Simulation graphique du comportement entrée/sortie des systèmes dynamiques*

F. Boussemart, V. Hoang Ngoc Minh
Laboratoire d'Informatique Fondamentale de Lille
U.A. 369 du C.N.R.S.- Université de Lille I.
59655 Villeneuve d'Ascq Cedex. FRANCE.

Abstract

We introduce in this paper a new method for studying nonlinear control systems behaviour through polynomial approximation in a graphical framework. In our implementation which is done in the *Scratchpad* programming language, we use the Bezier representation in order to have an interactive input modification on one hand and an efficient output drawing in displaying the system behaviour on the other hand.

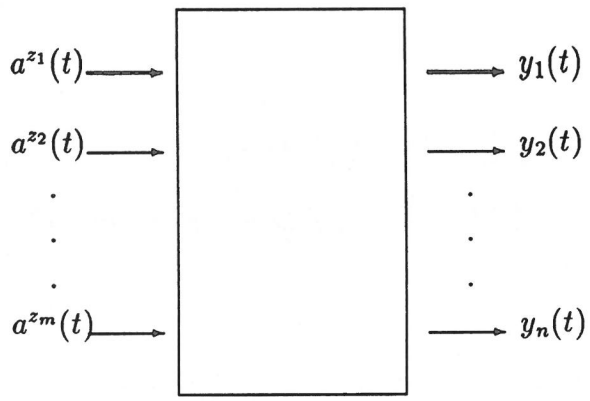
1 Séries génératrices

Nous présentons ici un procédé original pour étudier graphiquement le comportement entrée / sortie des systèmes analytiques par des approximations polynomiales.
Nous étudions les systèmes dynamiques analytiques de la forme:

$$(S)\begin{cases} \dot{q}(t) = \sum_{z \in Z} a^z(t) A_z(q) \\ y(t) = h(q(t)) \end{cases}$$

où $Z = \{z_0, z_1, ..., z_m\}$ est un alphabet, q appartient à la variété analytique réelle Q de dimension N, A_z est un champ de vecteurs analytique sur Q, a^z est une application réelle continue sur $\mathbb{R}_+$ ($a_{z_0} \equiv 1$), l'observation h est une application analytique de Q dans $\mathbb{R}^p$. Du point de vue de l'utilisateur, un système dynamique est une boite noire qui prend en entrée des signaux paramétrés par le temps et qui fournit en sortie des signaux du même type.

*travail partiellement supporté par IBM France et effectué dans le cadre d'une étude jointe avec IBM Yorktown, USA



La sortie d'un tel système pour une entrée a est donnée par la *formule fondamentale de Fliess* que nous écrivons sous la forme:

$$y(t) = \sum_{w \in Z^*} (A_w \circ h)_{|_q} \int_0^t \delta_a w$$

avec

$$\int_0^t \delta_a w = \begin{cases} 1 & si \quad w = \varepsilon \\ \int_0^t \left(\int_0^\tau \delta_a v \right) d\xi_z(\tau) & si \quad w = vz \end{cases}$$

Elle est donc entièrement caractérisée par sa série génératrice, qui est une série formelle en variables non commutatives:

$$\sigma h_{|_q} = \sum_{w \in Z^*} (A_w \circ h)_{|_q} w$$

Le but de ce travail est de simuler graphiquement le comportement entrée/sortie des séries génératrices polynomiales.

2 Evaluation

Soit $Z = \{z_0, z_1, ..., z_p\}$ un alphabet que nous appelons *alphavet de codage.* Soit A une R-algèbre engendrée par p opérateurs d'intégration $J_0, J_1, ..., J_p$. Les opérateurs J_k sont définis comme suit:

$$\forall f : R \to R \qquad J_k(f) = \int_0^t a_k(\tau) f(\tau) d\tau$$

où les a_k sont des fonctions de R dans R.
Nous décidons de coder chaque opérateur J_k par une lettre z_k de Z.
Tout polynôme P de $R < Z >$ devient un codage d'un opérateur d'intégration défini par combinaison des J_k: le produit de deux lettres engendre une composition, la somme reste une somme de termes.

