LINMA1170 – Numerical Analysis I – Homework 1

Prof. : P. Van Dooren. Teaching assistant: P.-A. Beaufort.

Robin Libois – 37581200
Louis Regout – 86291400
Julien Vaes – 80291100
Levi Tinel –

13 octobre 2016

Echauffement

Il nous est demandé de déterminer la suite de Sturm $(f_0(x),\ f_1(x),\ \ldots,\ f_m(x))$ pour le polynôme $f_0(x) = x^4 + 3x^2 + 2$. Pour déterminer cette dernière, il suffit d'appliquer la définition donnée à la page 2 du syllabus c-à-d :

$$\begin{aligned} f_1(x) &= f_0'(x) && (1)\\ f_i(x) &= q_{i+1}(x)f_{i+1}(x) - f_{i+2}(x), \quad i = 0, \ldots, m-2; && (2)\\ f_{m-1}(x) &= q_m(x)f_m. && (3) \end{aligned}$$

Ceci nous donne donc $f_1(x) = 4x^3 + 6x$. Pour les termes suivants de la suite il est nécessaire de résoudre l'équation 2. Nous allons illustrer la manière de procéder pour déterminer $f_2(x)$, les termes suivants de la suite se calculent de manière identique. D'après 2, il est nécessaire que $f_2(x)$ satifasse :

$$\begin{aligned} f_0(x) &= q_1(x)f_1(x) - f_2(x) && (4)\\ x^4 + 3x^2 + 2 &= (4x^3 + 6x)(a_0 + a_1x) - (r_0 + r_1x + r_2x^2 + r_3x^3) && (5) \end{aligned}$$

Il est a noter que le reste de la division est d'au moins un degré inférieur à celui de $f_0(x)$. D'autre part, dans cette division ci, $q_1(x)$ est de degré maximum égal à 1. Le quotient et le reste s'obtiennent aisément en appliquant la devision euclidienne [1] :

$$\begin{array}{llll} x^4 & +3x^2 & +2 & \big|\ 4x^3 + 6x \\ \cline{4-4} x^4 & +\frac{3}{2}x^2 & & \big|\ \frac{x}{4} \\ \cline{1-2} & \frac{6x^2}{4} & +2 & \big| \end{array}$$

Nous en déduisons donc que $f_2(x) = \frac{-3x^2}{2} - 2$ et $q_1(x) = \frac{x}{4}$. Si on applique la même méthode pour les autres polynômes de la suite de Sturm nous obtenons :

1. la division eulidienne donne $f(x) = q(x)d(x) + r(x)$, or dans la récurrence nous désirons $-r(x)$. Il faut donc inverser le signes du reste.

$f_0(x) = x^4 + 3x^2 + 2$	
$f_1(x) = 4x^3 + 6x$	$q_1(x) = \frac{x}{4}$
$f_2(x) = \frac{-3x^2}{2} - 2$	$q_2(x) = \frac{-8x}{3}$
$f_3(x) = \frac{-2x}{3}$	$q_3(x) = \frac{9x}{4}$
$f_4(x) = 2$	$q_4(x) = \frac{-x}{3}$

	f_0	f_1	f_2	f_3	f_4	$V(x)$
$-\infty$	+	-	-	+	+	2
∞	+	+	-	-	+	2

D'après le théorème de Sturm, $N(\alpha, \beta)$ determine le nombre de racines réelles distinctes du polynôme $f_0(x)$ comprises entre α et β. Dans notre cas $V(-\infty) = 2$ et $V(+\infty) = 2$, nous en concluons que $f_0(x)$ ne possède pas de racines réelles et par conséquent possède deux pairs de racines complexes conjuguées.

Mise en oeuvre

- À l'aide de la fonction *algEuclide*, nous obtenons que $V = 0$ pour des valeurs de $\alpha = -\infty$ et $\beta = +\infty$. Tout comme dans la partie échauffement, nous pouvons conclure que $f_0(x)$ ne possède pas de racines réelles. Or comme les coefficients du polynôme sont réels, cela implique que les racines de $f_0(x)$ sont complexes conjuguées : $r_{1,2} = a \pm bi$ et $r_{3,4} = c \pm di$.
- Il nous est demandé de caratériser les racines du polynômes $f_0(x) = x^{11} - 10x^9 - 7x^8 + 27x^7 + 70x^6 - 20x^5 - 189x^4 + 50x^3 + 140x^2 - 350$. L'algortihme d'Euclide fournit le polynome suivant : $f_m(x) = 10^6 \cdot (-0.9700x^2 + 4.8502)$. De plus nous avons que $N(-\infty, 0) = 1$ et $N(0, +\infty) = 2$. Nous en déduisons donc que $f_0(x)$ possède trois racines réelles distinctes dont deux sont positives et l'une est négative.

