

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 Éric Herbert, Gwenaël Ruprich-Robert et Florence Leclerc.

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
1.5	Théorèmes de sélection de la vitesse pour KPP	5
2	Dynamique de Réseaux en Croissance	6
2.1	Explication des équations du système (6)	6
2.2	Dérivation de l'équation "KPP avec mémoire"	7
2.3	Propriétés de l'EDO "KPP avec mémoire"	7
3	Recherche de la vitesse d'onde des solutions progressives de l'Équation KPP avec Mémoire	9
3.1	Au voisinage de $(0, 0, C_0)$	9
3.1.1	Première condition : P' a deux annulations :	9
3.1.2	Deuxième condition : $\Delta > 0$:	9
3.1.3	Signe des racines au voisinage de $(0, 0, C_0)$	10
3.2	Au voisinage de $(0, \rho_\infty, 0)$	10
4	Schémas Numériques	11
4.1	Pour l'équation différentielle ordinaire	11
4.1.1	Schéma semi-implicite I pour l'EDO	11
4.1.2	Schéma semi-implicite II pour l'EDO	11
4.2	Pour l'équation aux dérivées partielles	12
4.2.1	Schéma semi-implicite I pour l'EDP	12
5	Résolution numérique	13
5.1	Résolution de l'EDO	13
5.1.1	Résultat de la simulation de l'EDO	13
5.1.2	Observations de la simulation de l'EDO	13
5.2	Résolution de l'EDP en 1D	15
5.2.1	Résultat de la simulation de l'EDP en 1D	15
	Appendices	17

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 (abrégés KPP).

Un exemple d'une telle équation est :

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

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) :

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

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

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 Piscounov 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 $u_0 \rightarrow]0, 1[$ 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 $t \rightarrow \infty$, u converge uniformément en x vers une solution d'onde h_{s^*} de (5) qui se déplace à vitesse minimale $s^* = 2$:

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

où $m(t) = 2t - (3/2) \log(t) + y_0$.

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

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 $u_0 \rightarrow]0, 1[$ vérifie $\liminf_{x \rightarrow -\infty} u_0(x) > 0$ 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

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

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 $x \rightarrow +\infty$ 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 :

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

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 :

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

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

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

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

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 $\int_0^1 \frac{f(x)}{x} dx < \infty$ alors μ est bornée.

Démonstration. On a $\partial_t \mu = f(C)(\mu + \rho) - \mu\rho \leq f(C)\mu + f(C)\rho$.

Montrons que $f(C)$ est intégrable :

$C(t) \leq C_{ste} e^{-b\rho(t_0)t}$ et f est croissante donc $\int_0^\infty f(C) dt \leq \int_0^\infty f(C_{ste} e^{-b\rho(t_0)t}) dt$.

Soit le changement de variable $u = C_{ste} e^{-b\rho(t_0)t}$, $du = -b\rho(t_0)u dt$:

$\int_0^\infty f(C_{ste} e^{-b\rho(t_0)t}) dt = \frac{1}{b\rho(t_0)} \int_0^1 \frac{f(u)}{u} du < \infty$ car $\int_0^{C_{ste}} \frac{f(x)}{x} dx < \infty$ donc $f(C)$ est intégrable.

Montrons que $\phi = f(C)\rho$ est intégrable :

Effectuons le changement de variable $u = C$, $du = -b\rho u dt$ dans $\int_0^\infty f(C)\rho dt$:

$\int_0^\infty f(C)\rho dt = \frac{1}{b\rho(t_0)} \int_0^{C_{ste}} \frac{f(u)}{u} du < \infty$ car $\int_0^{C_{ste}} \frac{f(x)}{x} dx < \infty$ donc $\phi = f(C)\rho$ est intégrable.

On a $\partial_t \mu \leq f(C)\mu + \phi$.

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

Dans la suite on se place dans le cas où f est croissante et $\int_0^1 \frac{f(x)}{x} dx < \infty$

Lemme 6. $\lim_{t \rightarrow +\infty} \mu = 0$ et $\lim_{t \rightarrow +\infty} \rho = \rho_\infty < +\infty$

Démonstration. μ est bornée, soit μ_n une suite extraite de la fonction μ qui tend vers ℓ .

