Modélisation des systèmes biologiques par systèmes de réactions chimiques
Université Lille 1
SV2, Master MOCAD

François Lemaire

12 octobre 2015

Table des matières

Presentation

Titre du cours : Modélisation des systèmes biologiques par systèmes de réactions chimiques

Organisation : 4 séances 1.5h Cours + 2h TD/TP

Objectifs du cours :

— modélisation déterministe par équations différentielles
— réduction approchée (simplification de systèmes)
— analyse qualitative (oscillation, stabilité, ...)

Evaluation :

— un examen final (après les 4 séances)
— rendus de TP
— petites interros en début de séance (contrôle de connaissance)

Chapitre 1

Introduction

1.1 Réactions chimiques

Les systèmes de réactions chimiques fournissent un formalisme simple pour la modélisation des systèmes dynamiques en biologie. Les réactions que l'on utilise sont inspirées de la chimie. Voici un exemple de réaction :

$$E + S \to C$$

Ici, E, S et C désignent des espèces chimiques (i.e. des types de molécules). La réaction signifie simplement qu'une molécule de E peut réagir avec une molécule S pour former la molécule de C. Pour l'instant on ne sait rien sur la fréquence de déclenchement de la réaction.

E et S sont les réactants de la réaction, C est le produit. Il pourrait y avoir plusieurs produits.

On peut également considérer la réaction inverse :

$$C \to E + S$$

Dans ce cas, c'est une molécule de C qui se transforme en une molécule de E et une molécule de S.

En regroupant les deux réactions précédentes, qui sont des réactions irréversibles, on peut former la réaction réversible :

$$E + S \rightleftharpoons C$$

Les différentes molécules de chaque espèce sont supposées indiscernables. À tout instant, et dans un espace précisé à l'avance (à l'intérieur d'une cellule par exemple), il y a un nombre bien défini de molécules E noté n_E. On peut également définir la concentration de E (notée $[E]$ voire E si il n'y a pas d'ambiguïté), par n_E/V, où V est le volume du compartiment choisi. Dans notre cours, il y aura (quasiment) toujours un seul compartiment, il n'y aura donc qu'un seul n_E et qu'une seule concentration $[E]$.

En biologie, on considère très souvent des réactions non équilibrées, i.e. des réactions qui ne conservent pas la quantité de matière. On parle alors de réactions chimiques généralisées. En voici un exemple :

$$C \to \varnothing$$

Cette réaction signifie que la molécule C est dégradée. On considère donc qu'elle disparaît : c'est une façon de dire qu'on ignore les produits de la réactions.

Pour finir, lorsqu'on s'intéresse à l'évolution dans le temps du nombre des espèces, on dit qu'on étudie la dynamique du sytème. Pour ce faire, il faut déterminer à quelles fréquences vont se produire les réactions. Dans ce cas, les flèches auront un attribut supplémentaire, en général une constante réelle positive. Par exemple :

$$E + S \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} C$$

L'interprétation de la constante dépendra du contexte. Toutefois, on peut dire que plus la constante est grande, plus la réaction a de chances de se produire.

1.2 Systèmes de réactions

Lorsqu'on veut modéliser un système biologique (ou une partie), ou même des dynamiques de populations, une approche classique est de considérer un système de réactions.

Un premier exemple très classique est celui de la dégradation enzymatique. Il décrit la transformation d'un substrat S en un produit P, en présence d'un enzyme E. Il y a formation d'un complexe intermédiaire C :

$$\begin{aligned} E + S &\underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} C \\ C &\xrightarrow{k_2} E + P \end{aligned}$$

Un second exemple concerne la dynamique de population. C'est une variante d'un modèle inventé par Alfred Lotka et Vito Volterra dans les années 1920. Les espèces H, L et Y représentent de l'herbe, des lapins et des lynx. La dynamique représentée par le modèle est la suivante : en présence d'herbe, les lapins se reproduisent. Les prélèvements effectués par les lapins ne sont pas suffisants pour faire baisser la quantité d'herbe, qu'on peut donc considérer comme constante. En présence de lapins, les lynx se multiplient (la rencontre d'un lynx et d'un lapin fait augmenter la population de lynx et diminuer la population de lapins). Pour des raisons non explicitées (maladie, chasse, grand âge,. . .), les lynx meurent.

$$\begin{aligned} H + L &\to H + 2L \\ L + Y &\to 2Y \\ Y &\to \varnothing \end{aligned}$$

Le langage SBML (Systems Biology Markup Language) est un langage basé sur XML, qui permet de décrire un système de réactions. C'est le standard incontournable du moment.

C'est en plus un format libre et documenté, et très largement utilisé dans les publications, et les logiciels de modélisation.

On peut citer également le format SBGN (Systems Biology Graphical Notation) qui est une représentation graphique standard de systèmes biologiques. Il permet de standardiser les diagrammes d'interactions entre espèces, de régulations de gènes.

Chapitre 2

Modélisation déterministe

Dans la réalité, les réactions chimiques se produisent à des instants plus ou moins aléatoires. Le déclenchement des réactions dépend entre autres, de la position des molécules, leur vitesse, de phénomènes extérieurs plus ou moins aléatoires, et l'on ne peut donc pas prédire l'évolution exacte. On parle alors de processus stochastique. Un même nombre d'espèces initiales peut produire différentes évolutions dans le temps. Pour illustrer ce point, étudions la simple dégradation $A \xrightarrow{k} \varnothing$, sur les figures suivantes.

La figure 2.1 montre deux évolutions possibles du nombre de molécules de A, en prenant $A(0) = 10$ (i.e. 10 molécules de A à t=0) et $k = 1$. L'axe des abscisses représente le temps. Cette simulation a été obtenue en utilisant l'algorithme de Gillespie, qui sera vu en SV3 dans les aspects stochastiques.

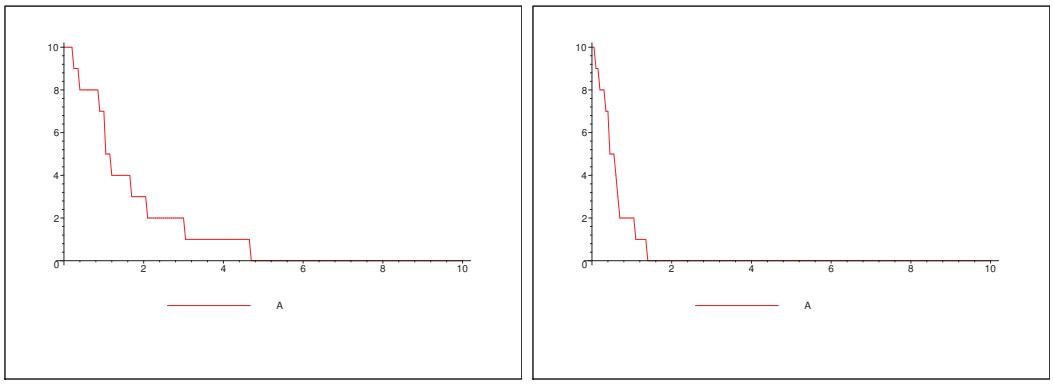


FIGURE 2.1 – Deux simulations différentes avec $A(0) = 10$ et k=1

La figure 2.2 montrent dix simulations stochastiques différentes sur le même graphe. Ce qu'il faut y voir est que les simulations, même si elles différent complétement, ont une évolution similaire.

Pour finir, on peut s'intéresser à la moyenne des simulations stochastiques : c'est ce que montre la figure 2.3, obtenue en faisant la moyenne de 1000 simulations stochastiques.

Ce qui est intéressant, c'est que l'on peut montrer que même si les simulations stochastiques sont aléatoires, leur moyenne est déterministe. Bien sûr, le calcul de cette moyenne

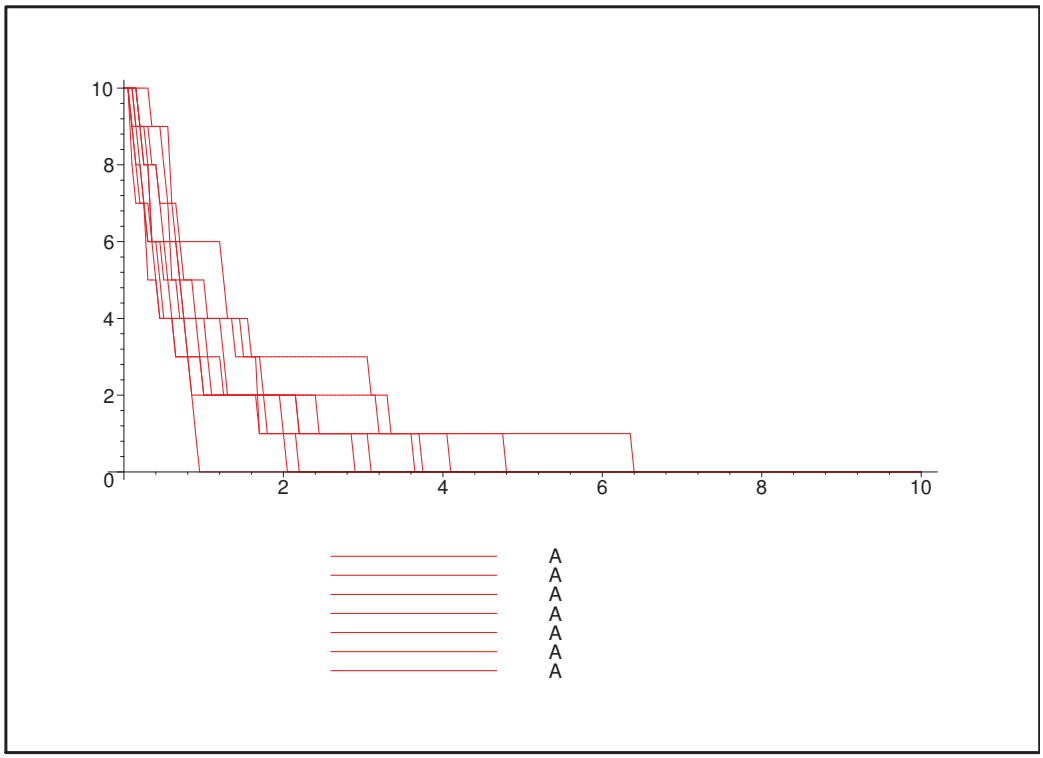


FIGURE 2.2 – Dix simulations superposées avec $A(0) = 10$ et k=1

n'est pas toujours suffisant (notamment dans le cas où peu de molécules sont présentes), mais cette moyenne possède quelques avantages :

- c'est une fonction à valeur dans $\mathbb{R}$ (et pas dans $\mathbb{N}$), on peut donc l'étudier avec des méthodes adaptées
- elle peut être approximée comme la solution d'un système d'équations différentielles (prochaine section)

2.1 Équations différentielles

Nous allons voir dans cette section comment traiter un exemple simple $A \xrightarrow{k} B$ par l'approche des équations différentielles, en faisant des rappels sur les équations différentielles et en introduisant les commandes Maple adaptées.