Exemple 2.1 :
Soit $Z = \{z_0, z_1\}$
Soit $P = z_0^2 + 2z_1 + 3$
P code l'opérateur d'intégration J_p *tel que :*

$$J_p = J_0 \circ J_0 + 2J_1 + 3$$

On obtient pour deux entrées a_0 *et* a_1 *:*

$$J_p(f) = \int_0^t a_0(\tau) \left(\int_0^\tau a_0(\sigma) f(\sigma) d\sigma \right) d\tau + 2 \int_0^t a_1(\tau) f(\tau) d\tau + 3f(t)$$

Pour faciliter la programmation, nous avons introduit la transformation d'évaluation qui associe à toute série génératrice la sortie correspondante pour une entrée donnée.

2.1 Evaluation des mots

L'évaluation d'un mot w de Z^* par rapport au noyau f pour l'entrée a est l'intégrale itérée définie par récurence sur la longueur de w par:

$$\mathcal{E}_a(f; w)(t) = \begin{cases} f(t) & si \quad w = \varepsilon \\ \int_0^t \mathcal{E}_a(f; v)(\tau) d\xi_z(\tau) & si \quad w = vz \end{cases}$$

En particulier, lorsque f est la fonction unité 1, nous retrouvons l'intégrale itérée $\int_0^t \delta_a w$ associée au mot w que nous notons $\mathcal{E}_a(w)(t)$.

2.2 Evaluation des polynômes

Nous appelons évaluation du polynôme P de $\mathbb{K} < Z >$ ($\mathbb{K} = \mathbb{R}$ ou $\mathbb{C}$) par rapport au noyau f pour l'entrée a la fonctionnelle:

$$\mathcal{E}_a(f; P) = \sum_{w \in supp(P)} < P \mid w > \mathcal{E}_a(f, w)$$

En particulier, pour f=1, l'évaluation de P de $\mathbb{K} < Z >$ pour a est la fonction

$$\mathcal{E}_a(P) = \mathcal{E}_a(1, P)$$

3 Le simulateur

Les travaux de Fliess et Sussman [Fliess 75,Fliess 76,Fliess 81,Sussman] ont montré qu'on peut approcher uniformément le comportement entrée/sortie des systèmes analytiques par

des systèmes bilinéaires (i.e. de série génératrice rationnelle), et même polynomiaux (i.e. de série génératrice finie).
En particulier, on peut approcher le comportement entrée/sortie d'un système de série génératrice g, par des polynômes g_k obtenus en tronquant g à un ordre k donné.

$$\forall w \in Z^* \quad g_k = \sum_{|w| \leq k} < g \mid w > .w$$

Nous nous attachons dans le présent travail à simuler le comportement entrée/sortie des séries génératrices finies pour des entrées polynomiales. Le calcul de la sortie peut se faire exactement, et nous l'avons implanté dans le cadre d'une représentation des entrées et des sorties par des polygones de Bézier (On trouve en annexe une description technique de la représentation Bézier).
Cette technique présente trois avantages principaux:

- On peut approximer uniformément une série quelconque sur tout l'intervalle de temps considéré,
- les calculs sont simples et rapides,
- le calcul des points de contrôle étant peu coûteux, il permettra de mieux exploiter l'interactivité.

4 Implantation

Pour implanter l'évaluation comme la simulation graphique, nous avons utilisé le langage de calcul formel *Scratchpad* .

