Université Bordeaux 1

UFR de Mathématiques et Informatique

Institut de Mathématiques de Bordeaux

Algèbre linéaire « Boîte noire » : L'algorithme de Wiedemann

Paul DARTHOS

10 Mai 2011

Sous la direction de Karim BELABAS

Master 2 spécialité « Mathématiques approfondies »

Année Universitaire 2010–2011

Remerciements

Je tiens à remercier particulièrement Karim BELABAS pour sa disponibilité à tout instant (malgré toutes les responsabilités qu'il endosse), ainsi que pour tous les éclaircissements qu'il a pu apporter à mes nombreuses interrogations, avec beaucoup de pédagogie. Je lui en suis très reconnaissant.

Je voudrais aussi en profiter pour remercier l'ensemble du personnel du Château Lamartine, tant enseignant qu'administratif, dont la bonne humeur et la disponibilité associées au charme du lieu font que la préparation à l'Agrégation s'en trouve moins éprouvante.

Table des matières

Chapitre 1

Prélude

1.1 Introduction

L'objet de cette étude est la résolution de systèmes d'équations linéaires, via la méthode dite d'algèbre linéaire « boîte noire ».

Considérant une matrice $A \in K^{n \times n}$, et un vecteur $b \in K^n$, où K désigne un corps commutatif fini, on cherche une solution (ou une description de l'ensemble des solutions) $y \in K^n$ de l'équation $Ay = b$.

L'algèbre linéaire classique nous donne, via l'algorithme d'élimination de Gauß, un moyen de résoudre ce problème en $O(n^3)$ opérations dans K.

Cependant, nous verrons que dans le cas où A est une matrice dite « creuse », Douglas Wiedemann a proposé dans [2] un algorithme asymptotiquement plus rapide que l'élimination de Gauß. L'étude de cet algorithme sera au centre de ce mémoire.

1.2 Définitions

Il convient de définir clairement toutes les notions qui seront utiles à cette étude. Fixons tout d'abord la matrice $A \in K^{n\times n}$ inversible et le vecteur $b \in K^n$, qui correspondront au problème $Ay = b$. Dans ce cas, il y a une seule solution.

Définition. *La fonction associée à la matrice A est dite **boîte noire** si on n'en connaît pas le fonctionnement (i.e. les coefficients de la matrice) mais si l'on connaît, étant donné un vecteur $y \in K^n$ quelconque, la valeur de Ay.*

Définition. *Une matrice $M \in K^{n\times n}$ est dite **creuse** si elle a peu de coefficients non nuls. On peut préciser le nombre s de tels coefficients pour détailler la complexité des algorithmes appliqués.*

Définition. *Soit $V \neq \{0\}$ un K-espace vectoriel de dimension finie. Une suite $a = (a_i)_{i\in\mathbb{N}} \in V^{\mathbb{N}}$ est dite **récurrente linéaire dans K** s'il existe $d \in \mathbb{N}$ et $f_0, \dots f_d \in K$, où $f_d \neq 0$, tels que :*

$$\forall i \in \mathbb{N}, \sum_{j=0}^{d} f_j a_{i+j} = f_0 a_i + f_1 a_{i+1} + \cdots + f_{d-1} a_{i+d-1} + f_d a_{i+d} = 0$$

*Le polynôme $f(T) = \sum_{j=0}^{d} f_j T^j$ est un **polynôme caractéristique** (également appelé **générateur**) de a.*

Exemple. $a = (a_i)_{i\in\mathbb{N}}$ définie par $a_0 = 0$, $a_1 = 1$ et la relation de récurrence $a_{i+2} = a_{i+1} + a_i$, est la célèbre suite de Fibonacci, et $f(T) = T^2 - T - 1$ est un polynôme caractéristique de a.

1.3 La suite de Krylov

Dans cette section, on s'intéresse à la suite de Krylov associée à A et b.

