TP4: Méthode de la puissance et puissance inverse pour le calcul de valeurs propres

Créer un répertoire "TP4-NOM" et lancer PYTHON dans ce répertoire.

Exercice 1

Considérons la matrice $A \in M_N(\mathbb{R})$ définie par

$$A = \begin{pmatrix} 2 & -1 & 0 & \dots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 2 & -1 \\ 0 & \dots & 0 & -1 & 2 \end{pmatrix}$$

Dans ce qui suit, on considérera les cas $N = 10$ et $N = 100$

1. Créer un fichier "puissance.py". Ecrire un script pour construire A pour toute valeur de N. Programmer la méthode de la puissance pour calculer la plus grande valeur propre de A. Tracer l'évolution de la valeur propre calculée en fonction du nombre d'itérations. Donner l'ordre et la vitesse de convergence.
2. Créer un fichier "puissance-inv.py". Programmer la méthode de la puissance *inverse* pour calculer la plus petite valeur propre de A. Tracer l'évolution de cette valeur propre calculée en fonction du nombre d'itérations. Donner l'ordre et la vitesse de convergence.
3. Créer un fichier "puissance-inv-1.py". On peut montrer que les valeurs propres de A sont comprises entre 0 et 2. Mettre en place un algorithme permettant de calculer la valeur propre de A la plus proche de 1.

In [21]:

```
import numpy as np
import matplotlib.pyplot as plt

# Fonction methode de la puissance (si inverse=0) ou puissance inverse (si inver
se=1)
# A : matrice
# v : vecteur initial
# eps : tolerance
# inverse = 1 pour la methode de la puissance inverse et 0 sinon
def Puissance(A,v,eps,inverse):
    v/=np.linalg.norm(v,1)
    l=[np.linalg.norm(np.dot(A,v),1)/np.linalg.norm(v,1)]

    cpt=0

    while cpt<10000:
        cpt += 1
        if inverse:
            v=np.linalg.solve(A,v)
        else:
            v=np.dot(A,v)
        v/=np.linalg.norm(v,1)
        # Attention pour le calcul de lambda, on divise par v[0] mais il faut q
u'il soit non nul !
        l.append((np.dot(A,v)[0]/v[0])[0])

        if np.linalg.norm(np.dot(A,v)-l[-1]*v,1)<eps:
            break
    return l
```

In [22]:

```
N=10

A=2*np.eye(N)-np.diag(np.ones((N-1)),1)-np.diag(np.ones((N-1)),-1)

v=np.ones((N,1))
v[0]=0

L=np.linalg.eigvals(A)
eps=1e-5

# Methode de la puissance

l=Puissance(A,v,eps,0)

print 'max des val propres de A avec eig =', max(L)
print 'max des val propres de A avec puissance =', l[-1]

plt.plot(l,'b+-')
plt.title('evolution de lambda')
plt.show()

plt.plot(np.arange(len(l)-2),np.log(np.abs(l[1:-1]-l[-1]))/np.log(np.abs(l[0:-2]
-l[-1])),'r+-')
plt.title('vitesse de convergence')
plt.axis([0,len(l)-1,0,2])
plt.show()
```

```
# Methode de la puissance inverse

l=Puissance(A,v,eps,1)

print 'min des val propres de A avec eig =', min(L)
print 'min des val propres de A avec puissance inverse =', l[-1]

plt.plot(l,'b+-')
plt.title('evolution de lambda')
plt.show()

plt.plot(np.arange(len(l)-2),np.log(np.abs(l[1:-1]-l[-1]))/np.log(np.abs(l[0:-2]
-l[-1])),'r+-')
plt.title('vitesse de convergence')
plt.axis([0,len(l)-1,0,2])
plt.show()

# Methode de la puissance inverse avec tau=1

tau=1
l=Puissance(A-tau*np.eye(N),v,eps,1)

print 'val propre de A proche de tau =', tau, 'avec eig =', np.min(np.abs(L-tau
))+1
print 'val propre de A proche de tau =', tau, 'avec puissance =', l[-1]+tau

plt.plot(l,'b+-')
plt.title('evolution de lambda')
plt.show()

plt.plot(np.arange(len(l)-2),np.log(np.abs(l[1:-1]-l[-1]))/np.log(np.abs(l[0:-2]
-l[-1])),'r+-')
plt.title('vitesse de convergence')
plt.axis([0,len(l)-1,0,2])
plt.show()
```

```
max des val propres de A avec eig = 3.91898594723
max des val propres de A avec puissance = 3.91897645033
min des val propres de A avec eig = 0.081014052771
min des val propres de A avec puissance inverse = 0.0810047679711
val propre de A proche de tau = 1 avec eig = 1.169169974
val propre de A proche de tau = 1 avec puissance = 1.16916356046
```

