INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES ROUEN

Projet de Physique P6
STPI/P6/2018-12

Cinétique Chimique

Étudiants :

Baptiste BILLARD
Marie LEMARIE
Yan CASAMAJOR
Hoang Trung NGUYEN

Enseignant-responsable du projet :
Monsieur GLEYSE

Date de remise du rapport : 18/06/2018

Référence du projet : STPI/P6/2018 – 12

Intitulé du projet : Cinétique Chimique

Type de projet : Biblio, modélisation

Objectifs du projet : Nous avons réalisé l'étude du modèle du Brusselator, un modèle théorique de réactions chimiques non linéaires et oscillantes. Le Brusselator est une réaction autocatalytique pouvant être modélisée par un système d'équations différentielles. Dans ce projet, nous cherchons à étudier la cinétique de ces réactions, leurs stabilités et à modéliser graphiquement l'évolution de ce système grâce à différentes méthodes d'approximation et d'analyse numérique. Nous étudierons le Brusselator selon différentes conditions initiales.

Mots-clefs du projet : Modélisation, approximation, algorithme.

Table des matières

Introduction **4**

1 Méthodologie, organisation du travail **5**

2 Travail réalisé et résultats **7**
2.1 Modélisation mathématique et étude de la stabilité 7
2.1.1 Système d'équations différentielles 7
2.1.2 Stabilité 7
2.2 Analyse Numérique 9
2.3 Analyse Graphique 10
2.4 Variation de [B] 11
2.4.1 Modélisation Mathématique 11
2.4.2 Analyse Numérique et Graphique 13

Conclusion et perspectives **15**

Bibliographie **16**

A Courbes Gnuplot **17**

B Codes de l'analyse numérique **19**

Introduction

Depuis de nombreuses années, les réactions chimiques non linéaires et oscillantes ont beaucoup été étudiées, caractérisées et modélisées, notamment le système du Brusselator. En effet, celui-ci décrit une réaction chimique oscillante premièrement décrite par Ilya Prigogine en 1968. C'est le plus petit modèle présentant des comportements oscillants. Le Brusselator est une réaction autocatalytique caractérisé de la manière suivante :

$$\begin{aligned} A &\to X \\ 2X + Y &\to 3X \\ B + X &\to Y + D \\ X &\to E \end{aligned}$$

Les deux espèces X et Y sont celles qui nous intéressent car ce sont des espèces autocatalytiques. Nous chercherons donc à étudier leurs évolutions selon différentes conditions initiales. Il sera aussi question d'une étude de stabilité des points d'équilibre de la réaction. De plus, les composants A et B étant en large excès, nous pouvons les considérer comme des constantes et finalement, D et E étant des produits de réactions, nous ne les impliquons pas dans notre modèle.

Ainsi nous suivrons le plan suivant. Dans une première partie, nous modéliserons le Brusselator par un système d'équations différentielles. Nous étudierons également un point d'équilibre du système et nous en étudierons sa stabilité d'après des critères définis. Dans une deuxième partie, différentes méthodes d'analyse numérique seront utilisées afin d'étudier le Brusselator, comme Euler ou Runge-Kutta 4. Nous utiliserons la méthode la plus fiable afin d'approximer au maximum le comportement réel du Brusselator. Finalement, nous étudierons ce modèle avec un changement de conditions initiales (prise en compte de la variation de B), nous pourrons ainsi également l'analyser numériquement et étudier les différences avec le premier modèle.

Chapitre 1

Méthodologie, organisation du travail

La cinétique chimique englobe une grande quantité de sujets d'études et pour notre projet nous nous sommes concentrés sur les réactions oscillantes et plus principalement sur le Brusselator. Pour nous quatre, c'était quelque chose de totalement nouveau et nous avons donc choisi de faire des recherches sur le sujet durant les trois premières séances.

Après plusieurs heures de lecture nous avons mieux compris de quoi il s'agissait et nous avons compris que ce problème nécessitait de comprendre plusieurs notions mathématiques (jacobienne, valeurs propres, point critique, stabilité . . .). Mais aussi que cela nécessiterait de faire un programme afin de simuler informatiquement comment évolue le problème dans la réalité.

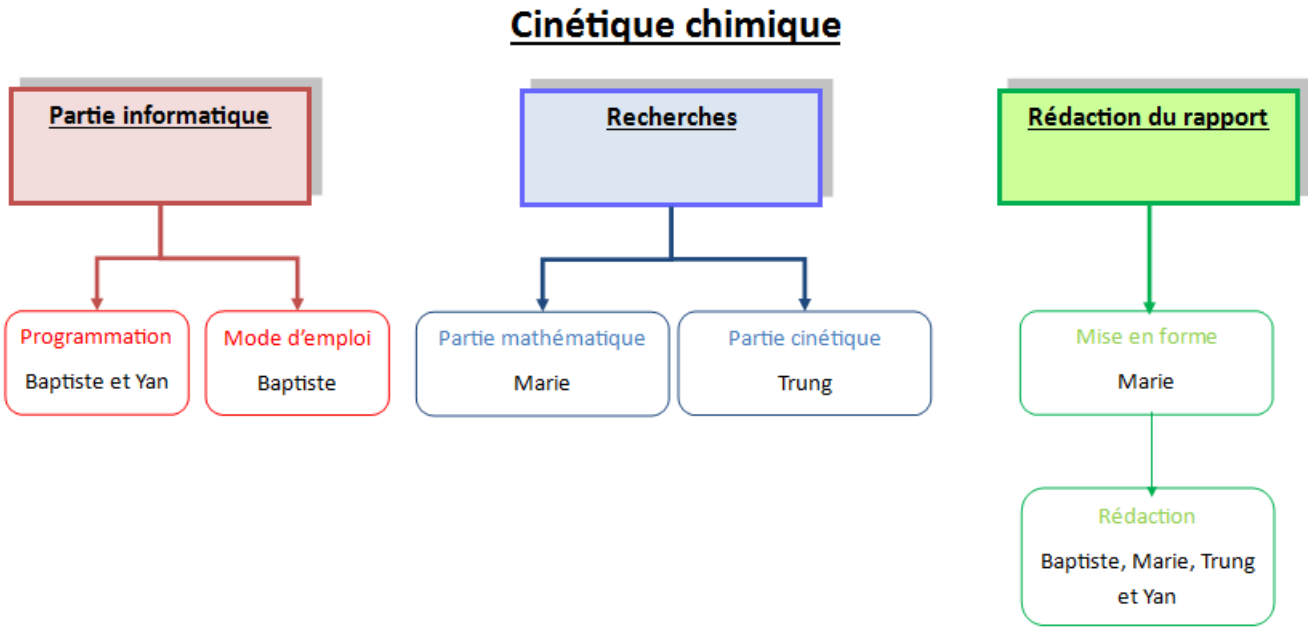
Nous nous sommes donc séparés en deux binômes : Baptiste et Yan sur la partie informatique, et Marie et Trung sur la partie mathématique.

En effet lors du semestre 3, Yan et Baptiste avaient codé un programme en C pour leur projet mathématique « Résolution numérique des équations différentielles ». Ce programme avait pour but de résoudre le problème de l'oscillateur de Van der Pol mais il constituait une base suffisamment solide (que ce soit en termes de structure de code ou de connaissances en programmation) pour permettre d'avoir un nouveau programme pour un nouveau système d'équations différentielles.

Ils ont donc créé puis amélioré le programme au fil des séances afin d'obtenir un générateur de courbes modélisant le problème du Brusselator.

