SCILAB *à l'École nationale des ponts et chaussées*

http://cermics.enpc.fr/scilab

Calculs des zéros de polynômes
Licence de mathématiques Nouméa 2004

Jean-Philippe Chancelier
(avec le soutien du Ministère de l'Outre-Mer)

15 mars 2005 (dernière date de mise à jour)

Table des matières

1 Calcul des valeurs d'un polynôme et de ses dérivées par l'algorithme de Horner

Soit $P(x)$ un polynôme de degré n :

$$P(x) \stackrel{\text{def}}{=} \sum_{i=0}^{n} a_{i+1} x^i$$

On cherche ici à évaluer la valeur du polynôme pour une valeur donnée y. Pour ce faire, on peut factoriser le polynôme $P(x)$ sous la forme :

$$P(x) = Q(x)(x - y) + R \tag{1}$$

où $Q(x)$ est un polynôme de degré $n-1$ et R une constante scalaire. Sous cette forme on voit que $P(y) = R$ et que $P'(y) = Q(y)$. Le polynôme $Q(x)$ est bien sur différent du polynôme $P'(x)$ mais leur valeurs coïncident quand x prends la valeur y.

Soit b_i les coefficients du polynôme Q :

$$Q(x) \stackrel{\text{def}}{=} \sum_{i=0}^{n-1} b_{i+1} x^i \tag{2}$$

en développant l'équation (1) on obtient facilement une équation récurrente permettant de calculer les coefficients du polynôme Q :

$$b_i = b_{i+1} y + a_{i+1}, \quad b_n = a_{n+1}, \quad \text{pour} \quad i = n-1, n-2, \ldots, 0 \tag{3}$$

La valeur de R est donnée par b_0. On notera que l'on peut initialiser la récurrence par $b_n = a_{n+1}$ ou quitte à rajouter un point par $b_{n+1} = 0$.

Cet Algorithme s'appelle algorithme d'Horner, il revient à évaluer la valeur du polynôme P en le factorisant sous la forme :

$$P(x) = ((\cdots(a_{n+1} * x + a_n) * x + a_{n-1}) * x + \cdots) * x + a_1$$

Nous avons noté que la dérivée de $P(x)$ au point y s'obtient en évaluant $Q(y)$. Il suffit donc de réappliquer l'algorithme d'Horner au polynôme Q pour évaluer $P'(y)$.

Question 1 *Écrire une fonction* `[Q,R]=Horner(P,xi)` *qui calcule la décomposition* `(Q,R)` *du polynôme P. Tester le code en utilisant les primitives Scilab* `horner` *et* `derivat`.

En fait, on peut calculer directement les valeurs $P(y)$ et $P'(y)$ sans passer par l'intermédiaire de la construction du polynôme Q, c'est à dire sans stocker les coefficients du polynôme Q. On écrit conjointement les deux algorithmes de Horner et pour faciliter la gestion des indices on initialise le deuxième avec $c_n = 0$ plutôt que $c_{n-1} = b_n$. Ainsi les deux vecteurs de coefficients à calculer auront même taille :

$$\begin{aligned} b_i &= b_{i+1} y + a_{i+1}, \quad b_n = a_{n+1}, \quad \text{pour} \quad i = n-1, n-2, \ldots, 0 \\ c_i &= c_{i+1} y + b_{i+1}, \quad c_n = 0, \quad \text{pour} \quad i = n-1, n-2, \ldots, 0 \end{aligned}$$

le calcul du couple (b_0, c_0) permet d'obtenir le couple de valeurs $(P(y), P'(y)$. En généralisant cette idée on peut calculer les p-dérivées successives du polynôme P en un point y. Posons $P_i(x) = P_{i+1}(x)(x-y)+R_i$ avec $P_0(x) = P(x)$. Par un algorithme de Horner on peut calculer en procédant comme dans (3) les valeurs $P_i(y)$ pour $i = 0, \ldots, p-1$. Il est ensuite facile de voir par dérivation composées que :

$$P_0^{(i)}(y) = i! P_i(y), \quad \text{pour} \quad i \geq 1 \tag{4}$$

et d'obtenir ainsi les dérivées successives de P en y.

Question 2 *Écrire une fonction* `[val]=Hornerp(P,xi,p)` *qui calcule $P^{(i)}(xi)$ pour $i = 0, \ldots, p-1$) et renvoit les valeurs calculées dans le vecteur* `val`. *On pourra à nouveau tester le code en utilisant les primitives Scilab* `horner` *et* `derivat`.

1.1 Utilisation de l'algorithme de Horner pour la factorisation

Supposons que l'on ait estimé un zéro ξ du polynôme P, dans la décomposition de $P(x) = Q(x)(x - \xi) + R$ on doit avoir $R = 0$ et la méthode de Horner permet d'éliminer la racine ξ du polynôme $P(x)$ puisqu'elle donne les coefficients du polynôme Q. Mais si ξ est une racine de $P(x)$ on doit aussi avoir $R = b_0 = 0$ dans l'algorithme de Horner.

On peut donc utiliser la récurrence (3) à partir de de $b_n = a_{n+1}$ (méthode forward) ou à partir de $b_0 = 0$ méthode backward. Si l'on cherche à éliminer successivement les racines d'un polynôme en éliminant à chaque fois la plus grande racine, la méthode backward donne une meilleur stabilité numérique.