On a $\ell - \mu(t) = \lim_{n \rightarrow +\infty} \int_t^{t_n} \partial_t \mu ds = \lim_{n \rightarrow +\infty} \int_t^{t_n} f(C)(\mu + \rho) - \mu\rho ds$.

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 a donc $\lim_{n \rightarrow +\infty} \int_t^{t_n} \mu\rho = \int_t^{+\infty} f(C)\mu + f(C)\rho dt + \ell - \mu(t)$

Or $\mu\rho = F_0 \rho \partial_t \rho = \frac{F_0}{2} \partial_t \rho^2$ donc $\int_t^{t_n} \mu\rho dt = \frac{F_0}{2} (\rho(t_n)^2 - \rho(t)^2)$.

Or ρ est croissante donc a une limite dans $[0, +\infty]$.

Ainsi ℓ est déterminée entièrement par la limite de ρ et ne dépend pas de la suite extraite.

Par critère séquentiel μ a une limite ℓ qui est finie car μ est bornée.

Mais alors $\lim_{t \rightarrow +\infty} \frac{F_0}{2} (\rho(t)^2 - \rho(T)^2) = \int_T^\infty (f(C)\mu + f(C)\rho) dt + \ell - \mu(T) < \infty$.

Donc ρ^2 a une limite finie et donc ρ aussi.

Comme $\mu = \frac{\partial_t \rho}{F_0}$ et ρ a une limite finie et μ aussi, μ tend nécessairement vers 0. □

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

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

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

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

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 Numériques

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 :

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

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 :

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

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 :

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

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 :

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

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 $-\Delta$:

