INRIA
2004, route des Lucioles
BP 93
06902 Sophia Antipolis

Université de Limoges
123, avenue Albert Thomas
87060 LIMOGES Cedex

Calcul du Genre d'une Courbe Algébrique

Emmanuelle Dottax

Faculté des Sciences de Limoges
D.E.A. de Cryptographie, Codage et Calcul mathématique

Stage effectué sous la responsabilité de Manuel Bronstein

4 Juillet 2001

Ce rapport présente le stage que j'ai effectué au sein du projet CAFE de l'INRIA Sophia-Antipolis, dans le cadre du DEA de Mathématiques de la Faculté des Sciences de Limoges. Je voudrais tout d'abord remercier Manuel Bronstein, qui m'a dirigée, notamment pour le beau sujet de ce stage et pour avoir finalement réussi à faire venir Barry Trager. Ces trois mois auront été très agréables et fort enrichissants, je remercie toutes les personnes qui ont contribué, par l' intérêt qu'ils ont montré pour mon travail et leur sympathie, au bon déroulement de ce stage. Je voudrais remercier en particulier Evelyne Hubert et Stéphane Dalmas pour leur charmant accueil et leur disponibilité. Tout ceci n'aurait pas été possible sans Jacques-Arthur Weil, que je remercie pour son écoute et sa compréhension, ainsi que tous les professeurs de l'Université de Limoges. Un dernier merci à mon collègue usuel Sébastien pour avoir, entre autres, porté mon plateau pendant un mois.

Table des matières

Introduction

Ce rapport de stage présente une méthode originale pour calculer le genre d'une courbe algébrique, basée sur les travaux de B.M. Trager. Celle-ci utilise un résultat classique, mais s'appuie sur l'équation différentielle linéaire satisfaite par une fonction algébrique pour calculer les invariants locaux nécessaires.

La première partie est un bref exposé des résultats théoriques. En premier lieu, nous verrons certaines propriétés locales des courbes algébriques, notamment les expansions des solutions en series de Puiseux, puis la formule de Hurwitz, sur laquelle s'appuie la méthode pour calculer le genre. Nous rappelerons ensuite quelques résultats classiques concernant la résolution d'équations différentielles linéaires à coefficients polynômiaux en termes de séries formelles, notamment la classification des singularités.

Dans la seconde partie, nous verrons comment générer, à partir d'une équation algébrique $P(x, y) = 0$, une équation différentielle linéaire à coefficients polynômiaux satisfaite par toutes les solutions de l'équation algébrique, ainsi que ses différentes propriétés, concernant en particulier l'espace de ses solutions. Nous pourrons alors voir comment calculer le genre à partir d'invariants fournis par cette équation. Cette méthode a été implémentée en Maple. Nous verrons les problèmes que pose la construction de l'équation différentielle linéaire et qui nous ont poussés à calculer ces invariants par un autre moyen.

Dans la troisième partie, on a transporté la méthode sur un système differentiel vérifiée par les solutions algébriques. Celui-ci est plus facile à construire, et en appliquant un algorithme, dû à S. Abramov et M. Bronstein, on pourra calculer des invariants qui nous donnerons le genre. Nous verrons les avantages de l'utilisation de ce système par rapport à la méthode initiale, et les améliorations que l'on peut encore apporter.

En annexe se trouvent les deux programmes Maple qui ont été implémentés durant ce stage, le premier utilise la méthode avec l'équation scalaire, et le second avec le système differentiel.

Chapitre 1

Généralités

Pour voir le lien qui existe entre les solutions d'une équation algébrique et les solutions de l'équation différentielle que nous verrons dans le chapitre suivant, nous aurons besoin de certaines notions et résultats dans les deux domaines. Ils sont brièvement exposés dans les sections qui suivent.

1.1 Sur les courbes algébriques

Nous allons dans cette section présenter rapidement des notions concernant les courbes algébriques qui nous seront utiles pour la compréhension de la formule de B.M.Trager.

1.1.1 Expansion en séries de Puiseux

Dans cette partie, k est un corps de caractéristique zéro, $\overline{k}$ une clôture algébrique de k, $F(X,Y)$ un polynôme à deux variables à coefficients dans k, absolument irréductible dans $k[x,y]$, de degré n (resp. m) en y (resp. en x), avec $n > 0$. F sera souvent considéré comme un plolynôme en y à coefficients dans $k[x]$:

$$F(x,y) = \sum_{i=0}^{n} a_i(x)\, y^i \quad \text{avec} \quad a_i(x) = \sum_{j=0}^{m} a_{ij} x^j \quad \text{et} \quad a_{ij} \in k.$$

Le corps des séries formelles à une variable t à coefficients dans un corps L sera noté $L((t))$ et le sous-anneau formé des éléments à valuation positive en t sera noté $L[[t]]$. Pour tout entier positif p, on choisit $x^{1/p}$ une racine d'ordre p de x dans une clôture algébrique de $k(x)$, de sorte que $(x^{1/p_1 p_2})^{p_2} = x^{1/p_1}$.

Définition 1.1.1 Le *corps des séries de Puiseux,* noté $k(x)^*$, est la réunion de tous les corps $\overline{k}((x^{1/p}))$. Par le théorème de Puiseux, c'est un corps algébriquement clos (cf [Wal50]).

Définition 1.1.2 Les *expansions en série de Puiseux* de $F(x,y)$ sont les racines en y du polynôme F dans $k(x)^*$.

Puisque F est irréductible sur $\overline{k}(x)[y]$ et que $k(x)^*$ est un corps algébriquement clos de caractéristique nulle, F admet n expansions de Puiseux distinctes $\overline{y}_1, \ldots, \overline{y}_n$ avec :

$$\overline{y}_k = \sum_{l=1}^{+\infty} b_{k,l}\, x^{n_{k,l}/e_k} \quad \text{pour } k = 1, \ldots, n.$$

On prendra e_k minimal, c'est-à-dire que e_k et les $n_{k,l}$ sont premiers entre eux. Un tel entier e_k est appelé *indice de ramification* de la série $\overline{y}_k$.

Notons C la courbe d'équation $F(x,y) = 0$ sur $\overline{k}$.

Définition 1.1.3 Une *paramétrisation* de la courbe C est un couple $(\tilde{x}, \tilde{y})$ d'éléments de $\overline{k}[[t]]$ tels que $F(\tilde{x}, \tilde{y}) = 0$, et $\tilde{x}$ et $\tilde{y}$ ne sont pas tous les deux dans $\overline{k}$.

Le *centre* de la paramétrisation est défini comme étant le point (x_0, y_0) tel que x_0 (resp. y_0) est le terme constant de la série $\tilde{x}$ (resp. $\tilde{y}$), c'est un point de la courbe C. Si $(\tilde{x}, \tilde{y})$ est une paramétrisation, et si $z \in \overline{k}[[t]]$ est non nul, de valuation positive en t , alors $(\tilde{x}_1, \tilde{y}_1)$, avec $(\tilde{x}_1(t) = \tilde{x}(z(t))$ et $(\tilde{y}_1(t) = \tilde{y}(z(t))$, est aussi une paramétrisation de la courbe, de même centre. Si la valuation de z est 1, les deux paramétrisations sont dites *équivalentes.* Cette relation est une relation d'équivalence, ainsi une paramétrisation définit une unique classe d'équivalence.

Si $\tilde{x}, \tilde{y} \in \overline{k}[[t^r]]$, où $r > 1$, on peut simplifier les séries en remplaçant t^r par une nouvelle variable t'. On dira qu'une paramétrisation $(\tilde{x}, \tilde{y})$ est *irréductible* s'il n'existe pas d'entier $r > 1$ tel que $\tilde{x}, \tilde{y} \in \overline{k}[[t^r]]$. Le théorème suivant fourni un critère d'irréductibilité :

Théorème 1.1.1 *La paramétrisation*

$$\tilde{x} = t^r, \quad \tilde{y} = \sum_{i \geq 1} b_i t^i, \quad \textit{où } r > 0$$

est irréductible si et seulement si $\{r\} \cup \{i \ / \ b_i \neq 0\}$ *sont premiers entre eux.*

Son utilité vient du théorème suivant :

Théorème 1.1.2 *Dans un système de coordonnées approprié, toute paramétrisation est équivalente à une de la forme :*

$$\tilde{x} = t^r, \quad \tilde{y} = \sum_{i \geq 1} b_i t^i, \quad \text{où } r > 0.$$

On appelle *place* de la courbe C une classe d'équivalence de paramétrisations irréductibles, son centre est le centre commun aux paramétrisations de la classe. On dit qu'une place est *au-dessus* d'un point x_0 de $\overline{k}$ si son centre est (x_0, y_0), pour un certain $y_0 \in \overline{k}$.

Pour chaque expansion en séries de Puiseux de F

$$\overline{y}_k = \sum_{l=1}^{+\infty} b_{k,l}\, x^{n_{k,l}/e_k},$$

posons

$$u_k(t) = t^{e_k} \quad \text{et} \quad v_k(t) = \sum_{l=1}^{+\infty} b_{k,l}\, t^{n_{k,l}}.$$

On a $\overline{y}_k = v_k(x^{1/e_k})$, et le couple (u_k, v_k) est une paramétrisation de la courbe C, irréductible car e_k a été choisi minimal. Ainsi, à chaque expansion en séries de Puiseux correspond une paramétrisation de la courbe. En notant $\nu(\tilde{x})$ la valuation de $\tilde{x}$, on a :

Théorème 1.1.3 *Soit $F(x, y) \in k[x, y]$. A chaque racine $\overline{y} \in k(x)^*$ de $F(x, y) = 0$ telle que $\nu(\overline{y}) > 0$ correspond une unique place de la courbe centrée en l'origine. Réciproquement, à chaque place $(\tilde{x}, \tilde{y})$ de la courbe centrée en l'origine correspondent $\nu(\tilde{x})$ racines de $F(x, y) = 0$ à valuation positive.*

Les indices de ramification que nous avons introduits vont jouer un rôle très important dans la suite, puisque la formule d'Hurwitz donne le genre en fonction de ceux-ci.

