SIMULATION DE LA DYNAMIQUE DES ONDES CHIMIQUES MULTI-ÉCHELLES PAR ALGORITHME PARARÉEL ET SÉPARATION D'OPÉRATEUR.

Auteur :

Max DUARTE

Encadrants :

Marc MASSOT, Professeur ECP
Frédérique LAURENT-NÈGRE, Chercheur CNRS
Stephane de CHAISEMARTIN, Doctorant au Laboratoire EM2C
en collaboration avec V. LOUVET et T. DUMONT,
Institut Camille Jordan, Univ. Claude Bernard Lyon I

10 septembre 2008

Résumé.

Des domaines aussi divers que la combustion, la modélisation de l'environnement, la dynamique des populations utilisent couramment de modèles de réaction-diffusion impliquant un large spectre d'échelles de temps. La résolution numérique complète de ces derniers par des méthodes classiques est souvent très coûteuse en temps et en ressources de calcul. Pour pallier à ce problème, les méthodes de séparation d'opérateur, en anglais *splitting*, sont couramment utilisées et vastement étudiées ; néanmoins, la récente introduction de la parallélisation en temps constitue un autre outil très prometteur. Alors, ce projet a comme objectif principal faire une étude précise et détaillée de l'utilisation de ces méthodes, en particulier de l'algorithme pararéel, appliquées à la simulation de problèmes d'évolution multi-échelles, en particulier, des ondes chimiques non linéaires. On présente alors la base théorique associée aux modèles d'étude ainsi que les méthodes utilisées : splitting et algorithme pararéel ; on essaie d'évaluer l'influence du caractère multi-échelle sur la performance de ces méthodes à partir des résultats numériques et des considérations théoriques.

Mots clés : *ondes chimiques multi-échelles, séparation d'opérateur, parallélisation en temps, algorithme pararéel.*

Table des matières

Résumé. **i**

Table des matières. **iii**

Généralités. **vii**
Plan. . . . vii
Remerciements. . . . vii

Introduction **1**

1 Modélisation des réactions chimiques non linéaires. **3**
1.1 Modèle d'une réaction chimique auto-catalytique. . . . 3
1.1.1 Présentation du modèle. . . . 3
1.1.2 Mise en équation. . . . 4
1.1.3 Solution analytique. . . . 4
1.1.4 Étude dans le plan de phase. . . . 6
1.1.5 Étude du caractère multi échelle du problème. . . . 6
1.2 Modélisation de la réaction Belousov-Zhabotinskii. . . . 7
1.2.1 Présentation du modèle. . . . 7
1.2.2 Mise en équation. . . . 7
1.2.3 Bifurcation et étude dans le plan de phase. . . . 8
1.2.4 Système de réaction - diffusion. . . . 12

2 Méthodes numériques de résolution. **13**
2.1 La méthode des lignes (MOL). . . . 13
2.1.1 Principe. . . . 13
2.1.2 Application de la méthode. . . . 13
2.2 Le solveur LSODE. . . . 14
2.2.1 La méthode BDF. . . . 15
2.2.2 Formulation canonique du problème. . . . 15
2.2.3 Schémas predicteur-correcteur. . . . 15
2.2.4 Méthode de Newton-Raphson. . . . 16
2.3 Méthodes de Runge-Kutta implicites. . . . 16
2.3.1 Le programme RADAU5. . . . 17
2.3.2 Reformulation du système non linéaire. . . . 17
2.3.3 Itérations de Newton simplifiées. . . . 18
2.3.4 Méthodes de Runge-Kutta diagonalement implicites. . . . 19
2.3.5 Méthodes du type Rosenbrock. . . . 19

2.4 Le solveur SEULEX. 20

3 Séparation d'operateur et parallélisation temporelle. 23
3.1 Les méthodes de splitting. 23
3.1.1 Approche générale. 24
3.1.2 Méthode de Strang. 24
3.2 Splitting de systèmes d'équations de réaction-diffusion raides. 25
3.2.1 Analyse de perturbation singulière. 25
3.2.2 Séparation d'operateur. 26
3.2.3 Estimation de l'ordre de l'erreur locale. 28
3.3 Parallélisation temporelle. 29
3.3.1 Principe général. 29
3.3.2 L'algorithme pararéel. 30
3.3.3 Analyse de convergence de l'algorithme pararéel. 30
3.3.4 L'algorithme pararéel appliqué aux systèmes raides. 32

4 Résultats. 35
4.1 L'Oregonator. 35
4.2 Modèle KPP. 38
4.2.1 Résolution quasi-exacte du problème 38
4.2.2 Résolution du problème en appliquant l'algorithme pararéel. 39
4.2.3 Évolution itérative de l'erreur. 44
4.2.4 Évolution itérative de l'algorithme vers la solution fine. 45
4.2.5 Évolution itérative de l'erreur L^2 et de la vitesse. 49
4.2.6 Convergence. 49
4.3 KPP raide. 51
4.3.1 Évolution itérative de l'erreur. 55
4.3.2 Évolution itérative de l'algorithme vers la solution fine. 58
4.3.3 Évolution itérative de l'erreur L^2 et de la vitesse. 62
4.3.4 Convergence. 63
4.4 Modèle BZ. Premier cas. 64
4.4.1 Résolution quasi-exacte du problème. 64
4.4.2 Résolution du problème en appliquant l'algorithme pararéel. 65
4.4.3 Analyse de convergence. 71
4.5 Modèle BZ. Deuxième cas. 73
4.5.1 Résolution du problème en appliquant l'algorithme pararéel. Première approche. 73
4.5.2 Analyse de convergence. 76
4.5.3 Évolution itérative de l'algorithme vers la solution fine. 77
4.5.4 Évolution itérative de l'erreur L^2 81
4.5.5 Résolution du problème en appliquant l'algorithme pararéel. Deuxième approche. 83
4.5.6 Analyse de convergence. 84
4.5.7 Évolution itérative de l'algorithme vers la solution fine. 87
4.5.8 Évolution itérative de l'erreur L^2. 91
4.6 Résolution du système complet BZ. 93
4.7 Environnement parallèle et temps de calcul. 96
4.7.1 Estimation du temps de calcul. 97

Conclusions. **103**

Bibliographie. **105**

A Évolution dans le plan de phase. **107**
A.1 Modèle KPP. 107
A.2 KPP raide. 109

Généralités.

Plan.

Tout d'abord, on fera une introduction générale sur la parallélisation en temps, le contexte de l'étude et ses objectifs principaux.

Dans une première partie, on présentera les modèles d'étude pris en compte, lesquelles se basent sur la cinétique chimique non linéaire. Le caractère multi-échelle de ces modèles est mis en relief.

Ensuite, on détaillera brièvement les méthodes numériques que nous avons utilisées pour la résolution des systèmes d'équations différentielles dits raides.

Par la suite, la base théorique des méthodes de séparation d'opérateur et de parallélisation temporelle sera introduite ainsi que les aspects liés aux systèmes d'équations de réaction-diffusion raides non linéaires.

Dans une dernière partie, on présentera et analysera les différents résultats atteints, pour finalement donner les conclusions à lesquelles nous sommes arrivés à la fin de ce projet.

Remerciements.

Je souhaite adresser mon remerciement à Marc Massot pour le temps qu'il m'a consacré, pour ses conseils et remarques précises.

Je voudrais également remercier à Stephane de Chaisemartin et Frédérique Laurent-Nègre, qui m'ont beaucoup aidé au long de ce projet.

Finalement, je remercie également Thierry Dumont et Violaine Louvet, pour leur collaboration et pour m'avoir bien accueilli pendant mon séjour à Lyon, ainsi que Stéphane Descombes, pour ses idées et remarques sur la partie théorique du projet.

Ils m'ont beaucoup apporté pendant ce projet et ce fut un plaisir de travailler avec eux.

Introduction.

Dans nombre d'applications industrielles ainsi que scientifiques, la prise en compte des mécanismes détaillés des différents phénomènes devient cruciale. Grâce aux progrès de la puissance de calcul des ordinateurs actuels et de la modélisation, les simulations numériques sont largement utilisées pour prédire précisément ces comportements. Cependant, tenir compte de tous ces aspects dans les simulations numériques est un challenge qui requiert des études fondamentales pour mettre en place des nouvelles méthodes numériques.

Le coût important de simulation de ces niveaux détaillés de modélisation demande un investissement important dans les méthodes numériques utilisées afin de parvenir à un fort niveau d'optimisation. Une voie concerne l'optimisation des techniques numériques sur architecture parallèle. Alors que la parallélisation issue d'une décomposition de domaine spatial est actuellement très utilisée (on découpe le domaine de calcul en sous domaines et on simule les sous domaines sur des processeurs différents), la voie de la parallélisation en temps en est encore à ses débuts. Elle consiste à combiner un solveur temporel grossier, mais rapide, et un solveur fin relativement beaucoup plus lent (car beaucoup plus précis). Pour cela on utilise la puissance de calcul parallèle via un système d'itérations permettant de converger à partir d'une résolution grossière vers le détail de la dynamique du système.

Il s'agit dans ce projet de mettre en oeuvre ce type de technique dans des configurations simples et d'en étudier l'intérêt et l'efficacité pour les applications proposées.

Chapitre 1

Modélisation des réactions chimiques non linéaires.

Dans cette première section, on va commencer par présenter les différents modèles sur lesquelles on s'est appuyés dans le cadre de ce projet. Ces modèles font partie des études qui portent sur la cinétique chimique non linéaire.

Plusieurs études ont été faites pour caractériser ces types des réactions. Dans la majorité des cas, ils s'appuient sur les modèles plus ou moins complexes, qui cherchent à reproduire et surtout à expliquer leurs comportements à partir des observations expérimentales.

Par la suite, on se concentrera sur deux modèles en particulier, l'un basé sur l'oxydation catalysée d'espèces organiques, et l'autre, sur une réaction chimique auto-catalytique.

1.1 Modèle d'une réaction chimique auto-catalytique.

Dans ce paragraphe, on souhaite caractériser l'évolution temporelle d'une réaction chimique démarrée par la présence d'un auto-catalyseur. On verra que cette évolution implique à la fin, la propagation d'une onde progressive. Il s'agit aussi d'un phénomène dissipatif puisqu'aucun paramètre en particulier ne se conserve.

1.1.1 Présentation du modèle.

D'abord, on prendra en main un modèle déjà bien étudié et défini [6; 1]. On considère un milieu réactif contenu dans un grand tube (cela permet de supposer que le milieu est unidimensionnel et de dimension infinie), pouvant contenir deux substances A et B, dont on notera les concentrations respectivement a et b.

Pour cette étude, on ne considérera qu'un processus auto-catalytique cubique comme le montre le schéma de réaction suivant :

$$A + 2B \rightarrow 3B.$$

La cinétique est décrite par la constante de réaction k. La vitesse de cette réaction est donc kab^2. A et B diffusent également dans le milieu. On suppose qu'il s'agit de substances ayant un même coefficient de diffusion D.

1.1.2 Mise en équation.

Le modèle est finalement représenté mathématiquement par un système d'équations aux dérivées partielles, où l'évolution temporelle des concentrations s'écrit comme la somme des contributions de la diffusion et de la réaction,

$$\frac{\partial a}{\partial t} = D\frac{\partial^2 a}{\partial r^2} - kab^2 \tag{1.1}$$

$$\frac{\partial b}{\partial t} = D\frac{\partial^2 b}{\partial r^2} + kab^2. \tag{1.2}$$

Avec des conditions aux limites au bord du tube,

$$a = 0 \quad b = a_0 \quad \text{pour } r \to -\infty \quad \text{et} \quad a = a_0 \quad b = 0 \quad \text{pour } r \to +\infty. \tag{1.3}$$

En sommant l'équation (1.1) el la (1.2), on obtient,

$$\frac{\partial (a+b)}{\partial t} = D\frac{\partial^2 (a+b)}{\partial r^2}. \tag{1.4}$$

Maintenant, si l'on ajoute aux conditions aux limites une condition initiale telle que $a(0,r)+b(0,r) = a_0$ pour tout $r \in \mathbb{R}$, la solution de (1.4) est alors la constante :

$$a + b = a_0 \qquad \forall (t,r).$$

En remplaçant a dans l'équation (1.2), en divisant par a_0 et en posant $\beta = \frac{b}{a_0}$, on obtient :

$$\frac{\partial \beta}{\partial t} = D\frac{\partial^2 \beta}{\partial r^2} + k\beta^2(1-\beta). \tag{1.5}$$

Cette équation est connue sous le nom *équation de Kolmogorov-Petrovskii-Piskunov* (KPP) [6; 9].

1.1.3 Solution analytique.

On peut résoudre l'expression (1.5) analytiquement en faisant d'abord un adimensionnement temporelle et spatiale,

$$\frac{\partial \beta}{\partial \tau} = \frac{\partial^2 \beta}{\partial x^2} + \beta^2(1-\beta), \tag{1.6}$$

avec les valeurs adimensionnées $\tau = kt$ et $x = (k/D)^{1/2}r$. Pour déterminer l'équation que vérifie le profil de l'onde, on cherche une solution sous la forme d'une onde progressive de vitesse constante c . On pose pour cela $\beta(x,\tau) = \beta(z)$ avec $z = x - c\tau$. On trouve alors l'équation suivante.

$$\frac{d^2\beta}{dz^2} + c\frac{d\beta}{dz} + \beta^2(1-\beta) = 0, \tag{1.7}$$

avec les conditions aux limites

$$\lim_{z\to-\infty} \beta = 1 \qquad \lim_{z\to+\infty} \beta = 0. \tag{1.8}$$

Il s'agit finalement d'une équation différentielle ordinaire, que l'on résout en remarquant qu'une solution doit être comprise entre 0 et 1 (pour éviter des concentrations négatives), que

le gradient de concentration doit certainement tendre vers zéro à l'infini (car les concentrations tendent vers des valeurs finies), et que par conséquence le gradient doit être négatif pour toute valeur finie de z. Une forme simple de β qui satisfait toutes ces conditions est l'équation parabolique suivante

$$\frac{d\beta}{dz} = -c\beta(1-\beta). \tag{1.9}$$

Un calcul permet alors de déterminer la vitesse c :

$$c = \frac{1}{\sqrt{2}}, \tag{1.10}$$

ainsi que la solution exacte en onde progressive :

$$\beta(z) = \frac{\exp\left(-\frac{1}{\sqrt{2}}(z-z_0)\right)}{1+\exp\left(-\frac{1}{\sqrt{2}}(z-z_0)\right)}. \tag{1.11}$$

En revenant aux donnés initiales, on en déduit la vitesse d'onde constante v et la solution exacte :

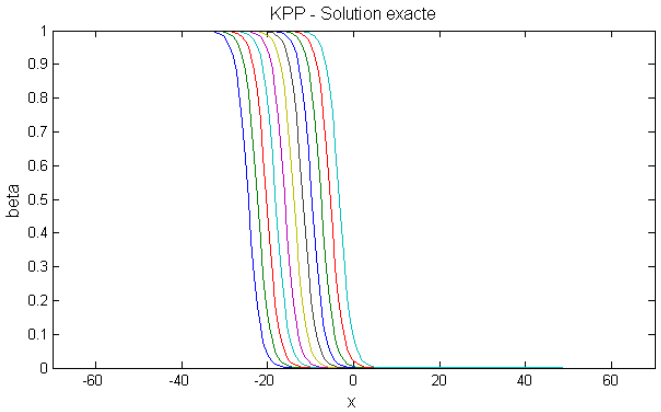
$$v = \frac{1}{\sqrt{2}}(kD)^{\frac{1}{2}} \tag{1.12}$$

$$\beta(r,t) = \frac{\exp\left(-\frac{1}{\sqrt{2}}(\frac{k}{D})^{\frac{1}{2}}(r-r_0-vt)\right)}{1+\exp\left(-\frac{1}{\sqrt{2}}(\frac{k}{D})^{\frac{1}{2}}(r-r_0-vt)\right)}. \tag{1.13}$$

A partir de cette expression, un calcul montre que la pente la plus forte de la solution exacte s'écrit

$$\left(\frac{\partial\beta}{\partial r}\right)_{\max} = -\frac{1}{\sqrt{32}}\left(\frac{k}{D}\right)^{\frac{1}{2}}. \tag{1.14}$$

FIG. 1.1: Solution exacte, KPP.



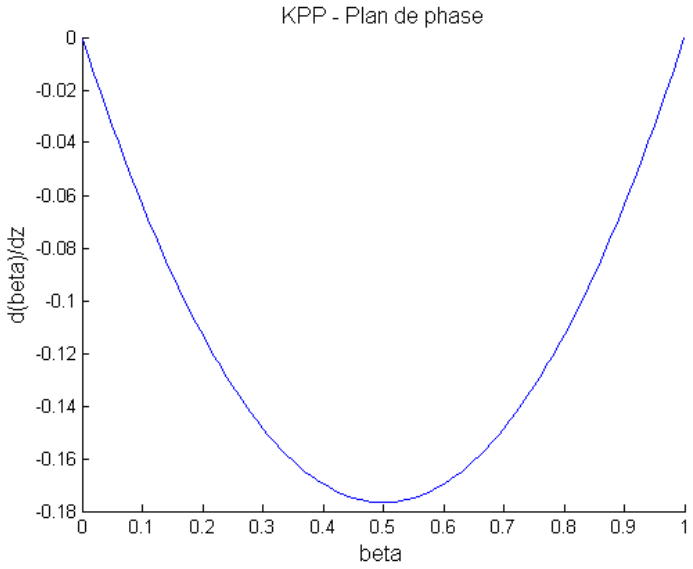
La figure 1.1 montre l'allure de cette solution à dix intervalles de temps régulièrement espacés, dans le cas $k = 1$ et $D = 1$ (cas non raide).

1.1.4 Étude dans le plan de phase.

Une étude dans le plan de phase $(\beta, d\beta/dz)$ montre que l'on a deux point stationnaires $(1, 0)$ et $(0, 0)$ (figure 1.2).

L'analyse de ceux-ci révèle que le premier est un point répulsif de la dynamique du système, et le deuxième, est un point attractif. [6; 1]

FIG. 1.2: Diagramme de phase, KPP.



Ce résultat montre clairement le comportement physique du phénomène, plus précisément, une onde qui se propage à vitesse constante (le repère z ayant été pris par rapport au front d'onde) à partir d'un état initial instable pour $\beta = 1\,(b = a_0)$ pour atteindre l'état stable $\beta = 0\,(b = 0)$, comme résultat des processus de diffusion et de la réaction chimique elle-même.

1.1.5 Étude du caractère multi échelle du problème.

En général, le caractère multi échelle du système n'est pas seulement relié aux effets non linéaires que peut générer le terme concernant la réaction chimique dans l'expression (1.5), mais aussi à la discrétisation spatial du terme diffusif. En effet, lorsque l'on augmente le nombre de points de discrétisation du laplacien, la dispersion des valeurs propres associées devient de plus en plus importante [6; 1]. En tout cas, pour ce cas particulier, le systéme dynamique associé n'est pas raide du tout.

Alors, la raideur du système est un résultat de cette dispersion, mais dépend aussi de la condition initiale pris en compte. En effet, lorsque la pente de la condition initiale augmente, de plus en plus des échelles de temps rapides vont intervenir et perturber la résolution numérique du système. En fait, une analyse spectral à partir de la transformée de Fourier en espace de la condition initiale [1] montre que les projections sur les exponentielles de nombre d'onde élevé devient d'autant plus grands que la pente augmente, en pratique, cela implique qu'on aura des phénomènes de haute fréquences difficiles à suivre lorsqu'on réalise l'intégration numérique temporelle.

1.2 Modélisation de la réaction Belousov-Zhabotinskii.

Dans l'approche la plus classique, la réaction *Belousov-Zhabotinskii* ($B-Z$) est une oxydation d'espèces organiques catalysée par la présence d'un ion d'acide à base de Brome (Br). Des observations expérimentales montrent un comportement oscillatoire des concentrations des espèces d'étude, dès qu'on a un système fortement mélangé. Ces oscillations ont des périodes avec des taux de variation assez faibles, séparés par des sauts importants. Au bout d'un temps assez long, le phénomène évolue vers son état d'équilibre et les oscillations cessent. Il faut noter qu'il s'agit d'un phénomène dissipatif (aucun paramètre en particulier ne se conserve) et que la dynamique oscillatoire existe sur une échelle de temps courte devant la convergence vers l'équilibre [6].

1.2.1 Présentation du modèle.

Un des modèles les plus répandus du phénomène est celui qui prend en compte des arguments simplificateurs du schéma de cinétique chimique, et il est connu sous le nom d'**Oregonator**.

Parmi les hypothèses simplificatrices, on ne va remarquer que les deux suivantes :

- premièrement, il y a des espèces initiales dont les concentrations restent à peu près constantes et égales à leurs valeurs initiales pendant la durée du phénomène ;
- et deuxièmement, il y en a d'autres qui apparaissent comme produit des réactions intermédiaires et qui peuvent ajuster leurs concentrations très rapidement, de manière à rester en "pseudo-équilibre" par rapport aux espèces qui ont une vitesse de variation plus faible.

Les espèces initiales ou intermédiaires qui restent après ces approximations sont celles qui vont déterminer la dynamique de la réaction.

1.2.2 Mise en équation.

A partir des hypothèses énoncées, le schéma chimique associé est,

$$\left.\begin{array}{rcll} A+Y & \rightarrow & X+P & \omega = k_3 AY, \\ X+Y & \rightarrow & 2P & \omega = k_2 XY, \\ A+X & \rightarrow & 2X+2Z & \omega = k_5 AX, \\ 2X & \rightarrow & A+P & \omega = k_4 X^2, \\ B+Z & \rightarrow & \frac{1}{2} fY & \omega = k_0 BZ, \end{array}\right\} \tag{1.15}$$

où ω représente le taux de réaction.

De cette manière, le système dynamique pour l'Oregonator, dans une configuration homogène et bien mélangée, s'écrit pour les concentrations X, Y et Z,

$$\frac{dX}{dt} = k_3 AY - k_2 XY + k_5 AX - 2k_4 X^2, \tag{1.16}$$

$$\frac{dY}{dt} = -k_3 AY - k_2 XY + \frac{1}{2} f k_0 BZ, \tag{1.17}$$

$$\frac{dZ}{dt} = 2k_5 AX - k_0 BZ. \tag{1.18}$$

Ce système est adimensionné en prenant,

$$b = \frac{X}{X_0}, \qquad a = \frac{Y}{Y_0}, \qquad c = \frac{Z}{Z_0}, \qquad \tau = \frac{t}{T_0}, \tag{1.19}$$

avec

$$X_0 = k_5A/2k_4, \qquad Y_0 = k_5A/k_2,$$
$$Z_0 = (k_5A)^2/k_4k_0B, \quad T_0 = 1/k_0B,$$

pour obtenir finalement,

$$\mu\frac{da}{d\tau} = -qa - ab + fc, \tag{1.20}$$
$$\epsilon\frac{db}{d\tau} = qa - ab + b(1-b), \tag{1.21}$$
$$\frac{dc}{d\tau} = b - c, \tag{1.22}$$

avec les paramètres

$$\epsilon = \frac{k_0B}{k_5A}, \qquad \mu = \frac{2k_0k_4B}{k_2k_5A}, \qquad q = \frac{2k_3k_4}{k_2k_5}. \tag{1.23}$$

En général, $\mu \ll \epsilon$ et $q \ll 1$.

On pourrait faire une analyse complète de bifurcation de Hopf sur ce système (1.20, 1.21 et 1.22) ; cela nous amènerait à trouver les conditions pour lesquelles on arrive à avoir un comportement oscillatoire à partir des valeurs des différents paramètres. Néanmoins, les valeurs de ces paramètres nous permettent de prendre une approche alternative.

Puisque les dérivées $db/d\tau$ et $da/d\tau$ sont multipliées par les petits paramètres ϵ et μ, où μ a une valeur beaucoup plus petite que ϵ, on pourrait s'attendre à ce que la concentration représentée par a s'ajuste rapidement à la composition instantanée du mélange réactif. De cette manière, on obtient la relation d'équilibre,

$$a = \frac{fc}{q+b}. \tag{1.24}$$

De cette manière, le système simplifié à deux variables s'écrit,

$$\epsilon\frac{db}{d\tau} = b(1-b) + \frac{f(q-b)c}{q+b}, \tag{1.25}$$
$$\frac{dc}{d\tau} = b - c, \tag{1.26}$$

où, la présence du petit paramètre ϵ, qui est petit, fait que b est la variable rapide par rapport à c, la lente.

1.2.3 Bifurcation et étude dans le plan de phase.

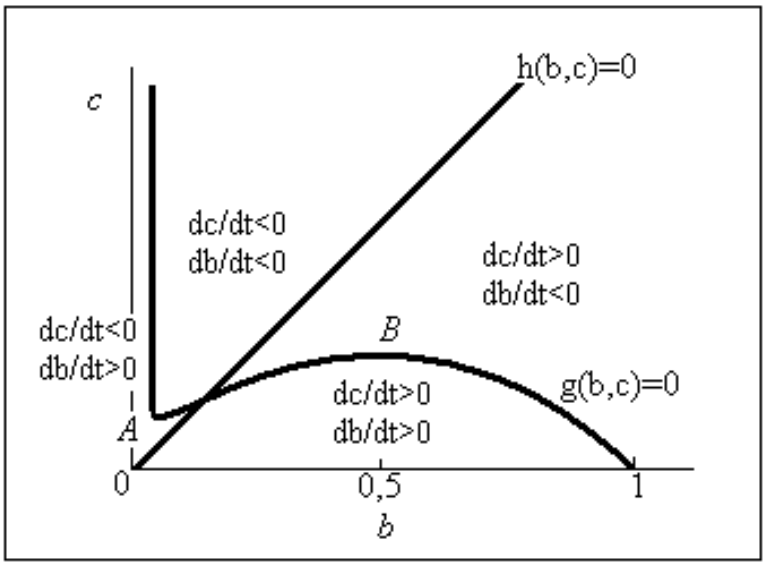
Dans un premier instant, une analyse qualitative du système (1.25 et 1.26), permet d'établir son comportement évolutif.

Considérons d'abord les variétés d'équilibre associées aux deux variables,

$$\left.\begin{aligned} g(b,c) &= b(1-b) + \frac{f(q-b)c}{q+b} &&= 0, \\ h(b,c) &= b - c &&= 0. \end{aligned}\right\} \tag{1.27}$$

La figure 1.3 montre une représentation schématique de celles-ci ; la condition $h(b,c)=0$ implique une droite de pente unitaire qui passe par l'origine. Au-dessus de cette ligne, $dc/d\tau$ est négative ; au-dessous, positive.

FIG. 1.3: Plan de phase, BZ.



La condition $g(b,c)=0$ définie la courbe cubique,

$$c=\frac{b(b-1)(q+b)}{f(q-b)}, \tag{1.28}$$

qui décroit depuis l'infinie en $b=q$, a un minimum et un maximum, et finalement, traverse l'axe en $b=1$.

En fait, l'analyse de la première dérivée nous amène à résoudre l'expression,

$$2b^3-(2q+1)b^2-2q(q-1)b+q^2=0, \tag{1.29}$$

qui pour $q\to 0$, nous donne les racines, $b=0$ et $b=\frac{1}{2}$.

En cherchant la racine proche de zéro, avec la condition $q\ll 1$ en (1.29), on trouve $b=(1\pm\sqrt{2})q$. Ces racines, avec (1.29), nous permettent d'obtenir la troisième racine $b=\frac{1}{2}-q$.

Une analyse de la deuxième dérivée nous donne les points

$$\begin{array}{lll} A: & b_A=(1+\sqrt{2})q, & c_A=\frac{1}{f}(1+\sqrt{2})^2q\left[1-(1+\sqrt{2})q\right]\approx\frac{1}{f}(1+\sqrt{2})^2q, \\ B: & b_B=\frac{1}{2}-q, & c_B=\frac{1}{4f}\frac{(1-2q)(1+2q)}{(1-4q)}\approx\frac{1}{4f}(1+2q), \end{array} \tag{1.30}$$

comme le minimum et maximum, respectivement.

Les intersections entre les deux varietés définissent clairement les points stationnaires, lesquelles vérifient $b_0=c_0$ à partir de (1.27) ; ce qui donne $b_0=c_0=0$, et la solution de

$$(1-b_0)(q+b_0)+f(q-b_0)=0, \tag{1.31}$$

s'écrit

$$b_0 = \frac{1}{2}\left\{1 - f - q \pm \left[(1-f-q)^2 + 4q(1+f)\right]^{1/2}\right\}. \tag{1.32}$$

En fait, pour tous q, et $f > 0$, on obtient une racine réelle positive et une autre négative.

Par ailleurs, l'équation (1.31) écrite sous la forme

$$b_0^2 - (1-f-q)b_0 - (1+f)q = 0, \tag{1.33}$$

nous donne pour $q \to 0$, les racines $b_0 = 0$ et $b_0 = 1 - f$. Maintenant, si l'on cherche la racine proche de zéro en (1.33), on obtient le point stationnaire écrit sous la forme

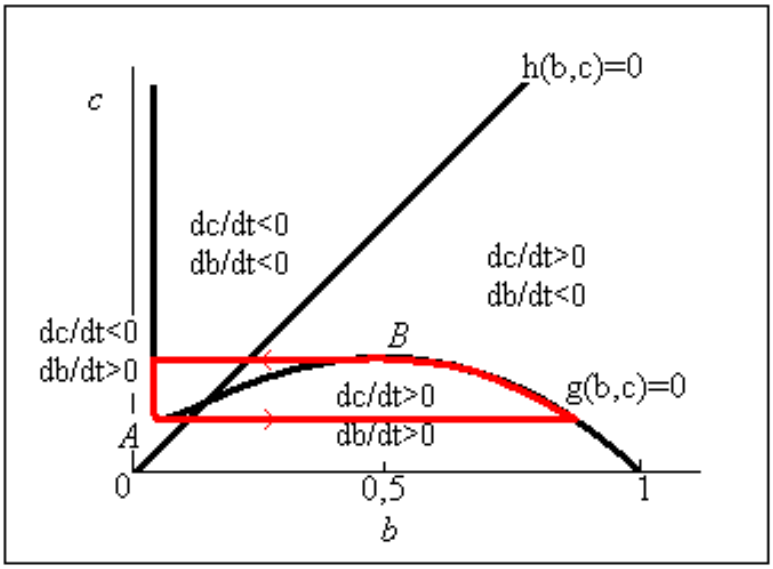
$$b_0 = \frac{q(f+1)}{(f+q-1)} \approx \frac{q(f+1)}{(f-1)}, \tag{1.34}$$

qui est une approximation de (1.32) dès que $f > 1$.

Clairement, la position du point stationnaire dépend des paramètres ; par exemple, la figure 1.3 montre le cas pour lequel on récupère un comportement oscillatoire. En fait, comme ϵ est petit, la variable rapide b, d'après l'équation (1.25), essaiera de rester sur la variété d'équilibre ; mais dès que cela ne sera pas possible, il y aura des sauts rapides.

Une analyse simple des signes des dérivées, permet de dessiner aisément l'évolution dans le plan de phase (figure 1.4).

FIG. 1.4: Cycle limite dans le plan de phase, BZ.



D'après ce qui a été dit, le point stationnaire est instable et des régimes oscillatoires sont générés, s'il reste sur la plage

$$(1+\sqrt{2})q = b_A < b_0 < b_B = \frac{1}{2} - q. \tag{1.35}$$

Pour un q fixé, la limite inferieure du comportement oscillatoire étant donnée par $b_A = c_A$, nous donne $f \approx (1+\sqrt{2})$; et $f \approx \frac{1}{2} + q$ pour $b_B = c_B$. C'est-à-dire, à la limite où ϵ est petit, les oscillations apparaissent sur l'intervalle :

$$0,5 \approx \frac{1}{2} + q < f < 1 + \sqrt{2} \approx 2,41. \tag{1.36}$$

On pourrait obtenir la même conclusion à partir d'une analyse de bifurcation de Hopf, en supposant q constante. En fait, étant donné le jacobien

$$\boldsymbol{J} = \begin{pmatrix} \frac{1}{\epsilon}\left[1 - 2b - \frac{2fqc}{(q+b)^2}\right] & \frac{f(q-b)}{\epsilon(q+b)} \\ 1 & -1 \end{pmatrix}, \tag{1.37}$$

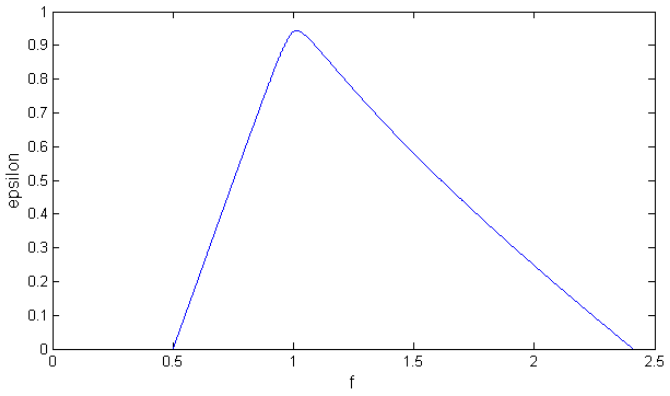
il y aura bifurcation dès que $\mathrm{tr}(\boldsymbol{J}) = 0$, c'est-à-dire $\partial(db/d\tau)/\partial b + \partial(dc/d\tau)/\partial c = 0$.

Cela implique

$$\epsilon = 1 - 2b\left(1 + \frac{fq}{(q+b)^2}\right). \tag{1.38}$$

Comme résultat, pour chaque f, on obtient le point stationnaire associé par (1.32), et ϵ par (1.38), comme le montre la figure 1.5.

