PROJET DE MATH 3A

WyderAnquetin

ESIREM-DIJON

I. Table des matières

II. Introduction

La résolution de polynôme est un problème important dans beaucoup de domaines, comme la gestion d'objet de dimension n. Problème connus, la détection de collision entre 2 sphères.

Dans ce document nous allons étudier les approches usuelles de résolution de polynôme des dégrée 3 à 5. Ainsi qu'une autre approche, celle du solveur de Bernstein et comparer ces deux approches.

III. Définitions

A. Monôme

Un monôme est une expression de la forme : ***ax****^{n}* ou ***a*** est un nombre réel (ou un nombre complexe) et ***n*** un entier naturel : le nombre ***a*** est appelé coefficient du monôme et le nombre ***n*** est appelé le degré du monôme.

Exemple :

- ***3x²*** est un monôme du second degré et de coefficient 3

B. Polynôme

Un polynôme est une somme de monôme.
Un polynôme s'exprime sous la forme:

$$P= a_0X^0+a_1X^1+a_2X^2+ .... +a_nX^n = \sum_{k=0}^{n} a_k X^k$$

Exemple :

- **3x² - 5x + 7** est un polynôme du second degré
- **-x3 + 4x - 9** est un polynôme du 3 ème degré

C. Base canonique

D. Base de Bernstein

E. Polynôme de Bernstein de degré n

Soit n appartenant à N – {0 ;1}.

Pour $i \in [\![0 ; n]\!]$, le i-ème polynôme de Bernstein de degré n est [D1] :

$$B_{i,n}(t) = C_n^i t^i (1-t)^{n-i}$$

$$\text{Avec } C_n^i = \frac{n!}{i!(n-i)!}$$

On retrouve aussi cette notation [D2] ci-dessous pour un Polynôme de Bernstein de dégrée n :

$$y[0,1](t) = \sum_{k=0}^{n} y_k \binom{n}{k} (1-t)^{n-k} t^k$$

Ou $\binom{n}{k}(1-t)^{n-k}t^k$ est la $k$ième base de Bernstein de dégrée n et y_k est le coefficient de Bernstein.

Un polynôme de Bernstein peut être définie par un domaine arbitraire en introduisant un changement de variable :

$$u = \frac{t-a}{b-a}$$

Ainsi nous avons :

$$y[a,b](t) = \sum_{k=0}^{n} y_k \binom{n}{k} (1-t)^{n-k} t^k = \sum_{k=0}^{n} y_k \binom{n}{k} \frac{(b-t)^{n-k}(t-a)}{(b-a)^n}$$

F. Courbe de Bézier

Des propriétés importantes sont associées avec un polynôme de Bernstein en la transformant en courbe de Bézier :

$$P_{[a,b]}(t) = \left(t, y_{[a,b]}(t)\right) = \sum_{k=0}^{n} P_k \binom{n}{k} (1-u)^{n-k} u^k$$

Avec comme point de control :

$$P_k = (a + \frac{k}{n}(b-a), y_k)$$

IV. Méthodes Usuelles

A. Polynôme de Dégrée 3

1. Méthode Cardan

Cardan [S1] résolut les équations du 3è degré de la forme :

$$x3 + px = q \ , \ x3 = px + q \ , \ x3 + px2 = q$$

où p et q sont des entiers naturels.

- Etude et résolution de l'équation $ax^3 + bx^2 + cx + d = 0$:

Soit l'équation :

(e1) x réel, $ax^3 + bx^2 + cx + d = 0$ (a, b, c et d réels, a non nul)

Divisons par a et posons $= X - \frac{b}{3a}$. On se ramène alors à la forme (e2) : $X^3 + pX + q = 0$ avec :

$$x = X - \frac{b}{3a}, p = \frac{c}{a} - \frac{b^2}{3a^2}, q = \frac{2b^3}{27a^3} + \frac{d}{a} - \frac{bc}{3a^2}$$

La fonction polynomiale f(X) = X3 + pX + q est de degré impair, elle admet donc au moins un zéro réel, que nous appellerons ici le zéro certain.

