```python
#!/usr/bin/python
# -*- coding: utf-8 -*-
import numpy as np
import scipy.optimize as op



def Av1(s1,s2,s3,s4,n1,n2,n3,n4):

    """
    Parameters :
    -----------
    si are the schoechimetric numbers associated with the reaction :
    s1 A + s2 B --> s3 C + s4 D

    ni are the associated quantities (mole)

    """


    xia = n1/s1
    xib = n2/s2

    if xia >= xib : #test reactif limitant
        xif = xib
    else :
        xif = xia

    n1f = n1 - s1*xif
    n2f = n2 - s2*xif
    n3f = n3 + s3*xif
    n4f = n4 + s4*xif

    print('%iA + %iB -->  %iC + %iD' %(s1,s2,s3,s4))
    print( '%0.2e  + %0.2e -->  %0.2e  + %0.2e '          % (n1f,n2f , n3f, n4f ))

    return n1f,n2f , n3f, n4f




n1,n2,n3,n4 = Av1(1,1,1,1,1,2,0,0)

#Fonction plus generale

n = [1,2,0,0] #Quantité de matière des espèces
nu = [-1,-1,1,1] #Nombres stoechiométriques

def Av1_gen(n,nu):
    """
    Parameters :
    -----------
    nu is list of the schoechimetric numbers associated with the reaction :


    n is a list of the associated quantities (mole)

    Returns :
    ---------   list of quantities in the final state
    """
    R_ind = []
    Xi = []
    for i in range(len(n)):
        if nu[i]<0:
            Xi.append(n[i]/nu[i])
```

```
    xi = min(Xi)

    Ef = []
    for i in range(len(n)):
        P.append(n[i]+nu[i]*xi)

    return P



def Av2(s1,s2,s3,s4,c1,c2,c3,c4,K):
    """
    Parameters :
    -----------
    si are the schoechimetric numbers associated with the reaction :
    s1 A + s2 B --> s3 C + s4 D

    ci are the associated concentrations (mol/L)

    """

    def f(x):
        return (c3+s3*x)**s3*(c4+s4*x)**s4-K*(((c1-s1*x)**s1)*((c2-s2*x)**s2))

    sol=op.fsolve(f,0.0)
    sel = []
    for i in sol :
        if i>0:
            sel.append(i)

    xi = min(sel)
    return c1-s1*xi,c2-s2*xi,c3+s3*xi,c4+s4*xi, xi
```

```
    return x

a = Av2(1,1,1,1,1,2,0,0,100)


def Av2_bis(a,b,c,d,Cai,Cbi,Cci,Cdi,K, pas=0.001):
    """
    La reaction considérée est aA + bB -> cC + dD

    Parameters :
    -----------
    a,b,c,d are the schoechimetric numbers associated with the reaction
    Ca,Cb,Cc,Cd are the associated concentration (mole/L)
    K equilibrium constant

    """


    # xmax est l'avancement maximal si l'un ou les deux produits sont totalement consommés
    # Ca - a*xmax = 0 ==> xmax = ni_a / a
    # Cb - b*xmax = 0 ==> xmax = ni_b / a
    # xmax est le miminum des xmax obtenus ci-dessus
    xmax = min(float(Cai) / a, float(Cbi) / b)

    # pour se passer de la méthode utilisant un solveur
    # on va incrémenter l'avancement avec un pas, plus le pas est
    # petit, plus le résultat sera précis.
```

```
    # pour cela, on utilise une boucle while avec comme conditions de sortie primaire
    # Qr = K et comme garde fou, x<xmax


    Qr = ( Cci**c * Cdi **d) / (Cai**a * Cbi**b)
    x = 0

    if Qr > K :
        print('réaction dans le sens indirect')
    else:
        while (Qr<K and x<xmax):  # tant que Qr <k et tant que nous n'avons pas atteint la limite du tableau

            x = x + pas
            Ca = Cai - a * x     # la concentration du réactif A
            Cb = Cbi - b * x     # la concentration du réactif B va se réduire de b*x
            Cc = Cci + c * x           # concentration du produit C va augmenter de c*x
            Cd = Cdi + d * x           # concentration du produit D va augmenter de d*x
            # On calcule Qr par la formule ci-dessous si A+b est différent de a+c il faudra saisir le volume de la solution
            Qr = float( Cc**c *Cd**d) / (Ca**a * Cb**b)
                                # passer à l'lément suivant du tableau

    return Ca,Cb,Cc,Cd,Qr,x



b = Av2_bis(1,1,1,1,1,2,0,0,100)
```