On ne le précisera plus dans la suite, mais on s'intéresse maintenant aux concentrations moyennes des d'espèces chimiques. Ainsi, $A(t)$ représente la moyenne de la concentration de A à l'instant t.

Fixons un instant t_0. Entre l'instant t_0 et $t_0 + \Delta_t$, la concentration B doit augmenter, d'une quantité que l'on va noter Δ. Ainsi, on a $B(t_0 + \Delta_t) = B(t_0) + \Delta$. Similairement, A va diminuer de Δ, d'où $A(t_0 + \Delta_t) = A(t_0) - \Delta$.

Des raisonnements complexes basés sur l'équation maîtresse (que nous ne verrons pas),

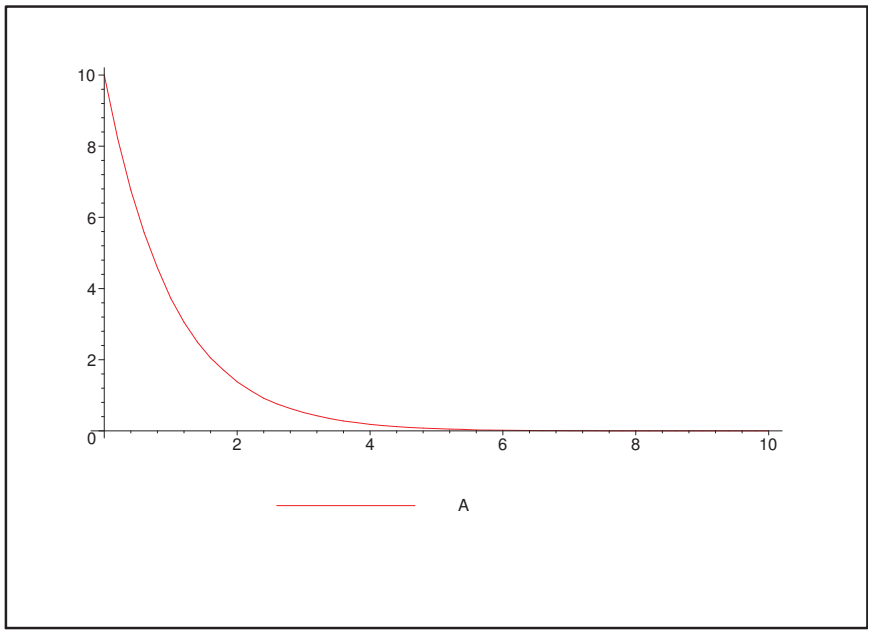


FIGURE 2.3 – Moyenne expérimentale de 1000 simulations stochastiques

nous amènent à poser $\Delta = kA(t_0)\Delta_t$.

On a donc :

$$\begin{aligned} A(t_0 + \Delta_t) &= A(t_0) - kA(t_0)\Delta_t \\ B(t_0 + \Delta_t) &= B(t_0) + kA(t_0)\Delta_t \end{aligned}$$

En simplifiant, et en utilisant le fait que $A'(t_0)$ (la dérivée de A en t_0) s'approxime par le taux de variation pendant Δ_t, on trouve

$$\begin{aligned} A'(t_0) \simeq \frac{A(t_0+\Delta_t)-A(t_0)}{\Delta_t} &= -kA(t_0) \\ B'(t_0) \simeq \frac{B(t_0+\Delta_t)-B(t_0)}{\Delta_t} &= +kA(t_0) \end{aligned}$$

Comme le raisonnement est vrai à tout instant t_0, on écrit alors :

$$\begin{aligned} A'(t) &= -kA(t) \\ B'(t) &= +kA(t) \end{aligned}$$

```
> edop1 := diff(A(t), t) = -k*A(t);
                                  d
                       edop1 := -- A(t) = -k A(t)
                                  dt
> edop2 := diff(B(t), t) = k*A(t);
                                  d
                       edop2 := -- B(t) = k A(t)
                                  dt
```

Intégration sous forme close. Le système étant très simple, on peut intégrer le système sous forme close, i.e. trouver une formule composée des fonctions élémentaires connues (exp, ln, ...).

```
> dsolve( {edop1, A(0)=A0}, A(t));
                                A(t) = A0 exp(-k t)
```

Cette intégration est en général impossible, ou inutile en pratique, à cause de la taille des expressions. On peut alors faire une étude numérique (en intégrant numériquement les systèmes d'équations en fixant des valeurs pour les paramètres) ou faire une analyse qualitative (qui permet d'analyser le système sans fixer les valeurs des paramètres).

Intégration numérique. On peut approximer la solution d'un système d'équations différentielles en procédant par petits pas, dits d'intégrations.

On a vu précédemment que $A(t_0 + \Delta_t) \simeq A(t_0) + A'(t_0)\Delta_t$. Comme la valeur de $A'(t_0)$ est donnée par le système d'équations, on peut approximer $A(t_0+\Delta_t)$ par $A(t_0)+A'(t_0)\Delta_t$.

Mais pour faire ce calcul, il faut fixer les valeurs de $A(0)$ et de k. En prenant $A(0) = 3$ et $k = 2$, pour un pas $\Delta_t = 0.01$, on trouverait $A(\Delta_t) \simeq A(0)+(-kA(0)\Delta_t) = 3-2\times3\times0.01 = 2.94$.

```
> edo1 := subs(k=2, edop1);
                                        d
                              edo1 := -- A(t) = -2 A(t)
                                        dt

> edo2 := subs(k=2, edop2);
                                        d
                              edo2 := -- B(t) = -2 B(t)
                                        dt

> soln := dsolve ({edo1, edo2, A(0)=3, B(0)=1}, {A(t), B(t)}, numeric);
                        soln := proc(x_rkf45)  ...  end proc
```

On peut alors tracer quelques courbes, montrant l'évolution de $A(t)$ et $B(t)$. On peut même tracer $A(t) + B(t)$ et se demander pourquoi $A(t) + B(t)$ semble égal à 4 (on écrit *semble* car une simulation numérique est toujours entachée d'erreur).

```
> with (plots):
> odeplot (soln, [t,A(t)], t = 0..5, view = [0..5, 0..4.1]);
> odeplot (soln, [t,B(t)], t = 0..5, view = [0..5, 0..4.1]);
> odeplot (soln, [t,A(t)+B(t)], t = 0..5, view = [0..5, 0..4.1]);
```

On voit que A tend vers zéro, et que B tend vers 4. Cela se comprend, A va être complétement épuisé, et se transformer complètement en B. Comme initialement il y a 3 A, et 1 B, il y a au final 4 B.

Analyse qualitative Nous allons montrer les conjectures que nous avons faites sur la simulation numérique. Pour montrer que $A(t) + B(t)$ vaut en permanence 4, on peut remarquer que la dérivée de $A(t) + B(t)$ vaut $A'(t) + B'(t) = -kA(t) + kA(t) = 0$. Par conséquent, $A(t) + B(t)$ est constant, et vaut $A(0) + B(0) = 4$. Cette première propriété sera appelée plus tard une loi de conservation.

Remarquons, et c'est là l'intérêt, que le raisonnement est valable quelles que soient les valeurs de $A(0)$, $B(0)$ et k, alors que la simulation numérique n'est valable que pour des valeurs fixées.

La force de l'analyse qualitative est qu'elle permet de trouver des propriétés indépendantes des valeurs numériques. Elle permet d'étudier des propriétés ou comportements particuliers en fonction des paramètres.

Une deuxième propriété que nous allons étudier est la présence de solutions stationnaires. Une solution stationnaire est une solution constante, i.e. une solution de la forme $A(t) = a_0, B(t) = b_0$, où a_0 et b_0 sont des constantes. Ces solutions se trouvent en injectant $A(t) = a_0, B(t) = b_0$ dans le système d'équations différentielles. Ici, nous obtenons :

$$\begin{aligned} 0 &= -ka_0 \\ 0 &= +ka_0 \end{aligned}$$

qui fournit $a_0 = 0$ (en supposant $k > 0$) et b_0 quelconque. On peut d'ailleurs montrer que toute solution du système d'équations différentielles va tendre vers une solution stationnaire $A(t) = 0, B(t) = b_0$, où $b_0 = A(0) + B(0)$.

Une bifurcation. Voici un exemple de ce qu'on appelle une bifurcation. L'exemple est académique, et montre qu'une petite variation sur les valeurs des paramètres peuvent avoir un grand effet sur les solutions. Considérons le système $A \xrightarrow{d} 0$, $A \xrightarrow{k} A + A$, qui pourrait être un exemple d'une levure qui meurt et se duplique.

L'équation différentielle associée est $A'(t) = (k - d)A(t)$.

Trois cas peuvent se produire :

— $k > d$: la levure prospère et sa quantité tend vers l'infini !

— $k = d$: (physiquement impossible), la quantité de levure est constante

— $k < d$: la levure meurt et sa quantité tend vers 0,

Ainsi, si les valeurs de k et d sont proches, on peut soit observer l'extinction de la levure, soit sa prolifération.

Dans la pratique, une forte sensibilité à des petites variations sur des paramètres ou des concentrations permet de créer des mécanismes sophistiqués dans les cellules : alertes quand un seuil est dépassé, comportements bistables, oscillants, ...

L'étude qualitative des systèmes d'équations différentielles est un domaine actif en recherche en calcul formel. En effet, les points de bifurcation sont solutions de systèmes d'équations et d'inéquations, et peuvent être traités efficacement lorsque les systèmes ne sont pas trop gros.

2.2 Modèle déterministe naturel

Ce modèle se construit d'après la loi d'action de masse, qui donne une approximation de la vitesse de chaque réaction. En notant X le vecteur de concentrations des espèces, M la matrice stœchiométrique, et V le vecteur des vitesses, on obtient le système suivant :

$$\frac{dX}{dt} = M.V$$

Il faut donc définir ce que sont M et V.

La matrice M a ses lignes indicées par les espèces, et ses colonnes indicées par les réactions. Le coefficient à la ligne ℓ et la colonne c est égal au nombre de molécules de ℓ produite par la réaction c (i.e. le nombre de ℓ dans la partie produits, moins le nombre de ℓ dans la partie réactants).

Le vecteur V est indicé par les réactions. La vitesse d'une réaction est égale à la constante de la réaction (le terme sur la flèche) fois le produit des espèces des produits (si une espèce apparaît deux fois dans les produits, elle sera au carré dans la vitesse).

