# The script below computes evolution of probability distribution for the following reaction network:

```
      /* B
X<->A/   |
     *\  /
       \C*
```

In [72]:
```
from scipy import *
from scipy import linalg
from scipy import sparse
```

In [210]:
```
Nq=3 # The number of "quanta" allowed in this network (equal to the sum of
Nx=Nq+1 # The number of states for the molecule X
Nc=2**3 # The number of states in the cycle A->B->C->A...
t=0.5 # time elapsed in seconds
kab=4.5 # s^-1
kbc=2.5 # s^-1
kca=0.5 # s^-1
kxa=0.8 # s^-1
kax=0.3 # s^-1
```

In [211]:
```
# Defining an outer product state space
# 8=2**3 (becuase A, B, and C are 2-state)
def xabc_index(x,a,b,c):
    if(x<0 or x>Nq or a<0 or a>1 or b<0 or b>1 or c<0 or c>1):
        raise IndexError("One of indices is out of range in xabc_index(a,b
    return 8*x+4*a+2*b+c
```

In [214]:
```
xabc_index(0,0,0,0), xabc_index(3,1,1,1)
```

Out[214]: $(0, \quad 31)$

In [258]:
```
# This is based on the master equation of this system (my notes)
R=zeros([8*Nx,8*Nx])
for x in r_[0:Nx]:
    for a in [0,1]:
        for b in [0,1]:
            for c in [0,1]:
                if((x+a+b+c)>Nq):
```

```
                    continue
                i=xabc_index(x,a,b,c)
                # X->A, Part 1
                if(a==0):
                    R[i,i]+=-kxa*x
                # X->A, Part II
                if(x!=Nq and a==1):
                    j=xabc_index(x+1,a-1,b,c)
                    R[i,j]+=kxa*(x+1)
                # A->X Part I
                if(a==1):
                    R[i,i]+=-kax
                # A->X, Part II
                if(x!=0 and a==0):
                    j=xabc_index(x-1,a+1,b,c)
                    R[i,j]+=kax
                # A->B, Part I
                if(a==1 and b==0):
                    R[i,i]+=-kab
                # A->B, Part II
                if(a==0 and b==1):
                    j=xabc_index(x,a+1,b-1,c)
                    R[i,j]+=kab
                # B->C, Part I
                if(b==1 and c==0):
                    R[i,i]+=-kbc
                # B->C, Part II
                if(b==0 and c==1):
                    j=xabc_index(x,a,b+1,c-1)
                    R[i,j]+=kbc
                # C->A, Part I
                if(c==1 and a==0):
                    R[i,i]+=-kca
                # C>A, Part II
                if(c==0 and a==1):
                    j=xabc_index(x,a-1,b,c+1)
                    R[i,j]+=kca
print R[0:12,0:12]
```

```
[[ 0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0. ]
 [ 0.  -0.5  2.5  0.   0.   0.   0.   0.   0.   0.   0.   0. ]
 [ 0.   0.  -2.5  0.   4.5  0.   0.   0.   0.   0.   0.   0. ]
 [ 0.   0.   0.  -0.5  0.   4.5  0.   0.   0.   0.   0.   0. ]
 [ 0.   0.5  0.   0.  -4.8  0.   0.   0.   0.8  0.   0.   0. ]
 [ 0.   0.   0.   0.   0.  -4.8  2.5  0.   0.   0.8  0.   0. ]
 [ 0.   0.   0.   0.5  0.   0.  -2.8  0.   0.   0.   0.8  0. ]
 [ 0.   0.   0.   0.   0.   0.   0.  -0.3  0.   0.   0.   0.8]
 [ 0.   0.   0.   0.   0.3  0.   0.   0.  -0.8  0.   0.   0. ]
 [ 0.   0.   0.   0.   0.   0.3  0.   0.   0.  -1.3  2.5  0. ]
 [ 0.   0.   0.   0.   0.   0.   0.3  0.   0.   0.  -3.3  0. ]
```

```
 [ 0.   0.   0.   0.   0.   0.   0.   0.3  0.   0.   0.  -1.3]]
```

In [259]:
```
print sum(sum(R,axis=0))
```

```
1.16573417586e-15
```

In [260]:
```
ONES=mat(ones(8*Nx))
print ONES, ONES.shape
```

```
[[ 1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.
   1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.]] (1, 32)
```