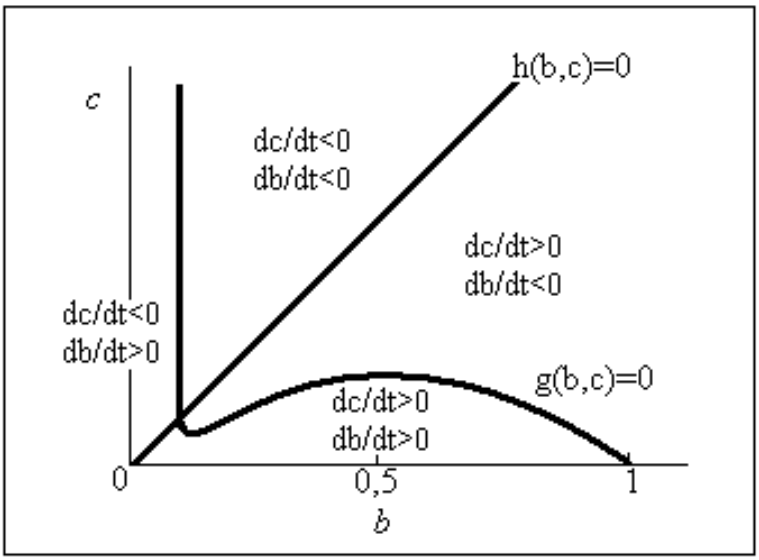
FIG. 1.5: Diagramme de bifurcation, BZ.



A la limite où $\epsilon \to 0$, on récupère les mêmes conditions trouvés avant (1.36). Au fur et à mesure que ϵ augmente, la plage d'instabilité décroit. Les oscillations ne sont plus possibles si $\epsilon > 0,94$.

Si le point stationnaire reste en dehors de l'intervalle donné par (1.35), comme le montre la figure 1.6, une analyse simple, tout à fait similaire à la précédente permet de montrer qu'il devient un puits, c'est-à-dire, un attracteur de la dynamique du système.

FIG. 1.6: Plan de phase, point stationnaire stable, BZ.



1.2.4 Système de réaction - diffusion.

Si l'on considère maintenant une configuration spatiale avec des coefficients de diffusion D_a, D_b et D_c, pour les espèces a, b et c, respectivement, on retrouve le système de réaction - diffusion suivant,

$$\begin{cases} \dfrac{\partial a}{\partial \tau} - D_a \Delta a &= \frac{1}{\mu}(-qa - ab + fc), \\ \dfrac{\partial b}{\partial \tau} - D_b \Delta b &= \frac{1}{\epsilon}\left(qa - ab + b(1-b)\right), \\ \dfrac{\partial c}{\partial \tau} - D_c \Delta c &= b - c. \end{cases} \tag{1.39}$$

Ainsi que, le système simplifié,

$$\begin{cases} \dfrac{\partial b}{\partial \tau} - D_b \Delta b &= \frac{1}{\epsilon}\left(b(1-b) + \frac{f(q-b)c}{q+b}\right), \\ \dfrac{\partial c}{\partial \tau} - D_c \Delta c &= b - c. \end{cases} \tag{1.40}$$

Pour le cas monodimensionnel, le système parabolique (1.40) possède des solutions de type onde progressive [12], alors que pour le cas bidimensionnel, on retrouve des ondes spirales [2].

Il faut tenir compte, cette fois-ci, des phénomènes multi échelles qui sont associés non seulement à la réaction, mais aussi à l'operateur laplacien. En plus, les configurations spatiales initiales jouent un rôle important, dès lorsqu'elles s'éloignent de la variété d'équilibre du système.

Chapitre 2

Méthodes numériques de résolution.

Dans cette section, on va présenter les différentes méthodes numériques que l'on a utilisées tout au long de ce projet.

En fait, on fera un bref résumé sur les méthodes de résolution des équations différentielles ordinaires utilisées. En commençant para la méthode des lignes, laquel sert à la résolution d'équations aux dérivées partielles, pour passer aux méthodes appropiées à la résolution de systèmes raides. Celles-ci incluent des méthodes d'intégration temporelle mono-pas (Runge-Kutta implicite) ainsi que multi-pas (LSODE).

De cette manière, on se limitera à une étude assez superficielle des méthodes de résolution des EDO. Celles-ci étant vastement commentées et étudiées dans la littérature et dans les références données [7; 8]. Cependant, cette caractérisation nous permettra de comprendre les principes des méthodes, lesquelles seront utiles pendant l'analyse des résultats.

2.1 La méthode des lignes (MOL).

2.1.1 Principe.

La méthode des lignes (*Méthode of Lines*, ou MOL) est une technique de résolution d'équations aux dérivées partielles.

Elle consiste à en effectuer une discrétisation spatiale, souvent par différences finies, pour obtenir un système d'équations différentielles ordinaires. Celui-ci se résout alors par l'une des nombreuses méthodes de calcul disponibles.

2.1.2 Application de la méthode.

On se propose d'appliquer la méthode à un cas général, mais qui sera finalement, celui que l'on va trouver dans cette étude.

Soit le système d'équations du type :

$$\begin{cases} \dfrac{\partial u}{\partial t} = D\dfrac{\partial^2 u}{\partial x^2} + f(u) \\ \dfrac{\partial u}{\partial x}(a,t) = 0 \\ \dfrac{\partial u}{\partial x}(b,t) = 0 \\ u(x,0) = u_0(x) \end{cases} \qquad x \in [a,b],\ t \in [0,T]. \tag{2.1}$$

On a pris des conditions aux limites de Neumann homogènes. En accord avec la méthode des lignes, on effectuera une discrétisation en espace par différences finies. On fixe le nombre $(n+1) \in \mathbb{N}^*$ de points de discrétisation et on pose

$$h = \frac{b-a}{n}.$$

On définit alors la subdivision régulière du segment $[a,b]$, $(x_i)_{i\in[0,n]}$, de pas h par

$$\forall i \in [0,n],\ x_i = a + ih.$$

La fonction $u(x,t)$ sera approchée par le vecteur $\boldsymbol{U}(t) \in \mathbb{R}^{n+1}$ tel que $\forall i \in [0,n],\ U_i(t) \approx u(t,x_i)$. On approche la dérivée seconde au point $x_i,\ i \in [1,n-1]$ par la formule :

$$\frac{\partial^2 u}{\partial x^2}(x_i) = \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2} + \mathcal{O}(h^2). \tag{2.2}$$

Au bord, les conditions font qu'on a :

$$\begin{cases} \dfrac{\partial^2 u}{\partial x^2}(x_0) = \dfrac{u(x_1) - u(x_0)}{h^2} + \mathcal{O}(h^2), \\ \dfrac{\partial^2 u}{\partial x^2}(x_n) = \dfrac{u(x_{n-1}) - u(x_n)}{h^2} + \mathcal{O}(h^2). \end{cases} \tag{2.3}$$

A partir de (2.1), (2.2) et (2.3), on trouve que la solution approchée $\boldsymbol{U}$ est la solution d'une équation différentielle ordinaire :

$$\frac{d\boldsymbol{U}}{dt} = D\left(\frac{n}{b-a}\right)^2 \boldsymbol{A_{nh}}\boldsymbol{U} + \boldsymbol{F}(\boldsymbol{U}), \tag{2.4}$$

où $\boldsymbol{A_{nh}}$ est la matrice de discrétisation du laplacien avec des conditions de Neumann homogènes :

$$\boldsymbol{A_{nh}} = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 \\ & & \ddots & \ddots & \ddots & \\ 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix} \in \mathcal{M}_{n+1}(\mathbb{R}), \tag{2.5}$$

et $\boldsymbol{F}(\boldsymbol{U})$ le vecteur ${}^t\,(f(U_0), f(U_1), \ldots, f(U_n))$.

2.2 Le solveur LSODE.

Le solveur LSODE (*Livermore Solver for Ordinary Diferential Equations*) résout numériquement les équations différentielles ordinaires, autrement dit les problèmes de la forme :

$$\begin{cases} y'(t) &= f(t, y(t)), \\ y(t_0) &= y_0. \end{cases} \tag{2.6}$$

L'objectif est de déterminer la fonction solution y en un ou plusieurs points de l'intervalle de résolution. La méthode de résolution du solveur consiste, à partir de la condition initiale, à générer des approximations de la solution exacte en des points eux-mêmes déterminés par

le solveur. La distance entre deux points d'intégration temporelle est alors variable. La principale caractéristique du solveur LSODE est sa capacité à résoudre les problèmes raides grâce à l'utilisation de la méthode des différentiations rétrogrades (BDF) [8].

2.2.1 La méthode BDF.

Le solveur utilise des schémas de résolution multi-pas linéaires pour résoudre le système (2.6). Ces schémas sont définis par

$$y_n = \sum_{j=1}^{K_1} \alpha_j y_{n-j} + h_n \sum_{j=0}^{K_2} \beta_j f_{n-j}. \tag{2.7}$$

Les y_n sont des valeurs approchées de y aux points t_n, et les f_n valent $f(y_n)$.

Pour les problèmes raides, la méthode des différentiations rétrogrades (*Backward Differentiation Formulas*, ou BDF) est parfois préférentiellement utilisée. Elle vérifie une relation de la forme (2.7) avec $K_1 = q$ et $K_2 = 0$, où q désigne l'ordre de la méthode. Elle consiste à approcher la fonction y par une polynôme interpolateur Q_n de degré inférieur ou égal à q qui interpole y_n ainsi que les q valeurs précédentes $(y_k)_{k\in[n-q,n-1]}$ aux points $(t_k)_{k\in[n-q,n-1]}$ [7].

Il faut remarquer qu'au début de l'intégration temporelle on ne connaît que la condition initial du problème (2.6). Cela implique que les q valeurs initiales nécessaires pour obtenir y_n dans (2.7) seront estimées par des méthodes d'ordre inférieur à q.

2.2.2 Formulation canonique du problème.

On peut réécrire le schéma (2.7) sous la forme

$$y_n = \psi_n + h_n \beta_0 f(y_n), \tag{2.8}$$

où ψ_n contient uniquement des termes connus, calculés aux rangs précédents et est donné pour la méthode BDF par

$$\psi_n = \sum_{j=1}^{q} \alpha_j y_{n-j}. \tag{2.9}$$

Dans cette méthode, les coefficients sont déterminés de façon à que l'équation (2.7) soit exacte si la solution du système différentiel est un polynôme de degré au plus q. Cette méthode est dite implicite car dans (2.8), y_n est solution d'une équation généralement non linéaire.

2.2.3 Schémas predicteur-correcteur.

Les schémas predicteur-correcteur sont utilisés afin de calculer la solution de l'équation non linéaire (2.8). Ils consistent en la donnée de deux schémas multi-pas :

- un schéma explicite défini par $(\alpha_j^\star)$, $q \geq 1$ et $\beta_1^\star$,
- et un schéma implicite défini par (α_j), $q \geq 1$ et β_0.

On détermine alors pour tout n une suite récurrente $y_n^{[m]}$ qui converge vers la solution de (2.8).

Le premier terme $y_n^{[0]}$, appelé predicteur, est calculé à partir des $(y_k)_{k\in[n-q,n-1]}$, grâce à la méthode explicite. Les termes suivants sont déterminés à partir de (2.8) (et par conséquence,

à partir du schéma implicite) grâce à des méthodes itératives de résolution de systèmes non linéaires.

La plus utilisé est la méthode de Newton-Raphson. Numériquement, on prend $y_n = y_n^{[M]}$ une fois que, pour l'itération M, la méthode itérative a convergé (selon le critère de convergence choisi).

2.2.4 Méthode de Newton-Raphson.

Soit la suite $(y_k)_{k\in[0,n-1]}$ des valeurs approchées de $y(t_k)$ précédemment calculées. Nous allons réécrire l'équation (2.8) afin que le problème soit équivalent à la recherche des racines d'une fonction que l'on notera $R(y_n)$. On a alors,

$$R(y_n) = y_n - \psi_n - h_n\beta_0 f(y_n) = 0. \tag{2.10}$$

On va à définir la suite $y_n^{[m]}$ qui converge vers la solution de (2.10).

On définit le premier terme ou predicteur par

$$y_n^{[0]} = \sum_{j=1}^{q} \alpha_j^\star y_{n-j} + h_n\beta_1^\star f(y_{n-1}). \tag{2.11}$$

Pour obtenir la valeur $y_n^{[m+1]}$, la méthode Newton-Raphson suppose que $y_n^{[m+1]}$ est proche de $y_n^{[m]}$ et que $R(y_n^{[m+1]}) \approx 0$ (car c'est le résultat que l'on cherche). En faisant un développement limité au voisinage de $y_n^{[m]}$ on obtient

$$\boldsymbol{P}(y_n^{[m+1]} - y_n^{[m]}) = -R(y_n^{[m]}), \tag{2.12}$$

où $\boldsymbol{P}$ est la matrice jacobienne de R. On a alors

$$y_n^{[m+1]} = y_n^{[m]} - \boldsymbol{P}^{-1}R(y_n^{[m]}). \tag{2.13}$$

La connaissance du jacobienne de R (et plus précisément de son inverse) nous permet donc de déterminer $y_n^{[m+1]}$. En effet, on a $\boldsymbol{P} = \boldsymbol{I} - h_n\beta_0\boldsymbol{J}$, où $\boldsymbol{J}$ est la matrice jacobienne de f. Par exemple, on peut calculer numériquement cette matrice de la manière suivante :

$$J_{ij} = \frac{f_i(y_j + \Delta y_j) - f_i(y_j)}{\Delta y_j}. \tag{2.14}$$

Remarquons qu'en pricipe celle-ci doit être calculée à chaque itération, néanmoins plusieurs approximations peuvent être faites pour réduire le temps de calcul.

Une fois connue cette matrice, on itère selon le schéma (2.13) pour résoudre le système non linéaire (2.10).

2.3 Méthodes de Runge-Kutta implicites.

Selon la *deuxième barrière de Dahlquist* [7], une méthode multi-pas *A*-stable ne peut pas être d'un ordre plus grand que 2. Afin de franchir cette barrière, on cherche des méthodes de Runge-Kutta implicites avec ordre élevé (les méthodes explicites nécessitant des pas d'intégration h trop petits pour maintenir la stabilité).

Soient b_i, a_{ij} $(i, j = 1, \ldots, s)$ des réels, la méthode de Runge-Kutta à s étages s'écrit :

$$\begin{cases} k_i &= f(x_0 + c_i h, y_0 + h \sum_{j=1}^{s} a_{ij} k_j) \qquad i = 1, \ldots, s \\ y_1 &= y_0 + h \sum_{i=1}^{s} b_i k_i. \end{cases} \tag{2.15}$$

Quand $a_{ij} = 0$ pour $j \geq i$, la méthode est explicite. Si $a_{ij} = 0$ pour $j > i$, et qu'au moins un des $a_{ii} \neq 0$, la méthode est dit diagonalement implicite (*diagonal implicit RK*, DIRK). Si de plus tous les termes diagonaux son égaux $a_{ii} = \gamma$ pour $i = 1, \ldots, s$, on parle de *"singly diagonal implicit method"* (SDIRK). Dans tous les autres cas, la méthode est implicite.

Contrairement au cas des méthodes explicites, pour les méthodes implicites, les k_i ne peuvent plus être évaluées de façon successives, et le système (2.15) constitue un système d'équations implicites pour la determination des k_i.

Comme décrit par [8; 11], Butcher introduit des méthodes de Runge-Kutta implicite basée sur les formules de quadrature de Radau et Lobatto ; cependant, ces méthodes ne sont pas *A*-stable. Pourtant, Ehle reprend les idées de Butcher et construit des méthodes avec des excellentes conditions de stabilité. L'une de ces méthodes, appelée méthode de RadauIIA, s'est montrée particulièrement performante. Elle est d'ordre $p = 2s - 1$ et est *A-stable*. C'est cette méthode, à $s = 3$ étages et d'ordre $p = 5$, qui est implémentée dans le code Radau5 [8]. Le tableau (2.1) montre les coefficients utilisés arrangés à l'aide du schéma $\begin{array}{c|c} c_i & a_{ij} \\ \hline & b_i \end{array}$.

TAB. 2.1: RadauIIA, méthode d'ordre 5.

$\frac{4-\sqrt{6}}{10}$	$\frac{88-7\sqrt{6}}{360}$	$\frac{296-169\sqrt{6}}{1800}$	$\frac{-2+3\sqrt{6}}{225}$
$\frac{4+\sqrt{6}}{10}$	$\frac{296+169\sqrt{6}}{1800}$	$\frac{88+7\sqrt{6}}{360}$	$\frac{-2-3\sqrt{6}}{225}$
1	$\frac{16-\sqrt{6}}{36}$	$\frac{16+\sqrt{6}}{36}$	$\frac{1}{9}$
	$\frac{16-\sqrt{6}}{36}$	$\frac{16+\sqrt{6}}{36}$	$\frac{1}{9}$

2.3.1 Le programme RADAU5.

L'implémentation de RadauIIA à $s = 3$ étages et ordre $p = 5$, a été faite par [8] dans le code RADAU5. Il faut se rappeler que la résolution du système d'équations différentielles par une méthode de RK implicite conduit à la résolution d'un système non-linéaire pour les inconnues $k_1, \ldots, k_s$.

Une résolution efficace de ces systèmes est l'un des problèmes majeurs de l'implémentation des méthodes de RadauIIA. Par la suite, quelques aspects du solveur RADAU5 seront présentés de manière succincte.

2.3.2 Reformulation du système non linéaire.

La méthode peut se réécrire sous la forme :

$$\left\{ \begin{array}{rcll} g_i & = & y_0 + h\sum_{j=1}^{s} a_{ij} f(x_0 + c_j h, g_j) & i = 1, \ldots, s \\ y_1 & = & y_0 + h\sum_{j=1}^{s} b_j f(x_0 + c_j h, g_j). & \end{array} \right. \tag{2.16}$$

On choisit de travailler aves des quantités plus petites afin de réduire les problèmes d'erreurs d'arrondis :

$$z_i = g_i - y_0, \tag{2.17}$$

ce qui permet de réécrire la première équation de (2.16) comme :

$$z_i = h\sum_{j=1}^{s} a_{ij} f(x_0 + c_j h, y_0 + z_j) \qquad i = 1, \ldots, s. \tag{2.18}$$

Ainsi, lorsque les variables $z_1, \ldots, z_s$ sont connues, la deuxième équation de (2.16) devient explicite pour y_1. Si la matrice $A = (a_{ij})$ est régulière, on peut réécrire (2.18) sous la forme :

$$\begin{pmatrix} z_1 \\ \vdots \\ z_s \end{pmatrix} = A \begin{pmatrix} hf(x_0 + c_1 h, y_0 + z_1) \\ \vdots \\ hf(x_0 + c_s h, y_0 + z_s) \end{pmatrix}. \tag{2.19}$$

Et ainsi obtenir,

$$y_1 = y_0 + \sum_{i=1}^{s} d_i z_i, \tag{2.20}$$

avec

$$(d_1, \ldots, d_s) = (b_1, \ldots, b_s) A^{-1}. \tag{2.21}$$

Dans le cas qui nous intéresse, c'est-à-dire $s = 3$, $d = (0, 0, 1)$, puisque $b_i = a_{si}\ \forall i$ (tableau 2.1).

2.3.3 Itérations de Newton simplifiées.

La résolution du système non linéaire (2.18) se fait de façon itérative par une méthode de Newton, qui conduit à chaque itération à l'inversion de :

$$\begin{pmatrix} I - ha_{11}\dfrac{\partial f}{\partial y}(x_0 + c_1 h, y_0 + z_1) & \ldots & -ha_{1s}\dfrac{\partial f}{\partial y}(x_0 + c_s h, y_0 + z_s) \\ \vdots & & \vdots \\ -ha_{s1}\dfrac{\partial f}{\partial y}(x_0 + c_1 h, y_0 + z_1) & \ldots & I - ha_{ss}\dfrac{\partial f}{\partial y}(x_0 + c_s h, y_0 + z_s) \end{pmatrix}. \tag{2.22}$$

Si l'on remplace le jacobien par une approximation :

$$J \approx \frac{\partial f}{\partial y}(x_0, y_0), \tag{2.23}$$

les itérations de Newton simplifiées deviennent :

$$\left\{ \begin{array}{rcl} (I - hA \otimes J)\Delta Z^k & = & -Z^k + h(A \otimes I)F(Z^k) \\ Z^{k+1} & = & Z^k + \Delta Z^k, \end{array} \right. \tag{2.24}$$

où $Z^k = (z_1^k, \ldots, z_s^k)^T$ est la *k*ème approximation de la solution et $\Delta Z^k = (\Delta z_1^k, \ldots, \Delta z_s^k)^T$ sont les incréments ; $F(Z^k) = (f(x_0 + c_1 h, y_0 + z_1^k), \ldots, f(x_0 + c_s h, y_0 + z_s))^T$.

On peut constater que chaque itération nécessite s évaluations de f, et la résolution d'un système linéaire de taille ns. Par contre, la matrice $(I - hA \otimes J)$ est la même pour toutes les itérations. Sa décomposition LU n'est donc réalisée qu'une seule fois.

2.3.4 Méthodes de Runge-Kutta diagonalement implicites.

Comme on avait vu avant, pour intégrer un système de n équations différentielles, une méthode implicite avec une matrice complète $s \times s$ implique la résolution simultanée de ns équations implicites (généralement non linéaires) à chaque pas de temps. Une façon d'éviter cette difficulté consiste à utiliser une matrice (a_{ij}) triangulaire inférieure (c'est-à-dire, $a_{ij} = 0 \ \forall i < j$) ; en effet, les équations pouvent être résolues en s étages successifs, et il faudra résoudre seulement un système *n*-dimensionnel à chaque étage. Ce sont des méthodes RK diagonalement implicites (DIRK).

Lorsqu'on résout les systèmes *n*-dimensionnel par une méthode itérative du type Newton, on voudrait résoudre à chaque étage, des systèmes linéaires avec des matrices des coefficients du type $I - ha_{ii}\partial f/\partial y$ (voire 2.22). Si tous les a_{ii} étaient égaux, on pourrait utiliser plusieurs fois la factorisation LU stockée d'une telle matrice. Les méthodes DIRK avec cette propriété additionnelle sont appelées SDIRK (*singly diagonal matrix*).

De cette manière, avec un schèma du type

$$\begin{array}{c|cccc} c_1 & \gamma & & & \\ c_2 & a_{21} & \gamma & & \\ \vdots & \vdots & \vdots & \ddots & \\ c_s & a_{s1} & a_{s2} & \ldots & \gamma \\ \hline & b_1 & b_2 & \ldots & b_s \end{array} \tag{2.25}$$

on peut construire des méthodes d'intégration temporelle mono-pas d'ordre 4 comme SDIRK4. Cette méthode n'est pas seulement *A*-stable, mais aussi *L*-stable [8]. Cette dernière proprieté implique dans la pratique que des erreurs liées ou generées par la présence des échelles rapides seront atenuées proprement.

2.3.5 Méthodes du type Rosenbrock.

Parmi les méthodes qui réussissent à résoudre des systèmes d'équations raides, les méthodes du type Rosenbrock (comme RODAS) sont les plus faciles à coder [8]. En général, ces méthodes appartiennent aux types de méthodes qui essaient d'éviter des systèmes non linéaires, en les remplaçant par une séquence de systèmes linéaires. Ainsi, ces méthodes sont appelées *méthodes RK linéairement implicites*.

En fait, on considère la méthode RK diagonalement implicite,

$$\begin{cases} k_i &= hf(y_0 + \sum_{j=1}^{i-1} a_{ij}k_j + a_{ii}k_i) \qquad i = 1, \ldots, s \\ y_1 &= y_0 + \sum_{i=1}^{s} b_i k_i, \end{cases} \tag{2.26}$$

appliquée à l'équation différentielle autonome

$$\begin{cases} y'(t) &= f(y(t)) \\ y(t_0) &= y_0 \end{cases} \tag{2.27}$$

L'idée principale consiste à linéariser l'expression (2.26), et on obtient le schéma

$$\begin{cases} k_i &= hf(g_i) + hf'(g_i)a_{ii}k_i \\ g_i &= y_0 + \sum_{j=1}^{i-1} a_{ij}k_j, \end{cases} \tag{2.28}$$

qui peut être interprété comme l'application d'*une* itération de Newton sur chaque étage dans (2.26) avec des values initiales $k_i^{(0)} = 0$. Au lieu de continuer les itérations jusqu'à la convergence, on prend (2.28) comme un nouveau type de méthode. On aboutit à des avantages en terme de calcul dès qu'on remplace les jacobiens $f'(g_i)$ par $J = f'(y_0)$; la méthode fait alors ces calculs seulement une fois.

En plus, on rend la méthode plus performante dès qu'on introduit des combinaisons linéaires additionnelles des termes Jk_i dans (2.28). Finalement, une méthode Rosenbrock à s étages est donnée par le schéma

$$\begin{cases} k_i &= hf\left(y_0 + \sum_{j=1}^{i-1} \alpha_{ij}k_j\right) + hJ\sum_{j=1}^{i} \gamma_{ij}k_j \qquad i = 1, \ldots, s \\ y_1 &= y_0 + \sum_{j=1}^{s} b_j k_j, \end{cases} \tag{2.29}$$

où $\alpha_{ij}, \gamma_{ij}, b_i$ sont des coefficients et $J = f'(y_0)$.

Alors, chaque étage de cette méthode est constitué par un système d'équations linéaires avec les inconnues k_i et la matrice $I - h\gamma_{ii}J$. Plus intéressantes encore sont les méthodes pour lesquelles $\gamma_{11} = \cdots = \gamma_{ss} = \gamma$, pour lesquelles on n'aura besoin que d'une seule décomposition LU par pas.

En particulier, nous, on utilisera le programme RODAS, lequel est basé sur des méthodes de Rosenbrock d'ordre 4 [8].

2.4 Le solveur SEULEX.

En principe, le solveur SEULEX est une méthode d'extrapolation basée sur l'implémentation de la méthode d'Euler linéairement implicite. En général, l'extrapolation est une technique qui permet de créer, à partir d'une méthode de base très simple, des approximations de grande précision. Dans ce cas particulier, on considère d'abord la méthode d'Euler implicite,

$$y_{i+1} = y_i + hf(y_{i+1}), \tag{2.30}$$

qui, une fois linéarisée est mis sous la forme,

$$y_{i+1} = y_i + h(I - hJ)^{-1} f(y_i), \tag{2.31}$$

où, $J \approx f'(y_0)$ est une approximation de la matrice jacobienne de $f(y)$. L'idée de l'extrapolation repose sur le fait que l'erreur globale de cet intégrateur possède un développement asymptotique en puissances de h.

En fait, prenons une suite de nombres entiers $n_1 < n_2 < n_3 < \ldots$, par exemple $n_j = j$; on fixe $h_j = H/n_j$ et l'on définit

$$T_{j1} = y_{h_j}(x_0 + H), \tag{2.32}$$

comme la solution numérique obtenue en appliquant la méthode de base n_j fois avec un pas h_j. Ensuite, avec l'algorithme d'extrapolation d'Aitken-Neville [8]

$$T_{j,k+1} = T_{j,k} + \frac{T_{j,k} - T_{j-1,k}}{n_j/n_{j-k} - 1}, \tag{2.33}$$

on peut éliminer les premiers termes du développement asymptotique et on obtient ainsi un tableau des approximations de $y(t_0 + H)$:

$$\begin{array}{llll} T_{11} & & & \\ T_{21} & T_{22} & & \\ T_{31} & T_{32} & T_{33} & \\ \vdots & \vdots & \vdots & \ddots \end{array} \tag{2.34}$$

De cette manière, les approximations dans la première colonne sont d'ordre 1 (d'après la méthode base), celles de la deuxième d'ordre 2, et celles de la *k*ième colonne d'ordre k.

Par ailleurs, on peut vérifier numériquement que les approximations données par T_{j1} et T_{j2} sont toutes *A*-stables, les autres sont *A*(α)-stables avec α très proche de $\pi/2$ [8].

Chapitre 3

Séparation d'operateur et parallélisation temporelle.

Dans ce chapitre on présentera la base théorique sur laquelle s'appuient les techniques qu'on a utilisé pour simuler la dynamique des ondes chimiques multi-échelles : les méthodes de splitting et l'algorithme pararéel.

Premièrement, on verra une approche générale des méthodes de séparation d'opérateur (splitting) dans un cas d'application simple sur des systèmes d'équations différentielles linéaires, pour présenter tout de suite la méthode de Strang, laquelle sera utilisée en particulier au long de ce projet.

Dans un deuxième temps, on évaluera le comportement de la méthode lorsque l'on applique sur des systèmes plus complexes, dans notre cas, des systèmes raides non linéaires. Une analyse de perturbation singulière nous permettra finalement estimer l'ordre de l'erreur locale.

Par la suite, on décrira les principes de la parallélisation temporelle en général, et de l'algorithme pararéel en particulier. La démarche d'application de l'algorithme est montrée et une étude de convergence développée.

Finalement, avec ces éléments, on décrira les différents aspects qui se manifestent lors de l'application des méthodes sur la simulation des ondes chimiques multi-échelles. Ces remarques seront très utiles pendant l'analyse des résultats et permettront tirer les conclusions finales du travail.

3.1 Les méthodes de splitting.

Dans cette étude on doit résoudre des systèmes d'équations de réaction-diffusion. Ces deux phénomènes pris séparément ont un comportement différent. Le système complet est donc complexe à résoudre car il mêle couplages d'inconnues et différentiations spatiales.

Les méthodes de splitting résolvent séparément les équations de réaction et de diffusion et déduisent la solution du système global couplé, afin d'éviter la résolution simultanée de problèmes couplant différents points de l'espace et de problèmes couplant toutes les inconnues. L'intérêt est d'autant plus grand que des méthodes numériques très efficaces existent pour résoudre chacun des systèmes séparément.

Il existe plusieurs méthodes de splitting [3; 2], dans le cadre de ce projet on s'intéresse seulement à la méthode de Strang.

3.1.1 Approche générale.

Considérons le cas d'une seule équation différentielle ordinaire linéaire, afin de présenter le principe,

$$\begin{cases} \dfrac{dy}{dt} = Ay + By, \\ y(0) = y_0, \end{cases} \tag{3.1}$$

dont on connaît la solution exacte,

$$y(t) = \exp\left(t(A+B)\right) y_0. \tag{3.2}$$

Ainsi, après un pas de temps δt , la solution est

$$y(\delta t) = \exp\left(\delta t(A+B)\right) y_0. \tag{3.3}$$

On considère maintenant les systèmes différentiels qui découplent les opérateurs de (3.1). D'abord, on a

$$\begin{cases} \dfrac{dy}{dt} = Ay, \\ y(0) = y_0, \end{cases} \tag{3.4}$$

avec la solution $y(t) = \exp(tA)y_0$; soit après un pas de temps δt,

$$y(\delta t) = \exp(\delta tA)y_0. \tag{3.5}$$

En prenant compte du deuxième système,

$$\begin{cases} \dfrac{dy}{dt} = By, \\ y(0) = \exp(\delta tA)y_0, \end{cases} \tag{3.6}$$

on obtient cette fois-ci $y(t) = \exp(tB)\exp(\delta tA)y_0$ qui, après un pas de temps δt, devient

$$y(\delta t) = \exp(\delta tB)\exp(\delta tA)y_0. \tag{3.7}$$

On remarque que si le commutateur $[A, B] = AB - BA$ vaut zéro alors A et B commutent, et on a $\exp\left(\delta t(A+B)\right) = \exp(\delta tA)\exp(\delta tB)$ et donc le composé des solutions des systèmes découplés est exactement égal à la solution du système (3.1). Par contre, si elles ne commutent pas, on aura une erreur lorsque l'on approxime $\exp\left(\delta t(A+B)\right)$ par $\exp(\delta tA)\exp(\delta tB)$, faisant intervenir le commutateur $[A, B]$.

3.1.2 Méthode de Strang.

La méthode de Strang $A-B-A$ consiste à approcher $e^{t(A+B)}$ par $e^{\frac{t}{2}A}e^{tB}e^{\frac{t}{2}A}$. Cela consiste à résoudre le système découplé en A (sur un demi pas de temps), puis celui en B (sur un pas de temps), et enfin, une nouvelle fois en A (sur un demi pas de temps). On remarque qu'elle est d'ordre local 3 en t, donc d'ordre 2 global [3; 2].

On a en effet,

$$e^{t(A+B)} - e^{\frac{t}{2}A}e^{tB}e^{\frac{t}{2}A} = -t^3\left(-\frac{1}{24}\left[A,[A,B]\right] + \frac{1}{12}\left[B,[B,A]\right]\right) + \mathcal{O}(t^4). \tag{3.8}$$

De la même manière, la méthode de Strang $B - A - B$ consiste à approcher $e^{t(A+B)}$ par $e^{\frac{t}{2}B}e^{tA}e^{\frac{t}{2}B}$ et elle est aussi d'ordre local 3.

3.2 Splitting de systèmes d'équations de réaction-diffusion raides.

Maintenant, on analysera le comportement de la méthode de Strang appliquée aux systèmes plus complexes, les systèmes d'équations de réaction-diffusion raides non linéaires. Pour ce travail, on se limitera aux cas avec une diffusion linéaire diagonale.

Par ailleurs, des études plus profondes ont été faites [3; 2], mais elles constituent jusqu'à aujourd'hui un sujet actuel de recherche.

Ainsi, on considère le système réaction-diffusion suivant :

$$\begin{cases} \partial_t u^\epsilon - \Delta u^\epsilon = f(u^\epsilon, v^\epsilon), & x \in \mathbb{R}^d,\ t > 0, \\ \partial_t v^\epsilon - \Delta v^\epsilon = \dfrac{g(u^\epsilon, v^\epsilon)}{\epsilon}, & x \in \mathbb{R}^d,\ t > 0, \\ u^\epsilon(0, x) = u_0(x), & x \in \mathbb{R}^d, \\ v^\epsilon(0, x) = v_0(x), & x \in \mathbb{R}^d, \end{cases} \tag{3.9}$$

où d est un entier et ϵ un paramètre petit. On appelle $T_\epsilon^t(u_0, v_0)$ le semi-flot associé au système (3.9). C'est très important prendre en compte que pour ce cas, les termes diffusifs ne couplent pas les variables rapides et lentes.