L'exemple de Robertson illustre bien les (petites) difficultés d'écriture de ce système :

$$\begin{aligned} A &\xrightarrow{k_1} B \\ 2B &\xrightarrow{k_2} B + C \\ B + C &\xrightarrow{k_3} A + C \end{aligned}$$

On fixe le vecteur des espèces (l'ordre des espèces est arbitraire) :

$$X = \begin{pmatrix} A(t) \\ B(t) \\ C(t) \end{pmatrix}.$$

Le vecteur de vitesse (l'ordre des réactions est aussi arbitraire) :

$$V = \begin{pmatrix} k_1\, A(t) \\ k_2\, B(t)^2 \\ k_3\, B(t)\, C(t) \end{pmatrix}.$$

La matrice de stœchiométrique (qui sur l'exemple est une matrice carrée)

$$M = \begin{pmatrix} -1 & 0 & 1 \\ 1 & -1 & -1 \\ 0 & 1 & 0 \end{pmatrix}.$$

Le système développé est :

$$\begin{aligned} A'(t) &= -k_1 A(t) + k_3 B(t) C(t) \\ B'(t) &= k_1 A(t) - k_2 B(t)^2 - k_3 B(t) C(t) \\ C'(t) &= k_2 B(t)^2 \end{aligned}$$

2.3 Lois de conservations

Une loi de conservation est une combinaison linéaire d'espèces qui est constante au cours du temps. Sur l'exemple de Robertson de la section 2.2, on peut montrer que $A(t) + B(t) + C(t) = A(0) + B(0) + C(0)$ est une loi de conservation. On peut le faire à la main, en montrant que $A'(t) + B'(t) + C'(t) = 0$ (ce qui implique que $A(t) + B(t) + C(t)$ est constant). Une autre solution consiste à utiliser directement la matrice de stœchimétrie, en remarquant la chose suivante : la quantité

$$aA'(t) + bB'(t) + cC'(t),$$

où a, b, et c sont des constantes inconnues peut s'écrire comme le produit de vecteur ${}^tU.X'$, où U est le vecteur

$$U = \begin{pmatrix} a \\ b \\ c \end{pmatrix}.$$

Si l'on trouve un vecteur U tel que ${}^tU.X' = 0$, alors on a trouvé une loi de conservation.

Or ${}^tU.X' = {}^tU.M.V$. Si ${}^tU.M$ vaut le vecteur nul, on a trouvé une loi de conservation. Il suffit de remarquer que l'on peut résoudre aussi bien :

$${}^tM.U = 0.$$

Il s'agit donc de résoudre un système d'équations linéaires. C'est assez simple en Maple :

```
> with (LinearAlgebra):
> M := <<-1, 1, 0> | <0, -1, 1> | <1, -1, 0>>;
                                  [-1     0     1]
                                  [               ]
                             M := [ 1    -1    -1]
                                  [               ]
                                  [ 0     1     0]

> NullSpace (Transpose (M));
                                    [1]
                                    [ ]
                                   {[1]}
                                    [ ]
                                    [1]
```

La commande `NullSpace` renvoie une base des solutions. Chaque vecteur renvoyé par `NullSpace` va fournir une loi de conservation. Ici, une seule valeur est renvoyée, correspondant à la loi de conservation $A(t) + B(t) + C(t) = cste$.

Calcul à la main Une manière de procéder à la main consiste à faire un pivot de Gauss habituel sur la transposée de M, et de poursuivre en :

— mettant les pivots à 1 (en divisant la ligne par la valeur du pivot)

— mettant à 0 les coefficients qui sont dans une colonne contenant un pivot (en soustrayant comme il faut la ligne contenant le pivot aux autres lignes)

Sur notre exemple, en traitant la transposée de M, on obtient la matrice :

$$\begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{pmatrix}$$

Les colonnes ne contenant pas de pivot correspondent à des paramètres. Ici, les pivots sont a et b (colonnes 1 et 2), et il y a un paramètre c. Grâce à la forme spéciale de la matrice, on obtient des équations de la forme : pivot = combinaison linéaire de paramètres.

Sur notre exemple, la première ligne fournit $a - c = 0$, d'où $a = c$. La seconde ligne fournit $b - c = 0$, d'où $b = c$. Les deux pivots sont exprimés en fonction du paramètre c. En posant $c = 1$, on retrouve notre solution $a = b = c = 1$.

On peut obtenir autant de lois de conservations que de paramètres. On obtient successivement chaque loi en posant successivement chaque paramètre à 1 et les autres à 0.

2.4 Illustration des notions avec le paquetage MABSys

Le paquetage MABSys [4] a été écrit par AslıÜrgüplü et François Lemaire, dans le langage Maple. Le titre signifie *Modeling and Analysis of Biological Systems*. L'objectif du paquetage est de fournir une série d'outils d'études de systèmes biologiques (par l'approche des réactions chimiques) facilement utilisables par les non initiés. Les fonctions cachent des techniques complexes comme l'algèbre linéaire, l'élimination différentielle, la recherche de symétries, qui sont des domaines privilégiés du Calcul Formel.

Création des réactions et du système

```
> R1 := NewReaction(E+S,C,MassActionLaw(k1),fast=true);
             R1 := Reaction([E, S], [C], MassActionLaw(k1), true)

> R2 := NewReaction(C,E+S,MassActionLaw(km1),fast=true);
             R2 := Reaction([C], [E, S], MassActionLaw(km1), true)

> R3 := NewReaction(C,E+P,MassActionLaw(k2));
             R3 := Reaction([C], [E, P], MassActionLaw(k2), false)

> RS := [R1,R2,R3]:
```

Dynamique

```
> RateVector(RS);
                                  [k1 E S]
                                  [      ]
                                  [km1 C ]
```

```
                                  [      ]
                                  [ k2 C ]

> StoichiometricMatrix(RS, [E,S,C,P]);
                               [-1     1     1]
                               [               ]
                               [-1     1     0]
                               [               ]
                               [ 1    -1    -1]
                               [               ]
                               [ 0     0     1]

> ReactionSystem2ODEs(RS, [E,S,C,P]);
 d
[-- E(t) = -k1 E(t) S(t) + km1 C(t) + k2 C(t),
 dt

    d
    -- S(t) = -k1 E(t) S(t) + km1 C(t),
    dt

    d                                              d
    -- C(t) = k1 E(t) S(t) - km1 C(t) - k2 C(t), -- P(t) = k2 C(t)]
    dt                                             dt

> ConservationLaws(RS);
               [E + C - E_0 - C_0, -E + S + P + E_0 - S_0 - P_0]

> Equilibria(RS);
                        [-k1 E S + km1 C + k2 C, -k2 C]
```

Tracés

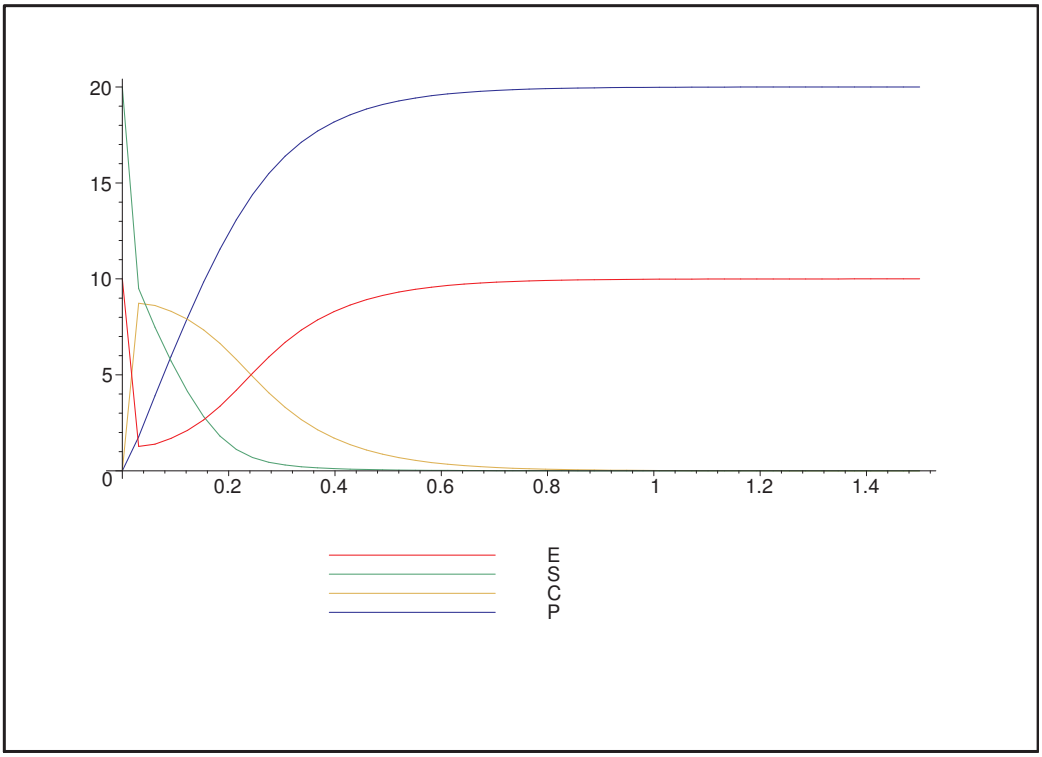
```
> sol := NumericalSolution(RS,
>                          ics = { C(0)=0, P(0)=0, E(0)=10, S(0)=20},
>                          params = { k1=10, km1=5, k2=8} );
sol := [proc(x_rkf45)  ...  end proc,

    {C(0) = 0, P(0) = 0, E(0) = 10, S(0) = 20}, {k1 = 10, km1 = 5, k2 = 8},

    [E, S, C, P]]

> plotsetup(X11); # Pour tracer depuis le mode texte

> PlotSolution(sol, t=0..1.5);
```


20
15
10
5
0
0.2
0.4
0.6
0.8
1
1.2
1.4
E
S
C
P

Chapitre 3

Analyse de points fixes

Nous avons vu qu'une solution d'un système d'équations différentielles admettait parfois une limite quand t tend vers l'infini (voir l'exemple $A \xrightarrow{k} B$). Lorsqu'on étudie un modèle, la limite des solutions est l'une des premières choses à étudier pour plusieurs raisons : cela fournit des informations sur la stabilisation éventuelle du système, et c'est une question plus simple que d'étudier la solution à tout instant.

Si une solution admet une limite, alors les derivées des concentrations tendent vers 0. Ainsi, toute limite de solutions est solution du système d'équations différentielles dans lequel on a remplacé les dérivées par 0.

Définition 1 (Points fixes) *Soit un système d'équations différentielles de la forme $X'(t) = F(K, X(t))$, où K est un vecteur de valeurs paramètres. On appelle point fixe du système, tout vecteur constant x_0 tel que $F(K, x_0)$ vaut 0.*

La définition montre bien que le point fixe dépend des paramètres. Suivant les valeurs des paramètres, il pourra exister aucun, un ou plusieurs points fixes.