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

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

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

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

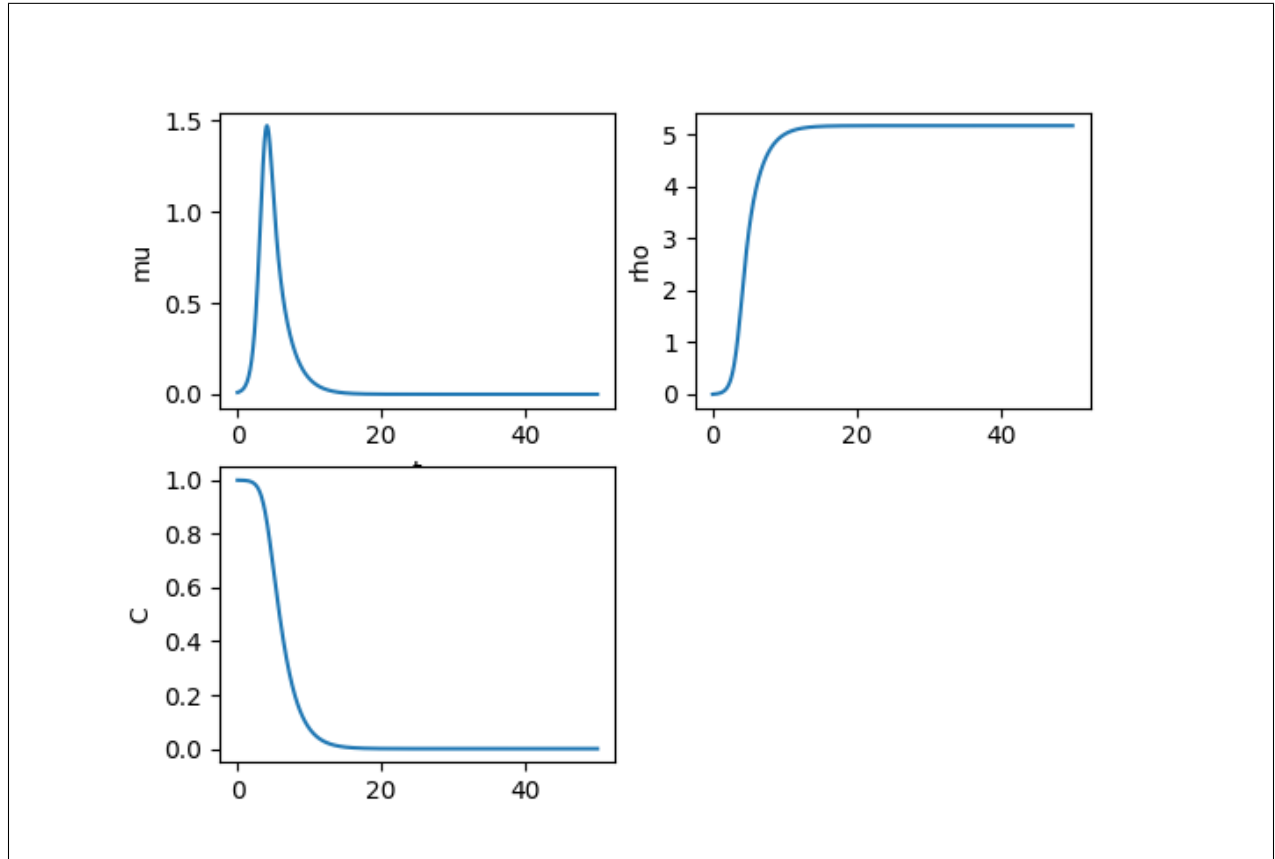


FIGURE 2 – Résolution du schéma implicite pour l'EDO

5.1.2 Observations de la simulation de 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.

-iv) Les solutions ont un comportement exponentiel autour des états stationnaires et ce comportement est bien prédit par la linéarisation de l'EDO autour de ces états :

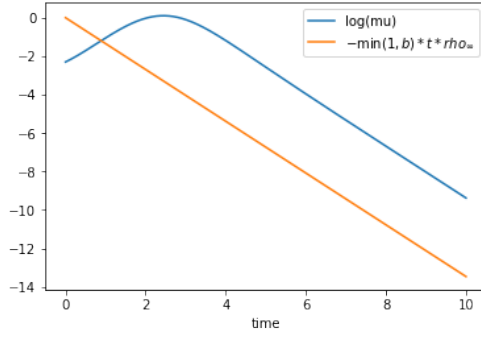


FIGURE 3 – Comportement de $\log(\mu)$, en particulier autour de $(\mu, \rho, C) = (0, \rho_\infty, 0)$. $\log(\mu)$ est bien linéaire autour des états stationnaires et sa pente (le facteur dans l'exponentielle) correspond exactement à $-\min(1, b)\rho_\infty$ ce qui est un résultat obtenu dans la partie Linéarisation

-v) L'ordre de convergence de l'EDO observé est de 1 :

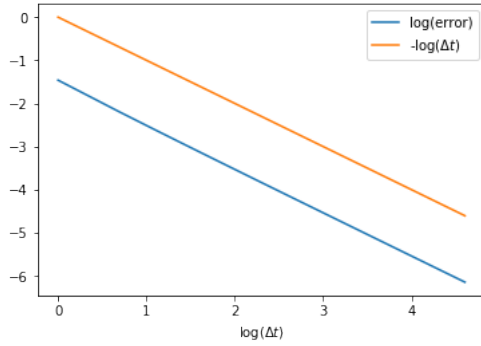


FIGURE 4 – Comportement du $\log(\text{erreur})$ quand $\Delta t \rightarrow 0$. La solution du schéma semi implicite I est comparée à différents pas de temps Δt à la solution numérique du schéma de Runge-Kutta avec $\Delta t = 2 * 10^{-7}$. On observe bien que la convergence est d'ordre 1.