Quant à Trung et Marie, ils ont pris en charge la partie mathématiques et recherches du projet. En effet Marie a choisi la spécialisation GM/ASI et avait donc plus de facilités à comprendre les notions mathématiques nécessaires pour le projet et a permis de mieux comprendre comment utiliser les équations différentielles dans la partie informatique. Trung a lui choisi la spécialisation CFI/MRIE et s'est donc occupé de rechercher plus en détail les concepts chimiques qu'illustre le modèle du Brusselator.

Enfin pour l'élaboration du compte rendu et du poster nous nous sommes répartis le travail en fonction de ce que nous avions fait de manière à écrire avec plus de pertinence les différentes parties du rapport.


Cinétique chimique
Partie informatique
Programmation
Baptiste et Yan
Mode d'emploi
Baptiste
Recherches
Partie mathématique
Marie
Partie cinétique
Trung
Rédaction du rapport
Mise en forme
Marie
Rédaction
Baptiste, Marie, Trung et Yan

Chapitre 2

Travail réalisé et résultats

2.1 Modélisation mathématique et étude de la stabilité

2.1.1 Système d'équations différentielles

Tout d'abord, afin d'étudier le Brusselator, nous avons posé les hypothèses suivantes :
— Concentrations de A et B considérées comme constantes.
— Constantes de vitesse égales à 1 ($k_1 = k_2 = k_3 = k_4 = 1$).
Grâce à la loi de Van't Hoff appliquée à chacune des 4 étapes du mécanisme, nous obtenons les vitesses de réactions suivantes :

$$\begin{aligned} v_1 &= [A] \\ v_2 &= [X]^2[Y] \\ v_3 &= [X][B] \\ v_4 &= [X] \end{aligned}$$

La loi de Van't Hoff et les vitesses nettes de production nous permettent d'obtenir le système d'équations différentielles suivant :

$$\begin{cases} \dot{x} = 1 - (b+1)x + ax^2y \\ \dot{y} = bx - ax^2y \end{cases}$$

2.1.2 Stabilité

Point d'équilibre

Ensuite, nous avons recherché les points d'équilibre de ce système différentiel. Par définition, un point d'équilibre est atteint lorsque la vitesse du produit est constante, c'est à dire quand $\dot{x} = \dot{y} = 0$. Nous obtenons donc le système à deux inconnues suivant :

$$\begin{cases} 0 = 1 - (b+1) + ax^2y \ (1) \\ 0 = bx - ax^2y \qquad\quad (2) \end{cases}$$

$(1) + (2)$ donne : $1 - x = 0$ d'où $x = 1$. Ainsi (2) donne : $y = \frac{b}{a}$. Donc, le point $(1, \frac{b}{a})$ est un point d'équilibre du système. Etudions sa stabilité. Pour cela, nous devons calculer la matrice Jacobienne du système différentiel.
Soit : $J = \begin{pmatrix} \frac{\partial(1)}{\partial x} & \frac{\partial(1)}{\partial y} \\ \frac{\partial(2)}{\partial x} & \frac{\partial(2)}{\partial y} \end{pmatrix} = \begin{pmatrix} -b + 2axy - 1 & ax^2 \\ b - 2axy & -ax^2 \end{pmatrix}$. Ensuite, on applique la Jacobienne au

point d'équilibre $(1;\frac{b}{a})$: $J_{éq} = J(1;\frac{b}{a}) = \begin{pmatrix} b-1 & a \\ -b & -a \end{pmatrix}$.
Afin de déterminer la stabilité des points d'équilibre, nous recherchons les valeurs propres de cette Jacobienne et un critère sur celles-ci pour évaluer la stabilité. Ainsi :

$$\begin{aligned} det(J_{éq} - \lambda Id_2) &= \begin{vmatrix} b-1-\lambda & a \\ -b & -a-\lambda \end{vmatrix} \\ &= \lambda^2 + \lambda(1+a-b) + a \\ &= P(\lambda) \end{aligned}$$

On résout $P(\lambda) = 0 \Leftrightarrow \lambda^2 + \lambda(1+a-b) + a = 0$

$\Delta = (1+a-b)^2 - 4a$

Nous cherchons à étudier le signe de Δ. Si $\Delta \geq 0$ alors $-4a \geq (1+a-b)^2$. Or $-4a \leq 0$. Par l'absurde, on montre $\Delta < 0$. Il y a donc 2 valeurs propres distinctes : $\lambda_{\pm} = \frac{1}{2}(b-a-1 \pm i\sqrt{4a-(1+a-b)^2}$.

Recherche d'un critère

Maintenant, cherchons un critère de stabilité sur les valeurs propres du système. On réalise le développement de Taylor au voisinage du point d'équilibre. Pour cela, on pose $f(x;y) = \dot{x}$ et $g(x;y) = \dot{y}$. D'où :

$$\begin{aligned} &\begin{cases} \dot{x} &= f(x;y) \backsimeq f(x_{éq};y_{éq}) + (x-x_{éq})\frac{\partial f}{\partial x}(x_{éq};y_{éq}) + (y-y_{éq})\frac{\partial f}{\partial y}(x_{éq};y_{éq}) \\ \dot{y} &= g(x;y) \backsimeq g(x_{éq};y_{éq}) + (x-x_{éq})\frac{\partial g}{\partial x}(x_{éq};y_{éq}) + (y-y_{éq})\frac{\partial g}{\partial y}(x_{éq};y_{éq}) \end{cases} \\ \Longleftrightarrow &\begin{pmatrix} f(x;y) - f(x_{éq};y_{éq}) \\ g(x;y) - g(x_{éq};y_{éq}) \end{pmatrix} = \begin{pmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} \end{pmatrix} * \begin{pmatrix} x-x_{éq} \\ y-y_{éq} \end{pmatrix} \end{aligned}$$

On pose $\dot{\overrightarrow{C}} = \begin{pmatrix} f(x;y) - f(x_{éq};y_{éq}) \\ g(x;y) - g(x_{éq};y_{éq}) \end{pmatrix}$, $J(x_{éq};y_{éq}) = \begin{pmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} \end{pmatrix}_{\lfloor (x_{éq};y_{éq})}$ et donc $\overrightarrow{C} = \begin{pmatrix} x-x_{éq} \\ y-y_{éq} \end{pmatrix}$.
On sait que J diagonalisable, d'où J peut s'écrire PDP^{-1}. On a donc :

$$\begin{aligned} \dot{\overrightarrow{C}} &= PDP^{-1}\overrightarrow{C} \\ \dot{P^{-1}\overrightarrow{C}} &= DP^{-1}\overrightarrow{C} \end{aligned}$$

En posant $\overrightarrow{Z} = P^{-1}\overrightarrow{C}$ avec $Z = \begin{pmatrix} z \\ w \end{pmatrix}$, on obtient : $\begin{pmatrix} \dot{z} \\ \dot{w} \end{pmatrix} = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}\begin{pmatrix} z \\ w \end{pmatrix} = \begin{pmatrix} \lambda_1 z \\ \lambda_2 w \end{pmatrix}$. Ce qui nous conduit au système suivant :

$$\begin{aligned} \begin{cases} \dot{z} - \lambda_1 z &= 0 \\ \dot{w} - \lambda_2 w &= 0 \end{cases} &\Longleftrightarrow \begin{cases} z(t) &= z_0 e^{\lambda_1 t} \\ w(t) &= w_0 e^{\lambda_2 t} \end{cases} \\ &\Longleftrightarrow \begin{cases} z(t) &= z_0 e^{Re(\lambda_1 t)t} e^{iIm(\lambda_1)t} \\ w(t) &= w_0 e^{Re(\lambda_2 t)t} e^{iIm(\lambda_2)t} \end{cases} \end{aligned}$$

Or, on sait que $\lim_{t\to+\infty} \overrightarrow{C} = \begin{pmatrix}0\\0\end{pmatrix}$ car $x \underset{+\infty}{\to} x_{éq}$ donc $Z \underset{+\infty}{\to} \begin{pmatrix}0\\0\end{pmatrix}$. En conclusion, pour avoir cette limite nulle, il nous faut : $e^{Re(\lambda_1)t} \underset{+\infty}{\to} 0$et $e^{Re(\lambda_2)t} \underset{+\infty}{\to} 0$. Ainsi, le critère de stabilité est la négativité des parties réelles des valeurs propres de la Jacobienne du système. Donc, il est nécessaire d'avoir $b - a - 1 < 0$ pour un avoir un point d'équilibre stable.