3.1 Cas de la dégradation enzymatique

Si l'on reprend l'exemple de Michaelis-Menten, on a le système d'équations :

$$\begin{aligned} E'(t) &= -k_1 E(t)S(t) + k_{-1}C(t) + k_2 C(t) \\ S'(t) &= -k_1 E(t)S(t) + k_{-1}C(t) \\ C'(t) &= k_1 E(t)S(t) - k_{-1}C(t) - k_2 C(t) \\ P'(t) &= k_2 C(t) \end{aligned}$$

Chercher les points fixes, c'est donc chercher une solution constante $(E(t), S(t), C(t), P(t)) = (e_0, s_0, c_0, p_0)$. En reportant la solution constante dans le système d'équation, on trouve :

$$\begin{aligned} 0 &= -k_1 e_0 s_0 + k_{-1} c_0 + k_2 c_0 \\ 0 &= -k_1 e_0 s_0 + k_{-1} c_0 \\ 0 &= k_1 e_0 s_0 - k_{-1} c_0 - k_2 c_0 \\ 0 &= k_2 c_0 \end{aligned}$$

On obtient un système non linéaire, où les inconnues sont (e_0, s_0, c_0, p_0) et où les paramètres sont (k_1, k_{-1}, k_2).

Ici, le système peut se résoudre facilement :

— de $k_2 c_0 = 0$, on déduit que $c_0 = 0$

— comme $c_0 = 0$, les trois premières équations se résument à $k_1 e_0 s_0 = 0$, ce qui implique $e_0 = 0$ ou $s_0 = 0$

— pour l'instant, p_0 est quelconque

Il reste une indétermination, car on ne sait pas si c'est e_0 ou s_0 qui est nul, et on ne connaît pas p_0. Peut-on lever ce doute ? La réponse est oui, il faut penser à utiliser les lois de conservations. En effet, ces lois sont valables à tout instant t, et également à la limite. On a donc :

$$\begin{aligned} e_0 + c_0 &= E(0) + C(0) \\ -e_0 + s_0 + p_0 &= -E(0) + S(0) + P(0) \end{aligned}$$

Comme $c_0 = 0$, on a donc $e_0 = E(0) + C(0)$. Si on suppose que l'on a $E(0) + C(0) > 0$, alors c'est donc s_0 qui est nul. En reportant dans la deuxième loi de conservation, on trouve

$$-(E(0) + C(0)) + 0 + p_0 = -E(0) + S(0) + P(0),$$

ce qui donne

$$p_0 = C(0) + P(0) + S(0).$$

Pour résumer, on a donc :

$$\begin{aligned} e_0 &= E(0) + C(0) \\ s_0 &= 0 \\ c_0 &= 0 \\ p_0 &= C(0) + P(0) + S(0) \end{aligned}$$

3.2 Résolution automatisée

La résolution à la main n'était pas si difficile, mais peut se compliquer sur des exemples plus gros. Voyons comment on pourrait traiter l'exemple avec des outils symboliques comme le paquetage `RegularChains` (en standard dans Maple), dont la première version a été écrite par François Lemaire et Marc Moreno Maza.

Création du modèle

```
> with(MABSys):
> R1 := NewReaction(E+S,C,MassActionLaw(k1),fast=true):
> R2 := NewReaction(C,E+S,MassActionLaw(km1),fast=true):
> R3 := NewReaction(C,E+P,MassActionLaw(k2)):
> RS := [R1,R2,R3]:
```

Points fixes et lois de conservation. Les conditions initiales $E(0)$, $S(0)$, ... sont notées E_0, S_0,... par MABSys.

```
> eq := Equilibria(RS);
                      eq := [k1 E S - km1 C - k2 C, k2 C]

> cl := ConservationLaws(RS);
            cl := [C + P + S - C_0 - P_0 - S_0, C + E - C_0 - E_0]

> sys := [ op(eq), op(cl) ];
sys := [k1 E S - km1 C - k2 C, k2 C, C + P + S - C_0 - P_0 - S_0,

    C + E - C_0 - E_0]
```

Chargement du paquetage RegularChains.

```
> with(RegularChains);
[ChainTools, ConstructibleSetTools, DisplayPolynomialRing, Equations,

    ExtendedRegularGcd, FastArithmeticTools, Inequations, Info, Initial,

    Inverse, IsRegular, MainDegree, MainVariable, MatrixCombine, MatrixTools,

    NormalForm, ParametricSystemTools, PolynomialRing, Rank, RegularGcd,

    RegularizeInitial, SemiAlgebraicSetTools, Separant, SparsePseudoRemainder,

    Tail, Triangularize]
```

Création de l'anneau de polynômes

```
> R := PolynomialRing([E,S,C,P,E_0,S_0,C_0,P_0,k1,km1,k2] );
                              R := polynomial_ring
```

Mise sous forme triangulaire

```
> rcs := Triangularize(sys, R);
      rcs := [regular_chain, regular_chain, regular_chain, regular_chain]
```

On obtient quatre systèmes d'équations, appelés chaînes régulières. Le premier cas est le cas général. Les trois suivants sont des cas particuliers, obtenus par des scindages. Un scindage se produit lorsqu'un produit de deux termes est nul, et qu'on traite chaque cas indépendamment. Les trois cas particuliers correspondent respectivement à $C_0 + E_0 = 0$, $k_1 = 0$ et $k_2 = 0$.

```
> Equations(rcs[1],R);
                  [E - C_0 - E_0, S, C, P - C_0 - P_0 - S_0]

> Equations(rcs[2],R);
                  [E, S + P - C_0 - P_0 - S_0, C, C_0 + E_0]

> Equations(rcs[3],R);
                [E - C_0 - E_0, S + P - C_0 - P_0 - S_0, C, k1]

> Equations(rcs[4],R);
                                                          2
[C + E - C_0 - E_0, C + P + S - C_0 - P_0 - S_0, k1 C

     + (-km1 + k1 P - 2 k1 C_0 - k1 P_0 - k1 S_0 - E_0 k1) C

                                                                    2
     + (-k1 C_0 - E_0 k1) P + (k1 C_0 + k1 P_0 + k1 S_0) E_0 + k1 C_0

     + C_0 k1 P_0 + C_0 k1 S_0, k2]
```

On peut supprimer les cas particuliers en ajoutant des inéquations en deuxième paramètre. On peut ainsi supposer que $E(0)$ et les paramètres k_1, k_{-1} et k_2 sont non nuls. On remarque que l'on n'a pas pu écarté le cas particulier où $C(0) + E(0) = 0$. La raison est que `Triangularize` ne gère pas les inégalités.

```
> rcs := Triangularize(sys, [E_0,k1,km1,k2], R);
                     rcs := [regular_chain, regular_chain]

> Equations(rcs[1],R);
                  [E - C_0 - E_0, S, C, P - C_0 - P_0 - S_0]

> Equations(rcs[2],R);
                  [E, S + P - C_0 - P_0 - S_0, C, C_0 + E_0]
```

Chapitre 4

Résolution numérique des points fixes

On montre quelques outils mathématiques et informatiques afin d'étudier numériquement les points fixes des systèmes d'équations différentielles. Par souci de simplicité (les modèles biologiques étant un peu trop complexes), on considère dans ce chapitre une version "améliorée" du système proie-prédateur de Lotka Volterra (décrit dans `http://www.dma.ens.fr/culturemath/maths/ps/analyse/interactions.ps`) :

$$\dot{u} = u(1-u) - auv/(u+d) \tag{4.1a}$$
$$\dot{v} = bv(1-v/u) \tag{4.1b}$$

- u est la proie, v le prédateur.
- a, b et d sont des paramètres du modèle
- $u(1-u)$ est le terme de croissance
- $auv/(u+d)$ est la prédation de u. Ce terme est proportionnel à v. Il tend vers av quand u tend vers l'infini, contrairement au terme de prédation classique auv. L'idée est que les prédateurs sont rassasiés !
- $bv(1-v/u)$ est le terme de croissance de v. Il s'inverse quand v atteint la même valeur que u.

Remarque : le système (4.1) a été obtenu par réduction (translation et dilatation) d'un autre modèle. Cela explique que les solutions u et v vérifient $0 \leq u(t) \leq 1$ et $0 \leq v(t) \leq 1$.

4.1 Définition du système

```
> sys_t := [
>     diff(u(t),t) = u(t)*(1-u(t)) - a*u(t)*v(t)/( u(t)+d ),
>     diff(v(t),t) = b*v(t) * (1 - v(t)/u(t))
>        ] :
> sys := map ( rhs, sys_t):
> sys := subs ( {u(t)=u, v(t)=v}, sys):
> sys := map(numer, sys);
                                    2
             sys := [-u (-u - d + u  + u d + a v), b v (u - v)]
```

4.2 Mise sous forme triangulaire

On va utiliser le paquetage `RegularChains` (disponible en Maple) qui permet de transformer un ensemble de polynômes (ici `sys`) en une ou plusieurs chaînes régulières. Une chaîne régulière est un ensemble triangulaire disposant de propriétés supplémentaires. Pour un ordre fixé sur les inconnues, on dit qu'un ensemble d'équations est triangulaire si chaque équation a une variable principale différente (la variable principale d'une équation est la plus grande variable dont dépend l'équation).

```
> X := [u,v,a,b,d];
                              X := [u, v, a, b, d]

> R := PolynomialRing(X);
                              R := polynomial_ring
```

On a défini l'ordre $u > v > a > b > d$ grâce à la variable `X`, et on a défini l'anneau de polynôme `R`.

```
> out := Triangularize(sys, R);
bytes used=12011016, alloc=1113908, time=0.27
out := [

    regular_chain, regular_chain, regular_chain, regular_chain, regular_chain]

> map(Equations, out, R);
          2                                    2
[[u - v, v  + (a - 1 + d) v - d], [u, v], [u  + (-1 + d) u - d, v], [u, b],

      2
    [u  + (-1 + d) u + a v - d, b]]
```

Le système d'entrée `sys` a été décomposé en cinq chaînes régulières. Seule la première de ces chaînes est intéressante :

$$\begin{aligned} u - v &= 0 \\ v^2 + (a - 1 + d)v - d &= 0 \end{aligned}$$

L'équation $u - v$ (resp. $v^2 + (a - 1 + d)v - d$) admet u (resp. v) comme variable principale. Si l'on veut éviter certaines entrées superflues, on peut indiquer une liste d'inéquations.

```
> Ineqs := [u,v,a,b,d];
                            Ineqs := [u, v, a, b, d]

> out := Triangularize(sys, Ineqs, R);
                             out := [regular_chain]

> map(Equations, out, R);
```

```
                                  2
                        [[u - v, v  + (a - 1 + d) v - d]]
```

```
> Equi := Equations(out[1],R);
                                       2
                     Equi := [u - v, v  + (a - 1 + d) v - d]
```

On montre facilement que ce système a une infinité de solutions réelles. Toutefois, si on fixe les valeurs des paramètres a et d, le système a au plus deux solutions réelles positives.

