Dynamique de réseaux multi-échelles complexes sous contraintes: Modélisation et Analyse

Liam Toran

Stage de fin de M2A 2019 au Laboratoire J.A. Dieudonné de l'Université de Nice
sous la supervision de Yves D'Angelo, Rémi Catellier et Laurent Monasse

Thèmes : Mathématiques et leurs Interactions, Modélisation, Analyse, Processus Stochastiques, Équations aux dérivées partielles et ordinaires, Stabilité, Réaction-Diffusion, Ondes progressives, Simulation Numérique.

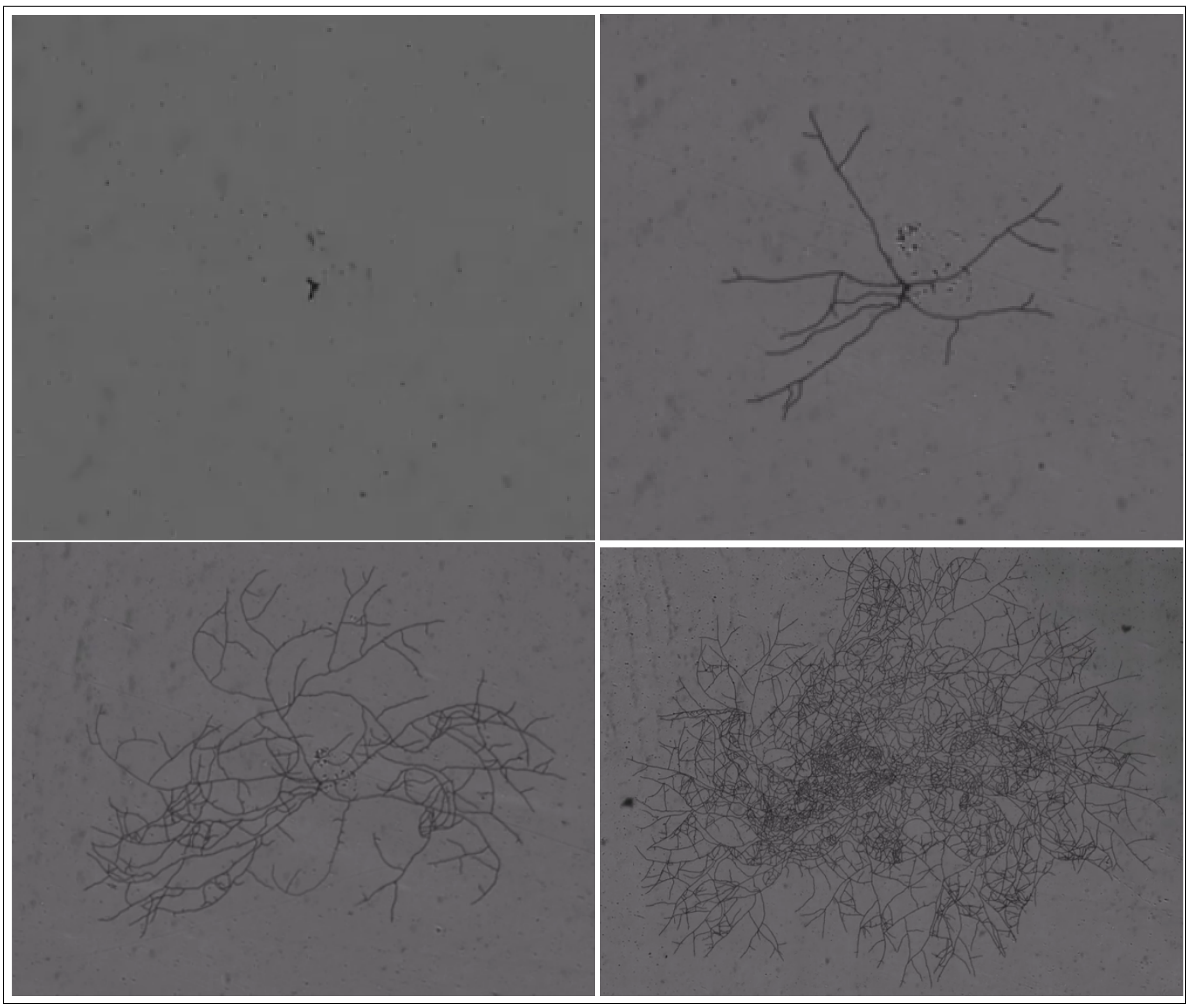

Figure 1 – Capture d'un réseau de champignon en expansion, par

Table des matières

1 L'équation de Fisher ou KPP

1.1 Préliminaire

Notre point de départ est l'équation de diffusion :

$$\partial_t u = \Delta u \tag{1}$$

En plus de la diffusion, considérons des modèles où le taux d'accroissement de u dépend aussi de la densité u.
Ceci donne les équations de reaction-diffusion :

$$\partial_t u = \Delta u + F(u) \tag{2}$$

où F est assez lisse.
Il est souvent naturel dans les modèles de considérer $F(u)$ proportionnel à u pour u petit ("croissance"), et quand u devient proche de 1, l'accroissement $F(u)$ s'arrête : $F(1) = 0$ ("saturation").
Ces types de modèles ont étés introduits et examinés par les travaux de Fisher[1] et Kolmogorov, Petrovsky et Piscounuv (abrégés KPP).
Un exemple d'une telle équation est :

$$\partial_t u = \Delta u + ru(1-u) \tag{3}$$

où $r > 0$, qui sera dans la suite étudiée dans le cas 1-dimensionnel en x : $u = u(x,t)$.

1.2 Réaction

En observant les solutions constantes en x : $u(x,t) = v(t)$ dans (3), l'équation différentielle ordinaire (EDO ou ODE) :

$$\partial_t v = r(v - v^2) = F(v) \tag{4}$$

est obtenue.
Il y a deux équilibres ($F(v) = 0$)) pour $v = 0$ et $v = 1$.Par le théorème de stabilité de Lyapunov, $F'(0) > 0$ montre que $v = 0$ est instable et $F'(1) < 0$ montre $v = 1$ est asymptotiquement stable.

1.3 Réaction-Diffusion

Dans l'espace $X = C^0_{b,unif}(\mathbb{R},\mathbb{R})$ des fonctions bornées et uniformément continues, il y a existence locale et unicité des solutions de l'équation de Fisher-KPP (2). Grâce à un principe du maximum, il y a aussi existence globale et unicité des solutions.

Théorème 1. :
Existence et Unicité de la solution dans X : Soit $U_0 \in X$. Il existe une unique solution de l'équation de Fisher-KPP (2) $U \in C([0,\infty[, X)$ avec condition initiale U_0.