3.2.1 Analyse de perturbation singulière.

Considérons premièrement l'analyse de perturbation singulière du système dynamique de dimension fini :

$$\begin{cases} \mathrm{d}_t \overline{u}^\epsilon = f(\overline{u}^\epsilon, \overline{v}^\epsilon), & t > 0, \\ \mathrm{d}_t \overline{v}^\epsilon = \dfrac{g(\overline{u}^\epsilon, \overline{v}^\epsilon)}{\epsilon}, & t > 0, \\ \overline{u}^\epsilon(0) = \overline{u}_0, \\ \overline{v}^\epsilon(0) = \overline{v}_0, \end{cases} \tag{3.10}$$

lequel correspond au système homogène sans diffusion. Le système réduit associé à (3.10) s'écrit :

$$\begin{cases} \mathrm{d}_t \overline{u} = G(\overline{u}), & t > 0, \\ \overline{u}(0) = \overline{u}_0, \\ \overline{v}(t) = h(\overline{u}(t)), & t \geq 0, \end{cases} \tag{3.11}$$

où $G(\overline{u}) = f(\overline{u}, h(\overline{u}))$. La couche intérieure peut être considérée comme une projection sur la variété d'équilibre partiel $h(\overline{u}(t))$ pour la variable rapide v. La variable associée à v centrée en $h(\overline{u}_0)$ sera représentée par $\Pi_0\overline{v}$. Ainsi, avec l'échelle de temps $\tau = t/\epsilon$, on définit :

$$\begin{cases} \mathrm{d}_\tau \Pi_0 \overline{v} = g(u_0, h(u_0) + \Pi_0\overline{v}), & \tau > 0, \\ \Pi_0\overline{v}(0) = v_0 - h(u_0). \end{cases} \tag{3.12}$$

De cette manière, on présente le théorème suivant, lequel a été démontré dans le cadre d'un système chimique plus général [13] :

Théorème 1 *Soit $(\overline{u}_0, \overline{v}_0) \in K$, avec les hypothèses introduites dans [3] ; on suppose que K reste invariant pas seulement par le système (3.9), mais aussi par la couche intérieure (3.12) et la couche extérieur (3.11). Pour ϵ suffisamment petite, on a pour $t \in [0, +\infty)$:*

$$\overline{v}^\epsilon(t,\epsilon) = \Pi_0\overline{v}\left(\frac{t}{\epsilon}\right) + \overline{v}(t) + \mathcal{O}(\epsilon), \tag{3.13}$$

$$\overline{u}^\epsilon(t,\epsilon) = \overline{u}(t) + \mathcal{O}(\epsilon), \tag{3.14}$$

et l'estimation pour la couche limite intérieure

$$\Pi_0\overline{v}\left(\frac{t}{\epsilon}\right) = \mathcal{O}\left(\exp\left(-C\frac{t}{\epsilon}\right)\right). \tag{3.15}$$

Alors, on introduit le problème réduit pour le système complet :

$$\begin{cases} \partial_t u - \Delta u = G(u), & x \in \mathbb{R}^d,\ t > 0, \\ u(0,x) = u_0(x), & x \in \mathbb{R}^d, \\ v(t,x) = h(u(t,x)), & x \in \mathbb{R}^d, t \geq 0, \end{cases} \tag{3.16}$$

avec le semi-flot $T^t(u_0, v_0)$ associé.

Les résultats obtenus dans le théorème 1 peuvent être appliqués au système réduit (3.16) comme le montre [13] ; cela nous permet de conclure que pour ϵ suffisamment petite et $t \in [0, +\infty)$, on a :

$$\|u^\epsilon(t,\epsilon) - u(t)\|_{L^2} = \mathcal{O}(\epsilon), \tag{3.17}$$

$$\|v^\epsilon(t,\epsilon) - \Pi_0\overline{v}\left(\frac{t}{\epsilon}\right) - h(u(t))\|_{L^2} = \mathcal{O}(\epsilon), \tag{3.18}$$

et l'estimation pour la couche limite intérieure

$$\|\Pi_0\overline{v}\left(\frac{t}{\epsilon}\right)\|_{L^2} = \mathcal{O}\left(\exp\left(-C\frac{t}{\epsilon}\right)\right). \tag{3.19}$$

3.2.2 Séparation d'operateur.

On introduit la décomposition classique des termes de diffusion et réaction dans (3.9). Plus précisément, on appelle $X^t(u_0, v_0)$ la solution de l'équation de diffusion :

$$\begin{cases} \partial_t u_d^\epsilon - \Delta u_d^\epsilon = 0, & x \in \mathbb{R}^d,\ t > 0, \\ \partial_t v_d^\epsilon - \Delta v_d^\epsilon = 0, & x \in \mathbb{R}^d,\ t > 0, \end{cases} \tag{3.20}$$

avec les données initiales $u_d(0,\cdot) = u_0(\cdot)$ et $v_d(0,\cdot) = v_0(\cdot)$. De la même manière, on appelle $Y_\epsilon^t(u_0, v_0)$ la solution du terme de réaction où la coordonnée x peut être considérée comme un paramètre :

$$\begin{cases} \partial_t \overline{u}_r^\epsilon = f(\overline{u}_r^\epsilon, \overline{v}_r^\epsilon), & x \in \mathbb{R}^d, t > 0, \\ \partial_t \overline{v}_r^\epsilon = \dfrac{g(\overline{u}_r^\epsilon, \overline{v}_r^\epsilon)}{\epsilon}, & x \in \mathbb{R}^d, t > 0, \end{cases} \tag{3.21}$$

avec les données initiales $\overline{u}_r^\epsilon(0,\cdot) = u_0(\cdot)$ et $\overline{v}_r^\epsilon(0,\cdot) = v_0(\cdot)$.

L'approximation de Strang qui finit avec le demi-pas de réaction est définie par :

$$S_\epsilon^t(u_0, v_0) = Y_\epsilon^{t/2} X^t Y_\epsilon^{t/2}(u_0, v_0). \tag{3.22}$$

Si l'on introduit maintenant le problème réduit quand ϵ tend vers 0, $Y^t(u_0, v_0)$ est défini par la solution du système :

$$\begin{cases} \partial_t \overline{u}_r = f(\overline{u}_r, h(\overline{u}_r)) = G(\overline{u}_r), & x \in \mathbb{R}^d, t > 0, \\ \overline{u}_r(0,x) = u_0(x), & x \in \mathbb{R}^d, \\ \overline{v}_r(t,x) = h(\overline{u}_r(t,x)), & x \in \mathbb{R}^d, t \geq 0, \end{cases} \tag{3.23}$$

et ne dépende pas de v_0, alors on écrit $(\overline{u}_r, \overline{v}_r) = Y^t u_0$. Ce problème réduit nous permet de définir le schèma réduit de splitting correspondant à (3.22), où les échelles rapides ont été relaxées dans le terme réactif en utilisant le problème réduit ; ainsi, il s'écrit :

$$S^t u_0 = Y^{t/2} X^t Y^{t/2} u_0. \tag{3.24}$$

On pourrait exprimer les composants

$$u(t) = S_1^t u_0 = Y_1^{t/2} X_1^t Y_1^{t/2} u_0 \tag{3.25}$$

et

$$v(t) = S_2^t u_0 = Y_2^{t/2} X_2^t Y_2^{t/2} u_0 = h(Y_1^{t/2} X_1^t Y_1^{t/2} u_0). \tag{3.26}$$

à partir des dèveloppements asymptotiques. C'est-à-dire,

$$\begin{aligned} Y_1^{t/2} u_0 &= u_0 + \int_0^{t/2} G(u(s))\,\mathrm{d}s \\ &= u_0 + \frac{t}{2} G(u_0) + \mathcal{O}(t^2), \\ e^{t\Delta} Y_1^{t/2} u_0 &= e^{t\Delta}\left(u_0 + \frac{t}{2} G(u_0) + \mathcal{O}(t^2)\right) \\ &= u_0 + \frac{t}{2} G(u_0) + t\Delta u_0 + \mathcal{O}(t^2), \\ Y_1^{t/2} e^{t\Delta} Y_1^{t/2} u_0 &= e^{t\Delta} Y_1^{t/2} u_0 + \int_0^{t/2} G(u(s))\,\mathrm{d}s; \end{aligned}$$

ainsi,

$$u(t) = u_0 + t(G(u_0) + \Delta u_0) + \mathcal{O}(t^2). \tag{3.27}$$

Alors, à partir de

$$h(Y_1^{t/2} e^{t\Delta} Y_1^{t/2} u_0) = h\left(u_0 + t(G(u_0) + \Delta u_0) + \mathcal{O}(t^2)\right),$$

on retrouve

$$v(t) = h(u_0) + t h'(u_0)(G(u_0) + \Delta u_0) + \mathcal{O}(t^2). \tag{3.28}$$

3.2.3 Estimation de l'ordre de l'erreur locale.

Basé sur les études faites par [3], on peut arriver aux estimations suivantes :

$$\|u_{err}\|_{L^2} = \mathcal{O}(t^3), \qquad \|v_{err}\|_{L^2} = \mathcal{O}(t^3) \tag{3.29}$$

où

$$(u_{err}, v_{err}) = T^t u_0 - S^t u_0, \tag{3.30}$$

pour t suffisamment petite.

En fait, par définition, u_{err} est la différence entre u solution de (3.16) et le terme $Y_1^{t/2} e^{t\Delta} Y_1^{t/2} u_0$, donc u_{err} constitue l'erreur locale usuelle pour la formule de Strang.

Par ailleurs,

$$v_{err} = h(u) - h(Y_1^{t/2} e^{t\Delta} Y_1^{t/2} u_0)$$

et comme h est une fonction Lipschitz associée au compacte invariant, on a

$$\|v_{err}\|_{L^2} \leq C_1 \|u - Y_1^{t/2} e^{t\Delta} Y_1^{t/2} u_0\|_{L^2} = C_1 \|u_{err}\|_{L^2}, \tag{3.31}$$

c'est-à-dire que v_{err} a le même comportement que u_{err}.

Finalement, on pourrait estimer l'erreur locale du système complet

$$\|T_\epsilon^t(u_0, v_0) - S_\epsilon^t(u_0, v_0)\|_{L^2},$$

dès que

$$\begin{aligned} T_\epsilon^t(u_0, v_0) - S_\epsilon^t(u_0, v_0) &= T_\epsilon^t(u_0, v_0) - T^t u_0 \\ &\quad + T^t u_0 - S^t u_0 \\ &\quad + S^t u_0 - S_\epsilon^t(u_0, v_0), \end{aligned}$$

et

$$\begin{aligned} \|T_\epsilon^t(u_0, v_0) - S_\epsilon^t(u_0, v_0)\|_{L^2} &\leq \|T_\epsilon^t(u_0, v_0) - T^t u_0\|_{L^2} \\ &\quad + \|T^t u_0 - S^t u_0\|_{L^2} \\ &\quad + \|S^t u_0 - S_\epsilon^t(u_0, v_0)\|_{L^2}; \end{aligned}$$

expression que nous amène à l'estimation de l'erreur locale donnée par

$$\|T_\epsilon^t(u_0, v_0) - S_\epsilon^t(u_0, v_0)\|_{L^2} = \mathcal{O}(t^3) + \mathcal{O}\left(\exp\left(-C\frac{t}{\epsilon}\right)\right) + \mathcal{O}(\epsilon). \tag{3.32}$$

Ces résultats montrent que l'ordre de l'erreur locale de la méthode de Strang avec une séquence R-D-R (réaction-diffusion-réaction) est préservé. Néanmoins, il faut noter que ces estimations ont un caractère local et des études au niveau de l'erreur globale sont encore sujet de recherche.

Cette préservation de l'ordre de l'erreur, au moins au niveau locale, fait cette méthode préférable par rapport aux autres, pour lesquelles il y a perte de l'ordre, même au niveau local [3].

3.3 Parallélisation temporelle.

Suite à la demande toujours croissante des simulations numériques de plus en plus, plus précises et par conséquence, plus coûteuses, les méthodes de parallélisation du domaine temporel sont envisagées comme des outils de résolution puissants qui permettent de réduire ces coûts en conservant une précision tout à fait acceptable.

Parmi ces méthodes, l'on étudiera en particulier l'algorithme pararéel [10; 4; 5], lequel a été implémenté au long de ce projet.

3.3.1 Principe général.

Comme l'on a déjà dit, l'algorithme pararéel est un algorithme de parallélisation temporelle de résolution des systèmes d'équations différentielles ordinaires, en général non linéaires du type

$$\boldsymbol{u}'(t) = \boldsymbol{f}\left(\boldsymbol{u}(t)\right), \quad t \in (0,T), \quad \boldsymbol{u}(0) = \boldsymbol{u}^0, \tag{3.33}$$

où $\boldsymbol{f} : \mathbb{R}^M \to \mathbb{R}^M$ et $\boldsymbol{u} : \mathbb{R} \to \mathbb{R}^M$. Pour obtenir un algorithme de parallélisation temporelle pour (3.33), on décompose le domaine temporel $\Omega = (0,T)$ en N sous domaines $\Omega_n = (T_n, T_{n+1})$, $n = 0, 1, \ldots, N-1$, avec $0 = T_0 < T_1 < \ldots < T_{N-1} < T_N = T$, et $\Delta T_n := T_{n+1} - T_n$ et, finalement, on considère pour chaque sous domaine temporel le problème d'évolution

$$\boldsymbol{u}_n'(t) = \boldsymbol{f}\left(\boldsymbol{u}_n(t)\right),\ t \in (T_n, T_{n+1}),\ \boldsymbol{u}_n(T_n) = \boldsymbol{U}_n,\ n = 0, 1, \ldots, N-1, \tag{3.34}$$

où les valeurs initiales $\boldsymbol{U}_n$ sont nécessaires, pour que les solutions des sous domaines Ω_n soient cohérentes avec les contraintes de la solution de (3.33) sur Ω_n, c'est-à-dire, les $\boldsymbol{U}_n$ doivent satisfaire le système d'équations

$$\boldsymbol{U}_0 = \boldsymbol{u}^0,\ \boldsymbol{U}_n = \boldsymbol{\varphi}_{\Delta T_{n-1}}\left(\boldsymbol{U}_{n-1}\right),\ n = 0, 1, \ldots, N-1, \tag{3.35}$$

où $\boldsymbol{\varphi}_{\Delta T_n}\left(\boldsymbol{U}\right)$ est la solution de (3.33) avec la condition initiale $\boldsymbol{U}$ après un temps ΔT_n.

Si l'on écrit $\boldsymbol{U} = (\boldsymbol{U}_0^T, \ldots, \boldsymbol{U}_{N-1}^T)^T$, le système (3.35) s'écrit comme

$$\boldsymbol{F}(\boldsymbol{U}) = \begin{pmatrix} \boldsymbol{U}_0 - \boldsymbol{u}^0 \\ \boldsymbol{U}_1 - \boldsymbol{\varphi}_{\Delta T_0}\left(\boldsymbol{U}_0\right) \\ \vdots \\ \boldsymbol{U}_{N_1} - \boldsymbol{\varphi}_{\Delta T_{N-2}}\left(\boldsymbol{U}_{N-2}\right) \end{pmatrix} = \boldsymbol{0}, \tag{3.36}$$

où $\boldsymbol{F} : \mathbb{R}^{M \cdot N} \to \mathbb{R}^{M \cdot N}$.

Le système (3.36) permettrait d'obtenir les valeur initiales $\boldsymbol{U}_n$ inconnues pour chaque sous domaine temporel, et il faut le résoudre, en général, par une méthode itérative.

Si l'on applique la méthode de Newton sur (3.36), on obtient après les calculs

$$\begin{aligned} \boldsymbol{U}_0^{k+1} &= \boldsymbol{u}^0, \\ \boldsymbol{U}_n^{k+1} &= \boldsymbol{\varphi}_{\Delta T_{n-1}}\left(\boldsymbol{U}_{n-1}^k\right) + \frac{\partial \boldsymbol{\varphi}_{\Delta T_{n-1}}}{\partial \boldsymbol{U}_{n-1}}\left(\boldsymbol{U}_{n-1}^k\right)\left(\boldsymbol{U}_{n-1}^{k+1} - \boldsymbol{U}_{n-1}^k\right), \end{aligned} \tag{3.37}$$

où $n = 0, 1, \ldots, N-1$.

Des études montrent que la méthode (3.37) converge quadratiquement, une fois que les approximations sont assez proches de la solution [4]. Néanmoins, en général, le calcul exact des termes des Jacobiennes a un coût très élevé.

3.3.2 L'algorithme pararéel.

L'algorithme pararéel constitue une approche récente pour résoudre le système (3.37), comme on verra par la suite.

D'abord, l'algorithme considère deux approximations avec des précisions différentes :

- soit $\boldsymbol{F}(T_n, T_{n-1}, \boldsymbol{U}_{n-1})$ une approximation précise de la solution $\boldsymbol{\varphi}_{\Delta T_{n-1}}(\boldsymbol{U}_{n-1})$ sur le sous domaine Ω_{n-1},
- et $\boldsymbol{G}(T_n, T_{n-1}, \boldsymbol{U}_{n-1})$, une approximation moins précise, par exemple avec un maillage plus grossier, ou une méthode d'ordre plus bas, ou même, une approximation en utilisant un modèle plus simple que (3.33).

Avec ces deux approximations, on approche la solution du sous domaine temporel dans (3.37) par

$$\boldsymbol{\varphi}_{\Delta T_{n-1}}\left(\boldsymbol{U}_{n-1}^{k}\right) \approx \boldsymbol{F}(T_n, T_{n-1}, \boldsymbol{U}_{n-1}^{k}), \tag{3.38}$$

et le terme jacobien par

$$\frac{\partial \boldsymbol{\varphi}_{\Delta T_{n-1}}}{\partial \boldsymbol{U}_{n-1}}\left(\boldsymbol{U}_{n-1}^{k}\right)\left(\boldsymbol{U}_{n-1}^{k+1} - \boldsymbol{U}_{n-1}^{k}\right) \approx \boldsymbol{G}(T_n, T_{n-1}, \boldsymbol{U}_{n-1}^{k+1}) - \boldsymbol{G}(T_n, T_{n-1}, \boldsymbol{U}_{n-1}^{k}). \tag{3.39}$$

Et finalement, avec (3.38) et (3.39), l'on approche (3.37) par

$$\begin{aligned} \boldsymbol{U}_0^{k+1} &= \boldsymbol{u}^0, \\ \boldsymbol{U}_n^{k+1} &= \boldsymbol{F}(T_n, T_{n-1}, \boldsymbol{U}_{n-1}^{k}) + \boldsymbol{G}(T_n, T_{n-1}, \boldsymbol{U}_{n-1}^{k+1}) - \boldsymbol{G}(T_n, T_{n-1}, \boldsymbol{U}_{n-1}^{k}). \end{aligned} \tag{3.40}$$

L'expression (3.40) donne finalement, le schéma de l'algorithme pararéel. La solution initiale est obtenu par la méthode grossière, c'est-à-dire,

$$\boldsymbol{U}_n^0 = \boldsymbol{G}(T_n, T_{n-1}, \boldsymbol{U}_{n-1}^{0}). \tag{3.41}$$

3.3.3 Analyse de convergence de l'algorithme pararéel.

On reprend le problème d'évolution (3.34) avec les considérations suivantes :

- tous les sous domaines sont égaux, c'est-à-dire,

$$\Delta T_n = \Delta T := \frac{T}{n},\ n = 0, 1, \ldots, N-1, \tag{3.42}$$

- l'approximation fine $\boldsymbol{F}$ est la solution exacte, c'est-à-dire

$$\boldsymbol{F}(T_n, T_{n-1}, \boldsymbol{U}_{n-1}^{k}) = \boldsymbol{\varphi}_{\Delta T_{n-1}}\left(\boldsymbol{U}_{n-1}^{k}\right), \tag{3.43}$$

- $\boldsymbol{G}(T_n, T_{n-1}, \boldsymbol{U}_{n-1}^{k})$ est une approximation grossière de la solution exacte avec une erreur locale de troncature limitée par $C_3 \Delta T^{p+1}$.

Comme présenté par [4], l'analyse de convergence de l'algorithme, elle est basée principalement sur deux conditions :

- pour ΔT petit, la différence entre l'approximation grossière donnée par $\boldsymbol{G}$ et la solution exacte peut être développée de la manière suivante,

$$\boldsymbol{F}(T_n, T_{n-1}, \boldsymbol{x}) - \boldsymbol{G}(T_n, T_{n-1}, \boldsymbol{x}) = c_{p+1}(\boldsymbol{x})\Delta T^{p+1} + c_{p+2}(\boldsymbol{x})\Delta T^{p+2} + \ldots, \tag{3.44}$$

où les $c_j,\ j = p+1, p+2, \ldots$ sont continument différenciables ;

– l'approximation grossière satisfait une condition Lipschitz, c'est-à-dire,

$$\|\boldsymbol{G}(t+\Delta T,t,\boldsymbol{x})-\boldsymbol{G}(t+\Delta T,t,\boldsymbol{y})\|\leq(1+C_2\Delta T)\|\boldsymbol{x}-\boldsymbol{y}\|. \tag{3.45}$$

En fait, à partir de la définition de l'algorithme (3.40) et l'hypothèse (3.43), on obtient l'expression

$$\begin{aligned}\boldsymbol{u}(T_n)-\boldsymbol{U}_n^{k+1} &= \boldsymbol{F}(T_n,T_{n-1},\boldsymbol{u}(T_{n-1}))-\Big(\boldsymbol{F}(T_n,T_{n-1},\boldsymbol{U}_{n-1}^k)\\ &\quad+\boldsymbol{G}(T_n,T_{n-1},\boldsymbol{U}_{n-1}^{k+1})-\boldsymbol{G}(T_n,T_{n-1},\boldsymbol{U}_{n-1}^k)\Big),\end{aligned}$$

soit,

$$\begin{aligned}\boldsymbol{u}(T_n)-\boldsymbol{U}_n^{k+1} &= \boldsymbol{F}(T_n,T_{n-1},\boldsymbol{u}(T_{n-1}))-\boldsymbol{G}(T_n,T_{n-1},\boldsymbol{u}(T_{n-1}))\\ &\quad-\Big(\boldsymbol{F}(T_n,T_{n-1},\boldsymbol{U}_{n-1}^k)-\boldsymbol{G}(T_n,T_{n-1},\boldsymbol{U}_{n-1}^k)\Big)\\ &\quad+\boldsymbol{G}(T_n,T_{n-1},\boldsymbol{u}(T_{n-1}))-\boldsymbol{G}(T_n,T_{n-1},\boldsymbol{U}_{n-1}^{k+1}).\end{aligned} \tag{3.46}$$

Maintenant, en considérant d'abord la condition (3.44) pour les deux premiers termes et (3.45) pour le dernier, on prend des normes et on arrive à l'estimation,

$$\|\boldsymbol{u}(T_n)-\boldsymbol{U}_n^{k+1}\|\leq C_1\Delta T^{p+1}\|\boldsymbol{u}(T_{n-1})-\boldsymbol{U}_{n-1}^k\|+(1+C_2\Delta T)\|\boldsymbol{u}(T_{n-1})-\boldsymbol{U}_{n-1}^{k+1}\|. \tag{3.47}$$

Laquelle nous amène à l'étude de la relation de récurrence

$$e_n^{k+1}=\alpha e_{n-1}^k+\beta e_{n-1}^{k+1},\quad e_n^0=\gamma+\beta e_{n-1}^0, \tag{3.48}$$

où $\alpha=C_1\Delta T^{p+1}$, $\beta=1+C_2\Delta T$ et $\gamma=C_3\Delta T^{p+1}$; dès qu'on définit e_n^k comme limite supérieure à $\|\boldsymbol{u}(T_n)-\boldsymbol{U}_n^k\|$. Lorsqu'on multiplie (3.48) par ζ^n et on fait la somme sur n, on voit que la fonction génératrice $\rho_k(\zeta):=\sum_{n\geq1}e_n^k\zeta^n$ s'ajuste à la relation de récurrence

$$\rho_{k+1}(\zeta)=\alpha\zeta\rho_k(\zeta)+\beta\zeta\rho_{k+1}(\zeta),\quad \rho_0(\zeta)=\gamma\frac{\zeta}{1-\zeta}+\beta\zeta\rho_0(\zeta).$$

Par induction, on obtient pour $\rho_k(\zeta)$,

$$\rho_k(\zeta)=\gamma\alpha^k\frac{\zeta^{k+1}}{(1-\zeta)}\frac{1}{(1-\beta\zeta)^{k+1}}\leq\gamma\alpha^k\frac{\zeta^{k+1}}{(1-\beta\zeta)^{k+2}},$$

où

$$\frac{1}{(1-\beta\zeta)^{k+2}}=\sum_{j\geq0}\left(\begin{array}{c}k+1+j\\ j\end{array}\right)\beta^j\zeta^j,$$

par le développement du binomiale. Pour le *n*-ième coefficient e_n^k, on estime la limite

$$e_n^k\leq\gamma\alpha^k\beta^{n-k-1}\left(\begin{array}{c}n\\ k+1\end{array}\right),$$

et en revenant aux valeurs initiales, on atteint l'estimation finale,

$$\begin{aligned}\|\boldsymbol{u}(T_n) - \boldsymbol{U}_n^k\| &\leq \frac{C_3}{C_1}\frac{(C_1\Delta T^{p+1})^{k+1}}{(k+1)!}(1+C_2\Delta T)^{n-k-1}\prod_{j=0}^{k}(n-j), \\ &\leq \frac{C_3}{C_1}\frac{(C_1 T_n)^{k+1}}{(k+1)!}e^{C_2(T_n-T_{k+1})}\Delta T^{p(k+1)}. \end{aligned} \tag{3.49}$$

3.3.4 L'algorithme pararéel appliqué aux systèmes raides.

Dans cette section on analysera l'influence de la raideur sur la performance de l'algorithme pararéel. A ce respect, on va se baser sur l'analyse de convergence qu'on vient de faire.

D'abord, on considère un système d'équations différentielles raides comme (3.9).

On reprend les hypothèses (3.42) et (3.43), c'est-à-dire, on considère des intervalles de temps égaux et que l'approximation fine est la solution exacte elle-même.

Supposons qu'on choisisse comme approximation grossière la méthode de séparation d'opérateur de Strang (3.22). A priori, ce choix est totalement justifié car d'une part celle-là ne montre pas de pertes d'ordre au moins au niveau local, et d'autre, on pourrait profiter des avantages liés au traitement séparé d'opérateurs.

Pour retrouver les conditions équivalentes à (3.44) et (3.45), on reprend l'analyse de perturbation singulière faite, c'est-à-dire, on considère le système réduit (3.16) avec les estimations (3.17), (3.18) et (3.19).

De cette manière, la condition (3.44) est représentée par l'expansion asymptotique de l'erreur locale (3.32) ; alors que la deuxième peut être reconstruite en estimant la variation de $S_\epsilon^t(u_0, v_0)$ pour des conditions initiales différentes, en considérant toujours l'analyse de perturbation singulière.

Une fois établies les conditions (3.44) et (3.45), la démarche est tout à fait similaire à celle faite pour le cas général, et alors, on pourrait trouver des estimations qui prouvent la convergence du schéma pararéel.

Par contre, il y a des remarques à faire qui constituent à la fois des contraintes à prendre en compte :

- L'analyse de perturbation singulière qui permet l'étude des couches intérieur et extérieur, est totalement locale, c'est-à-dire, elle ne tient pas en compte d'autres phénomènes qui peuvent se présenter lors de l'évolution temporelle du système dans un contexte global, comme par exemple la présence de bifurcations ou de cycles limites ; alors qu'une étude au niveau local de la performance ou la convergence de l'algorithme pararéel a aucun sens.
- L'introduction de l'expansion asymptotique de l'erreur locale (3.32) comme condition préliminaire implique au même temps l'introduction des termes exponentielles qui doivent êtres estimés proprement, puisque ceux-là sont liés justement aux échelles rapides qu'on voudrait résoudre.
- De la même manière, une condition de Lipschitz globale semble assez difficile car la présence des échelles rapides ainsi que la structure particulière des variétés impliquent des comportements non uniformes sur tout le domaine.

Alors, on voit que des considérations et hypothèses appropriés sont nécessaires pour faire une analyse plus détaillée du problème. Il résulte évident qu'une des majeures difficultés consiste à trouver une façon de décrire globalement l'influence des différents phénomènes locaux.

Néanmoins, les analyses faites nous permettront évaluer et analyser des résultats numériques qui peuvent donner des conclusions intéressantes et par ailleurs, serviront de base pour des études futures.

Chapitre 4

Résultats.

Dans cette section, on va présenter les différents résultats obtenus lors de l'application de l'algorithme pararéel avec les analyses correspondantes. Dans un premier temps, l'application de celui-ci sur un système dynamique raide comme l'Oregonator, nous permettra d'obtenir des conclusions préliminaires intéressantes, lesquelles introduiront quelques critères nécessaires à prendre en compte lors de son application sur des systèmes plus complexes.

Ensuite, on évaluera la performance atteinte pour le modèle KPP non raide classique et on la comparera à celle qui est atteinte lorsqu'on introduit des forts gradients spatiaux (KPP raide).

Puis, on étudiera le modèle BZ avec d'abord le système simplfié à deux variables pour lequel on résumera les différents problèmes qui se posent comme conséquence de la raideur du système, avant de finir avec le système complet.

Finalement, une estimation du coût de calcul est faite en considérant plusieurs méthodes de résolution pour le modèle BZ bidimensionnel ainsi comme les calculs sur architecture parallèle du modèle KPP.

4.1 L'Oregonator.

On va considérer le modèle de l'Oregonator étudié dans [8] qui reproduit une réaction chimique non linéaire multi-échelle et s'écrit,

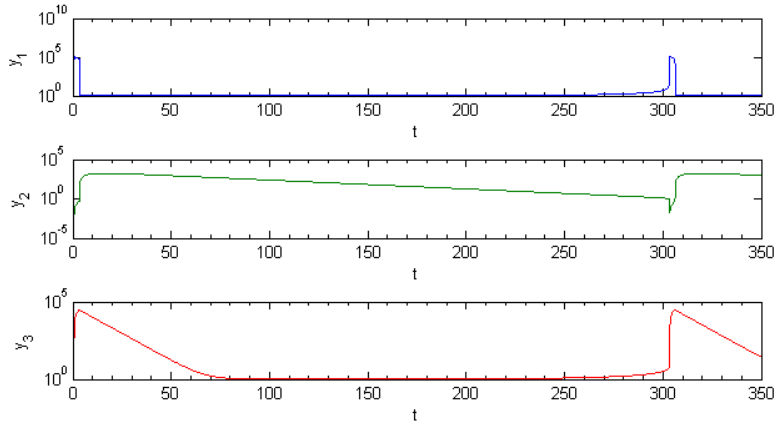
$$\begin{cases} y_1' &= 77,27\left(y_2 + y_1(1 - 8,375.10^{-6}y_1 - y_2)\right), \\ y_2' &= \dfrac{1}{77,27}\left(y_3 - (1+y_1)y_2\right), \\ y_3' &= 0,161(y_1 - y_3). \end{cases} \tag{4.1}$$

Avec ces paramètres, on retrouve un comportement oscillatoire qui tend vers un cycle limite. Le système étant raide, il faut le résoudre avec une méthode appropriée. Il est sans espoir d'essayer de résoudre ce système par une méthode explicite. Nous utilisons donc un des algorithmes les plus efficaces que nous ayons pour les problèmes raides, le code RADAU5. Les solutions sont représentées dans la figure 4.1.

Pour appliquer l'algorithme pararéel il faut choisir deux méthodes de résolution, une fine et l'autre grossière. Ici, on utilise RADAU5 pour les deux, sauf que pour la résolution fine on prendra des tolérances les plus petites possibles, alors qu'on les relaxera pour la grossière.

Il faut tenir compte du fait que la taille et le nombre des intervalles d'intégration n'ont aucun intérêt pratique dès que la méthode numérique utilise des pas d'intégration automatiquement

FIG. 4.1: Solutions de l'Oregonator.



adaptés selon les tolérances fixées. Autrement dit, ce choix reste en tout cas une conséquence du nombre de processeurs à utiliser.

On a choisi arbitrairement un état initial donné par $y_1(0) = 3$, $y_2(0) = 1$ et $y_3(0) = 2$.

Dans une première approche on intégrera le système sur un domaine temporel $[0, 200]$ plus petit que la période des oscillations. Le tableau 4.1 montre les résultats lorsqu'on relaxe de plus en plus les tolérances de la méthode grossière ; les tolérances fines étant fixées à 1.10^{-15}. De cette manière on voit que la méthode n'arrive pas à converger quand on utilise une résolution très grossière pour le système raide. En plus, on récupère un comportement global tout à fait logique : lorsqu'on approxime le système par une résolution de plus en plus grossière, on a besoin de plus d'itérations de l'algorithme pour converger ; on verra plus tard comment cela affecte la performance de l'algorithme.

TAB. 4.1: Influence du solveur grossier, premier cas, Oregonator.

tolérence grossière	**nombre d'itérations**
1.10^{1}	121
1.10^{-1}	47
1.10^{-3}	45
1.10^{-5}	23
1.10^{-7}	3
1.10^{-9}	2
1.10^{-11}	1

Maintenant, on intègre le système sur un domaine temporel $[0, 600]$ qui dépasse une période d'oscillation, c'est-à-dire, qu'on récupère effectivement le caractère oscillatoire du phénomène.

Le tableau 4.2 montre les résultats pour ce cas-ci. Evidement, la performance se dégrade par rapport au premier cas considéré. En fait, dès que les itérations de l'algorithme essaient de corriger l'approximation grossière initial, la présence de variations rapides justement à la fin

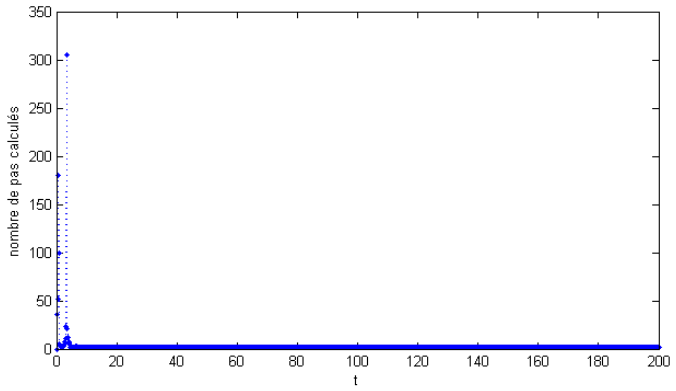
du domaine d'intégration ajoutée aux erreurs propagées, rendront cette approximation initiale encore plus grossière et par ailleurs, plus coûteuse.