4.1 Généricité en scratchpad

L'idée de la généricité est de permettre de paramétrer des fonctions non seulement par des variables, mais également par le type de celles-ci.
Par exemple, l'algorithme utilisé pour effectuer l'opération $a * b$ sera déterminé en fonction du type de a et b. a et b seront indifféremment des entiers, des rationnels, des matrices, ...
Cette notion de généricité s'étend aux types. Par exemple, le type *matrice* prend en paramètre le type de ses éléments: le même domaine *Scratchpad* permet de créer des matrices d'entiers, de complexes, etc.
Cette notion très forte permet d'appliquer les algorithmes à des classes les plus étendues possibles de variables.

4.2 Représentation récursive des polynômes

Scratchpad permet une représentation récursive des données [Oussous & Petitot]. Cela signifie par exemple qu'un polynôme à deux variables x et y sera traité comme un polynôme en x dont les coefficients sont des polynômes en y.
Cette technique est particulièrement bien adaptée à l'implantation des polynômes non commutatifs. Rappelons la définition du résiduel d'un mot:

$$w \triangleright z = \begin{cases} v & si \quad w = zv \\ 0 & sinon \end{cases}$$

Nous pouvons utiliser directement le lemme de reconstruction:
Soit $P \in I\!R < Z >$

$$P = P_0 + \sum_{z \in Z} z(P \triangleright z)$$

En mémoire, la représentation d'un polynôme non commutatif sera la liste de ses résiduels par les éléments de Z. Chaque élément étant lui même un polynôme non commutatif, on pourra réitérer le processus.

Exemple 4.1 :
Soit $Z = \{z_0, z_1\}$
Soit $P = 1 + z_0^2 + 2z_0z_1 + z_1z_0 + z_1$
$\quad = 1 + z_0(z_0 + 2z_1) + z_1(1 + z_0)$
P est représenté par la liste:

$$(1, (z_0, P \triangleright z_0), (z_1, P \triangleright z_1))$$

avec

$$P \triangleright z_0 = ((z_0, 1), (z_1, 2))$$
$$P \triangleright z_1 = (1, (z_0, 1))$$

4.3 Evaluation des polynômes en scratchpad

Nous utilisons directement le lemme de reconstruction:
Soit

$$P = P_0 + \sum_{z \in Z} z(P \triangleright z)$$

Alors

$$\mathcal{E}(P) = P_0 + \sum_{z \in Z} \mathcal{E}(\xi_z; P \triangleright z)$$

avec

$$\xi_z = \int_0^t a_z(\tau) d\tau$$

Nous avons ici un procédé totalement récursif de calcul des intégrales itérées.

4.4 Evaluation pour le simulateur

Lors de l'implantation en *Scratchpad* , le principe de généricité décrit plus haut permet l'évaluation d'un type quelconque de polynômes pourvu que les opérateurs de produit, de somme et d'intégration soient définis.
Pour ce qui nous concerne, les entrées et sorties du système sont décrites par des *courbes Bézier* dont la représentation interne est un polygone (voir annexe).
Le système dynamique reçoit une liste de points de l'espace. Chaque entrée a^{z_i} est la liste des coordonnées des *points de contrôle* P_i sur un axe donné. Cette liste est la représentation interne d'un polynôme dans la base de Bernstein (voir annexe). Les sorties sont du même type.
Les lettres z_i de l'alphabet de codage représentent les opérateurs d'intégration J_i qui travaillent dans la base de Bernstein (voir annexe). De tels intégrateurs, très rapides, donnent au simulateur tout son intérêt: un temps d'exécution très court entre la donnée des entrées et l'affichage de la sortie.
A titre indicatif, pour des entrées et une série polynomiale de degré inférieur à 6, les temps de réponse (tracé compris) restent de l'ordre de 10 secondes.

5 Conclusion

Le simulateur apporte un progrès réel pour l'étude du comportement des systèmes dynamiques: les entrées sont facilement exprimées sous leur forme symbolique et seront plus tard modifiables de manière interactive par manipulation des points de contrôle. Les sorties dont l'image graphique est obtenue très rapidement, donnent une première idée presque indispensable du comportement du système.
L'étude des approximations polynomiales s'est par conséquent avérée tout à fait satisfaisante, aussi nous envisageons d'étendre le développement du simulateur à des classes plus étendues de systèmes dynamiques et d'entrées qu'on pourra représenter par des courbes rationnelles [Fiorot & Jeannin] ou des splines [Sablonière] .