1.1.2 La formule de Hurwitz

Nous n'allons pas donner ici de définition précise du genre ni de démonstration de la formule de Hurwitz, mais seulement tenter d'expliquer en quoi le genre est un invariant important. Pour des résultats précis et les démonstrations, on renvoie à [Wal50], [BK86] et [Per95].

Soit k un corps algébriquement clos. Soit C une courbe plane irréductible, elle est définie par un polynôme irréductible $F(x,y) \in k[x,y]$, de degré n en y. L'*algèbre affine* de C est $A = \dfrac{k[x,y]}{(F)}$, et le *corps des fractions rationnelles* est $K = Frac(A)$. On a les extensions de corps $k \subset k(x) \subset K$, où $K : k(x)$ est algébrique de degré n et $k(x) : k$ est transcendante pure.

Si on se donne une fraction rationnelle non constante $\varphi \in k$, $k(\varphi)$ est une extension transcendante pure de k, et il existe un élément primitif $\psi \in K$ sur $k(\varphi)$ tel que $K = k(\varphi, \psi)$. En considérant le polynôme minimal de ψ sur $k(\varphi)$, on construit une nouvelle courbe C' qui a le même corps de fractions rationnelles K que C : C et C' sont dites *birationnellement équivalentes*. Remarquons que les degrés des courbes C et C' ne sont pas égaux en général.

Soit p un point de la courbe, choisissons un représentant d'une place $\mathcal{P}$ centrée en ce point. C'est un homomorphisme :

$$\gamma : A = \frac{k[x,y]}{(F)} \longrightarrow k[[t]]$$

Définitions 1.1.1 On appelle *valuation* en une place $\mathcal{P}$ d'une fraction rationnelle $\varphi \in K$ la valuation dans $k((t)) = k[[t]][1/t]$ de $\gamma\varphi$ où $\gamma : K \to k((t))$ représente $\mathcal{P}$. On la note $v_{\mathcal{P}}(\varphi)$.
On appelle *uniformisante* en $\mathcal{P}$ un élément $u \in K$ tel que $v_{\mathcal{P}}(u) = 1$.

On montre l'existence d'une uniformisante en toute place $\mathcal{P}$, et que dans une équivalence birationnelle entre deux courbes algébriques irréductibles, les places se correspondent bijectivement (cf [Bri97]).

Proposition 1.1.1 *Soit φ une fraction rationnelle non constante. Alors :*

$$\sum_{v_{\mathcal{P}}(\varphi)>0} v_{\mathcal{P}}(\varphi) = [K : k(\varphi)]$$

Toute fraction rationnelle φ sur C a un nombre fini de zéros (les places $\mathcal{P}$ où $v_{\mathcal{P}}(\varphi) > 0$) et un nombre fini de pôles (les places $\mathcal{P}$ où $v_{\mathcal{P}}(\varphi) < 0$) et on a :

$$\sum_{v_{\mathcal{P}}(\varphi)>0} v_{\mathcal{P}}(\varphi) = -\sum_{v_{\mathcal{P}}(\varphi)<0} v_{\mathcal{P}}(\varphi) = [K : k(\varphi)]$$

Définitions 1.1.2 On appelle *diviseur* sur C une combinaison linéaire finie à coefficients dans $\mathbb{Z}$ de places : $D = \sum \vartheta_{\mathcal{P}}\, \mathcal{P}$.
On appelle *degré du diviseur* D : $\deg(D) = \sum \vartheta_{\mathcal{P}}$.

Le diviseur associé à une fraction rationnelle φ est : $(\varphi) = \sum v_{\mathcal{P}}(\varphi)\mathcal{P}$. On appelera *diviseur principal* tout diviseur associé à une fraction rationnelle. Ce sont des diviseurs de degré nul.
On appelle *groupe des classes* de diviseurs le quotient du groupe des diviseurs par le sous-groupe des diviseurs principaux. Le degré est bien defini sur le groupe des classes.
On peut ordonner le groupe des diviseurs par :

$$D = \sum \vartheta_{\mathcal{P}}\, \mathcal{P} \geq 0 \quad \text{si et seulement si} \quad \forall\, \mathcal{P},\ \vartheta_{\mathcal{P}} \geq 0.$$

Nous allons maintenant parler des *formes différentielles* sur C. Soit :

$$\Omega^1(C) = \frac{\{A\mathrm{d}x + B\mathrm{d}y \ / \ (A, B) \in K^2\}}{K.(F_x\mathrm{d}x + F_y\mathrm{d}y)}$$

C'est un K-espace vectoriel de dimension 1, muni d'une dérivation naturelle $K \stackrel{\mathrm{d}}{\to} \Omega^1(C)$.
Soient $\mathcal{P}$ une place de C centrée en p, γ un représentant de $\mathcal{P}$, et $\gamma^* : \Omega^1 \longrightarrow k((t))\mathrm{d}t$ l'application naturelle.

Proposition et définition 1 *Soit $\omega \in \Omega^1$, posons $\gamma^*\omega = h(t)dt$. La valuation de h ne dépend que de la place $\mathcal{P}$, on l'appelle la valuation de ω en $\mathcal{P}$ et on la note $v_{\mathcal{P}}(\omega)$.*

En un point à distance finie $p = (x_0, y_0)$ de C :
- si $F_y(x_0, y_0) \neq 0$, $x - x_0$ est une uniformisante en p et en cette place $\mathcal{P}$ on a $v_{\mathcal{P}}(\mathrm{d}x) = 0$;
- si $F_y(x_0, y_0) = 0$, si $\mathcal{P}$ est une place en ce point, $v_{\mathcal{P}}(x - x_o) = m$ (c'est l'indice de ramification au-dessus de x en $\mathcal{P}$) et $v_{\mathcal{P}}(\mathrm{d}x) = m - 1$.

On appelle diviseur associé à ω : $(\omega) = \sum v_{\mathcal{P}}(\omega)\mathcal{P}$. Pour toute forme $\omega \in \Omega^1$ et toute fraction rationnelle φ on a $(\varphi\omega) = (\varphi) + (\omega)$. Donc (ω) et $(\varphi\omega)$ définissent la même classe de diviseurs, et comme Ω^1 est un K-espace vectoriel de dimension 1, deux formes différentielles non nulles définissent la même classe de diviseurs : la *classe canonique* $\mathcal{K}$. On pose $\deg(\mathcal{K}) = 2g - 2$, g est le *genre* de la courbe. Le fait que $g \in \mathbb{N}$ résulte du théorème de Riemann-Roch.

La formule de Hurwitz donne le genre en fonction du degré de la courbe et des indices de ramification :

Théorème 1.1.4 (Hurwitz) *Soit $F(x,y) \in k[x,y]$ de degré n en y. Notons $(\mathcal{P}_i)_{i\in I}$ les places de la courbe où elle se ramifie au-dessus de x, r_i les indices de ramification correspondants. On a :*

$$2g-2 = -2n + \sum_{i\in I}(r_i - 1)$$

On montre qu'en fait, toute courbe irréductible est birationnellement équivalente à une courbe projective lisse (i.e. sans points singuliers). Si on se place sur le corps des nombres complexes $\mathbb{C}$, le genre a une interprétation topologique : une courbe algébrique lisse projective est aussi une variété différentiable de dimension 1 sur $\mathbb{C}$, donc 2 sur $\mathbb{R}$. C'est une surface compacte et orientable et elle est alors homéomorphe à une spère ($g = 0$), à un tore ($g = 1$) ou à un tore à g trous.

1.2 Sur les solutions formelles d'équations différentielles linéaires

Dans cette partie, nous allons nous intéresser aux équations de la forme :

$$L[y] = a_n(x)y^{(n)} + a_{n-1}y^{(n-1)} + \cdots + a_0 y = 0, \text{ où } a_i(x) \in \mathbb{Q}[x] \tag{1.1}$$

et plus particulièrement à la résolution de ces équations en termes de séries formelles, c'est-à-dire à la recherche de solutions de la forme $y = \sum_{i=0}^{\infty} c_i x^i$.

On sait par le théorème de Cauchy qu'au voisinage d'un point *ordinaire* (i.e. un point x_0 tel que $a_n(x_0) \neq 0$), on a un système fondamental de solutions analytiques de la forme $y = \sum_{i=0}^{\infty} c_i y^i$. Une méthode pour trouver les coefficients de la série consiste à l'injecter dans l'équation différentielle (1.1), à regrouper les puissances de $(x - x_0)$ et annuler les coefficients des puissances successives de $(x - x_0)$. On obtient ainsi un ensemble de relations linéaires entre les coefficients de la série cherchée qui permet de les déterminer.

Considérons maintenant un point *singulier* (i.e. un zéro de a_n), et supposons que c'est l'origine (sinon on s'y ramène par translation). Supposons qu'il existe une solution de la forme $y = x^r(b_0 + b_1 x + \cdots + b_i x^i + \cdots)$ avec $b_0 \neq 0$, sans que forcément r soit un entier positif. A nouveau, injectons cette série dans l'équation (1.1) et annulons le coefficient du terme de plus bas degré en x. Ce coefficient est soit indépendant de r, soit un polynôme $P(r)$ en r de degré inférieur ou égal à l'ordre de l'équation différentielle.

Définition 1.2.1 Le polynôme P(r) est appelé *équation indicielle en 0,* et ses racines les *exposants.* Si P(r) est un polynôme de degre n, on dit que 0 est un point singulier *régulier.*
Sinon, on dit que 0 est un point singulier *irrégulier.*

Pour que la classification des points soit complète, il faut considérer aussi le point à l'infini, ce qui peut se faire de la façon suivante : la transformation $x = z^{-1}$ ramène le point à l'infini sur l'origine, et les critères précédents concernant les singularités régulières et irrégulières s'appliquent directement à la nouvelle équation.