Définition. *La **suite de Krylov** associée à la matrice A et au vecteur b est la suite de vecteurs $(A^i b)_{i\in\mathbb{N}}$.*

Lemme. *La **suite de Krylov** associée à la matrice A et au vecteur b est récurrente linéaire.*

Démonstration. Constatons que le degré du polynôme caractéristique associé à A, noté $\chi_A(T) = \sum_{i=0}^{n} c_i T^i$, est égal à n. Par conséquent, toute famille de $n+1$ vecteurs de la suite $(A^i)_{i\in\mathbb{N}}$ est liée. Ainsi, toute famille de $n+1$ vecteurs de la suite de Krylov $(A^i b)_{i\in\mathbb{N}}$ est liée :

$$\chi_A(A)b = c_0 b + c_1 Ab + \cdots + c_n A^n b = 0$$

Et l'on en tire :

$$\forall i \in \mathbb{N}, c_0 A^i b + c_1 A^{i+1} b + \cdots + c_n A^{i+n} b = 0$$

ce qui montre bien que la suite est récurrente linéaire. □

Comme A est inversible, on a : $c_0 = det(A) \neq 0$, et l'on peut alors poser

$$x := -\frac{c_1}{c_0} b - \cdots - \frac{c_n}{c_0} A^{n-1} b$$

et dans ce cas :

$$Ax = \frac{-c_1 Ab - \cdots - c_n A^n b}{c_0} = \frac{c_0 b}{c_0} = b$$

x est alors une solution de l'équation $Ax = b$.

Cependant, on peut être plus précis en utilisant le lemme suivant :

Lemme. *L'ensemble des polynômes $P \in K[T]$ vérifiant $P(A)b = 0$ est un idéal de $K[T]$ engendré par un polynôme unitaire m de degré au plus n, appelé* ***polynôme minimal*** *de la suite a.*

Démonstration. Il est facile de voir que cet ensemble est un idéal de $K[T]$. De plus, tout idéal de $K[T]$ est principal, donc ici l'idéal est principal. □

Supposant ce polynôme minimal connu, la solution de $Ax = b$ est donnée par $x = m^*(A)b$, où $m^*(T) = \frac{m(0)-m(T)}{m(0)T}$ par ce qui précède. On a en fait : $x = -\frac{m_1}{m_0} b - \cdots - \frac{m_d}{m_0} A^{d-1} b$ en notant $m(T) = \sum_{i=0}^{d} m_i T^i$.

Connaissant m, on obtiendra donc la solution x en effectuant $d - 1 \leq n$ évaluations de A sur un vecteur, et $O(n^2)$ opérations dans K pour additionner les vecteurs et les multiplier par des scalaires.

On peut écrire un premier algorithme :

Algorithme 1 Résolution d'un système linéaire associé à une matrice carrée inversible

ENTREE : $A \in K^{n \times n}$ inversible, $b \in K^n$

SORTIE : $y = A^{-1} b \in K^n$

1: Déterminer le polynôme minimal m de la suite linéaire récurrente $(A^i b)_{i \in \mathbb{N}}$

2: $m^*(T) \leftarrow \frac{m(0)-m(T)}{m(0)T} \in K[T]$

3: Evaluer $x = m^*(A)b$

4: **renvoyer** x

Le problème de départ sera donc résolu si l'on trouve un moyen de déterminer le polynôme minimal m_a de la suite $a = (A^i b)_{i \in \mathbb{N}}$.

Chapitre 2

Détermination du polynôme minimal

Ici, l'idée sera d'écrire un algorithme qui détermine le polynôme minimal d'une suite récurrente linéaire d'éléments de K, puis d'étendre ce procédé de manière probabiliste à une suite récurrente linéaire de vecteurs de K^n, et de l'appliquer à la suite a.

2.1 Algorithme de Berlekamp-Massey

Dans cette section, a est une suite d'éléments de K, supposée linéairement récurrente. Nous allons étudier une méthode permettant de calculer son polynôme minimal.

Notation. *Etant donné un polynôme $f \in K[T]$, noté $f(T) = \sum_{j=0}^{d} f_j T^j$, on appellera* ***renversement de*** *f et on notera $renv(f)(T) := \sum_{j=0}^{d} f_{d-j} T^j = f_d + \cdots + f_0 T^d$*

On a le résultat :

Lemme. *Notons $h(T) := \sum_{i \in \mathbb{N}} a_i T^i \in K[[T]]$ la série formelle dont les coefficients sont ceux de la suite a. Soit $f \in K[T] - 0$ de degré d, et $r(T) := renv(f)(T)$ son renversement. Alors :*

(i) Sont équivalents :