On peut comparer les deux méthodes avec le polynôme $P(x) \stackrel{\text{def}}{=} \prod_{j=0}^{13}(x-2^{-j})$ en utilisant la fonction `roots` de Scilab pour estimer les valeurs des racines.

Question 3 *Écrire les deux fonctions* `[Q]=deflate_forward(P,xi)` `[Q]=deflate_backward` *et les utiliser pour factoriser le polynôme* $P(x) \stackrel{def}{=} \prod_{j=0}^{13}(x - 2^{-j})$ *en évaluant à chaque fois la plus grande racine d'un polynôme à l'aide de la fonction Scilab* `roots`*.*

1.2 La méthode de Horner backward

L'équation (3) permettant de calculer les coefficients du polynôme Q peut-être vu comme un système dynamique discret. La stabilité de ce système ou de celui obtenu pour le calcul simultané de P et de ses dérivées au point y dépend de $|y|$. Pour $|y| > 1$ le système est instable et on va donc pour des polynômes de grande taille amplifier les erreurs de calcul si l'on utilise (3). Mais on peut remarquer que $P(x)$ s'écrit aussi :

$$P(x) = x^n \tilde{P}(1/x), \quad \text{avec} \quad \tilde{P}(x) \stackrel{\text{def}}{=} \sum_{i=0}^{n} a_{i+1} x^{(n-i)}) \tag{5}$$

Pour une valeur de y telle que $|y| > 1$. on se ramène à un système dynamique stable en utilisant l'algorithme de Horner pour l'évaluation de $\tilde{P}$ et de ses dérivées en $1/y$ et on utilise la relation (5) pour les relier aux valeurs de P et de ses dérivées en y.

Question 4 *Écrire l'algorithme de Horner backward pour estimer la valeur d'un polynôme et de sa dérivée première en un point* y*. Comparer les deux algorithmes.*

2 Recherche des zéros d'un polynôme

On utilise ici la méthode de Newton pour trouver les zéros d'un polynôme. Soit $P(x)$ un polynôme de degré n, on utilise pour trouver les zéros l'algorithme de Newton qui consiste à effectuer les itérations suivantes :

$$x_{k+1} = x_k - P(x_k)/P'(x_k) \tag{6}$$

Il faut pouvoir évaluer la valeur du polynôme et de sa dérivée au point courant x_k et nous avons vu qu'un algorithme d'Horner permet d'évaluer ces deux quantités.

La convergence de l'algorithme de Newton dans ce cas particulier est donnée par le théorème suivant :

Théorème 1 *Soit P un polynôme de degré n dont toutes les racines sont réelles. La méthode de Newton donne une suite strictement décroissante pour $x_0 > \xi_1$ où xi_1 est la plus grande racine de P.*

On trouvera la preuve de la convergence dans [1]. Pour implémenter cet algorithme il faut trouver un majorant de ξ_1. Plusieurs formules de majoration sont données dans la littérature. Par exemple :

$$|\xi_1| < \max\left\{1, \sum_{j=0}^{n-1} \frac{|a_{j+1}|}{|a_{n+1}|}\right\} \quad \text{si} \quad P(x) = \sum_{i=0}^{n} a_{i+1}x^i \tag{7}$$

$$|\xi_1| < \max\left\{\frac{|a_1|}{|a_{n+1}|}, 1 + \frac{|a_2|}{|a_{n+1}|}, \ldots, 1 + \frac{|a_n|}{|a_{n+1}|}\right\} \quad \text{si} \quad P(x) = \sum_{i=0}^{n} a_{i+1}x^i \tag{8}$$

Question 5 *Programmer l'algorithme de Newton et le tester sur le polynôme : $P(x) \stackrel{def}{=} \prod_{j=0}^{13}(x - 2^{-j})$*

En éliminant à chaque fois la plus grande racine trouvée on peut chercher toutes les racines du polynôme $P(x)$. On peut constater dans le code qui suit que la méthode d'élimination choisie (forward ou backward) n'est pas anodine !

Question 6 *Programmer la recherche de toutes les racines du polynôme : $P(x) \stackrel{def}{=} \prod_{j=0}^{13}(x - 2^{-j})$ en utilisant l'algorithme de Newton et les méthodes d'élimination* `[Q]=deflate_forward(P,xi)` *et* `[Q]=deflate_backward`*. Comparer les résultats*

Plutôt que d'essayer de simplifier le polynôme P on peut utiliser la méthode de Maehly (1954) qui consiste à appliquer la méthode de Newton à la fraction rationnelle $P(x)/(x - \xi)$ plutôt que d'éliminer la racine ξ du polynôme $P(x)$. Ainsi si l'on a estimé les j premières racines du polynôme, notées ξ_i, on cherche la $j + 1$-ème racine par l'algorithme de Newton :

$$x_{k+1} = x_k - Q(x_k) \quad \text{avec} \quad Q(x) \stackrel{\text{def}}{=} \frac{P(x)}{P^{(1)}(x) - \sum_{i=1}^{j} \frac{P(x)}{x-\xi_i}} \tag{9}$$

Pour initialiser l'algorithme, on peut utiliser la dernière racine trouvée (à epsilon près pour éviter une division par zéro).

Question 7 *Programmer la recherche de toutes les racines du polynôme : $P(x) \stackrel{def}{=} \prod_{j=0}^{13}(x - 2^{-j})$ en utilisant l'algorithme de Maehly.*

Références

[1] J. Stoer and R. Burlisch. *Introduction to Numerical Analysis.* Springer-Verlag, 1980.