5.2 Résolution de l'EDP en 1D

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

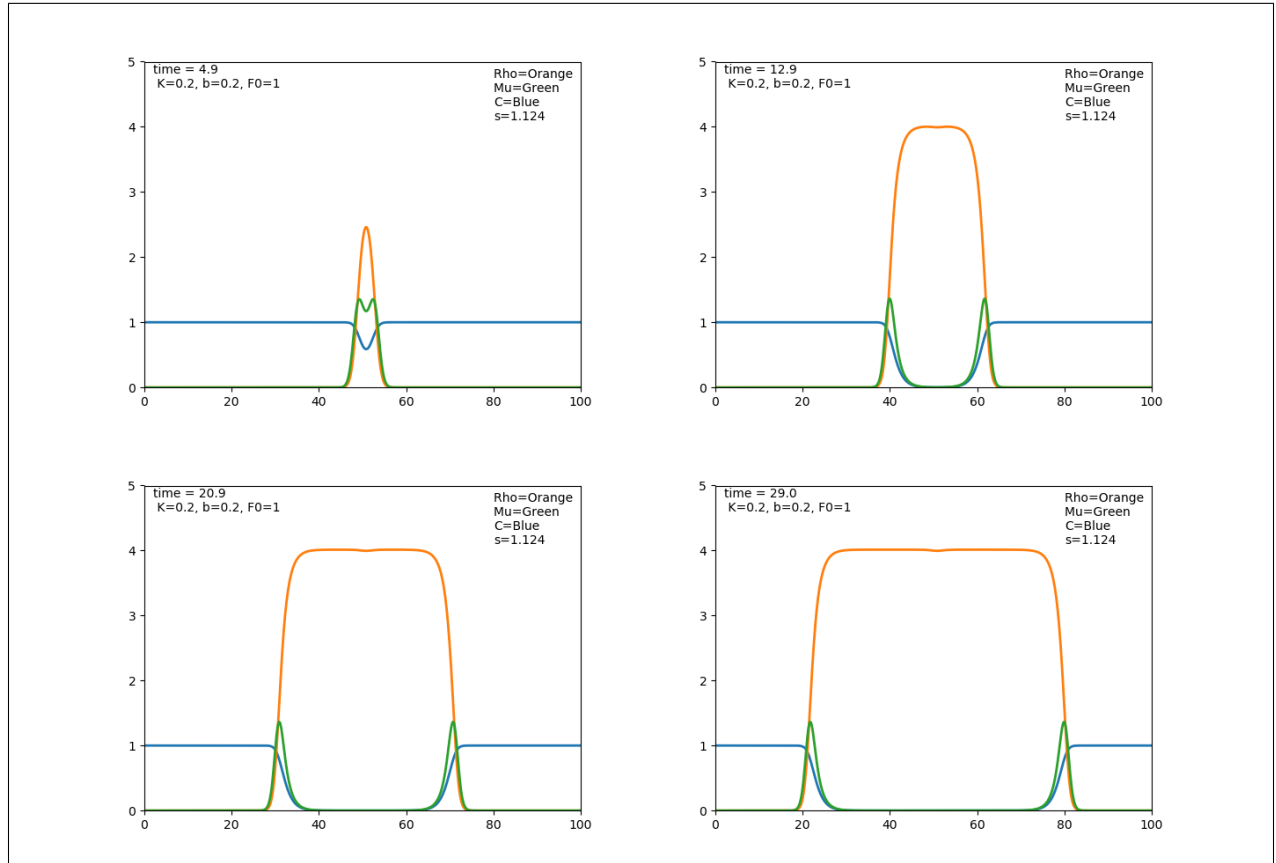


FIGURE 5 – Résolution du schéma semi implicite I pour l'EDP en 1D

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 :