References

[De Casteljau] **P. De Casteljau.** Formes à pôles.
Mathématiques et C.A.O., Hermès 1985

[Farouki & Rajan] **R.T. Farouki, V.T. Rajan.** Algorithms for polynomials in Bernstein form.
Computer Aided Geometric Design 1988

[Fiorot] **J.C. Fiorot.** Courbes Bézier.
publication de l'U.F.R. d'I.E.E.A. de Lille I

[Fiorot & Jeannin] **J.C. Fiorot, P. Jeannin.** Courbes et surfaces rationnelles, application à la C.A.O.
ed. Masson 1989

[Fliess 75] **M. Fliess.** Séries de Volterra et séries formelles non commutatives.
C.R. académie des sciences Paris 1975

[Fliess 76] **M. Fliess.** Un outil algébrique; les séries formelles non commutatives.
Notes Econom. Math. Syst. 131, p. 122-148 Springer-Verlag Berlin 1976

[Fliess 81] **M. Fliess.** Fonctionnelles causales non linéaires et indéterminées non commutatives.
Bull, soc. math. France, 109 1981

[Hoang] **V. Hoang Ngoc Minh** Evaluation transform with kernel function.
A paraître dans T.C.S. 1992

[Lane & Riesenfeld] **J.M. Lane, R.F. Riesenfeld.** A theoretical development for the computer generation and display of piecewise polynomial surfaces.
IEEE Transactions on Pattern Analysis and machine Intelligence 2, p. 35-46 1980

[Oussous & Petitot] **N.E. Oussous, M. Petitot** Polynômes non commutatifs: représentation et traitement par les systèmes de calcul formel.
Formal power series and algebraic combinatorics. Bordeaux may 1991

[Sablonière] **P.Sablonière.** Bases de Bernstein et approximants splines.
Thèse d'état 1982

[Sussman] **H.J. Sussman.** Semigroup representations, bilinear approximation of input-output maps and generalized inputs.
Mathematical Systems Theory. Lect. notes econom. math. syst. 131, p. 172-191 Springer-Verlag Berlin 1976

Annexe

La représentation Bézier

L'outil mathématique nécessaire au tracé des courbes Bézier est la base de Bernstein [Fiorot].

Definition 5.1 *Soit $\mathbb{P}_n$ l'espace des polynômes de degré inférieur ou égal à n.*
Soit $B_i^n(t) = \begin{pmatrix} n \\ i \end{pmatrix} t^i(1-t)^{n-i}$ *avec* $\begin{pmatrix} n \\ i \end{pmatrix} = \frac{n!}{i!(n-i)!}$
alors l'ensemble des $B_i^n(t)$ $t \in \{0, 1, .., n\}$ *est une base de* $\mathbb{P}_n$.
Il est appelé Base de Bernstein.

produit de deux polynômes

La multiplication de deux polynômes P_1 et P_2 respectivement exprimés dans les bases de Bernstein de degré m et n donnera un polynôme exprimé dans la base de degré n+m [Farouki & Rajan]:
Soit

$$P_1(t) = \sum_{i=0}^{m} a_i B_i^m(t)$$
$$P_2(t) = \sum_{i=0}^{n} b_i B_i^n(t)$$

