

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, Laurent Monasse

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

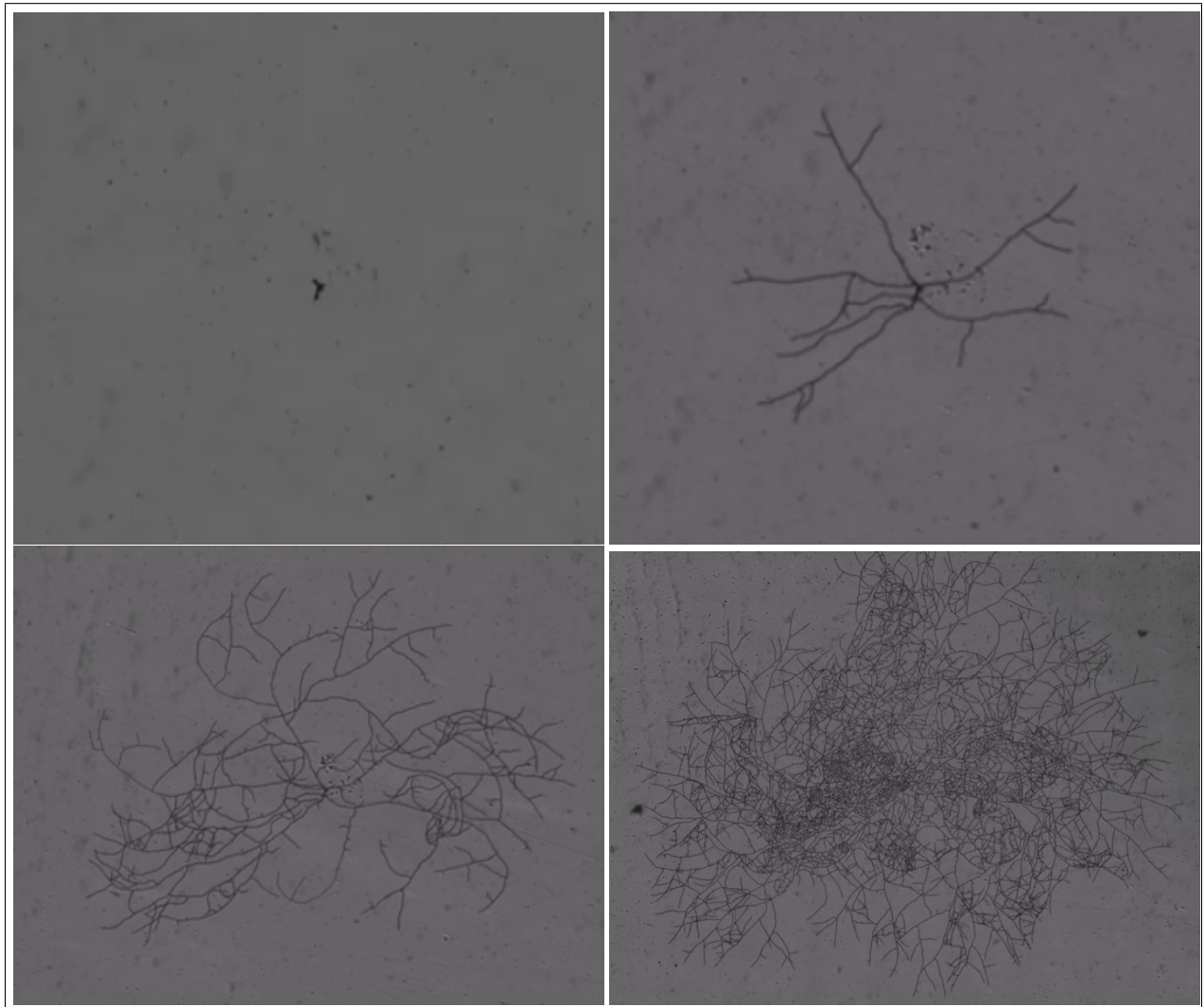


FIGURE 1 – Capture d'un réseau de champignon en expansion, par

Table des matières

1	L'équation de Fisher ou KPP	3
1.1	Preliminaire	3
1.2	Réaction	3
1.3	Réaction-Diffusion	3
1.4	Solutions en onde plane stationnaire / onde progressive	4
2	Dynamique de Réseaux en Croissance	5
2.1	Explication des équations du système (6)	5
2.2	Dérivation de l'équation "KPP avec mémoire"	6
2.3	Propriétés de l'EDO "KPP avec mémoire"	6
3	Recherche de la vitesse d'onde des solutions progressives de l'Équation KPP avec Mémoire	8
3.1	Au voisinage de $(0, 0, C_0)$	8
3.1.1	Première condition : P' a deux annulations :	8
3.1.2	Deuxième condition : $\Delta > 0$:	8
3.1.3	Signe des racines au voisinage de $(0, 0, C_0)$	9
3.2	Au voisinage de $(0, \rho_\infty, 0)$	9
4	Schémas et Positivité	10
4.1	Pour l'équation différentielle ordinaire	10
4.1.1	Schéma semi-implicite I pour l'EDO	10
4.1.2	Schéma semi-implicite II pour l'EDO	10
4.2	Pour l'équation aux dérivées partielles	11
4.2.1	Schéma semi-implicite I pour l'EDP	11
5	Résolution numérique	12
5.1	Résolution de l'EDO	12
5.1.1	Résultat de la simulation de l'EDO	12
5.2	Résolution de l'EDP en 1D	13
5.2.1	Résultat de la simulation de l'EDP en 1D	13
	Appendices	14