2.2 Analyse Numérique

Afin de modéliser ce problème, nous avons modifié le programme que Yan et Baptiste avaient créé en projet maths en langage C le semestre précédent pour l'appliquer au Brusselator. Initialement, notre programme permettait la résolution de systèmes d'équations linéaires sous la forme d'un problème de Van der Pol.

Nous avons donc dû modifier les équations initiales pour modéliser le Brusselator. Pour obtenir les deux équations suivantes :

$$\begin{cases}\dot{x} = 1 - (b+1)x + ax^2y\\ \dot{y} = bx - ax^2y\end{cases}$$

Nous avons utilisé un vecteur X à deux composantes $X[0]$ et $X[1]$ correspondant respectivement à X et Y ainsi qu'un vecteur F à deux composantes $F[0]$ et $F[1]$ correspondant respectivement à $\dot{X}$ et $\dot{Y}$. On obtient donc :

$$\begin{aligned}F[0] &= 1 - (b+1)X[0] + aX^2[0]X[1]\\ F[1] &= bX[0] - aX^2[0]X[1]\end{aligned}$$

Maintenant que notre système d'équations est posé et codé en C, nous pouvons tenter de le résoudre. Pour se faire, nous avons utilisé la méthode Runge-Kutta d'ordre 4, qui permet d'approximer les solutions de ce genre de systèmes d'équations différentielles avec l'équation :

$$\begin{aligned}y_n + 1 = y_n + (\frac{h}{6})(k_0 + 2k_1 + 2k_2 + k_3) \text{ où}\\ k_0 = f(t_n, y_n)\\ k_1 = f(t_n + h_2, y_n + h_2 * k_0)\\ k_2 = f(t_n + h_2, y_n + h_2 * k_1)\\ k_3 = f(t_n + h, y_n + h)\end{aligned}$$

On utilise donc quatre évaluations de f, ce qui apporte une approximation plutôt précise, mais qui nécessite un temps de calcul assez important pour des fonctions compliquées.
Parlons un peu du programme : nous commençons par créer le système d'équations avec la fonction "Fonction" et à allouer de l'espace mémoire à notre matrice Y qui va contenir toutes les solutions Y_n du problème. Ces solutions sont stockées dans un fichier à l'exécution du programme, et sont ensuite utilisées pour tracer des courbes.
Dans cette fonction RungeKutta, nous avons donc les quatre vecteurs k_0, k_1, k_2 et k_3, qui sont les évaluations de f que l'on doit stocker pour pouvoir les utiliser aux "itérations" suivantes, ainsi que le pas h. Une fois les K calculés, on remplit la matrice solution Y avec la formule présentée ci-après :

$$y_n + 1 = y_n + (\frac{h}{6})(k_0 + 2k_1 + 2k_2 + k_3)$$

Ce qui correspond dans le programme à :

$$y_i[k] = y_i[k] + \frac{h}{6}(K_0[k] + 2K_1[k] + 2K_2[k] + K_3[k])$$

Enfin, dans le programme principal, avant d'appeler la fonction RungeKutta, nous pouvons régler les conditions initiales : t_0 et T correspondent aux temps min et max. N correspond au nombre d'itérations (un N élevé permet un grand nombre de solutions calculées, donc un grand nombre de points, donc une courbe plus lisse. N et m correspondent donc à la taille de la matrice solution.
Le vecteur a possède ici deux coordonnées $a[0]$ et $a[1]$ correspondant respectivement aux valeurs choisies pour X_0 et Y_0, les concentrations initiales de X et Y. Les paramètres a et b présents dans les équations du système n'ont pas de variables, nous les entrons directement au moment de créer les formules au début dans la fonction "Fonction". Ainsi, nous pouvons facilement changer les conditions initiales et voir ce qui en résulte sur les courbes que l'on trace.

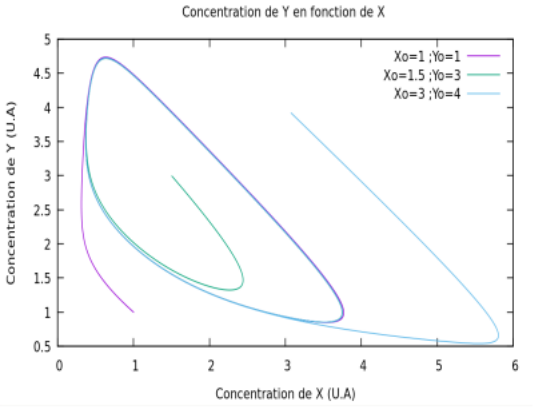
Après avoir calculé les approximations de solutions du système d'équations différentielles, nous pouvons donc, à la fin de notre programme principal, tracer des courbes :la concentration de X en fonction de la concentration de Y, et les concentrations de X et de Y en fonction du temps. Nous pouvons obtenir plusieurs versions de ces courbes en faisant varier les conditions initiales (d'autres courbes avec d'autres paramètres sont disponibles en annexe).

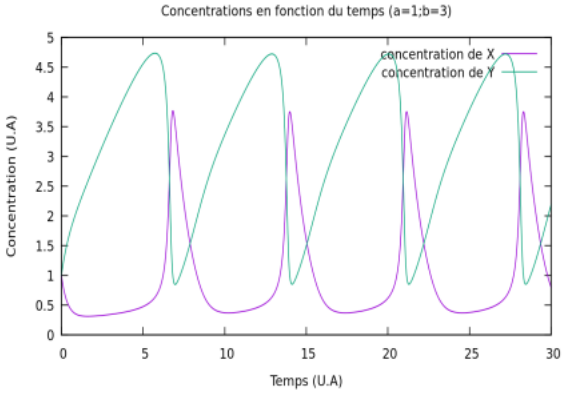
2.3 Analyse Graphique

Pour étudier graphiquement ce genre de système d'équations différentielles possédant des paramètres, il faut se pencher brièvement sur certaines notions : les bifurcations. En effet, selon les paramètres choisis, ces systèmes peuvent avoir différents comportements aymptotiques. Ils peuvent notamment tendre vers un point fixe (équilibre), ou bien vers un cycle limite. Également, la notion d'attracteur est importante à connaître : "Dans l'étude des systèmes dynamiques, un attracteur (ou ensemble-limite) est un ensemble ou un espace vers lequel un système évolue de façon irréversible en l'absence de perturbations" (Wikipédia). Une bifurcation est donc un changement d'état d'un attracteur, par exemple lorsque l'attracteur du système était un équilibre et devient un cycle. La valeur du paramètre que l'on a modifié pour arriver à cette bifurcation est appelée valeur de bifurcation.