- *(a) f est un polynôme caractéristique de a*
- *(b) $r.h$ est un polynôme de degré inférieur à d*
- *(c) Il existe $g \in K[T]$ avec $\deg(g) < d$ tel que $h = \frac{g}{r}$*

(ii) Si f est le polynôme minimal de a, alors $d = \max\{1 + \deg(g), \deg(r)\}$ et $pgcd(g, r) = 1$

Démonstration. Montrons la première partie du lemme par triple implication :

(a) $\Rightarrow$ (b)

f est un polynôme caractéristique de a, donc :

$$\forall i \in \mathbb{N}, \sum_{j=0}^{d} f_j a_{i+j} = f_0 a_i + f_1 a_{i+1} + \cdots + f_{d-1} a_{i+d-1} + f_d a_{i+d} = 0$$

.

On s'intéresse à $r.h(T)$:

$$\begin{aligned} r(T).h(T) &= \sum_{i\in\mathbb{N}} r(T).a_i.T^i \\ &= \sum_{i\in\mathbb{N}} a_i.(f_d.T^i + f_{d-1}T^{i+1} + \cdots + f_0.T^{i+d}) \\ &= \sum_{i\in\mathbb{N}} a_i.f_d.T^i + \sum_{i\in\mathbb{N}} a_i.f_{d-1}.T^{i+1} + \cdots + \sum_{i\in\mathbb{N}} a_i.f_1.T^{i+d-1} + \sum_{i\in\mathbb{N}} a_i.f_0.T^{i+d} \end{aligned}$$

en scindant la somme

$$\begin{aligned} &= \sum_{i\in\mathbb{N}} a_{i+d}.f_d.T^{i+d} + \sum_{i=0}^{d-1} a_i.f_d.T^i + \sum_{i\in\mathbb{N}} a_{i+d-1}.f_{d-1}.T^{i+d} + \sum_{i=0}^{d-2} a_i.f_{d-1}.T^{i+1} + \dots \\ &+ \sum_{i\in\mathbb{N}} a_{i+1}.f_1.T^{i+d} + \sum_{i=0}^{0} a_i.f_1.T^{i+d-1} + \sum_{i\in\mathbb{N}} a_i.f_0.T^{i+d} \end{aligned}$$

en séparant chaque somme et en décalant leurs indices

Soit alors $g(T) := \sum_{i=0}^{d-1} a_i.f_d.T^i + \sum_{i=0}^{d-2} a_i.f_{d-1}.T^{i+1} + \cdots + \sum_{i=0}^{0} a_i.f_1.T^{i+d-1}$ qui est un polynôme en T de degré $\deg(g) < d$. On a donc :

$$\begin{aligned} r(T).h(T) &= g(T) + \sum_{i\in\mathbb{N}} (a_{i+d}.f_d + a_{i+d-1}.f_{d-1} + \cdots + a_{i+1}.f_1 + a_i.f_0).T^{d+i} \\ &= g(T) + 0 \\ &\text{car } f \text{ est un générateur de } a \end{aligned}$$

D'où la première implication.

(b) $\Rightarrow$ (c) est immédiat : il suffit de prendre $g(T) := r(T).h(T)$

(c) $\Rightarrow$ (a)

Supposons : $h(T).r(T) = g(T)$ où g est un polynôme de degré $\deg(g) < d$. Alors :

$$\begin{aligned}
g(T) &= (\sum_{i\in\mathbb{N}} a_i.T^i).(f_d + f_{d-1}.T + \cdots + f_0.T^d) \\
&= \sum_{i\in\mathbb{N}} a_i.T^i.(f_d + f_{d-1}.T + \cdots + f_0.T^d) \\
&= \sum_{i\in\mathbb{N}} a_i.f_d.T^i + \sum_{i\in\mathbb{N}} a_i.f_{d-1}.T^{i+1} + \cdots + \sum_{i\in\mathbb{N}} a_i.f_0.T^{i+d} \\
&= \sum_{i=0}^{d-1} a_i.f_d.T^i + \sum_{i\in\mathbb{N}} a_{i+d}.f_d.T^{i+d} + \sum_{i=0}^{d-2} a_i.f_{d-1}.T^{i+1} + \sum_{i\in\mathbb{N}} a_{i+d-1}.f_{d-1}.T^{i+d} + \ldots \\
&+ \sum_{i\in\mathbb{N}} a_i.f_0.T^{i+d} \\
&= \sum_{i\in\mathbb{N}} (a_{i+d}.f_d + a_{i+d-1}.f_{d-1} + \cdots + a_{i+1}.f_1 + a_i.f_0).T^{d+i} + q(T)
\end{aligned}$$