Nous allons voir comment s'exprime l'équation indicielle en 0 de l'équation (1.1). Posons $a_i(x) = \sum_{j=0}^{m_i} b_{i,j} x^j$ où $b_{i,j} \in \mathbb{Q}$ et notons $\lambda^{\underline{i}} = \lambda(\lambda - 1)\cdots(\lambda - i + 1)$. Appliquons l'opérateur différentiel $a_i(x)D^i$ au terme x^λ, on a :

$$b_{i,j}\, x^j D^i(x^\lambda) = b_{i,j}\, \lambda^{\underline{i}}\, x^{\lambda+j-i} \quad \text{pour} \quad j = 1, \ldots, m_i$$

Comme on s'intéresse au coefficient du terme de plus bas degré, il faut chercher les couples (i, j) tels que $j - i$ soit minimal. Si on note $\nu_{(x-\alpha)}(a_i) = \max\{n \in \mathbb{N} \text{ t.q. } (x-\alpha)^n / a_i(x)\}$, on cherche les couples i tels que $\nu_x(a_i) - i$ soit minimal. On a alors, en appelant E_0 l'équation indicielle de (1.1) en 0,

$$E_0(\lambda) = \sum_{i \,\in\, I_0} tc(a_i)\, \lambda^{\underline{i}}$$

où $I_0 = \{i \text{ t.q. } \nu_x(a_i) - i \text{ minimal}\}$ et $tc(a_i)$ désigne le coefficient du terme de plus bas degré de a_i. Plus généralement, si on veut l'équation indicielle en un point quelconque α, on trouve, en injectant de la même façon $(x - \alpha)^\lambda$ dans l'équation différentielle,

$$E_\alpha(\lambda) = \sum_{i \,\in\, I_\alpha} \left[a_i(x)(x-\alpha)^{-\nu_{(x-\alpha)}(a_i)} \right]_{(x=\alpha)} \lambda^{\underline{i}}$$

avec $I_\alpha = \{i \text{ t.q. } \nu_{(x-\alpha)}(a_i) - i \text{ minimal}\}$.

On peut ainsi calculer l'équation indicielle en un point fini quelconque. Pour le point à l'infini, on a une formule analogue

$$E_\infty(\lambda) = \sum_{i \,\in\, I_\infty} lc(a_i)\, (-\lambda)^{\underline{i}}$$

où $I_\infty = \{i \text{ t.q. } \deg(a_i) - i \text{ maximal}\}$ et $lc(a_i)$ est le coefficient du terme de plus haut degré de a_i.

1.2 Sur les solutions formelles d'équations différentielles linéaires

La connaissance des exposants en un point singulier apporte beaucoup d'information sur l'espace des solutions, nous nous contenterons ici de rappeler quelques résultats, renvoyant à [In59] pour plus de précisions. Dans le cas d'une singularité régulière, lorsque l'équation indicielle admet n racines distinctes $r_1, \ldots, r_n$ ne différant pas d'un entier, on a un système fondamental de solutions de la forme $y_1(x) = x^{r_1} \sum_{i=0}^{\infty} a_{1,i} x^i, \ldots, y_n(x) = x^{r_n} \sum_{i=0}^{\infty} a_{n,i} x^i$, où les r_i sont dans $\overline{\mathbb{Q}}$, et les $a_{j,0}$ sont non nuls. Lorsque certaines racines de l'équation indicielle diffèrent d'un entier, il est possible que des termes logarithmiques apparaissent parmi les fonctions qui forment le système fondamental de solutions, notamment dans le cas où on a des racines multiples.

Chapitre 2

Calcul du genre à partir d'une équation différentielle

Dans ce chapitre, nous allons voir comment générer l'èquation diffŕentielle asociée à une courbe algébrique, ses propriétés et le lien entre ses solutions et les solutions de l'équation algébrique. Puis nous en viendrons au calcul du genre à partir de l'équation diffŕentielle.

2.1 De l'équation algébrique à l'équation différentielle

Comme expliqué dans l'introduction, le principe de l'algorithme est de générer une équation différentielle linéaire satisfaite par une fonction algébrique donnée. Nous allons voir comment construire cette équation à partir de celle de la courbe, puis ses particularités, notamment concernant ses solutions.

2.1.1 Construction de l'équation différentielle

Tout d'abord, pour établir l'existence de l'équation différentielle, il suffit de noter que pour un élément quelconque f d'un corps K, extension finie d'un corps $k(x)$, sa dérivée $f' = df/dx$ est encore un élément de K, et par suite toutes les autres. Si n est le degré de l'extension $K : k(x)$, alors il existe au plus n éléments de K linéairement indépendants sur $k(x)$. Puisque f, f', $f'' = d^2f/dx^2$, … , $f^{(n)} = d^nf/dx^n$ sont $n+1$ éléments de K, ils sont linéairement dépendants sur $k(x)$.

La relation qu'ils vérifient

$$a_nf^{(n)} + a_{n-1}f^{(n-1)} + \cdots + a_1f' + a_0f = 0$$

où $a_i \in k(x)$ pour $i = 0, \ldots, n$ sont non tous nuls, est l'équation différentielle linéaire satisfaite par la fonction $f = f(x)$.

On considère donc une équation algébrique

$$P(x, y) = 0 \tag{2.1}$$

où $P(x, y)$ est un polynôme de $k[x, y]$ de degré n en y, avec k un corps de caractéristique nulle. Une équation (2.1) définit une "fonction" $y = y(x)$. On suppose que le polynôme $P(x, y)$ est irréductible, ainsi les polynômes $P(x, y)$ et $P_y = \partial P / \partial y$ sont premiers entre eux. On peut faire appel à une procédure qui calcule les coefficients de Bézout et on obtient $A(x, y)$, $B(x, y)$ dans $k(x)[y]$ tels que

$$A(x, y) \cdot P(x, y) + B(x, y) \cdot P_y(x, y) = 1.$$

Soit $y(x)$ solution de (2.1), il résulte de l'équation précédente que

$$B(x, y(x)) \cdot P_y(x, y(x)) = 1.$$

D'autre part, en dérivant (2.1) par rapport à x on obtient

$$P_x(x, y(x)) + (dy/dx) \cdot P_y(x, y(x)) = 0$$

qui s'écrit

$$\frac{dy}{dx} = -B(x, y(x)) \cdot P_x(x, y(x)).$$

Grâce à l'équation algébrique (2.1), on peut éliminer du membre de droite toutes les puissances de $y(x)$ supérieures à $n - 1$. On a donc

$$\frac{dy}{dx} = R_1(x, y)$$

où R_1 est un polynôme en y de degré au plus $n - 1$ dont les coefficients sont des fractions rationnelles en x. On dérive à nouveau et on a :

$$\frac{d^2y}{dx^2} = \frac{\partial R_1}{\partial x} + \frac{dy}{dx}\,\frac{\partial R_1}{\partial x}\cdot$$

D'où, en remplaçant dy/dx par R_1 et en éliminant les puissances de $y(x)$ comme précédemment, on a

$$\frac{d^2y}{dx^2} = R_2(x, y)$$

où R_2 est du même type que R_1. En répétant ce procédé, on obtient le système d'équations suivant

$$\frac{d^iy}{dx^i} = R_i(x, y),\ i = 0, \ldots, n$$

où les $R_i = R_i(x, y)$ sont $n+1$ polynômes en y de degré en y au plus $n-1$, à coefficients rationnels en x (avec $R_0(x, y) = y$). Ils sont par conséquent linéairement dépendants sur $k(x)$. Pour trouver la relation de dépendance, on peut par exemple construire la matrice, carrée d'ordre n, M dont les colonnes sont les coefficients des $d^i y/dx^i$ pour $i = 0, \ldots, n-1$ dans la base $(1, y, y^2, \cdots, y^{n-1})$, ainsi que le vecteur N représentant, de la même manière, $d^n y/dx^n$. En supposant que la matrice M soit de rang maximal, il est alors possible de résoudre le système $MX = N$, obtenant de cette façon les coefficients cherchés. En effet, on a alors :

$$\sum_{i=1}^{n} X_i \frac{d^i y}{dx^i} = \frac{d^n y}{dx^n} \quad \text{avec } X_i \in k(x).$$

En multipliant cette relation par le dénominateur commun des X_i, on obtient l'équation différentielle linéaire à coefficients polynomiaux en x :

$$\sum_{i=0}^{n} a_i(x) \frac{d^i y}{dx^i} \,. \tag{2.2}$$

Cette équation correspond à l'équation algébrique (2.1) et est indépendante du choix de la solution $y(x)$ puisque nous avons utilisé (2.1) seulement pour éliminer les puissances de y.

On obtient ainsi, à condition que la matrice M soit de rang maximal, une équation différentielle linéaire d'ordre n, vérifiée par toutes les solutions de l'équation algébrique (2.1). Nous allons maintenant voir ce que cette condition signifie pour les solutions de l'équation algébrique.

Pour construire une équation différentielle à coefficients dans $k(x)$ d'ordre minimal satisfaite par toutes les solutions $y = y(x)$ d'une équation algébrique (2.1), supposons que nous avons n solutions distinctes $y_1 = y_1(x), \ldots, y_n = y_n(x)$ de (2.1) (ce qui est le cas quand le polynôme est irréductible). Choisissons alors un sous-ensemble de $(y_1, \ldots, y_n)$ de d éléments linéairement indépendants sur k, avec d maximal. Quitte à renuméroter, soit $(y_1, \ldots, y_d)$ cet ensemble. On a alors, en notant $W(f_1, \ldots, f_n)$ le Wronskien des fonctions $f_1, \ldots, f_n$:

Proposition 2.1.1 *L'expression*

$$L[y] = (-1)^d \frac{W(y, y_1, \ldots, y_d)}{W(y_1, \ldots, y_d)}$$

est un opérateur différentiel linéaire à coefficients dans $k(x)$, tel que toute solution $y = y(x)$ de l'équation algébrique (2.1) est solution de l'équation différentielle $L[y] = 0$, et réciproquement toute solution de l'équation différentielle $L[y] = 0$ est une combinaison linéaire (à coefficients dans k) de solutions de l'équation algébrique (2.1).

Preuve : Remarquons d'abord que l'expression $L[y]$ n'est pas nulle puisque $y_1, \ldots, y_d$ sont linéairement indépendantes sur k. Par construction, toute solution de $L[y] = 0$ est une combinaison linéaire (à coefficients dans k) des $y_1, \ldots, y_d$. Puisque l'espace engendré linéairement par $(y_1, \ldots, y_n)$ est celui engendré par $(y_1, \ldots, y_d)$, toute solution de $L[y] = 0$ est une combinaison linéaire des solutions de (2.1). □

Cette proposition donne une équation différentielle d'ordre minimal à coefficients dans $k(x)$, satisfaite par les n solutions de l'équation algébrique (2.1). Cette équation, $L[y] = 0$, coïncide avec l'équation (2.2) normalisée dans le cas où les n solutions algébriques $y_1, \ldots, y_n$ sont linéairement indépendantes .