In [261]:
```
sum(ONES*R)
```

Out[261]:
```
1.1657341758564144e-15
```

In [262]:
```
T=linalg.expm2(R*t)
```

In [263]:
```
print T[0:12,0:12]
```

```
[[  1.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
    0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
    0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00]
 [  0.00000000e+00   8.23858086e-01   6.28521703e-01  -8.97351414e-17
    4.28028777e-01  -9.93132100e-17   4.67170942e-16   4.39860031e-17
    6.90916874e-02  -6.60283086e-16   8.72801679e-17  -5.77027427e-17]
 [  0.00000000e+00   8.56057554e-02   3.21040724e-01   4.56695826e-17
    4.03420571e-01  -2.17915307e-16   1.07983445e-16  -2.98779597e-17
    1.28678206e-01   5.01266648e-17  -4.86440542e-17   7.16573703e-17]
 [  0.00000000e+00   1.55054479e-16   2.46457839e-16   8.22546598e-01
   -1.21063029e-16   7.44964961e-01   4.11732937e-01   1.93622966e-16
    1.18058787e-15   1.70636671e-01   1.07988222e-01  -3.12082087e-17]
 [  0.00000000e+00   8.28715103e-02   4.75587530e-02  -3.33668776e-17
    1.23426534e-01   8.88600017e-17   2.38235381e-17  -1.54462545e-17
    1.20330979e-01   7.96752210e-17   6.19896451e-17  -3.27256624e-17]
 [  0.00000000e+00  -2.73258380e-18  -1.23654539e-16   4.57481041e-02
    6.27855065e-17   1.22066970e-01   2.10627358e-01  -1.16153408e-16
   -4.88934529e-17   1.07509734e-01   1.00802446e-01  -5.15876681e-17]
 [  0.00000000e+00   9.93800770e-17   4.88593263e-17   1.19000692e-01
    6.10180083e-17   8.21991960e-02   2.85961680e-01   6.37517357e-17
    8.23670871e-17   2.12293988e-02   9.72200556e-02   3.18027932e-17]
 [  0.00000000e+00  -1.89408914e-16   6.28037921e-16   5.49232063e-16
    1.27907844e-15  -2.20118988e-15  -1.66430444e-15   8.82971312e-01
   -4.83439469e-16   1.35247418e-15   3.68151098e-16   2.78016989e-01]
 [  0.00000000e+00   7.66464817e-03   2.87882031e-03   2.94267977e-17
    4.51241173e-02   4.03571576e-17  -1.14067271e-17   3.48905782e-18
    6.81899127e-01  -6.17230753e-18  -1.72037248e-17   5.24673452e-18]
 [  0.00000000e+00  -2.36057689e-16  -1.63380741e-16   5.66099703e-03
    4.65095740e-17   4.07767483e-02   5.21983421e-02  -1.90260341e-16
```

```
  -6.47397697e-17   5.64870953e-01   4.31787189e-01  -1.76689103e-16]
 [  0.00000000e+00  -2.70314311e-17  -6.38037792e-18   6.76141497e-03
    2.13031050e-17   6.23882501e-03   3.59969228e-02   2.24369127e-17
  -2.98860443e-17   6.50228225e-02   2.24048378e-01  -1.63017451e-17]
 [  0.00000000e+00   9.85911054e-17   1.62215029e-17   1.96344339e-17
    5.13244496e-16   5.72760395e-17  -3.93197181e-16   1.04256371e-01
  -1.36492655e-16  -1.89139309e-17  -2.64388279e-16   5.76159280e-01]]
```

In [264]:
```
p0=zeros(8*Nx)
i=xabc_index(3,0,0,0)
p0[i]=1
p0=mat(p0)
print p0, p0.shape
```

```
[[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
   0.  0.  0.  0.  0.  0.  1.  0.  0.  0.  0.  0.  0.  0.]] (1, 32)
```

In [265]:
```
pt=T*p0.T
print pt, pt.shape
```

```
[[  0.00000000e+00]
 [ -3.87865978e-16]
 [  3.72428067e-16]
 [ -6.67282026e-18]
 [ -8.42491537e-17]
 [  2.23855839e-16]
 [  3.55358433e-17]
 [  1.68732351e-03]
 [  9.84144777e-18]
 [ -1.18250376e-16]
 [ -1.18768095e-16]
 [  2.03270262e-02]
 [  9.99155896e-18]
 [  3.69035811e-02]
 [  6.54101778e-02]
 [  0.00000000e+00]
 [ -1.75637985e-17]
 [  1.12185339e-01]
 [  2.20241422e-01]
 [  0.00000000e+00]
 [  2.22723317e-01]
 [  0.00000000e+00]
 [  0.00000000e+00]
 [  0.00000000e+00]
 [  3.20521813e-01]
 [  0.00000000e+00]
 [  0.00000000e+00]
 [  0.00000000e+00]
 [  0.00000000e+00]
 [  0.00000000e+00]
```

```
[  0.00000000e+00]
[  0.00000000e+00]] (32, 1)
```

In [267]:
```
def find_p_x(x, pt):
    ptx=0
    for a in [0,1]:
        for b in [0,1]:
            for c in [0,1]:
                if((x+a+b+c)!=(Nx-1)):
                   continue
                i=xabc_index(x,a,b,c)
                #print "find_p_x", i
                ptx+=pt[i,0]
    return ptx

def find_p_a(pt):
    pta=0
    a=1
    for x in r_[0:Nx]:
        for b in [0,1]:
            for c in [0,1]:
                if((x+a+b+c)!=(Nx-1)):
                   continue
                i=xabc_index(x,a,b,c)
                #print "find_p_x", i
                pta+=pt[i,0]
    return pta

def find_p_b(pt):
    ptb=0
    b=1
    for x in r_[0:Nx]:
        for a in [0,1]:
            for c in [0,1]:
                if((x+a+b+c)!=(Nx-1)):
                   continue
                i=xabc_index(x,a,b,c)
                #print "find_p_x", i
                ptb+=pt[i,0]
    return ptb

def find_p_c(pt):
    ptc=0
    c=1
    for x in r_[0:Nx]:
        for a in [0,1]:
            for b in [0,1]:
                if((x+a+b+c)!=(Nx-1)):
                   continue
                i=xabc_index(x,a,b,c)
```

```
                #print "find_p_x", i
                ptc+=pt[i,0]
    return ptc
```

In [270]:
```
ptx=[find_p_x(x,pt) for x in r_[0:Nx]]
pa=find_p_a(pt)
pb=find_p_b(pt)
pc=find_p_c(pt)
```

In [271]:
```
p_analyt=[]
p_analyt.extend(ptx)
p_analyt.extend([pa, pb, pc])
print p_analyt
```

```
[0.001687323512088279, 0.12264078507458409, 0.55515007879166167,
0.3205218126216664, 0.32672439967797662, 0.30766594955383336,
0.17110327024528463]
```

In [ ]:
```

```