Rapport de projet Algorithme Numérique
Interpolation, matrice de Vandermonde

Group i1pro212 :

Coordinateur rapporteur:
BOUBEKKI Amirouche
Architecte integrateur
EL AFRIT Mohamed Amine
Développeurs:
LE MAOUT Yves
MERBETH Gaël
SALLIER Mathilde
LaTeX

19 mars 2007

Table des matières

Chapitre 1

Introduction

1.1 Interpolation polynomiale : matrices de Vandermonde

On cherche l'unique polynôme de degré n passant par les points (x_i, y_i) pour $i = 0 \dots n$, en supposant les x_i tous distincts.

Le polynôme d'interpolation peut donc s'écrire

$$p_n(x) = \sum_{i=0}^{n} a_i x^i \tag{1.1}$$

tel que pour tout i, $p_n(x_i) = y_i$, soit en notation matricielle à l'aide de la matrice de Vandermonde :

$$\begin{pmatrix} 1 & x_0^1 & \dots & x_0^n \\ 1 & x_1^1 & \dots & x_1^n \\ \vdots & \vdots & \dots & \vdots \\ 1 & x_n^1 & \dots & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix} \tag{1.2}$$

1.2 Propriété de la matrice de vandermonde

1.2.1 Propriété

On considère une matrice V de Vandermonde carrée. Elle est inversible si et seulement si les x_i sont deux à deux distincts.

Cette propriété va nous permettre de savoir au préalable s'il est possible de trouver le polynome nécessaire en se basant sur les points données.

1.2.2 Demonstration

De façon matricielle, elle se présente ainsi :

$$V_{i,j} = \alpha_i^{j-1} \; \forall \; i \; \text{et} j$$

$$V = \begin{pmatrix} 1 & \alpha_1 & \alpha_1^2 & \dots & \alpha_1^{n-1} \\ 1 & \alpha_2 & \alpha_2^2 & \dots & \alpha_2^{n-1} \\ 1 & \alpha_3 & \alpha_3^2 & \dots & \alpha_3^{n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \alpha_m & \alpha_m^2 & \dots & \alpha_m^{n-1} \end{pmatrix}$$

Si deux coefficients α_i sont identiques, la matrice a deux lignes identiques, donc n'est pas inversible. Pour la réciproque, on peut procéder au calcul du déterminant

Le déterminant d'une matrice $n \times n$ de Vandermonde ($m = n$ dans ce cas) peut s'exprimer ainsi : $\det(V) = \prod_{1 \leq i < j \leq n} (\alpha_j - \alpha_i)$

Une preuve d'inversibilité plus rapide est cependant de considérer V comme la matrice du Système d'équations linéaires :système linéaire homogène "VX=0" pour "X" de composantes $x_0, x_1 ... x_{n-1}$

$$\begin{cases} x_0 + \alpha_1 x_1 + \alpha_1^2 x_2 + & \cdots + \alpha_1^{n-1} x_{n-1} = 0 \\ & \cdots \\ x_0 + \alpha_n x_1 + \alpha_n^2 x_2 + & \cdots + \alpha_n^{n-1} x_{n-1} = 0 \end{cases}$$

Mais en introduisant le polynôme $P = \sum_{i=0}^{n-1} x_i X^i$ On voit que si "X" vérifie l'équation "VX=0", alors "P" admet "n" racines distinctes, soit plus que son degré. Donc "P" est nul, et ainsi "X=0". Ce qui prouve que "V" est inversible.

1.3 Exemple d'interpolation

On cherche à déterminer le polynôme de degré 3

$$p_3(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 \tag{1.3}$$

passant par les points $(-2, 10), (-1, 4), (1, 6)$ et $(2, 3)$.