Ainsi, la condition sur la matrice M du paragraphe précédent, à savoir qu'elle soit de rang maximal, signifie que les n solutions de l'équation algébrique (2.1) sont linéairement indépendantes sur k. Dans toute la suite, nous supposerons que c'est le cas (on peut toujours s'y ramener par une transformation du type $z = c_0(x) + c_1(x)y + \ldots + c_{n-1}y^{n-1}$, en remplaçant y par z dans l'équation algébrique).

2.1.2 Caractéristiques de l'équation différentielle

D'après ce que nous venons de voir, les n solutions de l'équation algébrique $y_1, \ldots, y_n$ sont solutions de l'équation différentielle que nous avons construite, et elles sont linéairement indépendantes sur k. Par conséquent, elles forment un système fondamental de solutions de l'équation différentielle. Ainsi, toutes les solutions de cette équation différentielle sont algébriques, puisque ce sont des combinaisons linéaires des solutions de l'équation algébrique. On en déduit que cette équation est Fuchsienne et qu'elle admet, en tout point singulier x_0 (forcément régulier donc) un système de solutions de la forme :

$$Y_1 = (x - x_0)^{\lambda_1} \sum_{j=0}^{+\infty} b_{1j}(x - x_0)^j, \ldots, Y_n = (x - x_0)^{\lambda_n} \sum_{j=0}^{+\infty} b_{nj}(x - x_0)^j$$

où $\lambda_1, \ldots, \lambda_n \in \mathbb{Q}$ sont les exposants en x_0 de l'équation différentielle. C'est de l'existence de ces deux systèmes fondamentaux de solutions que découle le calcul du genre par la formule de B.M. Trager, ce que nous allons voir dans le paragraphe suivant.

2.2 Des exposants au genre

Nous allons voir dans cette section comment, par la formule de Trager, on peut calculer le genre d'une courbe à partir des exposants de l'équation différentielle que nous venons de construire, et comment cette méthode a été implémentée en Maple.

2.2.1 La formule de Trager

Notons comme précédemment $y_1, \dots, y_n$ les solutions de l'équation algébrique, et $Y_1, \dots, Y_n$ le système fondamental de solutions de l'équation différentielle. Soit $P(x,y) \in \mathbb{Q}[x,y]$ irréductible, de degré n en y, et L l'opérateur différentiel correspondant. Supposons que l'on travaille à l'origine, comme on l'a vu en 1.1.1, une solution $\overline{y}$ de l'équation algébrique admet une expansion en série de Puiseux :

$$\overline{y} \in \overline{\mathbb{Q}}((x^{1/r}))$$

où r est l'indice de ramification. On a vu qu'en chaque place, il existe une uniformisante, i.e. il existe $z_1 \in \overline{\mathbb{Q}}((x^{1/r}))$ dont la valuation en x est $1/r$, par suite :

$$\forall\, 1 \leq i \leq r, \quad \exists\, z_i \in \overline{\mathbb{Q}}(x,\overline{y}) \;:\; z_i = x^{i/r} + \text{(degrés supérieurs)}$$

Or on a :

$$\begin{aligned} \overline{\mathbb{Q}}(x,\overline{y}) &= \overline{\mathbb{Q}}(x)\_\,[1,\overline{y},\dots,\overline{y}^{n-1}] \\ &= \overline{\mathbb{Q}}(x)\_\,[\overline{y},\overline{y}'\dots,\overline{y}^{(n-1)}] \end{aligned}$$

Donc chaque z_i s'écrit comme combinaison linéaire à coefficients dans $\overline{\mathbb{Q}}(x)$ des Y_i et leurs dérivées. Or ces opérations ne modifient les exposants de x que par des entiers, on en déduit que pour $1 \leq i \leq r$, i/r est la partie fractionnaire d'un des exposants λ_i.

Considérant la formule de Hurwitz (1.1.4) et le fait que $\sum_{0 \leq i \leq r} \frac{i}{r} = \frac{r-1}{2}$, on a :

Théorème 2.2.1 (Trager) *Notons S l'ensemble des points singuliers de l'équation $L[y] = 0$ et $\lambda_{\alpha 1} \dots \lambda_{\alpha n}$ les exposants en $\alpha \in S$. Soient $r_{\alpha i} \geq 0$ tels que $r_{\alpha i} \equiv \lambda_{\alpha i} \bmod 1$. On a alors, en notant g le genre de la courbe $F(x,y) = 0$:*

$$g = 1 - n + \sum_{\alpha \in S} \sum_{i=1}^{n} r_{\alpha i}$$

2.2.2 Programmation

Le programme *genus-sca* Maple qui se trouve en annexe A utilise cette méthode pour calculer le genre d'une courbe, il passe par l'équation scalaire vérifiée par une solution algébrique. Il prend en entrée un polynôme absolument irréductible $P \in \mathbb{Q}[x,y]$ et retourne le genre de la courbe $P(x,y)=0$.

Afin de pouvoir appliquer la formule de Trager, le programme doit en premier lieu calculer les coefficient de l'équation différentielle associée à la courbe. C'est l'étape la plus couteuse, puisque on a vu qu'il fallait tout d'abord calculer les dérivées successives $y', y'', \ldots, y^{(n)}$,et résoudre un système linéaire de taille $n \times n$.

Nous devons ensuite calculer les exposants de l'équation différentielle en chaque zéro du discriminant $d(x) = \mathrm{res}(P, P_y)$. En effet, on a vu que la courbe se ramifiait au-dessus de x en les points (x_0, y_0) tels que $P(x_o, y_0) = P_y(x_0, y_0) = 0$. Rappelons la formule de l'équation indicielle en un point α telle que nous l'avons vue en 1.2. Pour une équation différentielle de la forme

$$a_n(x)y^{(n)} + a_{n-1}y^{(n-1)} + \cdots + a_0 y = 0, \text{ où } a_i(x) \in \mathbb{Q}[x]$$

on a :

$$E_\alpha(\lambda) = \sum_{i \in I_\alpha} \left[a_i(x)(x-\alpha)^{-\nu_{(x-\alpha)}(a_i)} \right]_{(x=\alpha)} \lambda^{\underline{i}}$$

où $I_\alpha = \{i \text{ t.q. } \nu_{(x-\alpha)}(a_i) - i \text{ minimal}\}$.

Soit $q(x) = \prod(x-\alpha_j)$ un diviseur irréductible du discriminant d. Alors l'équation indicielle est la même quelque soit le point α_i considéré : on sait que les exposants sont rationnels, ils restent donc invariants sous l'action de la conjugaison, et on a $\nu_{(x-\alpha_j)}(a_i) = \nu_q(a_i)$ pour tout i et tout j. Il nous suffit donc de calculer l'équation indicielle correspondant à chaque facteur irréductible du discriminant. Pour éviter d'avoir à travailler avec une racine de q, considérons l'expression :

$$E'_q(x,\lambda) = \sum_{i \in I_q} \left[a_i(x)\, q^{-\nu_q(a_i)}(x) (q')^i(x) \bmod q \right] \lambda^{\underline{i}}$$

où $I_q = \{i \text{ t.q. } \nu_q(a_i) - i \text{ minimal}\}$.

Posons

$$q(x) = (x-\alpha)\,\overline{q}(x)\,, \quad \text{avec } \overline{q}(\alpha) \neq 0,$$

on a alors

$$q'(x) = (x-\alpha)\overline{q}\,'(x) + \overline{q}(x).$$

En notant $\nu_i = \nu_q(a_i)$ et μ la valeur minimale des $\nu_q(a_i) - i$, on a, pour $i \in I_q$:

$$\begin{aligned}
\left[\, a_i(x)\,(\overline{q}\,')^i(x)\; q^{-\nu_i}(x)\,\right]_{(x=\alpha)} &= \left[\, a_i(x)\,\overline{q}^i(x-\alpha)^{-\nu_i}\,\overline{q}^{-\nu_i}\,\right]_{(x=\alpha)} \\
&= \overline{q}^{-\mu(\alpha)}\left[\, a_i(x)\,(x-\alpha)^{-\nu_i}\,\right]_{(x=\alpha)}
\end{aligned}$$

On en déduit que :

$$E'_q(\alpha,\lambda) = \overline{q}^{-\mu(\alpha)}(\alpha)\sum_{i\in I_q}\left[\, a_i(x)\,(x-\alpha)^{-\nu_i}\,\right]_{(x=\alpha)}\lambda^{\underline{i}}$$

On retrouve bien l'équation indicielle, à une constante multiplicative non nulle près, telle qu'elle a été définie. Supposons que λ_0 est un exposant de l'équation indicielle en α, zéro du facteur irréductible q, c'est équivalent à :

$$\begin{cases} q(\alpha) = 0 \\ E'_q(\alpha,\lambda_0) = 0 \end{cases}$$

et $E'_q(x,\lambda)$ étant réduit modulo q, il s'écrit :

$$E'_q(x,\lambda) = e_0(\lambda) + e_1(\lambda)x + \ldots + e_{d-1}(\lambda)x^{d-1}$$

en notant d le degré du polynôme q. Ceci nous donne :

$$E'_q(\alpha,\lambda_0) = e_0(\lambda_0) + e_1(\lambda_0)\,\alpha + \ldots + e_{d-1}(\lambda_0)\,\alpha^{d-1}$$

Or, q étant irréductible, les α^i, pour $i = o,\ldots,d-1$ sont linéairement indépendants sur $\mathbb{Q}$, par conséquent :

$$\begin{aligned}
E'_q(\alpha,\lambda_0) = 0 \;&\Leftrightarrow\; e_0(\lambda_0) = e_1(\lambda_0) = \ldots = e_{d-1}(\lambda_0) = 0 \\
&\Leftrightarrow\; \mathrm{pgcd}(e_0,\ldots,e_{n-1})(\lambda_0) = 0 \\
&\Leftrightarrow\; \mathrm{cont}_x(E'_q(x,\lambda_0)) = 0
\end{aligned}$$

où $\mathrm{cont}_x(Q,x)$ désigne le contenu du polynôme Q par rapport à x, i.e. le pgcd de ses coefficients. Ainsi, il suffit de calculer les racines de

$$\mathrm{cont}_x\left(\sum_{i\,\in\, I_q}\left[a_i(x)\,q^{-\nu_q(a_i)}(x)(q')^i(x)\bmod q\right]\lambda^{\underline{i}}\right)$$

pour avoir tous les exposants correspondant au facteur irréductible q du discriminant, puis de les compter avec la multiplicité $\deg(q)$.