où $q(T)$ désigne l'ensemble des sommes restantes, qui forme un polynôme en T de degré inférieur à d. Par identification terme à terme, on peut conclure :

$$\forall i \in \mathbb{N}, \sum_{j=0}^{d} f_j a_{i+j} = f_0 a_i + f_1 a_{i+1} + \cdots + f_{d-1} a_{i+d-1} + f_d a_{i+d} = 0$$

donc f est un polynôme caractéristique de a.

Intéressons-nous maintenant à la preuve du (ii) :

Remarquons dans un premier temps que $\deg(r) \leq d$, le cas d'égalité se produisant si et seulement si $T \nmid f$. On a alors : $d \geq \max\{1 + \deg(g), \deg(r)\}$. Maintenant, supposons que f est le polynôme minimal de a, et supposons de plus par l'absurde que $d > \max\{1 + \deg(g), \deg(r)\}$. Alors $T \mid f$, et on peut facilement vérifier que l'on a : $r(T) = renv(f(T)/T)$. Par le (i), $f(T)/T$ est alors un polynôme caractéristique de a, ce qui contredit la minimalité de f relativement à a. On conclut : $d = \max\{1 + \deg(g), \deg(r)\}$.

Soit maintenant $u := pgcd(g, r)$. Posons $f^*(T) := f(T)/renv(u)(T)$; c'est un polynôme de degré $d - \deg(u)$, et : $renv(f^*)(T) = renv(f)(T)/renv(renv(u))(T) = r(T)/u(T)$. Comme $r.h = g$, on a : $(r/u).h = (g/u)$ qui est un polynôme en T de degré strictement inférieur à $d - \deg(u)$. D'après le (i), f^* est alors un polynôme caractéristique de a, et la minimalité de d nous donne : $\deg u = 0$, d'où : $u(T) = 1$.

Le lemme est démontré.

□

Supposons que l'ordre de récursion de a est majoré par $l \in \mathbb{N}$. On peut alors résoudre le problème suivant à l'aide de l'algorithme d'Euclide étendu :

$$h \equiv \frac{s}{t} \pmod{T^{2l}},\ T \nmid t(T),\ \deg(s) < l,\ \deg(t) \leq l,\ pgcd(s,t) = 1 \tag{2.1}$$

Le lemme nous donne une solution à ce problème, qui est $(s,t) = (g,r)$. Montrons que 2.1 admet une unique solution à multiplication par un scalaire près :

Démonstration. Supposons que (s_1,t_1) et (s_2,t_2) vérifient tous deux 2.1. Dans ce cas, nous avons : $h \equiv \frac{s_1}{t_1} \equiv \frac{s_2}{t_2} \pmod{T^{2l}}$, d'où : T^{2l} divise $s_1t_2 - s_2t_1$.

Or la condition 2.1 précise aussi que : $\deg(s_1) < l$, $\deg(s_2) < l$, $\deg(t_1) \leq l$, $\deg(t_2) \leq l$, donc : $\deg(s_1t_2 - s_2t_1) < 2l$.

Dans ce cas, $s_1t_2 - s_2t_1 = 0$, ce qui signifie bien : $\exists \lambda \in K^\times | (s_1,t_1) = \lambda(s_2,t_2)$. □

Cette solution peut être déterminée grâce à l'algorithme d'Euclide étendu, en effectuant $O(n^2)$ opérations dans K. Ainsi, en résolvant 2.1, on obtiendra un couple de polynômes dont le second élément sera $r = renv(m)$ où m désigne le polynôme minimal de a.