4.3 Calcul des équilibres pour des valeurs de paramètres données

Si on fixe les valeurs de a, b et d, l'équilibre se calcule numériquement.

```
> sys0 := subs( {a=1, b=2, d=4}, sys);
                                            2
                  sys0 := [-u (3 u - 4 + u  + v), 2 v (u - v)]
```

```
> solve(sys0);
{v = 0, u = 0}, {v = 0, u = -4}, {v = 0, u = 1}, {v = 0, u = 0}, {

                                 2
    u = 2 RootOf(2 _Z - 1 + _Z , label = _L2),

                                 2
    v = 2 RootOf(2 _Z - 1 + _Z , label = _L2)}
```

```
> evalf(%);
{v = 0., u = 0.}, {v = 0., u = -4.}, {v = 0., u = 1.}, {v = 0., u = 0.},

    {u = 0.8284271248, v = 0.8284271248}
```

La fonction `solve` de Maple est numérique ; elle est rapide mais ce n'est pas une méthode garantie, et elle risque donc de rater des solutions et d'être imprécise sur certains exemples.

On peut aussi utiliser les chaînes régulières pour trouver les solutions. Dans ce cas, on a un résultat garanti :

- toutes les solutions sont obtenues ;
- chaque solution est fournie par un encadrement (un produit cartésien d'intervalles) raffinable à volonté.

La fonction `RealRooIsolate` (disponible à partir de Maple 13) prend en entrée une chaîne régulière qui doit être :

- sans carré (il faut passer l'option `radical=yes` à `Triangularize`)
- de dimension 0 (autant d'équations que d'inconnues)

La fonction `RealRootIsolate` renvoie une liste de boites, chaque boîte étant une liste d'intervalles pour chaque inconnue. Chaque racine du système initiale est isolée dans une et une seule boite, que l'on peut rafiner à volonté.

```
> Ineqs0 := [u,v];
                                Ineqs0 := [u, v]

> R0 := PolynomialRing([u,v]);
                               R0 := polynomial_ring

> out := Triangularize(sys0, Ineqs0, R0, radical=yes);
                              out := [regular_chain]

> rc0 := out[1]:
>
> with(SemiAlgebraicSetTools);
[BoxValues, DisplayParametricBox, DisplayQuantifierFreeFormula,

    IsParametricBox, PartialCylindricalAlgebraicDecomposition,

    RealRootCounting, RealRootIsolate, RefineBox, RefineListBox,

    RepresentingBox, RepresentingChain, RepresentingQuantifierFreeFormula,

    RepresentingRootIndex, VariableOrdering]

>
> sols := RealRootIsolate(rc0, R0);
                                sols := [box, box]

> map(BoxValues, sols);
            [[u = [-1, 6], v = [0, 5]], [u = [-6, 1], v = [-5, 0]]]
```

Le raffinage de solution se fait avec `RefineListBox` pour raffiner une liste de boite, ou simplement avec `RefineBox` pour raffiner une boite.

```
> refined_sols := RefineListBox(sols, 1/2^10, R0);
bytes used=20015504, alloc=5176396, time=0.41
                           refined_sols := [box, box]

> map(BoxValues, refined_sols);
       -39555  -9887        -5062973  -1265743
[[u = [------, -----], v = [--------, --------]],
        8192   2048         1048576    262144

          6781  1697        1737337  868669
    [u = [----, ----], v = [-------, -------]]]
          8192  2048        2097152  1048576

> evalf(%);
[[u = [-4.828491211, -4.827636719], v = [-4.828427315, -4.828426361]],

    [u = [0.8277587891, 0.8286132812], v = [0.8284268379, 0.8284273148]]]
```

On remarque ici qu'on trouve une solution $u = -4.82.., v = -4.82$ que la fonction `solve` n'avait pas trouvé !

On peut également limiter une zone dans laquelle on recherche les solutions, en passant un paramètre supplémentaire à `RealRootIsolate`. Ici on sait que u et v sont tous deux compris entre 0 et 1.

```
> sols := RealRootIsolate(rc0, R0,
>                         constraints=
>                         [u=[0,1], v=[0,1]]);
                                sols := [box]

> refined_sols := RefineListBox(sols, 1/2^10, R0);
                              refined_sols := [box]

> map(BoxValues, refined_sols);
                           53  849         54291  13573
                    [[u = [--, ----], v = [-----, -----]]]
                           64  1024        65536  16384

> evalf(%);
    [[u = [0.8281250000, 0.8291015625], v = [0.8284149170, 0.8284301758]]]
```

4.4 Isolation de racines réelles

Cas d'un polynôme en une variable On peut isoler les racines facilement si l'on dispose des deux fonctions suivantes :

- une fonction `borneNombreDeRacines`$(p, [a, b])$ qui renvoie un majorant du nombre de racines de p dans l'intervalle $[a, b]$. Ce majorant est exact lorsque la fonction vaut 0 ou 1. Il faut que p n'ait que des racines simples ;
- une fonction `borneRacines`(p) qui renvoie une borne B telle que p a toutes ses racines contenues dans l'intervalle $]-B, B[$.

L'idée est d'appliquer un algorithme de type diviser-pour-régner sur l'intervalle $]-B, B[$ et d'arrêter chaque branche quand la fonction `borneNombreDeRacines` renvoie 0 ou 1. Pour une telle branche, si il n'y a pas de racines, on laisse de côté l'intervalle, sinon l'intervalle obtenu est un encadrement (vu qu'il n'y a qu'une racine). Pour supprimer les racines multiples de p, il suffit de remplacer p par $p/gcd(p, p')$ au départ.

En Maple, la fonction `realroot(p)` permet d'isoler les racines d'un polynôme quelconque p. On peut passer en deuxième paramètre la précision afin que la largeur de chaque intervalle soit inférieure à cette précision.

```
> realroot(x^2-2);
                              [[0, 2], [-2, 0]]

> realroot(x^2-2, 1e-5);
                       185363  46341    -46341  -185363
                     [[------, -----], [------, -------]]
                       131072  32768    32768   131072

> evalf(%);
          [[1.414207458, 1.414215088], [-1.414215088, -1.414207458]]
```

Cas d'un ensemble triangulaire On peut aussi calculer les solutions d'une chaîne régulière qui a un nombre fini de solutions. Pour cela il faut qu'elle contienne autant d'équations que d'inconnues (et qu'il n'y ait pas de racines multiples, voir l'option `radical=yes` déjà évoquée).

On généralise simplement la méthode pour un polynôme en une variable, en suivant le principe suivant :

— résoudre le polynôme en une variable ;
— pour chaque solution, reporter dans le polynôme en deux variables pour obtenir un polynôme en une variable ;
— procéder ainsi pour toutes les équations.

Chapitre 5

Théorème de déficience zéro

Ce chapitre présente un théorème historique assurant l'existence, l'unicité et la stabilité d'un équilibre dans un réseau de réactions chimiques. Ce théorème qui date des années 70 est dû à Martin Feinberg [2, 3].

Théorème 1 (Théorème de déficience zéro (simplifié)) *Soit un réseau de réactions chimiques faiblement réversible et de déficience zéro. On suppose aussi que les vitesses de réactions suivent la loi d'action de masse. Il existe un unique équilibre pour chaque classe de compatibilité. De plus, cet équilibre est globalement attractif, et la concentration de chaque espèce est non nulle.*

Remarque 1 *Un équilibre est dit attractif toute solution dont la condition initiale est proche de l'équilibre tend vers l'équilibre. Il est dit globalement attractif quand les solutions tendent vers l'équilibre quelles que soient les conditions initiales.*

Exemple 1 *Le système suivant satisfait le théorème 1 :*

$A \underset{k_b}{\overset{k_a}{\rightleftharpoons}} 2B$

$A + C \underset{k_d}{\overset{k_{ac}}{\rightleftharpoons}} D$

$B + E \xrightarrow{k_{be}} A + C$

$D \xrightarrow{k'_d} B + E$

Exemple 2 *Le système suivant satisfait le théorème 1 :* $A + B \underset{k_2}{\overset{k_1}{\rightleftharpoons}} C$

Exemple 3 *Le système suivant ne satisfait le théorème 1 :* $A + B \xrightarrow{k_1} C$

Définition 2 (Réseau de réactions chimiques) *On appelle réseau de réactions chimiques un triplet* $(\mathcal{A}, \mathcal{C}, \mathcal{R})$ *où*

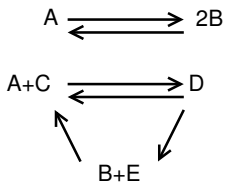
— $\mathcal{A}$ *est l'ensemble des espèces ;*

— $\mathcal{C}$ *est l'ensemble des complexes, un complexe étant une combinaison linéaire d'espèces à coefficients entiers ;*
— $\mathcal{R}$ *est l'ensemble des réactions, une réaction étant de la forme* $C_i \to C_j$ *où* C_i *et* C_j *sont deux complexes*

Exemple 4 (suite de l'exemple 1) $\mathcal{A} = \{A, B, C, D, E\}$, $\mathcal{C} = \{A, 2B, A{+}C, D, B{+}E\}$, $\mathcal{R} = \{A \to 2B, 2B \to A, A + C \to D, D \to A + C, B + E \to A + C, D \to B + E\}$.

Définition 3 (Graphe associé à un réseau de réactions chimiques) *On appelle graphe associé à un réseau de réactions chimiques le graphe orienté tel que :*
— *les sommets sont les complexes (un complexe ne doit apparaître qu'une seule fois dans le graphe même s'il apparaît dans plusieurs réactions) ;*
— *les arcs sont définis par les réactions i.e. un arc va de* C_i *à* C_j *si et seulement si la réaction* $C_i \to C_j$ *existe.*

Exemple 5 (suite de l'exemple 1) *Le graphe associé au réseau est :*



Définition 4 (Réseau de réactions chimiques faiblement réversible) *Se dit d'un réseau qui vérifie l'hypothèse suivante : pour tous complexes* C_i *et* C_j*, si il existe un chemin orienté allant de* C_i *à* C_j*, alors il existe un chemin orienté de* C_j *à* C_i*.*

Remarque 2 *Le chemin allant de* C_j *à* C_i *n'est pas nécessairement l'"inverse" du chemin allant de* C_i *à* C_j*.*

Remarque 3 *En particulier, un réseau composé uniquement de réactions réversibles est faiblement réversible.*

Exemple 6 *Les exemples 1 et 2 sont faiblement réversibles. L'exemple 3 n'est pas faiblement réversible.*

Définition 5 (Déficience d'un réseau de réactions chimiques) *On appelle déficience d'un réseau de réactions chimiques, souvent notée* δ*, la valeur* $c - \ell - \operatorname{rank} S$*, où :*
— c *est le nombre de complexes ;*
— ℓ *est de nombre de classes de liaison (*linkage classes*) du réseau, qui par définition est le nombre de composantes connexes du graphe du réseau ;*
— $\operatorname{rank} S$ *est le rang de la matrice de stœchiométrie du réseau*

Remarque 4 *Le rang d'une matrice peut se calculer en comptant le nombre de lignes non nulles à la fin du pivot de Gauss.*

Exemple 7 (suite de l'exemple 1) *Il y a $c = 5$ complexes et il y a $\ell = 2$ classes de liaison. La matrice de stœchiométrie est :*

$$\begin{pmatrix} -1 & 1 & -1 & 1 & 1 & 0 \\ 2 & -2 & 0 & 0 & -1 & 1 \\ 0 & 0 & -1 & 1 & 1 & 0 \\ 0 & 0 & 1 & -1 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

Ici le rang vaut 3, donc la déficience vaut 0.

