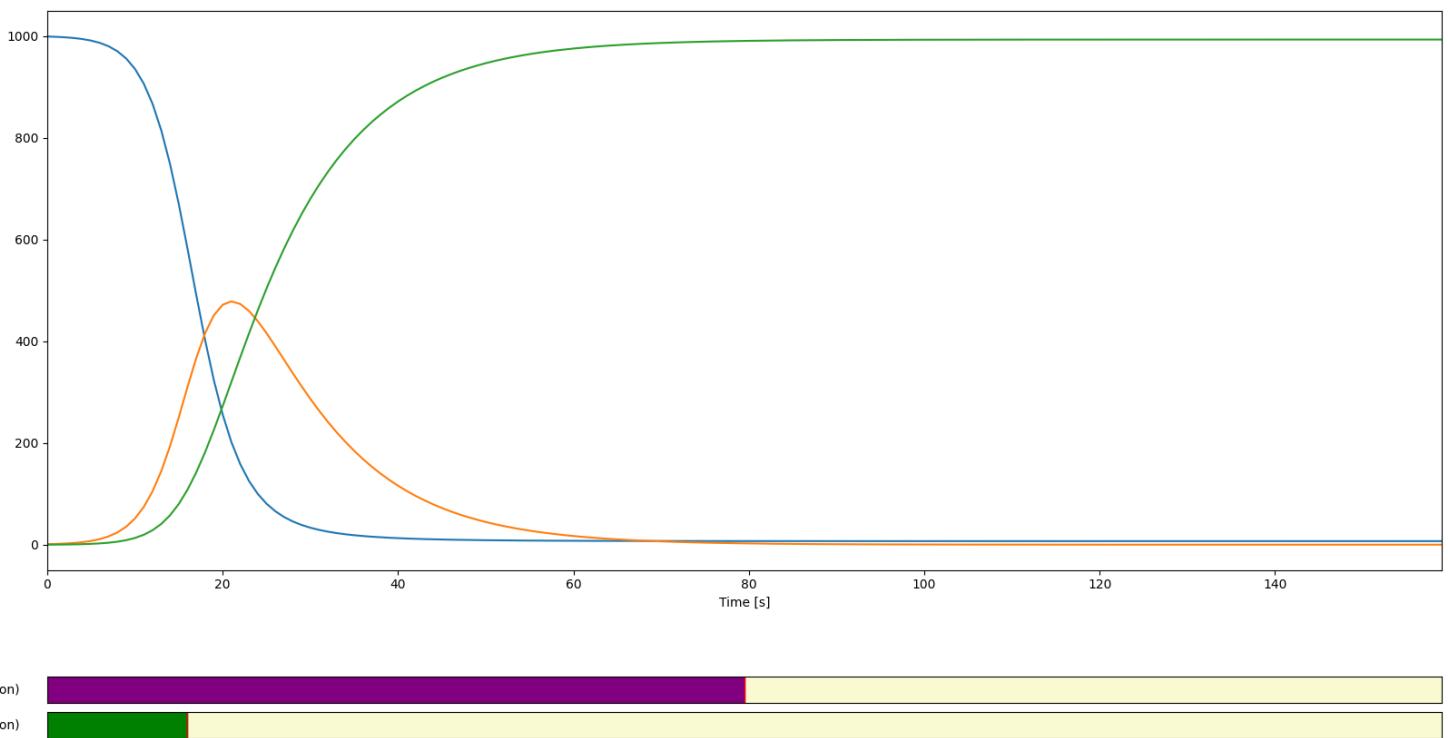
Les personnes saines n'ont pas encore été contaminé par le virus, alors que les personnes retirées sont guéries et immunisées. Autrement dit, le grand problème de ce modèle est que les personnes retirées ne sont plus prises en compte à la fin. Par conséquent, le modèle SIR ne s'occupe pas directement de prédire la mortalité de l'épidémie. Pour résoudre cet inconvénient, il faudrait utiliser des modèles plus complexes. Une alternative partielle est le modèle SEIR qui prend en compte la natalité et la mortalité naturelle, qui permet de cycler la population en continu.

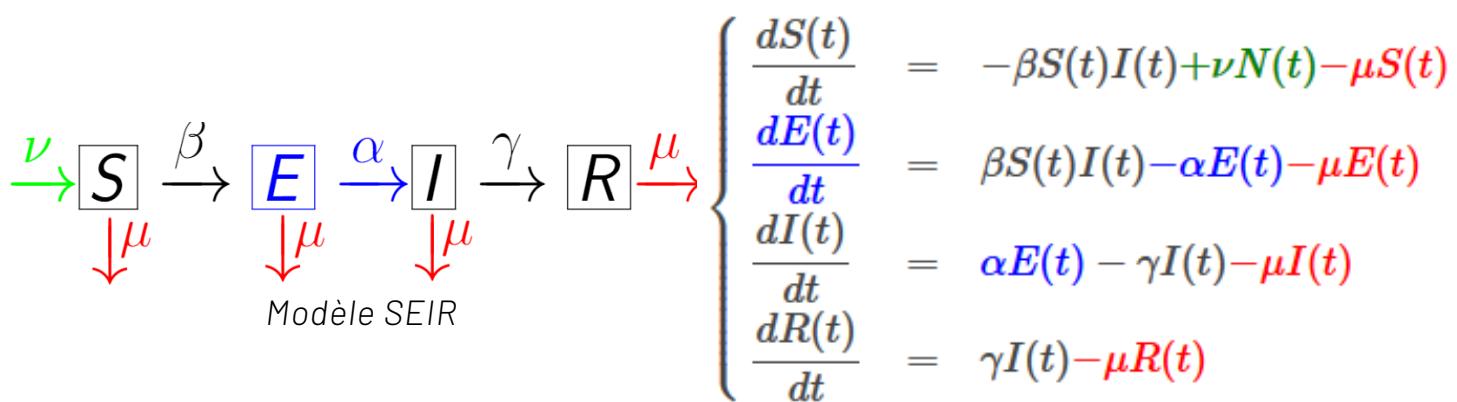
Il est important de rappeler que ce modèle mathématique ainsi que le suivant sont évidemment des modèles simplifiés, ayant des limites dont il faut être conscient. Ils restent néanmoins très utiles pour avoir un aperçu "rapide" de l'évolution d'une épidémie.

Le système contenant ces différentes équations utilisent des dérivées que l'on écrit dX/dt . Ces fonctions permettent de savoir si la variation est croissante ou décroissante sur les fonctions S, I et R en fonction du temps t, afin d'en décrire l'évolution au cours du temps.

$S(t)I(t)$ est le nombre de contacts entre des personnes saines et des personnes infectées. β est le taux de transmission qui signifie qu'il y a dès lors $\beta S(t)I(t)$ personnes nouvellement infectées à l'instant t, celles-ci ne font plus partie des personnes saines $S(t)$, et s'ajoutent aux personnes infectées $I(t)$. De plus, parmi les personnes infectées, certaines vont se rétablir et devenir immunisées.

γ étant le taux de guérison, il a $\gamma I(t)$ personnes nouvellement guéries à l'instant t qui se retirent des personnes infectées $I(t)$ et s'ajoutent aux personnes retirées $R(t)$. Cependant comme dit précédemment, ce modèle ne prend pas en compte la mortalité, en effet, les personnes retirées $R(t)$ peuvent être guéries et immunisées, mais elles peuvent également être décédées car dans les 2 cas elles "sortent" du modèle. Voici un aperçu de notre modèle en action ci-dessous.





Notre deuxième et dernier modèle de simulation épidémique à compartiments est le modèle SEIR. C'est une version plus élaborée du modèle SIR vu précédemment qui prend en compte des hypothèses en plus sur notre population $N(t)$.

Une nouvelle sous-population est représentée par la fonction $E(t)$, qui sont les personnes asymptomatiques. Elles sont contaminées par l'épidémie mais ne sont donc pas contagieuses. Un nouveau paramètre intervient pour contrôler cette population : le taux d'incubation α .

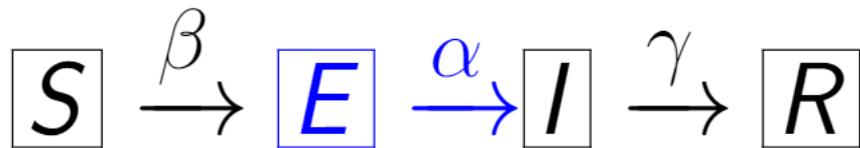
Ce modèle ajoute aussi un recyclage de la population avec les paramètres : le taux de natalité ν ainsi que le taux de mortalité μ .