1 L'équation de Fisher ou KPP

1.1 Préliminaire

Notre point de départ est l'équation de diffusion :

$$\partial_t u = \Delta u \quad (1)$$

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 :

$$\partial_t u = \Delta u + F(u) \quad (2)$$

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 été introduits et examinés par les travaux de Fisher[1] et Kolmogorov, Petrovsky et Piscounov[2] (abrégés KPP).

Un exemple d'une telle équation est :

$$\partial_t u = \Delta u + u(1 - u) \quad (3)$$

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) :

$$\partial_t v = v - v^2 = F(v) \quad (4)$$

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 $X = C_{b,unif}^0(\mathbb{R}, \mathbb{R})$ 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 existe 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

Rappelons 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 :

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

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. :

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 $h_s : \mathbb{R} \rightarrow]0, 1[$ de classe C^3 .

Cette solution est unique à translation près.

- ii) $\forall c < c_*$ l'équation (5) n'a pas de solution $h : \mathbb{R} \rightarrow [0, 1]$

Remarque : Dans le second cas il existe des solutions 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.

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 :

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

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 représentant 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 :

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

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 à 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 :

$$+K\nabla\mu = -\mu v \quad (8)$$

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

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

dit "KPP avec mémoire".

2.3 Propriétés de l'EDO "KPP avec mémoire"

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

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

avec $f(0) = 0$ On s'intéresse 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 :

$$\mu(t) \geq 0 \forall t \leq t^*$$

$$\partial_t \mu(t^*) \leq 0 \text{ par définition de } t^*. \text{ (Sinon } \mu(t^* + \epsilon) > 0 \forall \epsilon \ll 1)$$

$$\rho(t) > 0 \forall t \leq t^* \text{ car } \partial_t \rho = F_0\mu \text{ et } F_0 > 0$$

$$\partial_t \mu(t^*) = f(C(t^*))\rho(t^*) > 0 \text{ ce qui est en contradiction avec la deuxième affirmation.} \quad \square$$

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 $\lim_{t \rightarrow +\infty} C(t) = 0$

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 $\forall t \geq t_0$, $\rho(t) \geq \rho(t_0)$.

$$\text{Donc } \forall t \geq t_0, 0 < C(t) = C(0) \exp(-b \int_0^t \rho(s) ds) \leq C_{ste} e^{-b\rho(t_0)t} \xrightarrow[t \rightarrow +\infty]{} 0$$

Donc $\lim_{t \rightarrow +\infty} C(t) = 0$. \square

Lemme 5. Si f est croissante et $f(C) = \mathcal{O}_{C \rightarrow 0}(C)$ 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 $f(C) = \mathcal{O}_{C \rightarrow 0}(C)$, C décroît vers 0, f est croissante et $C(t) \leq C_{ste} e^{-b\rho(t_0)t}$: $f(C)$ est intégrable

Comme $f(C) = \mathcal{O}_{C \rightarrow 0}(C)$ 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 :

$$\begin{aligned} \mu(t) &\leq \mu(0) + \int_0^t \phi(s) ds + \int_0^t \phi(s) f(C)(s) \exp\left(\int_s^t f(C)(u) du\right) ds \\ &\leq \mu(0) + \int_0^{+\infty} \phi(s) ds + \int_0^t \phi(s) f(C)(s) \exp\left(\int_0^{+\infty} f(C)(u) du\right) ds \\ &\leq \mu(0) + \int_0^{+\infty} \phi(s) ds + \exp\left(\int_0^{+\infty} f(C)(u) du\right) \int_0^t \phi(s) f(C)(s) ds \\ f(C) &\text{ est bornée et } \phi \text{ est intégrable donc } f(C)\phi \text{ est intégrable.} \end{aligned}$$

On a donc : $\mu(t) \leq \mu(0) + \int_0^{+\infty} \phi(s) ds + \exp\left(\int_0^{+\infty} f(C)(u) du\right) \int_0^{+\infty} \phi(s) f(C)(s) ds \ \forall t$ □

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

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

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 $\lim_{t \rightarrow +\infty} \frac{f(C)\rho}{\mu} = 0$. De plus C décroît vers 0 donc $\lim_{t \rightarrow +\infty} \partial_t C = 0$ donc $\lim_{t \rightarrow +\infty} f(C)\rho = 0$.

Ainsi $\exists T / \forall t > T : \partial_t \log \mu < 0$ et donc, μ décroît à partir de T .

μ décroît à partir de T et μ est bornée donc μ a une limite ℓ et $\lim_{t \rightarrow +\infty} \partial_t \mu = 0$.