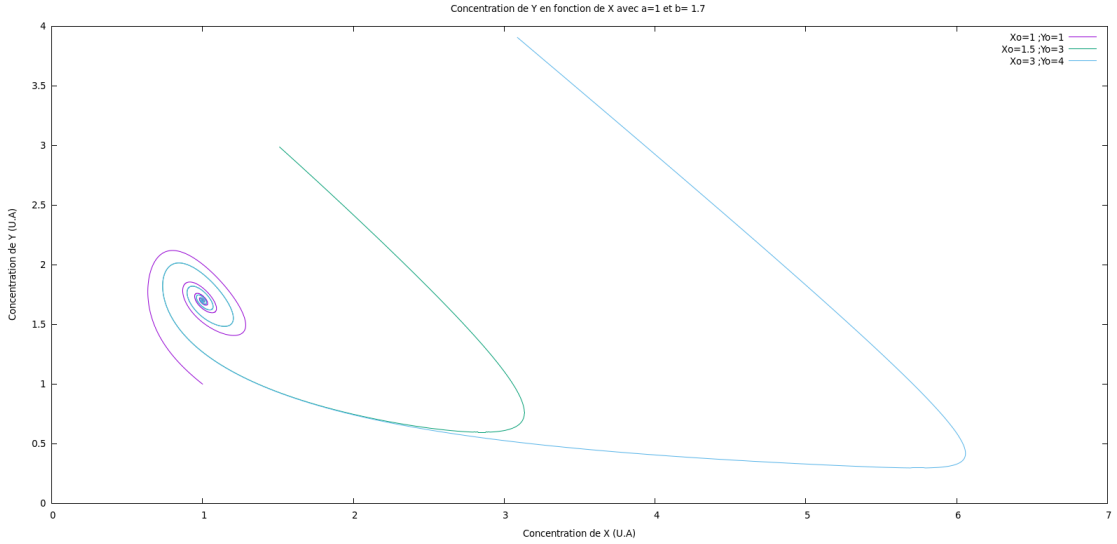
Dans notre cas, le point fixe perd sa stabilité et nous avons vu que le critère de stabilité est la négativité des parties réelles des valeurs propres de la Jacobienne du système, donc notre cas est une bifurcation de Hopf. Nous pouvons déduire de nos calculs précédents que cette bifurcation se produit pour $b = a + 1$.

Nous ne détaillerons pas plus ces méthodes d'analyse, car elles reposent sur de nombreuses notions qui nous sont inconnues, et dont le détail ne serait pas tout à fait à propos dans ce rapport (voir le pdf "Brusselator" cité dans la bibliographie pour plus de détails). Cependant, la connaissance de ces quelques notions va nous permettre une meilleure analyse des courbes que nous avons obtenues grâce à notre programme.





Ces deux courbes montrent que les concentrations de X et de Y en fonction du temps sont périodiques. La concentration de Y en fonction de la concentration de X pour différentes valeurs des conditions initiales X_0 et Y_0 n'apporte pas beaucoup d'informations. Ici, on a $a = 1$ et $b = 3$, mais en faisant varier ces paramètres on peut obtenir, pour $a = 1$ et $b = 1.7$, la courbe suivante, où l'on peut voir un point fixe, donc un équilibre du système en $(1; 1.7)$:



D'autres courbes ont été mises en annexe pour ne pas encombrer le rapport.

2.4 Variation de [B]

2.4.1 Modélisation Mathématique

Nous avons jugé intéressant de traiter le Brusselator avec des hypothèses différentes de celles étudiées précédemment. Pour ce faire, nous avons cette fois-ci considéré B comme non constante. Sa concentration varie donc et on pose z cette concentration en fonction du temps. B est injecté dans le mélange à la vitesse ν. On applique la loi de Van't Hoff au système

initiale :

$$\begin{aligned} v_1 &= [A] \\ v_2 &= [X]^2[Y] \\ v_3 &= [B][X] \\ v_4 &= [X] \end{aligned}$$

Ensuite, nous obtenons les équations suivantes :

$$\begin{cases} \frac{d[X]}{dt} & = v_1 - 2v_2 + 3v_2 - v_3 = [A] + [X]^2[Y] - ([B]-1)[X] \\ \frac{d[Y]}{dt} & = v_3 - v_3 = [B][X] - [X]^2[Y] \\ \frac{d[B]}{dt} & = -v_3 + \nu = -[B][X] + \nu \text{ avec } \nu \text{ la vitesse d'injection de B} \end{cases}$$

$$\iff \begin{cases} \dot{x} & = a - (z+1)x + x^2 y = f_1(x;y;z) \\ \dot{y} & = zx - x^2 y \qquad\qquad = f_2(x;y;z) \\ \dot{z} & = -zx + \nu \qquad\qquad = f_3(x;y;z) \end{cases}$$

De même que dans le premier cas, nous avons calculé la Jacobienne associée au point d'équilibre $(1;\nu;\nu)$: $J_{éq} = \begin{pmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial z} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial z} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & \frac{\partial f_3}{\partial z} \end{pmatrix} = \begin{pmatrix} \nu - 1 & 1 & -1 \\ -\nu & -1 & 1 \\ -\nu & 0 & -1 \end{pmatrix}$. De même, à partir de la Jacobienne, nous recherchons les valeurs propres associées grâce au calcul du polynôme caractéristique ci-dessous :

$$\begin{aligned} det(J_{éq} - \lambda I d_3) &= \begin{vmatrix} \nu - 1 - \lambda & 1 & -1 \\ -\nu & -1 - \lambda & 1 \\ -\nu & 0 & -1 - \lambda \end{vmatrix} \\ &= \lambda^3 - \lambda^2(\nu - 3) - \lambda(2\nu - 3) + 1 \\ &= P(\lambda) \end{aligned}$$

A partir de $P(\lambda)$, nous allons étudier le critère de stabilité de ce point d'équilibre en recherchant une condition sur ν. Pour cela, nous utilisons le critère de Routh-Hurwitz. Pour cela nous étudions $P(\lambda)$ comme un polynôme de Hurwitz. Il est donc de la forme :

$$\begin{aligned} P(\lambda) &= \lambda^3 - \lambda^2(\nu - 3) - \lambda(2\nu - 3) + 1 \\ &= a_3\lambda^3 + a_2\lambda^2 + a_1\lambda + a_0 \end{aligned}$$

Il s'agit maintenant de construire le tableau des coefficients de Routh et d'en étudier le critère. Pour cela, comme P de degré n=3 (impair), le tableau sera de la forme :

a_3	a_1
a_2	a_0
b_1	0
c_0	

Avec $b_1 = -\frac{1}{a_2}\begin{vmatrix} a_3 & a_1 \\ a_2 & a_0 \end{vmatrix}$ et $c_0 = -\frac{1}{b_1}\begin{vmatrix} a_2 & a_0 \\ b_1 & 0 \end{vmatrix}$. On obtient donc le tableau suivant :

$a_3 = 1$	$a_1 = 3 - 2\nu$
$a_2 = 3 - \nu$	$a_0 = 1$
$b_1 = \frac{2\nu^2 - 9\nu + 8}{3 - \nu}$	0
$c_0 = 1$	

D'après Routh-Hurwitz, le système est stable si et seulement si les pivots (première colonne) sont strictement positifs. On recherche donc une condition sur ν pour que tous les pivots soient positifs. On obtient le système suivant :

$$\begin{cases} 3-\nu & >0 \\ 2\nu^2-9\nu+8 & >0 \end{cases}$$
$$\iff \begin{cases} \nu & <3 \\ \nu & \in]0, \frac{9-\sqrt{17}}{4}[\cup]\frac{9+\sqrt{17}}{4}, +\infty[\end{cases}$$
$$\iff \nu \in]0, \frac{9-\sqrt{17}}{4}[$$

Ainsi, le point d'équilibre $(1;\nu;\nu)$ est stable si et seulement si $\nu \leq 1.22$.