  D'autre part, puisque $f_m(x)$ est non constant et qu'il est le pgcd de $f_0(x)$ et $f_1(x)$, toute racine de $f_m(x)$ est au moins une racine double de $f_0(x)$. L'algortihme d'Euclide appliqué à $f_m(x)$ fournit le polynome suivant : $h_m(x) = (-4.8502) \cdot 10^6$. Ainsi comme $h_m(x)$ est un polynôme constant, chaque racine de $f_m(x)$ est simple. De plus, pour $f_m(x)$ nous avons $N(-\infty, 0) = 1$ et $N(0, +\infty) = 1$, ce qui implique que $f_m(x)$ possède une racine positive et une négative. Ainsi $f_0(x)$ possède deux racines réelles doubles : une négative et une autre positive.

  En conclusion les racines réelles de $f_0(x)$ peuvent être caractérisées de la manière suivante :

Racine	Signe	Multiplicité
1)	-	2
2)	+	1
3)	+	2

Réflexion

Soit le polynôme $f_n(x) = a_n x^n + a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_1 x + a_0$ de degré n. Afin de démontrer que le tableau de Routh d'un polynôme peut être généré par l'algorithme d'Euclide : considérons pour cela les polynômes suivants :

$$\begin{aligned} P_n(x) &= a_n x^n - a_{n-2}x^{n-2} + a_{n-4}x^{n-4} - \dots && (6)\\ P_{n-1}(x) &= a_{n-1}x^{n-1} - a_{n-3}x^{n-3} + a_{n-5}x^{n-5} - \dots && (7)\\ & && (8)\end{aligned}$$

A l'aide de l'algorithme d'Euclide, il est possible de trouver un polynôme $P_{n-2}(x) = b_1 x^{n-2} - b_2 x^{n-4} + \dots$ où

$$\begin{aligned} P_n(x) &= Q_{n-1}(x)P_{n-1}(x) - P_{n-2}(x)\\ \Leftrightarrow P_{n-2}(x) &= \frac{a_n}{a_{n-1}}x \cdot P_{n-1}(x) - P_n\\ \Leftrightarrow P_{n-2}(x) &= \frac{a_n x P_{n-1}(x) - a_{n-1}P_n}{a_n} \qquad (9)\end{aligned}$$

Ainsi d'après (9), nous obtenons que $b_1 = \frac{a_{n-2}a_{n-1} - a_n a_{n-3}}{a_{n-1}}$, $b_2 = \frac{a_{n-4}a_{n-1} - a_n a_{n-5}}{a_{n-1}}$, ... On en déduit que $b_i = \frac{a_{n-2i}a_{n-1} - a_n a_{n-2i-1}}{a_{n-1}}$. Obtenons maintenant le polyôme $P_{n-3}(x) = c_1 x^{n-3} - c_3 x^{n-5} + \dots$ comme étant le reste de la division de $P_{n-1}(x)$ avec $P_{n-2}(x)$:

$$\begin{aligned} P_{n-1}(x) &= Q_{n-2}(x)P_{n-2}(x) - P_{n-3}(x)\\ \Leftrightarrow P_{n-3}(x) &= \frac{a_{n-1}}{b_1}x \cdot P_{n-2}(x) - P_{n-1}\\ \Leftrightarrow P_{n-3}(x) &= \frac{a_{n-1} x P_{n-2}(x) - b_1 P_{n-1}}{b_1} \qquad (10)\end{aligned}$$

Ce qui nous donne que $c_i = \frac{b_1 a_{n-2i-1} - b_{i+1}a_{n-1}}{b_1}$. De manière identique le polyôme $P_{n-4}(x) = d_1 x^{n-4} - d_2 x^{n-6} + \dots$ est le reste de la division de $P_{n-2}(x)$ avec $P_{n-3}(x)$:

$$\begin{aligned} P_{n-2}(x) &= Q_{n-3}(x)P_{n-3}(x) - P_{n-4}(x)\\ \Leftrightarrow P_{n-4}(x) &= \frac{b_1}{c_1}x \cdot P_{n-3}(x) - P_{n-2}\\ \Leftrightarrow P_{n-4}(x) &= \frac{b_1 x P_{n-3}(x) - c_1 P_{n-2}}{c_1} \qquad (11)\end{aligned}$$

Ce qui nous donne que $d_i = \frac{c_1 b_{i+1} - b_1 c_{i+1}}{c_1}$. Et ainsi de suite ...