On utilisera l'algorithme suivant :

Algorithme 2 Détermination du polynôme minimal d'une suite d'éléments de K

ENTREE : Une borne supérieure $l \in \mathbb{N}$ de l'ordre de récursion de a, les $2l$ premiers termes $a_0, \ldots a_{2l-1}$ de la suite a

SORTIE : Le polynôme minimal $m \in K[T]$ de la suite a

1: $h \leftarrow a_{2l-1}T^{2n-1} + \ldots a_1T + a_0$

2: Utiliser l'algorithme d'Euclide étendu pour déterminer $s,t \in K[T]$ tels que $t(0) = 1$ et s,t vérifient 2.1

3: $d \leftarrow \max\{1 + \deg(s), \deg(t)\}$

4: **renvoyer** $renv_d(t)(T)$

Exemple 1. Plaçons-nous ici dans $K = \mathbb{F}_5$. Soit $a = (4, 3, 1, 2, 4, 3, \dots)$ une suite linéaire récurrente d'ordre au plus 3. On pose donc $h(T) = 3T^5 + 4T^4 + 2T^3 + T^2 + 3T + 4$, et on effectue l'algorithme d'Euclide étendu sur h et T^6. On obtient donc :

j	q_{j-1}	r_j	u_j	v_j
0		T^6	1	0
1		$3T^5 + 4T^4 + 2T^3 + T^2 + 3T + 4$	0	1
2	$2T + 4$	4	1	$3T + 1$

Au rang 2, on observe que l'on a : $r_2 = 4$ et $v_2 = 3T + 1$, où :

$h(T) \equiv \frac{4}{3T+1} \pmod{T^6}$, *id est* $s = 4$ et $t = 3T + 1$ vérifient 2.1.

On en tire : $d = max\{1 + 0, 1\} = 1$, et $m(T) = renv_1(t)(T) = T + 3$.

On a donc trouvé le polynôme minimal de la suite récurrente linéaire $a = (4, 3, 1, 2, 4, 3, \dots)$ d'ordre majoré par 3 : $m(T) = T + 3$.

Exemple 2. Plaçons-nous encore dans $K = \mathbb{F}_5$. Soit $a = (4, 0, 2, 1, 4, 0, \dots)$ une suite linéaire récurrente d'ordre au plus 3. On pose donc $h(T) = 4T^4 + T^3 + 2T^2 + 4$, et on effectue l'algorithme d'Euclide étendu sur h et T^6. On obtient donc :

j	q_{j-1}	r_j	u_j	v_j
0		T^6	1	0
1		$4T^4 + T^3 + 2T^2 + 4$	0	1
2	$4T^2 + 4T + 2$	$4T + 2$	1	$T^2 + T + 3$

Au rang 2, on observe que l'on a : $r_2 = 4T + 2$ et $v_2 = T^2 + T + 3$, où :

$h(T) \equiv \frac{4T+2}{T^2+T+3} \pmod{T^6}$, *id est* $s = 4T + 2$ et $t = T^2 + T + 3$ vérifient 2.1.

On en tire : $d = max\{1 + 1, 2\} = 2$, et $m(T) = renv_2(t)(T) = 3T^2 + T + 1$.

On a donc trouvé le polynôme minimal de la suite récurrente linéaire $a = (4, 0, 2, 1, 4, 0, \dots)$ d'ordre majoré par 3 : $m(T) = 3T^2 + T + 1$.

2.2 Algorithme de Wiedemann

Nous avons maintenant un algorithme qui permet de déterminer le polynôme minimal d'une suite récurrente linéaire d'éléments de K. Pour se ramener à ce cas à partir de la suite a, nous allons procéder de manière aléatoire :

Nous allons tirer uniformément au hasard un vecteur $u \in K^n$, et déterminer le polynôme minimal m de la suite récurrente linéaire $({}^tuA^ib)_{i\in\mathbb{N}}$ à l'aide de l'algorithme de Berlekamp-Massey, puis vérifier si m est bien le polynôme minimal de a en calculant $m(A)b$.