Ce modèle prend donc en compte 5 paramètres :

- β qui représente le taux d'infection.
- γ qui représente le taux de guérison.
- α qui représente le taux d'incubation.
- ν qui représente le taux de natalité naturelle.
- μ qui représente le taux de mortalité naturelle.

Les personnes infectées ne vont plus directement dans le compartiment "Infectés" mais passent par celui des "Exposés" avant d'arriver dans celui-ci. Il y a donc $\alpha E(t)$ personnes nouvellement infectées à l'instant t , celles-ci ne font plus partie des personnes $E(t)$, et s'ajoutent aux personnes $I(t)$.

Maintenant rentrons dans les détails et expliquons comment passer du modèle SIR à celui-ci en ajoutant l'équation différentielle de $E(t)$, le taux d'incubation, le taux de natalité naturelle et le taux de mortalité.



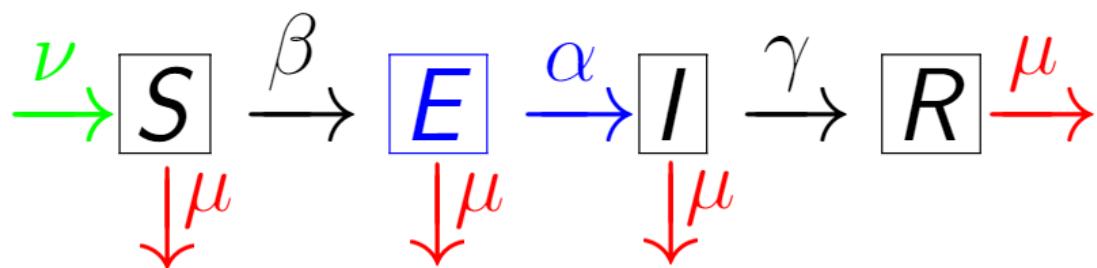
$$\begin{cases} \frac{dS(t)}{dt} = -\beta S(t)I(t) \\ \frac{dE(t)}{dt} = \beta S(t)I(t) - \alpha E(t) \\ \frac{dI(t)}{dt} = \alpha E(t) - \gamma I(t) \\ \frac{dR(t)}{dt} = \gamma I(t) \end{cases}$$

Notre nouvelle population $E(t)$ nous permet désormais de prendre en compte la durée d'incubation grâce α en ajoutant un terme $\alpha E(t)$ qui va se soustraire aux personnes asymptomatiques et s'ajouter à la population des infectés $I(t)$.



$$\begin{cases} \frac{dS(t)}{dt} = -\beta S(t)I(t) + \nu N(t) \\ \frac{dE(t)}{dt} = \beta S(t)I(t) - \alpha E(t) \\ \frac{dI(t)}{dt} = \alpha E(t) - \gamma I(t) \\ \frac{dR(t)}{dt} = \gamma I(t) \end{cases}$$

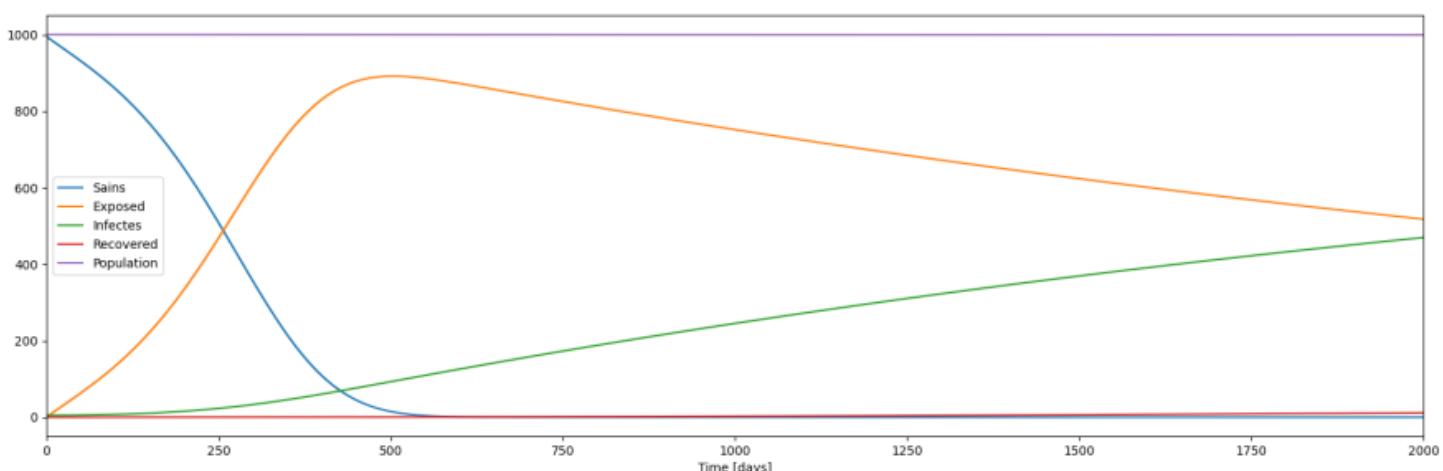
Après le taux d'incubation, il est temps d'ajouter notre taux de natalité ν en ajoutant le terme $\nu N(t)$. Nous supposons que les personnes naissant au sein de notre population $N(t)$ sont saines, par conséquent nous allons additionner le taux de natalité à la population saine $S(t)$.



$$\begin{cases} \frac{dS(t)}{dt} = -\beta S(t)I(t) + \nu N(t) - \mu S(t) \\ \frac{dE(t)}{dt} = \beta S(t)I(t) - \alpha E(t) - \mu E(t) \\ \frac{dI(t)}{dt} = \alpha E(t) - \gamma I(t) - \mu I(t) \\ \frac{dR(t)}{dt} = \gamma I(t) - \mu R(t) \end{cases}$$