- Cas triviaux =0 et/ou q=0 :

- si p = 0, il n'y a qu'une seule racine; elle est réelle, c'est la racine cubique de - q.

- si, de plus q = 0, X = 0, - b/3a est une solution triple car on peut écrire : $(X + \frac{b}{3a})^3 = 0$

C'est le cas par exemple de l'équation x3 + 3x2 + 3x + 1 = 0, c'est à dire (x + 1)3 = 0, pour laquelle -1 est racine triple.

-si q = 0, p 0, on se ramène au second degré par factorisation : $X(X2 + p) = 0$

- Cas général :

Dans le cas général, posons dans (e2) : $X = u + v$ et on développe l'expression obtenue : en imposant la condition $3uv = -p$, l'équation (e2) prend alors la forme système équivalente :

$$u3 + v\,3 = -q \;\; et \;\; u\,3.v\,3 = -p3/27 \qquad \text{(e3)}$$

Il s'agit donc de rechercher deux nombres connaissant leur somme et leur produit : la résolution de (e2) est ramenée au second degré. Posons désormais

$$\Delta = \frac{q2}{4} + \frac{p3}{27}$$

-Cas Δ > 0 :

L'équation (e2) $X3 + pX + q = 0$ admet l'unique solution réelle :

$$X_1 = u + v = \sqrt[3]{\frac{-q}{2} + \sqrt{\frac{-q^2}{4} + \frac{p^3}{27}}} + \sqrt[3]{\frac{-q}{2} - \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}} \quad \text{(x1)}$$

Si l'équation est donnée, comme souvent eu égard, historiquement, à Cardan, sous la forme x3 = px + q, il faut alors changer p et q en -p et -q et la formule devient alors :

$$X_1 = u + v = \sqrt[3]{\frac{q}{2} + \sqrt{\frac{q^2}{4} - \frac{p^3}{27}}} + \sqrt[3]{\frac{q}{2} - \sqrt{\frac{q^2}{4} - \frac{p^3}{27}}} \quad \text{(x1)}$$

-Cas Δ < 0 :

La formule de Cardan semble en défaut, mais comme le fit Bombelli sur des cas particuliers, remarquons que D peut se mettre sous la forme $-z2 = z2 \times (-1)$ et n'hésitons pas alors à poser provisoirement :

$$\Delta_2 = z\sqrt{-1}$$

L'application de la formule de Cardan amène à une solution X de la forme :

$$X = \sqrt[3]{s + z\sqrt{-1}} + \sqrt[3]{s - \sqrt{-1}}$$

Cherchons un nombre dont le cube serait $s + z\sqrt{-1}$ sous la même forme $a + b\sqrt{-1}$ Il vient, puisque $(\sqrt{-1})^2 = -1$:

$$a^3 - 3ab^2 + (3a^2b - b^3)\sqrt{-1} = s + z\sqrt{-1}$$

Il faut donc avoir $a^3 - 3ab^2 = s$ et $3a^2b - b^3 = z$ et pour faire de même avec $a - z\sqrt{-1}$, il suffirait de changer b en $-b$: la partie réelle est invariante.

La solution X serait ainsi de la forme : $(a + b\sqrt{-1}) + (a - b\sqrt{-1})$ soit $x = 2a$. C'est un nombre réel : la formule de Cardan fournit donc en fait systématiquement le zéro certain.

Sachant aujourd'hui que tout nombre complexe non nul admet trois racines cubiques distinctes, on en déduit que si D est négatif, l'équation du 3e degré **possède trois solutions réelles distinctes**.

Revenons alors à (e3) où u3 et v3 sont solutions d'une équation du second degré en Z. Un calcul simple conduit à :

$$Z = -\frac{q}{2} \pm i\sqrt{-\frac{q^2}{4} - \frac{p^3}{27}}$$

vec $i^2 = -1$. Les nombres u^3 et v^3 sont donc complexes conjugués. Posons alors :

$u^3 = r \times (cost + i.sint)$, forme trigonométrique de $Z = u^3$

Vu que $\Delta < 0$, on a $p < 0$ et :

$$\text{(e4)}\left|\begin{array}{c} r = \sqrt{\frac{-p^3}{27}}, \cos t = -\frac{q}{2r} \\ u_k = \sqrt{\frac{-p}{3}}\left[\cos\frac{t+2k\pi}{3} + i.\sin\frac{t+2k\pi}{3}\right] \\ v_k = u_k, k \in \{0,1,2\} \end{array}\right.$$

Par conséquent, l'équation (e2) admet trois solutions réelles X_k (éventuellement égales suivant la valeur de q).