Algorithme 3 Détermination du polynôme minimal d'une suite de vecteurs

ENTREE : $A \in K^{n\times n}$, $b \in K^n$

SORTIE : Le polynôme minimal $m \in K[T]$ de la suite $(A^ib)_{i\in\mathbb{N}}$

1: **si** $b = 0$ **alors**
2: **renvoyer** 1
3: **fin si**
4: Tirer $u \in K^n$ uniformément au hasard
5: Calculer ${}^tuA^ib \in K$ pour $0 \leq i < 2n$
6: Déterminer à l'aide de l'algorithme 2 le polynôme minimal (noté m) de $({}^tuA^ib)_{i\in\mathbb{N}}$ avec la borne de récursion n
7: **si** $m(A)b = 0$ **alors**
8: **renvoyer** m
9: **sinon**
10: aller à 5
11: **fin si**

Exemple. Plaçons-nous ici dans $K = \mathbb{F}_5$. Soient alors A la matrice et b le vecteur dans le problème de départ :

$$A = \begin{pmatrix} 2 & 3 & 0 \\ 0 & 1 & 0 \\ 0 & 4 & 2 \end{pmatrix}, b = \begin{pmatrix} 3 \\ 1 \\ 2 \end{pmatrix}.$$

Nous avons alors : $A^0b = \begin{pmatrix} 3 \\ 1 \\ 2 \end{pmatrix}$, $A^1b = \begin{pmatrix} 4 \\ 1 \\ 3 \end{pmatrix}$, $A^2b = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$, $A^3b = \begin{pmatrix} 0 \\ 1 \\ 4 \end{pmatrix}$, $A^4b = \begin{pmatrix} 3 \\ 1 \\ 2 \end{pmatrix}$,

$A^5 b = \begin{pmatrix} 4 \\ 1 \\ 3 \end{pmatrix}$ qui sont les 6 premiers termes de la suite de Krylov associée à A et b. Dans l'algorithme 3, comme ici $b \neq \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$, on tire alors un vecteur $u \in \mathbb{F}_5^3$ uniformément au hasard. Soit donc, par exemple, $u = \begin{pmatrix} 0 \\ 1 \\ 4 \end{pmatrix}$. Dans ce cas, nous avons : $({}^t u A^i b)_{i \in \mathbb{N}} = (4, 3, 1, 2, 4, 3, \dots)$ les 6 premiers termes d'une suite récurrente linéaire d'ordre au plus 3. Le polynôme minimal associé à cette suite de scalaires a été calculé dans l'exemple 1 (section 2.1) à l'aide de l'algorithme 2 : $m(T) = T + 3$. Alors : $m(A) = \begin{pmatrix} 0 & 3 & 0 \\ 0 & 4 & 0 \\ 0 & 4 & 0 \end{pmatrix}$, et $m(A)b = \begin{pmatrix} 3 \\ 4 \\ 4 \end{pmatrix} \neq \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$.

Ce polynôme minimal n'est donc pas celui de la suite de Krylov.

Prenons un autre vecteur u tiré au hasard, mettons : $u = \begin{pmatrix} 3 \\ 4 \\ 3 \end{pmatrix}$.

Dans ce cas, nous avons : $({}^t u A^i b)_{i \in \mathbb{N}} = (4, 0, 2, 1, 4, 0, \dots)$ les 6 premiers termes d'une suite récurrente linéaire. Le polynôme minimal associé à cette suite de scalaires a été calculé dans l'exemple 2 (section 2.1) à l'aide de l'algorithme 2 : $m(T) = 3T^2 + T + 1$.

Alors : $m(A) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$, et $m(A)b = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$

On a donc le bon polynôme minimal de la suite de Krylov, à savoir : $m(T) = 3T^2 + T + 1$.

Calculons maintenant, à l'aide de l'algorithme 1, le vecteur solution y du problème :

$m^*(T) = \frac{m(0) - m(T)}{m(0)T} = \frac{2T^2 + 4T}{T} = 2T + 4$, donc :

$$y = m^*(A)b = (2A + 4I_3)b = \begin{pmatrix} 3 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 3 & 3 \end{pmatrix} \cdot \begin{pmatrix} 3 \\ 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 4 \end{pmatrix}.$$

Conclusion : $y = \begin{pmatrix} 0 \\ 1 \\ 4 \end{pmatrix}$ est solution du problème $Ay = b$, où $A = \begin{pmatrix} 2 & 3 & 0 \\ 0 & 1 & 0 \\ 0 & 4 & 2 \end{pmatrix}$, et $b = \begin{pmatrix} 3 \\ 1 \\ 2 \end{pmatrix}$.

Chapitre 3

Modélisation des algorithmes

Pour programmer les différents algorithmes qui vont suivre, j'ai utilisé le logiciel Maple (version 13.00). Tous les programmes qui suivent sont écrits dans ce langage.