TAB. 4.2: Influence du solveur grossier, deuxième cas, Oregonator.

tolérence grossière	**nombre d'itérations**
1.10^{-4}	68
1.10^{-6}	2
1.10^{-8}	1

D'autre part, si l'on regarde le nombre de pas calculés par la méthode, soit acceptés ou rejetés à cause du test d'erreur, celui-ci varie de 2 jusqu'à 305, pour la configuration particulière du premier cas. La majeure partie des pas est nécessaire justement là où le système devient plus raide (figure 4.2), cela implique que dans un environnement parallèle, le temps de calcul pour chaque processeur varierait selon la même distribution, donnant comme résultat une répartition très hétérogène du travail de calcul parmi les différents processeurs. Evidemment, cette remarque est valable lorsqu'on a une configuration temporelle raide bien localisée comme dans ce cas, ce qui n'est pas le cas général.

FIG. 4.2: Nombre de pas calculés, Oregonator.



Cette première approche montre en particulier comment le choix du solveur grossier devient un critère critique lors de l'application de l'algorithme dès qu'on essaie de résoudre des systèmes raides.

4.2 Modèle KPP.

Le modèle étudié est un système de réaction-diffusion. Sous une forme compacte il est représenté par (1.5).

D'abord, on s'intéressera au cas non raide du modèle. Cela implique que l'on prend $k = 1$ et $D = 1$ sur l'intervalle spatial $[-70, 70]$ et un temps de 30 unités, la vitesse de l'onde étant proportionnelle à $(kD)^{1/2}$ et la pente maximale à $(k/D)^{1/2}$

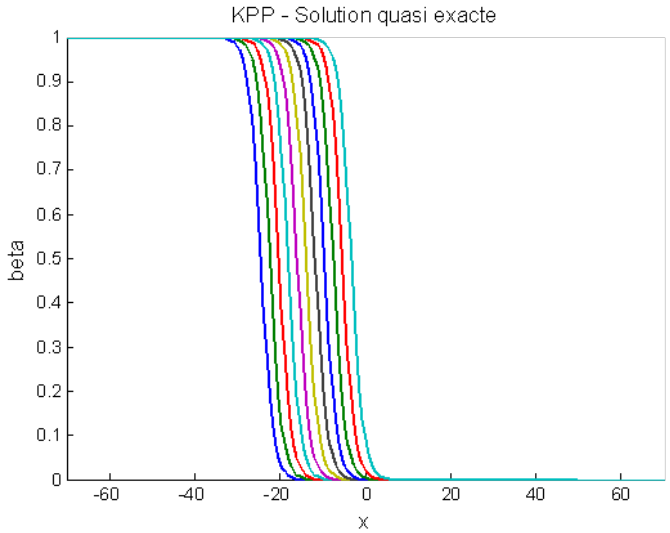
4.2.1 Résolution quasi-exacte du problème

Pour résoudre le système on a utilisé d'abord la méthode des lignes (*Method of Lines*, ou MOL). Le système d'équations différentielles ordinaires obtenu est résolu grâce à l'utilisation du solveur LSODE (*Livermore Solver for Ordinary Differential Equations*). Toutes les tolérances du solveur LSODE sont fixées à 1.10^{-15}, ce qui permet de s'assurer que les erreurs dues à la méthode numérique d'intégration temporelle sont négligeables devant toutes les autres erreurs que l'on va mesurer, et donc de pouvoir considérer que la solution donnée par LSODE est la solution "exacte" du système discrétisé spatialement.

Des études précédents [12; 1] montrent que, pour une discrétisation spatiale de 5000 points, l'erreur maximale obtenue par rapport à la solution analytique est de l'ordre de 1.10^{-5}. Cet ordre d'erreur justifie de prendre comme référence dans les calculs cette solution, que l'on appellera onde quasi-exacte (Figure 4.3), et non la solution analytique.

De même, on observe que cette erreur diminue lorsqu'on augmente la discrétisation. Cependant, un calcul avec 30000 points, par exemple, demande un temps de calcul dix fois supérieur au calcul avec 5000 points, pour obtenir un niveau de précision qui n'est pas nécessaire pour pouvoir observer des résultats intéressants dans le cadre de ce projet. Le choix de 5000 points constitue, de cette manière, un bon compromis, et il sera adopté tout au long de cette étude, sauf indication contraire.

FIG. 4.3: Solution quasiexacte, KPP.



4.2.2 Résolution du problème en appliquant l'algorithme pararéel.

Dans cette application les deux méthodes, fine et grossière, seront la méthode de Strang RDR avec LSODE comme méthode d'intégration temporelle (la méthode MOL ayant été utilisée pour obtenir le système d'EDO) . Par contre, les pas de splitting ainsi que les tolérances du solveur LSODE seront différents selon le solveur.

En général, la demarche sera la suivante :

- on divise le temps d'étude en $nc - 1$ intervalles égaux ΔT ; si le temps final est de T unités,
$$\Delta T = \frac{T}{(nc-1)};$$
- on applique la méthode grossière sur chaque intervalle avec un pas de temps fixé hc (pour ce cas particulier $hc = \Delta T$) (démarche *séquentielle*) ;
- et en suivant l'algorithme pararéel, on appliquera la méthode fine avec un autre pas de temps fixé hf ($hf = \Delta T/nf$) (démarche *parallèle*) ;

où nc et nf sont les nombres de points considérés sur tout le domaine temporel par la méthode grossière et fine, respectivement.

Par ailleurs, $tolg$ représente la tolérance du solveur LSODE prise en compte lorsque on intègre avec la méthode grossière. De même, $tolf$ représente la tolérance de la méthode fine. Les calculs qui suivent ont été faits sur 128 intervalles grossiers dans un environnement séquentiel, en tenant compte d'une possible parallélisation effective avec 128 processeurs ou plus, ce qui ne modifiera que les temps de calcul.

Premièrement, on considère la norme L^2 sur tout le domaine temporel de la solution pararéelle par rapport à la solution quasi-exacte. La figure 4.4 montre les résultats pour différents rapports de points fins et grossiers. Celui-là est équivalent au rapport entre les pas de splitting fin et grossier, le dernier étant fixe. Pour ce cas, on a considéré $tolg = 1.10^{-10}$ et $tolf = 1.10^{-15}$.

Il est clair que l'algorithme converge très vite (autour de 5 itérations) même pour des rapports assez grands (solveur plus grossier par rapport au fin). De plus, il faut tenir compte du fait que l'ordre d'erreur obtenu dans les plateaux est celui qu'on aurait eu si les calculs avaient été faits avec la méthode fine sur tout le domaine temporel [12]. Cela implique que l'algorithme pararéel converge bien vers la solution du solveur fin.

Il est intéressant de noter que le comportement itératif de l'algorithme est à peu près indépendant du rapport de pas, c'est-à-dire que pour les différents rapports de pas, l'approximation grossière (laquelle est la même pour tous) peut suivre l'évolution du phénomène lorsque le solveur fin progressivement la détaille plus précisément.

Si on analyse maintenant les erreurs en tenant compte du pas de temps fin (pas de splitting du solveur fin), on retrouve une droite de pente $1,999918$, ce qui correspond à l'ordre 2 de l'erreur globale de la méthode fine (méthode de Strang) [3; 2] (Figure 4.5).

Si l'on considère l'erreur L^2 pour différents instants, on obtient de même, le même ordre, comme le montre le tableau (4.3) et la figure 4.6.

Par ailleurs, on considère maintenant la vitesse de l'onde, laquelle a été calculée et moyennée pour 14 instants différents avec une approximation d'ordre 2 ; on obtient les résultats suivants (Figure 4.7).

En le comparant avec la vitesse de l'onde quasi-exacte, qui vaut $0,70710388$, on récupère une droite de pente $2,00372$ (Figure 4.8). C'est-à-dire, que l'algorithme reproduit de manière cohérente le comportement atteint par le solveur fin pas seulement au niveau de l'erreur L^2, mais aussi de la vitesse de l'onde [12; 1].

FIG. 4.4: Erreur L^2 de la solution pararéelle pour différents rapports de points fins et grossiers nf/nc, KPP.

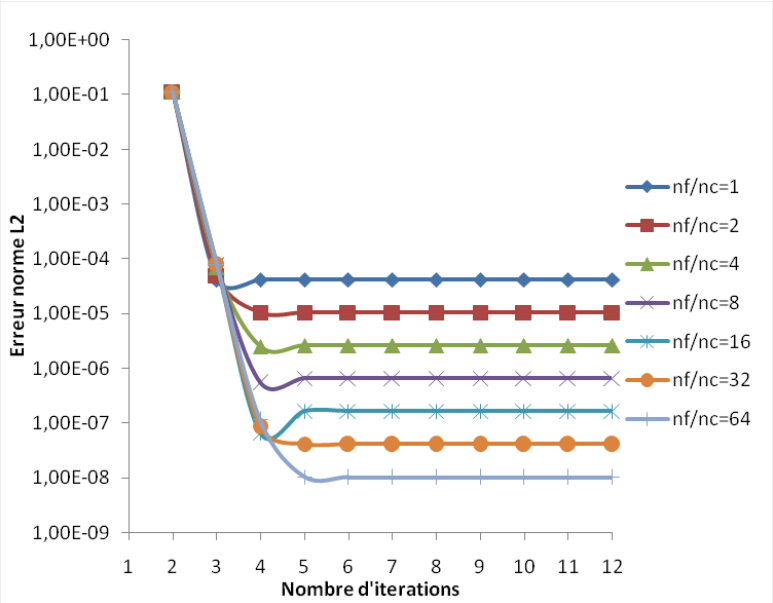
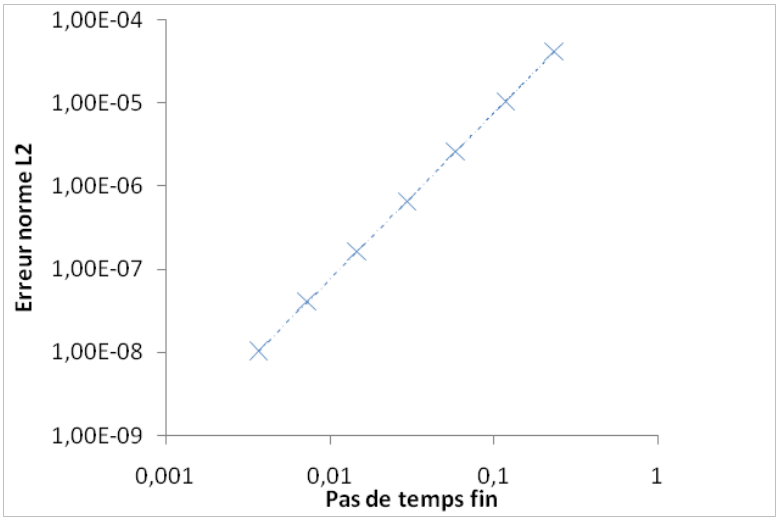


FIG. 4.5: Erreur L^2 en fonction du pas de temps fin, KPP.



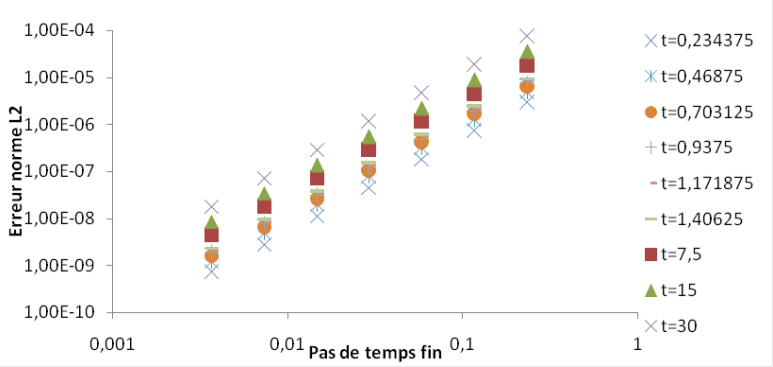
Suivant l'objectif de réduire le coût de calcul, on relaxe la tolérance du solveur grossier pour le rendre plus rapide. On voit que l'algorithme a du mal à converger et par contre, ses résultats restent autour l'approximation grossière. Ces calculs ont été faits avec les mêmes pas de temps, fin et grossier, de manière à ne considérer que l'influence de cette tolérance sur la convergence de l'algorithme (Figure 4.9).

En augmentant maintenant le rapport de pas, on obtient pour un rapport de 2 et 32, ce qui

TAB. 4.3: Ordre de l'erreur globale, KPP.

t	$ordre$
0,234375	1,999225
0,46875	1,996256
0,703125	1,999503
0,9375	1,999578
1,171875	1,99963
1,40625	1,999668
7,50	1,999818
15,00	1,999899
30,00	1,999919

FIG. 4.6: Erreur L^2 en fonction du pas de temps fin pour différents temps t, KPP.



montre la figure 4.10 suivante.

On s'aperçoit d'une légère amélioration, mais l'algorithme continue à avoir des problèmes de convergence. On a constaté que ce problème apparait comme conséquence des instabilités crées au niveau des frontières du domaine spatiale. Plus précisément, dès qu'on relaxe la tolérance du solveur, celui-ci a du mal à préserver les conditions de frontières définies (condition de Neumann homogène). Par ailleurs, cette contrainte limite notablement le choix du solveur grossier et constitue une barrière aux applications moins coûteuses.

FIG. 4.7: Vitesse de l'onde en fonction du pas de temps fin, KPP.

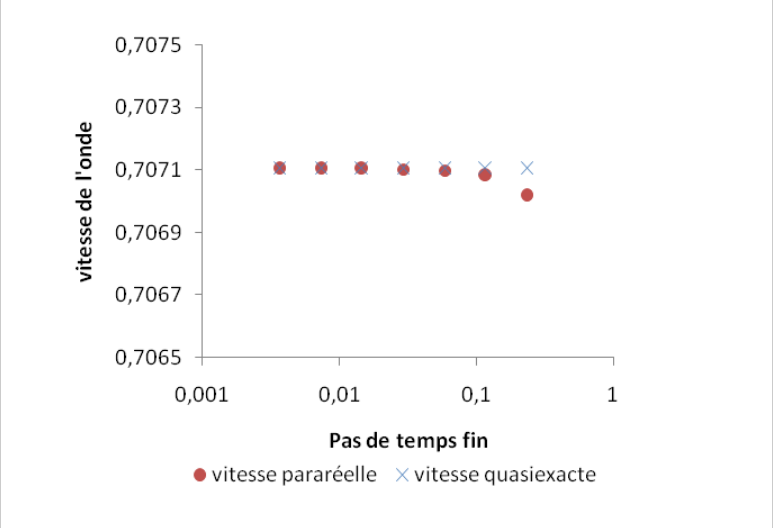


FIG. 4.8: Erreur de la vitesse en fonction du pas de temps fin, KPP.

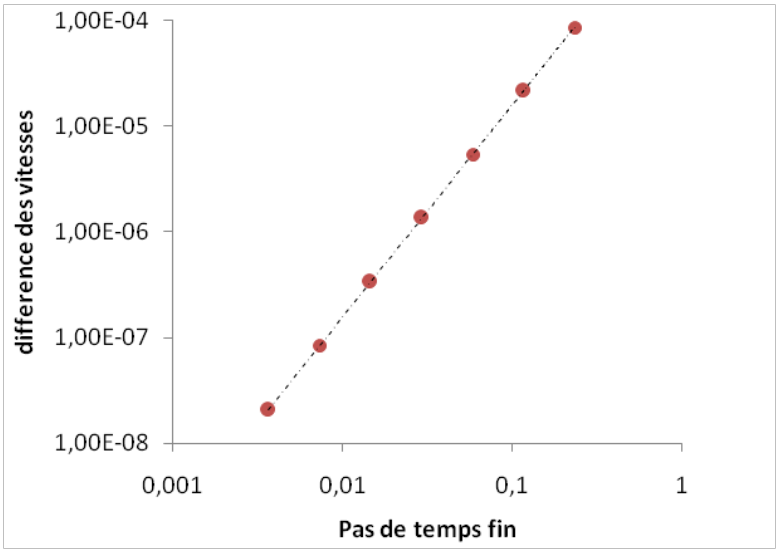


FIG. 4.9: Erreur L^2 en fonction des différentes tolérances grossières, KPP.

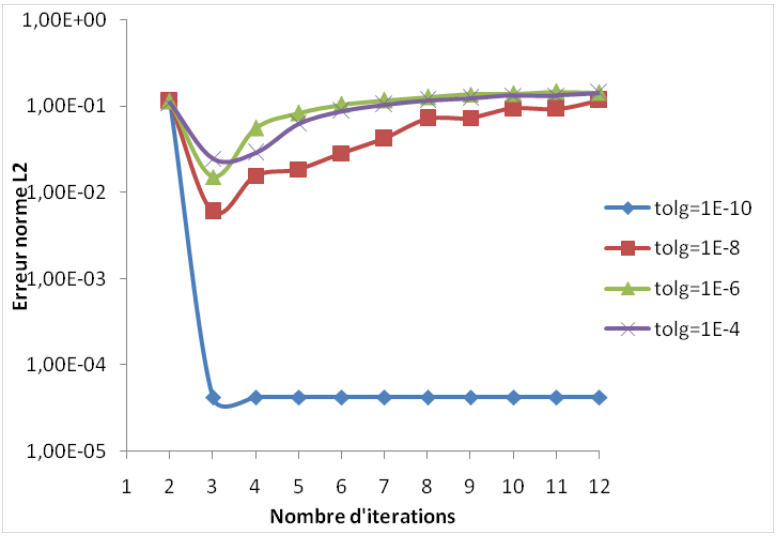
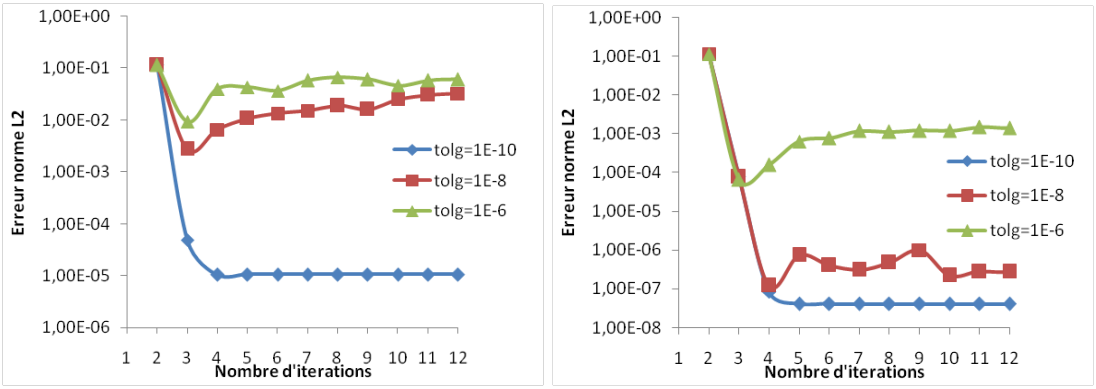


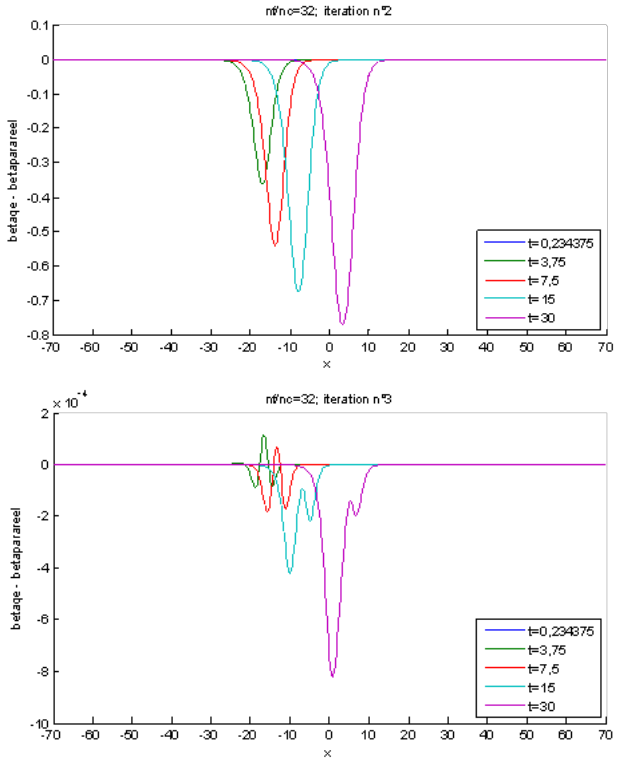
FIG. 4.10: Erreur L^2 en fonction des différentes tolérances grossières, KPP. $nf/nc = 2$ (gauche),$nf/nc = 32$ (droite).



4.2.3 Évolution itérative de l'erreur.

Dans les figures 4.11 et 4.12, on suit l'évolution de la différence entre la solution pararéelle et la quasi-exacte pour les instants $t = 0,234375;\ 3,75;\ 7,5;\ 15$ et 30. Le premier instant correspond au premier pas de temps grossier, et constitue l'erreur du solveur fin. On a considéré un rapport $nf/nc = 32$.

FIG. 4.11: Evolution itérative de l'erreur $\beta_{qe}(x,t) - \beta_{para}(x,t)$ (itérations 2-3), KPP.

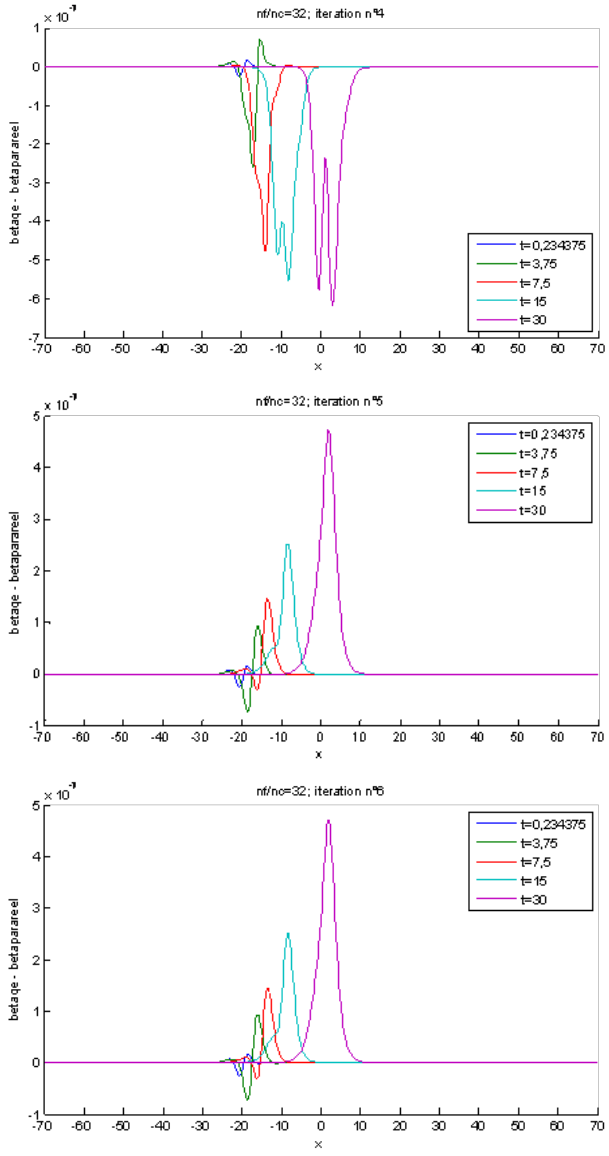


Cette différence pourrait être considérée comme la somme de deux différences, celle qu'il y a entre la solution quasi-exacte et la fine (résultat du solveur fin), et d'autre part, la différence entre les solutions fine et pararéelle ; autrement dit,

$$\beta_{qe}(x,t) - \beta_{para}(x,t) = \beta_{qe}(x,t) - \beta_{fine}(x,t) + \beta_{fine}(x,t) - \beta_{para}(x,t). \tag{4.2}$$

La première reste constante pendant toute l'évolution, alors que la deuxième décroît au fur et à mesure que l'algorithme tend vers la solution fine. Évidemment, une fois que la somme devient constante, on espère avoir convergé. De cette manière, d'après l'évolution obtenue, on voit que l'algorithme converge après cinq itérations.

FIG. 4.12: Evolution itérative de l'erreur $\beta_{qe}(x,t) - \beta_{para}(x,t)$ (itérations 4-6), KPP.



4.2.4 Évolution itérative de l'algorithme vers la solution fine.

Si l'on veut regarder le comportement de l'algorithme, il vaut mieux évaluer la différence entre la solution pararéelle et celle du solveur fin pour les mêmes instants $t = 3,75;\ 7,5;\ 15$ et 30, et pour le même rapport $nf/nc = 32$ (Figures 4.13 et 4.14).

En plus, en considérant l'erreur du solveur fin (Figure 4.15), on peut suivre aisément comment l'erreur est-elle structurée selon l'expression (4.2) et tirer les mêmes conclusions que précédement.

FIG. 4.13: Évolution itérative de l'algorithme vers la solution fine $\beta_{fine}(x,t) - \beta_{para}(x,t)$ (itérations 2-4), KPP.

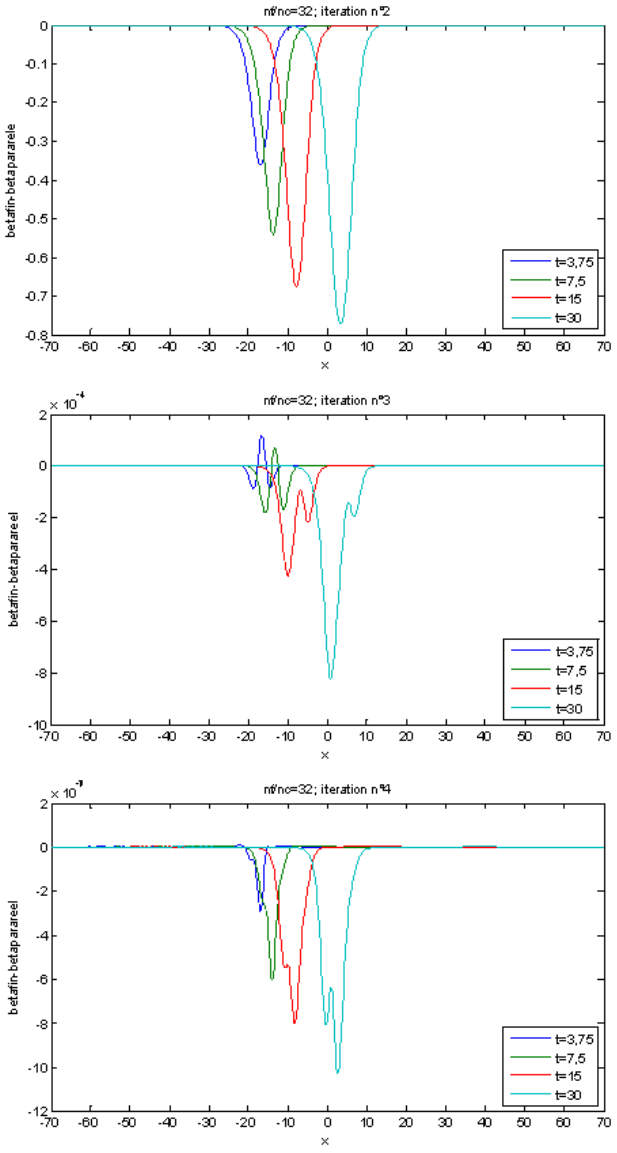


FIG. 4.14: Évolution itérative de l'algorithme vers la solution fine $\beta_{fine}(x,t) - \beta_{para}(x,t)$ (itérations 5-7, KPP.

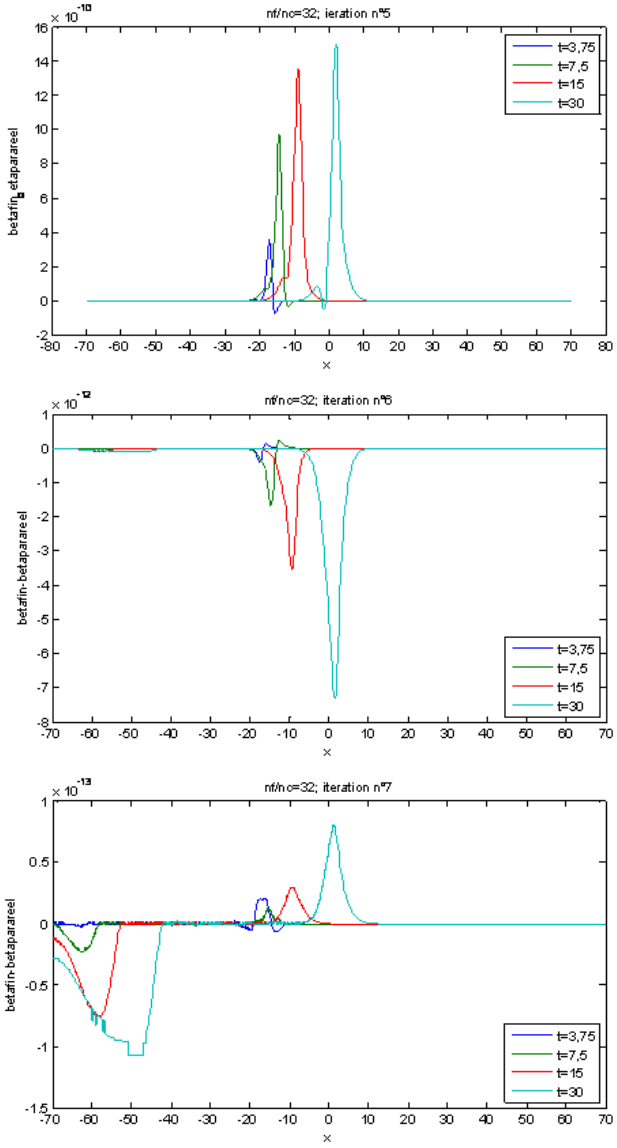
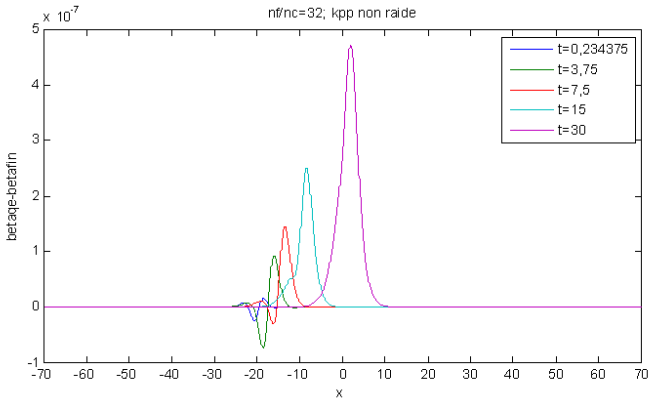


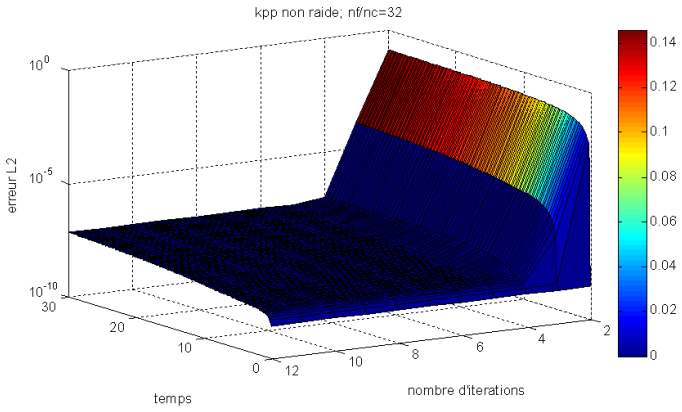
FIG. 4.15: Erreur entre la solution fine et la quasi-exacte $\beta_{qe}(x,t) - \beta_{fine}(x,t)$, KPP.



4.2.5 Évolution itérative de l'erreur L^2 et de la vitesse.

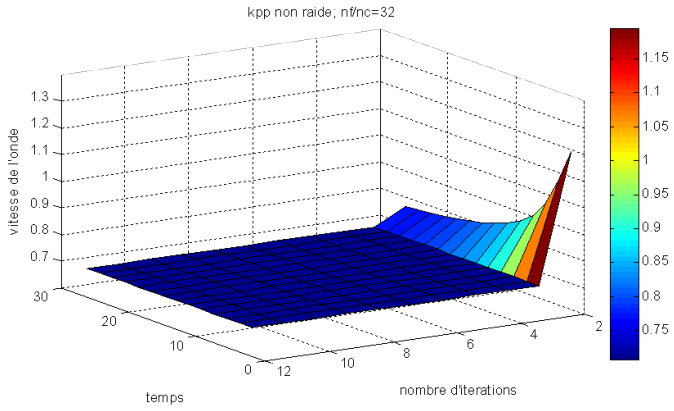
Lorsqu'on évalue l'erreur L^2 pour chaque instant, on voit bien que sa structure évolutive est bien représentée par l'erreur L^2 calculée avant sur tout le domaine temporelle. Par exemple, la figure 4.16 montre le cas $nf/nc = 32$.

FIG. 4.16: Évolution itérative de l'erreur L^2 de la solution pararéelle par rapport à la quasi-exacte, KPP.



Dès qu'on considère la vitesse de l'onde calculée pour différents instants, on voit comment le solveur grossier génère une onde plus rapide que l'onde quasi-exacte (ceci est cohérent avec l'évolution de la différence entre les deux solutions) et qu'au fur et mesure qu'on itère, le solveur fin finie par s'approcher d'une onde légèrement plus lente (Figure 4.17).

FIG. 4.17: Évolution itérative de la vitesse de l'onde, KPP.

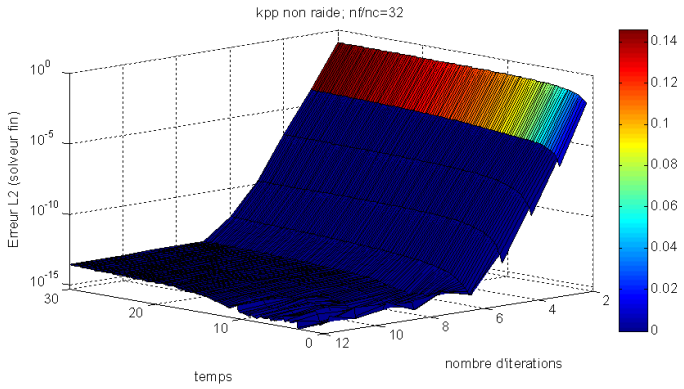


4.2.6 Convergence.

Maintenant, on évalue la différence entre l'approximation pararéelle et la solution du solveur fin en regardant l'erreur L^2 spatiale pour chaque instant. La figure 4.18 montre le cas $nf/nc =$

32.