L'équation initiale (e1) admet ainsi trois solutions $x_k = X_k - b/3a$.

-Cas Δ = 0 :

On a ici $\frac{q^2}{4} = \frac{-p^3}{27}$ et, nécessairement $p \leq 0$: on est enclin à envisager l'existence d'une solution double voire triple. Si tel est le cas, cette solution annule $3X^2 + p$, expression dérivée de l'équation. On vérifie qu'il en est ainsi et que la 3ème solution est alors :

- $\frac{q}{p}$ si $p < 0$ (**une solution double**), $X_1 = -\sqrt{\frac{-p}{3}}, X_2 = -X_1, X_3 = \frac{3q}{p}$
- 0 si $p = 0$, donc si $= 0$: $x_1 = x_2 = x_3 = 0$, on retrouve la solution triple évoquée ci-dessus.

2. Résultats

Résultat de l'application de la méthode de Cardan en C++ (Annexe 1):

Pour $4x^3 - 5x^2 - 23x + 6 = 0$

```
Trois solutions trouvees!

X1 = 3
X2 = 0.25
X3 = -2
Continuer ? tapez 1
```

Pour $1x^3 + 3x^2 + 5x + 6 = 0$

```
Une solution trouvee!

X1 = -2
Continuer ? tapez 1
```

B. Polynôme de Dégrée 4

1. Méthode Ferrari

L'équation du 4ème degré selon Ludovico Ferrari [S2] fonctionne de la manière suivante :
Soit l'équation d'inconnue x, à coefficients réels, n'a non nul :

$$\text{(e)}: \; ax^4 + bx^3 + cx^2 + dx + e = 0$$

Divisons par a et posons $= X - \frac{b}{4a}$. Le terme en x^3disparaît et (e) se ramène alors à la forme équivalente :

$$\text{(e1)}: \; x = X - \frac{b}{4a} \text{ et} : X^4 + AX^2 + BX + C = 0$$

Avec $A = \frac{-3b^2}{8a^2} + \frac{c}{a}, B = \frac{(b/2)^3}{a^3} - \frac{0.5\times bc}{a^2} + \frac{d}{a}, C = -3(\frac{b}{4a})^4 + \frac{c(\frac{b}{4})^2}{a^3} - \frac{0.25\times bd}{a^2} + \frac{e}{a}$.

- Si B = 0, on se ramène au second degré en posant $X^2 = Y$ (forme $X^4 + AX^2 + C = 0$: équation bicarrée)
- Supposons B non nul. On peut avoir l'idée de faire apparaître un carré en considérant $X^4 + AX^2$ comme le début du carré de $X^2 + \frac{A}{2}$, mais cela ne conduit à rien. Maintenons cependant cette idée en introduisant une inconnue auxiliaire u en calculant

$$(X^2 + \frac{u}{2})^2 = X^4 + uX^2 + u^2/4,$$

Ce qui permet d'écrire :

$$\text{(e2)}: \quad (X^2 + \frac{u}{2})^2 = (u - A)X^2 - BX + u^2/4 - C$$

Équivalente à (e1) pour toute valeur de u.

- On impose les conditions u ≠ A et on force Δ = 0, où Δ est le discriminant de l'équation en X du second membre, ce qui conduit à une équation du 3ème degré, dite équation résolvante :

$$\text{(e3)}: \quad u^3 - Au^2 - 4Cu + 4AC - B^2 = 0$$

Équation du 3e degré que l'on sait résoudre selon la formule de Cardan.