Alors

$$P_1(t).P_2(t) = \sum_{i=0}^{n+m} c_i B_i^{n+m}(t)$$

avec

$$c_k = \sum_{j=max(0,k-n)}^{min(k,m)} \frac{\begin{pmatrix} m \\ j \end{pmatrix}\begin{pmatrix} m \\ k-j \end{pmatrix}}{\begin{pmatrix} m+n \\ k \end{pmatrix}} a_j b_{k-j}$$

intégration des polynômes

L'intégration d'un polynôme exprimé dans la base de Bernstein de degré n donne un résultat dans la base de degré n+1 par la formule suivante:
Soit

$$P(t) = \sum_{i=0}^{n} a_i B_i^n(t)$$

alors

$$\int P(t) = \sum_{i=0}^{n+1} \sigma a_i B_i^{n+1}(t)$$

avec

$$\begin{cases} \sigma a_0 = 0 \\ \\ \sigma a_k = \dfrac{1}{n+1} a_{k-1} + \sigma a_{k-1} \qquad k \in [1, .., n+1] \end{cases}$$

Courbes polynomiales et polygones Bézier

Les courbes polynomiales qui nous intéressent sont données sous la forme de m polynômes en la variable t. Il faut alors les exprimer dans une même base de Bernstein: celle du degré le plus élevé. On peut ensuite exprimer la courbe sous forme d'un seul polynôme à coefficients dans $\mathbb{R}^m$.
Nous obtenons finalement l'expression d'une courbe Bézier dans l'espace $\mathbb{R}^m$:

$$B_n(\mathcal{Q}, t) = \sum_{i=0}^{n} B_i^n(t) Q_i \qquad Q_i \in \mathbb{R}^m$$

La courbe $B_n(t)$ est appelée la courbe associée au polygone Bézier $\mathcal{Q} = \{Q_0, Q_1, ..., Q_n\}$
Les Q_i sont appelés les points de contrôle de la courbe.

Tracé de la courbe

Algorithme de De Casteljau

Le but de cet algorithme est de retrouver la valeur du polynôme P(t) au temps t_0 à partir de son polygone Bézier [De Casteljau].
Soit $\mathcal{Q} = \{Q_0, Q_1, ..., Q_n\}$ le polygone associé à P(t):

pour i = 0 à n posons $Q_i^{(0)}(t_0) = Q_i$
pour j = 1 à n faire
    pour i = 0 à n-j faire
        $Q_i^{(j)}(t_0) = (1 - t_0) Q_i^{(j-1)}(t_0) + t_0 Q_{i+1}^{(j-1)}(t_0)$

résultat : $P(t_0) = Q_0^{(n)}(t_0)$

Algorithme de subdivision

Cet algorithme est effectivement utilisé pour le calcul des points de la courbe [Fiorot, Fiorot & Jeannin].

Considérons la courbe $B_n(t)$ associée au polygone $\mathcal{Q} = \{Q_0, ..., Q_n\}$. Calculons sa valeur en $t_0 = \frac{1}{2}$ par l'algorithme de De Casteljau. On montre [Lane & Riesenfeld] que les deux parties de la courbe correspondant respectivement à $t \in [0, \frac{1}{2}]$ et $t \in [\frac{1}{2}, 1]$ sont décrites par les deux courbes $B_1(t)$ et $B_2(t)$ respectivement associées aux polygones $\mathcal{Q}_1 = \{Q_0^{(0)}, Q_0^{(1)}, ..., Q_0^{(n)}\}$ et $\mathcal{Q}_2 = \{Q_0^{(n)}, Q_1^{(n-1)}, ..., Q_n^{(0)}\}$ où les $Q_i^{(j)}$ sont les points obtenus lors des calculs intermédiaires. Remarquons que ces points, lorsque $t_0 = \frac{1}{2}$ se calculent simplement:

$$Q_i^{(j)} = \frac{Q_i^{(j-1)} + Q_{i+1}^{(j-1)}}{2}$$

On peut alors itérer le processus sur $\mathcal{Q}_1$ et $\mathcal{Q}_2$ et ainsi jusqu'à obtenir une définition suffisante de l'image.