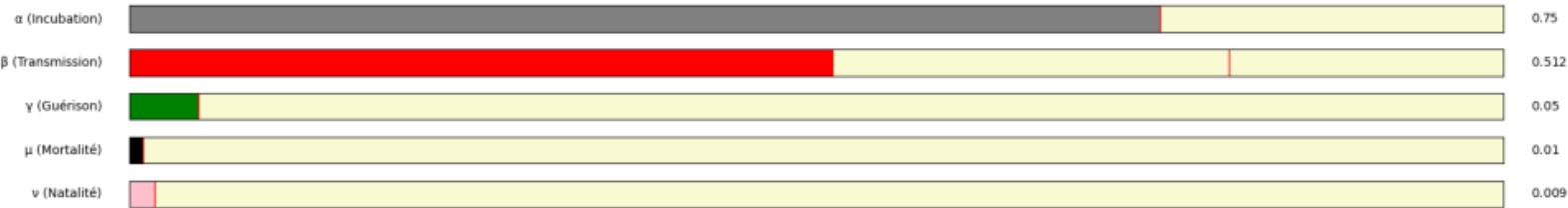
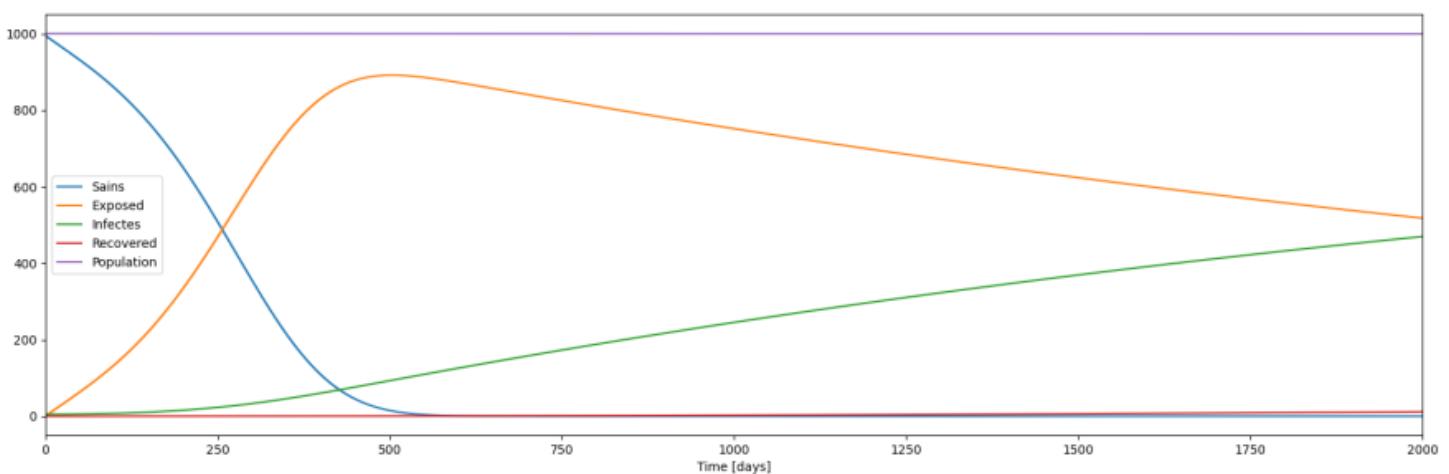
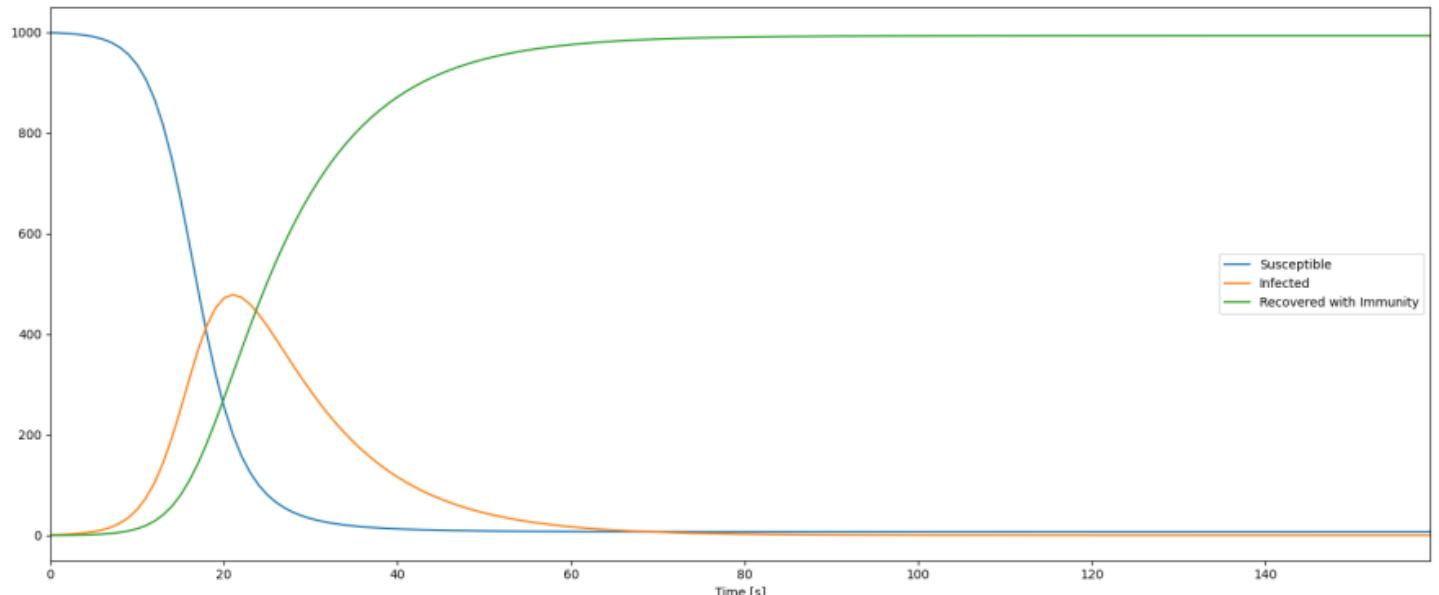
Ensuite, il faut ajouter notre taux de mortalité μ en ajoutant cette fois-ci plusieurs termes. En effet, une personne peut mourir quel que soit son état de santé (S, E, I ou R) d'une cause naturelle non liée à l'épidémie. On va alors ajouter 4 termes $\mu S(t)$, $\mu E(t)$, $\mu I(t)$ et $\mu R(t)$ qui vont se soustraire à toutes nos sous-populations.

Voici un aperçu de notre modèle en action ci-dessous.



α (Incubation)	<div style="width: 150px;"></div>	0.75
β (Transmission)	<div style="width: 100px;"></div>	0.512
γ (Guérison)	<div style="width: 10px;"></div>	0.05
μ (Mortalité)	<div style="width: 5px;"></div>	0.01
ν (Natalité)	<div style="width: 10px;"></div>	0.009

Pour conclure, voici à quoi ressemblent nos 2 graphiques utilisant ces modèles. On constate facilement des variations d'un graphique à un autre, celles-ci sont causées par l'influence de chaque paramètre et des différentes équations.



Pour commencer, il faut d'abord définir nos valeurs pour pouvoir les implémenter dans notre code.

Voici un tableau résumant toutes ces valeurs importantes que nous avons importées/utilisées dans notre code.

On y retrouve la population globale ainsi que les sous-populations implémentées avec leurs équations différentielles ainsi que nos paramètres.

NO	Population totale initiale.
S0	Effectif initial de personnes Saines
E0	Effectif initial de personnes infectées non-infectieuses.
I0	Effectif initial de personnes infectées infectieuses.
R0	Effectif initial de personnes retirées.
SIM_PRECISION	Nombre de pas à effectuer par simulation.
SIM_MULTIPLIER	Longueur de la simulation.
INIT_ALPHA	Taux d'incubation initial.
INIT_BETA	Taux de transmission initial.
INIT_GAMMA	Taux de guérison initial.
INIT_MICRO	Taux de mortalité naturelle initial.
INIT_NU	Taux de natalité naturelle initial.

```

1 def solve(S0, E0, I0, R0, alpha, beta, gamma, micro, nu):
2     S, E, I, R = [S0], [E0], [I0], [R0]
3     h = 1/SIM_PRECISION
4     for o in range(SIM_PRECISION*SIM_MULTIPLIER):
5         St, Et, It, Rt = S[o-1], E[o-1], I[o-1], R[o-1]
6
7         #Equations
8         dSdt = -beta*St*It + nu*(St+Et+It+Rt) - micro*St
9         dEdt = beta*St*It - alpha*Et - micro*Et
10        dIdt = alpha*Et - gamma*It - micro*It
11        dRdt = gamma*It - micro*Rt
12
13        S.append(St+h* dSdt)
14        E.append(Et+h* dEdt)
15        I.append(It+h* dIdt)
16        R.append(Rt+h* dRdt)
17    return S, E, I, R
18

```

Maintenant, il est temps d'implémenter toutes ces valeurs en code pour pouvoir générer nos graphiques interactifs en utilisant différentes méthodes et fonctions. Pour commencer, notre fonction `solve()` nous permet de résoudre les équations du système SEIR grâce à la méthode d'Euler explicite :

- les variables dX/dt représentent les dérivées du modèle.
 - S, E, I, R sont directement les listes de points de nos courbes respectives.
 - `SIM_PRECISION` représente le nombre de pas auxquels nous calculons le modèle.
- `SIM_MULTIPLIER` définit la longueur de la simulation.

Cette fonction calcule le résultat des équation différentielles tous les $1/SIM_PRECISION$ avec la méthode d'euler explicite.

Nous allons sortir une liste des résultats aux différents temps t . On a donc :

$$S[t + h] = S[t] * h + dSdt$$

La fonction retourne enfin un n-uplet des listes de points de S, E, I, R (et plus si d'autres modèles)

```

fig, ax = plt.subplots()
ax.margins(x=0)

line1, = plt.plot(S, label="Healthy")
line2, = plt.plot(E, label="Exposed")
line3, = plt.plot(I, label="Infected")
line4, = plt.plot(R, label="Recovered")
line5, = plt.plot(V, label="Vaccinated")
line6, = plt.plot(M, label="Dead by Disease")
line7, = plt.plot(N, label="Population")

# Ajustement des tracés principaux pour faire de la place aux sliders
plt.subplots_adjust(left=0.1, bottom=0.5, top=1)
ax.set_xlabel('Time [days]')
ax.legend()

```

Pour l'affichage nous récupérons les listes de points (S, E, I, R, ...) et on utilise la méthode `plot()` sur la ligne correspondante avec la liste de points en paramètre et un label pour une meilleure lecture.