- (e2) devient équivalente à :

$$(X^2 + \frac{u}{2})^2 = (u - A)(X - z)^2$$

où z désigne la solution double du second membre de (e2) correspondant à une racine réelle u de (e3), à savoir :

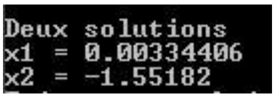
$$z = \frac{B}{2(u - A)}$$

Dans ces conditions, la résolution se ramène à deux équations du second degré. On voit que, dans R, l'équation du quatrième degré possède 0, 2 ou 4 solutions (éventuellement multiples).

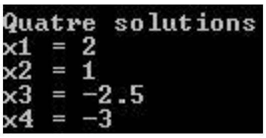
2. Résultats

Résultat de l'application de la méthode de Ferrari C++ (Annexe 2):

Pour $-49X^4 + 158X^3 - 23X^2 - 598X + 2 = 0$



Pour $4X^4 + 10X^3 - 28X^2 - 46X + 60 = 0$



C. Polynôme de Dégrée 5

Pour des polynômes de dégrée 5 il n’existe pas de méthode usuelle pour résoudre celles-ci. Cependant il existe des cas particulier que nous pouvons résoudre.

V. Solveur de Bernstein

VI. Comparaison des deux approches

VII. Sources

[D1] Garnier, Lionel. Mathématiques : pour la modélisation géométrique, la représentation 3D et la synthèse d'images. Ellipses, 2007.

[D2] Spencer, Melvin R. "Polynomial real root finding in Bernstein form." (1994).

[S1] Serge Mehl. Résolution complète de l'équation du 3è degré, http://serge.mehl.free.fr/anx/equ3_cardan.html

[S2] Serge Mehl. L'équation du 4ème degré selon Ludovico Ferrari, http://serge.mehl.free.fr/anx/equ4_ferrari.html

VIII. Annexes

A. Annexes 1

```
#include "Poly3.h"
#define TwoPi  6.28318530717958648
#define PI     3.141592653589793
const double eps = 1e-14;
// fonction pour avoir le signe d'un nombre (1 ou -1)
template <typename T> int sgn(T val) {
    return (T(0) < val) - (val < T(0));
}

Poly3::Poly3(double a, double b, double c) :
    m_a(a),
    m_b(b),
    m_c(c)
{
}
Poly3::Poly3(double a, double b, double c, double d) :
    m_a(a),
    m_b(b),
    m_c(c),
    m_d(d)
{
}



void Poly3::cardan()
{
    // les variables utilis鑼//
```

```
    double p, q, det, u = 0;

    // les trois solutions avec le nombre de sol. //
    double X1 = 0;
    double X2 = 0;
    double X3 = 0;

    p = (m_c / m_a) - (pow(m_b, 2.0)) / (3.0 * pow(m_a, 2.0));
    q = (2.0 * pow(m_b, 3.0)) / (27.0 * pow(m_a, 3.0)) - (m_b * m_c) / (3.0
* pow(m_a, 2.0)) + m_d / m_a;
    //std::cout << "b/3a" << m_b / (3.0 * m_a) << std::endl;
    if (p == 0)
    {
        if (sgn(q) == 1)
        {
            X1 = -m_b / (3.0 * m_a) - pow(q, (1.0 / 3.0));