Définition 6 (Classe de compatibilité d'un réseau de réactions chimiques) *On dit que deux vecteurs de concentrations X_1 et X_2 à coordonnées toutes positives sont dans la même classe de compatibilité si le vecteur $X_2 - X_1$ est une combinaison linéaire colonnes de la matrice de stœchiométrie. Une classe de compatibilité est donc un espace affine, intersecté avec l'espace des concentrations strictement positives.*

Remarque 5 *La notion de classe de compatibilité est complémentaire avec celle de loi de conservation. Lorsque le réseau n'a aucune loi de conservation, il n'y a qu'une seule classe de compatibilité (composé de toutes les valeurs possibles). Si des lois de conservations existent*[1]*, alors les solutions vivent sur une sorte de feuilletage, chaque feuille constituant une classe de compatibilité.*

Remarque 6 *On peut aussi dire que deux vecteurs de concentrations X_1 et X_2 à coordonnées toutes positives sont dans la même classe si ils satisfont les mêmes lois de conservations (à condition de ne prendre que celles obtenues avec la transposée de la matrice de stœchimétrie).*

Exemple 8 (suite de l'exemple 1) *Le système admet deux lois de conservation : $2A + B - C + D = cste$ et $C + D + E = cste$. Ainsi les conditions initiales $X_1 = (1, 7, 4, 3, 5)$ et $X_2 = (2, 5, 5, 4, 3)$ fournissent deux solutions avec le même point d'équilibre car on a $2 * 1 + 7 - 4 + 3 = 8 = 2 * 2 + 5 - 5 + 4$ et $4 + 3 + 5 = 12 = 5 + 4 + 3$.*

On peut aussi montrer que $X_2 - X_1 = (1, -2, 1, 1, -2)$ est une combinaison linéaire des colonnes de la matrice de stœchimétrie.

1. uniquement celle obtenu avec la transposée de la matrice de stœchimétrie

Chapitre 6

Réduction des systèmes d'équations

Les réductions consistent à approximer le modèle initial en un modèle plus simple :
— moins de variables
— moins de paramètres
— intégration plus facile
— étude de stabilité plus simple
— ...

L'idée est donc de simplifier le modèle initial, sans en changer le comportement ni les propriétés. Pour ce faire, il y a de nombreuses méthodes à classer en au moins deux rubriques : les exactes et les approchées. Dans ce qui suit, nous en présentons quelques unes.

6.1 Réductions exactes

Par réduction exacte, on entend simplification du système qui ne change pas le comportement du système. Ainsi l'étude du système réduit est strictement équivalente à l'étude du système initial. Pour ce faire, on utilise des méthodes basées sur les symétries, que l'on abordera que de manière intuitive. On va voir deux types de symétries élémentaires, basées sur les translations et les dilatations. Ces symétries permettent de réduire le nombre de paramètres. Le paquetage `ExpandedLiePointSymmetry` (voir la rubrique software de `http://www2.lifl.fr/~sedoglav/`) de A. Sedoglavic et A. Ürgüplü permet de faire ce type de réductions. On se base sur le modèle de Verhulst avec prédation linéaire :

$$\dot{x} = x(a - bx) - cx \tag{6.1}$$

— x : le nombre d'individus ;
— a : son taux de croissance ;
— b : la capacité d'accueil (si x augmente, le facteur $a - bx$ finit par limiter l'augmentation de x)
— c : le taux de prédation de x

6.1.1 Les translations

On peut réécrire l'équation en $\dot{x} = (a-c)x - bx^2$ On voit que l'équation ne dépend que de $a - c$: ainsi $a - c$ peut être remplacé par un nouveau paramètre a_2. Cela ne se voit pas directement sur l'équation initiale, on a donc besoin d'une méthode algorithmique.

Translation laissant invariant le système. Considérons la transformation ϕ_1 :

$$a \to a + \lambda \tag{6.2a}$$
$$b \to b \tag{6.2b}$$
$$c \to c + \lambda \tag{6.2c}$$
$$x \to x \tag{6.2d}$$

qui est une translation (au même sens que les translations en géométrie).

Si on applique la transformation ϕ_1 à (6.1), on obtient :

$$\dot{x} = x(a + \lambda - bx) - (c + \lambda)x = x(a - bx) - cx$$

Cela signifie que pour tout λ, la transformation ϕ_1 laisse "invariante" l'équation de Verhulst.

Intérêt de la translation L'intérêt est le suivant : imaginons qu'on doive étudier (6.1) pour des valeurs $a = a^0$, $b = b^0$, $c = c^0$. Étudier le cas $a = a^0$, $b = b^0$, $c = c^0$ est identique à étudier le cas $a = a^0 + \lambda$, $b = b^0$, $c = c^0 + \lambda$ pour tout λ.

On peut choisir λ intelligement, par exemple en prenant $\lambda = -c^0$. Étudier le cas $a = a^0$, $b = b^0$, $c = c^0$ revient à étudier le cas $a = a^0 - c^0$, $b = b^0$, et $c = 0$. Ainsi, en utilisant ϕ_1, si on sait étudier le système (6.1) dans le cas particulier où $c = 0$, on sait aussi étudier (6.1) pour c quelconque. On dit que la transformation ϕ_1 a permis de réduire le paramètre c.

On a ainsi réduit le modèle et (6.1) est équivalente à

$$\dot{x} = x(a - bx) \tag{6.3}$$

Détermination des translations Voici pour information comment trouver les translations. Les calculs relèvent de l'algèbre linéaire.

— on remplace dans le système initial chaque inconnue v par $v + n_v\lambda$, où les n_v et λ sont supposés indépendants du temps

$$\dot{x} = (a - c + n_a\lambda - n_c\lambda)(x + n_x\lambda) - (b + n_b\lambda)(x + n_x\lambda)^2$$

— on développe en éliminant les termes en λ^2, λ^3, ...

$$\dot{x} = (a - c)x + (a - c)n_x\lambda + (n_a - n_c)x\lambda - (bx^2 + 2bn_x x\lambda + n_b x^2\lambda)$$

— on identifie le système obtenu avec le système initial

$$n_x a - n_x c + (n_a - n_c)x - 2n_x bx - n_b x^2 = 0$$

— ce qui fournit un système linéaire d'équations en les n_v, sachant que le système doit être vrai pour toutes valeurs des variables et des paramètres. Pour ce faire, on fait apparaître les monômes en les variables et paramètres.

$$n_x a - n_x c + (n_a - n_c)x - 2n_x bx - n_b x^2$$

$$\begin{cases} n_x &= 0 \\ -n_x &= 0 \\ n_a - n_c &= 0 \\ -2n_x &= 0 \\ n_b &= 0 \end{cases}$$

— on résout ce système pour obtenir les valeurs des n_v : $n_x = n_b = 0$ et $n_a = n_c$.
— on choisit une solution non nulle (quelconque). Par exemple $n_x = n_b = 0$ et $n_a = n_c = 1$ permet de retrouver la transformation ϕ_1
— recommencer jusqu'à ce qu'il n'y ait plus de translation possible (c'est à dire quand le système linéaire en les n_v n'admet plus que 0 comme solution).

En Maple, on peut automatiser le processus avec :

```
verh := diff(x(t),t) - (x(t)*(a-b*x(t)) - c *x(t));

phi1 := { a=a+na*l,
          b=b+nb*l,
          c=c+nc*l,
          x(t)=x(t)+nx*l };

verhphi1 := subs(phi1, verh);
verhphi1 := expand(verhphi1);
verhphi1 := algsubs(l^2=0, verhphi1);

cond := verh - verhphi1;
cond := subs(x(t)=x, cond);
cond := expand(cond);
syst := [ coeffs(cond, [x,a,b,c,l]) ];

solve(syst);
```

6.1.2 Les dilatations

La méthode est similaire concernant les dilatations.

Dilatation laissant invariant le système. Considérons la transformation ϕ_2 :

$$t \to t \tag{6.4a}$$
$$a \to a \tag{6.4b}$$
$$b \to \lambda b \tag{6.4c}$$
$$x \to \frac{x}{\lambda} \tag{6.4d}$$

qui est une dilatation (ce qu'on appelle aussi homothétie en géométrie).

Si on applique la transformation ϕ_2 à (6.3), on obtient :

$$\frac{\dot{x}}{\lambda} = \frac{x}{\lambda}\left(a - \lambda b \frac{x}{\lambda}\right)$$

ce qui donne (en multipliant par λ à gauche et à droite)

$$\dot{x} = x(a - bx)$$

Cela signifie que pour tout λ, la transformation ϕ_2 laisse "invariante" (6.3).

Intérêt de la dilatation Imaginons qu'on doive étudier (6.3) pour des valeurs $a = a^0$, $b = b^0$. Étudier le cas $a = a^0$, $b = b^0$ est identique à étudier le cas $a = a^0$, $b = \lambda b^0$, pour tout λ.

On peut choisir $\lambda = 1/b^0$. Étudier le cas $a = a^0$, $b = b^0$ revient à étudier le cas $a = a^0$, $b = 1$. Ainsi, en utilisant ϕ_2, si on sait étudier le système (6.3) dans le cas particulier où $b = 1$, on sait aussi étudier (6.3) pour a et b quelconques. La dilatation a permis de réduire le paramètre b.

On a ainsi réduit le modèle et (6.3) est équivalente à

$$\dot{x} = x(a - x) \tag{6.5}$$

Une autre dilatation On peut encore réduire l'équation et faire disparaître le dernier paramètre a. Pour ce faire, utilisons la transformation ϕ_3 :

$$t \to \frac{t}{\lambda} \tag{6.6a}$$
$$a \to \lambda a \tag{6.6b}$$
$$x \to \lambda x \tag{6.6c}$$

En appliquant ϕ_3 à (6.5), et écrivant $\dot{x} = \frac{dt}{dt}$, on obtient :

$$\frac{\lambda dx}{\frac{dt}{\lambda}} = \lambda x(\lambda a - \lambda x)$$

qui se simplifie en

$$\frac{dx}{dt} = \dot{x} = x(a - x)$$

En raisonnant de manière similaire, on pose $\lambda = \frac{1}{a}$ et on s'est ramené au système :

$$\frac{dx}{dt} = \dot{x} = x(1 - x)$$

Détermination des dilatations Voici pour information comment les trouver. Les calculs relèvent également de l'algèbre linéaire.