Et enfin nous utilisons `plt.plot()` pour afficher le graphe initial.

```

def update(_x):
    """
    Méthode appelée à chaque changement des sliders. Recalcule les courbes et les affiche.
    """
    S, E, I, R, V, M = solve(S0, E0, I0, R0, V0, M0, alpha_slider.val, beta_slider.val, gamma_slider.val, micro_slider.val, nu_slider.val, epsilon_slider.val, delta_slider.val)
    line1.set_ydata(S)
    line2.set_ydata(E)
    line3.set_ydata(I)
    line4.set_ydata(R)
    line5.set_ydata(V)
    line6.set_ydata(M)
    N = [S[i] + E[i] + I[i] + R[i] + V[i] for i in range(len(S))]
    line7.set_ydata(N)
    fig.canvas.draw_idle()

```

Pour la mise a jour avec les sliders, chaque changement va appeler la fonction `update()` qui va recuperer les valeurs des sliders avec `slider.val`, exécuter le solver avec ces nouvelles valeurs et mettre a jour les courbes avec `line.set_ydata()`.

Enfin on apelle `fig.canvas.draw_idle()` pour mettre a jour la fenêtre.

Chaque slider est déclaré de cette facon:

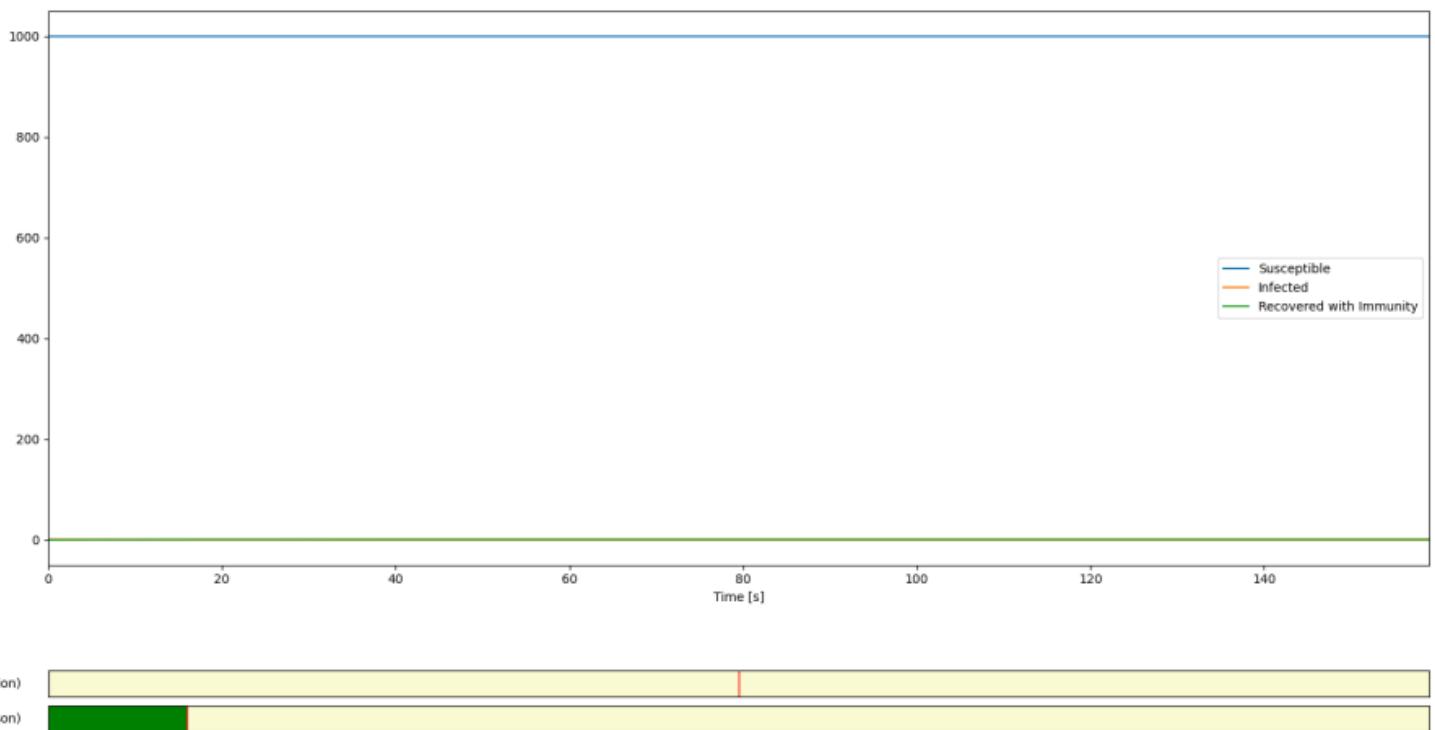
```

# Slider Horizontal alpha
alpha_slider = Slider(
    ax=plt.axes([0.1, 0.31, 0.8, 0.03], facecolor="lightgoldenrodyellow"),
    label='α (Incubation)',
    valmin=0,
    valmax=1,
    valinit=INIT_ALPHA,
    color="grey"
)

```

Après avoir construit notre système d'équations différentielles, ajouté les différents paramètres, termes et implémenté le tout dans notre code, on peut passer à l'analyse de nos graphiques basés sur les 2 modèles.

Modèle SIR - 1ère Observation : Lorsque le taux de transmission est à 0.



$$\frac{dS}{dt} = -\beta * S * I / N$$

Comme $\beta = 0$ $dS/dt = 0$ donc la courbe est constante.

$$\frac{dI}{dt} = (\beta * S * I / N) - (\gamma * I)$$

Comme $\beta = 0$, $dI/dt = -(\gamma * I)$ la courbe des personnes infectées est décroissante.

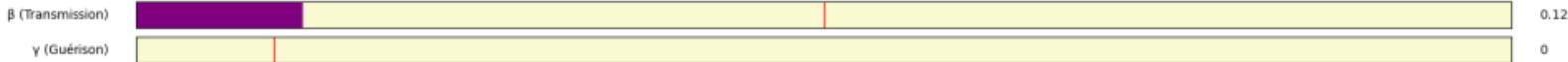
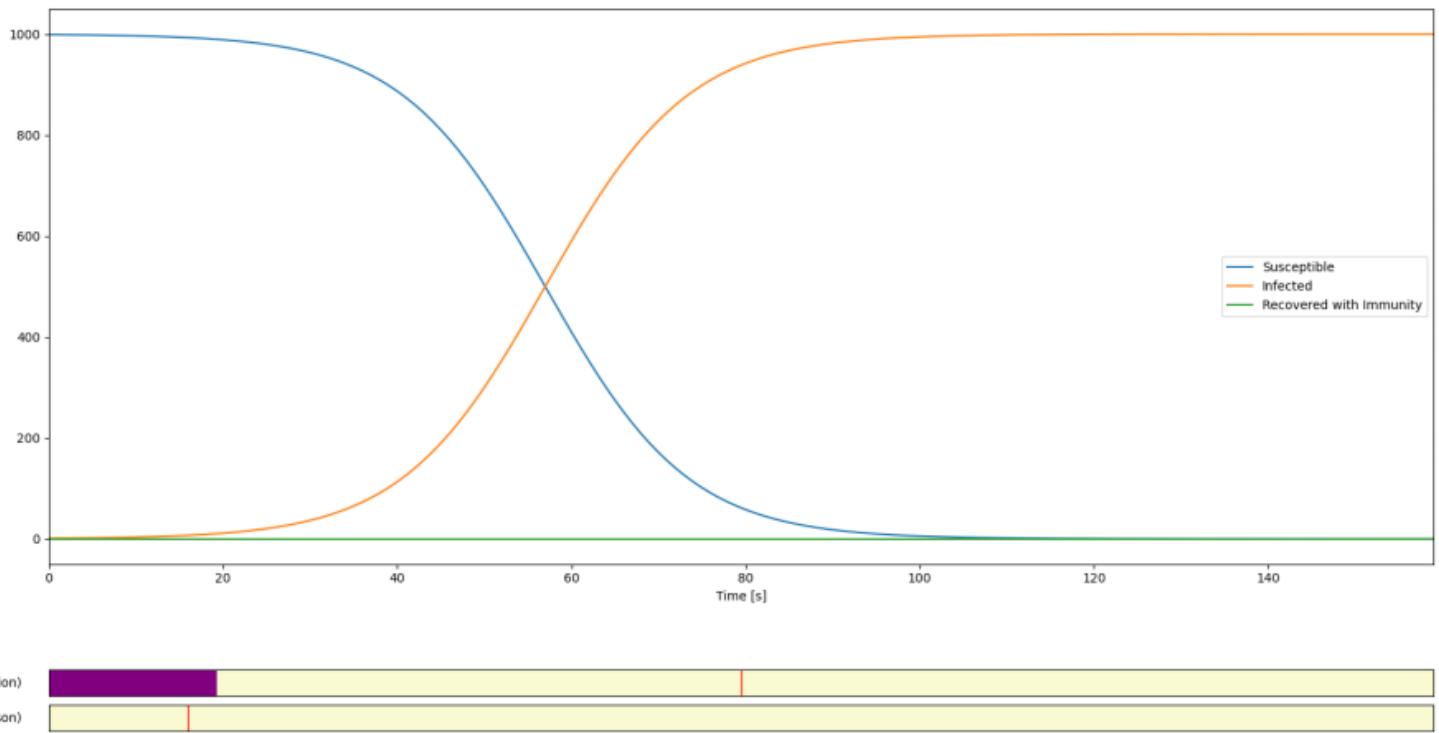
$$\frac{dR}{dt} = \gamma * I$$