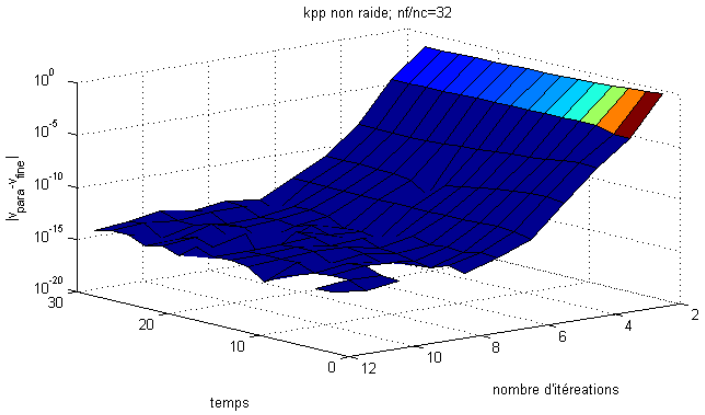
FIG. 4.18: Évolution itérative de l'erreur L^2 de la solution pararéelle par rapport à la fine, KPP.



On retrouve bien une structure similaire pour tout le domaine temporel ; la convergence est elle super linéaire et reproduit les résultats attendus par la théorie [4; 5].

La figure (4.19) montre l'évolution de la différence des vitesses pararéelle et fine. On s'aperçoit qu'elle a le même évolution que l'erreur L^2 (Figure 4.18) ; cela peut impliquer que ces deux sont étroitement liées. Cependant, d'après l'analyse de l'évolution des profils d'onde (section (4.2.4), ainsi qu'une comparaison d'ordre des différentes erreurs, on ne peut pas conclure si l'erreur est régie par le défaut de vitesse d'onde ou par l'erreur des profils spatiaux. On voit seulement que les deux ont un comportement évolutif semblable et que l'algorithme corrige ces deux défauts au même temps. On peut tirer la même conclusion en regardant l'évolution dans le plan de phase (voir annexe A.1).

FIG. 4.19: Évolution itérative de la différence des vitesses pararéelle et fine, KPP.



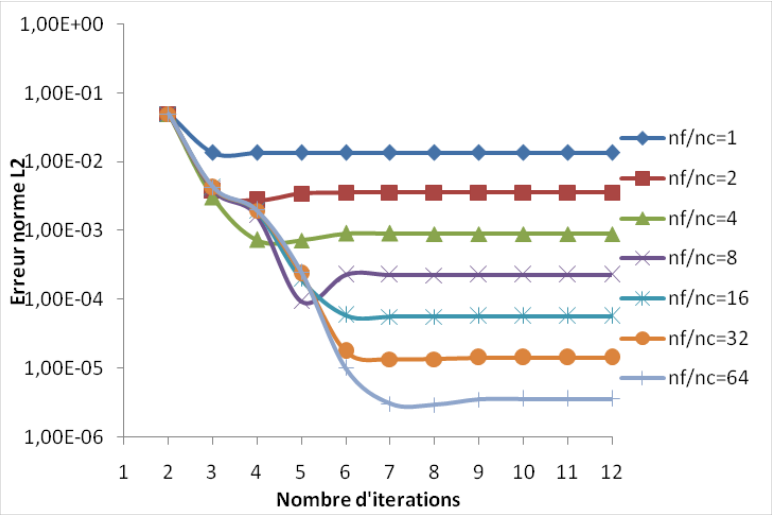
4.3 KPP raide.

C'est le cas avec $k = 10$ et $D = 0,1$, c'est-à-dire, avec la *même* vitesse de l'onde mais un gradient maximum dix fois plus *grand*. Cette fois-ci, la raideur est associée directement à la présence de forts gradients spatiaux.

De la même manière, on considère d'abord la norme L^2 sur tout le domaine temporel de la solution pararéelle par rapport à la solution quasi-exacte.

La figure 4.20 montre les résultats pour différents rapports de points fins et grossiers. On a considéré $tolg = 1.10^{-10}$ et $tolf = 1.10^{-15}$.

FIG. 4.20: Erreur L^2 de la solution pararéelle pour différents rapports de points fins et grossiers nf/nc, KPP raide.



Il est clair qu'encore une fois l'algorithme converge très vite (autour de 7 itérations pour le pire cas). Encore une fois, l'ordre d'erreur obtenu est celui qu'on aurait eu si les calculs avaient été faits avec la méthode fine sur tout le domaine temporel [12]. Remarquons que la précision atteinte est limitée par la méthode fine, et cette fois-ci, celle-ci est dégradée par rapport au cas non raide. Néanmoins, d'après les résultats, l'algorithme pararéel converge bien vers son solveur fin.

Dès qu'on analyse les erreurs en fonction du pas de temps fin, on retrouve une droite de pente $1,993746$, ce qui corresponde à l'ordre 2 de l'erreur globale de la méthode fine (méthode de Strang) [3; 2] (Figure 4.21).

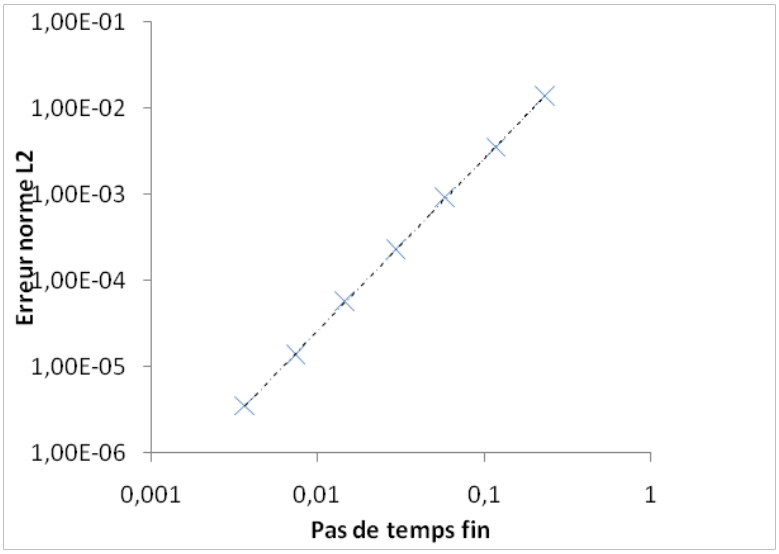
En considèrant l'erreur L^2 pour différents instants, on obtient le même ordre, comme le montre le tableau (4.4) et la figure 4.22.

Si l'on considère maintenant la vitesse de l'onde, laquelle a été calculée et moyennée pour 14 instants différents avec une approximation d'ordre 2, on obtient les résultats suivants (Figure 4.23).

En la comparant avec la vitesse de l'onde quasi-exacte, qui est de $0,70687274$, on récupère une droite de pente de $2,045459$ (Figure 4.24).

Encore une fois, l'algorithme réussit à obtenir les mêmes résultats que ceux obtenus par l'application directe du solveur fin, soit au niveau de l'erreur L^2, soit au niveau de la vitesse de l'onde.

FIG. 4.21: Erreur L^2 en fonction du pas de temps fin, KPP raide.



Si maintenant, on relaxe la tolérance du solveur grossier, on voit que l'algorithme évolue de la même manière, avec les pas d'intégration égaux, ou différents (8 dans ce cas en particulier). Par ailleurs, les vitesses d'onde résultantes sont les mêmes. Dans ce cas on ne retrouve pas le problème qu'on avait avant à partir du moment où le front de l'onde reste toujours assez éloigné des frontières et empêche la création des instabilités aux bords pendant les itérations pararéelles. Ceci peut nous permettre de prendre des tolérances plus faibles et de gagner en temps de calcul (Figure 4.25).

TAB. 4.4: Ordre de l'erreur globale, KPP raide.

t	$ordre$
0,234375	1,979958
0,46875	1,981891
0,703125	1,972982
0,9375	2,010346
1,171875	1,987691
1,40625	1,988657
7,50	1,99081
15,00	2,003515
30,00	2,04073

FIG. 4.22: Erreur L^2 en fonction du pas de temps fin pour différents temps t, KPP raide.

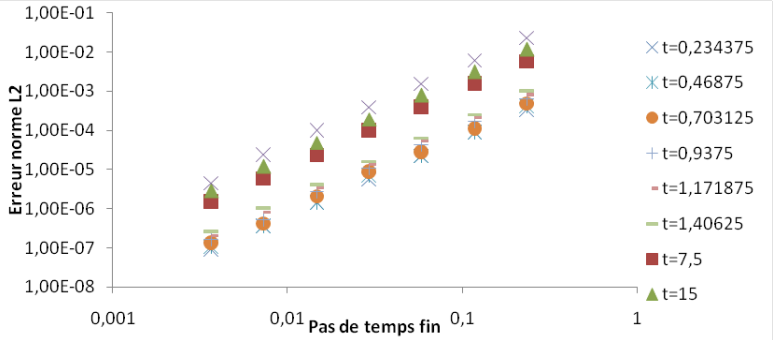


FIG. 4.23: Vitesse de l'onde en fonction du pas de temps fin, KPP raide.

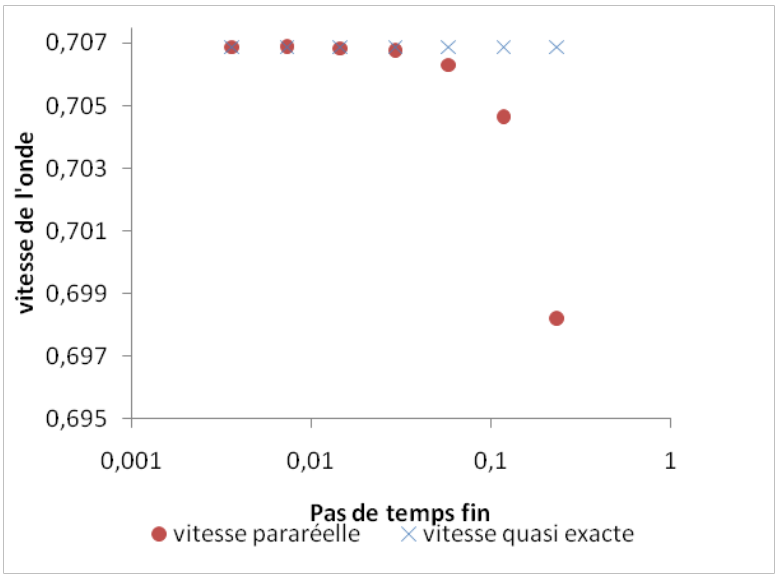


FIG. 4.24: Erreur de la vitesse en fonction du pas de temps fin, KPP raide.

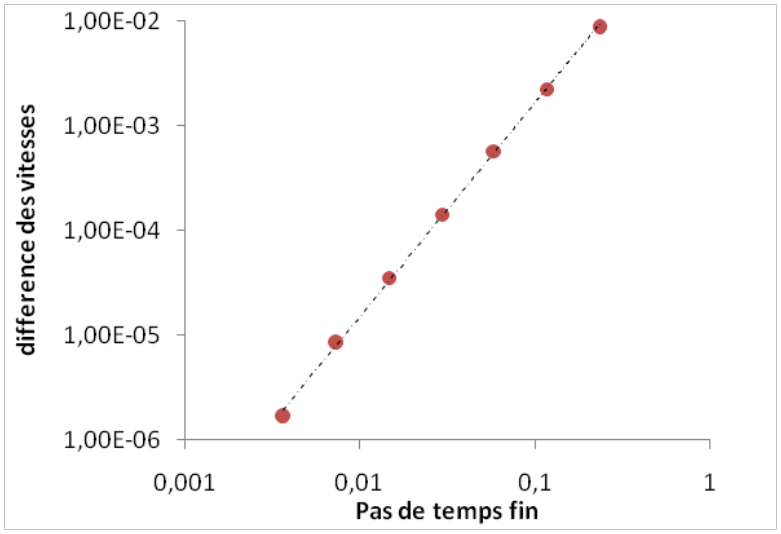
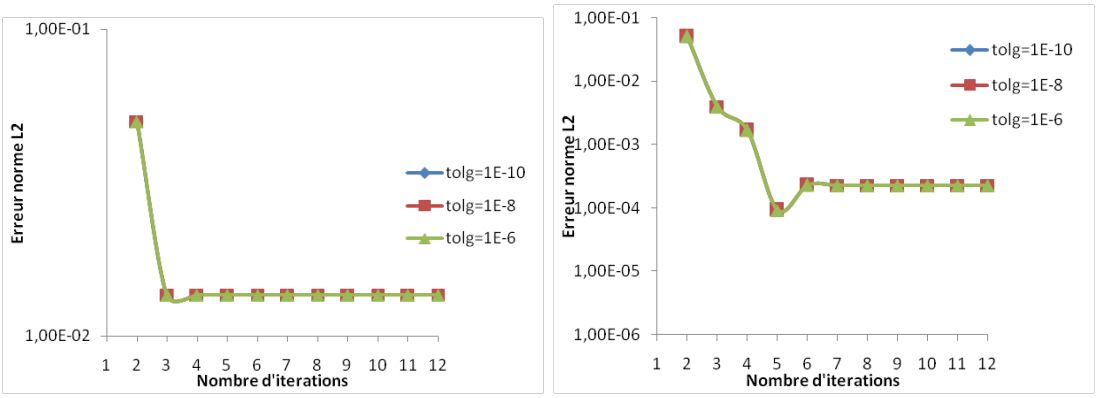


FIG. 4.25: Erreur L^2 en fonction des différentes tolérances grossières, KPP raide. $nf/nc = 1$ (gauche),$nf/nc = 8$ (droite)



4.3.1 Évolution itérative de l'erreur.

On effectue les mêmes calculs que le cas raide pour les mêmes instants et le même rapport (Figures 4.26 et 4.27).

On constate que l'algorithme a un comportement différent quand on avance sur le domaine temporelle. En fait, pour les premiers pas de temps, on récupère le même comportement que précédement, c'est-à-dire, les erreurs initiales d'approximation sont progressivement atténuées jusqu'à retrouver l'erreur du solveur fin, selon l'expression (4.2). Cependant, pour des temps plus longs, les corrections successives faites par l'algorithme ne suffisent pas pour reconstruire le phénomène avec le même taux à partir de l'approximation initial grossière.

FIG. 4.26: Evolution itérative de l'erreur $\beta_{qe}(x,t) - \beta_{para}(x,t)$ (itération 2-5), KPP raide

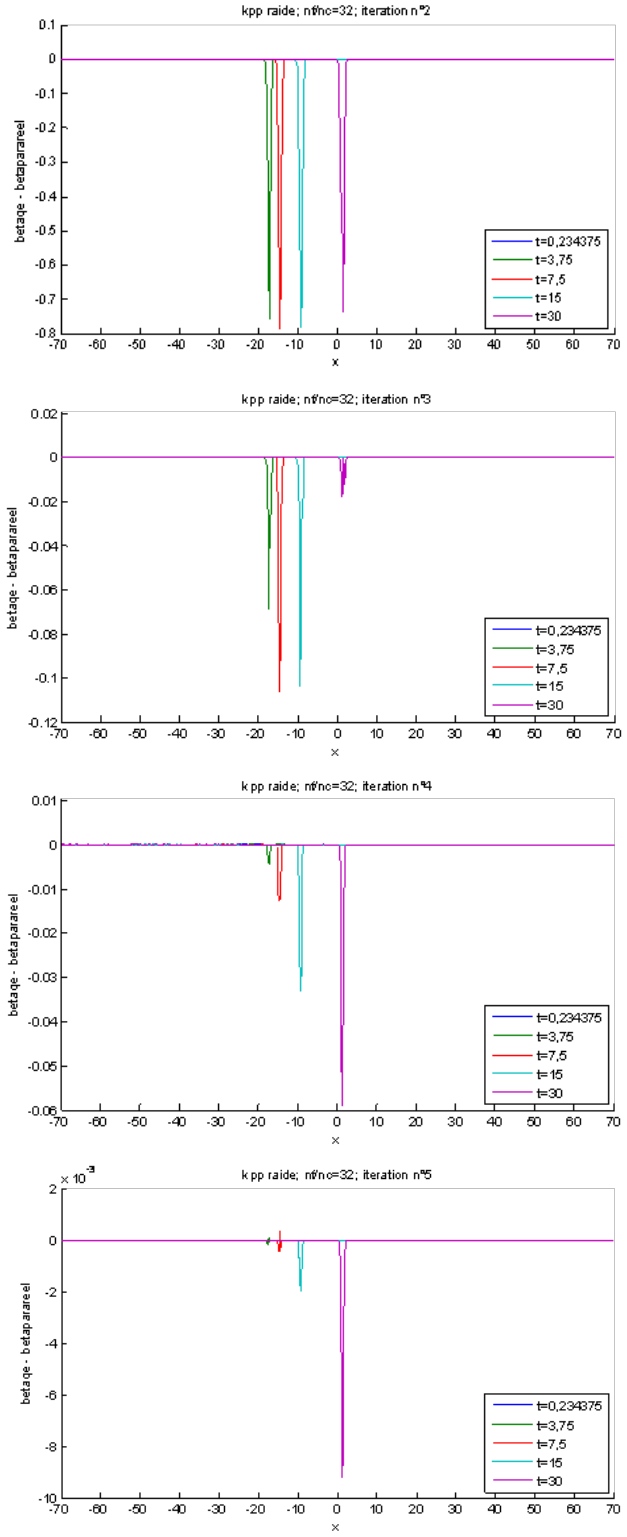
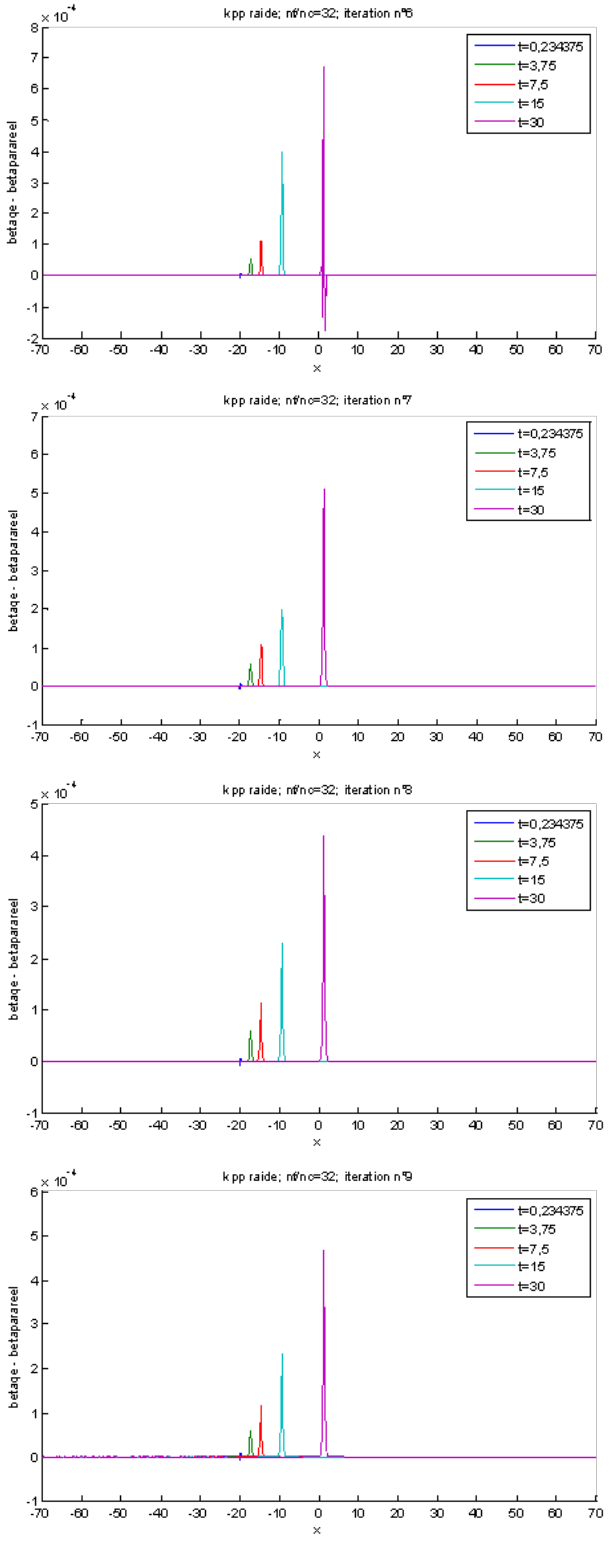


FIG. 4.27: Evolution itérative de l'erreur $\beta_{qe}(x,t) - \beta_{para}(x,t)$ (itération 6-9), KPP raide



4.3.2 Évolution itérative de l'algorithme vers la solution fine.

En regardant l'évolution itérative vers la solution fine (pour les mêmes instants et le même rapport (Figures 4.28et 4.29), ainsi que l'erreur du solveur fin (Figure 4.30), on peut reconstruire aisément la structure de l'erreur.

Cette fois-ci, la différence entre les solutions pararéelle et fine devient à peu près négligeable à partir de la septième itération ; cependant, la vitesse de convergence est plus faible. En fait, la présence de forts gradients spatiaux fait que l'algorithme a du mal à décrire précisement l'onde retrouvée par la méthode fine, parce qu'un tout petit décalage entre les deux fronts d'onde introduit des différences plus importantes, surtout à la fin du domaine temporel.

FIG. 4.28: Évolution itérative de l'algorithme vers la solution fine $\beta_{fine}(x,t) - \beta_{para}(x,t)$ (itération 2-5), KPP raide

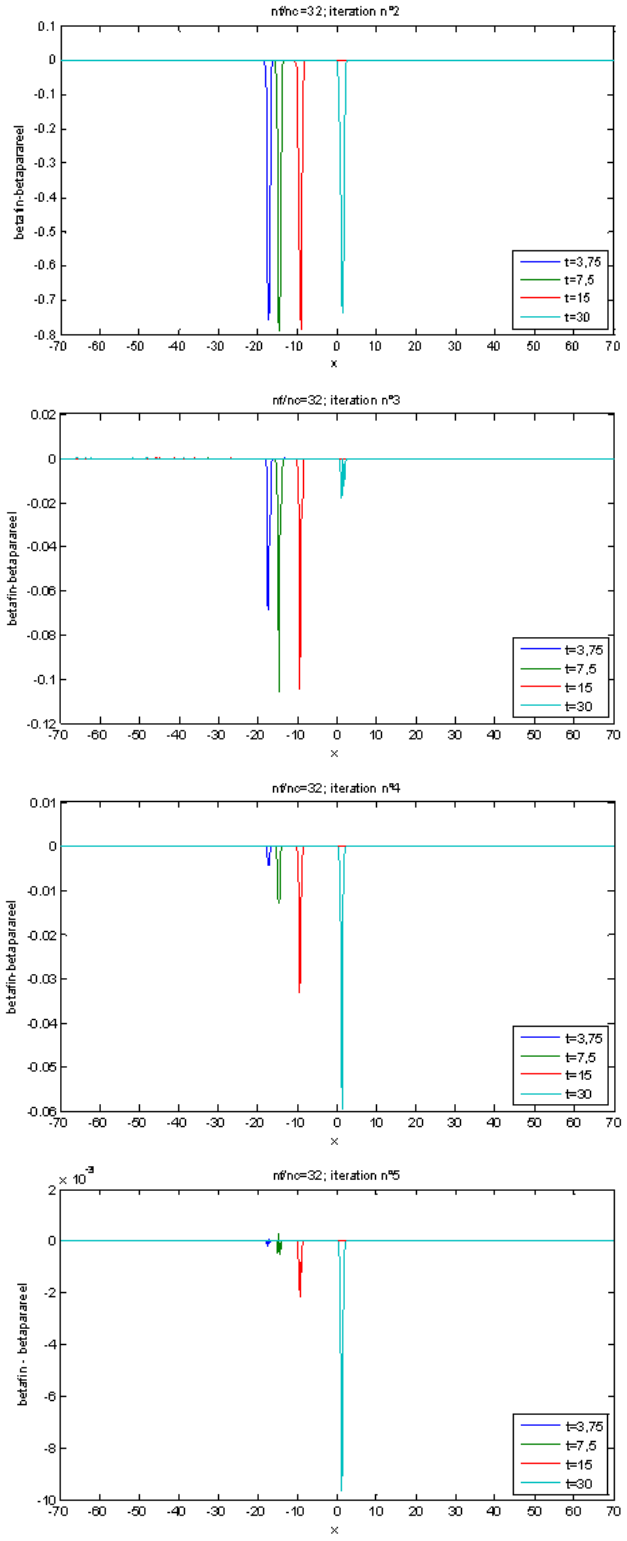


FIG. 4.29: Évolution itérative de l'algorithme vers la solution fine $\beta_{fine}(x,t) - \beta_{para}(x,t)$ (itération 6-9), KPP raide

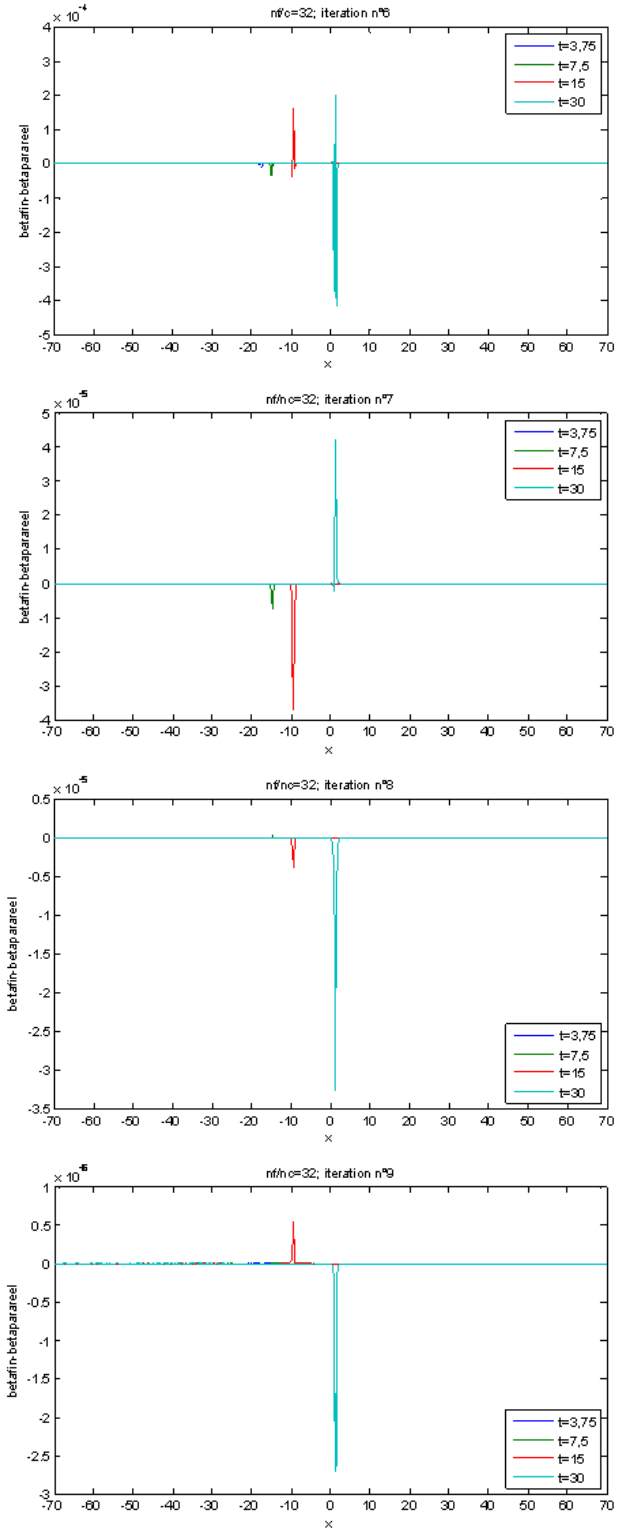
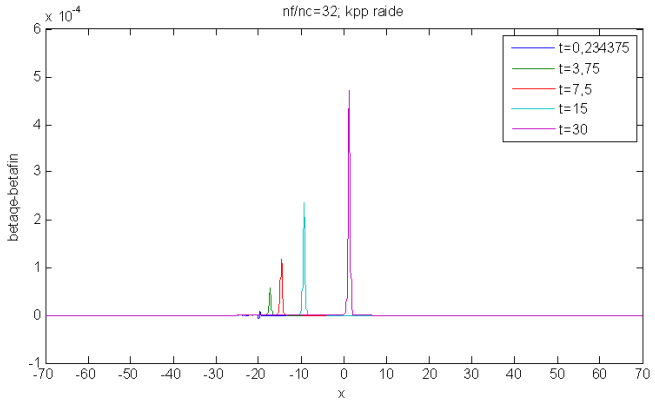


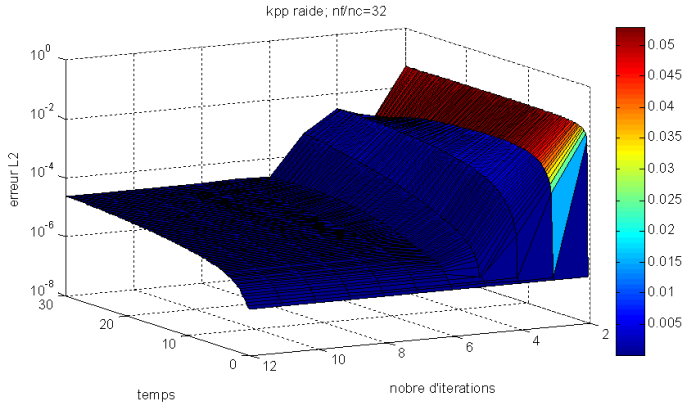
FIG. 4.30: Erreur entre la solution fine et la quasi-exacte, KPP raide.



4.3.3 Évolution itérative de l'erreur L^2 et de la vitesse.

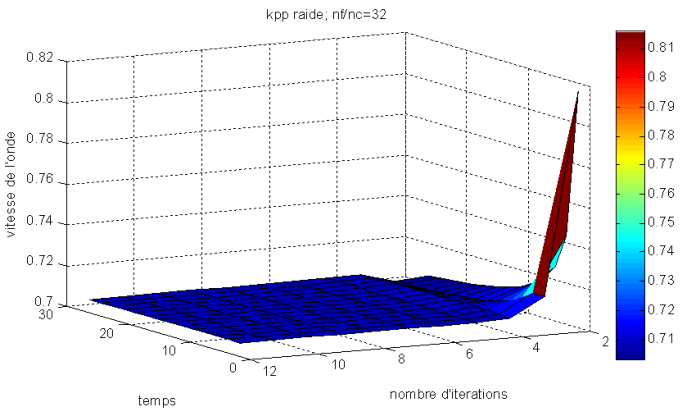
Maintenant, pout le cas raide, on s'aperçoit que l'évolution de l'erreur n'est plus la même sur tout le domaine temporel (Figure 4.31) et que, à partir d'un instant donné, elle change considérablement. Ceci crée le comportement vu lorsqu'on considérait la norme L^2 sur tout le domaine (Figure 4.20).

FIG. 4.31: Evolution itérative de l'erreur L^2 de la solution pararéelle par rapport à la quasi-exacte, KPP raide



L'évolution de la vitesse (Figure 4.32) montre comment les approximations pararéelles initiales compensent rapidement le défaut de vitesse lié à l'approximation initiale grossière pour les pas de temps éloignés. Cela explique la soudaine amélioration de l'erreur L^2 à ces endroits, que l'on avait aperçu pendant les premières itérations. Finallement, les fronts évoluent jusqu'à leurs valeurs finales, lesquelles sont au-dessous de la valeur quasi-exacte. Ce comportement est tout à fait cohérent avec les résultats vus pour les différences des profils d'onde.

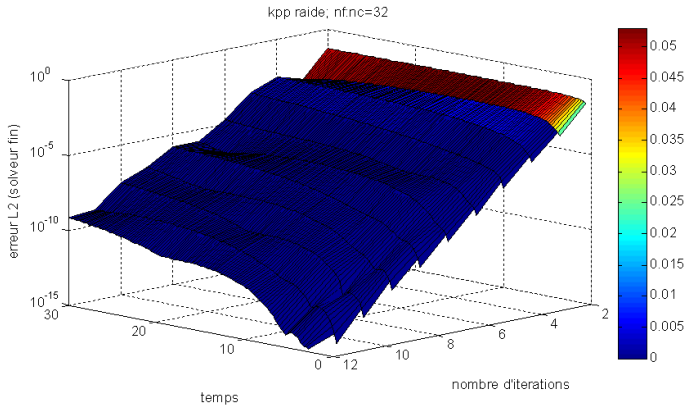
FIG. 4.32: Évolution itérative de la vitesse de l'onde, KPP raide.



4.3.4 Convergence.

En considérant l'erreur de l'algorithme par rapport au solveur fin, on voit le comportement dans la figure 4.33.

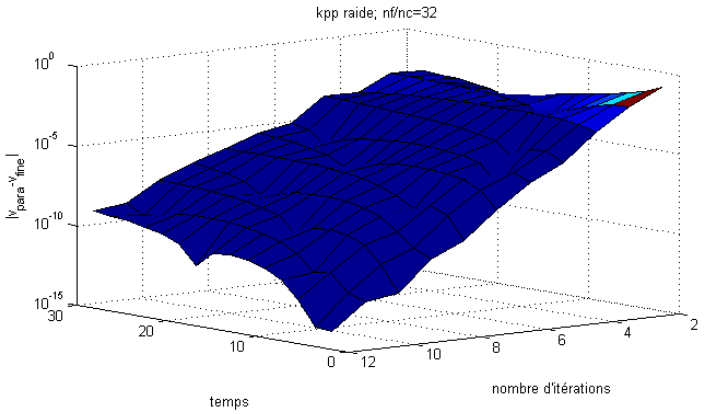
FIG. 4.33: Evolution itérative de l'erreur L^2 de la solution pararéelle par rapport à la fine, KPP raide



La vitesse de convergence est considérablement ralentie et, on ne voit plus de convergence super linéaire. Evidement, ce phénomène est dû à la présence de forts gradients spatiaux qui fait qu'une correction plus précise des approximation est plus difficile à obtenir le long du front d'onde.

