

Projet d'algorithme numérique N 5

[IS104]

Rapport de projet

21/05/2007

Résolution d'un système non linéaire

Application : Détermination des racines d'un polynome par la
méthode de Bairstow

Équipe de développement :

Coordination et Architecture { Mohamed Amine EL AFRIT
Aurélien VASSEUR

développement { Jordane PINTO
Ilhem MAAZAOUI
Adnane BENITTO

Table des matières

1	Introduction	3
2	Méthode de <i>Bairstow</i>	4
2.1	Principe	4
2.2	Mise en oeuvre de la méthode	4
3	Algorithmes	7
4	Rapport d'activités	8
4.1	Coordinateur-rapporteur	8
4.2	Architecte-intégrateur	8
4.3	Développeurs	8
5	Annexe : Code sources	9

Chapitre 1

Introduction

Le but de ce projet consiste à déterminer les racines d'un polynôme à coefficients réels. Ceci en effectuant des divisions euclidiennes par des trinômes (Méthode dite de Bairstow). Dans la pratique, la plupart des problèmes se ramènent à la résolution d'une équation de la forme : $f(x) = 0$.

La résolution de cette équation dépend de la classe à laquelle appartient la fonction f . Si f est un polynôme de degré n , on sait que l'équation possède n racines complexes. Si l'équation est transcendante, elle peut avoir un nombre fini, voire nul, ou infini de racines.

Le problème est alors de trouver la racine dont on sait l'existence et dont, parfois, on connaît une valeur approchée. Les méthodes de résolution sont toujours des méthodes itératives.

Chapitre 2

Méthode de *Bairstow*

2.1 Principe

La méthode de *Bairstow* permet de calculer des racines réelles ou complexes d'une équation polynomiale à coefficients réels. La méthode consiste à extraire (le plus exactement possible ¹) les racines (réelles ou complexes) deux à deux (à la fin, il en reste éventuellement une), jusqu'à épuisement des n racines.

2.2 Mise en oeuvre de la méthode

Soit à trouver les racines de l'équation polynomiale suivante :

$$f(x) = a_0.x^n + a_1.x^{n-1} + a_2.x^{n-2} + a_3.x^{n-3} + \dots + a_n$$

- On effectue la division euclidienne de f par le trinôme $x^2 + p.x + q$, où p et q sont à priori deux nombres quelconques :

$$f(x) = (x^2 + p.x + q).(b_0.x^{n-2} + b_1.x^{n-3} + \dots + b_{n-3}.x + b_{n-2}) + r.x + S.$$

Les coefficients $b_j (j \in [1, n-2[s])$ dépendent de p et q , de même que r et S .

- Si $r = S = 0$, alors $f(x) = 0$ permet de donner $x^2 + p.x + q = 0$ (donc deux racines de $f(x)$ déjà).
- Si r et/ou S ne sont pas nuls, la méthode de *Bairstow* va consister à déterminer par approximations successives les valeurs de p et q qui annulent r et S :

$$\begin{cases} r(p,q)=0 \\ S(p,q)=0 \end{cases}$$

Ce système peut être non linéaire. On le supposera linéaire au voisinage de chaque couple (p, q) fixé.

Pour cela, on se donne 2 valeurs p_0 et q_0 arbitraires. On calcule alors successivement Δp et Δq de telle sorte que :

$$\begin{cases} r(p + \Delta p, q + \Delta q) = 0 \\ S(p + \Delta p, q + \Delta q) = 0 \end{cases}$$

¹car il s'agit d'une méthode de calcul numérique

Soit au premier ordre en Δp et Δq

$$\begin{cases} r(p_0, q_0) + \left(\frac{\partial r}{\partial p}\right)_0 \cdot \Delta p + \left(\frac{\partial r}{\partial q}\right)_0 \cdot \Delta q = 0 \\ S(p_0, q_0) + \left(\frac{\partial S}{\partial p}\right)_0 \cdot \Delta p + \left(\frac{\partial S}{\partial q}\right)_0 \cdot \Delta q = 0 \end{cases}$$

si l'on pose :

$$\left(\frac{\partial r}{\partial p}\right)_0 \left(\frac{\partial S}{\partial q}\right)_0 - \left(\frac{\partial S}{\partial p}\right)_0 \left(\frac{\partial r}{\partial q}\right)_0 = \delta$$

et

$$\begin{cases} S_0 \cdot \left(\frac{\partial r}{\partial q}\right)_0 - r_0 \cdot \left(\frac{\partial S}{\partial q}\right)_0 = P \\ r_0 \cdot \left(\frac{\partial S}{\partial q}\right)_0 - S_0 \cdot \left(\frac{\partial r}{\partial q}\right)_0 = Q \end{cases}$$