Théorème 2. :
Principe du Maximum : Soit u_1 et u_2 deux solutions de (2). Si il éxiste t_0 tel que $u_1(x,t_0) < u_2(x,t_0)$ $\forall x$ alors $u_1(x,t) < u_2(x,t)$ $\forall x$ et $\forall t > t_0$

1.4 Solutions en onde plane stationnaire / onde progressive

Rappellons la définition d'une solution en onde plane stationnaire / onde progressive :

Définition 1.1. Solutions en onde plane stationnaire
Une solution en onde plane stationnaire est une solution de la forme $u(x,t) = h(x - st)$ où $c \in \mathbb{R}$.
On fera parfois l'abus de notation $u(x,t) = u(x - st)$

Sous des hypothèses "faibles" sur F, l'équation (2) : $\partial_t u = \Delta u + F(u)$ a alors la propriété surprenante et importante de posséder des solutions en ondes planes stationnaires liant les états d'équilibre $u = 1$ (à $-\infty$) et $u = 0$ (à $+\infty$).
Les hypothèses sur F portent en partie sur le fait que (2) doit posséder :
- Deux états d'équilibre $u = 1$ et $u = 0$: $F(0) = F(1) = 0$:
- Un phénomène de "croissance" : $F'(0) > 0$
- Un phénomène de "saturation" : $F'(1) < 0$

Étude des solutions en ondes progressive de (2) :
En substituant $u(x,t) = h(x - st) = h(y)$ pour $y = x - st$ dans (2), les équations obtenues sur h sont :

$$\begin{cases} h''(y) + sh'(y) + F(h(y)) = 0 \\ h(-\infty) = 1 \\ h(+\infty) = 0 \end{cases} \tag{5}$$

qui est une équation elliptique non linéaire. Le problème est donc de trouver s et $h \in C^2$ tels que le système (5) soit vérifié. Le théorème obtenu est le suivant :

Théorème 3. Existence de solutions en onde progressive pour les équations de reaction-diffusion :
Soit $F \in C^1([0,1])$ tel $F(0) = F(1) = 0$ et $F \geq 0$.
Il existe une vitesse critique s_* telle que $s_*^2 \geq 4F'(0)$ et :

- i) $\forall s \geq s_*$, l'équation (5) a une solution $h_s : \mathbb{R} \to]0,1[$ de classe C^3.
Cette solution est unique à translation près.
- ii) $\forall s < s_*$ l'équation (5) n'a pas de solution $h : \mathbb{R} \to [0,1]$

Remarques :
Dans le cas ii) il existe des solutions en ondes planes mais elles ne sont pas confinées dans [0,1] ni dans $\mathbb{R}^+$, ce qui ne fait pas de sens dans une étude de densité de population.
Dans le cas de l'équation de Fisher-KPP, c'est à dire pour $F(u) = r(u - u^2)$, on a $s_*^2 = 4F'(0) = 4r$: la vitesse minimale de propagation est $s^* = 2\sqrt{r}$.

1.5 Théorèmes de sélection de la vitesse pour KPP

Le théorème important suivant est du aux travaux de Kolmogorov, Petrovsky et Piscounuv de 1937. C'est l'article et le résultat fondateur de la théorie des ondes planes dans les systèmes de réaction-diffusion.

Théorème 4. Convergence vers une solution d'onde à vitesse minimale pour les solutions de l'équation de Fisher-KPP avec une donnée initiale à support compact

Soit $u_0 \to]0,1[$ une donnée initiale à support compact. Soit u la solution de l'équation de Fisher-KPP (3) avec $r = 1$ et de donnée initiale u_0. Alors quand $t \to \infty$, u converge uniformément en x vers une solution d'onde h_{s^*} de (5) qui se de déplace à vitesse minimale $s^* = 2$:

$$\sup_{y \in \mathbb{R}} |u(y + m(t), t) - h_{s^*}(y)| \to_{t \to \infty} 0$$

où $m(t) = 2t - (3/2)\log(t) + y_0$.

Remarque : La vitesse du front est alors $s(t) = \partial_t m(t) = 2 - \frac{3}{2t} \to_{t \to \infty} 2$.

Ce résultat à été raffiné par la suite par Uchiyama, Bramson et Lau. Leurs travaux apportent plus d'informations sur comment la vitesse du front se sélectionne en fonction de la donnée initiale, et comment il est possible d'obtenir d'autres vitesses de fronts que la vitesse minimale en fonction de la donnée initiale.

Théorème 5. Sélection de la vitesse pour les solutions de l'équation de Fisher-KPP en fonction de la donnée initiale
Si $u_0 \to]0,1[$ vérifie $\liminf_{x \to -\infty} u_0(x) > 0$ et $\int_0^{+\infty} x e^x u_0(x)/dx < \infty$
alors il existe $y_0 \in \mathbb{R}$ tel que la solution de (3) avec données initiales u_0 vérifie

$$\sup_{y \in \mathbb{R}} |u(y + m(t), t) - h_{s^*}(y)| \to_{t \to \infty} 0$$

où $m(t) = 2t - (3/2)\log(t) + y_0$.

D'autres vitesses peuvent être sélectionnées : Si la donnée initiale vérifie $u_0(x) \approx e^{-\lambda_-(s)x}$ quand $x \to +\infty$ alors la solution converge vers une onde progressive de vitesse s.

2 Dyamique de Réseaux en Croissance

Dans cette section et par la suite nous étudions le modèle sur la croissance de réseaux dynamiques branchant, par exemple un champignon, proposé par Rémi Catellier, Yves D'Angelo et Cristiano Ricci, avec rescaling adéquat :

$$\begin{cases} \partial_t \mu + \nabla(\mu v) = f(C)(\mu + \rho) - \mu\rho \\ \partial_t(\mu v) + \nabla(\mu v \times v) + T\nabla\mu = -\lambda\mu v + \mu\nabla C - \mu v\rho \\ \partial_t \rho = F(v)\mu \\ \partial_t C = -b\rho C \end{cases} \tag{6}$$

L'inconnue μ représente la densité des apex du champignon.
L'inconnue ρ représente la densité des hyphes/ du réseau.
L'inconnue v représente la vitesse des apex.
L'inconnue C représente la concentration des nutriments.
Les paramètres T, λ et b sont des scalaires représentants la température, l'amortissement fluide sur la vitesse des apex, et le taux de consommation des nutriments par le réseau.
La fonction f indique l'influence de la concentration de nutriments sur la croissance du champignon. Pour avoir un état stationnaire sur la croissance du champignon,$f(0) = 0$ et $f(x)/x$ dans L^1 proche de 0 sont imposés.
La fonction F représente l'inverse du temps moyen passé par les apex dans un point donné, et est donné par l'expression :

$$F(V) = (\frac{1}{2\pi T})^{\frac{d}{2}} \int_{\mathbb{R}^d} |v| \exp(-\frac{|v-V|^2}{2T}) dv \tag{7}$$

où d est la dimension du problème. Ceci est souvent simplifié en substituant $F(V)$ par une constante : $F(V) = F_0$.

2.1 Explication des équations du système (6)

Le champignon est un réseau branchant dynamique qui peut être étudié en deux parties : les apex (pointes du réseau) représentés par leur densité μ et les hyphes (branches du réseau) représentés par leur densité ρ
Les lignes du système (6) représentent :
i) La première ligne du système est le bilan de masse sur les apex avec le terme gauche classique $\partial_t \mu + \nabla(\mu v)$. Le terme de droite est composé de : - $f(C)(\mu + \rho)$ correspondant a une croissance proportionnelle à la concentration de nutriments du réseau et la masse existante d'apex et d'hyphes, - et un terme $-\mu\rho$ qui correspond à l'anastomose : une pointe qui rencontre une branche va fusionner avec elle et être détruite. Il y a un terme de croissance et un terme de saturation comme pour le modèle KPP.
ii) La deuxième ligne est le bilan de vitesse avec le terme de gauche classique $\partial_t(\mu v) + \nabla(\mu v \times v)$. Le terme $T\nabla\mu$ représente le mouvement brownien suivi par les apex. Le terme $-\lambda\mu v$ représente un amortissement fluide dans la physique du problème. Le terme $+\mu\nabla C$ représente la tendance des apex à aller vers les milieux de forte concentration. Le terme $-\mu v\rho$ représente la perte de vitesse du à l'anastomose.
iii) La troisième ligne correspond à la relation entre les branches et les pointes : la trace laissée par les apex sont les branches.
iv) La quatrième ligne décrit l'évolution de la concentration de nutriments : ils sont consommés par les hyphe avec un taux bC où b est une constante positive.