On a donc $0 = \lim_{t \rightarrow +\infty} f(C)(\mu + \rho) - \mu\rho = \lim_{t \rightarrow +\infty} -\mu\rho \leq -\ell\rho(t_0)$ donc $\ell = 0$. □

Lemme 7. $\lim_{t \rightarrow +\infty} \rho(t) = \rho_\infty < +\infty$

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 :

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

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 $\xi = x - st$.

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

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

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

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

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

ce qui devient

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

de polynôme caractéristique

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

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

$$\boxed{s^2 > 3Kf_0} \quad (16)$$

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

$$\begin{aligned} \Delta &= \frac{1}{K^4}f_0^2s^2 - 18\frac{f_0^2F_0}{K^3} - 27\frac{F_0^2f_0^2}{K^2s^2} - 4\frac{f_0^3}{K^3} + 4\frac{F_0f_0s^2}{K^4} \\ &= s^2\frac{f_0(f_0 + 4F_0)}{K^4} - \frac{f_0^2(18F_0 + 4)}{K^3} - \frac{27F_0^2f_0^2}{K^2}\frac{1}{s^2} \\ &= \frac{f_0}{K^4s^2}[(f_0 + 4F_0)s^4 - Kf_0(18F_0 + 4)s^2 - 27K^2F_0^2f_0] \end{aligned}$$

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

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

de discriminant d :

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

On obtient donc la condition sur la positivité de Δ :

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

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

On sait déjà que $r_3 < 0$. Comme $r_1 r_2 r_3 < 0$, 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

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

la deuxième ligne donne

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

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

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

est une EDO d'ordre trois en ϵ avec terme source $\frac{F_0 f(C)}{Ks}\rho_\infty$ de polynôme caractéristique :

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

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 :

$$-s\mu' - Ks\mu'' = f(C)\rho_\infty - \mu\rho_\infty \quad (23)$$

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 $\mu_H = Ae^{r_1 y} + Be^{r_2 y}$ (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 $\mu = Ae^{r_1 y} + Be^{r_2 y} + Me^{\frac{b\rho_\infty}{s}y}$ et donc $\rho = \rho_\infty + \alpha e^{r_1 y} + \beta e^{r_2 y} + \Gamma \exp(\frac{b\rho_\infty}{s}y)$ pour $\Gamma = \frac{Ms}{b\rho_\infty}$

4 Schémas et Positivité

On a le modèle suivant (“KPP avec mémoire”) :

$$\begin{cases} \partial_t \mu = K \Delta \mu + C(\mu + \rho) - \mu \rho \\ \partial_t \rho = F_0 \mu \\ \partial_t C = -b \rho C \end{cases} \quad (24)$$

4.1 Pour l'équation différentielle ordinaire

Sans dépendance spatiale :

$$\begin{cases} \partial_t \mu = C(\mu + \rho) - \mu \rho \\ \partial_t \rho = F_0 \mu \\ \partial_t C = -b \rho C \end{cases} \quad (25)$$

4.1.1 Schéma semi-implicite I pour l'EDO

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

$$\boxed{\begin{cases} \mu^{n+1} = \mu^n + \Delta t (C^n(\mu^{n+1} + \rho^{n+1}) - \mu^{n+1} \rho^n) \\ \rho^{n+1} = \rho^n + \Delta t (F_0 \mu^{n+1}) \\ C^{n+1} = C^n - \Delta t (b \rho^{n+1} C^{n+1}) \end{cases}} \quad (26)$$

Ce schéma donne :

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

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 :

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

4.1.2 Schéma semi-implicite II pour l'EDO

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

$$\boxed{\begin{cases} \mu^{n+1} = \mu^n + \Delta t (C^n(\mu^{n+1} + \rho^{n+1}) - \mu^n \rho^n) \\ \rho^{n+1} = \rho^n + \Delta t (F_0 \mu^{n+1}) \\ C^{n+1} = C^n - \Delta t (b \rho^{n+1} C^{n+1}) \end{cases}} \quad (28)$$

Ce schéma donne :

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

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 :

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

et

$$\boxed{\rho^n < \frac{1}{\Delta t}} \quad (30)$$

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 :

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

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 $-\Delta$:

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

A étant symétrique définie positive, afin de préserver la positivité, on obtient la même condition (suffisante) que pour l'EDO :