D'après la figure (4.34) qui montre l'évolution de la différence des vitesses pararéelle et fine, on récupère une situation pareille au cas non raide précédent. C'est-à-dire, elle a aussi le même développement que l'erreur L^2 (Figure 4.33) et basée sur l'analyse de l'évolution des profils d'onde (section 4.3.2) ainsi que sur l'ordre des erreurs, on ne peut pas déterminer si l'erreur est régie par le défaut de vitesse d'onde ou par l'erreur des profils spatiaux. Par ailleurs, on peut aussi tirer la même conclusion en regardant l'évolution dans le plan de phase (voir annexe A.2).

FIG. 4.34: Evolution itérative de la différence des vitesses pararéelle et fine, KPP raide



4.4 Modèle BZ. Premier cas.

Le modèle étudié est un système d'équations de réaction-diffusion, lequel est représenté par l'équation (1.39).

Comme $\mu \ll \epsilon$, on analysera d'abord le système avec deux variables (1.40), où b constitue la variable rapide du système, et c, la lente, avec $\epsilon = 0,01$.

Les autres paramètres ont été pris de la manière suivante : $f = 3$ et $q = 2.10^{-4}$, avec des coefficients de diffusion $D_b = 1$ et $D_c = 0,6$. Le temps d'étude est de 2 unités et on considère le cas monodimensionnel dans le domaine $[0, 80]$.

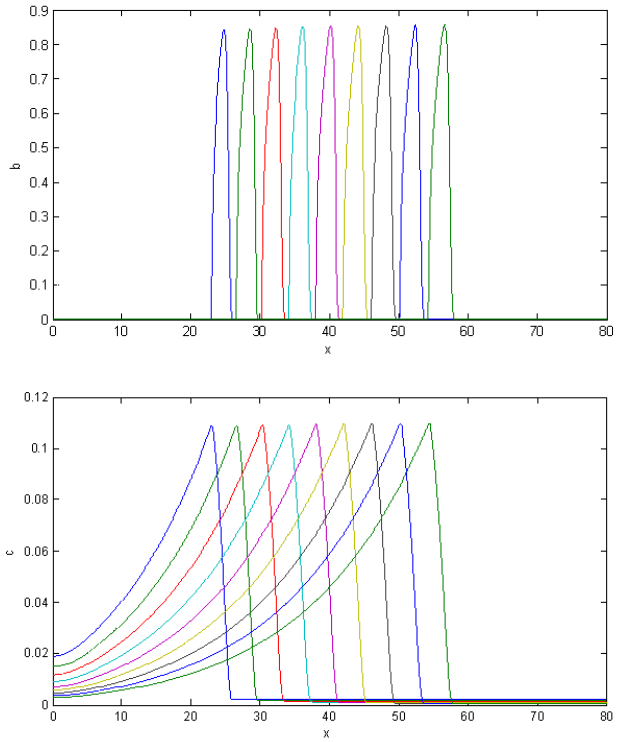
4.4.1 Résolution quasi-exacte du problème.

Pour résoudre le système on a utilisé d'abord la méthode des lignes (*Method of Lines*, ou MOL). Le système d'équations différentielles ordinaires obtenu est résolu grâce à l'utilisation du solveur LSODE (*Livermore Solver for Ordinary Differential Equations*).

Dans une première approche on a considéré une discrétisation spatiale de 4000 points.

L'évaluation du solveur LSODE avec un pas d'intégration de $2/512$ et des tolérances fixées à 1.10^{-10} donne les résultats montrés dans la figure 4.35.

FIG. 4.35: Solution quasi-exacte, BZ. Variable b en haut, c en bas



On a constaté que le solveur LSODE a du mal à résoudre le système avec des pas de temps d'évaluation plus grands que $2/512$. Et même pour ce cas, les tolérances ne peuvent pas être fixées très petites, car le solveur risque de ne pas converger. Néanmoins, pour évaluer les erreurs on prendra un pas de temps et des tolérances plus petites que celles prises ici.

4.4.2 Résolution du problème en appliquant l'algorithme pararéel.

Dans cette application les deux méthodes, fine et grossière, seront la méthode de Strang RDR avec des pas de temps de splitting différents. L'intégration temporelle est faite par le solveur LSODE pour le pas de réaction ainsi comme pour la diffusion. Les tolérances ont été fixées à 1.10^{-10}.

Des études précédentes [12] montrent que, pour un pas de splitting de $2/128$, la vitesse ainsi que le profil de l'onde sont assez perturbés. Néanmoins, dans un premier essai, on a pris ce pas de splitting pour le solveur grossier. Pour le solveur fin, les pas de splitting ont été fixés plus petits avec un rapport de 2, 4, 8 et 16 avec le grossier pour différents cas.

L'algorithme n'arrive pas à converger pour aucun des cas. Voyons par exemple le profils des solutions pour le cas avec un rapport de pas 16 pour cinq instants différents (Figures 4.36 et 4.37).

On voit bien que l'algorithme n'est pas capable de reproduire l'évolution du phénomène.

Si on essaie à nouveau avec un pas de splitting cette fois-ci de $2/256$ (c'est-à-dire, la moitié du cas précédent) pour la méthode grossière et les mêmes rapports vus avant, on récupère le même comportement. Les figures 4.38 et 4.39 montrent les résultats pour 8 instants différents et un rapport de pas de 16.

Le solveur grossier n'arrive pas à approximer l'évolution du phénomène, et de plus, l'itération de l'algorithme pararéel perturbe énormément le développement de l'onde.

En fait, si l'on considère le système dynamique associé au système d'équations, on s'aperçoit que, bien que pour les paramètres donnés il y a un point stationnaire stable (figure 1.6), l'évolution a un comportement différent qui dépend des conditions initiales. Ceci est la conséquence du fait que l'on a une variable rapide due à la présence du paramètre ϵ et de la structure particulière des variétés d'équilibre associées.

Les variétés d'équilibre étant données par les expressions (1.27), on les représente dans le plan de phase (courbes verte et rouge, respectivement dans la figure 4.40) et on considère plusieurs cas avec des conditions initiales différentes pour la variable lente c.

Si le point initial reste au-dessous du point minimum A (équation (1.30) et figure 1.6), il y a des sauts dans la variable rapide b qui créent des boucles jusqu'à arriver au point stationnaire. Cette particularité est propre aux *milieux excitables*.

Ces cas-ci reproduisent les conditions présentes aux bords pour le cas couplé avec la diffusion, dès qu'on peut la négliger à ces endroits d'après les conditions de frontière considérées. En fait, on a bien constaté que ces pertubations commencent effectivement aux bords du domaine.

FIG. 4.36: Solution pararéelle perturbée (pas grossier $2/128$), BZ variable b

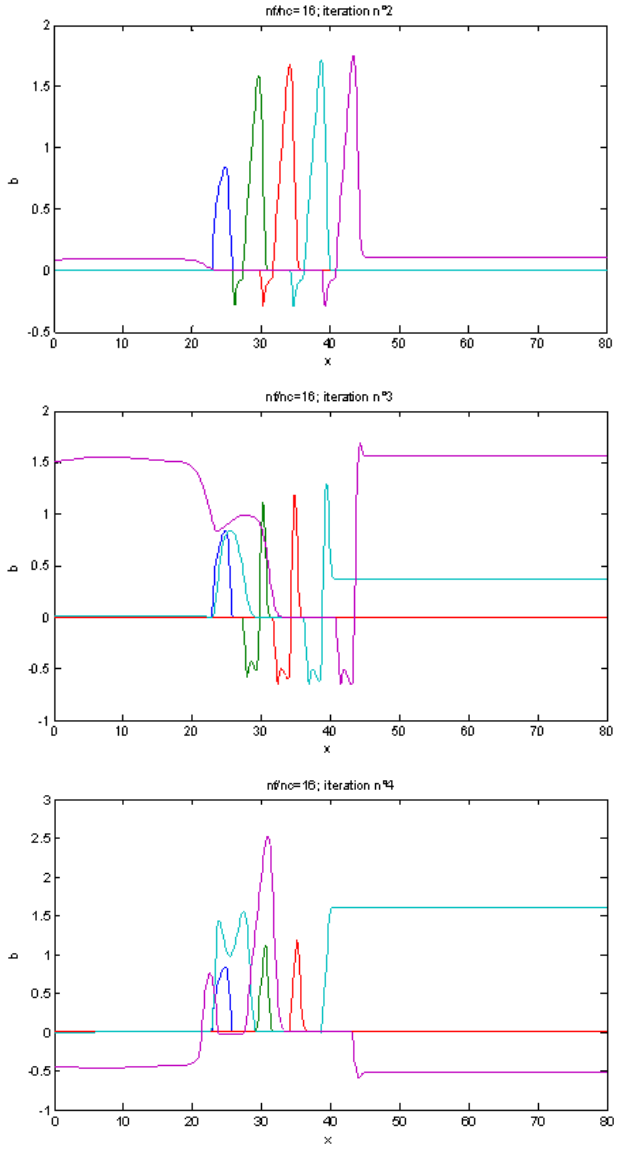


FIG. 4.37: Solution pararéelle perturbée (pas grossier $2/128$), BZ variable c

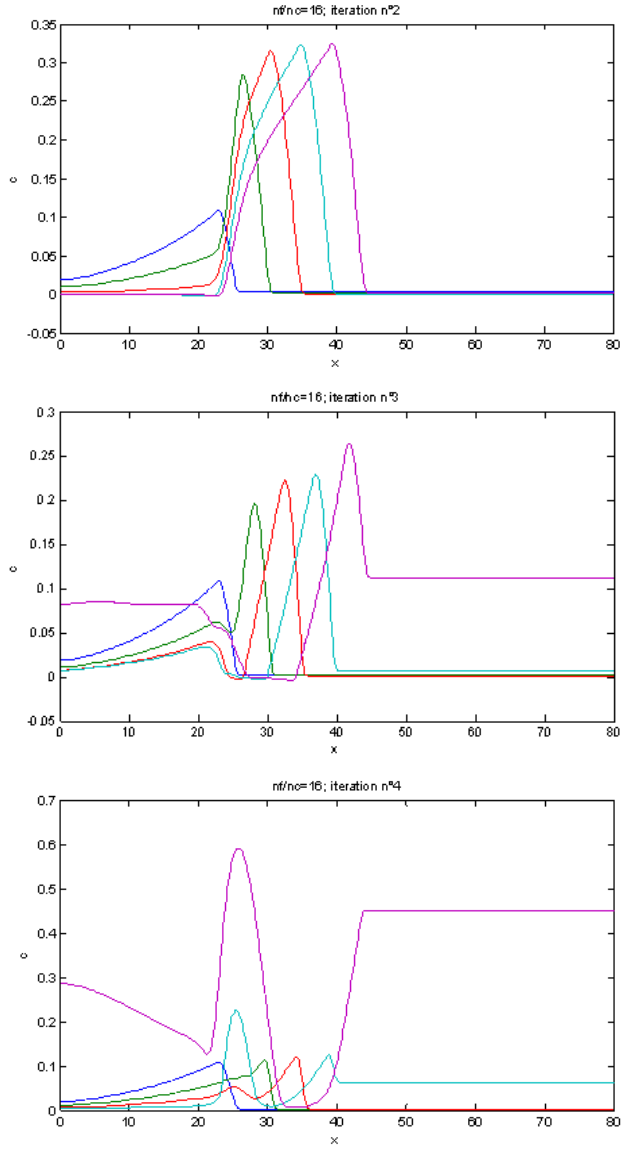


FIG. 4.38: Solution pararéelle perturbée (pas grossier $2/256$), BZ variable b

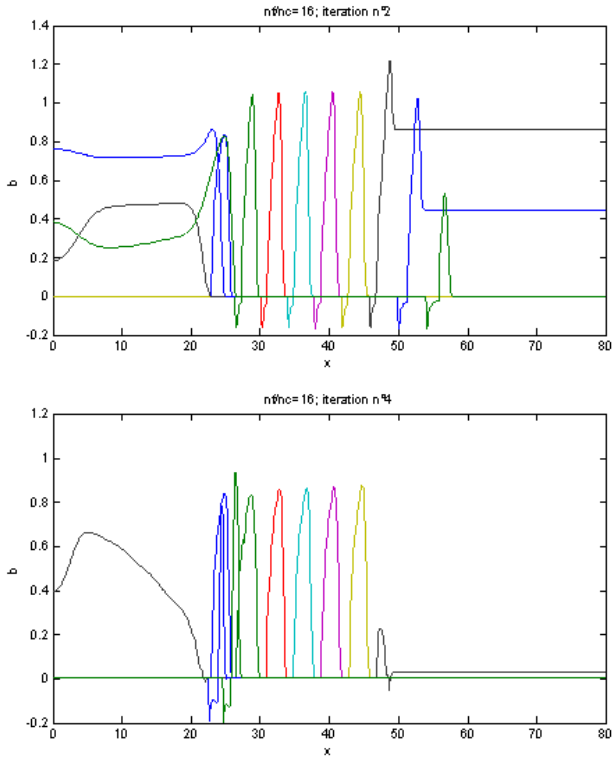


FIG. 4.39: Solution pararéelle perturbée (pas grossier $2/256$), BZ variable c

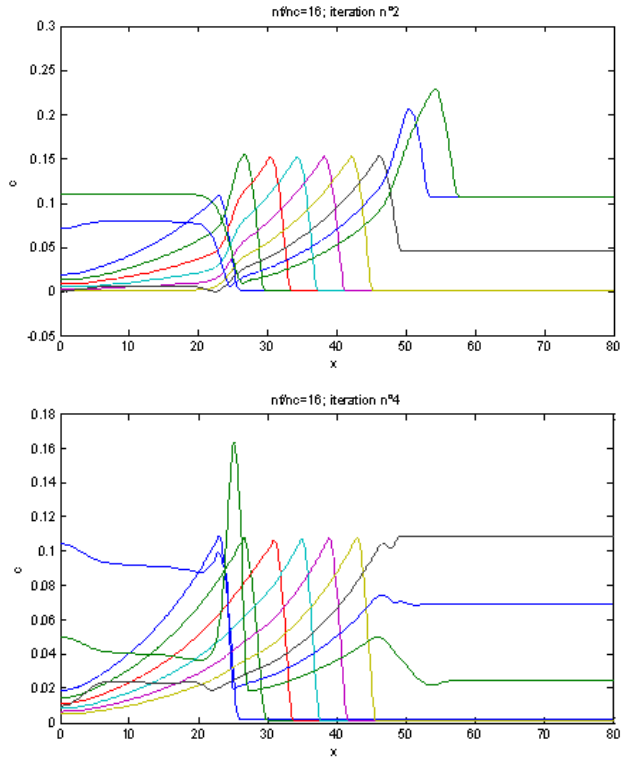
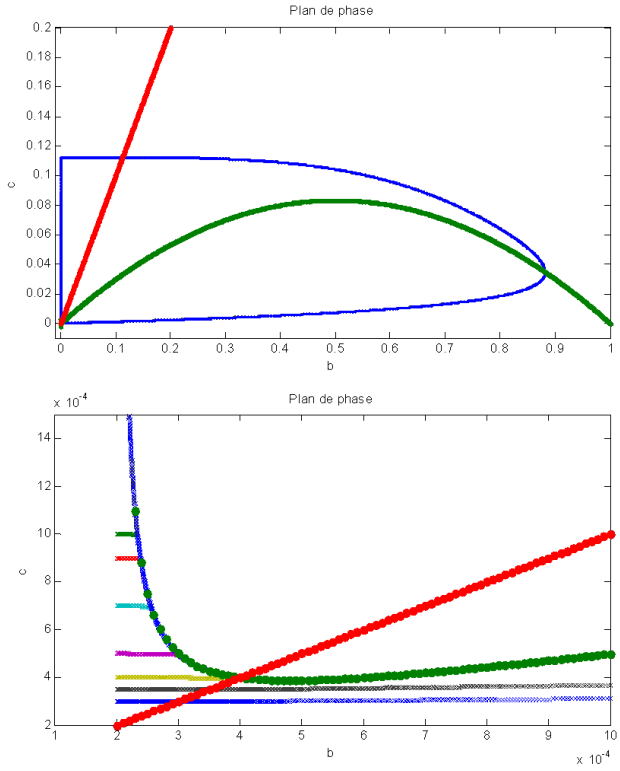


FIG. 4.40: Évolution dans le plan de phase (b, c), BZ. Figure agrandie en bas.

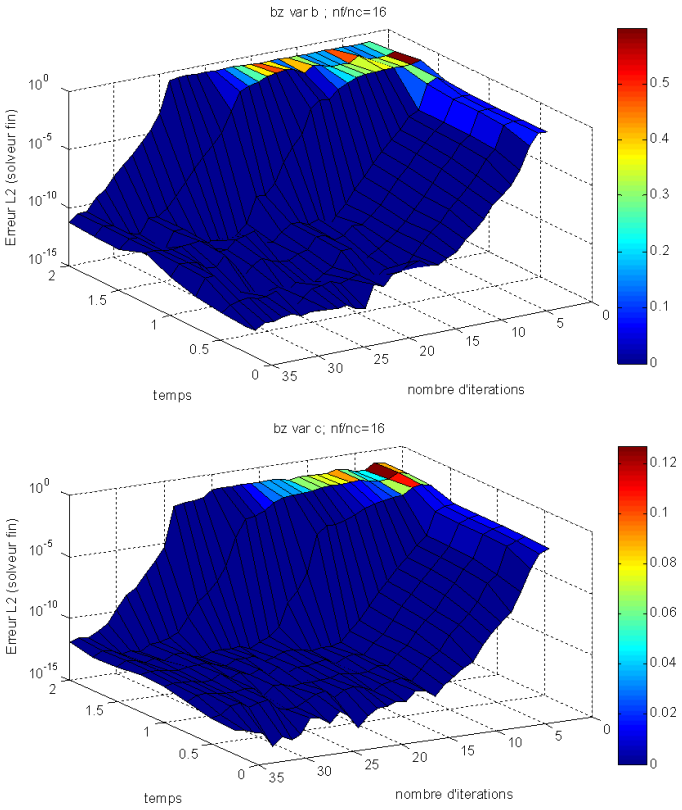


4.4.3 Analyse de convergence.

Maintenant, on prend un pas de splitting pour lequel on est sûr que le solveur grossier permet de retrouver des solutions dans la zone d'ordre 2, d'après des résultats des études précédentes [12].

Pour évaluer la convergence de l'algorithme, on calcule l'erreur L^2 par rapport à la solution du solveur fin, avec un pas de splitting de $2/512$ pour le solveur grossier et 16 fois plus petit pour le fin. Ces erreurs ayant été calculées pour 8 instants également séparés dans le domaine temporel d'étude (figure 4.41).

FIG. 4.41: Évolution itérative de l'erreur L^2 de la solution pararéelle par rapport à la fine, BZ. Variable b en haut, c en bas.

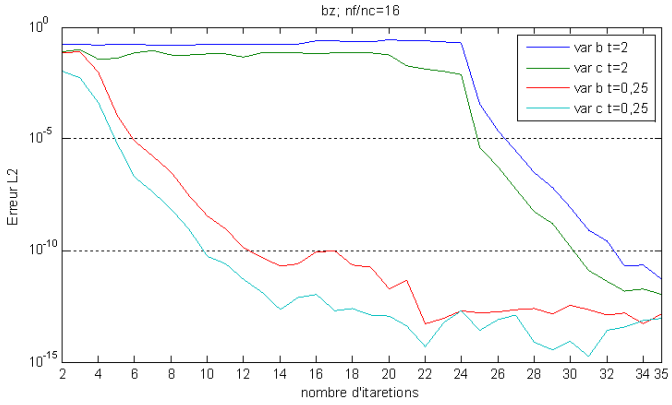


On s'aperçoit que l'algorithme prend à peu près 24 itérations seulement pour reconstruire le profil de l'onde initialement perturbé sur tout le domaine.

Si l'on regarde plus précisément l'erreur pour les limites du domaine ($t = 0,25$ et $t = 2$), on récupère le même comportement itératif, dès que le profil de l'onde a été reconstruit pour le deuxième temps considéré (figure 4.42). Néanmoins, les perturbations introduites initialement font qu'une analyse de l'évolution itérative n'est pas la plus appropriée à ce cas, ce qui nous a amené à considérer des méthodes de résolution différentes.

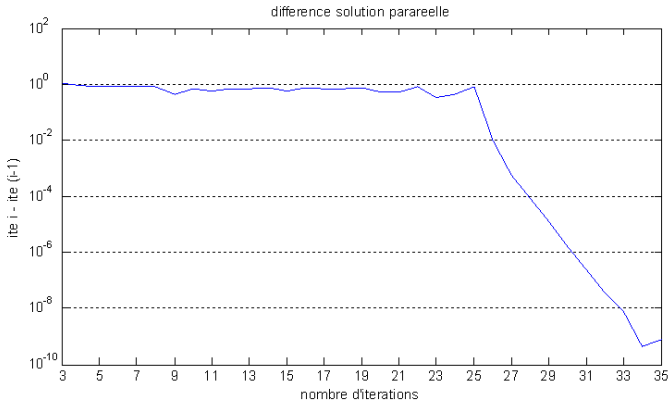
Le critère d'arrêt a été fixé selon la différence maximale entre deux solutions pararéelles d'itérations successives, lesquelles sont représentées dans la figure 4.43. Même si l'algorithme réussit à converger, l'influence des perturbations initiales introduites est évidente et représente

FIG. 4.42: Évolution itérative de l'erreur L^2 de la solution pararéelle par rapport à la fine sur les limites temporelles, BZ.



par ailleurs un coût assez élevé.

FIG. 4.43: Évolution itérative du critère de convergence, BZ.



4.5 Modèle BZ. Deuxième cas.

On avait vu les problèmes qui se posent lorsqu'on essaye d'appliquer l'algorithme au système (1.40) ; ceux-ci sont dus aux caractéristiques particulières du système dynamique associé et à la méthode d'intégration choisie (LSODE).

En effet, LSODE utilise la méthode BDF, laquelle est une méthode multi-pas ; comme celle-ci approche les premiers points avec une précision plus faible, et du fait que chaque intervalle constitue un problème indépendant, ce procesus se répète sur tout le domaine temporel et favorise considérablement l'apparition des perturbations.

L'utilisation d'une méthode mono-pas appropiée aux problèmes raides est vraiment conseillée. En fait, lorqu'on intégre le pas de réaction avec RADAU5, en laissant LSODE pour la diffusion, la performance de l'algorithme est notablement améliorée.

4.5.1 Résolution du problème en appliquant l'algorithme pararéel. Première approche.

De même que pour le cas antérieur, les deux méthodes, fine et grossière, seront la méthode de Strang RDR, avec des pas de temps de splitting différents. Cependant, pour l'intégration temporelle du pas de réaction on utilise le solveur RADAU5 et pour la diffusion, SDIRK4. Le choix de la deuxième, une méthode *L*-stable, convient très bien, car celle-ci est très performante pour atténuer les échelles associées aux nombres d'onde élevés, dues généralement aux erreurs numériques, lors de la discrétisation de l'operateur de diffusion [14].

Dans une première approche, les tolérances ont été fixées à 1.10^{-5}. Les figures 4.44 et 4.45 montrent les résultats pour un pas de splitting de $2/512$ pour le solveur grossier et 16 fois plus petit pour le fin.

Le profil de l'onde, initialement perturbé, est rapidement reconstruit à partir de la troisième itération. Par ailleurs, ces perturbations ont été attenuées par rapport au cas précédent (Figure 4.38 et 4.39).

FIG. 4.44: Solution pararéelle (Radau5-Sdirk4), première approche, BZ. Variable $b(x,t)$

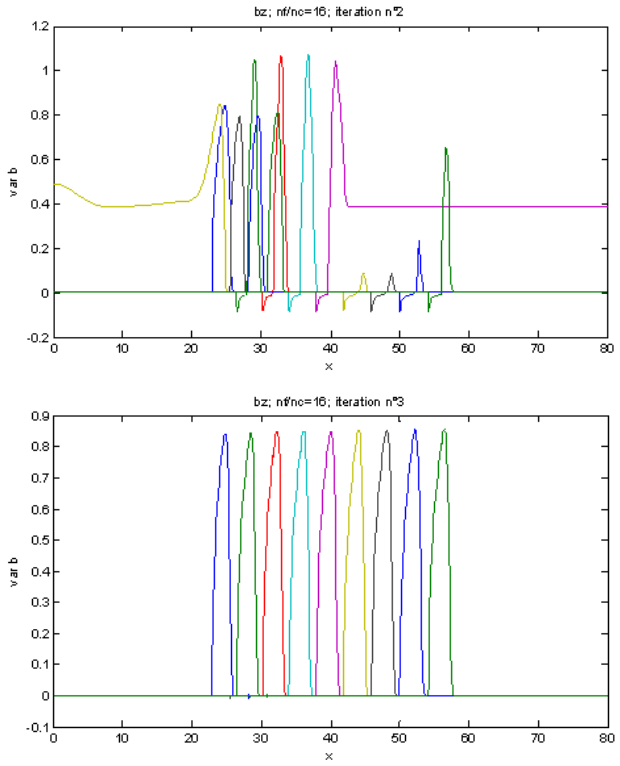
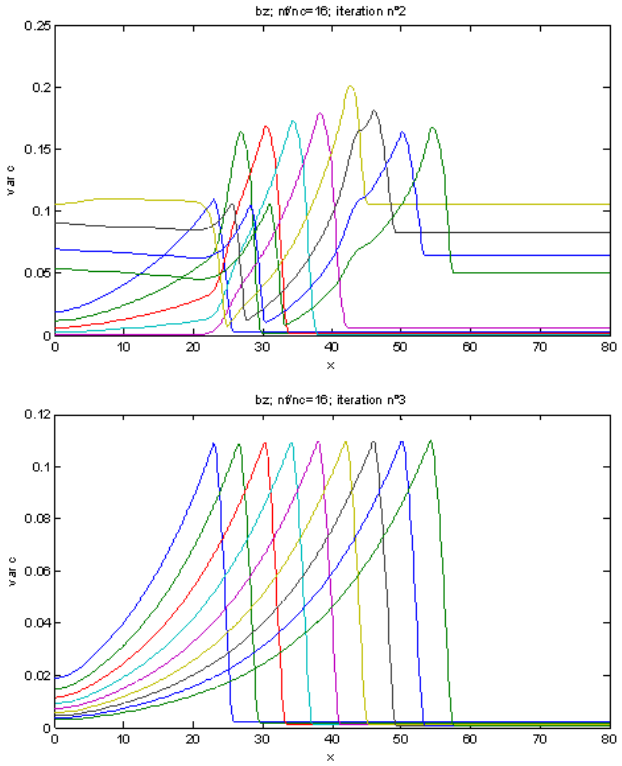


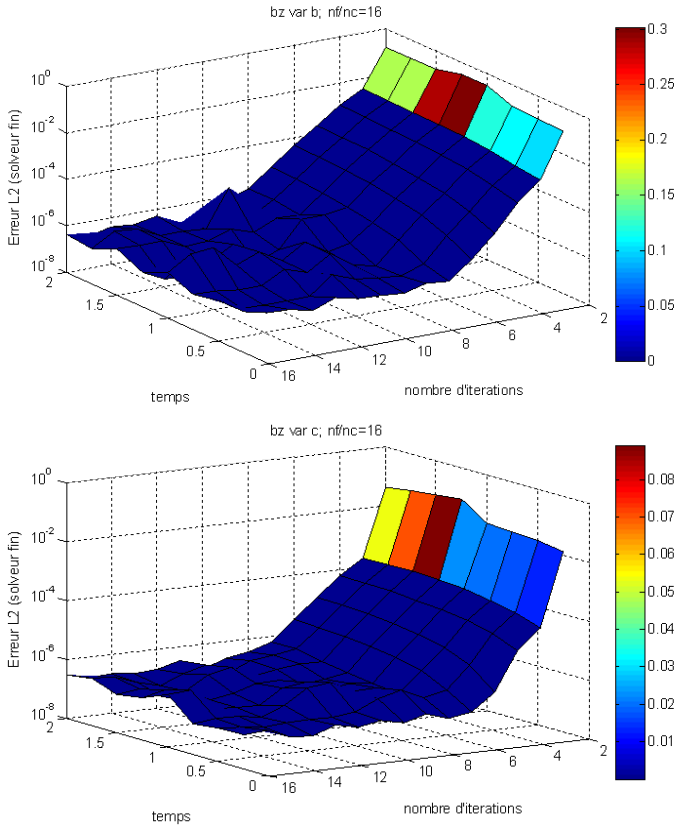
FIG. 4.45: Solution pararéelle (Radau5-Sdirk4), première approche, BZ. Variable $c(x,t)$



4.5.2 Analyse de convergence.

Pour évaluer la convergence de l'algorithme, on calcule l'erreur L^2 par rapport à la solution du solveur fin. Ces erreurs sont calculées pour 8 instants également séparés dans le domaine temporel d'étude (figure 4.46). On voit que la phase initial de correction a été complètement enlevée.

FIG. 4.46: Évolution itérative de l'erreur L^2 de la solution pararéelle par rapport à la fine (Radau5-Sdirk4), BZ. Variable b en haut, c en bas



Si l'on regarde plus précisément l'erreur pour les limites du domaine ($t = 0,25$ et $t = 2$), on voit que le taux de convergence devient plus faible au fur et à mesure que le temps avance (figure 4.47).

Le critère d'arrêt a été fixé en se basant sur la différence maximale entre deux solutions d'itérations successives, qui sont représentées dans la figure 4.48. On va voir que l'algorithme converge autour la sixième itération ; au-delà de la huitième itération, on ne voit plus que l'algorithme continue à s'approcher de la solution fine et les erreurs retrouvées entre les solutions fine et pararéelle restent du même ordre. En tout cas, les tolérances sont assez faibles et on peut les considérer comme conséquence de ce fait ; en fait, le solveur Radau5 retrouve des problèmes pendant l'intégration temporelle, alors que cette situation n'apparaît pas lorsqu'on résout le système directement sans appliquer l'algorithme.

FIG. 4.47: Évolution itérative de l'erreur L^2 de la solution pararéelle par rapport à la fine sur les limites temporelles (Radau5-Sdirk4), BZ.

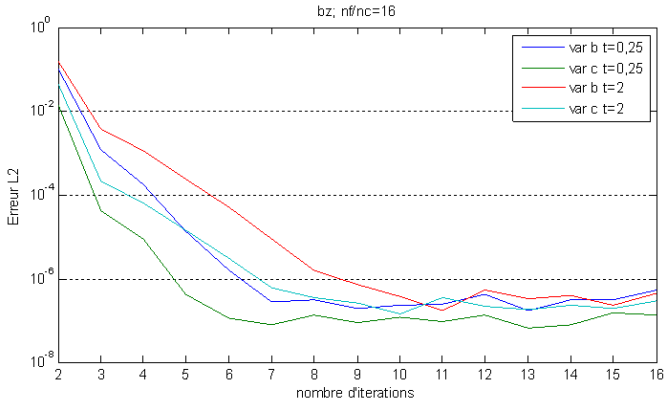
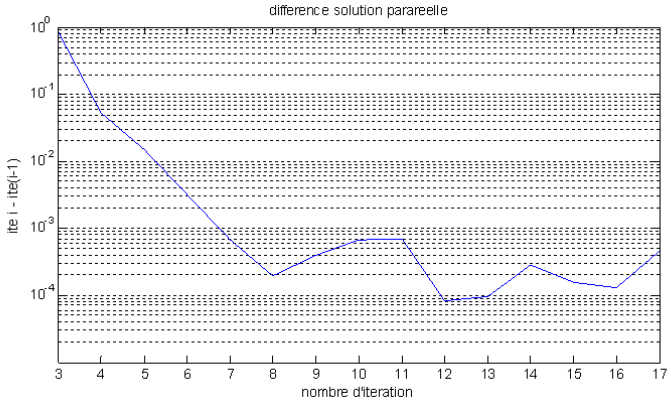


FIG. 4.48: Évolution itérative du critère de convergence (Radau5-Sdirk4), BZ.



4.5.3 Évolution itérative de l'algorithme vers la solution fine.

Maintenant, on regarde la différence entre les profils des ondes, fine et pararéelle (Figures 4.49 et 4.50).

Si on les compare avec la différence entre les profils quasi-exacte et fin (figure 4.51), on voit comment ceux-ci deviennent du même ordre pour la cinquième itération. La structure de l'erreur est aisément reconstruite.