2.2 Dérivation de l'équation "KPP avec mémoire"

En faisant tendre T et λ vers $+\infty$, avec $\frac{T}{\lambda} = K$ constant, la deuxième ligne de (6) donne :

$$+K\nabla\mu = -\mu v \tag{8}$$

En injectant ceci dans la ligne 1 du système, on obtient le système de 3 inconnues suivant :

$$\begin{cases} \partial_t \mu = K\Delta\mu + f(C)(\mu+\rho) - \mu\rho \\ \partial_t \rho = F_0 \mu \\ \partial_t C = -b\rho C \end{cases} \tag{9}$$

dit "KPP avec mémoire".

2.3 Propriétés de l'EDO "KPP avec mémoire"

Soit (μ, ρ, C) vérifiant le système d'équations suivant :

$$\begin{cases} \partial_t \mu = f(C)(\mu+\rho) - \mu\rho \\ \partial_t \rho = F_0 \mu \\ \partial_t C = -b\rho C \end{cases} \tag{10}$$

avec $f(0) = 0$ On s'interesse au comportement de (μ, ρ, C) sur $\mathbb{R}^+$

Lemme 1. C est de signe constant.
En effet on a $C(t) = C(0)\exp(-b\int_0^t \rho(s)ds)$.

Lemme 2. Soit (μ, ρ, C) tel que $(\mu(0), \rho(0)) > (0, 0)$ (les deux positifs, au moins un non nul), $C(0) > 0$.
Alors $\mu(t) \geq 0 \ \forall t > 0$

Démonstration. Supposons par l'absurde que μ devient négatif alors soit $t^* = \min(t > 0/\mu(t) < 0)$.
Alors :
$\mu(t) \geq 0 \ \forall t \leq t^*$
$\partial_t \mu(t^*) \leq 0$ par définition de t^*. (Sinon $\mu(t^* + \epsilon) > 0 \ \forall \epsilon << 1$)
$\rho(t) > 0 \ \forall t \leq t^*$ car $\partial_t \rho = F_0 \mu$ et $F_0 > 0$
$\partial_t \mu(t^*) = f(C(t^*))\rho(t^*) > 0$ ce qui est en contradiction avec la deuxième affirmation. □

Dans la suite on se place dans le cas où $(\mu(0), \rho(0)) > (0, 0)$, $C(0) > 0$:

Lemme 3. ρ est croissante car $\partial_t \rho = F_0 \mu \geq 0$. En particulier ρ est positive

Lemme 4. C est décroissante et $\lim\limits_{t \to +\infty} C(t) = 0$

Démonstration. ρ est positive donc C est décroissante.
$(\mu(0), \rho(0)) > (0, 0)$ et $\partial_t \rho = F_0 \mu$ impliquent qu'il existe un t_0 tel que $\rho(t_0) > 0$.
Comme ρ est croissante $\forall t \geq t_0$, $\rho(t) \geq \rho(t_0)$.
Donc $\forall t \geq t_0$, $0 < C(t) = C(0)\exp(-b\int_0^t \rho(s)ds) \leq C_{ste} e^{-b\rho(t_0)t} \underset{t \to +\infty}{\longrightarrow} 0$
Donc $\lim\limits_{t \to +\infty} C(t) = 0$. □

Lemme 5. Si f est croissante et $f(C) = \mathcal{O}_{C\to 0}(C)$ alors μ est bornée.

Démonstration. On a $\partial_t \mu = f(C)(\mu+\rho) - \mu\rho \leq f(C)\mu + f(C)\rho$.
Soit $\phi(t) = f(C(t))\rho(t)$.
Comme $f(C) = \mathcal{O}_{C\to 0}(C)$, C décroît vers 0, f est croissante et $C(t) \leq C_{ste} e^{-b\rho(t_0)t}$: $f(C)$ est intégrable
Comme $f(C) = \mathcal{O}_{C\to 0}(C)$ et C décroît vers 0, $\exists A, T \ \ \forall t > T$, $\phi(t) < AC(t)\rho(t)$.
On a donc $\int_T^{+\infty} \phi(t)\ dt < \int_T^{+\infty} AC\rho\ dt = -\frac{A}{b}\int_T^{+\infty} \partial_t C = \frac{A}{b}C(T)$.
ϕ est donc intégrable et $\partial_t\mu \leq f(C)\mu + \phi$.
Par le lemme de Gronwall :
$\mu(t) \leq \mu(0) + \int_0^t \phi(s)\ ds + \int_0^t \phi(s) f(C)(s) \exp(\int_s^t f(C)(u)du)\ ds$
$\leq \mu(0) + \int_0^{+\infty} \phi(s)\ ds + \int_0^t \phi(s) f(C)(s) \exp(\int_0^{+\infty} f(C)(u)du)\ ds$
$\leq \mu(0) + \int_0^{+\infty} \phi(s)\ ds + \exp(\int_0^{+\infty} f(C)(u)du) \int_0^t \phi(s) f(C)(s)\ ds$
$f(C)$ est bornée et ϕ est intégrable donc $f(C)\phi$ est intégrable.
On a donc : $\mu(t) \leq \mu(0) + \int_0^{+\infty} \phi(s)\ ds + \exp(\int_0^{+\infty} f(C)(u)du) \int_0^{+\infty} \phi(s) f(C)(s)\ ds\ \forall t$ □

Dans la suite on se place dans le cas où f est croissante et $f(C) = \mathcal{O}_{C\to 0}(C)$

Lemme 6. μ décroît à partir d'un certain temps et $\lim\limits_{t\to+\infty} \mu(t) = 0$

Démonstration. On a $\partial_t \mu = f(C)(\mu+\rho) - \mu\rho \leq \mu(f(C) - \rho(t_0)) + f(C)\rho$.
Donc $\forall t > t_0$ $\frac{\partial_t\mu}{\mu} \leq f(C) - \rho(t_0) + \frac{f(C)\rho}{\mu}$.
On a $\lim\limits_{t\to+\infty} f(C) = 0$. De plus C décroît vers 0 donc $\lim\limits_{t\to+\infty} \partial_t C = 0$ donc $\lim\limits_{t\to+\infty} f(C)\rho = 0$.
Ainsi $\exists T$ / $\forall t > T$: $\partial_t \log\mu < 0$ et donc, μ décroît à partir de T.
μ décroit à partir de T et μ est bornée donc μ a une limite ℓ et $\lim\limits_{t\to+\infty} \partial_t\mu = 0$.
On a donc $0 = \lim\limits_{t\to+\infty} f(C)(\mu+\rho) - \mu\rho = \lim\limits_{t\to+\infty} -\mu\rho \leq -l\rho(t_0)$ donc $\ell = 0$. □

Lemme 7. $\lim\limits_{t\to+\infty} \rho(t) = \rho_\infty < +\infty$

Démonstration. On a $\mu(t) = -\int_t^{+\infty} \partial_t\mu\ ds = -\int_t^{+\infty} f(C)(\mu+\rho) - \mu\rho\ ds$.
Or $f(C)$ est intégrable (c.f. preuve du lemme 5)et μ est bornée donc $f(C)\mu$ est intégrable.
De même $f(C)\rho$ est intégrable (c.f. preuve du lemme 5).
On peut donc en conclure que $\mu\rho$ est intégrable.
Or $\mu\rho = F_0\rho\partial_t\rho = \frac{F_0}{2}\partial_t\rho^2$. $t \mapsto \partial_t\rho^2$ est donc intégrable.
Ainsi ρ^2 possède une limite finie et donc ρ aussi. □

3 Recherche de la vitesse d'onde des solutions progressives de l'Équation KPP avec Mémoire

On a le modèle suivant :

$$\begin{cases} \partial_t\mu - \frac{T}{\lambda}\Delta\mu = f(C)(\mu+\rho) - \mu\rho \\ \partial_t\rho = F_0\mu \\ \partial_t C = -b\rho C \end{cases} \tag{11}$$

où $f(0) = 0$ et f est positive. Typiquement, $f(C) = C$:
Dans la suite on pose $K = \frac{T}{\lambda}$
On recherche des solutions en onde plane, on pose s la vitesse d'onde et $\xi = x - st$.