Les a_i peuvent être déterminés par la résolution du système linéaire :

$$\begin{pmatrix} 1 & -2 & 4 & -8 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} 10 \\ 4 \\ 6 \\ 3 \end{pmatrix} \tag{1.4}$$

La solution de ce système peut être obtenue de la façon suivante :

```
y = [10;4;6;3];
V= [1,-2,4,-8;1,-1,1,-1;1,1,1,1;1,2,4,8];
a = inv(V)*y;
```

1.4 Résolution du système

1.4.1 Génération de la matrice de Vandermonde

La matrice de Vandermonde `V` peut être déterminée à partir du vecteur `x` de la façon suivante :(en se basant sur la formule 1.2.2)

```
function V=cons_vand(x)
  n=length(x);
  V=zeros(n,n);
  for j=1:n
    for i=1:n
      V(i,j)=x(i)^(j-1);
    end
  end
endfunction;
```

1.4.2 Élimination de Gauss

L'algorithme de Gauss est une méthode permettant l'élimination de certains coefficiants non nuls et de rendre la matrice triangulaire. Le problème se situe au niveau du choix du pivot en effet, ceci peut influencer la "qualité" de la solution obtenue aprés resolution du système lineaire. Nous avons expérimenté trois strategies concernant le choix du pivot : stratégie du pivot maximum partiel,stratégie du pivot avec seuil, stratégie du pivot du pivot avec maximum total. Le choix du pivot doit permettre de minimiser les erreurs liées aux arrondis.

Stratégie du pivot maximum partiel

Voir sources 3.1
Cette strategie consiste à selectionner le plus grand pivot (en valeur absolue) parmi ceux qui sont dans la colonne où l'on travaille. On permute alors ligne avec la ligne où se trouve le pivot selectionné.

Stratégie du pivot avec seuil

Voir sources 3.1
Cette strategie cherche un autre pivot que si le pivot actuel est plus petit qu'un certains seuil. Ce seuil est sélectionné en fonction des capacités de calcul (en terme de précision) de la machine, pour un ordinateur classique on peut choisir 10^{-10}. Cette methode augmente les problèmes liés aux arrondis. Nous avons rencontré des problèmes dans certains cas :

$$\begin{pmatrix} 10^{-10} & 123456783 \\ 15753 & 1 \end{pmatrix} \tag{1.5}$$

La strategie du pivot avec seuil ne donne pas la bonne solution alors que les deux autres strategies le font correctement.

Stratégie du pivot avec maximum total

Voir sources 3.1
Cette methode recherche le plus grand pivot (en valeur absolue) dans toute la matrice. On permute alors les deux lignes et les deux colonnes. La difficulté etant qu'il faut mémoriser ces permutations pour les repercuter sur les resutlats. Cette methode nous permet de recuperer toujours le plus grand pivot mais ceci necessite beaucoup de calcul.

1.4.3 Résolution à partir d'une matrice triangulaire

Nous avons utilisé la fonction suivante qui permet de remonter la matrice triangulaire pour déterminer les solutions. Cette fonction suppose que la matrice passée en argument est triangulaire superieure. Elle procéde de la façon suivante :

- Tout d'abord, on prend la dernière ligne, qui nous donne X_n en fonction de A_{nn} et B_n :

$$X_n = \frac{B_n}{A_{nn}}$$

- Puis pour chaque ligne on a :

$$\forall i \in [|1, n-1|], X_i = \frac{B_i - \sum_{k=i+1}^{n} A_{ik} X^k}{A_{ii}}$$

1.5 Evaluation du polynôme d'interpolation

Indépendamment de la détermination des coefficients du polynôme d'interpolation, on considère dans cette section l'évaluation d'un polynôme $p_n(x) = a_0 + \ldots + a_n x^n$ pour $x = z$, en supposant z et les a_i connus.

En utilisant la formule suivante :

$$p_n(x) = (\ldots((a_n x + a_{n-1})x + a_{n-2})x + \ldots)x + a_0 \tag{1.6}$$

on peut évaluer le polynôme à l'aide de la fonction suivante :

```
function p=evaluation(v)
  s=poly(0,"s");
  n=length(v);
  p=v(n);
  for i=n-1:-1:1
    p= p*s + v(i);
  end
endfunction;
```

Chapitre 2

Conclusion

Le code de notre projet a la forme suivante : les valeurs sont saisies par l'utilisateur, elles sont mises sous forme matrice de Vandermonde et si cette matrice est inversible, alors le système est résolu avec la méthode du pivot de Gauss.