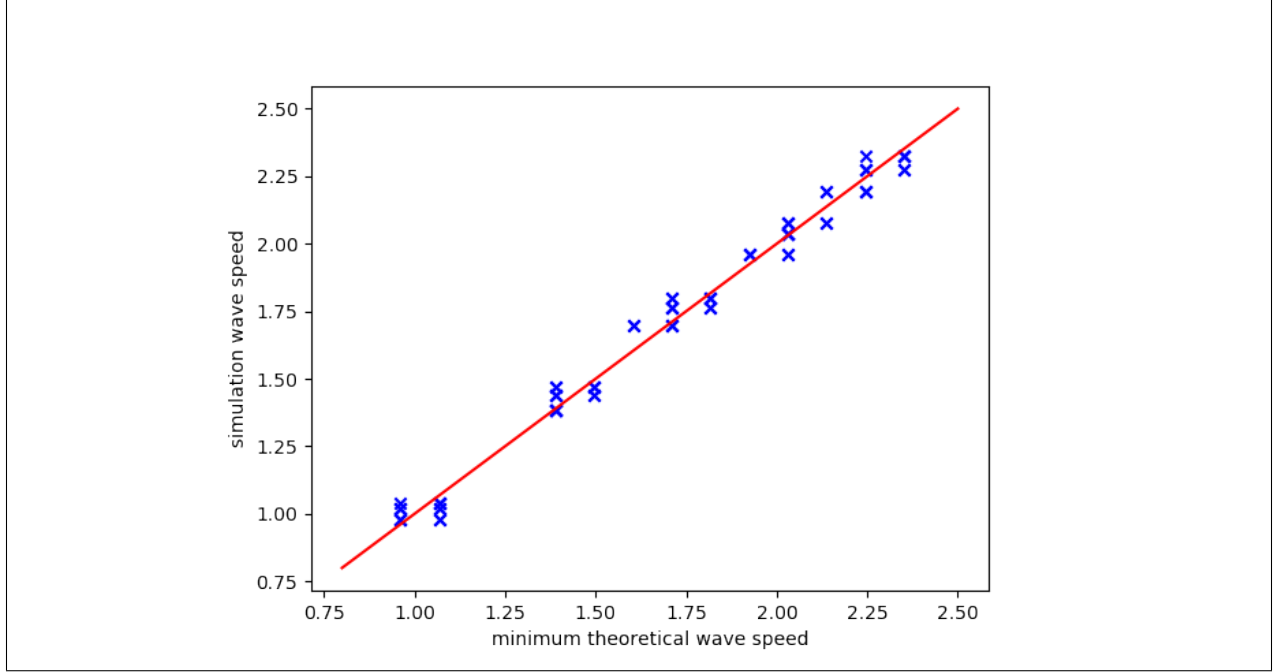


FIGURE 6 – Vitesse du front observée numériquement en fonction de la vitesse minimale théorique

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

```
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=20 # temps final de la simulation
1007 rho0=0 #rho initial
1008 mu0=.1 #mu initial
1009 c0=1 #concentration initiale
1010 n=2000 #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
1041 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

1067 #Résolution du schéma semi- implicite II
1068 def euler_semi_II_edo(b, F0, tf, rho0, mu0, c0, n):
1069     #t0<t1 , temps etudies ,
1070     #rho0, mu0,c0 reels positifs: condition initiale
1071     #n entier(nombre d'iterations)
1072     t=0
1073     h=tf/n #pas Deltat
1074     rho=rho0
1075     mu=mu0
1076     c=c0
1077     Rho=[rho0]
1078     Mu=[mu0]
1079     C=[c0]
1080     T=[0]
1081     for k in range(n):
1082         new_mu = (mu + h*c*rho-h*rho*mu)/(1-h*c*(1+h*F0))
1083         new_rho = rho + h*F0*new_mu
1084         new_c = c/(1 + b*h*new_rho)
1085         mu=new_mu
1086         rho=new_rho
1087         c=new_c
1088         t=t+h
1089         Mu.append(new_mu)
1090         Rho.append(new_rho)
1091         C.append(new_c)
1092         T.append(t)
1093     return T,Mu,Rho,C

1094 #Programmation de la méthode de Newton-Raphson
1095 def newton(f, gradf, newton_steps, x0):
1096     x=x0
1097     for k in range(newton_steps):
1098         x=x-f(x)/gradf(x)
1099     return x

1100 #Résolution du schéma implicite
1101 def euler_implicite_edo(b, F0, tf, rho0, mu0, c0, n):
1102     #t0<t1 , temps étudiés ,
1103     #rho0, mu0,c0 reels positifs: conditions initiale
1104     #n entier(nombre d'itérations)
1105     t=0
1106     newton_steps=10 #nombre d'itérations de la méthode de Newton-Raphson pour le
1107         calcul implicite
1108     h=tf/n #pas deltat
1109     rho=rho0