$$\begin{cases} -s\mu' - K\mu'' = f(C)(\mu+\rho) - \mu\rho \\ -s\rho' = F_0\mu \\ C' = \frac{b\rho C}{s} \end{cases} \tag{12}$$

Nos états stationnaires sont $(\mu, \rho, C) = \begin{cases} (0, 0, C_0) \\ (0, \rho_\infty, 0), \rho_\infty > 0 \end{cases}$

3.1 Au voisinage de $(0, 0, C_0)$

Au voisinage de $(0, 0, C_0)$ on a, en posant $f(C_0) = f_0$:

$$\begin{cases} -s\mu' - K\mu'' = f_0(\mu+\rho) \\ -s\rho' = F_0\mu \end{cases} \tag{13}$$

ce qui devient

$$\rho''' + \frac{s}{K}\rho'' + \frac{f_0}{K}\rho' - \frac{F_0 f_0}{Ks}\rho = 0 \tag{14}$$

de polynôme caractéristique

$$P(X) = X^3 + \frac{s}{K}X^2 + \frac{f_0}{K}X - \frac{F_0 f_0}{Ks} \tag{15}$$

Pour $s < 0$, $P(0) > 0$ donc P a une racine négative r_1 .
Pour que P ait deux autres racines réelles $r_3 > r_2 > r_1$ il faut (condition nécessaire et suffisante) que P' s'annule deux fois et que le discriminant Δ de P soit positif.

3.1.1 Première condition : P' a deux annulations :

$P'(X) = 3X^2 + 2\frac{s}{K}X + \frac{f_0}{K}$ a pour discriminant $\Delta' = 4\frac{1}{K^2}(s^2 - 3Kf_0)$ ce qui donne la condition

$$\boxed{s^2 > 3Kf_0} \tag{16}$$

3.1.2 Deuxième condition : $\Delta > 0$:

Pour $P = aX^3 + bX^2 + cX + d$ on a $\Delta = b^2c^2 + 18abcd - 27a^2d^2 - 4ac^3 - 4b^3d$ ce qui dans notre cas donne

$$\begin{aligned} \Delta &= \frac{1}{K^4}f_0^2 s^2 - 18\frac{f_0^2 F_0}{K^3} - 27\frac{F_0^2 f_0^2}{K^2 s^2} - 4\frac{f_0^3}{K^3} + 4\frac{F_0 f_0 s^2}{K^4} \\ &= s^2\frac{f_0(f_0 + 4F_0)}{K^4} - \frac{f_0^2(18F_0 + 4)}{K^3} - \frac{27F_0^2 f_0^2}{K^2}\frac{1}{s^2} \\ &= \frac{f_0}{K^4 s^2}[(f_0 + 4F_0)s^4 - Kf_0(18F_0 + 4)s^2 - 27K^2F_0^2 f_0] \end{aligned}$$

On est revenu à étudier le signe du polynôme en s^2

$$D(s^2) = (f_0 + 4F_0)s^4 - Kf_0(18F_0 + 4)s^2 - 27K^2F_0^2f_0 \tag{17}$$

de discriminant d :

$$\begin{aligned} d &= \Big(Kf_0(18F_0 + 4)\Big)^2 + 108(f_0 + 4F_0)K^2F_0^2f_0 \\ &= K^2f_0(f_0(18F_0 + 4)^2 + 108(f_0 + 4F_0)F_0^2) > 0 \end{aligned}$$

On obtient donc la condition sur la positivité de Δ :

$$\boxed{s^2 > K\frac{f_0(18F_0 + 4) + \sqrt{f_0(f_0(18F_0 + 4)^2 + 108(f_0 + 4F_0)F_0^2)}}{2(f_0 + 4F_0)}} \tag{18}$$

3.1.3 Signe des racines au voisinage de $(0, 0, C_0)$

On sait déjà que $r_3 < 0$. Comme $r_1r_2r_3 < 0$, on remarque que r_2 et r_1 sont du même signe.
De plus P' a un axe de symétrie $X = -\frac{s}{3K} > 0$ car $s < 0$ donc P atteint un minimum local (forcement négatif) en un point positif donc P a une racine positive.
On en déduit $r_1 > r_2 > 0$:
Sous les conditions (16) et (18), P a deux racines positives et une négative.

3.2 Au voisinage de $(0, \rho_\infty, 0)$

Autour de $(0, \rho_\infty, 0)$: Posons $(\mu, \rho, C) = (\mu, \rho_\infty + \epsilon, C)$. On a

$$\begin{cases} -s\mu' - K\mu'' = f(C)\rho_\infty - \mu\rho_\infty \\ C' = \frac{b\rho_\infty C}{s} \\ -s\epsilon' = F_0\mu \end{cases} \tag{19}$$

la deuxième ligne donne

$$C(y) = \Lambda \exp(\frac{b\rho_\infty}{s}y) \tag{20}$$

et la réunion de la première et la deuxième se traduit sur ϵ par :

$$s^2\epsilon'' + Ks\epsilon''' = f(C)F_0\rho_\infty + s\epsilon'\rho_\infty \tag{21}$$

est une EDO d'ordre trois en ϵ avec terme source $\frac{F_0f(C)}{Ks}\rho_\infty$ de polynôme caracteristique :

$$Q(X) = X^3 + \frac{s}{K}X^2 - \frac{\rho_\infty}{K}X \tag{22}$$

qui possède toujours trois racines : 0, une négative et une positive : $X = -\frac{1}{2K}(s \pm \sqrt{s^2 + 4\rho_\infty Ks})$
Sur μ on a :

$$-s\mu' - Ks\mu'' = f(C)\rho_\infty - \mu\rho_\infty \tag{23}$$

Dans le cas $f(C) = C$:
μ a pour polynôme caractéristique homogène $M(X) = X^2 + \frac{1}{K}X - \frac{\rho_\infty}{Ks}$ de racines :
$r_{1,2} = -\frac{1}{2K}(1 \pm \sqrt{1 + 4\frac{\rho_\infty K}{s}})$
donc $\mu_H = Ae^{r_1y} + Be^{r_2y}$ (On choisit $r_1 > r_2$).
En cherchant une solution particulière de la forme $\mu_p = M\exp(\frac{b\rho_\infty}{s}y)$ on obtient $M = -\frac{\Lambda}{b^2\rho_\infty K + b - 1}$
et donc $\mu = Ae^{r_1y} + Be^{r_2y} + Me^{\frac{b\rho_\infty}{s}y}$ et donc $\rho = \rho_\infty + \alpha e^{r_1y} + \beta e^{r_2y} + \Gamma\exp(\frac{b\rho_\infty}{s}y)$ pour $\Gamma = \frac{Ms}{b\rho_\infty}$

4 Schémas Numériques

On a le modèle suivant ("KPP avec mémoire") :

$$\left\{\begin{array}{l}\partial_t \mu = K\Delta\mu + C(\mu+\rho) - \mu\rho \\ \partial_t \rho = F_0 \mu \\ \partial_t C = -b\rho C\end{array}\right. \tag{24}$$

4.1 Pour l'équation différentielle ordinaire

Sans dépendance spatiale :

$$\left\{\begin{array}{l}\partial_t \mu = C(\mu+\rho) - \mu\rho \\ \partial_t \rho = F_0 \mu \\ \partial_t C = -b\rho C\end{array}\right. \tag{25}$$

4.1.1 Schéma semi-implicite I pour l'EDO

Soit le schéma semi-implicite I pour l'EDO :

$$\boxed{\left\{\begin{array}{l}\mu^{n+1} = \mu^n + \Delta t(C^n(\mu^{n+1}+\rho^{n+1}) - \mu^{n+1}\rho^n) \\ \rho^{n+1} = \rho^n + \Delta t(F_0\mu^{n+1}) \\ C^{n+1} = C^n - \Delta t(b\rho^{n+1}C^{n+1})\end{array}\right.} \tag{26}$$