Nous avons essayé de résoudre le problème en utilisant différentes stratégies : la stratégie du pivot maximal partiel, la stratégie du pivot avec seuil et la stratégie du pivot maximal total. Le plus difficile à coder est la stratégie du pivot maximal total car la permutation de colonnes pose problème.

Au cours de ce projet, un des obstacles a été l'apprentissage de la syntaxe de scilab. Nous avons découvert les mots clé de base, puis quelques fonctions plus évoluées permettant d'implémenter facilement les algorithmes numériques comme celui du pivot de Gauss.

Chapitre 3

Annexes

3.1 Code source

```
//Premier projet d'algorithme numÃ©rique:
//****Interpolation : matrice de Vandermonde****
//***********Ã©quipe************
//***Coordinateur rapporteur***
//Boubekki Amirouche
//***Architecte integrateur***
//EL AFRIT Mohamed Amine
//***DÃ©veloppeurs
//LE MAOUT Yves
//MERBETH Gael
//SALLIER Mathilde

****main1***

  clear;
  exec ~/projet1/saisie.sci,
  exec ~/projet1/cons_vand.sci,
  exec ~/projet1/pivot_gauss.sci,
  exec ~/projet1/resolution.sci,
  exec ~/projet1/evaluation.sci,

  [X,Y]=saisie(),
  M=cons_vand(X);
  [M,Y]=pivot_gauss(M,Y);
  Z=resolution(M,Y);
  p=evaluation(Z),

***saisi***
function [X,Y]=saisie()
  N = input("Combien de point souhaitez-vous utiliser (N) ");
  for i=1:N do
    disp(i, "Point ")
    X(i) = input("x ")
    Y(i) = input("y ")
  end;
endfunction;

***cons_vand***
function M=cons_vand(x)
  n=length(x);
  M=zeros(n,n);
  for j=1:n
    for i=1:n
      M(i,j)=x(i)^(j-1);
    end
  end
endfunction;

***pivot_gauss***
function [M,V] = pivot_gauss (M,V)
  c=size(M);
  n=c(1);
  if n<>1 then
    indmax=1;
    j=M(1,1);
    for i=2:n do
      if M(i,1)>j then indmax=i;
      end
    end
    //permutation
    auxV=V(1);
    V(1)=V(indmax);
    V(indmax)=auxV;

    auxM=M(1,1:n);
    M(1,1:n)=M(indmax,1:n);
    M(indmax,1:n)=auxM;

    V(1)= 1/M(1) *V(1);
    M(1,1:n)= 1/M(1,1) *M(1,1:n);
    //elimination de gauss
    for i=2 :n do
      V(i)=V(i) - M(i,1)*V(1);
```



```
      M(i,1:n)=M(i,1:n) - M(i,1)*M(1,1:n);
    end
    [M(2:n,2:n),V(2:n)]=pivot_gauss (M(2:n,2:n),V(2:n));
  else
    V(1)=1/M(1)*V(1);
    M(1)=1;
  end
endfunction

***rÃ©solution***

function X=resolution(A,B)
        [m,n] = size(A),
        if m == n then //si la matrice est carre
                X(n)=B(n)/A(n,n),
                for i=n-1:-1:1,
                        // Calcule de la somme
                        som = 0,
                        for k=i+1:n,
                                som = som + A(i,k)*X(k),//ici c'est bien A(i,k)
et non A(k,i)
                        end
                        if A(i,i) == 0 then
                                disp("Erreur: division par zero"),
                        else
                                X(i)=(B(i)-som)/A(i,i),
                        end
                end
        else
                disp("La matrice n''est pas carÃ©e"),
        end
endfunction;

***evaluation***

function p=evaluation(v)
  s=poly(0,"s");
  n=length(v);
  p=v(n);
  for i=n-1:-1:1
    p= p*s + v(i);
  end
endfunction;


***************amÃ©lioration***************


***pivot de gauss avec seuil ***

function [M,V] = pivot_gauss (M,V)
  seuil=10**(-16); //on choisit un seuil pour le pivot