Pour simplifier l'écriture (et les raisonnements), on se place dans le corps fini à p éléments, où p est premier. Notons alors : $K = \mathbb{F}_p$. Dans un souci de clarté, j'ai choisi de remplacer la variable T utilisée plus haut par x dans le code. On se placera donc dans $K[x]$ pour étudier les polynômes.

Ici, on étudie une suite récurrente linéaire dont on connaît un majorant de l'ordre de récursion, noté d. a désignera alors la liste des $2d$ premiers termes de cette suite. Les matrices seront écrites sous forme de listes de listes (en fait la liste des coefficients non nuls associés à leurs coordonnées), pour utiliser au mieux le fait qu'elles sont creuses.

Je suppose acquis les outils suivants (qui sont précisés dans l'annexe) : `List_to_Matrix` (respectivement `Matrix_to_List`) permet de convertir une liste en matrice (respectivement l'inverse), `mult_A_b` (respectivement `mult_A_B`) permet de multiplier une matrice sous forme de liste avec un vecteur (respectivement une autre matrice sous cette forme) modulo p, `poly_A` permet de calculer un polynôme en la matrice A sous forme de liste, et enfin `renv` permet de calculer $renv_k(t)$ pour un polynôme t et un degré k donnés.

3.1 Algorithme de Berlekamp-Massey

Munis de ces outils, nous pouvons programmer l'algorithme de Berlekamp-Massey. Pour cela, écrivons d'abord une version modifiée de l'algorithme d'Euclide étendu. Ce programme prend en entrée un polynôme $h(x)$, le degré d de récursion de la suite a, et le cardinal p du corps K. Il calcule ensuite les étapes successives de l'algorithme d'Euclide étendu, appliqué à la division de x^{2d} par $h(x)$ modulo p. Cependant, à la différence de l'algorithme d'Euclide, il s'arrête lorsque l'on a un couple (s,t) qui vérifie (2.1), donc ici les deux conditions sur les degrés des polynômes.

```
Euclet_modifie := proc(h,d,p) local r1,r2,rs,u1,u2,us,v1,v2,vs,q;
> r1:=x^(2*d); r2:=h; u1:=1; v1:=0; u2:=0; v2:=1;
> while r2<>0 do
>     q:=Quo(r1,r2,x,'r2') mod p;
>     rs:=r1; us:=u1; vs:=v1;
>     r1:=r2; u1:=u2; v1:=v2;
>     u2:=expand(us-q*u2) mod p; v2:=expand(vs-q*v2) mod p;
>     if degree(r2,x)<d then
>       if degree(v2,x)<=d then break fi fi;
> od;
> return r2,v2;
> end:
```

Enfin, voici l'algorithme de Berlekamp-Massey, qui, étant donnés $a = [a_0, a_1, \ldots, a_{2d-1}]$, p et d, définit h comme le polynôme associé à a, puis utilise l'algorithme précédent pour trouver le couple (s,t) vérifiant (2.1). Une fois ce couple déterminé, il calcule le polynôme minimal de a, qui n'est autre que $renv_k(t)(x)$, où $k = \max\{1 + \deg(s), \deg(t)\}$.

```
polymini_scal := proc(d,a,p) local h,i,k,s,t;
> h:=0;
> for i from 1 to 2*d do
>     h := h + a[i]*x^(i-1) od;
> s,t:=Euclet_modifie(h,d,p);
> k:=max(1+degree(s,x),degree(t,x));
> return renv(t,k) mod p;
> end:
```

3.2 L'algorithme de Wiedemann

Passons maintenant à la rédaction de l'algorithme de Wiedemann. Ici, nous nous plaçons dans $K = \mathbb{F}_p$ où p est premier, et l'on considère $A \in K^{n\times n}$, et $b \in K^n$, où $n \in \mathbb{N}^*$, écrite sous forme de liste.