Ce schéma donne :

$$\left\{\begin{array}{l}\mu^{n+1}(1-\Delta t(C^n(1+\Delta t F_0)) + \rho^n) = \mu^n + \Delta t C^n \rho^n \\ \rho^{n+1} = \rho^n + \Delta t(F_0\mu^{n+1}) \\ C^{n+1} = C^n \frac{1}{1+\Delta t b\rho^{n+1}}\end{array}\right.$$

Positivité Pour conserver la positivité il suffit que le terme $(1-\Delta t(C^n(1+\Delta t F_0)) + \rho^n)$ reste positif :
Par exemple :

$$\boxed{C^0 < \frac{1}{\Delta t(1+F_0\Delta t)}} \tag{27}$$

4.1.2 Schéma semi-implicite II pour l'EDO

Soit le schéma semi-implicite II pour l'EDO :

$$\boxed{\left\{\begin{array}{l}\mu^{n+1} = \mu^n + \Delta t(C^n(\mu^{n+1}+\rho^{n+1}) - \mu^n\rho^n) \\ \rho^{n+1} = \rho^n + \Delta t(F_0\mu^{n+1}) \\ C^{n+1} = C^n - \Delta t(b\rho^{n+1}C^{n+1})\end{array}\right.} \tag{28}$$

Ce schéma donne :

$$\left\{\begin{array}{l}\mu^{n+1}(1-\Delta t(C^n(1+\Delta t F_0))) = \mu^n + \Delta t\rho^n(C^n - \mu^n) \\ \rho^{n+1} = \rho^n + \Delta t(F_0\mu^{n+1}) \\ C^{n+1} = C^n \frac{1}{1+\Delta t b\rho^{n+1}}\end{array}\right.$$

Positivité Pour conserver la positivité il suffit que les terme $(1-\Delta t(C^n(1+\Delta t F_0)))$ et $\mu^n + \Delta t \rho^n (C^n - \mu^n)$ restent positif :
Par exemple :

$$\boxed{C^0 < \frac{1}{\Delta t(1+F_0\Delta t)}} \tag{29}$$

et

$$\boxed{\rho^n < \frac{1}{\Delta t}} \tag{30}$$

On obtient une condition de plus que le schéma semi-implicite I.

4.2 Pour l'équation aux dérivées partielles

4.2.1 Schéma semi-implicite I pour l'EDP

Soit le schéma semi-implicite I pour l'EDP :

$$\boxed{\begin{cases} \mu_i^{n+1} = \mu_i^n + K\Delta t \frac{\mu_{i+1}^{n+1} - 2\mu_i^{n+1} + \mu_{i-1}^{n+1}}{\Delta x^2} + \Delta t(C_i^n(\mu_i^{n+1} + \rho_i^{n+1}) - \mu_i^{n+1}\rho_i^n) \\ \rho_i^{n+1} = \rho_i^n + \Delta t(F_0 \mu_i^{n+1}) \\ C_i^{n+1} = C_i^n - \Delta t(b\rho_i^{n+1} C_i^{n+1}) \end{cases}} \tag{31}$$

Ce schéma donne :

$$\begin{cases} (1 + \frac{K\Delta t}{\Delta x^2} A - \Delta t(C^n(1+\Delta t F_0)) + \rho^n)\mu^{n+1} = \mu^n + \Delta t C^n \rho^n \\ \rho^{n+1} = \rho^n + \Delta t(F_0 \mu^{n+1}) \\ C^{n+1} = C^n \frac{1}{1+\Delta t b \rho^{n+1}} \end{cases}$$

où A est la matrice de $-\Delta$:

$$A = \begin{bmatrix} 2 & -1 & & 0 \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ 0 & & -1 & 2 \end{bmatrix} \tag{32}$$

Positivité Afin de préserver la positivité, on obtient la même condition (suffisante) que pour l'EDO :

$$\boxed{C^0 < \frac{1}{\Delta t(1+F_0\Delta t)}} \tag{33}$$

Démonstration. Supposons $\mu^0 > 0$.
Raisonnons par l'absurde et supposons que $n = \min n \mid \exists j \mid \mu_j^{n+1} < 0$ existe. Soit $j = \arg\min \mu_i^{n+1}$.
On a $(1 - \Delta t(C_j^n(1+\Delta t F_0)) + \rho_j^n)\mu_j^{n+1} = \mu^n + \Delta t C^n + \frac{K\Delta t}{\Delta x^2}(\mu_{j+1}^{n+1} - 2\mu_j^{n+1} + \mu_{j-1}^{n+1})$.
Or par définition de n et comme $C^0 < \frac{1}{\Delta t(1+F_0\Delta t)}$ et $C^n < C^0$:
$\mu^n + \Delta t C^n > 0$
$1 - \Delta t(C_j^n(1+\Delta t F_0)) + \rho_j^n > 0$
Et par définition de j :
$\mu_{j+1}^{n+1} - 2\mu_j^{n+1} + \mu_{j-1}^{n+1} \geq 0$
On a donc $\mu_j^{n+1} > 0$ mais $\mu_j^{n+1} = \min(\mu_i^{n+1}) < 0$ par définition de j et n : C'est absurde. $\square$