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

5 Résolution numérique

5.1 Résolution de l'EDO

5.1.1 Résultat de la simulation de l'EDO

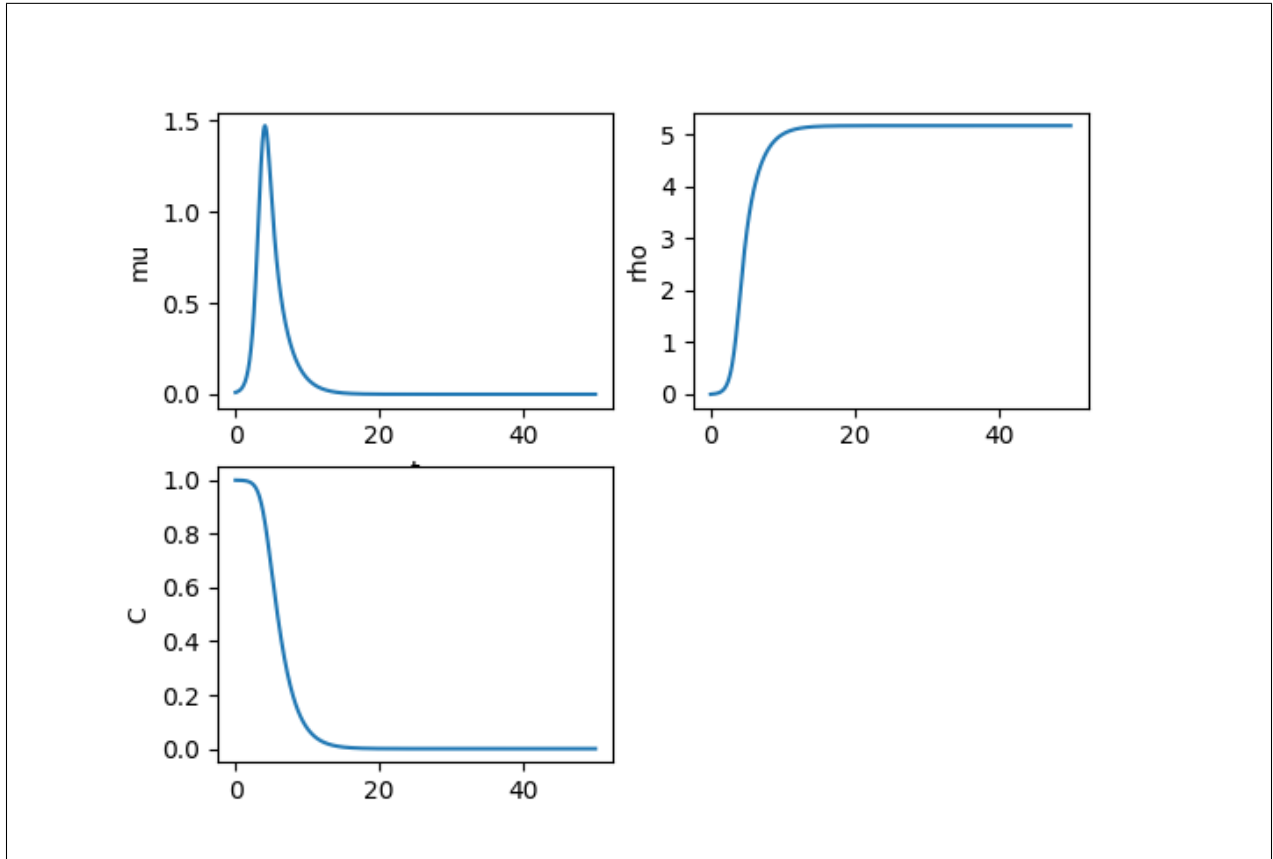


FIGURE 2 – Résolution du schéma implicite pour l'EDO

5.2 Résolution de l'EDP en 1D

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

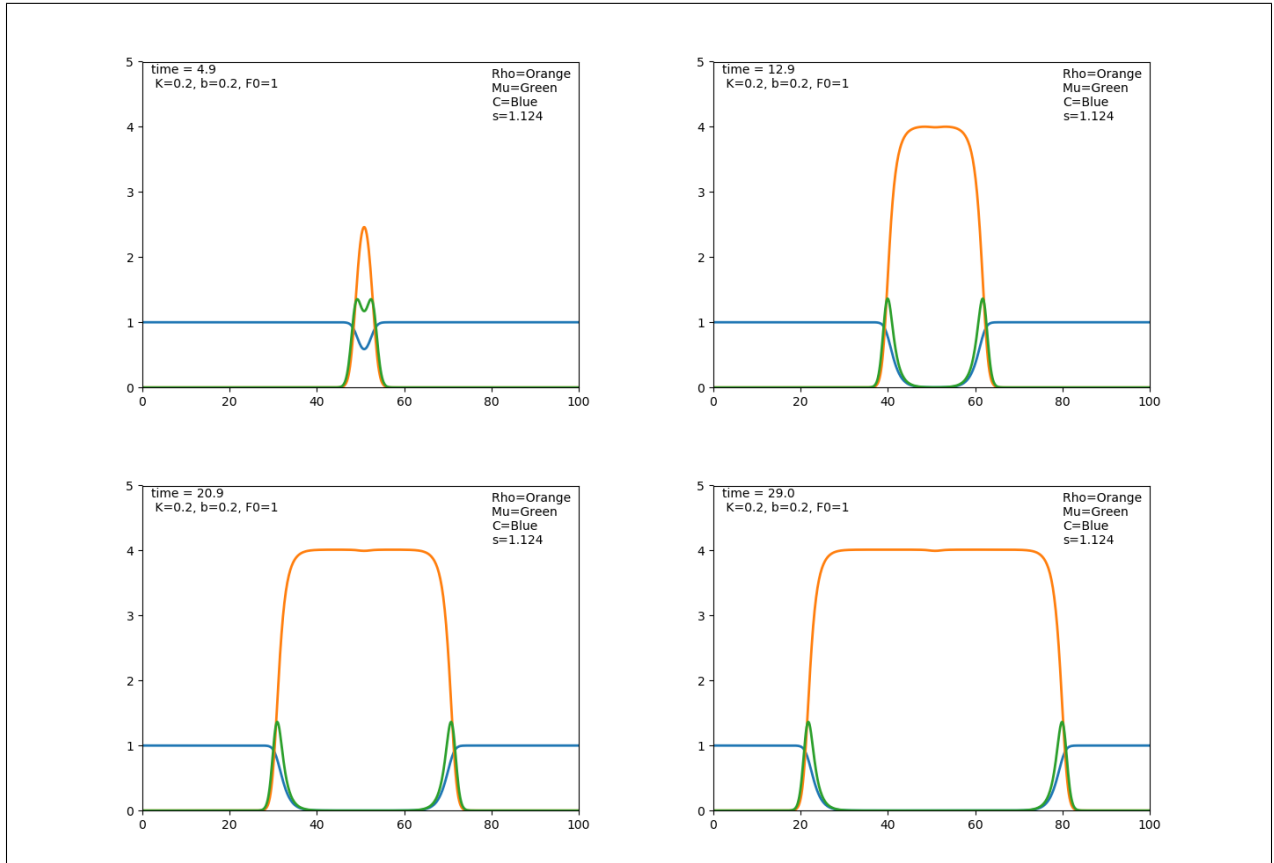


FIGURE 3 – Résolution du schéma semi implicite I pour l'EDP en 1D

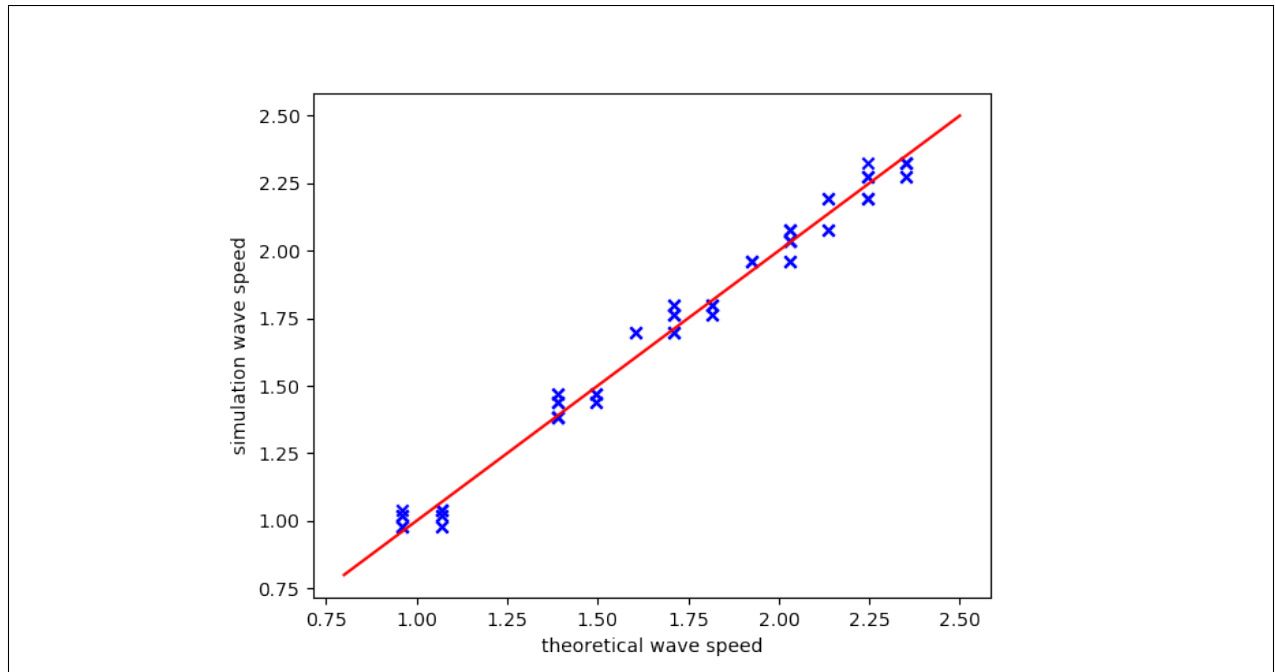


FIGURE 4 – Vitesse d’onde théorique en fonction de la vitesse observée sur la simulation

Appendices

Code de résolution de l’EDO

```

1000 import matplotlib.pyplot as plt
1001 from scipy.integrate import ode
1002 import numpy as np

1004 b=.5 # dtC=-b*rho*C
1005 F0=1 # dtRho = Fo*Mu
1006 tf=50 # temps final de la simulation
1007 rho0=0 #rho initial
1008 mu0=.1 #mu initial
1009 c0=1 #concentration initiale
1010 n=1000 #nombre de pas de temps

1012 #Résolution du schéma explicite
1013 def euler_explicite_edo(b, F0, tf, rho0, mu0, c0, n):
1014     #t0<t1 , temps etudies ,
1015     #rho0, mu0,c0 reels positifs: condition initiale
1016     #n entier(nombre d'iterations)
1017     h=tf/n #pas Deltat
1018     rho=rho0
1019     mu=mu0
1020     c=c0
1021     t=0
1022     Rho=[rho0]
1023     Mu=[mu0]
1024     C=[c0]
1025     T=[t]
1026     for k in range(n):
1027         new_mu = mu + h*(c*(mu+rho)-mu*rho)

