Chapitre 3 : corrigé de TP en SCILAB

Exercices 1 et 2

```
//programme valeurspropres.sce
clear
clf
format(10)
A0=[2 -1 0; -1 2 -1; 0 -1 2]
vpA=[2-sqrt(2), 2,2+sqrt(2)]

//A0=[1/sqrt(2) 1/sqrt(2);1/sqrt(2) -1/sqrt(2)] cas d'une matrice 2*2 orthogonale
'algorithme LR'
A=A0;
for n=1:15
    [L,U]=lu(A);
    A=U*L;
end
A
'algorithme QR'
A=A0;
for n=1:15
    [Q,R]=qr(A);
    A=R*Q;
end
A

//cas test 1
A=diag([1 2 -3])
P=[1 2 3; 0 1 1; 0 2 3]; //Rq : det(P)=1
A=inv(P)*A*P  // Rq : A est une matrice diagonalisable a coef. entiers

//A=[1 1 0 ; 0 1 0 ; 0 0 -2]; un autre cas-test : matrice non diagonalisable
'methode de la puissance'

y=[1 1 1]';//initialisation
format(20)
for n=1:20
n
    z=A*y;
    y=z/norm(z,'inf') //normalisation
    mu=y'*A*y/(y'*y) //coefficient de Rayleigh NB : y'*y=1
end

//vitesse de convergence pour le cas-test
y=[1 1 1]';
```

```
Taberreur=[];
nmax=100;
for n=1:nmax
    z=A*y;
    y=z/norm(z,'inf'); //normalisation
    mu=y'*A*y/(y'*y); //coefficient de Rayleigh NB : y'*y=1
    Taberreur=[Taberreur mu+3];
end
compare=zeros(1,nmax);
for n=1:nmax
   compare(n)=(2/3)^n;
end
clf
plot2d([1:nmax],abs(Taberreur),logflag='nl',style=1)
plot2d([1:nmax],compare,logflag='nl',style=2)
legend(['erreur(n)','(2/3)^n'])
xtitle('Erreur pour la methode de la puissance','iteration n','')
```

Ce programme produit la sortie suivante :

```
--> A0  =

    2.  - 1.    0.
  - 1.    2.  - 1.
    0.  - 1.    2.
 vpA  =

    0.5857864    2.    3.4142136
 ans  =

 algorithme LR
 A  =

    3.413286   - 1.           0.
  - 0.0013109    2.0009275  - 1.
    0.         - 4.009E-08    0.5857865
 ans  =

 algorithme QR
 A  =

    3.4142133  - 0.0006563    1.307E-16
  - 0.0006563    2.0000003    2.003E-08
    0.           2.003E-08    0.5857864
 A  =

    1.    0.    0.
    0.    2.    0.
    0.    0.  - 3.
 A  =
```

```
    1.    8.     12.
    0.    12.    15.
    0.  - 10.  - 13.
  ans  =

 methode de la puissance
n  =

    1.
 y  =

    0.77777777777777779
    1.
  - 0.85185185185185186
 mu  =

  - 1.20835785756327496
 n  =

    2.
 y  =

  - 1.
  - 0.53846153846153877
    0.743589743589744
 mu  =

  - 5.06064930431680526


.............................


 n  =

    18.
 y  =

  - 1.
  - 0.99918803571337567
    0.99945869150805344
 mu  =

  - 3.00225591644116641
 n  =

    19.
 y  =

    0.9994589823039391
```

```
    1.
  - 0.99981966042400905
 mu  =

  - 2.99849655438151075
 n  =

    20.
 y  =

  - 1.
  - 0.99963912595224602
    0.99975941741621632
 mu  =

  - 3.00100251772594895
```

On voit notamment que $\mu_n \to -3$ pour la méthode de la puissance, et -3 est bien la valeur propre de A qui est la plus grande en valeur absolue. Le vecteur $(-1)^n y_n$ tend vers un vecteur propre, comme prévue par la théorie.

Le programme produit également la figure suivante:

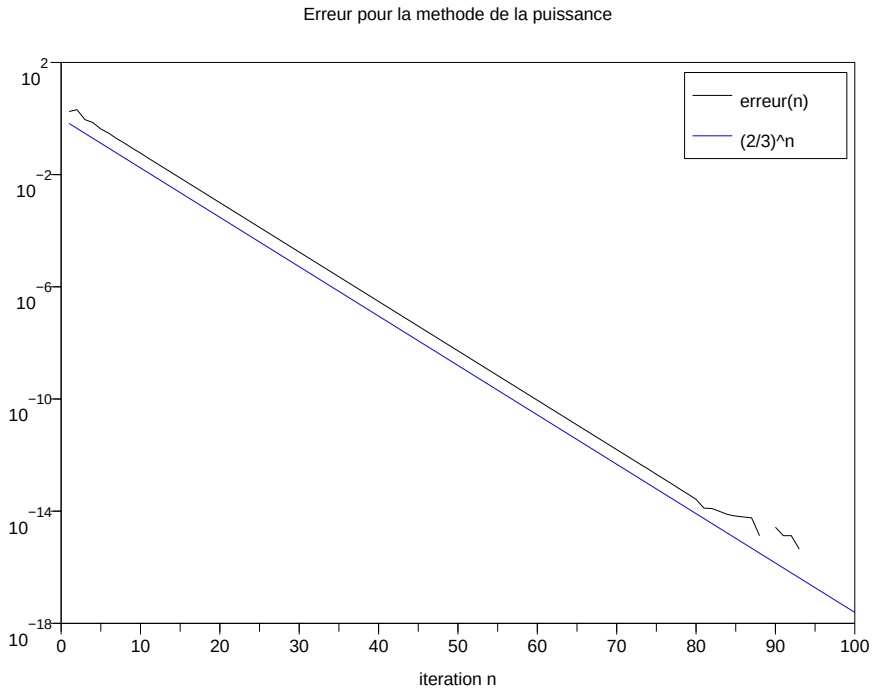

Figure 1: Erreur de convergence pour la méthode de la puissance

On voit que la méthode de la puissance converge bien en $O((2/3)^n)$ pour la matrice choisie (qui n'est pas normale), comme prévu par la théorie. Remarquer que lorsque l'erreur est de l'ordre de 10^{-16}, i.e. la précision de l'ordinateur, elle est parfois nulle.