5 Résolution numérique

5.1 Résolution de l'EDO

5.1.1 Résultat de la simulation de l'EDO

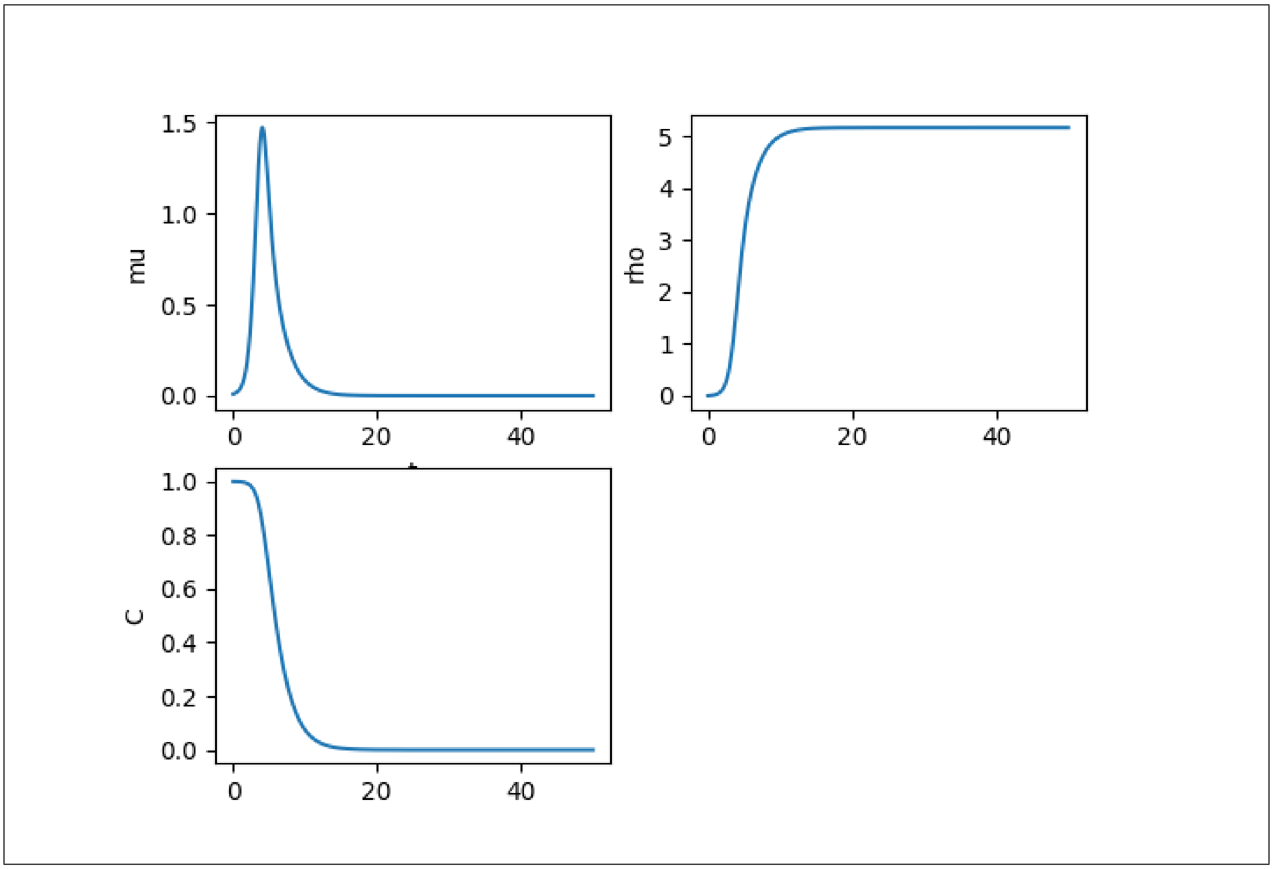

Figure 2 – Résolution du schéma implicite pour l'EDO

On observe les phénomènes attendus sur l'EDO :
-i) μ est bornée et tend vers 0.
-ii) ρ est croissante et bornée.
-iii) C décroît vers 0.

5.2 Résolution de l'EDP en 1D

5.2.1 Résultat de la simulation de l'EDP en 1D

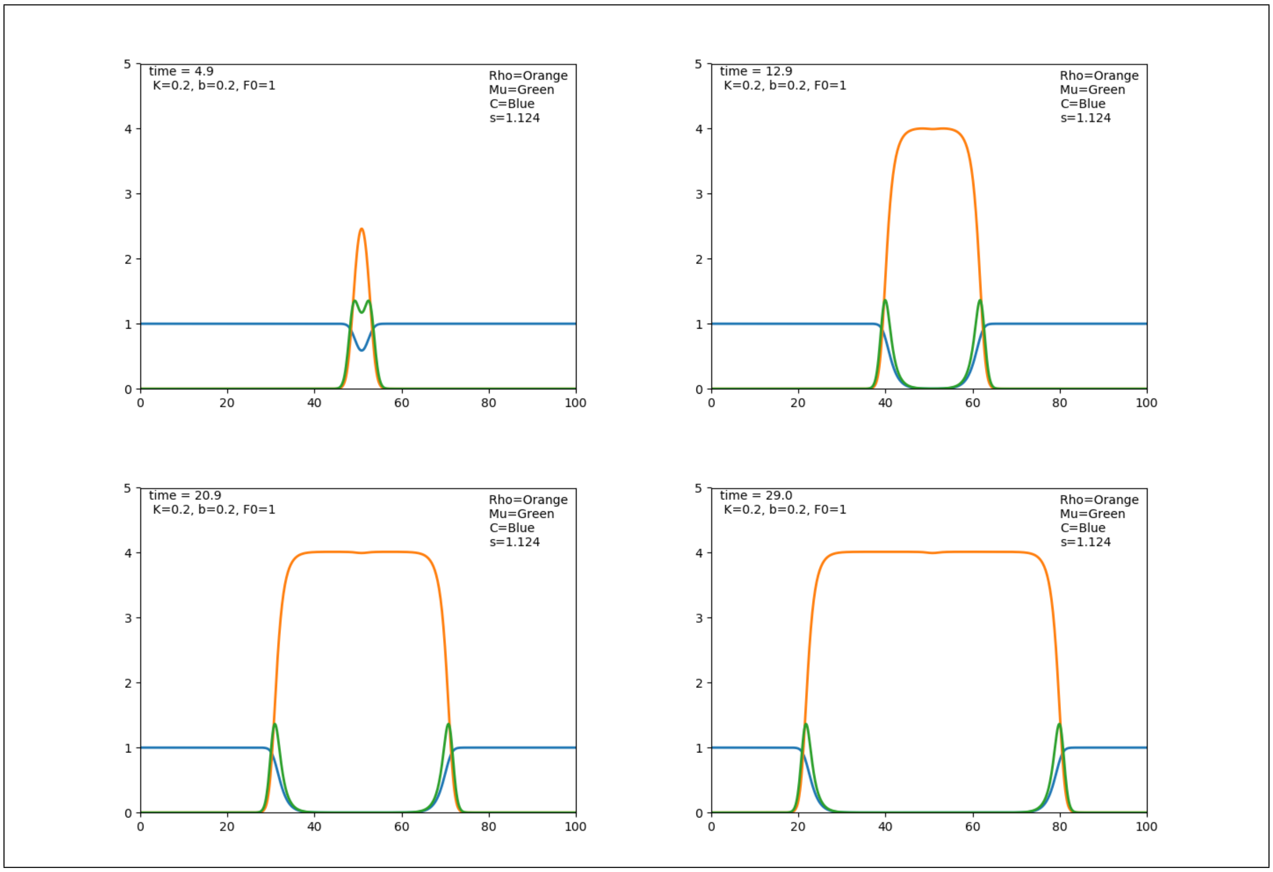

FIGURE 3 – Résolution du schéma semi implicite I pour l'EDP en 1D

On voit sur les simulations que la solution tend vers une solution de type onde plane stationnaire. Il est possible de calculer cette vitesse et de la comparer avec la vitesse théorique minimale obtenue dans la partie 3 :

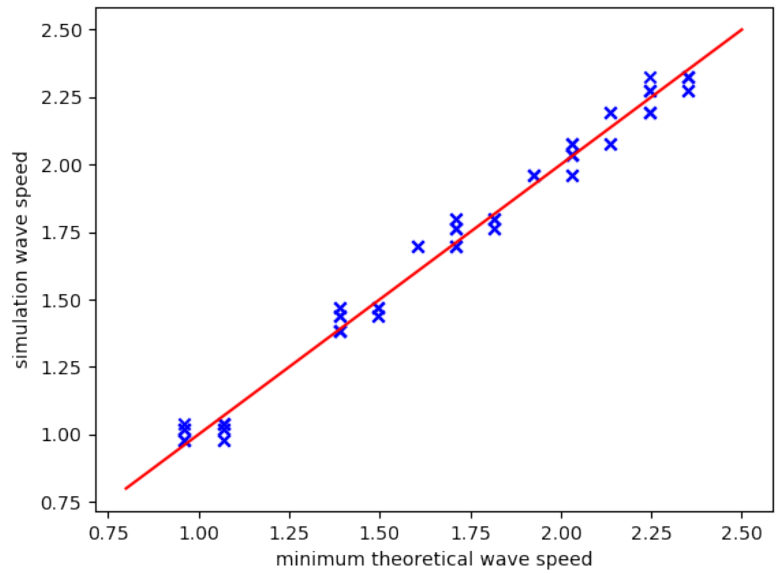

FIGURE 4 – Vitesse du front observée numériquement en fonction de la vitesse minimale théorique

Soit

$$s^*_{theorique} = K\frac{f_0(18F_0+4)+\sqrt{f_0(f_0(18F_0+4)^2+108(f_0+4F_0)F_0^2)}}{2(f_0+4F_0)}$$

la vitesse minimale théorique obtenue dans la partie 3.
Ce graphe représente par les points bleus la vitesse du front observée numériquement pour différentes simulations en fonction de la vitesse minimale théorique associée à cette simulation. La droite rouge est la droite $s_{simu} = s^*_{theorique}$.
On remarque que la vitesse du front observée numériquement est très proche de la vitesse minimale théorique : ce phénomène est similaire à celui de l'équation de Fisher-KPP : pour une donnée initiale à support compact, **le front se propage asymptotiquement à la vitesse minimale de l'équation d'onde associée à l'EDP**.