Le programme *genus-sca* utilise cette formule pour calculer les exposants. Pour l'équation indicielle à l'infini, on la calcule telle qu'elle a été définie en 1.2.
Notons que ce programme utilise une procédure qui calcule les valuations, et une pour calculer $\lambda^{\underline{i}}$. Les principales étapes sont :

1. calcul des n dérivées de y
2. génération de l'équation différentielle
   $\Rightarrow$ résolution d'un système linéaire $n \times n$
3. calcul des exposants en chaque zéro du discriminant
   calcul des exposants en l'infini
4. calcul du genre par la formule de Trager

Le calcul des exposants n'est pas couteux, puiqu'on travaille sans utiliser de nombre algébrique. Mais le calcul des n dérivées de y, et surtout la résolution du système $n \times n$, handicapent cette méthode. En effet les temps de calculs ne sont pas raisonnables, et si le système à résoudre est trop important, c'est impossible. Pour améliorer la complexité, il est nécessaire de supprimer la résolution du système, c'est ce que nous avons tenté de faire, en utilisant non plus une équation différentielle, mais un système différentiel.

Chapitre 3

En passant par un système différentiel

Pour éviter d'avoir à résoudre un système linéaire d'équations et tenter de rendre la méthode plus efficace, nous avons utilisé un systéme différentiel associé à la courbe et l'algorithme de [AB01] qui va nous permettre de calculer les invariants dont nous avons besoin.

3.1 Le système différentiel

Nous allons voir le système différentiel sur lequel nous nous sommes appuyés. Remarquons tout de suite que la génération du système correspondant a l'équation différentielle que nous avons utilisée précédemment coute aussi cher que celle-ci, puisque il s'écrit :

$$\begin{pmatrix} y' \\ y'' \\ \vdots \\ \vdots \\ y^{(n)} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \dots & \dots & 0 & 1 \\ * & * & * & * & * \end{pmatrix} \begin{pmatrix} y \\ y' \\ \vdots \\ \vdots \\ y^{(n-1)} \end{pmatrix}$$

où les $*$ sont les coefficients de l'équation diffŕentiellle vue précédemment. On a donc considéré un autre système, moins couteux à calculer.

Soit $y = y(x)$ une solution quelconque de l'équation algébrique (2.1) $P(x, y) = 0$. Considérons le vecteur

$$Y = \begin{pmatrix} 1 \\ y \\ y^2 \\ \vdots \\ y^{n-1} \end{pmatrix}$$

On a alors :

$$Y' = \begin{pmatrix} 0 \\ y' \\ 2yy' \\ \vdots \\ (n-1)y^{n-2}y' \end{pmatrix}$$

On calcule, comme nous l'avons fait en 2.1.1, $y'(x) = dy/dx = R_1(x, y)$ où $R_1 \in k(x)[y]$. D'où, en notant Y'_i le i^{me} coefficient de Y', on a $Y'_1 = 0$ et $Y'_i = (i-1)\, y^{i-2} R_1(x, y)$ pour $i = 2, \ldots, n$. On élimine les puissances de y supérieures à n, ainsi les coefficients de Y' peuvent s'exprimer dans la base $(1, y, \ldots, y^{n-1})$. On obtient alors un systeme différentiel du premier ordre :

$$Y' = MY \text{ où } M_{ij} \text{ est le coefficient en } y^{j-1} \text{ de } Y'_i, \text{ pour } i = 1, \ldots, n$$

Ainsi M est une matrice à coefficients dans $k(x)$, et toute solution de l'équation algébrique vérifie ce système.

En appliquant la transformation et l'algorithme que nous allons voir dans la section suivante, nous allons pouvoir récupérer, à peu de choses près, les invariants qui nous intéressent.

3.2 Transformation du système

Le système que nous venons de voir peut s'écrire sous la forme $AY = 0$, où A est une matrice à coefficients dans l'anneau $k[x][d/dx]$.

3.2.1 Le système de récurrences associé à un système différentiel

On va voir comment ramener la résolution d'un système différentiel linéaire du premier ordre en termes de séries formelles à la résolution d'un système de récurrences linéaires satisfaites par les coefficients des dites séries. L'algorithme repose sur la transformation d'un opérateur différentiel

$L : k[x] \to k[x]$ en une récurrence satisfaite par les coefficients d'une série formelle solution de $Ly = 0$.

Soit $\mathcal{B} = (P_n)_{n \geq 0}$ une suite de polynômes de $k[x]$ vérifiant :

- $\deg P_n = n$ pour $n \geq 0$
- $P_n \mid P_m$ pour $0 \leq n < m$

Notons que la première propriété fait de $\mathcal{B}$ une base de $k[x]$. Soit $k[[x]]$ la k-algèbre des séries formelles de la forme $\sum_{n \geq 0} c_n x^n$, où $c_n \in k$. Soit $k^{\mathbb{Z}}$ la k-algèbre des suites infinies à coefficients dans k. Notons κ l'application de $k[[x]]$ dans $k^{\mathbb{Z}}$ qui envoie une série $\sum_{n \geq 0} c_n x^n$ sur la suite de ses coefficients $c = (c_n)_{n \in \mathbb{Z}}$, en posant $c_n = 0$ pour $n < 0$. On dira qu'un endomorphisme $L \in \mathrm{End}_k(k[x])$ est *compatible avec* $\mathcal{B}$ s'il existe $A, B \in \mathbb{N}$ et des éléments $a_{i,n} \in k$,
pour $n \geq 0$ et $-A \leq i \leq B$ tels que :

$$LP_n = \sum_{i=-A}^{B} a_{i,n} P_{n+i} \ , \quad \text{avec } p_k = 0 \text{ pour } k < 0$$

Nous noterons $\mathrm{End}_{\mathcal{B}}(k[x])$ la k-algèbre des endomorphismes compatibles avec $\mathcal{B}$. Par linéarité, on peut étendre un tel endomorphisme à $k[[\mathcal{B}]]$.

Définissons d'autre part le *shift* $\phi : k^{\mathbb{Z}} \to k^{\mathbb{Z}}$ par $\phi(a) = b$ avec $b(n) = a(n+1)$ pour $n \in \mathbb{Z}$. Notons ε la k-algèbre des opérateurs de récurrence de la forme $R = \sum_{i=s}^{r} a_i(n)\, \phi^i$ où $r, s \in \mathbb{Z}$ et $a - i \in k$ pour $s \leq i \leq r$. Si a_s et a_r sont non nuls, on écrit $r = \deg_{\phi}(R)$ et $s = \nu_{\phi}(R)$. Le produit dans ε est défini par $\phi \cdot a = \phi(a) \cdot \phi$. On définit alors $\mathcal{R}_{\mathcal{B}} : \mathrm{End}_{\mathcal{B}}(k[x]) \to \varepsilon$ par :

$$\mathcal{R}_{\mathcal{B}} L = \sum_{i=-B}^{A} a_{-i,n+i} \phi^i \ ,$$

où A, B et $a_{i,n}$ sont ceux définis précédemment.

Théorème 3.2.1 [APR00] $\mathcal{R}_{\mathcal{B}}$ *est un isomorphisme de k-algèbres entre $End_{\mathcal{B}}(k[x])$ et ε.*

Par conséquent, $\mathcal{R}_{\mathcal{B}}$ est déterminé sur le sous-anneau $k[x][q_1, \ldots, q_m]$ de $\mathrm{End}_{\mathcal{B}}(k[x])$ par les images de $q_1, \ldots, q_m$. L'exemple qui nous intéresse est celui de l'anneau différentiel $k[x][d/dx]$, qui est compatible avec la base $\mathcal{P} = (x^n)_{n \geq 0}$ et défini par $\mathcal{R}_{\mathcal{P}}(x) = \phi^{-1}$ et $\mathcal{R}_{\mathcal{P}}(d/dx) = (n+1)\phi$. Ainsi quand L est un opérateur différentiel linéaire, $\mathcal{R}_{\mathcal{B}} L$ est dans $k[n][\phi, \phi^{-1}]$.