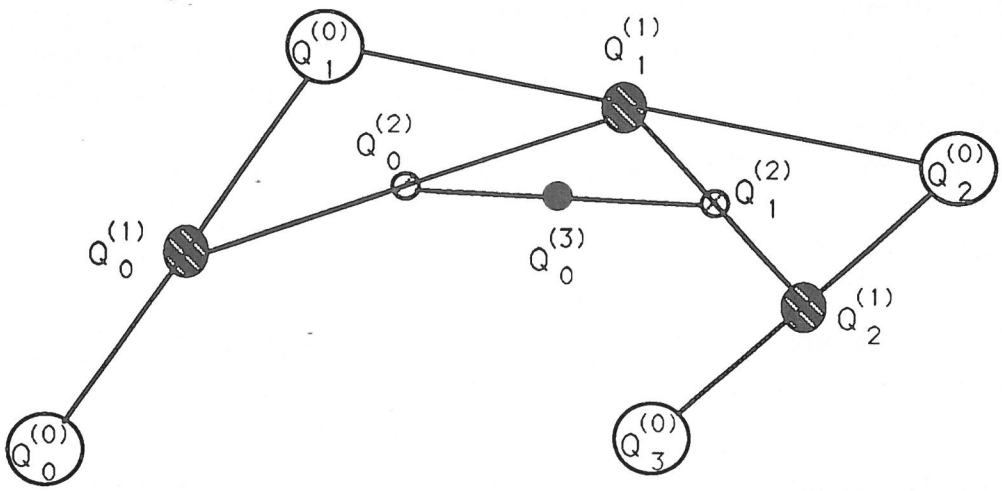


figure 1

La figure 1 montre une étape de l'algorithme de subdivision sur une courbe Bézier associée à un polygone de 4 points. La figure 2 nous donne les deux polygones $\mathcal{Q}_1 = \{P_0, P_1, P_2, P_3\}$ et $\mathcal{Q}_2 = \{Q_0, Q_1, Q_2, Q_3\}$ obtenus.