— on remplace chaque inconnue v par $\lambda^{n_v} v$, où les n_v et λ sont indépendants du temps. Attention, le temps fait partie des inconnues (cela revient à dilater le temps). En utilisant (6.3) :

$$\frac{\lambda^{n_x} dx}{\lambda^{n_t} dt} = \lambda^{n_x} x(\lambda^{n_a} a - \lambda^{n_b} b \lambda^{n_x} x)$$

— on réécrit l'équation sous la forme $\frac{dx}{dt} = \dot{x} =$

$$\frac{dx}{dt} = \dot{x} = \lambda^{n_t} x(\lambda^{n_a} a - \lambda^{n_b} b \lambda^{n_x} x)$$

— on identifie avec l'équation 6.3)

$$\lambda^{n_t} x(\lambda^{n_a} a - \lambda^{n_b + n_x} bx) = x(a - bx)$$

— ce qui fournit un système linéaire d'équations en les n_v, sachant que le système doit être vrai pour toutes valeurs des variables et des paramètres. Pour ce faire, on fait apparaître les monômes en les variables et paramètres.

$$(\lambda^{n_t + n_a} - 1)ax - (\lambda^{n_t + n_b + n_x} - 1)bx^2 = 0$$

$$\begin{cases} n_t + n_a &= 0 \\ n_t + n_b + n_x &= 0 \end{cases}$$

— le système est de dimension deux, il y a donc deux dilatations. On peut espérer réduire de deux paramètres (1 paramètre par dilatation). En prenant comme première solution $n_t = n_a = 0$ et $n_b = 1$ et $n_x = -1$, on retrouve notre première dilatation ϕ_2.

— rechercher d'autres dilatations jusqu'à épuisement (des dilatations). Ici on doit en retrouver une, correspondant à ϕ_3

Pour information, la première dilation peut être obtenue en Maple en utilisant :

```
verh2 := subs(c=0, verh);

phi2 := { a=a*l^na,
          b=b*l^nb,
          c=c*l^nc,
          x(t)=x(t)*l^nx };
```

```
verhphi2 := subs(phi2, verh2);

verhphi2 := diff(x(t),t) - solve(verhphi2, diff(x(t),t) );

cond := verh2 - verhphi2;
cond := subs(x(t)=x, cond);
cond := expand(cond);

syst := [ coeffs(cond, [x,a,b,c]) ];
syst := simplify(syst);
syst := map( eq->subs(l=1, diff(eq,l)), syst);

solve(syst);
```

La dernière dilatation peut être trouvée en utilisant

```
##########################################
verh3 := subs(b=1, verh2);

# on n'inclut pas encore la transformation sur le temps (voir plus bas)
phi3 := { a=a*l^na,
          x(t)=x(t)*l^nx
        };

verhphi3 := subs(phi3, verh3);

# on tient compte du temps grâce au l^nt
verhphi3 := diff(x(t),t) - l^nt*solve(verhphi3, diff(x(t),t) );

cond := verh3 - verhphi3;
cond := subs(x(t)=x, cond);
cond := expand(cond);

syst := [ coeffs(cond, [x,a,b,c]) ];
syst := simplify(syst);
syst := map( eq->subs(l=1, diff(eq,l)), syst);

solve(syst);
```

6.1.3 Remarques

Il est conseillé de d'abord traiter les symétries de translation, et ensuite de traiter les dilatations. Il faut aussi faire attention à traiter les cas spéciaux éventuels lors des dilatations car la dilatation n'a de sens que pour $\lambda \neq 0$ (pour que la dilatation soit une transformation inversible). Les cas spéciaux doivent être traités séparément (ils correspondent en général à l'annulation d'un coefficient). Un autre point important est le facteur de dilatation introduit sur le temps : il faut s'assurer que le facteur de dilatation sur le temps est positif pour préserver la stabilité des points fixes.

Propriétés préservées par les translations et dilatations :

— le nombre de points fixes ;
— la stabilité des points fixes (si le temps a été dilaté avec un facteur strictement positif) ;
— l'existence d'oscillations.

6.2 Illustration avec MABSys

```
> sys := [ diff(x(t),t) = x(t)*(a-b*x(t)) ];
                              d
                    sys := [-- x(t) = x(t) (a - b x(t))]
                              dt

> InvariantizeByScalings(sys, [a,b] );
            d                                                  x b
          [[-- x(t) = -x(t) (-1 + x(t))], [t = t a, x = ---], [a, b]]
            dt                                                 a
```

Le second paramètre passé `[a,b]` signifie qu'on essaie de faire disparaître en priorité le paramètre a, et ensuite b si on ne peut pas faire disparaitre a.

Le deuxième élément renvoyé est le changement de variable qu'il faut effectuer dans le système réduit pour obtenir le système initial.

6.3 Réductions approchées

Dans ce cas, le système réduit n'est plus strictement équivalent au système initial, et il faut donc être prudent avant de conclure qu'une propriété du système réduit est aussi vraie pour le système initial (absence/existence d'oscillations, nombre de points fixes, ...). Nous présentons ici une réduction très célèbre basée sur le théorème de Tykonov, basée sur le principe des variables lentes/rapides. Cette réduction est détaillée dans [1].

6.3.1 Théorème de Tikhonov

Le théorème de Tikhonov permet de simplifier l'étude d'un système dynamique en distinguant deux types de variables : les lentes et les rapides. Étudions un exemple simple

$$\dot{x} = f(x, y) \tag{6.7a}$$

$$\dot{y} = \frac{1}{\epsilon} g(x, y) \tag{6.7b}$$

où epsilon est un paramètre qu'on suppose petit (proche de 0). On suppose aussi que les fonctions f et g ont le même ordre de grandeur.

Imaginons le scénario suivant : on veut intégrer le système (6.7) à partir d'un point (x_0, y_0) tel que $g(x_0, y_0) \neq 0$. Comme f et g sont du même ordre de grandeur et que ϵ est

très petit, on a $|\dot{y}(0)| \gg |\dot{x}(0)|$. Cela signifie que la dynamique sur y est plus rapide que celle de x. On dit que y est une variable rapide et que x est une variable lente.

En supposant que la solution est attirée par la courbe $g(x, y) = 0$, voilà comment va se dérouler l'intégration de notre solution :

phase rapide la valeur de y va changer très vite pour rejoindre la courbe $g(x, y)$. Pendant ce temps, x change peu. On appelle cela le régime transitoire

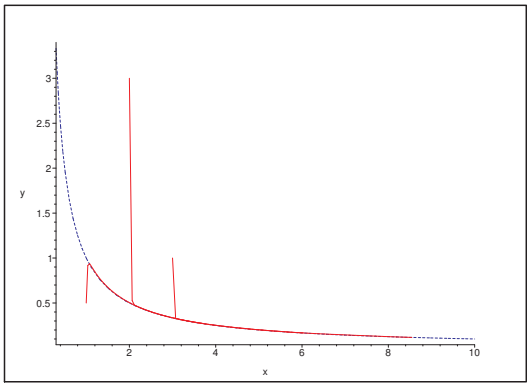
phase lente la solution va glisser le long (ou très près) de $g(x, y) = 0$, que l'on appelle variété lente

Exemple 9

$$\dot{x} = x + y \tag{6.8a}$$

$$\dot{y} = \frac{1}{\epsilon}(1 - xy) \tag{6.8b}$$

Une simulation numérique en Maple fournit



pour les $\epsilon = 0.01$ *et les trois conditions initiales* $(x_0, y_0) = (2, 3)$, $(3, 1)$ *et* $(1, 0.5)$. *On constate que la solution rejoint la courbe* $y = 1/x$ *(en pointillés) après un transitoire quasi vertical.*

Si la phase transitoire ne nous intéresse pas, on peut s'intéresser à la dynamique du système une fois que le système a rejoint la variété lente. Pour ce faire, on s'intéresse au système :

$$\dot{x} = f(x, y) \tag{6.9a}$$

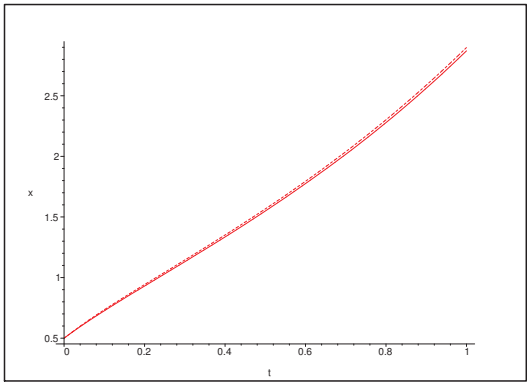
$$0 = g(x, y) \tag{6.9b}$$

Sous certaines hypothèses, on peut exprimer y en fonction de x grâce à l'équation $g(x, y) = 0$, et reporter la valeur de y ainsi obtenue dans l'équation en $\dot{x}$. On obtient alors une équation différentielle en x uniquement, et on a réduit le nombre de variables du système dynamique (puisque que y a disparu).

Exemple 10 (suite de l'exemple 9) *$g(x,y) = 0$ fournit $y = 1/x$. Ainsi, on a*

$$\dot{x} = x + 1/x \tag{6.10}$$

Pour vérifier la validité de cette approximation, on peut dessiner sur un même graphe les solutions de (6.10) en pointillés et de (6.8) en trait plein, pour la condition initiale $(0.5, 2)$ *choisie sur la variété lente.*



Pour information, voici les commandes utilisées pour générer les graphiques précédents :

```
with(plots);
plotsetup(X11);
sys := [
    diff(x(t),t) = x(t)+y(t),
    diff(y(t),t) = (1/eps)*(1-y(t)*x(t))
];

ics := [ [ x(0)=2, y(0)=3 ], [ x(0)=3, y(0)=1 ], [ x(0)=1, y(0)=0.5 ] ];

L := [];
for ic0 in ics do

    sys0 := subs(eps=0.01, sys);
    sol0 := dsolve( [ op(sys0), op(ic0)], numeric, stiff=true,
                    maxfun=1000000, abserr=1e-2, numpoints=1000);
    L := [ op(L), odeplot(sol0,[x(t),y(t)],t=0..1) ];
od:
display([ op(L), plot(1/x,x=0.3..10,color=blue,linestyle=DOT)]);


sys0 := subs(eps=0.01, sys);
sol0 := dsolve( [ op(sys0), x(0)=1/2, y(0)=2 ], numeric, stiff=true,
                maxfun=1000000, abserr=1e-2);
P0 := odeplot(sol0,[t,x(t)],t=0..1,linestyle=DOT);
sol1 := dsolve( [ diff(x(t),t) = x(t)+1/x(t) ,
                  x(0)=1/2], numeric, stiff=true,
                maxfun=1000000, abserr=1e-2);
P1 := odeplot(sol1, [t,x(t)],t=0..1);
display([P0,P1]);
```

6.3.2 Application aux systèmes biologiques

La réduction précédente s'applique aux équations de la biologie, à condition de les écrire sous la forme appropriée (i.e. dans les bonnes coordonneés).

Considérons un système biologique composé de réactions lentes et rapides. Par réaction rapide, on entend réaction qui atteint (beaucoup plus) rapidement son équilibre que les réactions lentes. La distinction entre réactions lentes et rapides peut se faire (en première approche) suivant les valeurs des constantes de réactions ; en divisant l'ensemble les constantes de réactions en grandes et petites, les réaction rapides sont les réactions dont l'une des constantes est grande.