Exercice 2

Dans ces exemple, on explore les limitations de la méthode de la puissance. Soit A la matrice

$$A = \begin{pmatrix} 0.5172 & 0.5473 & -1.224 & 0.8012 \\ 0.5473 & 1.388 & 1.353 & -1.112 \\ -1.224 & 1.353 & 0.03642 & 2.893 \\ 0.8012 & -1.112 & 2.893 & 0.05827 \end{pmatrix}.$$

Les valeurs propres de A sont $\mu_1 = -4$, $\mu_2 = 3$, $\mu_3 = 2$ et $\mu_4 = 1$. Les vecteurs propres correspondant à μ_1 et μ_2 sont respectivement.

$$w_1 = \begin{pmatrix} 0.3225 \\ -0.3225 \\ 0.6451 \\ -0.6129 \end{pmatrix}, \quad w_2 = \begin{pmatrix} -0.1290 \\ 0.1290 \\ 0.7419 \\ -0.6451 \end{pmatrix}$$

1. Créer un fichier "puissance-lim.py"
2. Programmer la méthode de la puissance avec comme vecteur initial $x_0 = (1\ 0\ 0\ 0)^T$. Etudier l'évolution de la valeur propre calculée et tracer $\|x_k\|_2$ en fonction de k.
3. On part maintenant du vecteur initial $x_0 = (1\ 1\ 1\ 1)^T$ (presque orthogonal à w_1). Même question. Expliquer le phénomène observé.
4. On part de $x_0 = (1\ 1\ 0\ 0)^T$ (orthogonal à w_1 et w_2). Même question.

In [36]:

```
def Puissance_lim(A,v,eps,inverse):
    v/=np.linalg.norm(v,1)
    l=[np.linalg.norm(np.dot(A,v),1)/np.linalg.norm(v,1)]
    V=[np.linalg.norm(v,2)]

    cpt=0

    while cpt<10000:
        cpt += 1
        if inverse:
            v=np.linalg.solve(A,v)
        else:
            v=np.dot(A,v)
        v/=np.linalg.norm(v,1)
        l.append((np.dot(A,v)[0]/v[0]))
        V.append(np.linalg.norm(v,2))

        if np.linalg.norm(np.dot(A,v)-l[-1]*v,1)<eps:
            break

    return l,V
```

In [37]:

```
a0=[0.5172, 1.388, 0.03642, 0.05827]
a1=[0.5473, 1.353, 2.893]
a2=[-1.224, -1.112]
a3=[0.8012]

A=np.diag(a1,1)+np.diag(a2,2)+np.diag(a3,3)
A=A+A.T+np.diag(a0)

L=np.linalg.eigvals(A)

eps=1e-8

v=np.array([1, 0, 0, 0])
#v=np.ones((4,1))
#v=np.array([1, 1, 0, 0])

l,V=Puissance_lim(A,v,eps,0);

print 'max des val propres de A avec eig =', max(np.abs(L))
print 'max des val propres de A avec puissance inverse =', l[-1]

plt.plot(l,'b+-')
plt.title('evolution de lambda')
plt.show()

plt.plot(V,'r+-')
plt.title('evolution de la norme du vecteur propre')
plt.show()
```

```
max des val propres de A avec eig = 3.99567071127
max des val propres de A avec puissance inverse = -3.9956707166
```

Exercice de synthèse

Soit $A \in M_N(\mathbb{R})$ la matrice définie par

$$A = \begin{pmatrix} 4 & 1 & 0 & \dots & 0 \\ 1 & 4 & 1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & 1 & 4 & 1 \\ 0 & \dots & 0 & 1 & 4 \end{pmatrix}$$

Dans ce qui suit, on considérera le cas $N = 20$ et $N = 200$.

1. Programmer la méthode de la puissance pour calculer la plus grande valeur propre de A. Tracer l'évolution de la valeur propre calculée en fonction du nombre d'itérations. Donner l'ordre et la vitesse de convergence.
2. Programmer la méthode de la puissance inverse pour calculer la plus petite valeur propre de A. Tracer l'évolution de cette valeur propre calculée en fonction du nombre d'itérations. Donner l'ordre et la vitesse de convergence.
3. On peut montrer que les valeurs propres de A sont comprises entre 2 et 6. Mettre en place un algorithme permettant de calculer la valeur propre de A la plus proche de 4.