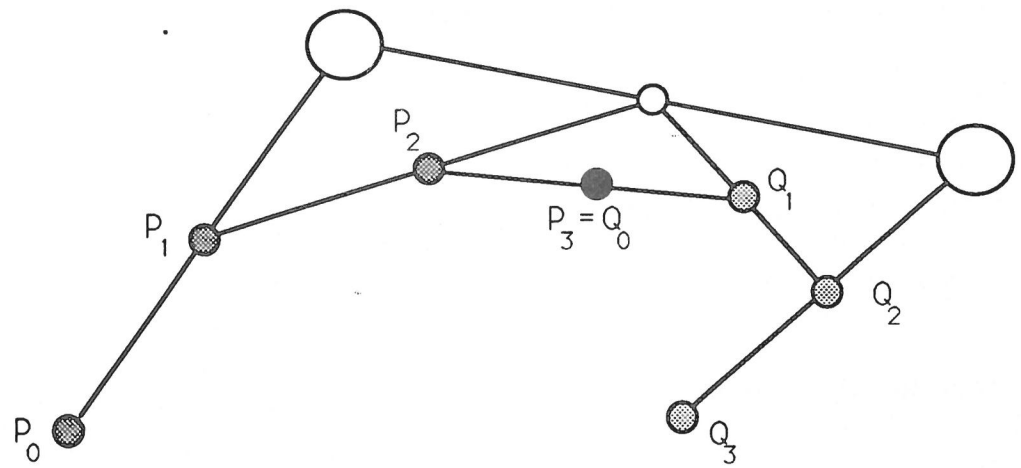
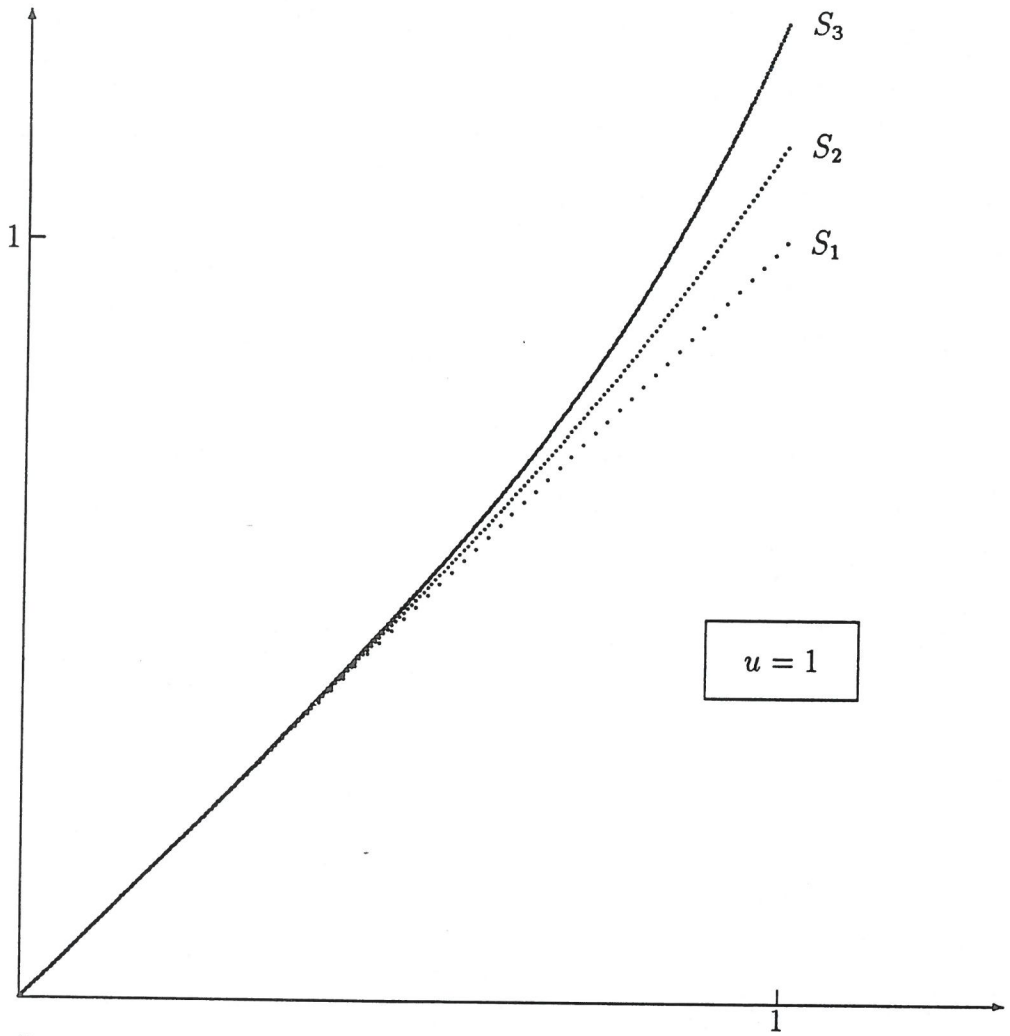


figure 2



Les trois courbes, fournies par le simulateur, décrivent les sorties du système suivant:

$$(S)\begin{cases} \dot{y} = u + y^2 \\ y(0) = 0 \end{cases}$$

L'expression de y devient:

$$y = \int_0^t u d\tau + \int_0^\tau y^2(\tau) d\tau$$

Ce qui revient à coder notre série sous la forme:

$$S = z_1 + S^{\text{ш}^2} z_0$$

Nous tronquons S à un ordre quelconque: le calcul du produit de mélange a une complexité exponentielle, aussi les valeurs de S sont très vite incalculables.
Le graphique représente les trois sorties pour:

$$S_1 = z_1$$
$$S_2 = z_1 + z_1^2 z_0$$
$$S_3 = z_1 + S_2^{\text{ш}^2} z_0$$

et pour l'entrée $u = 1$