FIG. 4.49: Évolution itérative de l'algorithme vers la solution fine $b_{fine}(x,t) - b_{para}(x,t)$ (Radau5-Sdirk4), BZ

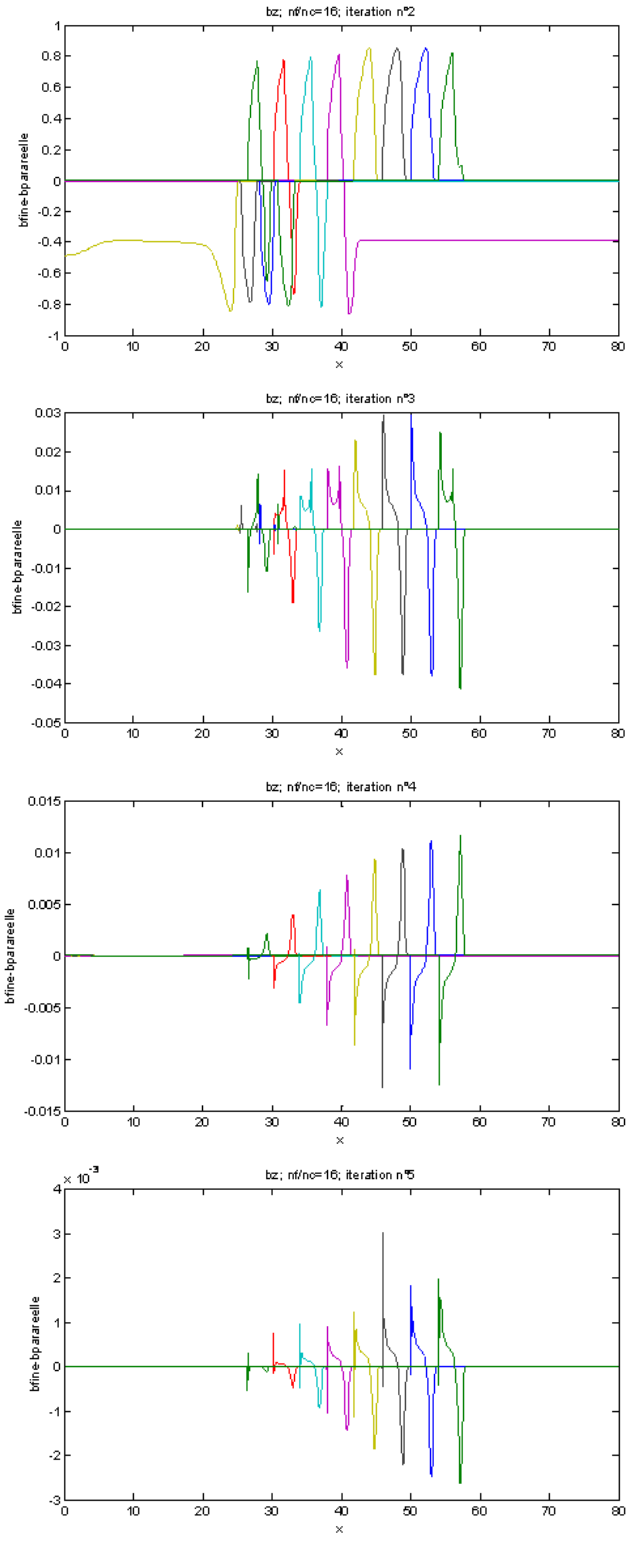


FIG. 4.50: Évolution itérative de l'algorithme vers la solution fine $c_{fine}(x,t) - c_{para}(x,t)$ (Radau5-Sdirk4), BZ

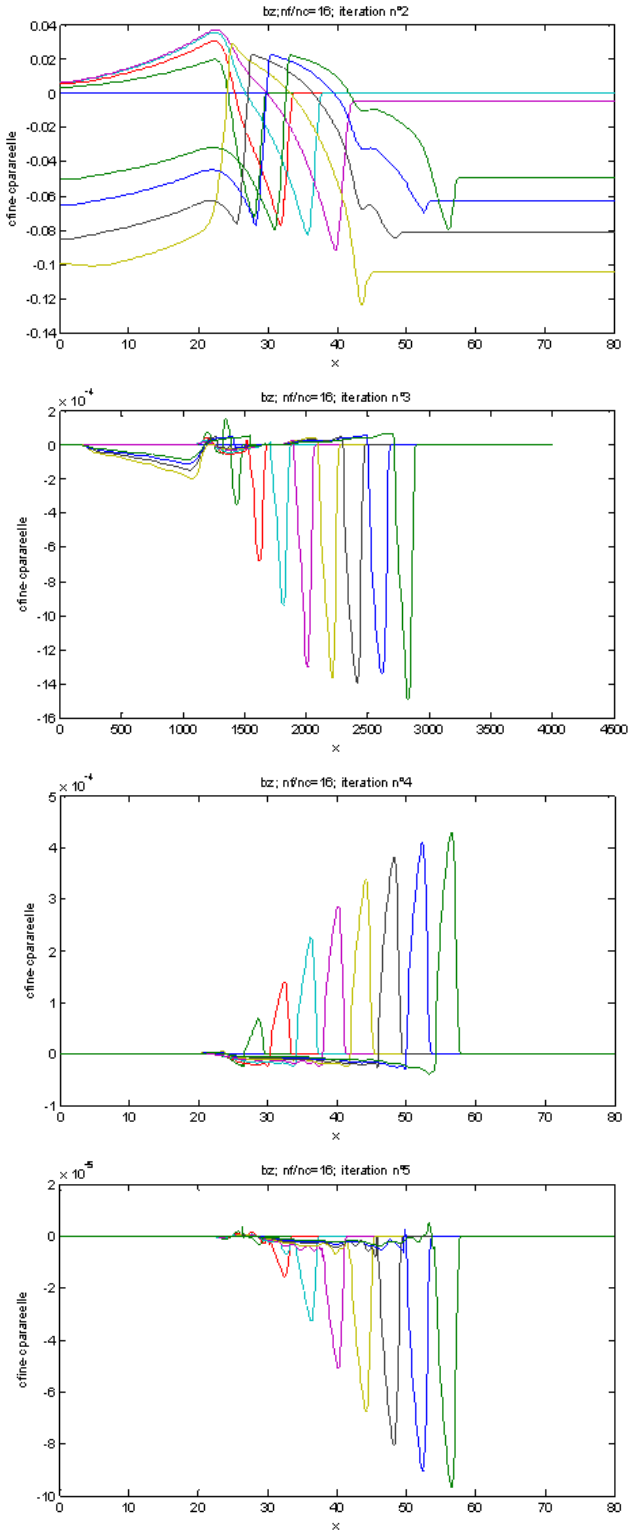
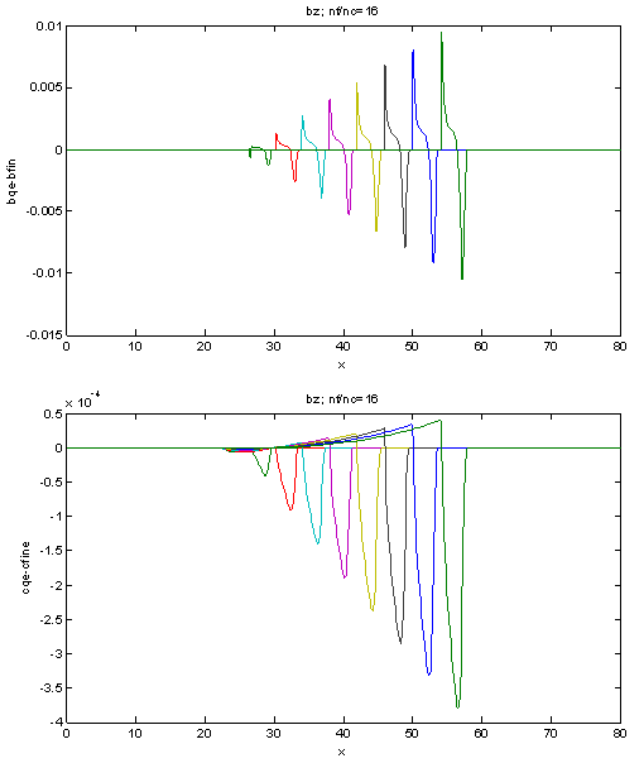


FIG. 4.51: Erreur entre la solution fine et la quasi-exacte $b_{qe}(x,t) - b_{fine}(x,t)$ (haut) et $c_{qe}(x,t) - c_{fine}(x,t)$ (bas), BZ



4.5.4 Évolution itérative de l'erreur L^2 .

On peut tirer les mêmes conclusions en regardant l'erreur L^2 cette fois-ci par rapport à la solution quasi-exacte (figures 4.52 et 4.53). En effet, les ordres des erreurs atteints correspondent tout à fait à ceux du solveur fin [12].

FIG. 4.52: Évolution itérative de l'erreur L^2 de la solution pararéelle par rapport à la quasi-exacte (Radau5-Sdirk4), BZ. Variable b en haut, c en bas.

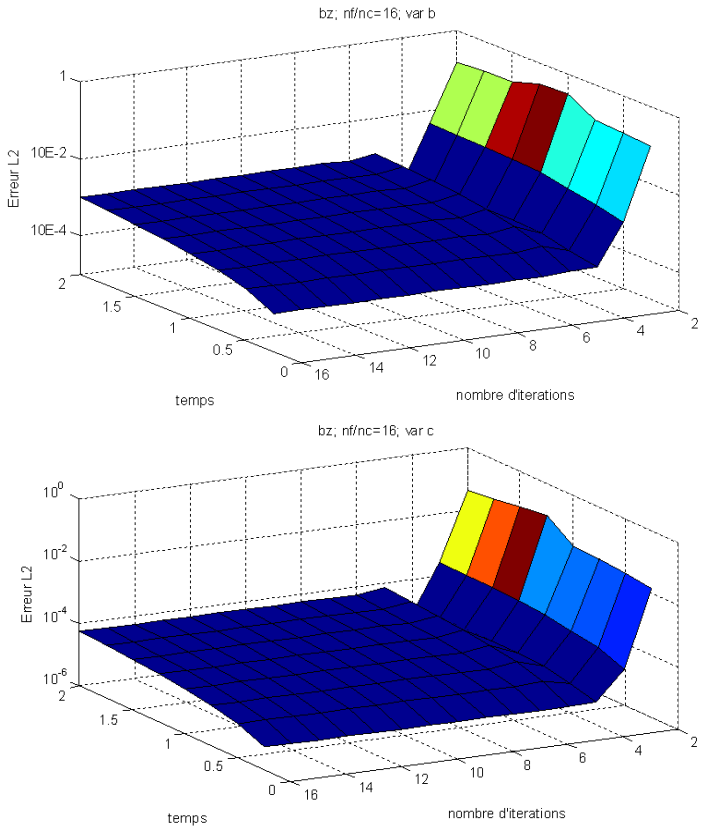
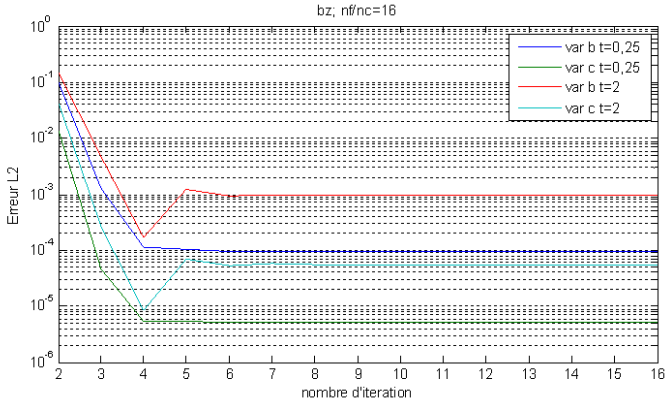


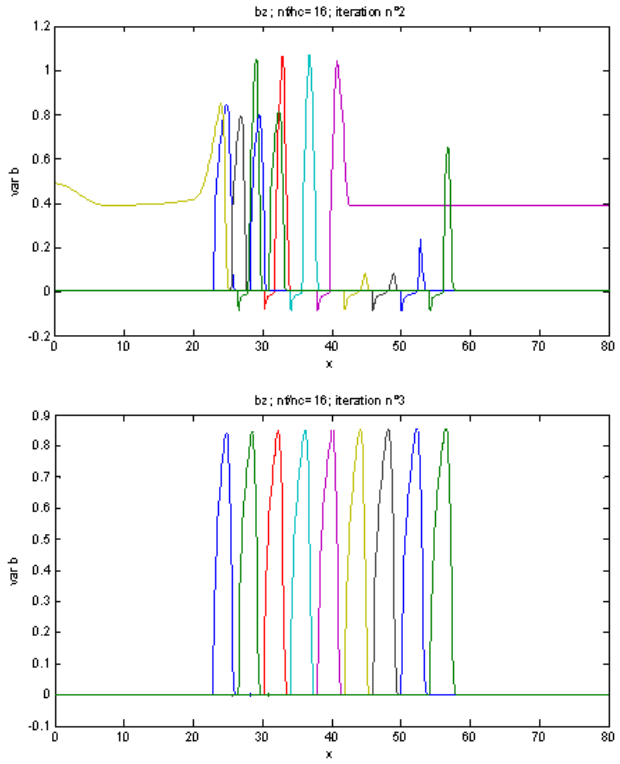
FIG. 4.53: Évolution itérative de l'erreur L^2 de la solution pararéelle par rapport à la quasi-exacte sur les limites temporelles (Radau5-Sdirk4), BZ.



4.5.5 Résolution du problème en appliquant l'algorithme pararéel. Deuxième approche.

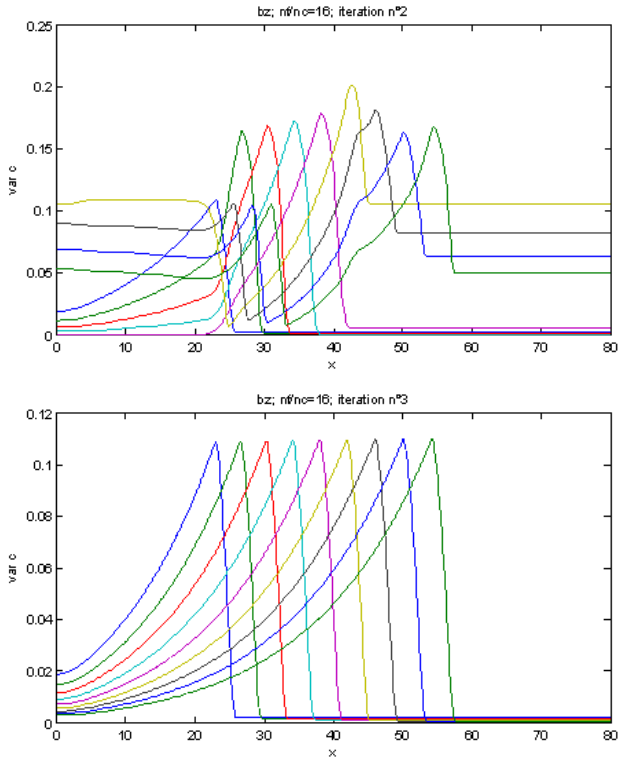
Maintenant, on reprend le même cas, mais cette fois-ci, toutes les tolérances sont fixées à 1.10^{-10} (Figures 4.54 et 4.55).

FIG. 4.54: Solution pararéelle (Radau5-Sdirk4), deuxième approche, BZ. Variable $b(x,t)$.



De la même manière, le profil de l'onde, initialement perturbé, est rapidement reconstruit à partir de la troisième itération.

FIG. 4.55: Solution pararéelle (Radau5-Sdirk4), deuxième approche, BZ. Variable $c(x,t)$.



4.5.6 Analyse de convergence.

Pour évaluer la convergence de l'algorithme, on calcule l'erreur L^2 par rapport à la solution du solveur fin. Ces erreurs ayant été calculées pour 8 instants également séparés dans le domaine temporel d'étude (figure 4.56). Cette fois-ci, le solveur ne retrouve aucun type d'inconvénient lors de l'intégration et en plus, atteint des valeurs d'erreur beaucoup plus faibles qu'auparavant.

On voit bien que la variable lente est résolue plus vite et plus précisément. En regardant plus précisément l'erreur pour les limites du domaine ($t = 0,25$ et $t = 2$), on voit que le taux de convergence devient plus faible au fur et à mesure le temps avance (figure 4.57).

Le critère d'arrêt est fixé sur la différence maximale entre deux solutions d'itérations successives, qui sont représentées dans la figure 4.58. On va voir que l'algorithme converge autour la sixième itération.

FIG. 4.56: Évolution itérative de l'erreur L^2 de la solution pararéelle par rapport à la fine (Radau5-Sdirk4), deuxième approche, BZ. Variable b en haut, c en bas.

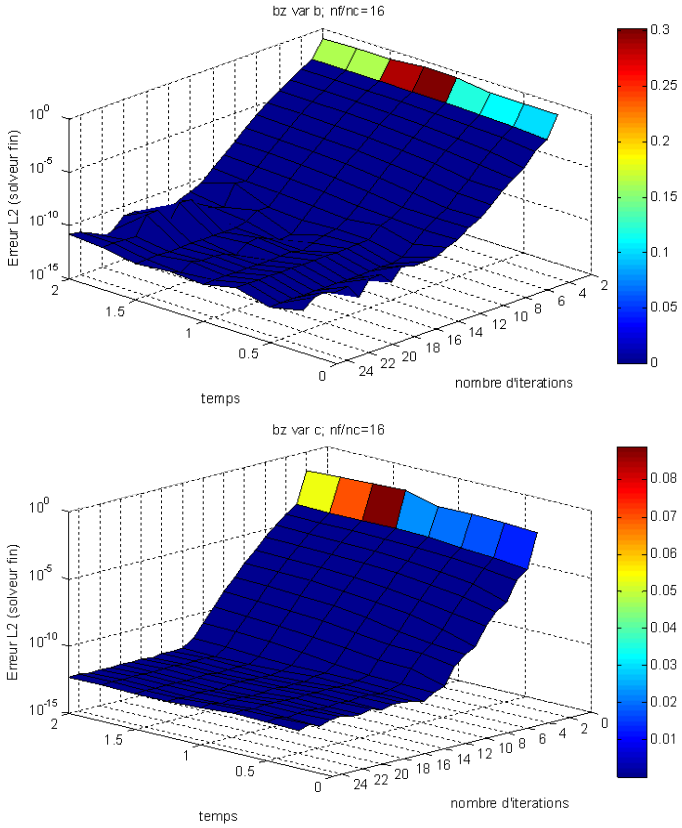


FIG. 4.57: Évolution itérative de l'erreur L^2 de la solution pararéelle par rapport à la fine sur les limites temporelles (Radau5-Sdirk4), deuxième approche, BZ.

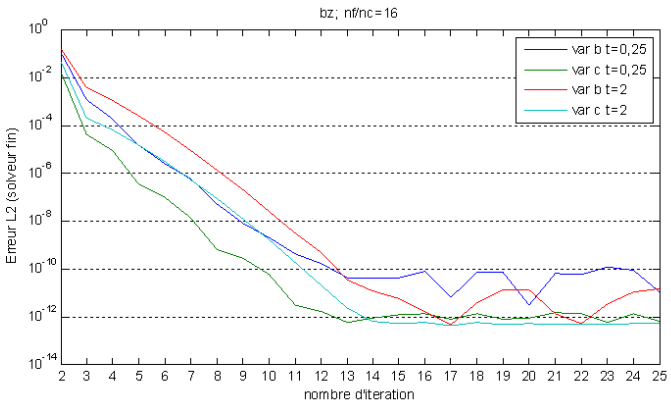
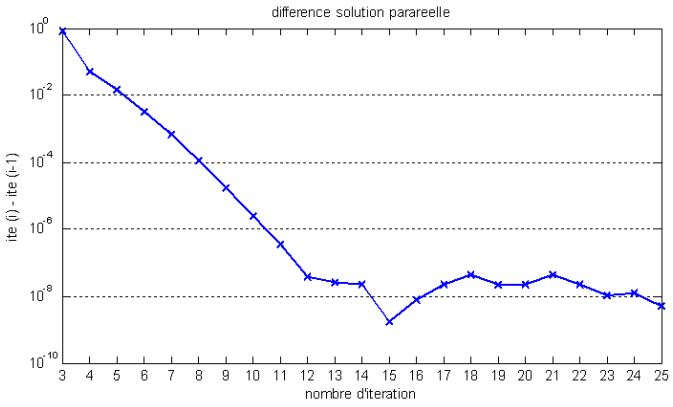


FIG. 4.58: Évolution itérative du critère de convergence (Radau5-Sdirk4), deuxième approche, BZ.



4.5.7 Évolution itérative de l'algorithme vers la solution fine.

Maintenant, on regarde la différence entre les profils des ondes fine et pararéelle (Figures 4.59 et 4.60).

Si l'on les compare avec la différence entre les profils quasi-exacte et fin (figure 4.61), on voit que celle-ci devient du même ordre pour la cinquième itération. On s'aperçoit que l'évolution est la même qu'on avait avant.

FIG. 4.59: Évolution itérative de l'algorithme vers la solution fine $b_{fine}(x,t) - b_{para}(x,t)$ (Radau5-Sdirk4), deuxième approche, BZ

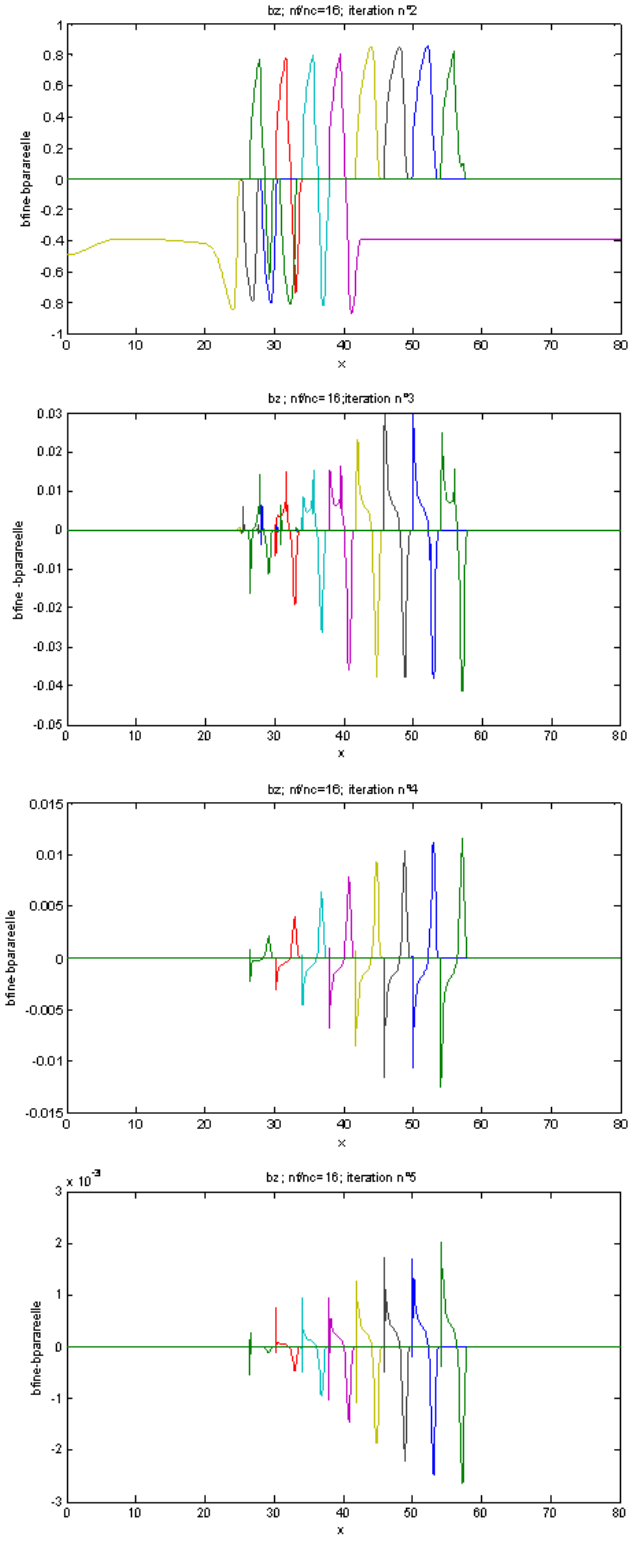


FIG. 4.60: Évolution itérative de l'algorithme vers la solution fine $c_{fine}(x,t) - c_{para}(x,t)$ (Radau5-Sdirk4), deuxième approche, BZ

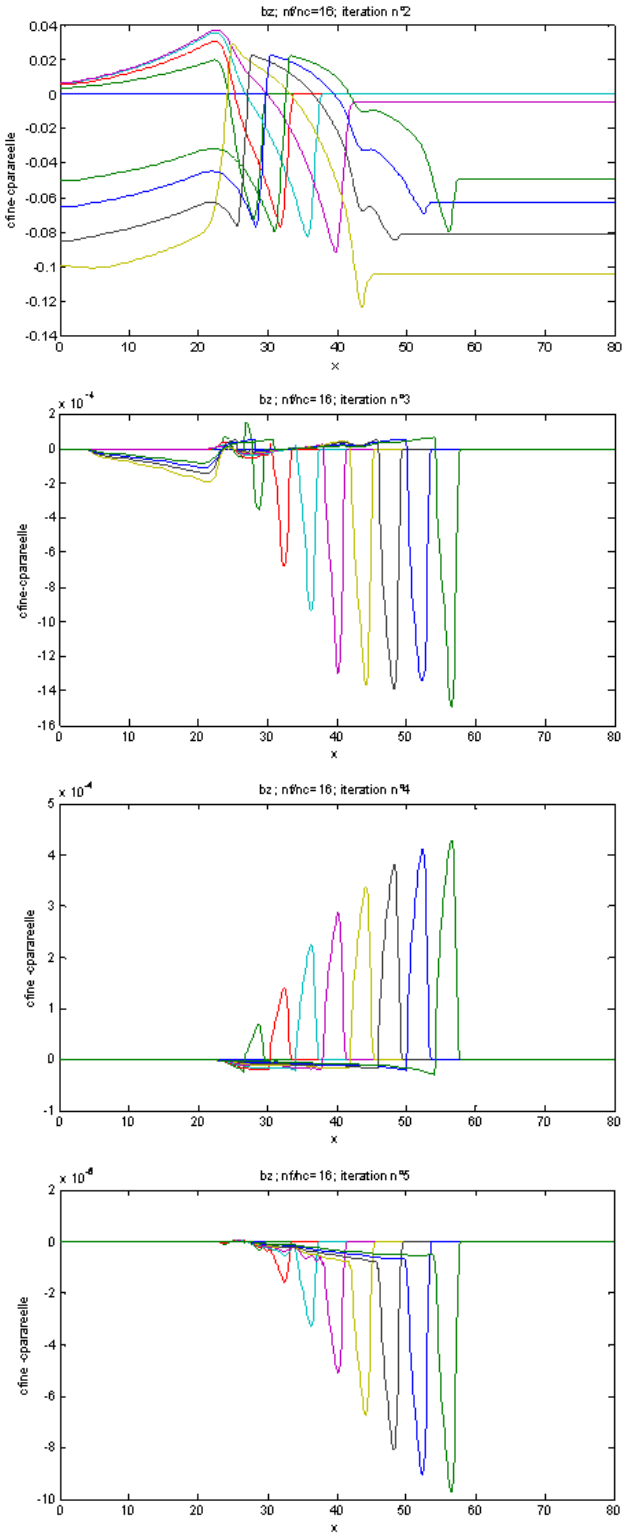
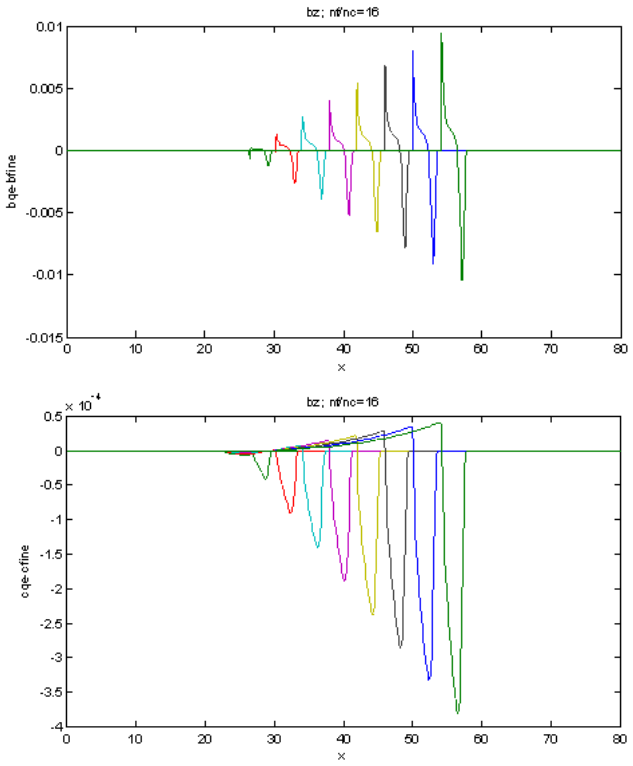


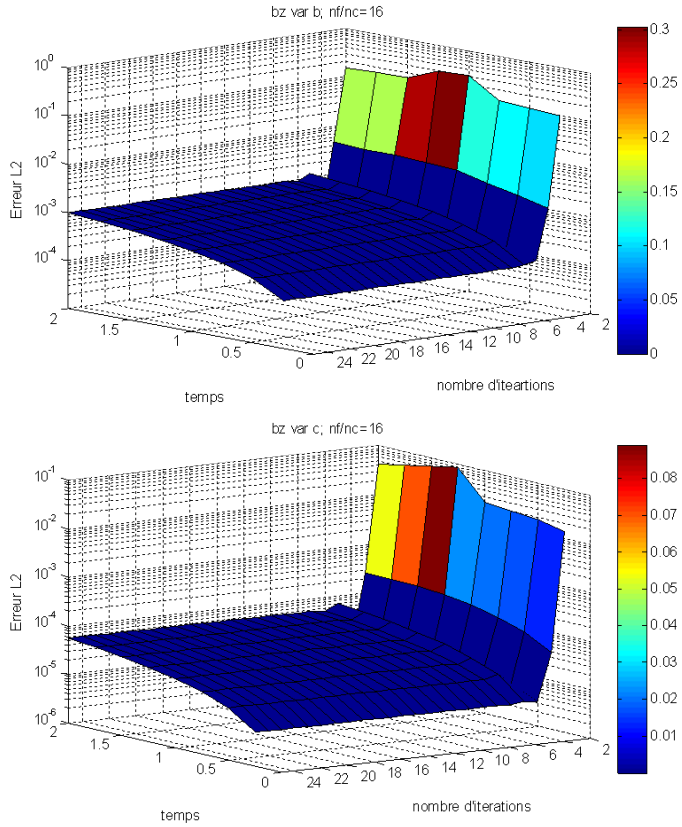
FIG. 4.61: Erreur entre la solution fine et la quasi-exacte $b_{qe}(x,t) - b_{fine}(x,t)$ (haut) et $c_{qe}(x,t) - c_{fine}(x,t)$ (bas), deuxième approche, BZ



4.5.8 Évolution itérative de l'erreur L^2.

On retrouve la même évolution en regardant l'erreur L^2 cette fois-ci par rapport à la solution quasi-exacte (figures 4.62 et 4.63).

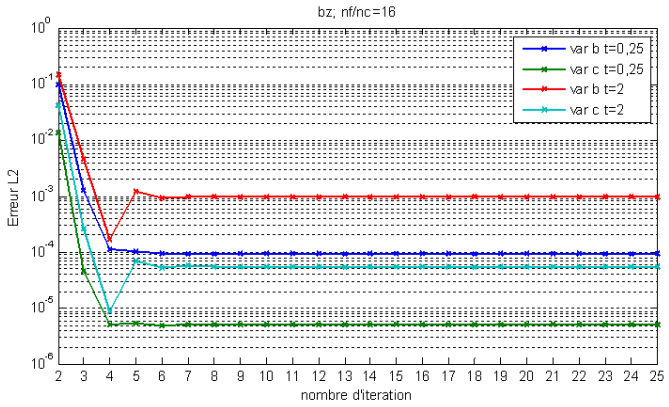
FIG. 4.62: Évolution itérative de l'erreur L^2 de la solution pararéelle par rapport à la quasi-exacte (Radau5-Sdirk4), deuxième approche, BZ. Variable b en haut, c en bas.



Dans ce dernière cas, où les tolérances prises en compte montrent une meilleure résolution, on peut regarder plus précisément l'effet de la présence des phénomènes multi-échelles sur le comportement de l'algorithme. En fait, il est assez semblable au cas KPP raide, pour ce qui est du taux de convergence.

En tout cas, la performance de l'algorithme ne change pas au niveau des résultats, comme le montre les figures ; sauf que le solveur Radau5 n'a pas d'ennui et, du coup, d'un côté on gagne du temps de calcul et de l'autre, on est sûr de son bon fonctionnement.

FIG. 4.63: Évolution itérative de l'erreur L^2 de la solution pararéelle par rapport à la quasi-exacte sur les limites temporelles (Radau5-Sdirk4), deuxième approche, BZ.



4.6 Résolution du système complet BZ.

Maintenant, on considère le système complet BZ (1.39), c'est-à-dire qu'on rajoute la troisième équation au système simplifié (1.40). De cette manière on essaie de résoudre l'évolution de la variable a aussi, qui aura des taux de variation plus prononcés à cause de la présence du paramètre μ.

Entre les configurations de calcul évaluées avant, on reprend celle pour laquelle on avait obtenu la meilleure performance de l'algorithme, de manière à évaluer l'influence de cette nouvelle variable rapide sur son comportement. C'est-à-dire qu'on considère un pas de splitting de $2/512$ et que les méthodes de résolution fine et grossière seront RADAU5 pour le pas de réaction et SDIRK4 pour la diffusion.

Les évolutions des solutions vers la solution fine montrent que celles-ci restent presque inchangées pour les variables b et c (figure 4.64) ; alors que la comparaison entre les évolutions des trois variables au début et à la fin du domaine d'intégration permet retrouver le même comportement que précédement : le taux de convergence diminue lorsqu'on s'éloigne de la condition initial (figure 4.65).

En analysant l'erreur par rapport à la solution quasi-exacte (figure 4.66) on s'aperçoit que la convergence de l'algorithme a été ralentie, puisqu'il converge autour de la huitième itération. Evidement, l'approximation grossière de la variable rapide a joue un rôle décisif : plus elle est mauvaise, plus le nombre d'itérations est élevé pour converger vers la solution du solveur fin, même si l'algorithme a déjà réussi à résoudre les autres variables plus lentes.

C'est-à-dire que cette fois-ci, c'est la variable a qui va déterminer le nombre d'itérations au lieu de la variable b. En fait, on aura toujours ce type de comportement : la performance de l'algorithme est étroitement liée à l'approximation grossière surtout des variables les plus rapides. D'autre part, c'est cette résolution qui limite le choix des méthodes numériques pour le solveur grossier et réduit la plage d'applicabilité de l'algorithme pararéel.

FIG. 4.64: Évolution itérative de l'erreur L^2 de la solution pararéelle par rapport à la fine, BZ complet. Variable a en haut,b au milieu, c en bas.