2.4.2 Analyse Numérique et Graphique

Il s'agit maintenant d'adapter notre programme à ce changement de conditions initiales, c'est à dire à faire en sorte que B soit désormais injecté dans la solution à une certaine vitesse ν dont on doit faire varier la valeur. On a donc à présent une nouvelle équation différentielle pour B :

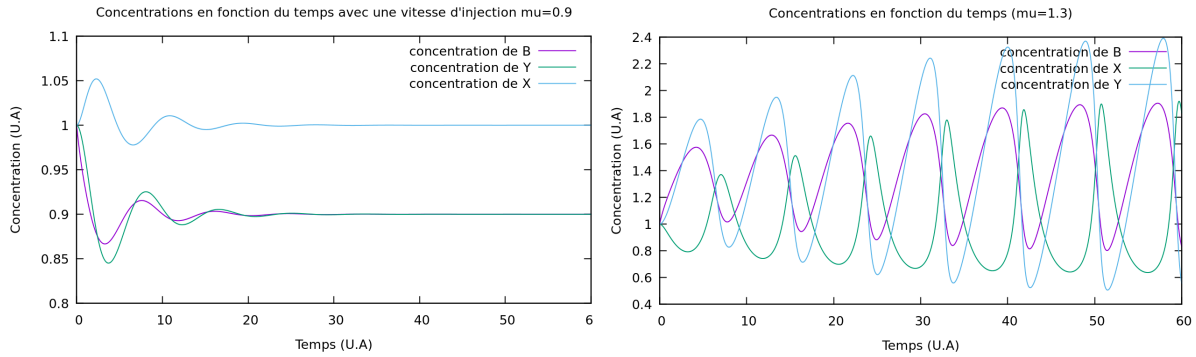
$$[\dot{B}] = [B][X] + \nu$$

Dans notre programme, nous avons donc ajouté une troisième composante au vecteur F, telle que l'équation ci-dessus devient :

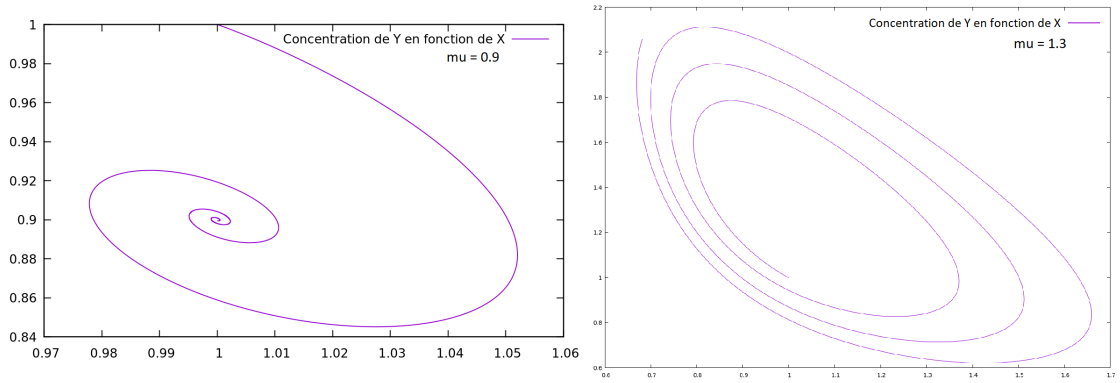
$$F[2] = -y[2] * y[0] + \nu$$

Également, la matrice solution Y gagne une colonne dans laquelle on met les solutions de l'équation de B.

Avec ces modifications, nous pouvons à présent tracer les mêmes courbes que précédemment, en stockant les solutions dans des fichiers (voir programme en annexe). Nous obtenons ainsi des courbes différentes pour les concentrations de X et Y en fonction du temps, qui changent selon la valeur de ν. Nous pouvons également tracer la concentration de B en fonction du temps, ou éventuellement B en fonction de X ou de Y.



Ici on remarque que la vitesse ν à laquelle B est introduit impacte le comportement des concentrations en fonction du temps : on a des concentrations qui convergent vers leur valeur initiale pour $\nu = 0.9$, alors qu'elles divergent pour $\nu = 1.3$.



On voit ici une bifurcation sur la concentration de Y en fonction de la concentration de X : on passe d'un point fixe de coordonnées $(1; 0.9)$ pour $\nu = 0.9$ à un cycle limite pour $\nu = 1.3$.

D'autres courbes ont été mises en annexe pour ne pas encombrer le rapport.

Conclusion et perspectives

En conclusion, lors de ce projet, nous avons modélisé graphiquement le Brusselator. Pour cela, nous avons réalisé mathématiquement une étude sur la stabilité de ces points d'équilibre. Nous avons ensuite analysé numériquement ce système grâce au système d'équations différentielles associé au Brusselator. Nous avons étudié différents types de méthodes d'analyse numérique et en avons conclu que RK4 était la plus appropriée pour ce projet. Nous avons ainsi pu étudier la bifurcation de Van't Hoff, les cycles limites, les points d'équilibre et l'évolution de ce système. Les notions de réaction autocatalytique et de cycles limites se sont éclaircies et nous avons pu concrètement étudier ces notions.

Grâce à cet E.C. projet, nous avons pu appliquer concrètement des aspects très théoriques de nos cours à l'INSA. En effet, les modélisations et études mathématiques appliquées à ce système ont permis de rendre plus concret nos cours et de nous rendre compte de l'utilité et de l'utilisation des théorèmes vus. De plus, la partie informatique de notre projet a permis à deux étudiants du projet d'approfondir leurs travaux précédemment réalisés en Projet Mathématiques à l'INSA. Pour les deux autres, ce fût une première approche du codage en C et une nouvelle expérience d'analyse numérique. Finalement, ce projet a permis de réunir plusieurs matières habituellement étudiées séparément : la chimie, l'informatique et les mathématiques. Il a été très intéressant de faire le lien entre ces différentes thématiques et d'apprendre à les étudier ensemble. De plus, la diversité des parcours suivis par les étudiants de ce projet a permis un réel échange de connaissances et a été un réel avantage pour l'étude des différents aspects de ce projet. Chaque étudiant a pu y trouver son point d'intérêt et ainsi s'épanouir au sein de ce projet.

Pour la poursuite de ce projet dans les années à suivre, nous trouvons intéressant l'idée de poursuivre l'analyse numérique en utilisant, pourquoi pas, de nouvelles méthodes, les comparer entre elles. Pourquoi pas également coder en d'autres langages ? Ensuite, il serait judicieux d'étudier ce système avec plus de variables, de nouvelles conditions initiales afin de se ramener le plus proche de la réalité possible. L'aspect chimique de ces réactions pourrait également être plus approfondi. Les futurs étudiants auront donc déjà nos travaux entre les mains, et pourront pousser nos raisonnements plus loin, apporter leurs connaissances personnelles et le faire évoluer. La suite de cet E.C. pourrait se révéler très intéressante au niveau culturel, pluridisciplinaire et appliqué. Ce système offre de larges possibilités d'ouvertures et d'études.