```

```

1112 mu=mu0
1113 c=c0
1114 Rho=[rho0]
1115 Mu=[mu0]
1116 C=[c0]
1117 T=[0]
1118 for k in range(n):
1119     #Calcul de new_mu par methode de Newton Raphson
1120     #coefficients du polynome d'ordre 3 en new_mu
1121     alpha = -h**4*F0**2*b
1122     beta= -F0*h**2*(b+1+2*rho*b*h)
1123     gamma= -(1+b*h*rho)+b*h**2*F0*mu+h*(c*(1+h*F0)-rho*(1+b*h*rho))
1124     delta=(1+b*h*rho)*mu + h*c*rho
1125     def P(X):
1126         return alpha*X**3+beta*X**2+gamma*X+delta
1127     def gradP(X):
1128         return 3*alpha*X**2+2*beta*X+gamma
1129     new_mu=newton(P,gradP,newton_steps,mu)
1130     new_rho = rho + h*F0*new_mu
1131     new_c = c/(1 + b*h*new_rho)
1132     mu=new_mu
1133     rho=new_rho
1134     c=new_c
1135     t=t+h
1136     Mu.append(new_mu)
1137     Rho.append(new_rho)
1138     C.append(new_c)
1139     T.append(t)
1140 return T,Mu,Rho,C

1141 #Utilisation des libraries python (scipy) pour résoudre l'EDO
1142 def black_box_edo(b, F0, tf, rho0, mu0, c0, n):
1143     def f(t,y,arg1,arg2):
1144         mu=y[0]
1145         rho=y[1]
1146         c=y[2]
1147         return [c*(rho+mu)-mu*rho, F0*mu, -b*rho*c]

1148     r = ode(f).set_integrator('zvode', method='adams')
1149     r.set_initial_value([mu0,rho0,c0],0).set_f_params(F0,b)
1150     dt=tf/(n-1)
1151     Rho=[rho0]
1152     Mu=[mu0]
1153     C=[c0]
1154     t=0
1155     T=[0]
1156     while r.t < tf:
1157         mu,rho,c = r.integrate(r.t+dt)
1158         Mu.append(mu)
1159         Rho.append(rho)
1160         C.append(c)
1161         T.append(r.t)
1162     return T,Mu,Rho,C

1163 #Résolution
1164 T,Mu,Rho,C = euler_semi_I_edo(b, F0, tf, rho0, mu0, c0, n)
1165
1166 rho_inf= Rho[n-1]

```

```

#Étude Asymptotique
1172 A=[np.log(mu) for mu in Mu]
B=[-min(1,b)*y*rho_inf for y in T]
1174
#Tracé des solutions et de l'étude asymptotique
1176 plt.subplot(221)
plt.plot(T,Mu)
1178 plt.ylabel('mu')
plt.xlabel('t')
1180 plt.subplot(222)
plt.plot(T,Rho)
1182 plt.ylabel('rho')
plt.subplot(223)
1184 plt.plot(T,C)
plt.ylabel('C')
1186 plt.subplot(224)
plt.plot(T,A)
1188 plt.plot(T,B)
plt.ylabel('log(mu), -b*rho_inf*t')
1190 plt.show()

```

edo.py

Code de la résolution de l'EDP en 1D

```

1000 # %load edp_1d.py
import matplotlib.pyplot as plt
1002 import numpy as np
import scipy.sparse as sp
1004 from scipy.sparse.linalg.dsolve import spsolve
import matplotlib.animation as animation
1006
#Coefficients physiques
1008 K=.5 #coefficient diffusion
b=.2 # dtC=-b*rho*C
1010 F0= 1 # dtRho = Fo*Mu

#Paramètres numériques
1012 n_t=3001 #nombre de pas de temps
1014 tf=20 # temps final de la simulation
xf = 150 #longueur de la simulation
1016 n_x =601 #nombres de points de la simulation

#Données initiales
1018 rho0=np.zeros(n_x) #rho initial
1020 mu0=np.zeros(n_x) #mu initial
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):
    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))
    1032 Mu[0]=mu0
    Rho[0]=rho0
    1034 C[0]=c0