La solution du système précédent est (*Cramer*) :

$$\begin{cases} \Delta p = \frac{P}{\delta} \\ \Delta q = \frac{Q}{\delta} \end{cases}$$

Les expressions qui entrent dans le calcul de δ , P et Q vont être évoluées par étapes. Les coefficients b_i sont liés aux coefficients a_i du polynôme initial par l'intermédiaire des relations suivantes :

$$(I) \begin{cases} b_0 = a_0 \\ b_1 = a_1 - p.b_0 \\ b_2 = a_2 - p.b_1 - q.b_0 \\ b_3 = a_3 - p.b_2 - q.b_1 \\ \vdots \\ b_{n-2} = a_{n-2} - p.b_{n-3} - q.b_{n-4} \end{cases}$$

Relations auxquelles on peut ajouter :

$$\begin{cases} b_{n-1} = a_{n-1} - p.b_{n-2} - q.b_{n-3} \\ b_n = a_n - p.b_{n-1} - q.b_{n-2} \end{cases}$$

en ayant défini b_{n-1} et b_n par : $b_{n-1} = r$ et $b_n = S - p.r$

Ce tableau (I) permet de calculer les b_i , r et S en fonction de p et q .

Posons maintenant :

$$\frac{\partial b_k}{\partial p} = -c_k, k \in [0, n-1] \text{ et } \frac{\partial b_n}{\partial p} = -c_{n-1} - b_{n-1}$$

$$(II) \left\{ \begin{array}{l} \frac{\partial b_0}{\partial p} = 0 \\ \frac{\partial b_1}{\partial p} = b_0 - p \cdot \frac{\partial b_0}{\partial p} \\ \frac{\partial b_2}{\partial p} = b_1 - p \cdot \frac{\partial b_1}{\partial p} - q \cdot \frac{\partial b_0}{\partial p} \\ \frac{\partial b_3}{\partial p} = b_2 - p \cdot \frac{\partial b_2}{\partial p} - q \cdot \frac{\partial b_1}{\partial p} \\ \vdots \\ \frac{\partial b_n}{\partial p} = b_{n-1} - p \cdot \frac{\partial b_{n-1}}{\partial p} - q \cdot \frac{\partial b_{n-2}}{\partial p} \end{array} \right. \Rightarrow \left\{ \begin{array}{l} c_0 = b_0 \\ c_1 = b_1 - p \cdot c_0 \\ c_2 = b_2 - p \cdot c_1 - q \cdot c_0 \\ c_3 = b_3 - p \cdot c_2 - q \cdot c_1 \\ \vdots \\ c_{n-2} = b_{n-2} - p \cdot c_{n-3} - q \cdot c_{n-4} \\ c_{n-1} = b_{n-1} - p \cdot c_{n-2} - q \cdot c_{n-3} \end{array} \right.$$

Ce tableau (II) permet de calculer les C_i $i \in [0, n-1]$

On pose de la même manière $\frac{\partial b_k}{\partial q} = -c'_{k-2}$ $k \in]2, n]$. On trouve alors que :

$$c_k = c'_k \quad \forall k \in [0, n-2] \Rightarrow \frac{\partial b_{n-1}}{\partial q} = -c_{n-3} \text{ et } \frac{\partial b_n}{\partial q} = -c_{n-2}$$

Par conséquent, les quantités δ , P et Q cherchés sont :

$$\left\{ \begin{array}{l} \delta = c_{n-2}^2 - c_{n-1} \cdot c_{n-3} \\ P = b_{n-1} \cdot c_{n-2} - b_n \cdot c_{n-3} \\ Q = b_n \cdot c_{n-2} - b_{n-1} \cdot c_{n-1} \end{array} \right.$$

Chapitre 3

Algorithmes

Mise en oeuvre de la méthode

Le calcul du premier trinôme $x^2 + p.x + q$ s'obtient à l'issue de l'algorithme suivant :

– On fixe arbitrairement ¹ deux valeurs p_0 et q_0 .