Bibliographie

[1] LIEN INTERNET, https://fr.wikipedia.org/wiki/Brusselator (Valide à la date du 06/02/2018)

[2] LIEN INTERNET, http://theses.univ-lyon2.fr/documents/getpart.php?id=lyon2.2004.petitgirard_l&part=194438 (Valide à la date du 06/02/2018)

[3] LIEN INTERNET, https://www-dimat.unipv.it/boffi/teaching/download/Brusselator.pdf (Valide à la date du 06/02/2018)

[4] LIEN INTERNET, https://fr.wikipedia.org/wiki/Bifurcation_de_Hopf (Valide à la date du 13/02/2018)

[5] LIEN INTERNET, https://fr.slideshare.net/RohitAggarwal9/brusselator (Valide à la date du 06/02/20118)

[6] LIEN INTERNET, http://lat.univ-tlemcen.dz/dl/chikhaoui_2009.pdf (Valide à la date du 29/05/2018)

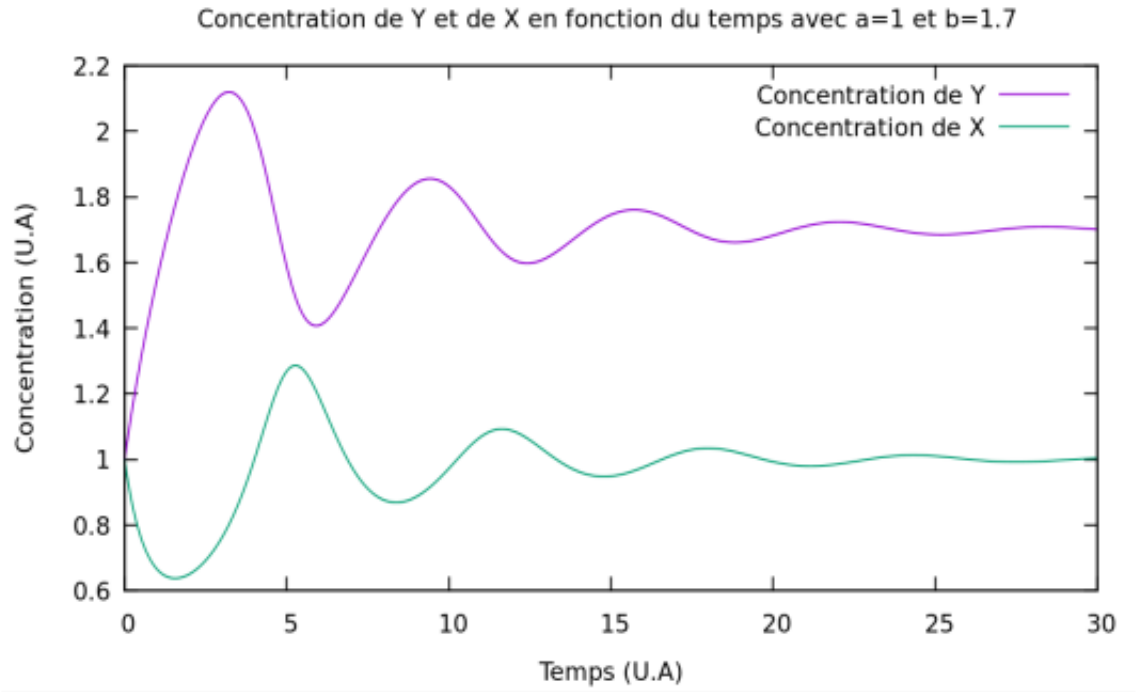
[7] LIEN INTERNET, http://www.jdotec.net/s3i/TD_Info/Routh/Routh.pdf (Valide à la date du 29/05/2018)

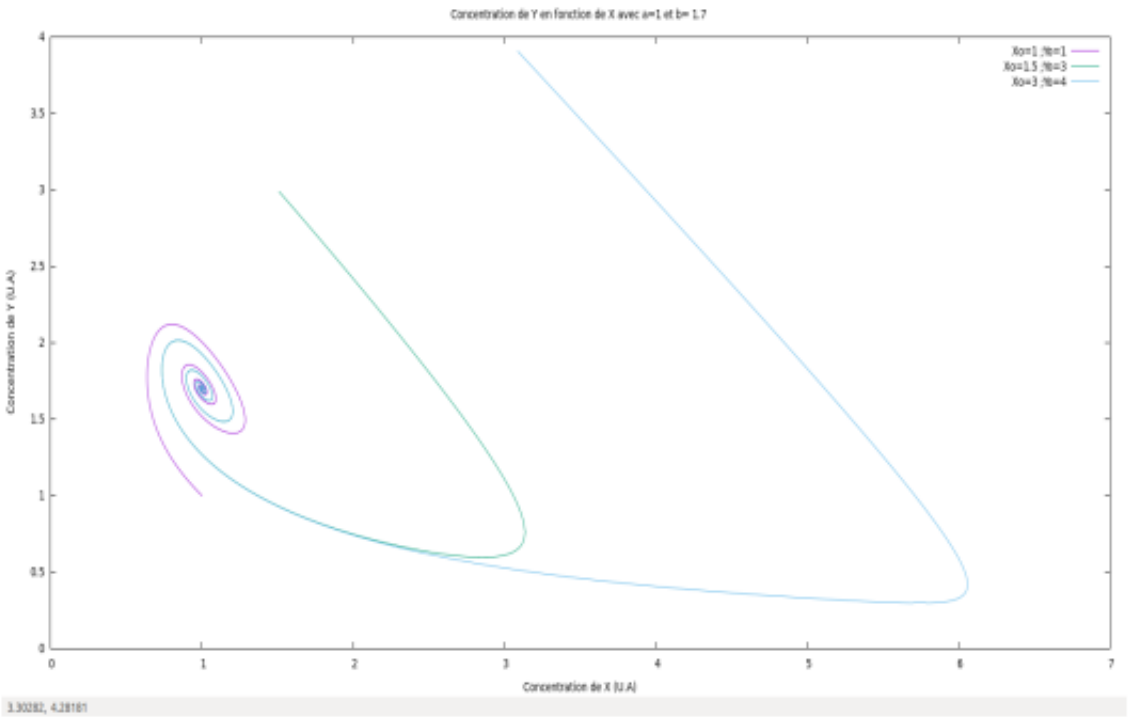
[8] LIEN INTERNET, https://www.obs-vlfr.fr/~irisson/teaching/ens/cours_bifurcations.pdf (Valide à la date du 29/05/2018)