Théorème 3.2.2 [APR00] *Pour tout $L \in End_{\mathcal{B}}(k[x])$ et pour tout $y \in k[[\mathcal{B}]]$, $\kappa L y = (\mathcal{R}_{\mathcal{B}} L)(\kappa y)$.*

Par ce théorème, la recherche des solutions de $Ly = f$ dans $k[x]$ est ramenée à la recherche des solutions $z \in k^{\mathbb{Z}}$ à support fini de $(\mathcal{R}_{\mathcal{B}} L)z = \kappa f$. Dans [AB01], ce procédé est généralisé aux systèmes de la forme :

$$\begin{pmatrix} L_{11} & \dots & L_{1m} \\ L_{21} & \dots & L_{2m} \\ \vdots & & \vdots \\ L_{r1} & \dots & L_{rm} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_r \end{pmatrix}$$

où $L_{ij} \in \mathrm{End}_{\mathcal{B}}(k[x])$ pour $1 \leq i \leq r$ et $1 \leq j \leq m$. On applique κ à la i^{me} ligne du système ci-dessus, et en utilisant la linéarité et le théorème 3.2.2, on a :

$$\kappa\left(\sum_{j=1}^{m} L_{ij} y_j\right) = \sum_{j=1}^{m} \kappa(L_{ij} y_i) = \sum_{j=1}^{m} (\mathcal{R}_{\mathcal{B}} L_{ij})(\kappa y_j)$$

On en déduit :

Théorème 3.2.3 [AB01] *Pour toute matrice L à coefficients dans $End_{\mathcal{B}}(k[x])$ et pour tout Y dans $k[[\mathcal{B}]]^m$, $\kappa L y = (\mathcal{R}_{\mathcal{B}} L)(\kappa y)$.*

Ainsi, on est ramené à chercher les solutions à support fini $Z \in (k^{\mathbb{Z}})^m$ de $(\mathcal{R}_{\mathcal{B}})Z = \kappa F$. Ecrivons ce système :

$$\begin{pmatrix} R_{11} & \dots & R_{1m} \\ R_{21} & \dots & R_{2m} \\ \vdots & & \vdots \\ R_{r1} & \dots & R_{rm} \end{pmatrix} \quad Z \quad = \quad G$$

où $G = \kappa F$ et $R_{ij} = \mathcal{R}_{\mathcal{B}} L_{ij}$. En notant M_k la matrice telle que M_{ij} soit le coefficient de ϕ^k dans R_{ij}, le système précédent est alors équivalent à $(\sum_{k=s}^{t} M_k(n)\, \phi^k)Z = G$, ce qui s'écrit :

$$\sum_{k=s}^{t} M_k(n) Z_{n+k} = G(n) \quad \text{pour tout } n \in \mathbb{Z}$$

où $s \leq t \in \mathbb{Z}$ et M_s et M_t sont non nulles. On a le résultat suivant :

Théorème 3.2.4 [AB01] *Soient L une matrice $r \times m$ à coefficients dans $End_{\mathcal{B}}(k[x])$, $F \in k[x]^r$, $y \in k[x]^m$ un vecteur non nul. Posons $N = \max_i(\deg(Y_i))$. Alors, si $LY = F$ on a soit $N \leq s + \max_i(\deg(F_i))$, ou bien $\ker(M_s(N-s)) \neq 0$.*

La base $\mathcal{P} = (x^n)_{n\geq 0}$ peut être étendue à la base $(x_n)_{n\in\mathbb{Z}}$ du corps $k((x))$, de même que la notion de compatibilité et l'application κ introduites plus haut. On a alors un analogue du théorème précédent :

Théorème 3.2.5 [AB01] *Soient $\mathcal{P} = (x^n)_{n\in\mathbb{Z}}$, L une matrice $r \times m$ à coefficients dans $End_{\mathcal{P}}(k[x])$, $F \in k[x]^r$, $y \in k[x]^m$ un vecteur non nul. Soit $N = \min_i(\nu_x(Y_i))$. Si $LY = F$, on a soit $N \geq t + \min_i(\nu_x(F_i))$, ou bien $\ker(M_t(N-t)) \neq 0$.*

Ainsi, lorsque M_s et M_t sont de rang maximal, les théorèmes 3.2.4 et 3.2.5 fournissent des bornes sur les degrés et valuations des solutions du système. La section suivante présente un algorithme permettant de transformer le système en un système équivalent, mais avec M_s (ou M_t) de rang maximal.

3.2.2 Présentation de l'algorithme

Nous allons voir comment transformer le système de façon à ce que la matrice M_s soit de rang maximal.

Soit un système $LY = F$, et le système correspondant $RZ = G$ comme précédemment. On a :

Lemme 3.2.1 *Soit $v \in (k^{\mathbb{Z}})^r$ tel que $v^T R = 0$ et $v^T G \neq 0$. Alors les systèmes $LY = F$ et $RZ = G$ n'ont pas de solutions.*

Soit $D_i(R) = \max_j(\deg(R_{ij}))$ pour $i = 1, \ldots, m$. Notons $\mathrm{rg}(M)$ le rang de la matrice M et supposons que $\mathrm{rg}(M_s) < m$. Soit alors $v \in \ker({M_s}^T)$ non nul et $w = v^T R$. Si $w = 0$ et $v^T G \neq 0$, alors, par le lemme précédent, les systèmes n'ont pas de solutions.

Sinon, posons $I_v = \{i \text{ t.q. } v_i \neq 0\}$ et $d = \max_{i\in I_v}(d_i(R))$ et prenons $i_0 \in I_v$ tel que $d_{i_0}(R) = d$. Si $w = 0$ et $v^T = 0$, soient R' la matrice R à laquelle on a enlevé la ${i_o}^{\text{ième}}$ ligne et G' le vecteur G sans le ${i_o}^{\text{ième}}$ coefficient. Enfin, si $w \neq 0$, soit R' la matrice R avec la ${i_o}^{\text{ième}}$ ligne remplacée par $(\phi^{-1}w_1, \ldots, \phi^{-1}w_m)$ et G' le vecteur G avec la ${i_o}^{\text{ième}}$ entrée remplacée par $\phi^{-1}v^T G$. Dans les deux cas, on remplace le système $RZ = G$ par le système $R'Z = G'$ et on répète le procédé jusqu'à ce que M_s soit de rang m ou ait moins de m lignes.

Par construction, les solutions du premier système $RZ = G$ sont parmi celles de $R'Z = G'$. Il est prouvé dans [AB01] que l'algorithme s'arrête : puisque $v^T M_s = 0$, on a $\nu_\phi(w_j) > s$ pour $j = 1, \ldots, m$, et donc $\nu_\phi(w_j) \geq s$. Par suite $\nu_\phi(R'_{ij}) \geq s$ pour tout i, j. D'autre part on a $\deg_\phi(w_j) \leq d$ pour $j = 1, \ldots, m$, donc $d_{i_0}(R') < d_{i_0}(R)$ et $\sum_i d_i(R') < \sum_i d_i(R)$. Donc, en remplaçant le système $RZ = G$ par $R'Z = G'$, on a soit réduit le nombre de colonnes du système, soit réduit $\sum_i d_i(R)$. Comme $\sum_i d_i(R) > rs$, l'algorithme s'arrête quand M_s est de rang m ou a moins de m lignes, auquel cas on ne peut calculer de borne puisque le système est sous-déterminé.

Comme remarqué dans [AB01], cet algorithme peut être aussi utilisé pour obtenir un système avec une matrice M_t de rang maximal : on prend $v \in \ker({M_t}^T)$, on pose $\nu_i(R) = \min_j(\nu_\phi(R_{ij})$ et on choisit $i_0 \in I_v$ tel que $\nu_{i_0}(R)$ soit minimal. Lorsque $w = v^T R \neq 0$, on remplace la ${i_o}^{\text{ième}}$ ligne de R par $(\phi w_1, \ldots, \phi w_m)$ et le ${i_o}^{\text{ième}}$ coefficient de G par $\phi v^T G$. Dans les autres cas, on prouve que le système n'a pas de solutions ou on supprime la ${i_o}^{\text{ième}}$ ligne du système comme précédemment. L'algorithme s'arrête lorsque M_t est de rang maximal, ou a moins de m lignes.

3.3 Calcul du genre

3.3.1 Calcul des exposants

Le système différentiel que nous avons construit en 3.1 est vérifié par les vecteurs :

$$Y = \begin{pmatrix} 1 \\ y \\ y^2 \\ \vdots \\ y^{n-1} \end{pmatrix}$$

où y est une solution quelconque de l'équation algébrique.
On a une base de l'espace des solutions de la forme :

$$Y_i = x^{\lambda_i} C_i + \ldots$$

où C_i est un vecteur de constantes. De la même façon qu'avec l'équation différentielle, on calcule le genre de la courbe à l'aide des exposants du système, qui sont des rationnels.

Un des problèmes auxquels nous sommes confrontés avec cette méthode est que le système obtenu après avoir appliqué l'algorithme qui rend la matrice M_s (ou M_t) de rang maximal n'est plus le même. Si toute solutions du système initial est solution du nouveau système, la réciproque est fausse. En effet, à chaque pas, on remplace une des lignes de la matrice M par $\phi^{-1} v^T R$, ceci revient à faire :

$$R' \longleftarrow \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & \phi^{-1} & & \\ & & & \ddots & \\ & & & & 1 \end{pmatrix} \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ v_1 & \dots & v_i & \dots & v_n \\ & & & \ddots & \\ & & & & 1 \end{pmatrix} R$$

C'est le terme v_i qui va intervenir dans le déterminant et rajouter des solutions superflues. Il fallait donc récupérer ces facteurs parasites et les éliminer. Au cours de l'algorithme, le programme conserve les v_i de degré non nul en n, et les shifte à chaque fois qu'on intervient sur la même ligne. Finalement, on divise le déterminant de la nouvelle matrice M_s par le produit de ces termes. On a conjecturé que c'était correct.

3.3.2 Programmation

Nous avons donc réussi à nous débarasser de la résolution du sytème linéaire. Maintenant, le programme est divisé de la façon suivante :

1. calcul de la $1^{\text{ère}}$ dérivée de y
2. génération du système différentiel
   calcul des exposants en l'infini
3. calcul des exposants en chaque zéro du discriminant
   $\Rightarrow$ travailler avec des nombres algébriques
4. calcul du genre

On ne calcule plus que la première dérivée de y. On génère le sytème différentiel pour calculer les exposants en l'infini, et en zéro si c'est un point singulier. Pour les autres zéros du discriminant, il faut effectuer une translation, générer à nouveau le système et appliquer l'algorithme. Ceci signifie que l'on doit faire de l'algèbre linàire avec des nombres algébriques. Mais en utilisant l'information contenue dans le discriminant, on peut éviter de calculer les exposants (et donc de générer un nouveau système sur lequel appliquer l'algorithme) pour certains points. Écrivons la décomposition sans facteurs carrés du discriminant D : $D = D_1 D_2^2 \dots D_s^s$. Les zéros de D_1 ne peuvent être des singularités puisque ce sont des pôles simples du discriminant, on connaît donc la contribution du facteur D_1 : $\sum(r-1) = \deg(D_1)$. Les programmes *genus-sca* et *genus-syst* utilisent cette information.