Appendices

Code de résolution de l'EDO

```
import matplotlib.pyplot as plt
from scipy.integrate import ode
import numpy as np

b=.5 # dtC=-b*rho*C
F0=1 # dtRho = Fo*Mu
tf=50 # temps final de la simulation
rho0=0 #rho initial
mu0=.1 #mu initial
c0=1 #concentration initiale
n=1000 #nombre de pas de temps

#Résolution du schéma éxplicite
def euler_explicite_edo(b, F0, tf, rho0, mu0, c0, n):
  #t0<t1 , temps etudies,
  #rho0, mu0,c0 reels positifs: condition initiale
  #n entier(nombre d'iterations)
  h=tf/n #pas Deltat
  rho=rho0
  mu=mu0
  c=c0
  t=0
  Rho=[rho0]
  Mu=[mu0]
  C=[c0]
  T=[t]
  for k in range(n):
    new_mu = mu + h*(c*(mu+rho)-mu*rho)
    new_rho = rho + h*F0*mu
    new_c = c - h*b*rho*c
    mu=new_mu
    rho=new_rho
    c=new_c
    t=t+h
    Mu.append(new_mu)
    Rho.append(new_rho)
    C.append(new_c)
    T.append(t)
  return T,Mu,Rho,C

#Résolution du schéma semi- implicite I
def euler_semi_I_edo(b, F0, tf, rho0, mu0, c0, n):
  #t0<t1 , temps etudies,
  #rho0, mu0,c0 reels positifs: condition initiale
  #n entier(nombre d'iterations)
  h=tf/n #pas Deltat
  rho=rho0
  mu=mu0
  c=c0
  t=0
  Rho=[rho0]
  Mu=[mu0]
  C=[c0]
  T=[0]
```

```
  for k in range(n):
    new_mu = (mu + h*c*rho)/(1+h*rho-h*c*(1+h*F0))
    new_rho = rho + h*F0*new_mu
    new_c = c/(1 + b*h*new_rho)
    mu=new_mu
    rho=new_rho
    c=new_c
    t=t+h
    Mu.append(new_mu)
    Rho.append(new_rho)
    C.append(new_c)
    T.append(t)
  return T,Mu,Rho,C

#Résolution du schéma semi- implicite II
def euler_semi_II_edo(b, F0, tf, rho0, mu0, c0, n):
  #t0<t1 , temps etudies,
  #rho0, mu0,c0 reels positifs: condition initiale
  #n entier(nombre d'iterations)
  t=0
  h=tf/n #pas Deltat
  rho=rho0
  mu=mu0
  c=c0
  Rho=[rho0]
  Mu=[mu0]
  C=[c0]
  T=[0]
  for k in range(n):
    new_mu = (mu + h*c*rho-h*rho*mu)/(1-h*c*(1+h*F0))
    new_rho = rho + h*F0*new_mu
    new_c = c/(1 + b*h*new_rho)
    mu=new_mu
    rho=new_rho
    c=new_c
    t=t+h
    Mu.append(new_mu)
    Rho.append(new_rho)
    C.append(new_c)
    T.append(t)
  return T,Mu,Rho,C

#Programmation de la méthode de Newton-Raphson
def newton(f,gradf,newton_steps,x0):
  x=x0
  for k in range(newton_steps):
    x=x-f(x)/gradf(x)
  return x
#Résolution du schéma implicite
def euler_implicite_edo(b, F0, tf, rho0, mu0, c0, n):
  #t0<t1 , temps etudiés,
  #rho0, mu0,c0 reels positifs: conditions initiale
  #n entier(nombre d'itérations)
  t=0
  newton_steps=10 #nombre d'itérations de la méthode de Newton-Raphson pour le
    calcul implicite
  h=tf/n #pas deltat
  rho=rho0
  mu=mu0
```

```
    c=c0
    Rho=[rho0]
    Mu=[mu0]
    C=[c0]
    T=[0]
    for k in range(n):
      #Calcul de new_mu par methode de Newton Raphson
      #coefficients du polynome d'ordre 3 en new_mu
      alpha = -h**4*F0**2*b
      beta= -F0*h**2*(b+1+2*rho*b*h)
      gamma= -(1+b*h*rho)+b*h**2*F0*mu+h*(c*(1+h*F0)-rho*(1+b*h*rho))
      delta=(1+b*h*rho)*mu + h*c*rho
      def P(X):
        return alpha*X**3+beta*X**2+gamma*X+delta
      def gradP(X):
        return 3*alpha*X**2+2*beta*X+gamma
      new_mu=newton(P,gradP,newton_steps,mu)
      new_rho = rho + h*F0*new_mu
      new_c = c/(1 + b*h*new_rho)
      mu=new_mu
      rho=new_rho
      c=new_c
      t=t+h
      Mu.append(new_mu)
      Rho.append(new_rho)
      C.append(new_c)
      T.append(t)
    return T,Mu,Rho,C

#Utilisation des libraries python (scipy) pour résoudre l'EDO
def black_box_edo(b, F0, tf, rho0, mu0, c0, n):
      def f(t,y,arg1,arg2):
          mu=y[0]
          rho=y[1]
          c=y[2]
          return [c*(mu+rho)-mu*rho, F0*mu, -b*rho*c]

      r = ode(f).set_integrator('zvode', method='adams')
      r.set_initial_value([mu0,rho0,c0],0).set_f_params(F0,b)
      dt=tf/(n-1)
      Rho=[rho0]
      Mu=[mu0]
      C=[c0]
      t=0
      T=[0]
      while r.t < tf:
          mu,rho,c = r.integrate(r.t+dt)
          Mu.append(mu)
          Rho.append(rho)
          C.append(c)
          T.append(r.t)
      return T,Mu,Rho,C


T,Mu,Rho,C = black_box_edo(b, F0, tf, rho0, mu0, c0, n)
rho_inf_theorique = .5*(rho0+np.sqrt(rho0**2 + 4*F0*(mu0+(c0/b))))
print(rho_inf_theorique)
rho_inf= Rho[n-1]
print(rho_inf)
```

```
A=[np.log(rho_inf-x) for x in Rho]
B=[-b*y*rho_inf+np.log((F0)/(b*rho_inf)) for y in T]
#Tracé des solutions
plt.subplot(221)
plt.plot(T,Mu)
plt.ylabel('mu')
plt.xlabel('t')
plt.subplot(222)
plt.plot(T,Rho)
plt.ylabel('rho')
plt.subplot(223)
plt.plot(T,C)
plt.ylabel('C')
plt.subplot(224)
plt.plot(T,A)
plt.plot(T,B)
plt.ylabel('log(rho_inf-rho), -b*rho_inf*t')
plt.show()
```