Cette courbe est croissante.

$$\begin{cases} \frac{dS(t)}{dt} &= -\beta S(t)I(t) \\ \frac{dI(t)}{dt} &= \beta S(t)I(t) - \gamma I(t) \\ \frac{dR(t)}{dt} &= \gamma I(t) \end{cases}$$

Conclusion : si le taux de transmission est à 0 , le nombre de personnes infectées n'augmente pas.

Modèle SIR - 2ème Observation : Lorsque le taux de guérison est à 0.



$$\frac{dS}{dt} = -\beta * S * I / N$$

$$\frac{dI}{dt} = (\beta * S * I / N) - (\gamma * I)$$

Comme $\gamma = 0$, dI/dt est croissante.

$$\frac{dR}{dt} = \gamma * I$$

Comme $\gamma = 0$, $dR/dt = 0$ donc la courbe est constante.

$$\begin{cases} \frac{dS(t)}{dt} &= -\beta S(t)I(t) \\ \frac{dI(t)}{dt} &= \beta S(t)I(t) - \gamma I(t) \\ \frac{dR(t)}{dt} &= \gamma I(t) \end{cases}$$

Conclusion : le nombre de personnes rétablies n'augmente pas et toutes les personnes seront forcément infectées après un temps t plus ou moins long selon le taux de transmission.

Modèle SEIR - Nos Observations : Lorsque la natalité est égale à la mortalité.

$$\frac{d/S}{dt} = -\beta * S * I + \nu * (S + E + I + R) - \mu * S$$

$$\frac{d/E}{dt} = \beta * S * I - \alpha * E - \mu * E$$

$$\frac{d/I}{dt} = \alpha * E - (\gamma + \mu) * I$$

$$\frac{d/R}{dt} = \gamma * I - \mu * R$$

$$N = S + E + I + R$$

$$\frac{d/N}{dt} = -\beta * S * I + \nu * (S + E + I + R) - \mu * S + \beta * S * I - \alpha * E - \mu * E + \alpha * E - (\gamma + \mu) * I + \gamma * I - \mu * R$$

Ce qui donne au final :

$$\frac{d/N}{dt} = \nu - \mu$$

Or si $\nu = \mu$, la dérivé est égale 0, donc la population reste constante.

Lorsque $\nu = \mu$, la dérivé est égale à 0 et que donc la population reste constante.

Cela signifie que ces 2 paramètres auront une influence sur le sens de variation de l'équation :

$$\frac{d/N}{dt} = \nu - \mu$$

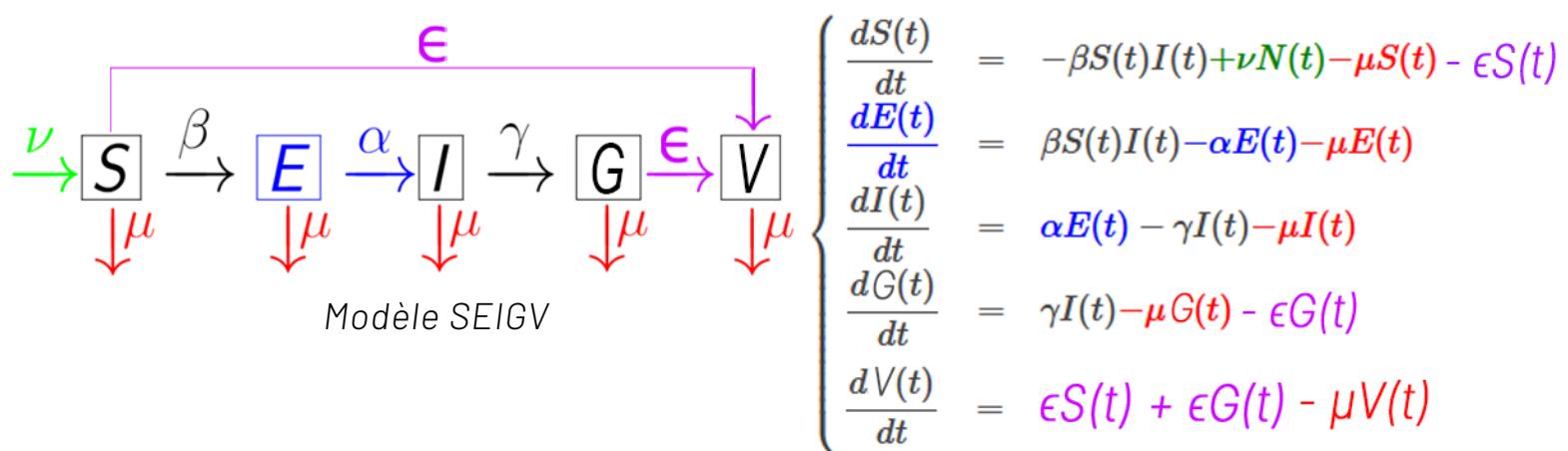
On en déduit l'existence de 2 théorèmes résultant de $d/N/dt$:

- Si le taux de natalité est à 0 et que donc $d/N/dt = -\mu$
Alors la population sera décroissante.

(dans le cas inverse)

- Si le taux de mortalité est à 0 et que donc $d/N/dt = \nu$
Alors la population sera croissante.

Dans cette partie, nous avons souhaité aller plus loin que le sujet initialement choisi. Dans un premier temps, nous nous sommes reconcentrés sur nos 2 modèles à compartiments afin de savoir si nous ne pouvions pas les rendre encore plus complexes avec des nouvelles sous-populations et des paramètres supplémentaires.



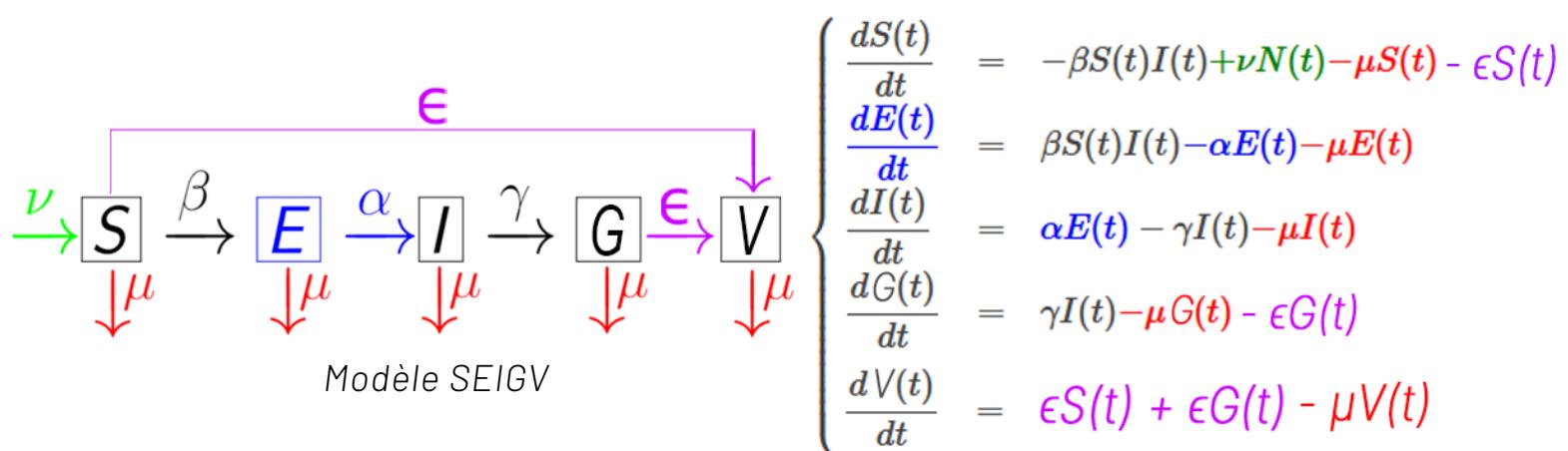
Notre premier modèle à compartiments que nous avons réalisé s'appelle SEIGV, il reprend le modèle SEIR mais cette fois-ci, on prendra en compte la vaccination. La population $R(t)$ qui symbolisait les "Retirés" a été renommé en $G(t)$ qui représentera désormais les "Guéris".