En conclusion, il est possible de definir facilement le tableau de Routh d'un polynôme à coefficients réels à l'aide de l'algorithme d'Euclide. En effet si nous définissions $P_n(x)$ et $P_{n-1}(x)$ comme precédement et si nous considérons la récurrence $P_i(x) = P_{i-1}(x)Q_{i-1}(x) - P_{i-2}(x)$ pour $i = 0, \dots, m-2$, nous pouvons affirmer que l'élément $t_{i,j}$ du tableau de Routh est égal à $(-1)^{j+1}\mathbf{a_j}(P_{n-i+1})$ où $\mathbf{a_j}(P_{n-i+1})$ est le $j^{\text{ième}}$ coefficient non nul du polynôme $P_{n-i+1}(x)$ (en regardant les coefficients du degré le plus élévé au moins élevé). Par exemple, $\mathbf{a_2}(P_n) = -a_{n-2}$ ainsi $t_{1,2} = a_{n-2}$.

Performances

Remarque : pour l'étude des performances, nous avons forcé Matlab a n'utiliser qu'un seul *thread*.

Complexité théorique de notre fonction Dans notre fonction *AlgEuclide*, une boucle *while* effectue la division euclidienne. Dans le pire des cas n divisions euclidiennes devront être calculées, et donc n itérations seront nécesssaires. La division euclidienne de polynômes se fait en $\mathcal{O}(n)$. De plus à l'intérieur de cette même boucle les fonctions *makeTrueZero*[2] et *signeVec*[3] ont toutes les deux une complexité en $O(n)$. Ainsi nous pouvons déduire que la complexité temporelle théorique de AlgEuclide est de $\mathcal{O}(n^2)$. En effet

$$\Rightarrow Complexité = \mathcal{O}(n * (n + n + n)) = \mathcal{O}(3 * n^2) = \mathcal{O}(n^2)$$

Complexité pratique de notre fonction Pour vérifier la complexité théorique trouvée de *AlgEuclide*, nous avons mesuré son temps d'exécution en fonction du degré n de f_0.

Degré de f_0	1000	1500	2000	2500	3000	3500	4000	4500	5000
Temps d'éxécution (ms)	14	24	34	50	65	173	101	132	146

Sur le graphe log-log suivant est représenté le temps necesssaire en fonction du degré du polynôme. Nous savons que si d est la pente de la regression linéaire, alors expérimentalement notre fonction aura une complexité de $\mathcal{O}(n^d)$. En effet si nous supposons que la complexité est en $\mathcal{O}(n^k)$, k est obtenu par la pente de la regression :

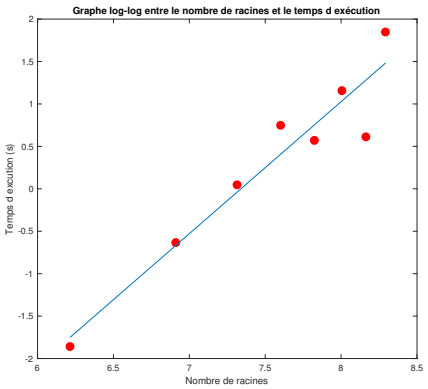
$$pente = \frac{log(Cn_1^k) - log(Cn_0^k)}{log(n_1) - log(n_0)} = k,$$

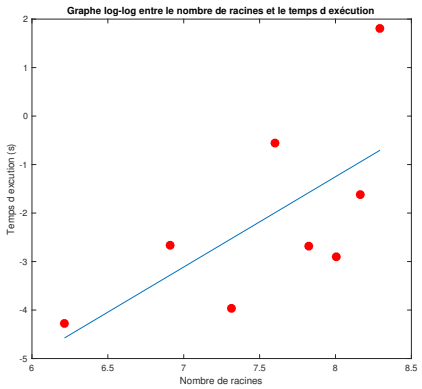
où n_0, n_1 sont des entiers représentant le dégré des polynômes et C est la constante lié à la complexité.