On peut voir sur l'exemple suivant la nette amélioration apportée par l'utilisation du système différentiel :

En passant par l'équation scalaire :

```
> genus_sca_fact(y^6+y^5*x^5+y^3*x^7+y^2*x^9+x^4,y,x);

                    tps_premiere_derivee = 3.110

                  tps_derivees_suivantes = 47.559

                 tps_resolution_systeme = 753.431

                      tps_discriminant = .050

                           facteur = x

   [[34/3, 1], [5, 1], [7, 1], [2/3, 1], [17/3, 1], [28/3, 1]]

                       tps_exposants = 5.000

    [[5, 1], [-5/2, 1], [2/3, 1], [4/3, 1], [-7, 1], [-3, 1]]

                      tps_expo_infini = .189

                            genre = 23

                       tps_total = 857.720
```

En passant par le système :

```
> genus_syst_fact(y^6+y^5*x^5+y^3*x^7+y^2*x^9+x^4,y,x);

                 " tps_premier_systeme" = 6.021
```

" a_l_infini"

nbre_de_tours_inf = 74

tps_algo_infini = 71.559

reste_de_la_div_du_det_par_les_facteurs_superflus = 0

solutions_infini = [[175/3, 1], [53, 1], [101/2, 1], [57, 1], [167/3, 1], [78, 1]]

$$\text{"facteur"} = x^{49} + 4/27\, x^{47} + 2/9\, x^{44} + 64/9\, x^{41} + \frac{275}{36} x^{40} + x^{39} - \frac{229}{27} x^{38} - 10/9\, x^{37} + 3/2\, x^{35} - 16/9\, x^{34} - \frac{625}{18} x^{33} + 54\, x^{32} + \frac{743}{36} x^{30} - 80\, x^{29} - 25/2\, x^{28} + \frac{3125}{108} x^{26} - 255\, x^{25} + 432\, x^{22} - 81\, x^{20} - 128\, x^{19} + 250\, x^{18} - 300\, x^{15} + 324\, x^{10} - 432$$

"facteur" = x

tps_generation_matrices = 0.

tps_algo_en_ce_zero = 3.301

nbre_de_tours = 5

pas_de_sol_superflues

solutions_en_ce_point

[[1, 1], [-1/3, 1], [5/3, 1], [7/3, 1],

```
genre = 23

tps_total = 81.610
```

Ainsi, les temps de calculs sont nettement améliorés si l'on passe par le sytème différentiel plutôt que par l'équation scalaire. Cependant on pourrait encore améliorer le programme en allant plus loin dans l'exploitation du discriminant. On a la décomposition sans facteurs carrés : $D = D_1 D_2^2 \dots D_s^s$. Pour $j > 1$, écrivons : $G_j = pgcd(D_j, res_y(P, P_x))$, $D_j = G_j \overline{D}_j$. Alors les zéros de $\overline{D}_j$ ne peuvent être des singularités, et on sait que :
$\sum(r-1) = j\deg(\overline{D}_j)$.
Une amélioration pourrait sans doute être apportée par l'implémentation dans un langage plus performant que Maple au niveau de l'algèbre linéaire. En effet, l'algorithme que l'on applique au système différentiel doit calculer des noyaux et des déterminants avec un nombre algébrique, et sur ce point Maple n'est pas très efficace.

Conclusion

Comme on a vu, l'utlilisation d'un système différentiel en place d'une équation scalaire couteuse à construire à permis de réduire nettement les temps de calculs.

En effet, pour améliorer la méthode originale de B. Trager. il était nécéssaire de supprimer la résolution du système linéaire nécessaire à la construction de l'équation différentielle. Il fallait donc trouver un objet plus facile à générer, et fournissant des invariants permettant de calculer le genre. Ceci a été fait puisqu'on a réussi à calculer le genre à partir d'un système différentiel beaucoup moins couteux. Nous sommes parvenus à éliminer les solutions parasites introduites au cours de l'algorithme et ainsi à récupérer les informations nécessaires. Cependant, le programme mériterait d'être encore amélioré, afin d'éviter le plus possible de travailler avec des nombres algébriques. En séparant au maximum les points singuliers de ce qui ne le sont pas, on pourrait éviter de calculer les exposants en de nombreux points et le faire uniquement pour les singularités. L'ajout d'un critère pour savoir si l'on doit calculer les exposants à l'infini devrait aussi rendre le programme plus performant.

Annexe A

Le programme *genus-sca*

```
genus_sca_fact := proc(p)

local d, q, i, M, N, Y, F, k, diffeq, B, j, l, nu, delta, mu, S, V, sol,
dg, maxi, somme, g, TPStotal, TPSderivee1, TPSsolve, TPSderivees, X, dc,
TPScalculmu, TPSboucle, TPSinfini, TPSdisc, TPSexpo;

    TPStotal := time();
    if irreduc(p) = false then ERROR(`Le polynome n'est pas irreductible`)
    end if;
    d := degree(p, y);

    ## calcul des derivees et de la relation de dependance
    TPSderivee1 := time();
    gcdex(p, diff(p, y), y, 'u', 'v');
    q[0] := y;
    q[1] := subs(RootOf(p, y) = y,
        evala(subs(y = RootOf(p, y), -diff(p, x)/diff(p, y))));
    TPSderivee1 := time() - TPSderivee1;
    print('tps_premiere_derivee' = TPSderivee1);
    TPSderivees := time();
    for i from 2 to d do q[i] :=
 simplify(rem(diff(q[i - 1], x) + diff(q[i - 1], y)*q[1], p, y))
    end do;
    print('tps_derivees_suivantes' = time() - TPSderivees);
    M := matrix(d, d, (i, j) -> coeff(q[j - 1], y, i - 1));
    if rank(M) < d then ERROR(`la matrice n'est pas de rang maxi`) end if;
    N := vector(d, [seq(coeff(q[d], y, i - 1), i = 1 .. d)]);
    TPSsolve := time();
```

```
X := linsolve(M, N);
TPSsolve := time() - TPSsolve;
print('tps_resolution_systeme' = TPSsolve);
dc := lcm(seq(denom(X[i]), i = 1 .. d));
for j to d do Y[j] := normal(dc*X[j]) end do;
Y[d + 1] := -dc;

## calcul des exposants en les facteurs du discriminant
TPSdisc := time();
B := factors(discrim(p, y));
print('tps_discriminant' = time() - TPSdisc);
for k to nops(B[2]) do
    if B[2][k][2] = 1 then sol[k] := roots(x - 1/2)
    else
        delta[k] := 'infinity';
        for i to d + 1 do
            if Y[i] = 0 then nu[i, k] := 'infinity'
            else nu[i, k] := val(Y[i], B[2][k][1], 'Q[i, k]') - i
            end if;
            if nu[i, k] < delta[k] then delta[k] := nu[i, k] end if
        end do
    end if
end do;
TPSexpo := time();
for k to nops(B[2]) do
    if B[2][k][2] <> 1 then
        S[k] := 0;
        for i to d + 1 do
            if nu[i, k] = delta[k] then S[k] := S[k] + rem(
                Q[i, k]*powmod(diff(B[2][k][1], x), i, B[2][k][1], x),
                B[2][k][1], x)*factd(lambda, i - 1)
            end if
        end do;
        sol[k] := roots(content(S[k], x), lambda);
        print('facteur' = B[2][k][1]);
        print(sol[k])
    end if
end do;
print('tps_exposants' = time() - TPSexpo);

## exposants a l'infini
TPSinfini := time();
for i to d + 1 do dg[i] := degree(Y[i], x) end do;
maxi := max(seq(dg[h] - h, h = 1 .. d + 1));
```

```
    somme := 0;
    for i to d + 1 do
        if dg[i] - i = maxi then
            somme := somme + lcoeff(Y[i])*factd(alpha, i - 1)
        end if
    end do;
    sol[inf] := roots(somme, alpha);
    print(sol[inf]);
    print('tps_expo_infini' = time() - TPSinfini);

    ## computing the genus
    g := 1 - d + sum('(-numer(sol[inf][i][1])) mod denom(sol[inf][i][1])/
        denom(sol[inf][i][1])', 'i' = 1 .. nops(sol[inf]));
    for k to nops(B[2]) do g := g + degree(B[2][k][1])*sum(
        'numer(sol[k][i][1]) mod denom(sol[k][i][1])/denom(sol[k][i][1])',
        'i' = 1 .. nops(sol[k]))
    end do;
    print('genre' = g);
    print('total' = time() - TPStotal)
end proc
```