Une nouvelle sous-population est représentée par la fonction $V(t)$, celle-ci correspond aux personnes vaccinées. Elles ont été contaminé par l'épidémie puis elles se sont fait vacciner et ne sont plus contagieuses. Un nouveau paramètre intervient pour contrôler cette population : le taux de vaccination ϵ .

Ce modèle prend donc en compte 6 paramètres :

- β qui représente le taux d'infection.
- γ qui représente le taux de guérison.
- α qui représente le taux d'incubation.
- ν qui représente le taux de natalité naturelle.
- μ qui représente le taux de mortalité naturelle.
- ϵ qui représente le taux de vaccination.

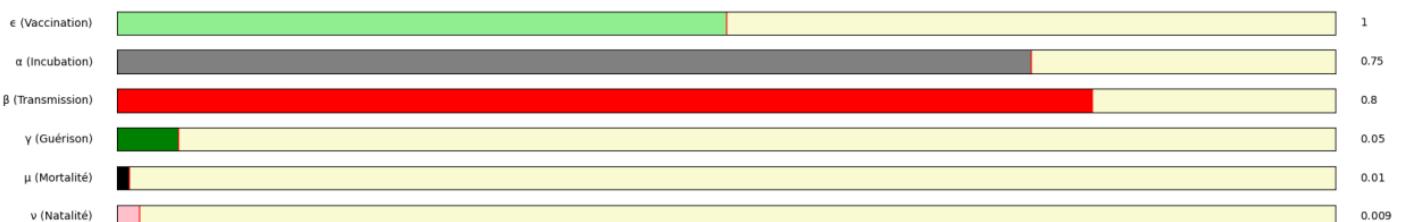
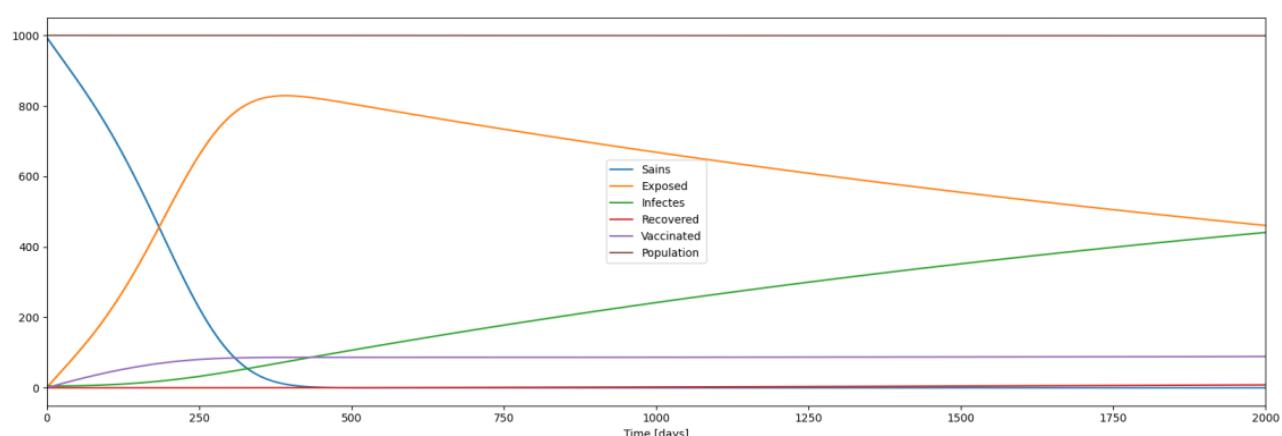
Désormais dans ce modèle, nous avons notre nouvelle population $V(t)$ qui nous permet de prendre en compte l'impact de la vaccination grâce au taux de vaccination ϵ .



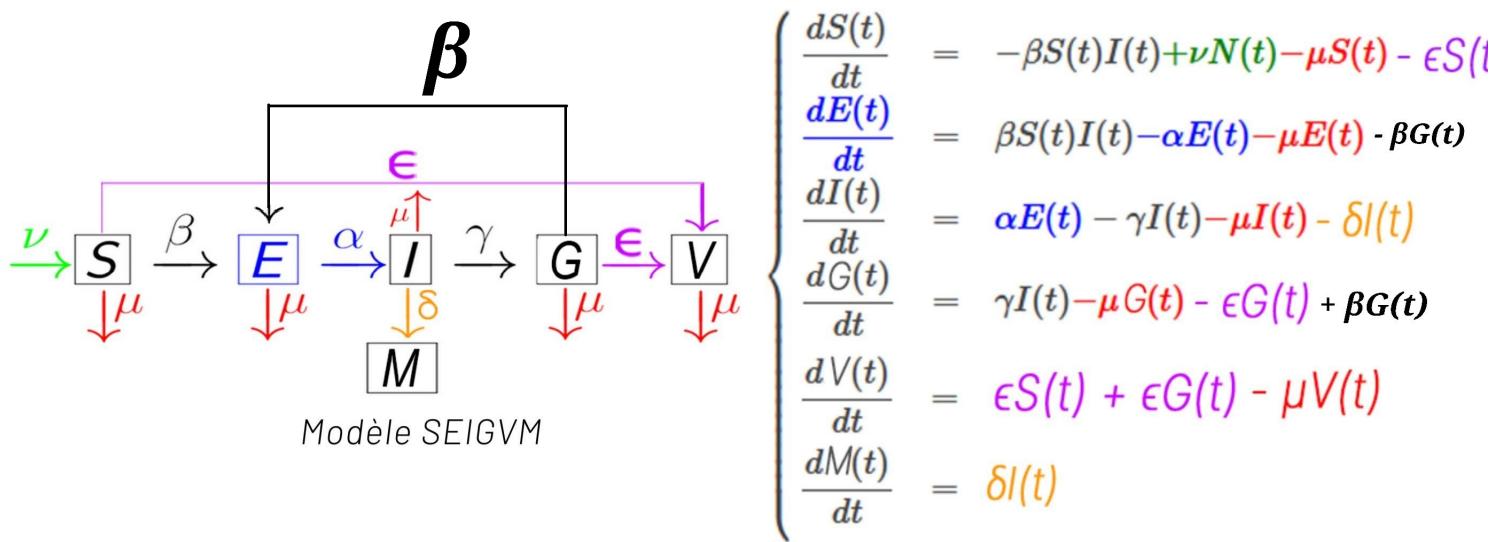
Maintenant, il faut l'ajouter grâce à plusieurs termes. En effet, les vaccinés ne sont plus seulement des "Guéris", il faut donc les soustraire à $G(t)$ en ajoutant le terme $\epsilon G(t)$ à l'équation. Par la même occasion, cette situation se reproduira également chez les "Saints" où l'on va devoir soustraire les nouveaux "Vaccinés" avec $\epsilon S(t)$.

Par conséquent, notre nouvelle équation $V(t)$ sera composée de ces 2 termes tout en enlevant les personnes mortes naturellement par le terme $\mu V(t)$.

Voici un aperçu du modèle en action ci-dessous.



Dorénavant, l'impact de la vaccination est un facteur sur notre population N. Or, il restait une problématique sur tous ces modèles, qui était la prédiction de la mortalité de l'épidémie. On a donc décidé de tenter l'expérience en créant un dernier modèle prenant en compte ce problème.