Nous avons distingué deux cas : à gauche les coeffients sont entiers pris aléatoirement entre 0 et 10, à droite les coefficients valent 1. Ce second cas est moins bien conditionné et devrait prendre plus de temps.

2. fonction qui transforme des petites valeurs ($< 10^{-10}$) en 0.

3. Fonction qui transforme un vecteur v de la manière suivante : si $v(i) < 0$ alors $v(i) = -1$, si $v(i) = 0$ alors $v(i) = 0$ et enfin si $v(i) > 0$ alors $v(i) = 1$.



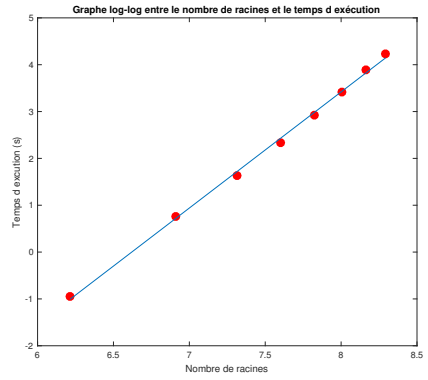


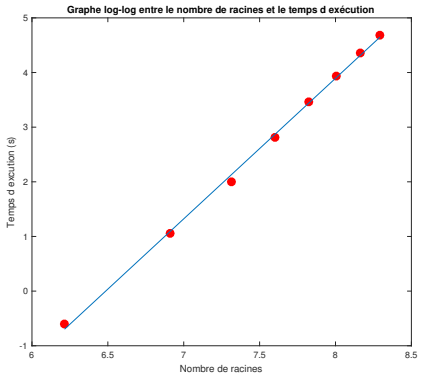
Après expérience nous observons une complexité de $\mathcal{O}(n^{1.55})$ pour le cas de gauche et de $\mathcal{O}(n^{1.86})$ pour le cas de droite. Il faut noté que la complexité obtenue théoriquement est celle du pire cas (n divisions euclidiennes nécessaires) et non celle du cas moyen, ceci peut donc expliquer nos valeurs expérimentales plus faibles.

Complexité de roots.m Pour étudier la complexité de roots, nous avons mesuré son temps d'exécution en fonction du degré n de f_0.

Degré de f_0	500	1000	1500	2000	2500	3000	3500	4000
Temps d'éxécution (s)	0.31	1.4	3.69	6.71	11.2	20.06	30.23	47.17

A nouveau, nous avons distingué deux cas : à gauche les coeffients sont entiers pris aléatoirement entre 0 et 10, à droite les coefficients valent 1. Ce second cas est moins bien conditionné et devrait prendre plus de temps.





Après expérience nous observons une complexité de $\mathcal{O}(n^{2.48})$ pour le cas de gauche et de $\mathcal{O}(n^{2.57})$ pour le cas de droite. Nous constatons également que la valeur théorique de 3 n'est pas atteinte. Nous imputons cette différence au fait que la complexité théorique est celle du pire des cas et que le notre, un polynome avec des coefficients unitaires [4], n'est pas le pire.

4. Un choix de coefficient aléatoire peut paraitre comme un choix plus pertinent, cependant il n'en est rien : comme le spectre des matrices de grandes tailles est en général assez homogène (valeurs propres suffisamment espacées), le problème est bien conditionné et le taux de convergence très rapide.

Stabilité des fonctions

- AlgEuclide.m : AlgEuclide renvoie pour n'importe quelle valeur de ϵ (même 0) le bon nombre de racines distinctes réelles (à savoir 2). Ceci est illustré sur la figure de gauche de l'image ci-dessous.
- root.m. : Le comportement de root.m renvoie les bonnes racines tant que $\epsilon > 10^{-30}$. En deçà de cette valeur, et jusque $\epsilon = 10^{-160}$, les racines restent justes, sauf celle sensée valoir ϵ. L'évolution de cette dernière est représentée sur le graphe de droite de la figure ci-dessous. Idéalement nous aurions du avoir un évolution linéaire, hors nous voyons que lorsque epsilon tend vers 0, il y a un palier ce qui peut être expliqué par la limitation des nombres représentables dans Matlab.