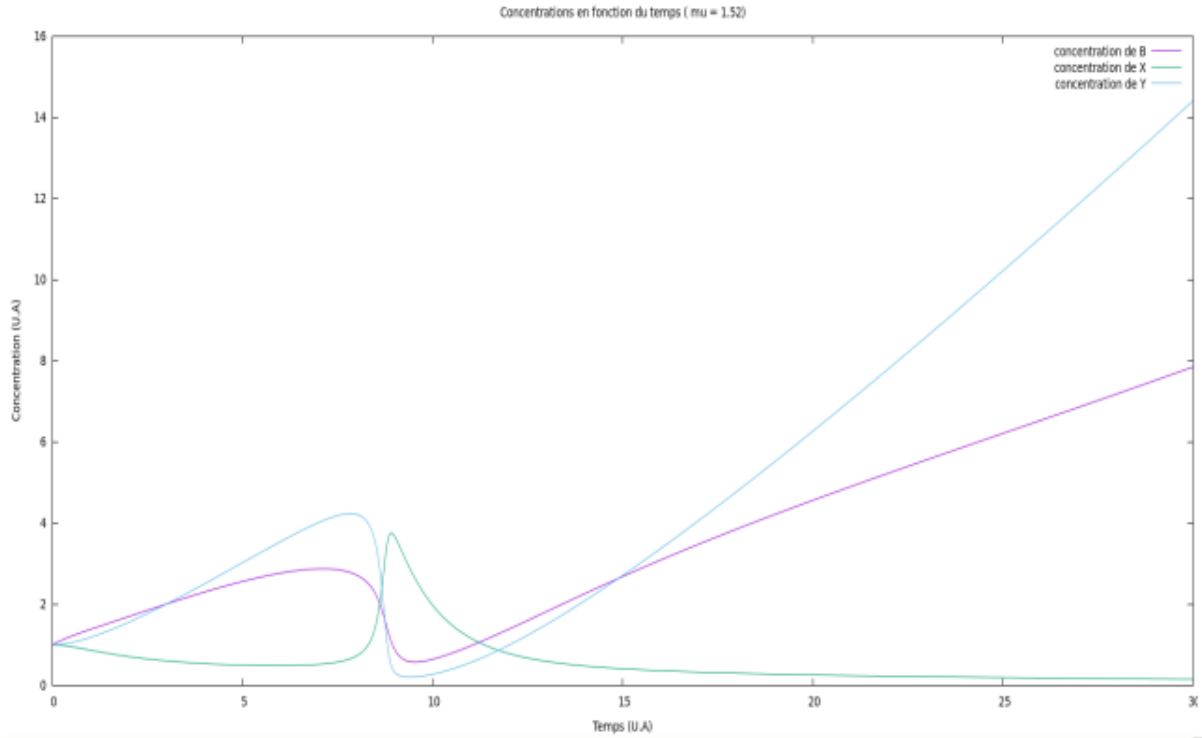
[9] LIEN INTERNET, https://fr.wikipedia.org/wiki/Attracteur (Valide à la date du 29/05/2018)

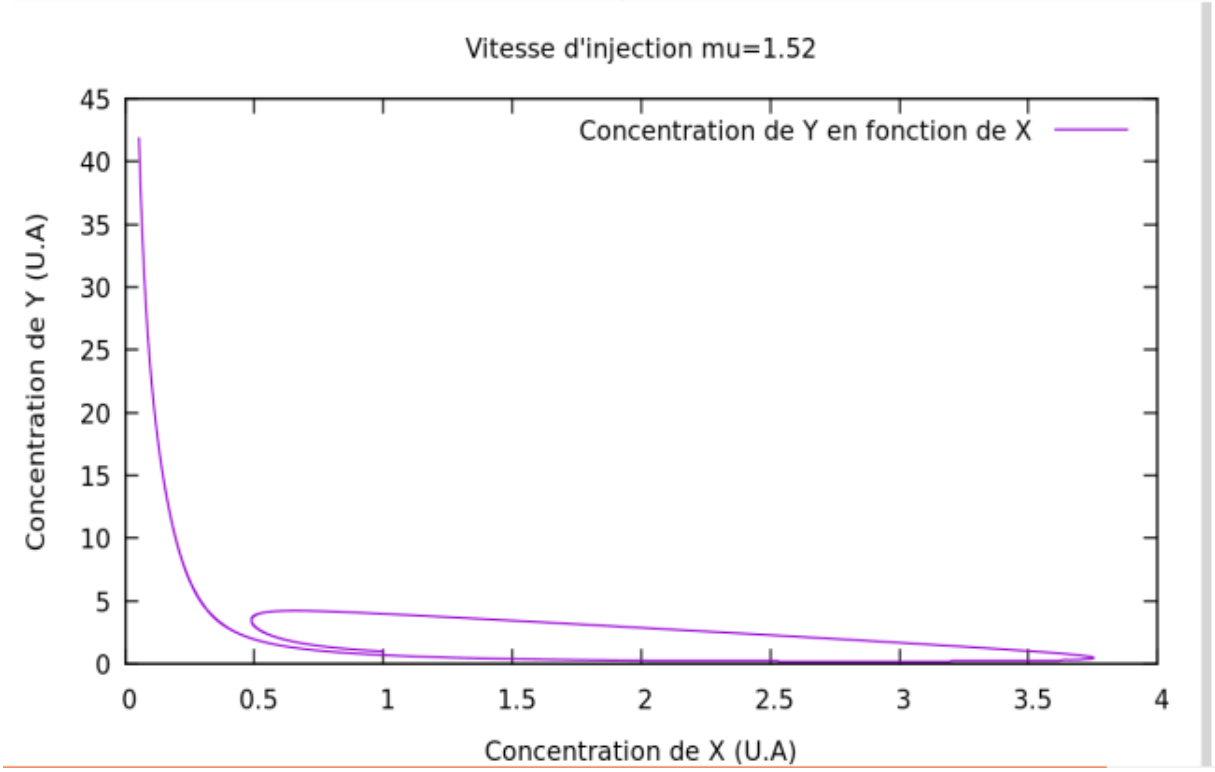
Annexe A

Courbes Gnuplot






Concentrations en fonction du temps (mu = 1.52)
concentration de B
concentration de X
concentration de Y
Concentration (U.A)
Temps (U.A)


Vitesse d'injection mu=1.52
Concentration de Y en fonction de X
Concentration de Y (U.A)
Concentration de X (U.A)