```

```

1036 #Résolution du schema explicite
1037 for n in range(0,n_t-1):
1038     RHS=np.zeros(n_x)
1039     alpha=-C[n]*dt*(1+dt*F0)+dt*Rho[n]+1
1040     RHS[1:-1]= dt*((K/(dx**2))*(Mu[n,:-2]-2*Mu[n,1:-1]+Mu[n,2:])+C[n,1:-1]*Rho[n
1041 ,1:-1])
1042     RHS[0]= dt*((K/(dx**2))*(-2*Mu[n,0]+Mu[n,1])+C[n,0]*Rho[n,0])
1043     RHS[-1]=dt*((K/(dx**2))*(-2*Mu[n,-1]+Mu[n,-2])+C[n,-1]*Rho[n,-1])
1044     Mu[n+1]=(1/alpha)*(Mu[n]+RHS)
1045     Rho[n+1]=Rho[n]+dt*F0*Mu[n+1]
1046     C[n+1]=C[n]/(1 + b*dt*Rho[n+1])
1047 return X,T,Mu,Rho,C
1048
1049 def edp_1d_semi_implicite_I(K, b, F0, rho0, mu0, c0, n_t , tf, xf, n_x):
1050     #Détermination des paramètres numériques deltat et deltax
1051     dt=tf/(n_t-1)
1052     dx=xf/(n_x-1)
1053     #Représentation de l'espace et du temps
1054     X=np.linspace(0,xf,n_x)
1055     T=np.linspace(0,tf,n_t)
1056     #Initialisation
1057     Mu=np.zeros((n_t,n_x))
1058     Rho=np.zeros((n_t,n_x))
1059     C=np.zeros((n_t,n_x))
1060     Mu[0]=mu0
1061     Rho[0]=rho0
1062     C[0]=c0
1063     #Résolution du schéma implicite-explicite I
1064     for n in range(0,n_t-1):
1065         #Matrice du Laplacien
1066         A=np.diag(-np.ones(n_x-1),-1)+np.diag(2*np.ones(n_x),0)+np.diag(-np.ones(n_x
1067 -1),1)
1068         #Laplacien Numerique
1069         A=A*K*dt/(dx**2)
1070         #Ajout des termes implicites
1071         alpha=-C[n]*dt*(1+dt*F0)+dt*Rho[n]+1
1072         A+=np.diag(alpha,0)
1073         A=sp.csc_matrix(A)
1074         #Résolution du système implicite
1075         Mu[n+1]= spsolve(A, Mu[n]+dt*C[n]*Rho[n])
1076         Rho[n+1]=Rho[n]+dt*F0*Mu[n+1]
1077         C[n+1]=C[n]/(1 + b*dt*Rho[n+1])
1078     return X,T,Mu,Rho,C
1079
1080 def edp_1d_semi_implicite_II(K, b, F0, rho0, mu0, c0, n_t , tf, xf, n_x):
1081     #Détermination des paramètres numériques deltat et deltax
1082     dt=tf/(n_t-1)
1083     dx=xf/(n_x-1)
1084     #Représentation de l'espace et du temps
1085     X=np.linspace(0,xf,n_x)
1086     T=np.linspace(0,tf,n_t)
1087     #Initialisation
1088     Mu=np.zeros((n_t,n_x))
1089     Rho=np.zeros((n_t,n_x))
1090     C=np.zeros((n_t,n_x))
1091     Mu[0]=mu0
1092     Rho[0]=rho0
1093     C[0]=c0