```

```

1028     new_rho = rho + h*F0*mu
1029     new_c = c - h*b*rho*c
1030     mu=new_mu
1031     rho=new_rho
1032     c=new_c
1033     t=t+h
1034     Mu.append(new_mu)
1035     Rho.append(new_rho)
1036     C.append(new_c)
1037     T.append(t)
1038     return T,Mu,Rho,C

1040 #Résolution du schéma semi- implicite I
def euler_semi_I_edo(b, F0, tf, rho0, mu0, c0, n):
1042     #t0<t1 , temps etudies ,
1043     #rho0, mu0,c0 reels positifs: condition initiale
1044     #n entier(nombre d'iterations)
1045     h=tf/n #pas Deltat
1046     rho=rho0
1047     mu=mu0
1048     c=c0
1049     t=0
1050     Rho=[rho0]
1051     Mu=[mu0]
1052     C=[c0]
1053     T=[0]
1054     for k in range(n):
1055         new_mu = (mu + h*c*rho)/(1+h*rho-h*c*(1+h*F0))
1056         new_rho = rho + h*F0*new_mu
1057         new_c = c/(1 + b*h*new_rho)
1058         mu=new_mu
1059         rho=new_rho
1060         c=new_c
1061         t=t+h
1062         Mu.append(new_mu)
1063         Rho.append(new_rho)
1064         C.append(new_c)
1065         T.append(t)
1066     return T,Mu,Rho,C

1068 #Résolution du schéma semi- implicite II
def euler_semi_II_edo(b, F0, tf, rho0, mu0, c0, n):
1070     #t0<t1 , temps etudies ,
1071     #rho0, mu0,c0 reels positifs: condition initiale
1072     #n entier(nombre d'iterations)
1073     t=0
1074     h=tf/n #pas Deltat
1075     rho=rho0
1076     mu=mu0
1077     c=c0
1078     Rho=[rho0]
1079     Mu=[mu0]
1080     C=[c0]
1081     T=[0]
1082     for k in range(n):
1083         new_mu = (mu + h*c*rho-h*rho*mu)/(1-h*c*(1+h*F0))
1084         new_rho = rho + h*F0*new_mu
1085         new_c = c/(1 + b*h*new_rho)
1086         mu=new_mu

```

```

1088     rho=new_rho
1090     c=new_c
1092     t=t+h
1094     Mu.append(new_mu)
1096     Rho.append(new_rho)
1098     C.append(new_c)
1100     T.append(t)
1102     return T,Mu,Rho,C

1104 #Programmation de la méthode de Newton-Raphson
1106 def newton(f,gradf,newton_steps,x0):
1108     x=x0
1110     for k in range(newton_steps):
1112         x=x-f(x)/gradf(x)
1114     return x

1116 #Résolution du schéma implicite
1118 def euler_implicite_edo(b, F0, tf, rho0, mu0, c0, n):
1120     #t0<t1 , temps étudiés ,
1122     #rho0, mu0,c0 reels positifs: conditions initiale
1124     #n entier(nombre d'itérations)
1126     t=0
1128     newton_steps=10 #nombre d'itérations de la méthode de Newton-Raphson pour le
1130     calcul implicite
1132     h=tf/n #pas deltat
1134     rho=rho0
1136     mu=mu0
1138     c=c0
1140     Rho=[rho0]
1142     Mu=[mu0]
1144     C=[c0]
1146     T=[0]
1148     for k in range(n):
1150         #Calcul de new_mu par methode de Newton Raphson
1152         #coefficients du polynome d'ordre 3 en new_mu
1154         alpha = -h**4*F0**2*b
1156         beta= -F0*h**2*(b+1+2*rho*b*h)
1158         gamma= -(1+b*h*rho)+b*h**2*F0*mu+h*(c*(1+h*F0)-rho*(1+b*h*rho))
1160         delta=(1+b*h*rho)*mu + h*c*rho
1162         def P(X):
1164             return alpha*X**3+beta*X**2+gamma*X+delta
1166         def gradP(X):
1168             return 3*alpha*X**2+2*beta*X+gamma
1170         new_mu=newton(P,gradP,newton_steps,mu)
1172         new_rho = rho + h*F0*new_mu
1174         new_c = c/(1 + b*h*new_rho)
1176         mu=new_mu
1178         rho=new_rho
1180         c=new_c
1182         t=t+h
1184         Mu.append(new_mu)
1186         Rho.append(new_rho)
1188         C.append(new_c)
1190         T.append(t)
1192     return T,Mu,Rho,C

1194 #Utilisation des libraries python (scipy) pour résoudre l'EDO
1196 def black_box_edo(b, F0, tf, rho0, mu0, c0, n):
1198     def f(t,y,arg1,arg2):
1200         mu=y[0]