– On calcule alors les deux valeurs :

$$\begin{cases} r(p + \Delta p, q + \Delta q) = 0 \\ S(p + \Delta p, q + \Delta q) = 0 \end{cases}$$

On construit le tableau (I) et (II) et on calcule δ , P et Q et on en déduit :

$$\Delta p = \frac{P}{\delta} \text{ et } \Delta q = \frac{Q}{\delta}$$

– Si $|r| < \epsilon$ et $|S| < \epsilon$, alors les racines de $x^2 + p.x + q$ sont les racines du polynôme initial.

– Sinon, on remonte en *deuxième étape* en reportant les valeurs p_1 et q_1 , et ainsi de suite.

– Les deux premières racines ayant été extraites, on applique de nouveau la méthode de *Bairstow* au polynôme quotient.

¹Le calcul de la première solution doit se faire avec le maximum de précision. C'est pour cela q'on va choisir au hasard 100 nombres complexes z_i et choisir z_k tel que $f(z_k)$ soit minimal. On prendra par la suite $p_0 = |z_k|^2$ et $q_0 = -2.Re(z_k)$

Chapitre 4

Rapport d'activités

4.1 Coordinateur-rapporteur

« ...mes commentaires... »

Mohamed Amine EL AFRIT

4.2 Architecte-intégrateur

« ...des commentaires... »

Aurélien VASSEUR

4.3 Développeurs

« ...des commentaires... »

Jordane PINTO

« ...des commentaires... »

Ilhem MAAZAOUI

« ...des commentaires »

Adnane BENITTO

Chapitre 5

Annexe : Code sources

22 mai 07 13:43

sources.sci

Page 1/2

```

//fonction second degre

function [x1,x2] = resolution (coef)

5   alpha=coef(1);
   betta=coef(2);
   delta= betta*betta - 4*alpha;

   if (delta=0) then
10      r1= -betta/2;
      r2=r1;
   else
      // sqrtDelta=sqrt(delta);
      r1= (-betta-sqrt(delta))/2;
15      r2= (-betta+sqrt(delta))/2;
   end
   x1=r1,
   x2=r2,
   return [x1,x2];
20 endfunction

//test
//coef=[3 2];
//[x1 x2]=resolution(coef),
25

//fonction pour calculer delta_p et delta_q

function [q1,p1] = suivant(A, B, C, q0,p0)

30   //determination de delta_p et delta_q
   petit_delta=C(n-2)*C(n-2)-C(n-1)*C(n-3);
   P=B(n-1)*C(n-2) - B(n)*C(n-3);
   Q=B(n)*C(n-2) - B(n-1)*C(n-1);
   delta_p = P/petit_delta;
35   delta_q = Q/petit_delta;

   //calcul du suivant
   p1=p0 + delta_p;
   q1=q0 + delta_q;
40 endfunction

//test
//A=[1, 2, 3, 4],
//B=[5,6,7,8],
45 //C=[2,1,3],
   //p=1, q=2,
   //J = suivant(A, B, C, p, q),

function [q1,p1] = approximation(pallier, A, B, C, q0, p0)
50   F = suivant(A, B, C, q0, p0)
   if (abs(A(n)-F(1)*B(n-2)) < pallier & abs(A(n-1)-F(2)*B(n-2)-F(1)*B(n-3))) the
n      return F
   else suivant(A, B, C, F(1), F(2))
   end
55 endfunction

function M = total(A, q0, p0, pallier)
60   if (length(A) > 2) then
      n=length(A);

```

22 mai 07 13:43

sources.sci

Page 2/2

```

        B=zeros(n);
        C=zeros(n);
65
        //remplissage du premier vecteur
        B(1)=A(1);
        B(2)=A(2)-p0*B(1);
        for i=3:n do
70            B(i)=A(i)-p0*B(i-1)-q0*B(i-2);
        end

        //remplissage du second vecteur
        C(1)=B(1);
75        C(2)=B(2)- p0*C(1);
        for i=3:n do
            C(i)=B(i) - p0*C(i-1) - q0*C(i-2);
        end

80        M=[M,resolution(approximation(pallier, A, B, C, q0, p0))];
        total(B, q0, p0, pallier);
        else if (length (A) = 2) then
            M=[M, resolution([A(2),A(1)])];
            return M;
85            else M=[M, [-A(1),1 000 000]];
            return M;
        end
    end
endfunction
90
//total([1, 2, 3, 4], 10, 12, 0.1);

```