Annexe B

Le programme *genus-syst*

```
genus_syst := proc(p, y::name, x::name)

local d, K, i, den, M, R, maxi, j, k, F, B, s, degmax, v, Iv, degmaxI, ligne, w,
S, A, t,degmin, degminI, tpstotal, tps, RR, FF, tps_inf, compteurinf, entrop,
tps_ker, denomi, G,determinant, Prod, quotient, sol, facteurs, l, alpha,
tps_matrice, compteur, g, Amat, Bmat,tps_algo;

    tpstotal := time();
    d := degree(p, y);

    ## matrice du systeme
    tps := time();
    K[0] := syst(p, y, x);
    maxi := -'infinity';
    for i to d do for j to d do
            if maxi < degree(numer(K[0][i, j]), x) then maxi :=
degree(numer(K[0][i, j]), x)
            end if;
            if maxi < degree(denom(K[0][i, j]), x) then maxi :=
degree(denom(K[0][i, j]), x)
            end if
        end do
    end do;
    for i to d do den[i] := lcm(seq(denom(K[0][i, j]), j = 1 .. d)) end do;
    Amat := array(sparse, 1 .. d, 1 .. d);
    for i to d do Amat[i, i] := D*den[i] end do;
    Bmat := matrix(d, d, (i, j) -> K[0][i, j]*den[i]);
    RR := simplify(scalarmul(morph(evalm(Amat - Bmat), x), E^maxi));
    for k from 0 to maxi + 1 do
```

```
    for i to d do
           for j to d do FF[k][i, j] := coeff(RR[i, j], E, k) end do
    end do
end do;
for k from 0 to maxi + 1 do FF[k] := matrix(d, d, (i, j) -> FF[k][i, j])
end do;
print(" tps_premier_systeme" = time() - tps);


## algo a l'infini
print(" a_l_infini");
tps_inf := time();
compteurinf := 0;
entrop := [];
M := RR;
B := FF[0];
for i while iszero(B) do B := FF[i] end do;
s := i - 1;
while det(B) = 0 do
    compteurinf := compteurinf + 1;

    ## calcul du max degres en E sur chaque ligne de M
    for i to d do
        degmax[i] := -'infinity';
        for j to d do
            if degmax[i] < degree(M[i, j], E) then
                 degmax[i] := degree(M[i, j], E)
            end if
        end do
    end do;

    ## vecteur du ker
    v := kernel(transpose(B))[1];
    denomi := lcm(seq(denom(v[i]), i = 1 .. d));
    v := simplify(scalarmul(v, denomi));

    ## calcul de la ligne a remplacer
    Iv := NULL;
    for i to d do if v[i] <> 0 then Iv := Iv, i end if end do;
    degmaxI := -'infinity';
    for i to nops([Iv]) do
        if degmaxI < degmax[op(i, [Iv])] then degmaxI := degmax[op(i, [Iv])]
        end if
    end do;
```

```
        for i while degmax[op(i, [Iv])] < degmaxI do  end do;
        ligne := op(i, [Iv]);

        ## garder et shifter les solutions
        if degree(v[ligne], n) <> 0 then entrop := [op(entrop), [v[ligne], ligne]]
        end if;
        for k to nops(entrop) do
            if entrop[k][2] = ligne then
                 entrop[k][1] := subs(n = n - 1, entrop[k][1])
            end if
        end do;

        ## changer la ligne
        w := scalarmul(evalm(`&*`(v, M)), 1/E);
        for i to d do w[i] := subs(n = n - 1, w[i]) end do;
        M := matrix([seq(row(M, i), i = 1 .. ligne - 1), w,
seq(row(M, j), j = ligne + 1 .. d)]);
        for k from 0 to maxi + 1 do
            for i to d do
                 for j to d do G[k][i, j] := coeff(M[i, j], E, k) end do
            end do
        end do;
        for k from 0 to maxi + 1 do
             G[k] := matrix(d, d, (i, j) -> G[k][i, j])
        end do;
        B := G[s]
    end do;
    print('nbre_de_tours_inf' = compteurinf);
    print('tps_algo_infini' = time() - tps_inf);

    ## calcul des exposants a l'infini
    determinant := det(B);
    if nops(entrop) = 0 then
        print('Pas_de_facteurs_superflus'); sol[inf] := roots(determinant)
    else
        Prod := product('entrop[k][1]', 'k' = 1 .. nops(entrop));
        quotient := quo(determinant, Prod, n, 'reste');
        print('reste_de_la_div_du_det_par_les_facteurs_superflus' = reste);
        sol[inf] := roots(quotient)
    end if;
    print('solutions_infini' = sol[inf]);

    ## en les zeros du discriminant
    facteurs := factors(discrim(p, y));
```

```
    for l to nops(facteurs[2]) do
        print("facteur" = facteurs[2][l][1]);
        if facteurs[2][l][2] = 1 then sol[l] := roots(x - 1/2)
        else
            alpha := RootOf(facteurs[2][l][1]);
            tps := time();

            ## generer la nouvelle matrice
            tps_matrice := time();
            if alpha = 0 then
                compteur := 0;
                M := RR;
                A := FF[maxi + 1];
                for i from maxi by -1 while iszero(A) do A := FF[i] end do;
                t := i + 1;
                entrop := []
            else
                for i to d do
                    for j to d do K[l][i, j] := subs(x = x + alpha, K[0][i, j])
                    end do
                end do;
                K[l] := matrix(d, d, (i, j) -> K[l][i, j]);
                maxi := -'infinity';
                for i to d do for j to d do
                        if maxi < degree(numer(K[l][i, j]), x) then
                            maxi := degree(numer(K[l][i, j]), x)
                        end if;
                        if maxi < degree(denom(K[l][i, j]), x) then
                            maxi := degree(denom(K[l][i, j]), x)
                        end if
                    end do
                end do;
                for i to d do den[i] := lcm(seq(denom(K[l][i, j]), j = 1 .. d))
                end do;
                R := simplify(scalarmul(morph(subs(u = D, evalm(diag(
seq(den[i]*u, i = 1 .. d)) - matrix(d, d, (i, j) -> K[l][i, j]*den[i]))), x), E^maxi));
                for k from 0 to maxi + 1 do for i to d do
                        for j to d do F[k][i, j] := coeff(R[i, j], E, k) end do
                    end do
                end do;
                for k from 0 to maxi + 1 do F[k] := matrix(d, d, (i, j) -> F[k][i, j])
                end do;
                compteur := 0;
                M := R;
```

```
    A := F[maxi + 1];
    for i from maxi by -1 while iszero(A) do A := F[i] end do;
    t := i + 1;
    entrop := []
end if;
print('tps_generation_matrices' = time() - tps_matrice);

## algo sur nouvelle matrice
tps_algo := time();
while det(A) = 0 do
    compteur := compteur + 1;

    ## calcul du min des degres en E sur chaque ligne
    for i to d do
        degmin[i] := 'infinity';
        for j to d do
            if ldegree(M[i, j], E) < degmin[i] then
                degmin[i] := ldegree(M[i, j], E)
            end if
        end do
    end do;

    ## vecteur du ker
    v := kernel(transpose(A))[1];
    denomi := lcm(seq(denom(v[i]), i = 1 .. d));
    v := simplify(scalarmul(v, denomi));

    ## calcul de la ligne a remplacer
    Iv := NULL;
    for i to d do if v[i] <> 0 then Iv := Iv, i end if end do;
    degminI := 'infinity';
    for i to nops([Iv]) do
      if degmin[op(i, [Iv])] < degminI then degminI := degmin[op(i, [Iv])]
      end if
    end do;
    for i while degminI < degmin[op(i, [Iv])] do  end do;
    ligne := op(i, [Iv]);

    ## garder et shifter les solutions superflues
    if degree(v[ligne], n) <> 0 then
        entrop := [op(entrop), [v[ligne], ligne]]
    end if;
    for k to nops(entrop) do
        if entrop[k][2] = ligne then
```

```
                    entrop[k][1] := subs(n = n + 1, entrop[k][1])
                end if
            end do;

             ## remplacer la ligne
             w := scalarmul(evalm(`&*`(v, M)), E);
            for i to d do w[i] := subs(n = n + 1, w[i]) end do;
            M := matrix([seq(row(M, i), i = 1 .. ligne - 1), w,
seq(row(M, j), j = ligne + 1 .. d)]);
            for k from 0 to maxi + 1 do for i to d do
                    for j to d do G[k][i, j] := coeff(M[i, j], E, k) end do
                end do
            end do;
            for k from 0 to maxi + 1 do
                 G[k] := matrix(d, d, (i, j) -> G[k][i, j])
            end do;
            A := G[t]
        end do;
        print('tps_algo_en_ce_zero' = time() - tps_algo);
        print('nbre_de_tours' = compteur);

        ## calcul des exposants
        determinant := det(A);
        if nops(entrop) = 0 then
            print('pas_de_sol_superflues'); sol[l] := roots(determinant)
        else
            Prod := product('entrop[k][1]', 'k' = 1 .. nops(entrop));
            quotient := quo(determinant, Prod, n, 'reste');
            print('facteur_superflu' = Prod);
            print('reste_de_la_div_du_det_par_les_non_shiftes' = reste);
            sol[l] := roots(quotient)
        end if;
        print('solutions_en_ce_point');
        print(sol[l])
    end if
end do;

## computing the genus
g := 1 - d + sum(
    '(-numer(sol[inf][i][1])) mod denom(sol[inf][i][1])/denom(sol[inf][i][1])',
    'i' = 1 .. nops(sol[inf]));
for k to nops(facteurs[2]) do g := g + degree(facteurs[2][k][1])*sum(
    'numer(sol[k][i][1]) mod denom(sol[k][i][1])/denom(sol[k][i][1])',
    'i' = 1 .. nops(sol[k]))
```

```
    end do;
    print('genre' = g);
    print('tps_total' = time() - tpstotal)
end proc
```

Bibliographie

[AB01] S.A. Abramov, M. Bronstein, *On Solutions of Linear Functional Systems*, ISSAC'2001.

[APR00] S.A. Abramov, M. Petkovšek, A. Ryabenko, *Special formal series solutions of linear operator equations*, Discrete Mathematics 210, p3-25, 2000.

[Bri97] J. Briançon, *Calculs de résidus sur les courbes algébriques planes*, Cours de DEA, Université de Nice-Sophia Antipolis, 1997.

[Br00] M. Bronstein, *Résolution symbolique des équations différentielles ordinaires linéaires*, Cours de DEA, 2000-2001.

[BK86] E. Brieskorn, H. Knörrer, *Plane Algebraic Curves*, Birkhäuser Verlag, 1986.

[CC86] D.V. Chudnovsky, G.V. Chudnovsky, *On Expansion of Algebraic Functions in Power and Puiseux Series, 1*, Journal of Complexity 2, p 271-294, 1986.

[CC87] D.V. Chudnovsky, G.V. Chudnovsky, *On Expansion of Algebraic Functions in Power and Puiseux Series, 2*, Journal of Complexity 3, p 1-25, 1987.

[Du89] D. Duval, *Rational Puiseux Expansions*, Compositio Mathematica 70, p 119,154, Kluwer Academic Publishers, 1989.

[In59] E.L. Ince, *Ordinary Differential Equations*, Dover, 1959.

[Map1] B.W. Char, K.O. Geddes, G.H. Gonnet, B.L. Leong, M.B. Monagan, S.M. Watt, *Maple V Language Reference Manual*, Springer-Verlag, 1991.

[Map2] B.W. Char, K.O. Geddes, G.H. Gonnet, B.L. Leong, M.B. Monagan, S.M. Watt, *Maple V Library Reference Manual*, Springer-Verlag, 1991.

[Per95] D. Perrin, *Géométrie algébrique, une introduction*, InterÉditions/CNRS Éditions, 1995.

[To87] E. Tournier, *Solutions formelles d'équations différentielles. Le logiciel de calcul formel DESIR. Etude théorique et réalisation*, Thèse, Université de Grenoble, 1987.

[vHo97] . van Hoeij, *Rational Parametrizations of Algebraic Curves using a Canonical Divisor*, J. Symbolic Computation 23, p 209-227, 1997.

[Wal50] R.J. Walker, *Algebraic Curves*, Springer-Verlag, 1950.