        }
        else
        {
            X1 = -m_b / (3.0 * m_a) + pow(-q, (1.0 / 3.0));
        }
        std::cout << "Une solution trouvee!\n\nX1 = " << X1;
    }
    else
    {
        det = pow(q, 2.) / 4. + pow(p, 3) / 27.;

        if (det > 0)
        {
            /* le teste suivant est utile car le c++ (comme le vb)
             ne savent pas faire des racine cubique n罵tives ...*/

            if (sgn(-q * 0.5 + pow(det, 0.5)) == 1)
            {
                u = pow((-q * 0.5 + pow(det, 0.5)), (1. / 3.));
            }
            else
            {
                u = -pow(-(-q * 0.5 + pow(det, 0.5)), (1. / 3.));
            }
            X1 = -m_b / (3.0 * m_a) + u - (p / (3. * u));
            std::cout << "Une solution trouvee!\n\nX1 = " << X1;
        }
        if (det == 0)
        {
            X1 = -m_b / (3. * m_a) + sgn(q) * pow((-p / 3.), 0.5);
            X2 = -m_b / (3. * m_a) - 2.0*sgn(q) * pow((-p / 3.), 0.5);
            std::cout << "Trois solutions trouvees dont une double!\n\nX1
et X2  = " << X1 <<" et "<< X1 << "\nX2 = " << X2;
        }
        if (det < 0)
        {
            double r = sqrt(-p / 3.);
            //double t = acos((-q)/(2*r));
            //double t = ((-q) / (2 * r));
            double vt = -(m_b) / (3. * m_a);

            double alf = 1. / 3. * acos(-q / 2. * pow(27. / (pow(-p, 3.)),
0.5));

            X1 = vt + 2. * sqrt(-p / 3.) * cos(alf);
```

```
            X2 = vt + 2. * sqrt(-p / 3.) * cos(alf+( 2. * PI) / 3.);
            X3 = vt + 2. * sqrt(-p / 3.) * cos(alf+( 4. * PI) / 3.);


            /* Autre m鴨ode qui revient au mΩ sans utiliser PI
            double omega = acos(-q / (2 * sqrt(pow(-p, 3) / 27)));
            X1 = vt + 2 * r * cos(omega / 3);
            X2 = vt - r * cos(omega / 3) + sqrt(p*( pow(cos(omega/3),2) ) -
p );
            X3 = vt - r * cos(omega / 3) - sqrt(p * (pow(cos(omega / 3),
2)) - p);
            */

            std::cout << "Trois solutions trouvees!\n\nX1 = " << X1 <<
"\nX2 = " << X2 << "\nX3 = " << X3;


        }
    }
    //permet de recuperer les racines pour des applications suivantes
    this->m_racines[0] = X1;
    this->m_racines[1] = X2;
    this->m_racines[2] = X3;
}

void Poly3::getRacines(double racines[])
{
    for (int i = 0; i < 3;i++) {
        racines[i] = this->m_racines[i];
    }
    //racines = this->m_racines;
}
```