Annexe B

Codes de l'analyse numérique

Premier code (A et B considérées constantes) :

```
#include<stdlib.h>
#include<stdio.h>
#include<math.h>
#include<malloc.h>



double** MonAlloc(int N, int m) //attribue une taille  la matrice
{
int i;
double *vect, **mat;
vect=(double*)malloc((N*m)*sizeof(double));//malloc dja dfinit
mat=(double**)malloc(N*sizeof(double*)); //attribue une taille  un vecteur
for(i=0;i<N;i++)
{mat[i]=vect;
vect=vect+m;
}
return(mat);
}

double* fonction(double t ,double* y) // On prend a=1 et b=3
{
double* F;

F=(double*)malloc(2*sizeof(double));
F[0]=1 - 4*y[0] + y[0]*y[0]*y[1] ;
F[1]=3*y[0] - y[0]*y[0]*y[1]; //a=1 //b=3
return(F);
}


//--------------------
// Runge Kutta
//--------------------
   double** RungeKutta(double to,double T,int N,int m, double* a) //dfinit la
       fonction
      { //RungeKutta
```

```
          int i,k;
          double h,ti, *yinter,**y,*yi,*K0,*K1,*K2,*K3;


          yinter=(double*)malloc(m*sizeof(double));
       yi=(double*)malloc(m*sizeof(double));
K0=(double*)malloc(m*sizeof(double));
K1=(double*)malloc(m*sizeof(double));
K2=(double*)malloc(m*sizeof(double));
K3=(double*)malloc(m*sizeof(double));

       y=MonAlloc(N,m);

          h=(T-to)/N;
          for(k=0;k<m;k++) yi[k]=a[k];

          for(i=0;i<N;i++)
           {
        ti=i*h;

        K0=fonction(ti,yi);

        for(k=0;k<m;k++) yinter[k]=yi[k]+(h/(2.0))*K0[k];
K1=fonction(ti+(h/(2.0)),yinter);

for(k=0;k<m;k++) yinter[k]=yi[k]+(h/(2.0))*K1[k];
K2=fonction(ti+(h/(2.0)),yinter);

for(k=0;k<m;k++) yinter[k]=yi[k] + h*K2[k];
K3=fonction(ti+(h/(2.0)),yinter);

for(k=0;k<m;k++)
    {
       yi[k]=yi[k] + (h/(6.0))*(K0[k] + 2*K1[k] + 2*K2[k] + K3[k]);
       y[i][k]=yi[k];
    }
          }

        return(y);
       }


//-------------------------
// programme principal
//-------------------------
int main()
{
int N, m,i;
double to,t,T,*a,**Y;
FILE* unit;

to=0; T=30;
N=10000;
```

```
m=2;

a=(double*)malloc(m*sizeof(double));
Y=MonAlloc(N,m);

a[0]=1; a[1]=1;



      Y=RungeKutta(to,T,N,m,a); //appelle la fonction RungeKutta y en fct
          de x
      unit=fopen("vP6_RK1_Y_X","w+");
    fprintf(unit,"%f \t %f \n",a[0],a[1]);
   for(i=1;i<N;i++) fprintf(unit,"%f \t %f \n",Y[i][0],Y[i][1]);




    unit=fopen("vP6_RK1_X","w+"); //concentrationX
    fprintf(unit,"%f \t %f \n",0.0,a[0]);
   for(i=1;i<N;i++) fprintf(unit,"%f \t %f \n",i*0.003,Y[i][0]);

    unit=fopen("vP6_RK1_Y","w+"); //concentrationY
    fprintf(unit,"%f \t %f \n",0.0,a[1]);
   for(i=1;i<N;i++) fprintf(unit,"%f \t %f \n",i*0.003,Y[i][1]);
fclose(unit);

a[0]=1.5; a[1]=3;
Y=RungeKutta(to,T,N,m,a); //appelle la fonction RungeKutta y en fct de x
      unit=fopen("vP6_RK2_Y_X","w+");
   for(i=0;i<N;i++) fprintf(unit,"%f \t %f \n",Y[i][0],Y[i][1]);
fclose(unit);

a[0]=3; a[1]=4;
Y=RungeKutta(to,T,N,m,a); //appelle la fonction RungeKutta y en fct de
      unit=fopen("vP6_RK3_Y_X","w+");
   for(i=0;i<N;i++) fprintf(unit,"%f \t %f \n",Y[i][0],Y[i][1]);
fclose(unit);

return(0);
}
```

Deuxième code (B considérée variable) :

```
#include<stdlib.h>
#include<stdio.h>
#include<math.h>
#include<malloc.h>
5
10 double** MonAlloc(int N, int m) //attribue une taille  la matrice
{
int i;
double *vect, **mat;
vect=(double*)malloc((N*m)*sizeof(double));//malloc dja dfinit
mat=(double**)malloc(N*sizeof(double*)); //attribue une taille  un vecteur
15 for(i=0;i<N;i++)
{mat[i]=vect;
vect=vect+m;
}
return(mat);
20 }

double* fonction(double t ,double* y) // On prend a=1 et b=3
{
double* F;
25
F=(double*)malloc(2*sizeof(double));
F[0]=1 - (y[2]+1)*y[0] + y[0]*y[0]*y[1] ;
F[1]=y[2]*y[0] - y[0]*y[0]*y[1]; //a=1 //binitial=1
F[2]= -y[2]*y[0] +1.52;
30 return(F);
}



35 //-----------------
// Runge Kutta
//-----------------
   double** RungeKutta(double to,double T,int N,int m, double* a) //dfinit la
       fonction
      { //RungeKutta
40       int i,k;
         double h,ti, *yinter,**y,*yi,*K0,*K1,*K2,*K3;


         yinter=(double*)malloc(m*sizeof(double));
45     yi=(double*)malloc(m*sizeof(double));
K0=(double*)malloc(m*sizeof(double));
K1=(double*)malloc(m*sizeof(double));
K2=(double*)malloc(m*sizeof(double));
K3=(double*)malloc(m*sizeof(double));
50
      y=MonAlloc(N,m);
```

```
          h=(T-to)/N;
          for(k=0;k<m;k++) yi[k]=a[k];

          for(i=0;i<N;i++)
           {
       ti=i*h;

       K0=fonction(ti,yi);

       for(k=0;k<m;k++) yinter[k]=yi[k]+(h/(2.0))*K0[k];
K1=fonction(ti+(h/(2.0)),yinter);

for(k=0;k<m;k++) yinter[k]=yi[k]+(h/(2.0))*K1[k];
K2=fonction(ti+(h/(2.0)),yinter);

for(k=0;k<m;k++) yinter[k]=yi[k] + h*K2[k];
K3=fonction(ti+(h/(2.0)),yinter);

for(k=0;k<m;k++)
    {
      yi[k]=yi[k] + (h/(6.0))*(K0[k] + 2*K1[k] + 2*K2[k] + K3[k]);
      y[i][k]=yi[k];
    }
          }

       return(y);
       }


//------------------------
// programme principal
//------------------------
int main()
{
int N, m,i;
double to,t,T,*a,**Y;
FILE* unit;

to=0; T=30;
N=10000;
m=3;

a=(double*)malloc(m*sizeof(double));
Y=MonAlloc(N,m);

a[0]=1; a[1]=1; a[2]=1;



      Y=RungeKutta(to,T,N,m,a); //appelle la fonction RungeKutta y en fct
          de x
      unit=fopen("P6_RK1_Y_X","w+");
```

```
    fprintf(unit,"%f \t %f \n",a[0],a[1]);
   for(i=1;i<N;i++) fprintf(unit,"%f \t %f \n",Y[i][0],Y[i][1]);

Y=RungeKutta(to,T,N,m,a); //appelle la fonction RungeKutta b en fct de x
      unit=fopen("P6_RK1_B_X","w+");
    fprintf(unit,"%f \t %f \n",a[0],a[1]);
   for(i=1;i<N;i++) fprintf(unit,"%f \t %f \n",Y[i][0],Y[i][2]);

    unit=fopen("P6_RK1_B","w+"); //concentrationB
    fprintf(unit,"%f \t %f \n",0.0,a[1]);
   for(i=1;i<N;i++) fprintf(unit,"%f \t %f \n",i*0.003,Y[i][2]);

    unit=fopen("P6_RK1_X","w+"); //concentrationX
    fprintf(unit,"%f \t %f \n",0.0,a[0]);
   for(i=1;i<N;i++) fprintf(unit,"%f \t %f \n",i*0.003,Y[i][0]);

    unit=fopen("P6_RK1_Y","w+"); //concentrationY
    fprintf(unit,"%f \t %f \n",0.0,a[1]);
   for(i=1;i<N;i++) fprintf(unit,"%f \t %f \n",i*0.003,Y[i][1]);
fclose(unit);

a[0]=1.5; a[1]=3;
Y=RungeKutta(to,T,N,m,a); //appelle la fonction RungeKutta y en fct de x
      unit=fopen("P6_RK2_Y_X","w+");
   for(i=0;i<N;i++) fprintf(unit,"%f \t %f \n",Y[i][0],Y[i][1]);
fclose(unit);

a[0]=3; a[1]=4;
Y=RungeKutta(to,T,N,m,a); //appelle la fonction RungeKutta b en fct de
      unit=fopen("P6_RK3_B_X","w+");
   for(i=0;i<N;i++) fprintf(unit,"%f \t %f \n",Y[i][0],Y[i][2]);
fclose(unit);

return(0);
}
```