```



```

1146     rho=y[1]
1147     c=y[2]
1148     return [c*(mu+rho)-mu*rho, F0*mu, -b*rho*c]

1149
1150     r = ode(f).set_integrator('zvode', method='adams')
1151     r.set_initial_value([mu0, rho0, c0], 0).set_f_params(F0, b)
1152     dt=tf/(n-1)
1153     Rho=[rho0]
1154     Mu=[mu0]
1155     C=[c0]
1156     t=0
1157     T=[0]
1158     while r.t < tf:
1159         mu, rho, c = r.integrate(r.t+dt)
1160         Mu.append(mu)
1161         Rho.append(rho)
1162         C.append(c)
1163         T.append(r.t)
1164     return T, Mu, Rho, C

1165
1166 T, Mu, Rho, C = black_box_edo(b, F0, tf, rho0, mu0, c0, n)
1167 rho_inf_theorique = .5*(rho0+np.sqrt(rho0**2 + 4*F0*(mu0+(c0/b))))
1168 print(rho_inf_theorique)
1169 rho_inf= Rho[n-1]
1170 print(rho_inf)
1171 A=[np.log(rho_inf-x) for x in Rho]
1172 B=[-b*y*rho_inf+np.log((F0)/(b*rho_inf)) for y in T]
1173 #Tracé des solutions
1174 plt.subplot(221)
1175 plt.plot(T, Mu)
1176 plt.ylabel('mu')
1177 plt.xlabel('t')
1178 plt.subplot(222)
1179 plt.plot(T, Rho)
1180 plt.ylabel('rho')
1181 plt.subplot(223)
1182 plt.plot(T, C)
1183 plt.ylabel('C')
1184 plt.subplot(224)
1185 plt.plot(T, A)
1186 plt.plot(T, B)
1187 plt.ylabel('log(rho_inf-rho), -b*rho_inf*t')
1188 plt.show()

```

edo.py

Code de la résolution de l'EDP en 1D

```

1000 # %load edp_1d.py
1001 import matplotlib.pyplot as plt
1002 import numpy as np
1003 import scipy.sparse as sp
1004 from scipy.sparse.linalg.dsolve import spsolve
1005 import matplotlib.animation as animation
1006
1007 #Coefficients physiques
1008 K=.5 #coefficient diffusion
1009 b=.5 # dtC=-b*rho*C
1010 F0=.8 # dtRho = Fo*Mu

```

```

1012 #Paramètres numériques
1013 n_t=2001 #nombre de pas de temps
1014 tf=25 # temps final de la simulation
1015 xf = 100 #longeur de la simulation
1016 n_x =500 #nombres de points de la simulation

1018 #Données initiales
1019 rho0=np.zeros(n_x) #rho initial
1020 mu0=np.zeros(n_x) #mu initial
1021 mu0[(n_x//2):(n_x//2 +10)]=.01
1022 c0=np.zeros(n_x)+1 #concentration initiale

1024 def edp_1d_explicite(K, b, F0, rho0, mu0, c0, n_t , tf, xf, n_x):
1025     dt=tf/(n_t-1)
1026     dx=xf/(n_x-1)
1027     X=np.linspace(0,xf,n_x)
1028     T=np.linspace(0,tf,n_t)
1029     Mu=np.zeros((n_t,n_x))
1030     Rho=np.zeros((n_t,n_x))
1031     C=np.zeros((n_t,n_x))
1032     Mu[0]=mu0
1033     Rho[0]=rho0
1034     C[0]=c0
1035     #Résolution du schema explicite
1036     for n in range(0,n_t-1):
1037         RHS=np.zeros(n_x)
1038         alpha=-C[n]*dt*(1+dt*F0)+dt*Rho[n]+1
1039         RHS[1:-1]= dt*((K/(dx**2))*(Mu[n,:-2]-2*Mu[n,1:-1]+Mu[n,2:])+C[n,1:-1]*Rho[n
1040 ,1:-1])
1041         RHS[0]= dt*((K/(dx**2))*(-2*Mu[n,0]+Mu[n,1])+C[n,0]*Rho[n,0])
1042         RHS[-1]=dt*((K/(dx**2))*(-2*Mu[n,-1]+Mu[n,-2])+C[n,-1]*Rho[n,-1])
1043         Mu[n+1]=(1/alpha)*(Mu[n]+RHS)
1044         Rho[n+1]=Rho[n]+dt*F0*MU[n+1]
1045         C[n+1]=C[n]/(1 + b*dt*Rho[n])
1046     return X,T,Mu,Rho,C

1048 def edp_1d_semi_implicite_I(K, b, F0, rho0, mu0, c0, n_t , tf, xf, n_x):
1049     #Détermination des paramètres numeriques deltat et deltax
1050     dt=tf/(n_t-1)
1051     dx=xf/(n_x-1)
1052     #Représentation de l'espace et du temps
1053     X=np.linspace(0,xf,n_x)
1054     T=np.linspace(0,tf,n_t)
1055     #Initialisation
1056     Mu=np.zeros((n_t,n_x))
1057     Rho=np.zeros((n_t,n_x))
1058     C=np.zeros((n_t,n_x))
1059     Mu[0]=mu0
1060     Rho[0]=rho0
1061     C[0]=c0
1062     #Résolution du schéma implicite-explicite I
1063     for n in range(0,n_t-1):
1064         #Matrice du Laplacien
1065         A=np.diag(-np.ones(n_x-1),-1)+np.diag(2*np.ones(n_x),0)+np.diag(-np.ones(n_x
1066 -1),1)
1067         #Laplacien Numerique
1068         A=A*K*dt/(dx**2)
1069         #Ajout des termes implicites