Exemple 11 *On considère un simple dimérisation entre une protéine et elle-même. On suppose que la protéine est dégradée. La dimerisation est supposée rapide, la dégradation supposée lente.*

$$2P \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} P_2 \tag{6.11a}$$

$$P \xrightarrow{k_2} \varnothing \tag{6.11b}$$

Si on a $k_1, k_{-1} \gg k_2$, la dimérisation peut être supposée rapide : pendant le temps qu'elle met à atteindre son équilibre ($k_1 P^2 \simeq k_{-1} P_2$), la dégradation ne s'est presque pas produite.

Pour exprimer le fait qu'une constante k_1 est très grande devant une constante k_2 (i.e. $k_1 \gg k_2$), on écrit $k_1 = \frac{1}{\epsilon}\bar{k}_1$, où k_2 et $\bar{k}_1$ sont du même ordre de grandeur. $\frac{1}{\epsilon}$ aura le même rôle que dans le théorème de Tikhonov. On peut alors écrire le système dynamique en faisant apparaître les $\frac{1}{\epsilon}$ dans l'espoir d'appliquer le théorème de Tikhonov.

Exemple 12 (Suite)

$$\dot{P} = -k_2 P + \frac{1}{\epsilon}(-2\bar{k}_1 P^2 + 2\bar{k}_{-1} P_2) \tag{6.12a}$$

$$\dot{P}_2 = \frac{1}{\epsilon}(\bar{k}_1 P^2 - \bar{k}_{-1} P_2) \tag{6.12b}$$

Le théorème de Tikhonov ne s'applique pas car $1/\epsilon$ apparaît dans la première équation.

Le problème est que la notion de réaction lente/rapide est différente de celle de variable lente/rapide. Ici la variable P apparaît à la fois dans une réaction lente et une réaction rapide, pour cette raison le théorème ne s'applique pas. Pour résoudre ce problème, il faut récrire le système d'équations (6.12) dans les bonnes coordonnées lentes/rapides.

Exemple 13 (Suite) *Sur notre exemple, on doit introduire une nouvelle variable lente. Pour la trouver, suivons l'intuition suivante : une variable lente ne doit pas varier vite lorsque la réaction rapide se produit. Il suffit donc de considérer la loi de conservation $n = P + 2P_2$ conséquence de la réaction rapide (6.11a). Il s'agit tout simplement de la quantité totale de protéine.*

$$\dot{n} = -k_2 P \tag{6.13a}$$

$$\dot{P}_2 = \frac{1}{\epsilon}(\bar{k}_1 P^2 - \bar{k}_{-1} P_2) \tag{6.13b}$$

La première équation ne contient plus de $1/\epsilon$ *mais il y a une inconnue de trop, à savoir* P*. On peut la faire disparaître en se servant de* $n = P + 2P_2$ *:*

$$\dot{n} = -k_2(n - 2P_2) \tag{6.14a}$$

$$\dot{P}_2 = \frac{1}{\epsilon}(\bar{k}_1(n - 2P_2)^2 - \bar{k}_{-1} P_2) \tag{6.14b}$$

Le théorème de Tiknonov peut enfin s'appliquer. On peut donc supposer $(\bar{k}_1(n - 2P_2)^2 - \bar{k}_{-1}P_2) = 0$ *Pour obtenir des calculs plus simples, on poursuit les calculs sur (6.13). Comme* $\bar{k}_1 P^2 = \bar{k}_{-1} P_2$*, on a* $2\bar{k}_1 P\dot{P} = \bar{k}_{-1}\dot{P}_2$*. En combinant cette dernière équation avec* $\dot{n} = \dot{P} + 2\dot{P}_2$ *et (6.13a), on obtient*

$$\dot{P} + 4\frac{\bar{k}_1}{\bar{k}_{-1}} P\dot{P} = -k_2 P$$

ce qui donne finalement

$$\dot{P} = -\frac{k_2 P}{4KP + 1}$$

où $K = \frac{\bar{k}_1}{k_{-1}} = \frac{k_1}{k_{-1}}$

6.3.3 Automatisation de la réduction de Tikhonov

La réduction précédente peut être entièrement automatisée en utilisant de l'élimination différentielle, dont nous allons donner l'intuition sur quelques exemples. Pour automatiser la méthode, on suit les étapes suivantes :

1. on introduit une nouvelle variable F_i pour chaque réaction rapide R_i. Cette variable F_i correspond à la vitesse de la réaction R_i que l'on suppose inconnue pour l'instant.
2. on écrit le système dynamique en utilisant les F_i pour les réactions rapides, au lieu de leur vitesse obtenue par la loi d'action de masse
3. on écrit les équations assurant que les réactions rapides sont à l'équilibre
4. on calcule les valeurs des F_i et on les remplace par leurs valeurs
5. on obtient alors un système dont les solutions "vivent" sur les équilibres des réactions rapides

Exemple 14 *Illustrons la méthode sur l'exemple précédent :*

1. *on introduit la grandeur* F_1 *comme étant la vitesse de la réaction rapide (6.11a)*

2.

$$\dot{P} = -k_2 P - 2F_1 + 2F_2 \tag{6.15a}$$

$$\dot{P}_2 = F_1 - F_2 \tag{6.15b}$$

3. on suppose que (6.11a) est à l'équilibre, ce qui donne $P_2 = KP^2$ *où* $K = \frac{k_1}{k_{-1}}$

4. on résout le système (voir plus bas) et on trouve

$$F_1 - F_2 = -\frac{2Kk_2P^2}{4KP+1}$$

5. on en déduit que

$$\dot{P} = -k_2 P - 2\left(-\frac{2Kk_2P^2}{4KP+1}\right) = -\frac{k_2 P}{4KP+1} \tag{6.16a}$$

$$\dot{P}_2 = -\frac{2Kk_2P^2}{4KP+1} \tag{6.16b}$$

La résolution du système à l'étape 4 peut se faire par élimination différentielle :

```
> with(diffalg):
> Sys := [ P[t] - (- k2*P -2*F1 + 2*F2 ), P2[t] - F1 + F2 ]:
> Equi := [ K*P^2-P2]:
> F := field_extension (transcendental_elements=[K,k2]):
> R := differential_ring (ranking=[ [F1,F2], [P2,P] ],
>                         derivations=[t], field_of_constants=F):
> Ids := Rosenfeld_Groebner ( [op(Sys),op(Equi)], R):
> rules := rewrite_rules( Ids[1] );
                                                  2
                    -4 F2[] P[] K - F2[] + 2 P[]  K k2             k2 P[]
rules := [F1[] = - ---------------------------------, P[t] = - -----------,
                              4 P[] K + 1                       4 P[] K + 1

                 2
    P2[] = K P[] ]
```

6.3.4 Illustration avec MABSys

```
> R1 := NewReaction(2*P, P2, MassActionLaw(k1), fast=true):
> R2 := NewReaction(P2, 2*P, MassActionLaw(km1), fast=true):
> R3 := NewReaction(P, 0, MassActionLaw(k2)):
>
> RS := [R1,R2,R3]:
> out := ModelReduce(RS, [P,P2]):
>
> sys := out[1][1];
               d             km1 k2 P(t)    d            2 km1 P2(t) k2
      sys := [-- P(t) = - ---------------, -- P2(t) = - ---------------]
               dt          km1 + 4 k1 P(t)  dt           km1 + 4 k1 P(t)
```

Annexe A

Langage SBML

Voici un exemple de fichier SBML, qui décrit la dégradation enzymatique.

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml level="2" version="3" xmlns="http://www.sbml.org/sbml/level2/version3">
  <model name="EnzymaticReaction">
    <listOfUnitDefinitions>
      <unitDefinition id="per_second">
        <listOfUnits>
          <unit kind="second" exponent="-1"/>
        </listOfUnits>
      </unitDefinition>
      <unitDefinition id="litre_per_mole_per_second">
        <listOfUnits>
          <unit kind="mole" exponent="-1"/>
          <unit kind="litre" exponent="1"/>
          <unit kind="second" exponent="-1"/>
        </listOfUnits>
      </unitDefinition>
    </listOfUnitDefinitions>
    <listOfCompartments>
      <compartment id="cytosol" size="1e-14"/>
    </listOfCompartments>
    <listOfSpecies>
      <species compartment="cytosol" id="ES" initialAmount="0" name="ES"/>
      <species compartment="cytosol" id="P" initialAmount="0" name="P"/>
      <species compartment="cytosol" id="S" initialAmount="1e-20" name="S"/>
      <species compartment="cytosol" id="E" initialAmount="5e-21" name="E"/>
    </listOfSpecies>
    <listOfReactions>
      <reaction id="veq">
        <listOfReactants>
          <speciesReference species="E"/>
          <speciesReference species="S"/>
        </listOfReactants>
        <listOfProducts>
          <speciesReference species="ES"/>
```

```
  </listOfProducts>
  <kineticLaw>
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      <apply>
        <times/>
        <ci>cytosol</ci>
        <apply>
          <minus/>
          <apply>
            <times/>
            <ci>kon</ci>
            <ci>E</ci>
            <ci>S</ci>
          </apply>
          <apply>
            <times/>
            <ci>koff</ci>
            <ci>ES</ci>
          </apply>
        </apply>
      </apply>
    </math>

    <listOfParameters>
      <parameter id="kon" value="1000000" units="litre_per_mole_per_second"/>
      <parameter id="koff" value="0.2" units="per_second"/>
    </listOfParameters>
  </kineticLaw>
</reaction>
<reaction id="vcat" reversible="false">
  <listOfReactants>
    <speciesReference species="ES"/>
  </listOfReactants>
  <listOfProducts>
    <speciesReference species="E"/>
    <speciesReference species="P"/>
  </listOfProducts>
  <kineticLaw>
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      <apply>
        <times/>
        <ci>cytosol</ci>
        <ci>kcat</ci>
        <ci>ES</ci>
      </apply>
    </math>
    <listOfParameters>
      <parameter id="kcat" value="0.1" units="per_second"/>
    </listOfParameters>
  </kineticLaw>
</reaction>
```

```
    </listOfReactions>
  </model>
</sbml>
```

Bibliographie

[1] François Boulier, Marc Lefranc, François Lemaire, and Pierre-Emmanuel Morant, *Model Reduction of Chemical Reaction Systems using Elimination*, lately accepted to MACIS 2007, `http://hal.archives-ouvertes.fr/hal-00184558/en/`, 2007.

[2] Martin Feinberg, *Complex balancing in general kinetic systems*, Archive for Rational Mechanics and Analysis **49** (1972), 187–194.

[3] ———, *Lectures on chemical reaction networks*, `http://www.che.eng.ohio-state.edu/~FEINBERG/LecturesOnReactionNetworks`, 1979.

[4] François Lemaire and Aslı Ürgüplü, *Mabsys : Modeling and analysis of biological systems*, Proceedings of ANB2010, 2010.