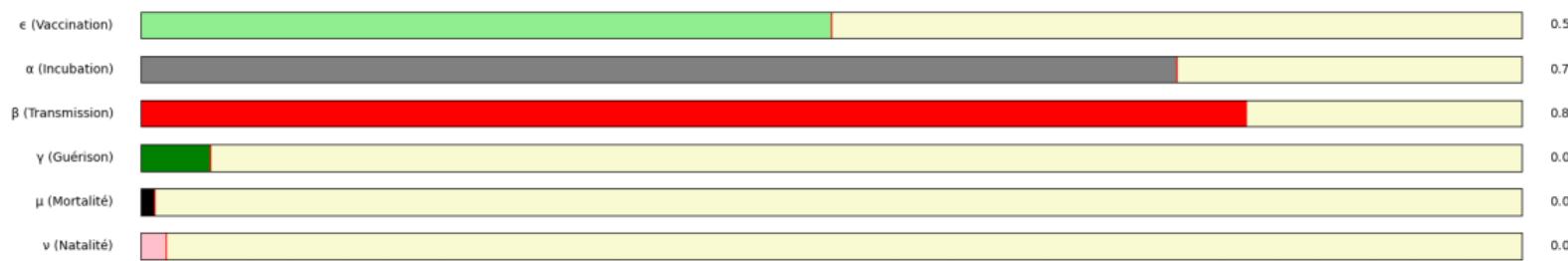
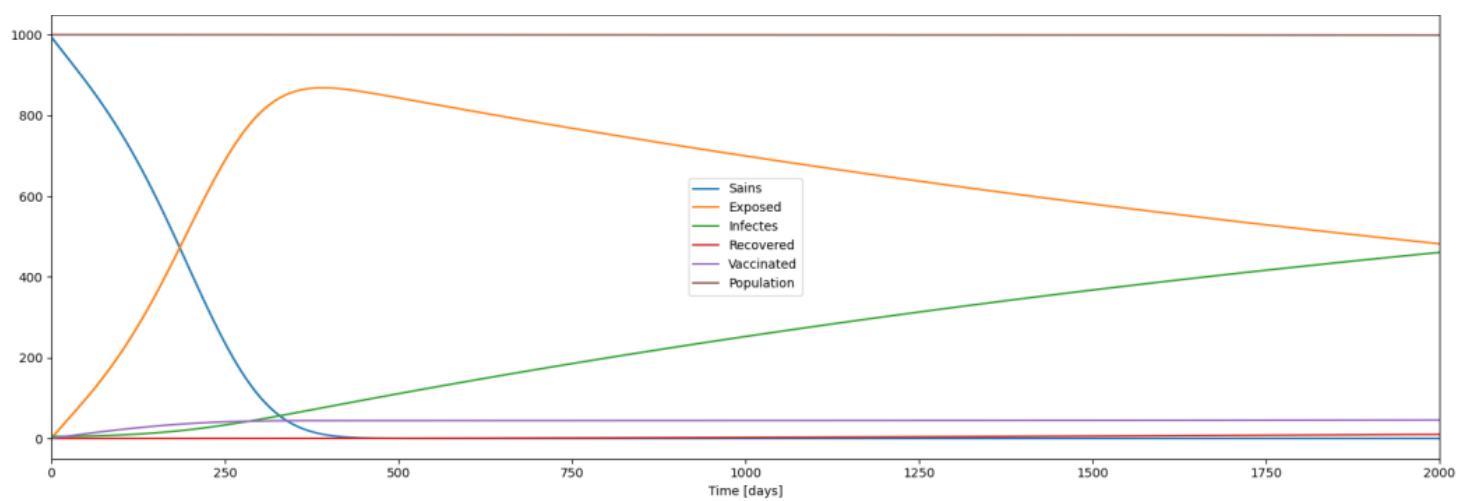
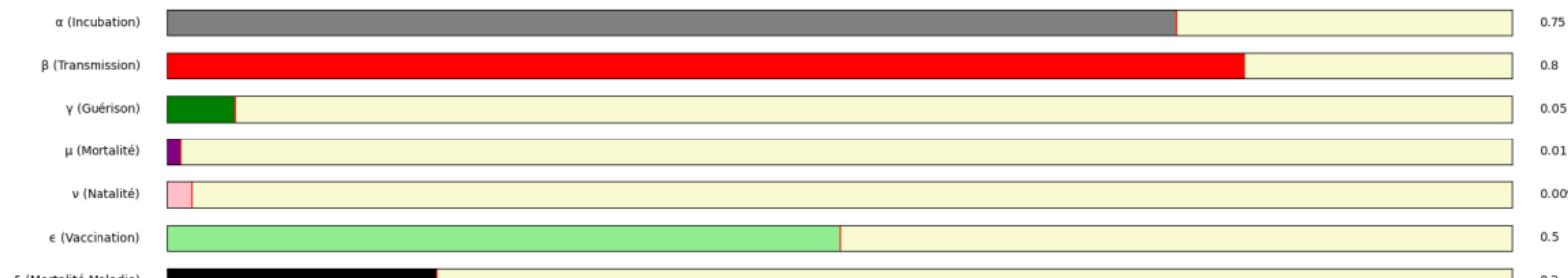
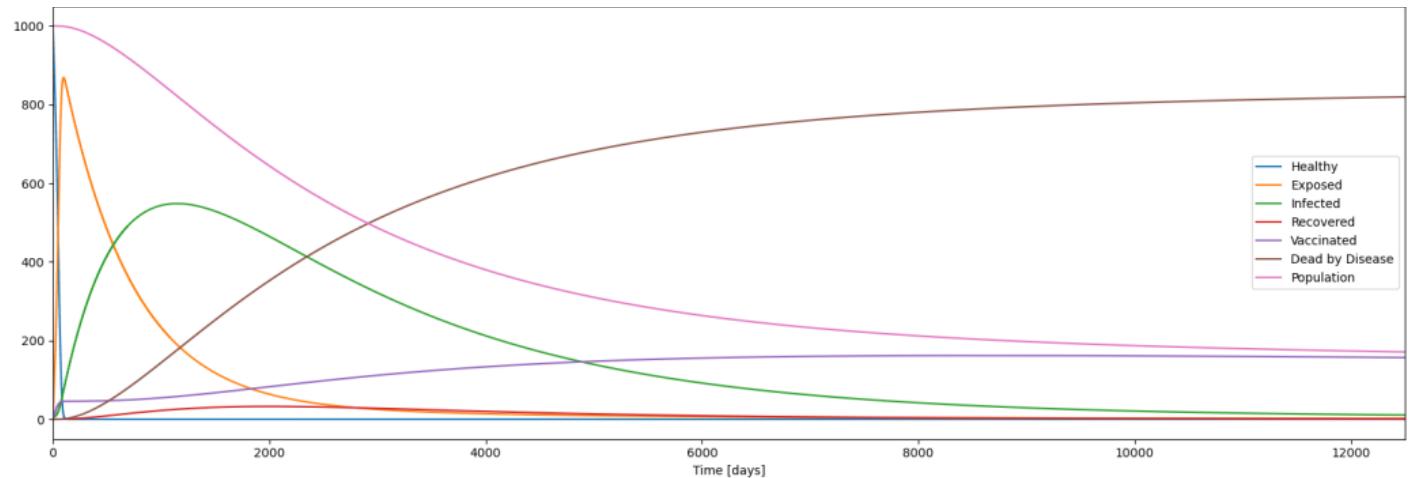
Notre second modèle à compartiments que nous avons réalisé s'appelle SEIGVM, La nouvelle sous-population représentée par la fonction $M(t)$ symbolisera les personnes "Mortes" du virus. Forcément, qui dit nouvelle sous-population dit aussi nouveau paramètre qui fait son apparition : le taux de mortalité du virus δ .

Ce modèle prend donc en compte 7 paramètres :

- β qui représente le taux d'infection.
- γ qui représente le taux de guérison.
- α qui représente le taux d'incubation.
- ν qui représente le taux de natalité naturelle.
- μ qui représente le taux de mortalité naturelle.
- ϵ qui représente le taux de vaccination.
- δ qui représente le taux de mortalité de l'épidémie.

Dans ce modèle, on considère que les personnes "Guéris" ne sont plus immunisées au virus, et peuvent donc par conséquent redevenir des personnes "Exposés". Cela crée donc une boucle dont on ne peut sortir que si toutes les personnes se vaccinent.

Il est logique que les personnes décédées de l'épidémie ont été infectées avant, c'est donc pour cela que l'équation de $M(t)$ sera composée seulement du terme $\delta I(t)$ que l'on soustrait dans $I(t)$. Voici un aperçu de nos 2 modèles ci-dessous.