B. Annexes 2

```
#include "Poly4.h"

template <typename T> int sgn(T val) {
    return (T(0) < val) - (val < T(0));
}

Poly4::Poly4(double a, double b, double c, double d, double e) :
    m_a(a),
    m_b(b),
    m_c(c),
    m_d(d),
    m_e(e)
{

}
void Poly4::bicar(double a, double c) {
    double delta = -4 * a * c;
    if (delta<0) {
        //pas de solution
        std::cout << "Pas de solution" << std::endl;
    }
    else {
        delta = sqrt(delta);
        double x12 = (-a + delta) / 2;
        double x32 = (-a - delta) / 2;
        bool t12 = 0;
```

```
        bool t22 = 0;

        double x1, x2, x3, x4 = 0;

        if (x12 >= 0) {
            x1 = sqrt(x12);
            x2 = -x1;

            t12 = 1;
        }
        if (x32 >= 0 && t12 == 1) {
            x3 = sqrt(x32);
            x4 = -x3;
            t22 = 1;
        }
        if (x32 >= 0) {
            x3 = sqrt(x32);
            x4 = -x3;
        }
        if (t12 && t22) {
            std::cout << "Il y a 4 racines : " << std::endl;
        }
        else if (t12) {
            std::cout << "Il y a 2 racines : " << std::endl;
            std::cout << "X1 : " << x1 <<"X2 : "<<x2<< std::endl;
        }
        else if (t22) {
            std::cout << "Il y a 2 racines : " << std::endl;
            std::cout << "X1 : " <<x3<<"X2 : "<<x4<< std::endl;
        }
    }
}
void Poly4::ferrrari()
{
    double A = (-3 * pow(m_b, 2)) / (8 * pow(m_a, 2)) + (m_c / m_a);
    double B = ((pow(m_b / 2, 3)) / pow(m_a, 3)) - ((0.5 * m_c * m_b) /
pow(m_a, 2)) + (m_d / m_a);
    double C = (-3 * pow(m_b / (4 * m_a), 4)) + ((m_c * pow(m_b / 4, 2)) /
pow(m_a, 3)) - ((0.25 * m_b * m_d) / pow(m_a, 2)) + (m_e / m_a);

    //std::cout << "A = " << A << "\n B = " << B << "\n C = " << C <<
std::endl;
    //Equation bicarr鑼de la forme X^4+AX^2+C =0 avec X^2 = Y
    if (B == 1e-14) {
        std::cout << "Solution bicarre" << std::endl;
        bicar(A, C);

    }
    else {
        //Resolution du degr鑼3
        double a = 1;
        double b = -A;
        double c = -4 * C;
        double d = 4 * A * C - pow(B, 2);
        double bs = m_b / 4 / m_a;
        Poly3 p3(a, b, c, d);
        std::cout << "Racines de la forme du polynome " << std::endl;
        p3.cardan();
        double racinesCardan[3];
        p3.getRacines(racinesCardan);
```

```
        double p = (c / a) - (pow(b, 2.0)) / (3.0 * pow(a, 2.0));
        double r = sqrt(-p / 3);
        //std::cout << "p = " << p << std::endl;
        double u;
        //for (int i = 0; i < 2; i++) {
        //  std::cout << "racines p3 x = " << racinesCardan[i] <<
std::endl;
        //}
        //si nb racines > 1
        if (sizetab(racinesCardan) > 1) {
            bool end = false;
            if (r == 0) {
                u = racinesCardan[0];
                //endFerrari();
                end = true;
            }
            if (racinesCardan[0] > A && end == false) {
                u = racinesCardan[0];
                //endFerrari();
                end = true;
            }
            if(end == false)
                u = racinesCardan[1];

            if (racinesCardan[2] > A && end == false) {
                u = racinesCardan[2];
            }

        }
        else { //si nb racines =1
            u = racinesCardan[0];
        }

        //endFerrari
        double uma = u - A;
        double z = B / (2 * uma);
        std::cout << "z = " << z << std::endl;  //z pas la bonne valeur
        //std::cout << "u-A = " << uma << std::endl;

        //R鳳lution polynome Degr鑼2

        double d1 = uma - 4 * (z * sqrt(uma) + u / 2);
        double t1 = 0;
        double x1, x2, x3, x4 = 0;
        std::cout << "bs : " << bs << std::endl;
        if (d1 >= 0) {
            x1 = (sqrt(uma) + sqrt(d1)) / 2 - bs ;
            x2 = (sqrt(uma) - sqrt(d1)) / 2 - bs ;

            t1 = 1;
        }
        double d2 = uma - 4 * (-z * sqrt(uma) + u / 2);
        double t2 = 0;
        if (d2 >= 0) {
            x3 = (-sqrt(uma) + sqrt(d2)) / 2 - bs ;
            x4 = (-sqrt(uma) - sqrt(d2)) / 2 - bs ;
            t2 = 1;
        }
        if (t1==0 && t2 == 0) {
            std::cout << "\n!!Pas de solutions !!\n" << std::endl;
```

```
        }
        if (t1 * t2 == 0 && (t1 == 1 || t2 ==1))
        {
            std::cout << "\nDeux solutions" << std::endl;

            if (t1 == 1) {
                std::cout << "x1 = " << x1 << "\nx2 = " << x2 << std::endl;
            }
            if (t2 == 1) {
                std::cout << "x1 = " << x3 << "\nx2 = " << x4 << std::endl;
            }
        }
        else if (t1 ==1 && t2 ==1)
        {
            std::cout << "\nQuatre solutions" << std::endl;
            std::cout << "x1 = " << x1 << "\nx2 = " << x2 << std::endl;
            std::cout << "x3 = " << x3 << "\nx4 = " << x4 << std::endl;

        }
    }
}

double Poly4::sizetab(double tab[])
{
    double compteur = 0;
    for (int i = 0; i < 2; i++) {
        if (tab[i] != 0) {
            compteur++;
        }
    }
    return compteur;
}
```