Algorithmes numériques – Rapport

Polynôme caractéristique, valeurs propres et vecteurs propres

Axel Delsol, Pierre-Loup Pissavy

Décembre 2013

Table des matières

1 Préambule

1.1 Structure du programme

Nous avons conçu un programme principal avec menus, présenté sous la forme suivante :

```
Menu principal : Polynôme caractéristique, valeurs propres et
  vecteurs propres

Entrez n la dimension de la matrice :
(Saisie des valeurs de la matrice...)

(Affichage de la matrice correspondants...)
Quelle résolution utiliser ?
1- Méthode de Leverrier
2- Méthode de Leverrier améliorée
9- Nouvelle série de points (Menu principal)
0- Quitter
Votre choix :
```

FIGURE 1.1 – Apercu : Menu Principal

Au lancement, le programme demande la saisie des valeurs, qu'il stocke dans un tableau, puis affiche le menu. Après chaque méthode, il est possible de réutiliser le jeu de données (chaque méthode qui doit modifier les valeurs utilise un duplicata).

Le menu principal est codé dans le fichier source `main.c`. Les méthodes sont codées dans des fichiers individuels sauf les méthodes de leverrier qui sont réunies. Les prototypes des fonctions sont écrits dans les headers correspondants. La liste de tous ces fichiers est présentée figure 1.3.

Le stockage des valeurs se fait en double précision (type `double`, 64 bits) afin d'obtenir des résultats suffisamment précis.
De plus, les méthodes de leverrier utilisent une structure de polynôme (composée du degré et des coefficients), présentée figure 1.2, pour faciliter la compréhension du code et les méthodes de puissances itérées et déflation utilisent une structure de file, présentée figure afin de stocker les valeurs propres et les vecteurs propres associées.

```
4 typedef struct polynome
5 {
6   int d; //degree
7   double* poln; //coefficients
8 } polynome;
```

FIGURE 1.2 – Code : Structure de Polynôme

```
leverrier.c
main.c
matrices.c
polynome.c
puissancesIt.c
useful.c
file.h
leverrier.h
matrices.h
polynome.h
puissancesIt.h
useful.h
makefile
```

FIGURE 1.3 – Apercu : Arborescence des fichiers `C` et `makefile`

1.2 Compilation et logiciels utilisés

2 Polynôme caractéristique

2.1 Méthode de Leverrier

2.1.1 Présentation

La méthode de Leverrier permet de déterminer le polynôme caractéristique d'une matrice $A \in M_{n,n}(\Re)$ c'est-à-dire le déterminant $|A - \lambda I_n| = P(\lambda)$ où I_n est la matrice identité. Le but est donc de déterminer les coefficients a_i du polynome $P(\lambda) = a_n + a_{n-1}\lambda + \cdots + a_0\lambda^n$.

Pour les trouver, on définit d'abord $S_p = Tr(A^p)$ puis on applique les identités de Newton l'aide de la formule suivante :

$$\forall p \in \{1, \cdots, n\}, \quad \begin{cases} a_0 = (-1)^n \\ a_p = -\frac{\sum_{i=0}^{p-1} a_i \cdot S_{p-1}}{p} \end{cases}$$

2.1.2 Programme

```
9  void leverrier(double** A, int n)
10 {
11   int i, j;
12   double tempo;
13   double* coeffs=(double*) malloc ((n+1)*sizeof(double)); //tableau coeffs
14   double* traces=(double*) malloc(n*sizeof(double)); //tableau traces
15   double** tmp;
16   polynome* p;
17
18   //On remplit notre tableau contenant les traces
19   for (i=1; i<=n; i++)
20   {
21     tmp=puissanceMatrice(A, n, i);
22     traces[i-1]=trace(tmp, n);
23     for (j=0; j<n; j++)
24     {free(tmp[j]);}
25     free(tmp);
26   }
27
28   //On remplit le tableau des coefficients
29   for(i=0; i<=n; i++)
30   {
31     coeffs[i]=0;
32   }
33   coeffs[0]=pow(-1.0,n);
34   for(i=1; i<=n; i++)
35   {
36     for(j=0;j<i;j++)
37     {
38       coeffs[i] = coeffs[i] - coeffs[j]*traces[i-j-1];
39     }
40     coeffs[i] = coeffs[i]/i;
41   }
42   //On inverse le tableau des coefficients
43   for(i=0;i<=(n/2)-1;i++)
```

```
44    {
45      tempo = coeffs[i];
46      coeffs[i] = coeffs[n-i];
47      coeffs[n-i] = tempo;
48    }
49    p=creerPoly(n+1, "tableau", coeffs);
50    menuAffichage(p);
51
52    //liberation memoire
53    //libération du tableau de trace
54    free(traces);
55    free(coeffs);
56    //libération du polynome
57    free(p->poln);
58    free(p);
59  }
```

FIGURE 2.1 – Code : leverrier.c

2.2 Méthode de Leverrier améliorée

2.2.1 Présentation

2.2.2 Programme

```
61  void leverrierA(double** A, int n)
62  {
63    int i, j;
64    double tempo;
65    double* coeffs=(double*) malloc ((n+1)*sizeof(double)); //tableau coeffs
66    double** Ak=(double**) malloc (n*sizeof(double*));
67    double** tmp;
68    double** B=(double**) malloc (n*sizeof(double*));
69    double** I=(double**) malloc (n*sizeof(double*));
70    polynome* p;
71
72    for (i=0; i<n; i++)
73    {
74      Ak[i]=(double*) malloc (n*sizeof(double));
75      I[i]=(double*) malloc (n*sizeof(double));
76      B[i]=(double*) malloc (n*sizeof(double));
77    }
78
79    //Création matrice identité, initialisation de B et copie de A
80    for (i=0; i<n; i++)
81    {
82      for (j=0; j<n; j++)
83      {
84        Ak[i][j]=A[i][j];
85        if (i==j)
86        {
87          I[i][j]=1;
88          B[i][j]=1;
89        }
90        else
91        {
92          I[i][j]=0;
93          B[i][j]=0;
94        }
95      }
96    }
97
98    //On remplit le tableau des coefficients
99    for(i=0; i<=n; i++)
100   {
101     coeffs[i]=0;
102   }
103
```

```
104   coeffs[0]=1;
105
106
107   for(i=1; i<=n; i++)
108   {
109     Ak=produitMatriciel(B,A,n,n,n);
110     coeffs[i]=-trace(Ak,n)/i;
111     tmp=produitSMatriciel(I,n,n,coeffs[i]);
112     for(j=0;j<n;j++)
113     {
114       free(B[j]);
115     }
116     free(B);
117     B=difference(Ak,tmp,n,n);
118     for(j=0;j<n;j++)
119     {
120       free(Ak[j]);
121       free(tmp[j]);
122     }
123     free(tmp); free(Ak);
124   }
125
126   //On inverse le tableau des coefficients
127   for(i=0;i<=(n/2)-1;i++)
128   {
129     tempo = coeffs[i];
130     coeffs[i] = coeffs[n-i];
131     coeffs[n-i] = tempo;
132   }
133   p=creerPoly(n+1, "tableau", coeffs);
134   menuAffichage(p);
135
136   //liberation memoire
137   for (i=0; i<n; i++)
138   {
139     free(B[i]);
140   }
141   free(B);
142   free(coeffs);
143   //libération du polynome
144   free(p->poln);
145   free(p);
146 }
```

FIGURE 2.2 – Code : leverrier.c

2.3 Résultat des tests