Etude de l'évolution d'un système chimique (version élève)

Ce programme permet d'étudier l'évolution des quantités de matière des réactifs et produits d'une réaction dont l'équation est du type : aA + bB –> cC + dD où a, b, c et d sont les nombres stoechiométriques respectifs des espèces chimiques A, B, C et D.

Le programme doit tout d'abord demander les valeurs des nombres stoechiométriques, pour ensuite demander les quantités de matière initiales des réactifs A et B et des produits C et D.

In [ ]:
```
# rattachement des librairies gérant les tracés
# de courbes et certains outils mathématiques
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
```

Entrée des nombres stoechiométriques

Sur le modèle de la ligne de code 2, ajouter les lignes de code nécessaires (lignes 3, 4 et 5) pour entrer les valeurs des nombres stoechiométriques b, c et d.

In [ ]:
```
#Nombres stoechiométriques
a=2   #par exemple !! Donc à adapter...
```

In [ ]:
```
# Affichage de l'équation de la réaction
print("l'équation étudiée est du type : ",
      a," A    +    ",b," B    -->     ",
      c," C      +    ",d," D")
```

Entrée des valeurs de quantités de matière initiales

Les valeurs des quantités de matière initiales des réactifs et des produits (exprimées en mole) seront stockées dans des variables notées n_0

(ex : nA_0 pour l'espèce chimique A).

Sur le modèle de la ligne de code 2, ajouter les lignes de code nécessaires (lignes 3, 4 et 5) pour entrer les quantités de matière initiales des autres espèces chimiques en jeu. Attention de bien entrer les valeurs en mol! Vous pourrez par exemple taper 2.5e-3 pour 2,5 mmol

In [ ]:
```
# Quantités de matières initiales
nA_0 =
nB_0 =
```

In [ ]:
```
#Initialisation des variables

# Initialisation de la chaine de caractère correspondant
# au réactif limitant
Rlimitant =''
```

```
# Avancement initial
x=0

# Pas d'avancement (on augmentera progressivement x de la valeur dx)
dx=0.001

# Création des listes contenant les quantités de matière
# et initialisation de ces listes avec la valeur initiale
nA=[nA_0]
nB=[nB_0]
nC=[nC_0]
nD=[nD_0]

# Création et nitialisation de la liste contenant l'avancement
X=[x]
```

Calculs des quantités de matière en cours d'avancement

Sur le modèle de la ligne 5, écrire les lignes de code 6, 7 et 8 permettant de calculer les quantités de matière du réactif B, ainsi que des produits C et D.

NOTE CODAGE : l'instruction "nA.append(nA_0-a*x)" permet d'ajouter la valeur indiquée entre parenthèses à la fin de la liste nA.

Détermination du réactif limitant

Compléter les tests des lignes de code 11 et 12 en choisissant parmi : <0, <=0, >0 et >=0.

Compléter la ligne de code 13 en choisissant l'opérateur logique adéquat parmi : and (ET logique) et or (OU logique).

NOTE CODAGE : l'indice -1 permet d'avoir accès à la dernière valeur de la liste.

Affichage du nom du réactif limitant et de l'avancement maximal

Il sera intéressant de modifier en ligne 18, le nombre de chiffres après la virgule afin de respecter le nombre de chiffres significatifs pour l'avancement x.

In [ ]:

```
# Calculs des quantités de matière en cours d'avancement
while nA[-1] > 0 and nB[-1] > 0:
    x=x+dx
    X.append(x)
    nA.append(nA_0-a*x)




#Détermination du réactif limitant
if nA[-1]                       : Rlimitant = 'A'
if nB[-1]                       : Rlimitant = 'B'
if nA[-1]<=0          nB[-1]<=0 :
    Rlimitant='A et B : le mélange est stoechiometrique'
```

```
#Affichage des résultats
print('Le réactif limitant est ',Rlimitant,
      '\n Avancement maximum : ','{0:.4f}'.format(x),
      'mol' )
#{0:.4f} permet daffficher un nombre arrondi à
# 4 chiffres après la virgule (ici).
```

Affichage des courbes permettant de suivre l'évolution des quantités de matière

La ligne de code 2 ci-dessous permet d'afficher le graphe de l'évolution de la quantité de matière de A en fonction de l'avancement X.

Compléter les lignes 3, 4 et 5 pour afficher les courbes correspondant aux évolutions des quantités de matière des espèces chimiques B, C et D en fonction de l'avancement.

NOTE CODAGE : la commande plt.plot peut être enrichie de divers arguments (comme ici avec r- = r pour red et - pour ligne) :

Couleur : r (red), k (black), b (blue), y (yellow), g (green)

Marqueur : o (gros point), - (ligne), . (pointillé), x (croix), s (square), v (triangle)

lw signifie linewidth (largeur de la ligne)

In [ ]:

```
plt.figure(figsize=(10,10))
plt.plot(X,nA,'r-',lw=1,label='nA')



plt.grid()
plt.xlabel('x (mol)')
plt.ylabel('n (mol)')
plt.legend()
plt.show()
```

Modélisation des droites obtenues

Les lignes de code suivantes vont permettre de modéliser chacune des 4 droites obtenues sur le graphe ci-dessus. Ces droites sont de type linéaire ou affine et peuvent être modélisées avec un polynôme de degré 1 : mx+p (où x est à la puissance 1). Les résultats des quatre modélisations sont ensuite affichés pour analyse.

Compléter les lignes de code 2, 3 et 4 (sur le modèle de la ligne 1) pour modéliser les courbes concernant l'évolution des quantités de matière des espèces chimiques B, C et D.

NOTE CODAGE : .0f en lignes 8, 13, 18 et 23 signifie qu'il y aura 1 seul chiffre significatif (pas de chiffre après la virgule), alors que .3f en lignes 9, 14, 19 et 24 signifie qu'il y aura 3 chiffres après la virgule.

In [ ]:

```
Amodel=np.polyfit(X, nA,1)



print ("la droite représentant l'évolution de nA"
       " en fonction de x a pour équation : nA = ",
```

```
       '{0:.0f}'.format(Amodel[0]),'{:5}'.format("x +"),
       "{0:.3f}".format(Amodel[1]) )

print ("la droite représentant l'évolution de nB"
       " en fonction de x a pour équation : nB = ",
       "{0:.0f}".format(Bmodel[0]),'{:5}'.format("x +"),
       "{0:.3f}".format(Bmodel[1]) )

print ("la droite représentant l'évolution de nC"
       " en fonction de x a pour équation : nC = ",
       "{0:.0f}".format(Cmodel[0]),'{:5}'.format("x +"),
       "{0:.3f}".format(Cmodel[1]) )

print ("la droite représentant l'évolution de nD"
       " en fonction de x a pour équation : nD = ",
       "{0:.0f}".format(Dmodel[0]),'{:5}'.format("x +"),
       "{0:.3f}".format(Dmodel[1]))
```

Commenter les équations des courbes modélisées....