edo.py

Code de la résolution de l'EDP en 1D

```
# %load edp_1d.py
import matplotlib.pyplot as plt
import numpy as np
import scipy.sparse as sp
from scipy.sparse.linalg.dsolve import spsolve
import matplotlib.animation as animation

#Coéfficients physiques
K=.5 #coefficient diffusion
b=.5 # dtC=-b*rho*C
F0=.8 # dtRho = Fo*Mu

#Paramêtres numériques
n_t=2001 #nombre de pas de temps
tf=25 # temps final de la simulation
xf = 100 #longeur de la simulation
n_x =500 #nombres de points de la simulation

#Données initiales
rho0=np.zeros(n_x) #rho initial
mu0=np.zeros(n_x) #mu initial
mu0[(n_x//2):(n_x//2 +10)]=.01
c0=np.zeros(n_x)+1 #concentration initiale

def edp_1d_explicite(K, b, F0, rho0, mu0, c0, n_t , tf, xf, n_x):
    dt=tf/(n_t-1)
    dx=xf/(n_x-1)
    X=np.linspace(0,xf,n_x)
    T=np.linspace(0,tf,n_t)
    Mu=np.zeros((n_t,n_x))
    Rho=np.zeros((n_t,n_x))
    C=np.zeros((n_t,n_x))
    Mu[0]=mu0
    Rho[0]=rho0
    C[0]=c0
    #Résolution du schema éxplicite
    for n in range(0,n_t-1):
```

```
        RHS=np.zeros(n_x)
        alpha=-C[n]*dt*(1+dt*F0)+dt*Rho[n]+1
        RHS[1:-1]= dt*((K/(dx**2))*(Mu[n,:-2]-2*Mu[n,1:-1]+Mu[n,2:])+C[n,1:-1]*Rho[n,1:-1])
        RHS[0]= dt*((K/(dx**2))*(-2*Mu[n,0]+Mu[n,1])+C[n,0]*Rho[n,0])
        RHS[-1]=dt*((K/(dx**2))*(-2*Mu[n,-1]+Mu[n,-2])+C[n,-1]*Rho[n,-1])
        Mu[n+1]=(1/alpha)*(Mu[n]+RHS)
        Rho[n+1]=Rho[n]+dt*F0*Mu[n+1]
        C[n+1]=C[n]/(1 + b*dt*Rho[n])
    return X,T,Mu,Rho,C

def edp_1d_semi_implicite_I(K, b, F0, rho0, mu0, c0, n_t , tf, xf, n_x):
    #Détermination des paramêtres numeriques deltat et deltax
    dt=tf/(n_t-1)
    dx=xf/(n_x-1)
    #Représentation de l'éspace et du temps
    X=np.linspace(0,xf,n_x)
    T=np.linspace(0,tf,n_t)
    #Initialisation
    Mu=np.zeros((n_t,n_x))
    Rho=np.zeros((n_t,n_x))
    C=np.zeros((n_t,n_x))
    Mu[0]=mu0
    Rho[0]=rho0
    C[0]=c0
    #Résolution du schéma implicite-explicite I
    for n in range(0,n_t-1):
        #Matrice du Laplacien
        A=np.diag(-np.ones(n_x-1),-1)+np.diag(2*np.ones(n_x),0)+np.diag(-np.ones(n_x-1),1)
        #Laplacien Numerique
        A=A*K*dt/(dx**2)
        #Ajout des termes implicites
        alpha=-C[n]*dt*(1+dt*F0)+dt*Rho[n]+1
        A+=np.diag(alpha,0)
        A=sp.csc_matrix(A)
        #Résolution du systême implicite
        Mu[n+1]= spsolve(A, Mu[n]+dt*C[n]*Rho[n])
        Rho[n+1]=Rho[n]+dt*F0*Mu[n+1]
        C[n+1]=C[n]/(1 + b*dt*Rho[n])
    return X,T,Mu,Rho,C


def edp_1d_semi_implicite_II(K, b, F0, rho0, mu0, c0, n_t , tf, xf, n_x):
    #Détermination des paramêtres numériques deltat et deltax
    dt=tf/(n_t-1)
    dx=xf/(n_x-1)
    #Représentation de l'éspace et du temps
    X=np.linspace(0,xf,n_x)
    T=np.linspace(0,tf,n_t)
    #Initialisation
    Mu=np.zeros((n_t,n_x))
    Rho=np.zeros((n_t,n_x))
    C=np.zeros((n_t,n_x))
    Mu[0]=mu0
    Rho[0]=rho0
    C[0]=c0
    #Résolution du schéma implicite-explicite II
    for n in range(0,n_t-1):
```

```
        #Matrice du Laplacien
        A=np.diag(-np.ones(n_x-1),-1)+np.diag(2*np.ones(n_x),0)+np.diag(-np.ones(n_x-1),1)
        A=A*K*dt/(dx**2) #Laplacien Numerique
        #Ajout des termes implicites
        alpha=-C[n]*dt*(1+dt*F0)+1
        A+=np.diag(alpha,0)
        A= sp.csc_matrix(A)
        #Résolution du système implicite
        Mu[n+1]= spsolve(A, Mu[n]+dt*C[n]*Rho[n]-dt*Mu[n]*Rho[n])
        Rho[n+1]=Rho[n]+dt*F0*Mu[n+1]
        C[n+1]=C[n]/(1 + b*dt*Rho[n])
    return X,T,Mu,Rho,C
 #X,T,Mu,Rho,C= edp_1d_semi_implicite_I(K, b, F0, rho0, mu0, c0, n_t , tf, xf, n_x)



def speed(X,Rho,rho_inf):
    #Position du front
    argmed=np.zeros(n_t)
    for i in range(n_t):
        argmed[i]= X[(n_x//2)+np.min(np.where(np.append(Rho[i,(n_x//2):],[0])<rho_inf/2))]
    #Vitesse du front
    s = ((n_t-1)/tf)*(argmed[(n_t//2)+150]-argmed[(n_t//2)])/(150)
    return s

memory =[]
for i in range(5):
    for j in range(5):
        K = .2 + .2*i
        b = .1 + .1*j
        X,T,Mu,Rho,C= edp_1d_semi_implicite_I(K, b, F0, rho0, mu0, c0, n_t , tf, xf, n_x)
         #Valeur de rho a l'infini
        rho_inf = Rho[n_t-1,(n_x//2)]
        s = speed(X,Rho,rho_inf)
        s_theorique = np.sqrt(K*((18*F0+4)+np.sqrt(((18*F0+4)**2)+108*(1+4*F0)*(F0**2)))/(2*(1+4*F0)))
        memory += [K,b,s,s_theorique]

np.savetxt('memory_data3.dat', memory)

#print('La vitesse de propagation de la simulation est s=',s)
s_theorique = np.sqrt(K*((18*F0+4)+np.sqrt(((18*F0+4)**2)+108*(1+4*F0)*(F0**2)))/(2*(1+4*F0)))
#Attention, ceci est pour C0=1
print('La vitesse théorique de propagation est s_theorique=', s_theorique)

#Animation
fig = plt.figure()

ax = plt.axes(xlim=(0, xf), ylim=(0, rho_inf+1))
line, = ax.plot([], [], lw=2)
line2, = ax.plot([], [], lw=2)
line3, = ax.plot([], [], lw=2)
line4, = ax.plot([], [], lw=2)
time_text = ax.text(0.02, 0.92, '', transform=ax.transAxes)
legend_text = ax.text(0.80, 0.82, '', transform=ax.transAxes)
```

```python

def init():
    line.set_data([], [])
    line2.set_data([], [])
    line3.set_data([], [])
    line4.set_data([], [])
    time_text.set_text('')
    legend_text.set_text('')
    return line,line2,line3,line4, time_text, legend_text


def animate(i):
    line.set_data(X, C[i])
    line2.set_data(X, Rho[i])
    line3.set_data(X, Mu[i])
    line4.set_data(50+((i*s)*tf/(n_t-1)),np.linspace(0,rho_inf+1,10))
    time_text.set_text('time = {0:.1f}\n K={1}, b={2}, F0={3} '.format(T[i],K,b,F0))
    legend_text.set_text('Rho=Orange \nMu=Green \nC=Blue\ns={0:.3f}'.format(s))
    return line,line2, line3,line4, time_text, legend_text


anim = animation.FuncAnimation(fig, animate, init_func=init,
                               frames=(n_t-1), interval=(tf*200)/(n_t-1), blit=True)



#anim.save('EDP_1D.gif',writer='imagemagick', fps=30)
plt.show()
```

edp_1d.py