Observations des rendus :

On cherche ici à déterminer grâce au modèle SEIGVM, quelle mesure est la plus utile pour que le nombre de morts total soit le plus bas possible.

Pour cela, on part des paramètres suivants :

- taux d'incubation $\alpha = 0,75$
- taux d'infection $\beta = 0,8$
- taux de guérison $\gamma = 0,5$
- taux de mortalité naturelle $\mu = 0,01$
- taux de natalité $\nu = 0,01$
- taux de vaccination $\epsilon = 0,5$
- taux de mortalité liée au virus $\delta = 0,2$

(sur 12500 jours et 1000 personnes)

Dans ce cas précis, le nombre de morts est de 379.

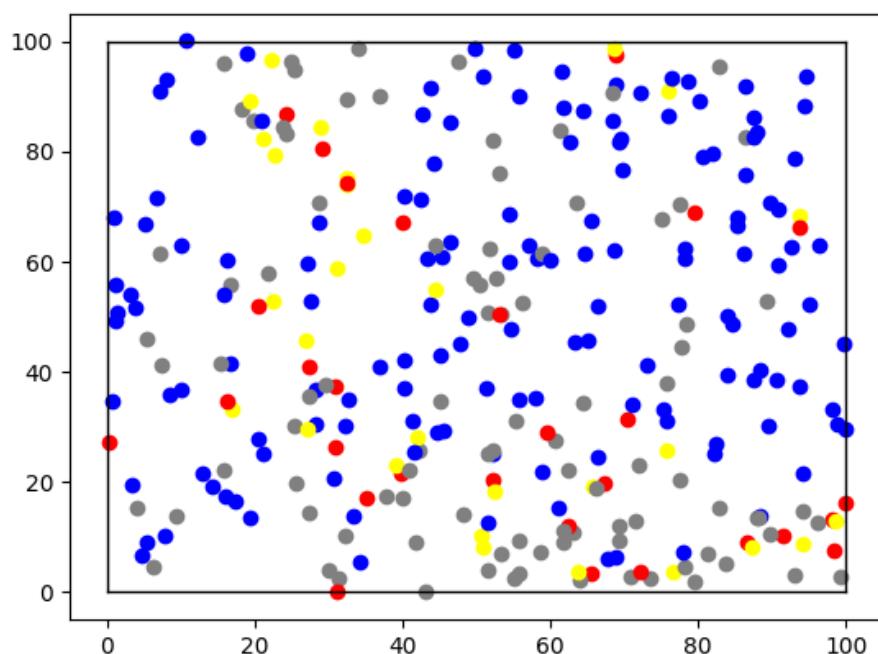
Selon des études apparues sur le site d'Europe 1, le premier confinement a fait baisser le taux de transmission du coronavirus de 84%. Si on baisse le taux de transmission de 84%, soit $0,8 - 0,8 * 84 / 100 = 0,8 - 0,672 = 0,128$, le nombre de morts final est de 348.

Si à l'inverse on augmente le taux de vaccination à 0,9, le nombre de morts est de 339. On voit donc bien que si énormément de personnes se vaccinent rapidement, le nombre de morts sera moins élevé que si on confine simplement.

Or, sur un nombre de jours moins élevé (1250) on obtient les chiffres suivants:

- 143 en ne faisant rien
- 120 en limitant la transmission
- 137 en augmentant la vaccination

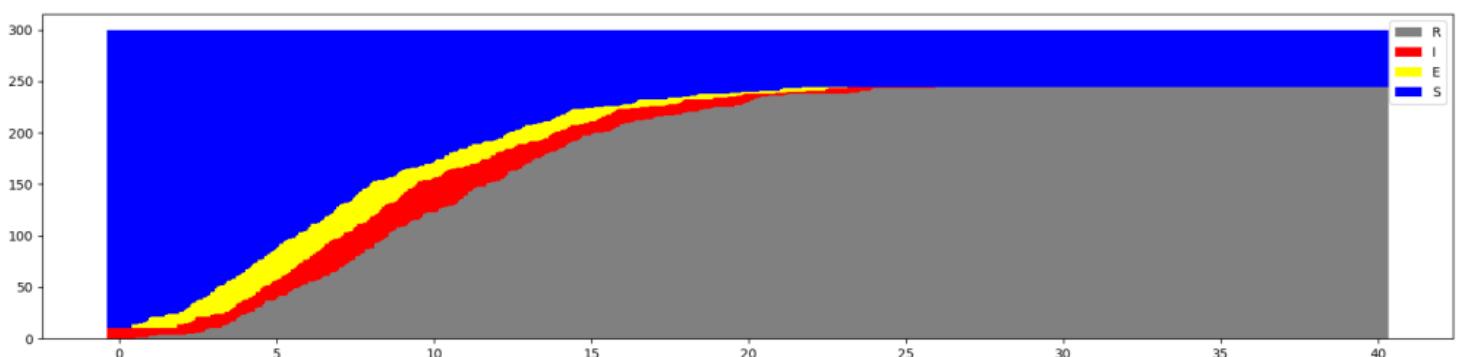
Sur une échelle plus courte, le confinement est plus efficace que la vaccination.



https://github.com/ozeliurs-MaximeBilly/SIR-Model/raw/main/gifs/sim_300.gif

Après avoir terminé nos courbes, on a souhaité se pencher sur une simulation interactive 2D qui illustrerait le principe de fonctionnement des courbes du modèle SEIR. Nous avons réalisé cette simulation sur une population de 300 personnes où les boules représentent chacune un individu qui peut soit se déplacer, se contaminer, incuber ou décéder.

Cette simulation nous permet d'avoir un aperçu beaucoup plus logique de nos équations et également vérifier si les résultats obtenus sur nos modèles à compartiments réagissent correctement dans un contexte plus réaliste.



Voilà un aperçu de la courbe résultante de cette simulation.

Comme avec nos modèles à compartiments, on retrouve les mêmes profils de courbes ainsi que les mêmes caractéristiques relevées dans nos précédentes observations.

Au final, notre projet nous aura permis d'apprendre et de concevoir de nombreuses choses. Nous avons appris à créer et à résoudre des systèmes d'équations différentielles ordinaires linéaires, et à l'implémenter dans un code grâce à des méthodes de résolution comme "Euler explicite". Ceci à contribuer à approfondir grandement nos connaissances mathématiques.

Au niveau du code, nous avons appris comment utiliser la bibliothèque MatPlotLib ainsi que son fonctionnement pour concevoir nos graphiques avec nos courbes ainsi que notre simulation 2D interactive. De plus, l'implémentation du "solver" paraissait simple en apparence mais faire le lien entre les mathématiques et l'implémentation sur Python s'est démontré être un défi assez fastidieux.

Ensuite, voyant que notre projet était stable, fonctionnel et plausible, nous sommes allé beaucoup plus loin dans la partie mathématique en essayant de créer notre propre modèle à compartiments tout en répondant aux problématiques les plus contraignantes.

Pour conclure, notre projet nous a ouvert les yeux sur l'importance des mathématiques au sein des langages de programmation ainsi qu'avoir un œil un peu plus critique sur le contexte sanitaire actuel.

La dynamique des populations

Documentation Python 3 :

- <https://docs.python.org/3/library/>

Morgan Morancey - Bref tutoriel sur SageMath :

- <http://mmorancey.perso.math.cnrs.fr/Tutoriel.html>

Documentation des modules Scipy utilisés :

- <https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.odeint.html>
- https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.solve_ivp.html#scipy.integrate.solve_ivp

Documentation Matplotlib :

- <https://matplotlib.org/>

Explication des équations différentielles et les méthodes de résolution :

- https://fr.wikipedia.org/wiki/Équation_déférentielle_ordinaire
- https://fr.wikipedia.org/wiki/Méthode_d'Euler

Explication du modèle SIR :

- https://en.wikipedia.org/wiki/Compartmental_models_in_epidemiology#The_SIR_model

Explication du modèle SEIR :

- https://en.wikipedia.org/wiki/Compartmental_models_in_epidemiology#The_SEIR_model

Images des Mathématiques - Modélisation d'une épidémie :

- <http://images.math.cnrs.fr/Modelisation-d-une-epidemie-partie-1.html>
- <http://images.math.cnrs.fr/Modelisation-d-une-epidemie-partie-2.html>