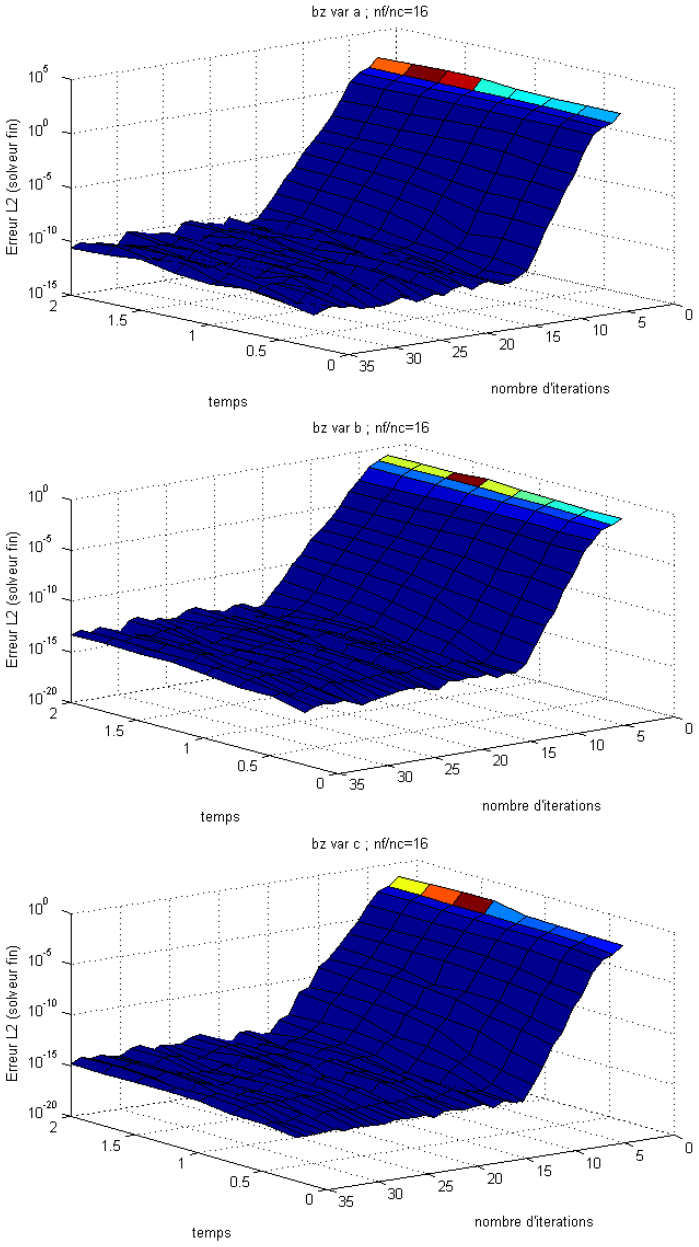


FIG. 4.65: Évolution itérative de l'erreur L^2 de la solution pararéelle par rapport à la fine (solveur fin) sur les limites temporelles, BZ complet.

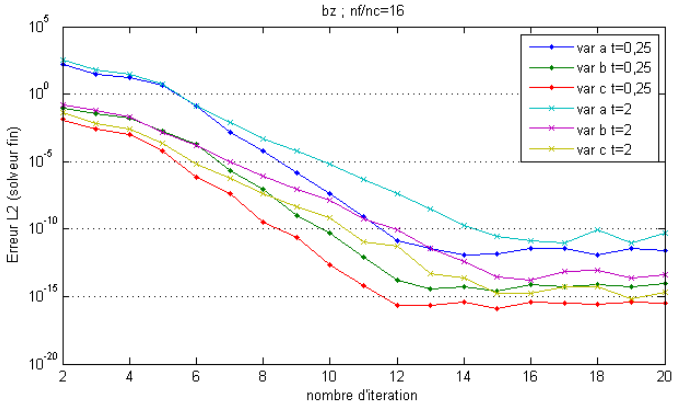
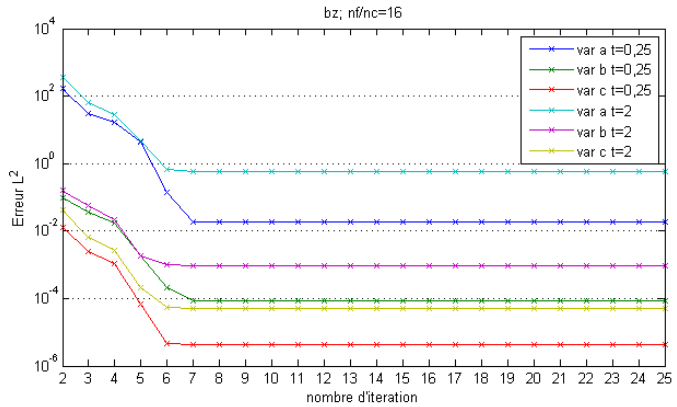


FIG. 4.66: Évolution itérative de l'erreur L^2 de la solution pararéelle par rapport à la quasi-exacte sur les limites temporelles, BZ complet .



4.7 Environnement parallèle et temps de calcul.

Avec les analyses faites, on peut mieux choisir les différents paramètres nécessaires pour effectuer la parallélisation effective. Dans un premier temps, présentons la démarche générale mise en oeuvre. La parallélisation a été faite en utilisant la librairie MPI (*Message Passing Interface*) qui permet l'échange d'information parmi plusieurs processeurs.

Pour cette application, la structure de la démarche a été la suivante :

1. Initialisation : tous les processeurs mis en jeu acquièrent les données d'entrée.
2. Première solution pararéel : tous les processeurs effectuent la même intégration temporelle sur tout le domaine (solveur *grossier*).
3. A partir de la solution pararéelle obtenue, chacun des processeurs effectue l'intégration temporelle sur un nombre réduit d'intervalles différents (solveur *fin*).
4. Les résultats obtenus par chaque processeur est trasmis aux autres en utilisant la librairie MPI.
5. Nouvelle solution pararéelle : tous les processeurs effectuent une intégration temporelle sur tout le domaine avec les corrections introduites (solveur *grossier*).
6. Vérification du critère d'arrêt.
7. Si le critère d'arrêt n'a pas été atteint, le processus continue à partir de 3. Sinon, c'est la fin du programme.

Selon la description faite, il faut remarquer que pendant toute la démarche, tous les processeurs font des calculs. La parallélisation temporelle est effective pendant le pas 3, lorsque chaque processeur intègre un intervalle de temps différent au même temps. Tous les calculs sur architecture parallèle ont été effectués par le cluster du laboratoire EM2C, "madness". De cette manière, on a fait des calculs pour des cas tests avec quelques paires de processeurs pour arriver aux cas les plus intéressants avec 72 ou jusqu'à 126 processeurs.

Comme cas test, on reprend le modèle KPP et on applique l'algorithme vu dans la section (4.2.2), cette fois-ci, avec plusieurs processeurs. On considérera des pas de splitting petits comme $h_{gros} = 30/72$ pour le solveur grossier et des rapports de 50 et 100 pour le solveur fin ($h_{fin} = h_{gros}/50$ ou $h_{fin} = h_{gros}/100$). Le tableau (4.5) résume les détails pour les deux configurations. T_{fin} et T_{para} représentent les temps utilisés par le solveur fin et le pararéel, respectivement.

TAB. 4.5: Résultats pararéels, KPP.

Rapport	**50**	**100**
$\boldsymbol{tolg}$	1.10^{-10}	
$\boldsymbol{tolf}$	1.10^{-15}	
$\boldsymbol{N_{proc}}$	72	
$\boldsymbol{T_{fin}}$	129,44	215,23
$\boldsymbol{T_{para}}$	34,74	48,13
$\frac{\boldsymbol{T_{fin}}}{\boldsymbol{TT_{para}}}$	$\mathbf{3,73}$	$\mathbf{4,47}$

Ces valeurs donnent une idée de l'ordre de grandeur des gains de temps pour des systèmes de réaction-diffusion. Ceux-ci sont loin d'être directement le nombre de processeurs utilisés et on verra par la suite comment ces rapports sont constitués.

4.7.1 Estimation du temps de calcul.

Dans cette section on évalue la performance de l'algorithme en considérant différentes méthodes de résolution. Dans ce cas, l'algorithme pararéel a été codé en $C++$, de maniére à prendre en main le code ZEBRE, logiciel développé par T. Dumont, qui offre une grande variété de solveurs puissants pour la résolution de systèmes d'équations de réaction-diffusion raides.

Le système pris en compte est le système complet BZ (1.39) avec les paramètres suivants : $\epsilon = 10^{-2}$, $\mu = 10^{-5}$, $f = 1,6$ et $q = 2.10^{-4}$.

Cette fois-ci on l'applique sur un domaine spatial bidimensionnel avec deux maillages spatiaux différents. Des études théoriques et numériques prédisent la formation des ondes spirales spatiales pour l'évolution des concentrations. On a repris les conditions initiales utilisés par [2], c'est-à-dire les distributions discontinues montrées dans la figure 4.67. Les ondes spirales développées sont représentées dans la figure 4.68.

L'étude faite par [2] permet de démontrer une résolution plus performante avec le schéma de Strang RDR (réaction - diffusion - réaction) par rapport à celle obtenue avec des méthodes multi pas implicites - explicites. De cette manière on utilisera comme solveur fin la méthode qu'ils ont utilisée, c'est-à-dire Strang avec un pas de splitting de $1,25.10^{-4}$ sur le domaine temporel $[0, 2]$. La méthode d'intégration est RADAU5 pour le pas de réaction et SDIRK4 pour la diffusion. Comme l'algorithme pararéel converge vers son solveur fin, on est sûr d'avoir atteint la même précision.

Pour la méthode grossière on utilise plusieurs méthodes. Comme SDIRK4 est une méthode tout à fait opportune pour la résolution du laplacien [14], on la garde pour le pas de diffusion ; de plus, la raideur du système est associée plus directement au système dynamique réactif, on pourrait attendre un temps de calcul plus important pour cette résolution. Du coup, on fera différents calculs avec RADAU5, RODAS et SEULEX et, dans tous les cas, on utilisera des tolérances totalement relaxées ; cela implique une correction minimale du pas d'intégration et donc, une vitesse de résolution supérieure. Pour des raisons de clarté, on présentera les résultats obtenus en suivant le schéma de calcul suivant,

- on divise le domaine temporel $[0, 2]$ en 100 sous-intervalles égaux h_{para}, c'est-à-dire $h_{para} = 2.10^{-2}$;
- chaque sous-intervalle h_{para} est résolu par le solveur grossier, lequel a un pas de splitting $h_{gros} = h_{para}/10 = 2.10^{-3}$;
- de même, le solveur fin est appliqué avec un pas de splitting $h_{fin} = h_{para}/160 = 1,25.10^{-4}$;
- le rapport des pas de splitting est $h_{gros}/h_{fin} = 16$.

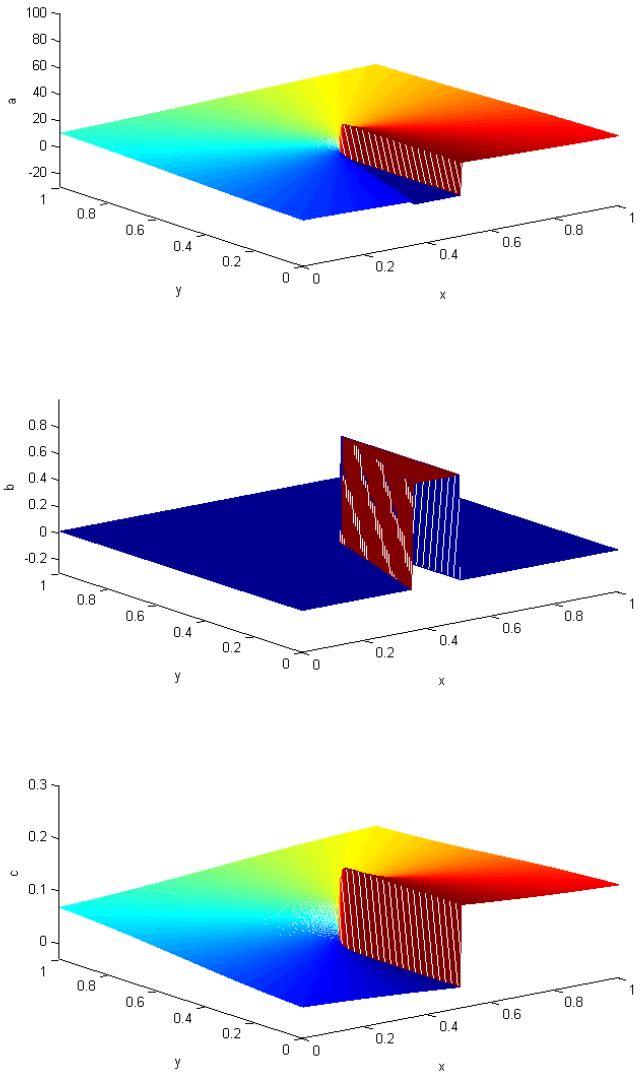
Pour des maillages spatiaux de 100×100 et 200×200, les résultats sont montrés dans les tableaux 4.6 et 4.7, respectivement.

TT_{gros} représente le temps de calcul total utilisé par le solveur grossier, alors que TT_{fin}, celui correspondant au fin, en supposant une parallélisation effective avec 100 processeurs, un pour chaque sous-intervalle. TT_{para} est la somme des deux. TT constitue le temps nécessaire pour résoudre le système avec la méthode fine sans l'application de l'algorithme pararéel. Ainsi, on obtient le rapport de temps de calcul qu'on gagne avec l'utilisation de l'algorithme.

On voit que le comportement est à peu près le même avec les différents solveurs. En tout cas, il faut se rappeler que ceux-ci sont appropriées aux problèmes raides. L'importance de l'approximation grossière est mise en évidence d'après les rapports obtenus.

A priori une approximation plus grossière implique un gain de temps de calcul, mais plus d'itérations sont alors nécessaires pour converger, alors un équilibre entre les deux conditions semble nécessaire. Évidement, la contrainte principale vient de la raideur du système, c'est-à-

FIG. 4.67: Conditions initiales, BZ bidimensionnel. Variable a en haut, b au milieu, c en bas.



dire que des approximations très grossières risquent de ne pas converger et, de plus, l'utilisation des méthodes appropriées mais coûteuses devient impérative.

En fait, une analyse qualitative de l'algorithme nous amène à l'estimation suivante ; pour

T_{fin} **:** temps de calcul du solveur fin sur tout le domaine temporel ;

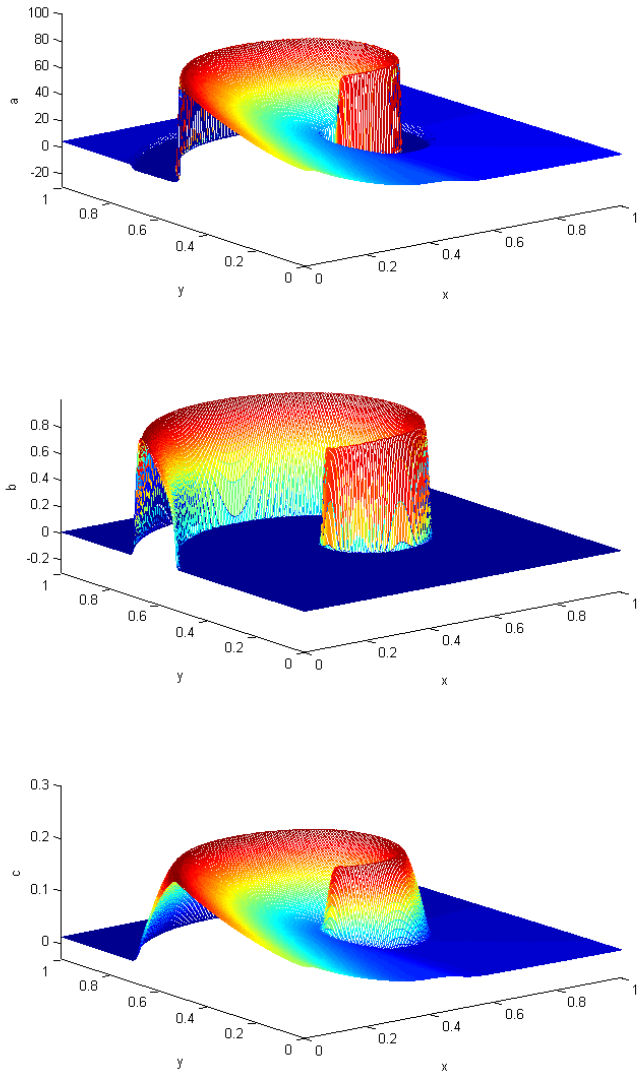
T_{gros} **:** temps de calcul du solveur grossier sur tout le domaine temporel ;

N_{proc} **:** nombre de processeurs ; et

N_{ite} **:** nombre d'itérations ;

on voudrait que le temps de calcul utilisé par l'algorithme, T_{para}, soit équivalent au temps de résolution nécessaire avec le solveur fin, distribué sur tous les processeurs, c'est-à-dire

FIG. 4.68: Ondes spirales, BZ bidimensionnel. Variable a en haut,b au milieu, c en bas.



$$T_{para} \approx \frac{T_{fin}}{N_{proc}}. \tag{4.3}$$

Sauf que généralement, plusieurs itérations sont nécessaires, autrement dit,

$$T_{para} \approx N_{ite} \times \frac{T_{fin}}{N_{proc}}. \tag{4.4}$$

On s'aperçoit que si l'on peut obtenir ce temps de calcul pour le problème considéré, la performance de l'algorithme devient presque optimale.

Néanmoins, à chaque itération, on a besoin de calculs faits avec le solveur grossier et de plus, ces calculs sont faits de manière séquentielle, c'est-à-dire, qu'on a,

TAB. 4.6: Temps de calcul, BZ bidimensionnel (maillage 100×100).

Solveur grossier	**RADAU5**	**RODAS**	**SEULEX**
$\boldsymbol{TT}$	$3099,88$		
nombre d'itérations	5	5	5
$\boldsymbol{TT_{gros}}$	1007,38	731,61	814,28
$\boldsymbol{TT_{fin}}$	154,12	153,28	152,21
$\boldsymbol{TT_{para}}$	1161,50	884,89	966,49
$\frac{\boldsymbol{TT}}{\boldsymbol{TT_{para}}}$	$\mathbf{2,67}$	$\mathbf{3,50}$	$\mathbf{3,21}$

TAB. 4.7: Temps de calcul, BZ bidimensionnel (maillage 200×200).

Solveur grossier	**RADAU5**	**RODAS**	**SEULEX**
$\boldsymbol{TT}$	12485		
nombre d'itérations	6	6	6
$\boldsymbol{TT_{gros}}$	4851,84	3551,73	3947,14
$\boldsymbol{TT_{fin}}$	750,32	751,20	752,03
$\boldsymbol{TT_{para}}$	5602,16	4302,93	4699,17
$\frac{\boldsymbol{TT}}{\boldsymbol{TT_{para}}}$	$\mathbf{2,23}$	$\mathbf{2,90}$	$\mathbf{2,66}$

$$T_{para} \approx N_{ite} \times T_{gros} + N_{ite} \times \frac{T_{fin}}{N_{proc}}. \tag{4.5}$$

Finalement, comme on a besoin d'une approximation initiale faite para le solveur grossier, on arrive à l'estimation suivante,

$$T_{para} \approx (N_{ite} + 1) \times T_{gros} + N_{ite} \times \frac{T_{fin}}{N_{proc}}. \tag{4.6}$$

Cette expression (4.6) montre clairement l'influence du solveur grossier sur la performance de l'algorithme. Le choix de la méthode utilisée devient critique et constitue la principale contrainte à prendre en compte.

En fait, si l'on estime les rapports obtenus par les expressions (4.4) et (4.6), on voit qu'on s'éloigne de la performance optimale, même si l'approximation donnée par le solveur grossier est assez faible et le solveur rapide (tablaux 4.8 et 4.9).

TAB. 4.8: Rapports du temps de calcul, BZ bidimensionnel (maillage 100×100).

Solveur grossier	**RADAU5**	**RODAS**	**SEULEX**
$\frac{\boldsymbol{T_{fin}}}{\boldsymbol{T_{gros}}}$	18,46	25,42	22,84
$(\frac{\boldsymbol{T_{fin}}}{\boldsymbol{T_{para}}})_{\boldsymbol{opt}}$	20,11	20,22	20,36
$\frac{\boldsymbol{T_{fin}}}{\boldsymbol{T_{para}}}$	$\mathbf{2,67}$	$\mathbf{3,50}$	$\mathbf{3,21}$

TAB. 4.9: Rapports du temps de calcul, BZ bidimensionnel (maillage 200×200).

Solveur grossier	**RADAU5**	**RODAS**	**SEULEX**
$\frac{T_{fin}}{T_{gros}}$	18,01	24,61	22,14
$(\frac{T_{fin}}{T_{para}})_{opt}$	16,64	16,62	16,60
$\frac{T_{fin}}{T_{para}}$	$\mathbf{2,23}$	$\mathbf{2,90}$	$\mathbf{2,66}$

Cependant, d'après ces résultats on voit bien que les rapports obtenus sont déjà intéressants si on prend en compte que la précision atteinte est hautement satisfaisante et que l'algorithme arrive à une résolution assez fine même à partir des approximations initiales grossières. Ceci devient plus claire lorsqu'on regarde les écarts au niveau des erreurs entre la première et la dernière itérations pour tous les cas qu'on a traité.

Il faut aussi noter que pendant toute cette étude on a toujours privilégié des résolutions les plus rigoureux, ce qui rendre les calculs beaucoup plus lourds et pousse l'algorithme aux niveaux les plus contraignants pour évaluer sa performance. C'est-à-dire, on envisage clairement des rapports plus importants pour des résolutions un peu moins précises mais qui permettent encore de décrire correctement les phénomènes.

Conclusions.

L'objectif principal de cette étude était évaluer l'applicabilité de l'algorithme pararéel sur des problèmes d'évolution temporelle non linéaires raides, puisque ceux-ci ont besoin de méthodes de résolution plus puissantes et surtout couteuses.

La première application sur un système dynamique non linéaire raide, l'Oregonator, nous a montré clairement que le choix de la méthode d'approximation grossière, solveur grossier, a un rôle prépondérant dans la résolution de ce type de problèmes. En fait, il faut que celui-là soit basé sur des méthodes tout à fait appropriées aux problèmes raides. Cette contrainte reste évidente au long des différentes applications.

L'étude des modèles KPP raide et non raide a mis en évidence l'influence de la présence de forts gradients spatiaux sur la performance de l'algorithme. En effet, le taux de convergence est clairement ralenti comme une conséquence directe. Néanmoins, les analyses faites montrent une résolution très performante au niveau du profil de l'onde ainsi comme de sa vitesse, même avec des approximations initiales assez grossières.

On a tiré les mêmes conclusions à partir du comportement retrouvé pour le modèle BZ. Cette fois-ci la raideur est associée principalement à la nature de la chimie non linéaire. D'après les résultats numériques, l'utilisation de solveurs basés sur des méthodes de résolution mono-pas est tout à fait conseillée.

Finalement, l'implémentation sur architecture parallèle a confirmé les remarques faites. Mêmes si les gains en temps de calcul ne sont pas énormes, on pourrait quand même profiter des avantages de la méthode. Il faut prendre en compte qu'on arrive à résoudre des systèmes très complexes à partir des approximations très mauvaises et en tout cas, des gains plus importants peuvent être obtenus selon le niveau de précision souhaité. En fait, pour cette étude on a privilégié toujours une résolution plus précise que rapide justement pour mettre en évidence la capacité de l'algorithme à réussir sous des conditions les plus contraignantes possibles.

En conclusion, cette étude a introduit des outils nouveaux pour la simulation des problèmes multi-échelles et les conclusions et remarques faites permettent de donner des critères d'application à prendre en compte pour des cas semblables. D'autre part, elle contribue à l'étude théorique de ce type de problèmes et on envisage des études complémentaires qui permettent de mieux les comprendre.

Bibliographie

[1] A. Auclert and M. Guillot. Méthodes de séparation d'opérateurs pour la simulation numérique de milieux réactifs en évolution impliquant un large spectre d'échelles de temps. Projet de deuxième année, Ecole Centrale Paris, 2006.

[2] S. Descombes, T. Dumont, and M. Massot. Operator splitting for stiff nonlinear reaction-diffusion systems : order reduction and application to spiral waves. In *Patterns and waves (Saint Petersburg, 2002)*, pages 386–482. AkademPrint, St. Petersburg, 2003.

[3] S. Descombes and M. Massot. Operator splitting for nonlinear reaction-diffusion systems with an entropic structure : singular perturbation and order reduction. *Numer. Math.*, 97(4) :667–698, 2004.

[4] M. Gander and E. Hairer. Nonlinear convergence analysis for the parareal algorithm. In *Domain Decomposition Methods in Science and Engineering XVII*, pages 45–56. Springer, Berlin, 2008.

[5] M. Gander and S. Vandewalle. Analysis of the parareal time-parallel time-integration method. *Society for Industrial and Applied Mathematics*, 29(2) :556–578, 2007.

[6] P. Gray and S. K. Scott. *Chemical oscillations and instabilites*. Oxford University Press, 1994.

[7] E. Hairer. Cours d'analyse numérique. chapitre III : équations différentielles ordinaires. Polycopie du cours, Université de Genève, 2004.

[8] E. Hairer and G. Wanner. *Solving ordinary differential equations II*. Springer-Verlag, Berlin, 1991. Stiff and differential-algebraic problems.

[9] A.N. Kolmogoroff, I.G. Petrovsky, and N.S. Piscounoff. Etude de l'équation de la diffusion avec croissance de la quantité de matière et son application a un problème biologique. *Bulletin de l'Université d'état Moscou, Série Internationale Section A Mathématiques et Mécanique*, 1 :1–25, 1937.

[10] J. L. Lions, Y. Maday, and G. Turinici. Résolution d'EDP par un schéma en temps "pararéel". *C. R. Acad. Sci. Paris Sér. I Math.*, 332(7) :661–668, 2001.

[11] V. Louvet. Radau5 : résolution d'ode raide. Rapport technique, Institut Camille Jordan. Université Claude Bernard, Lyon I, 2007.

[12] M. Massot. Quelques points intéressants sur les ondes et le splitting. Rapport technique, Laboratoire de Mathématiques Appliquées de Lyon. Université Claude Bernard, Lyon I, 2001.

[13] M. Massot. Singular perturbation analysis for the reduction of complex chemistry in gaseous mixtures using the entropic structure. *Discrete and Continuous Dynamical Systems - Series B*, 2(3) :433–456, 2002.

[14] D. Ropp and J. Shadid. Stability of operator splitting methods for systems with indefinite operators : reaction-diffusion systems. *Journal of Computational Physics*, 203 :449–466, 2005.

Annexe A

Évolution dans le plan de phase.

A.1 Modèle KPP.

Maintenant, voyons l'évolution du profil de l'onde uniquement, pour cela, on analysera les valeurs dans le plan de phase (β, $\frac{d\beta}{dz}$). Nous cherchons à tracer la différence $\beta'_{parareel} - \beta'_{fin}$ en fonction de β (la dérivation étant une dérivation spatiale pour un instant fixé).

Pour la première itération, on s'aperçoit que l'algorithme crée des solutions qui dépassent la limite du domaine. Ce fait est corrigé par les suivantes itérations (figure A.1).

A partir de la sixième itération, l'erreur de l'algorithme devient négligeable devant celle du solveur fin, comme l'on voit dans la figure A.2.

FIG. A.1: Évolution itérative de la différence $\beta'_{parareel} - \beta'_{fin}$, KPP

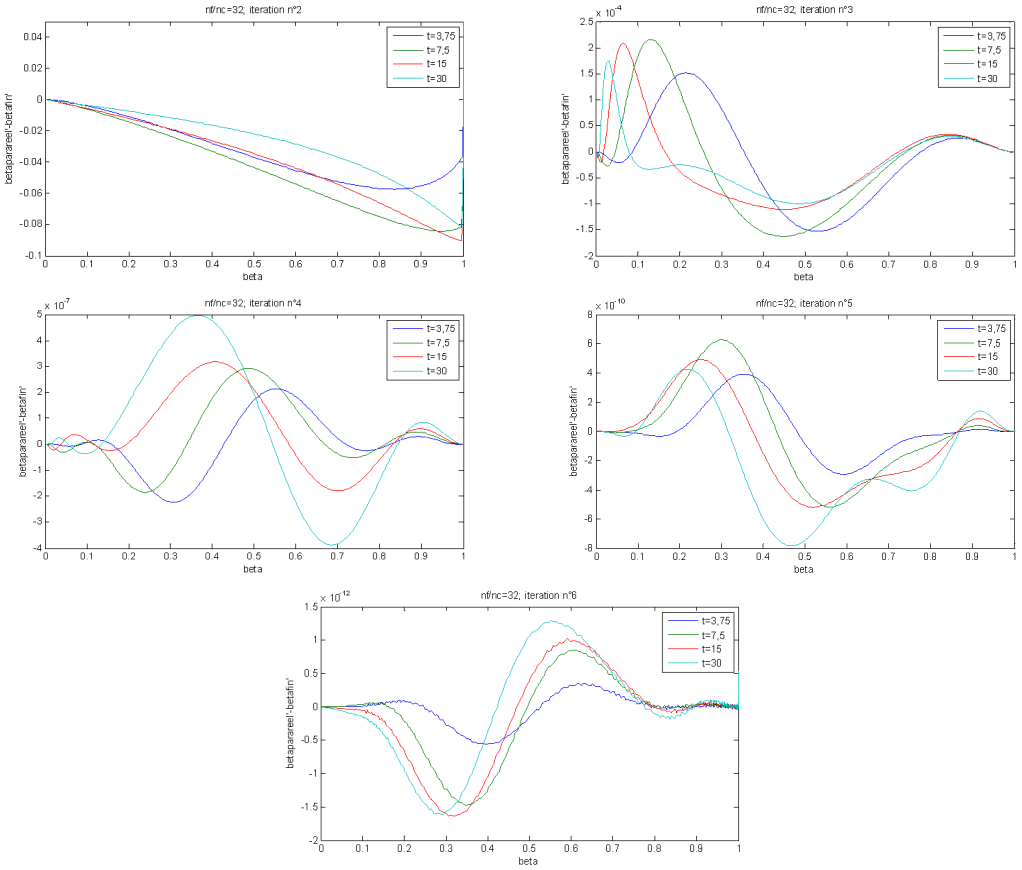
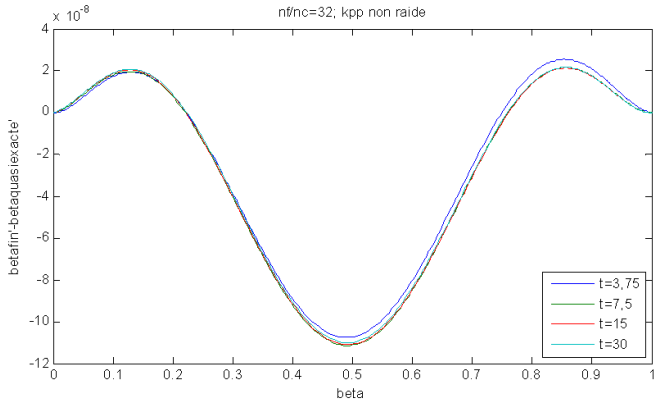


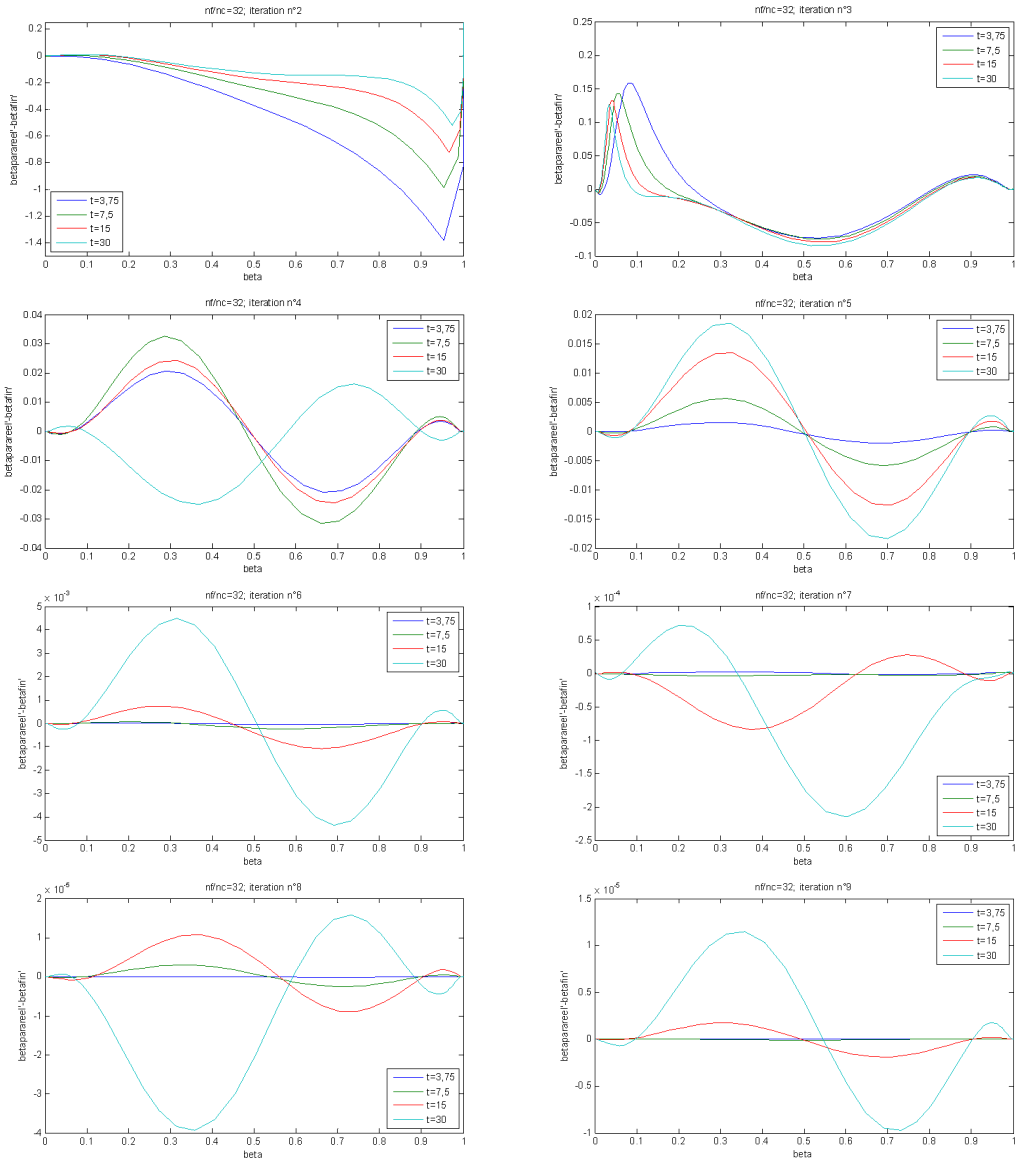
FIG. A.2: Différence $\beta'_{fin} - \beta'_{quasiexacte}$, KPP



A.2 KPP raide.

Pour le cas raid, on voit comment la convergence est ralentie dans l'évolution itérative (figure A.3).

FIG. A.3: Évolution itérative de la différence $\beta'_{parareel} - \beta'_{fin}$, KPP raide



On s'aperçoit que l'algorithme a mal à converger même après neuf itérations.

FIG. A.4: Différence $\beta'_{fin} - \beta'_{quasiexacte}$, KPP raide