```

```

1068     alpha=-C[n]*dt*(1+dt*F0)+dt*Rho[n]+1
1069     A+=np.diag(alpha,0)
1070     A=sp.csc_matrix(A)
1071     #Résolution du système implicite
1072     Mu[n+1]= spsolve(A, Mu[n]+dt*C[n]*Rho[n])
1073     Rho[n+1]=Rho[n]+dt*F0*Mu[n+1]
1074     C[n+1]=C[n]/(1 + b*dt*Rho[n])
1075     return X,T,Mu,Rho,C
1076
1077 def edp_1d_semi_implicite_II(K, b, F0, rho0, mu0, c0, n_t , tf, xf, n_x):
1078     #Détermination des paramètres numériques deltat et deltax
1079     dt=tf/(n_t-1)
1080     dx=xf/(n_x-1)
1081     #Représentation de l'espace et du temps
1082     X=np.linspace(0,xf,n_x)
1083     T=np.linspace(0,tf,n_t)
1084     #Initialisation
1085     Mu=np.zeros((n_t,n_x))
1086     Rho=np.zeros((n_t,n_x))
1087     C=np.zeros((n_t,n_x))
1088     Mu[0]=mu0
1089     Rho[0]=rho0
1090     C[0]=c0
1091     #Résolution du schéma implicite-explicite II
1092     for n in range(0,n_t-1):
1093         #Matrice du Laplacien
1094         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)
1095         A=A*K*dt/(dx**2) #Laplacien Numerique
1096         #Ajout des termes implicites
1097         alpha=-C[n]*dt*(1+dt*F0)+1
1098         A+=np.diag(alpha,0)
1099         A= sp.csc_matrix(A)
1100         #Résolution du système implicite
1101         Mu[n+1]= spsolve(A, Mu[n]+dt*C[n]*Rho[n]-dt*Mu[n]*Rho[n])
1102         Rho[n+1]=Rho[n]+dt*F0*Mu[n+1]
1103         C[n+1]=C[n]/(1 + b*dt*Rho[n])
1104     return X,T,Mu,Rho,C
1105     #X,T,Mu,Rho,C= edp_1d_semi_implicite_I(K, b, F0, rho0, mu0, c0, n_t , tf, xf, n_x)
1106
1107
1108
1109
1110 def speed(X,Rho,rho_inf):
1111     #Position du front
1112     argmed=np.zeros(n_t)
1113     for i in range(n_t):
1114         argmed[i]= X[(n_x//2)+np.min(np.where(np.append(Rho[i,(n_x//2):],[0])<
1115         rho_inf/2))]
1116     #Vitesse du front
1117     s = ((n_t-1)/tf)*(argmed[(n_t//2)+150]-argmed[(n_t//2)])/(150)
1118     return s
1119
1120 memory =[]
1121 for i in range(5):
1122     for j in range(5):
1123         K = .2 + .2*i
1124         b = .1 + .1*j
1125         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
1126     rho_inf = Rho[n_t-1,(n_x//2)]
    s = speed(X,Rho,rho_inf)
1128     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]
1130
np.savetxt('memory_data3.dat', memory)
1132
#print('La vitesse de propagation de la simulation est s=',s)
1134 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
1136 print('La vitesse théorique de propagation est s_theorique=', s_theorique)

1138 #Animation
fig = plt.figure()
1140
ax = plt.axes(xlim=(0, xf), ylim=(0, rho_inf+1))
1142 line, = ax.plot([], [], lw=2)
line2, = ax.plot([], [], lw=2)
1144 line3, = ax.plot([], [], lw=2)
line4, = ax.plot([], [], lw=2)
1146 time_text = ax.text(0.02, 0.92, '', transform=ax.transAxes)
legend_text = ax.text(0.80, 0.82, '', transform=ax.transAxes)
1148
def init():
1150     line.set_data([], [])
    line2.set_data([], [])
1152     line3.set_data([], [])
    line4.set_data([], [])
1154     time_text.set_text('')
    legend_text.set_text('')
1156     return line, line2, line3, line4, time_text, legend_text

1158
def animate(i):
1160     line.set_data(X, C[i])
    line2.set_data(X, Rho[i])
1162     line3.set_data(X, Mu[i])
    line4.set_data(50+((i*s)*tf/(n_t-1)), np.linspace(0, rho_inf+1, 10))
1164     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))
1166     return line, line2, line3, line4, time_text, legend_text

1168
anim = animation.FuncAnimation(fig, animate, init_func=init,
1170                             frames=(n_t-1), interval=(tf*200)/(n_t-1), blit=True)

1172
1174 #anim.save('EDP_1D.gif', writer='imagemagick', fps=30)
plt.show()

```

edp_1d.py