  c=size(M);
  n=c(1);
  if n<>1 then
    indmax=1;
    j=M(1,1);
    if j<seuil then  //test sur le seuil
      for i=2:n do
        if M(i,1)>j then indmax=i;
        end
      end
    end
    //permutation
    auxV=V(1);
    V(1)=V(indmax);
    V(indmax)=auxV;

    auxM=M(1,1:n);
    M(1,1:n)=M(indmax,1:n);
    M(indmax,1:n)=auxM;

    V(1)= 1/M(1) *V(1);
```

```
      M(1,1:n)= 1/M(1,1) *M(1,1:n);
      //elimination de gauss
      for i=2 :n do
        V(i)=V(i) - M(i,1)*V(1);
        M(i,1:n)=M(i,1:n) - M(i,1)*M(1,1:n);
      end
      [M(2:n,2:n),V(2:n)]=pivot_gauss (M(2:n,2:n),V(2:n));
    else
      V(1)=1/M(1)*V(1);
      M(1)=1;
    end
  endfunction

  ***main2***

    [X,Y]=saisie(),
    M=cons_vand(X);
    n=size(X);
    [M,Y,T]=pivot_gauss(M,Y,1:n(1)); \\T est un tableau qui mÃ©morise
                                     \\l'ordre des colonnes dans lequelles
                                     \\on a fait des permutation
    Z=resolution(M,Y,T);
    p=evaluation(Z),

  ***pivot de gauss avec maximum total***

  function [M,V,T] = pivot_gauss (M,V,T)
    c=size(M);
    n=c(1);
    if n<>1 then
      indmax1=1;
      indmax2=1;
      j=M(1,1);
      for a=2:n do
        for b=2:n do
          if M(a,b)>j then
            indmax1=a;
            indmax2=b;
          end
        end
      end

      //permutation des lignes
      auxV=V(1);
      V(1)=V(indmax1);
      V(indmax1)=auxV;
      auxM=M(1,1:n);
      M(1,1:n)=M(indmax1,1:n);
      M(indmax1,1:n)=auxM;
      //permutation des colonnes
      auxM2=M(1:n,1);
      M(1:n,1)=M(1:n,indmax2);
      M(1:n,indmax2)=auxM2;
      //permutation des indices du tableau d'indices
      aux=T(1);
      T(1)=T(indmax2);
      T(indmax2)=aux;
      //remttre Ã  1 le pivot
      V(1)= 1/M(1) *V(1);
      M(1,1:n)= 1/M(1,1) *M(1,1:n);
      //elimination de gauss
      for i=2 :n do
        V(i)=V(i) - M(i,1)*V(1);
        M(i,1:n)=M(i,1:n) - M(i,1)*M(1,1:n);
      end
      [M(2:n,2:n),V(2:n),T(2:n)]=pivot_gauss (M(2:n,2:n),V(2:n),T(2:n));
    else
      V(1)=1/M(1)*V(1);
      M(1)=1;
    end
  endfunction

  ***resolution 2 pour le pivot maximal total***
```



```
function X=resolution(A,B,T)
        [m,n] = size(A),
        if m == n then //si la matrice est carre
                X(n)=B(n)/A(n,n),
                for i=n-1:-1:1,
                        // Calcule de la somme
                        som = 0,
                        for k=i+1:n,
                            som = som + A(i,k)*X(k),
                        end
                        if A(i,i) == 0 then
                                disp("Erreur: division par zero"),
                        else
                                X(i)=(B(i)-som)/A(i,i),
                        end
                end
                for a=1:n do   \\lÃ  on remet dans l'ordre les solutions
                  for b=a+1:n do
                    if T(b)=a then
                      aux=X(a);
                      X(a)=X(b);
                      X(b)=aux;
                      aux=T(a);
                      T(a)=T(b);
                      T(b)=aux;
                    end
                  end
                end
        else
                disp("La matrice n''est pas carÃ©e"),
        end
endfunction;
```