```

```

1092 #Résolution du schéma implicite-explicite II
1093 for n in range(0,n_t-1):
1094     #Matrice du Laplacien
1095     A=np.diag(-np.ones(n_x-1),-1)+np.diag(2*np.ones(n_x),0)+np.diag(-np.ones(n_x
1096     -1),1)
1097     A=A*K*dt/(dx**2) #Laplacien Numerique
1098     #Ajout des termes implicites
1099     alpha=-C[n]*dt*(1+dt*F0)+1
1100     A+=np.diag(alpha,0)
1101     A= sp.csc_matrix(A)
1102     #Résolution du système implicite
1103     Mu[n+1]= spsolve(A, Mu[n]+dt*C[n]*Rho[n]-dt*Mu[n]*Rho[n])
1104     Rho[n+1]=Rho[n]+dt*F0*Mu[n+1]
1105     C[n+1]=C[n]/(1 + b*dt*Rho[n+1])
1106     return X,T,Mu,Rho,C
1107 #X,T,Mu,Rho,C= edp_1d_semi_implicite_I(K, b, F0, rho0, mu0, c0, n_t , tf, xf, n_x)
1108 X,T,Mu,Rho,C= edp_1d_semi_implicite_I(K, b, F0, rho0, mu0, c0, n_t , tf, xf, n_x)
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)+ \
1115             np.min(np.where(np.append(Rho[i,(n_x//2):],[0])<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 rho_inf = Rho[n_t-1,(n_x//2)]
1121 s = speed(X,Rho,rho_inf)
1122 print('La vitesse de propagation de la simulation est s=',s)
1123
1124
1125
1126 # Comparaison de s theorique et numerique pour plusieurs données initiales
1127 # ~ memory =[]
1128 # ~ for i in range(5):
1129     # ~ for j in range(5):
1130         # ~ K = .2 + .2*i
1131         # ~ b = .1 + .1*j
1132         # ~ X,T,Mu,Rho,C= edp_1d_semi_implicite_I(K, b, F0, rho0, mu0, c0, n_t , tf,
1133         xf, n_x)
1134         # ~ #Valeur de rho a l'infini
1135         # ~ rho_inf = Rho[n_t-1,(n_x//2)]
1136         # ~ s = speed(X,Rho,rho_inf)
1137         # ~ s_theorique = np.sqrt(K*((18*F0+4)+np.sqrt(((18*F0+4)**2)+108*(1+4*F0)*(
1138         F0**2)))/(2*(1+4*F0)))
1139         # ~ memory += [K,b,s,s_theorique]
1140 # ~ np.savetxt('memory_data3.dat', memory)
1141
1142 s_theorique = np.sqrt(K*((18*F0+4)+np.sqrt(((18*F0+4)**2)+108*(1+4*F0)*(F0**2)))/
1143     /(2*(1+4*F0)))
1144 #Attention, ceci est pour C0=1
1145 print('La vitesse théorique de propagation est s_theorique=', s_theorique)
1146
1147 #Animation
1148 fig = plt.figure()

```

```

1148 ax = plt.axes(xlim=(0, xf), ylim=(0, rho_inf+1))
      line , = ax.plot([], [], lw=2)
1150 line2 , = ax.plot([], [], lw=2)
      line3 , = ax.plot([], [], lw=2)
1152 line4 , = ax.plot([], [], lw=2)
      time_text = ax.text(0.02, 0.92, '', transform=ax.transAxes)
1154 legend_text = ax.text(0.80, 0.82, '', transform=ax.transAxes)

1156 def init():
      line.set_data([], [])
1158      line2.set_data([], [])
      line3.set_data([], [])
1160      line4.set_data([], [])
      time_text.set_text('')
1162      legend_text.set_text('')
      return line, line2, line3, line4, time_text, legend_text
1164

1166 def animate(i):
      line.set_data(X, C[i])
1168      line2.set_data(X, Rho[i])
      line3.set_data(X, Mu[i])
1170      #line4.set_data(xf/2+((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))
1172      legend_text.set_text('Rho=Orange \nMu=Green \nC=Blue\ns={0:.3f}'.format(s))
      return line, line2, line3, line4, time_text, legend_text
1174

1176 anim = animation.FuncAnimation(fig, animate, init_func=init,
                                frames=(n_t-1), interval=(tf*200)/(n_t-1), blit=True)
1178

1180 #anim.save('EDP_1D.gif', writer='imagemagick', fps=30)
1182 plt.show()

```

edp_1d.py