Voici un premier programme qui, étant donnés A et b, calcule de manière probabiliste le polynôme minimal de la suite de Krylov $(A^i.b)_{i\in\mathbb{N}}$ (en s'appuyant sur les algorithmes de la section précédente) :

```
polymini_Krylov := proc(A,b,n,p) local u,L,v,a,f,f_A,r,i;
> if Equal(b,Vector(n,0)) then return 1 fi;
> u:=Vector(n,0);
> r:=rand(0..p-1);
> while true do
>     for i from 1 to n do
>         u[i]:=r() mod p
>     od;
>     L:=NULL; v:=b;
>     for i from 0 to 2*n-1 do
>         L:=L,Multiply(p,u^+,v);
>         v:=mult_A_b(A,v,n,p)
>     od;
>     a:=[L];
>     f:=polymini_scal(n,a,p);
>     f_A:=poly_A(f,A,n,p);
>     if Equal(mult_A_b(f_A,b,n,p),Vector(n,0))
>         then return f
>     fi
> od;
> end:
```

Enfin, l'algorithme de Wiedemann, qui prend en entrée A (sous forme de liste) et b (sous forme vectorielle), et qui s'appuie sur le programme précédent pour calculer la solution $y \in K^n$ au problème $A.y = b$:

```
Wiedemann := proc(A,b,n,p) local m,h,y;
> m:=polymini_Krylov(A,b,n,p);
> h:=-(m-subs(x=0,m))/(subs(x=0,m)*x) mod p;
> y:=mult_A_b(poly_A(h,A,n,p),b,n,p);
> end:
```

3.3 Calcul de complexité

On rappelle que n correspond à la dimension de la matrice carrée A, et du vecteur b.

Notons $c(A)$ le coût d'évaluation de la matrice A appliquée à un vecteur. Dans le cas où A est creuse, avec s coefficients non nuls, on a $c(A) \leq 2s$. Dans le cas général, $c(A) = 2n^2 - n$. De plus, l'évaluation d'un produit de A avec une matrice carrée quelconque de taille n coûtera $n \cdot c(A)$ opérations dans K, ici au plus $2ns$ opérations.

L'algorithme `Euclet_modifie` a le même ordre de complexité que l'algorithme d'Euclide étendu, à savoir une complexité quadratique en le degré maximal des polynômes mis en jeu. Comme ici ces polynômes voient leurs degrés majorés par $2n$, on a donc une complexité de $O(n^2)$. L'algorithme `polymini_scal` n'effectue aucune opération dans le corps, sa complexité algébrique propre est donc nulle ; sa complexité algébrique totale est en $O(n^2)$.

Enfin, lors d'une itération de la boucle `while`, l'algorithme `polymini_Krylov` effectue $O(n^2)$ opérations lors des $2n-1$ produits de vecteurs de taille n, et au plus $2ns$ opérations lors des $2n-1$ produits de matrices avec des vecteurs. Lors du calcul du polynôme en A, celui-ci étant de degré majoré par n, l'opération la plus coûteuse serait le calcul de A^n, qui revient à effectuer $O(n^2 s)$ opérations dans K. De plus, dans chaque itération, l'algorithme `polymini_scal` est appelé une fois, ce qui a un coût en $O(n^2)$. Finalement, si l'on note k le nombre de boucles effectuées par l'algorithme, `polymini_Krylov` effectue au plus $k(2ns+O(n^2)) = 2kn(s+O(n))$ opérations dans le corps de base.

Pour conclure, comme `Wiedemann` a une complexité propre inférieure à $2n(s + O(n))$, la complexité totale est au plus $\mathbf{2(k+1)n(s+O(n))}$. Remarquons que si $s = n^2$ (cas d'une matrice « pleine », dont aucun des coefficients n'est nul), on a alors un coût algébrique de l'ordre de $O(kn^3)$, ce qui n'est donc pas meilleur que l'algorithme de Gauß.

On pourrait aller plus loin en donnant un ordre de grandeur de k par des méthodes probabilistes (liées au cardinal p du corps K), j'ai choisi de ne pas aborder cette question. Elle est traitée dans [2], section 12.4.

Chapitre 4

Références

[1] Joachim von zur Gathen, Jurgen Gerhard, *Modern Computer Algebra*, Cambridge University Press, 2003

[2] Douglas Wiedemann, *Solving Sparse Linear equations over Finite Fields*, IEEE Transactions on Information Theory, 1986

Annexe : Application sur Maple

Ce qui suit est, reproduite telle qu'elle est apparue sur l'écran de la machine, l'exécution des programmes évoqués auparavant dans Maple 13.00, suivie d'exemples illustrant le bon fonctionnement de ces programmes.