SA - EXERCICE 4

AUTOCATALYSE BIMOLECULAIRE

$$A + B \rightarrow 2\,B \qquad k \qquad (r\ 5)$$

Nous vous invitons en un premier temps à programmer vous même le calcul des concentrations au cours du temps pour la réaction (r 5). Il suffit pour cela d'adapter l'un des programmes précédents. Adoptez la démarche la plus simple et intuitive qui consiste à intégrer numériquement le système des 2 équations différentielles :

$$dA/dt = -k\,AB \qquad (21)$$

$$dB/dt = k\,AB \qquad (22)$$

Contentez-vous pour l'instant de ces deux concentrations. Nous vous suggérons juste de tester votre modèle avec les paramètres, du modèle et de calcul, suivants :

1. Paramètres

k / mol^{-1}.L.s^{-1}	**A_0** / mol.L^{-1}	**B_0** / mol.L^{-1}	***Fin*** / ***s***	***Incrément*** / ***s***	***Pas max*** / ***s***
10^3	10^{-3}	10^{-7}	20	0.1	10^{-2} + **Auto**
0.5×10^3	10^{-3}	10^{-7}	20	0.1	id.
2×10^3	10^{-3}	10^{-7}	20	0.1	id.
10^3	10^{-3}	10^{-6}	20	0.1	id.
10^3	10^{-3}	10^{-4}	20	0.1	id.
10^3	10^{-3}	10^{-3}	5	10^{-2}	id.
10^3	10^{-3}	5×10^{-2}	2	10^{-2}	id.

Si votre programme fonctionne correctement, vous devez retrouver l'essentiel des propriétés décrites dans le cours.

Autocat.cpp

télécharger autocat.cpp télécharger autocat.sac

Voici maintenant une version un peu plus complète, dont nous donnons ici les éléments qui méritent un commentaire :

```
1. //------------------------------------------------------
2. Sa_data S;
3. //------------------------------------------------------
4. void eqdiff (Sa_data x, Sa_data* y, Sa_data* dy)
5. {
6.    dy[0] = -p[0]*y[0]*(S - y[0]); // dA/dt
7. }
8. //------------------------------------------------------
9. void fappel()
10 {
11    S = ca[1][0]+ca[0][0];

12    srkvi(n_diff, &ca[first_var], ind, npt, h0, tol, iset, jacob,
h_compt, c_min);

13    Sa_data r_max = 0;
14    for (int i = 0; i < npt; ++i)
15    {
16       ca[1][i] = S - ca[0][i]; // B

17       ca[2][i] = p[0]*ca[0][i]*ca[1][i]; // r
18       if (ca[2][i] > r_max)
19          r_max = ca[2][i];

20       ca[3][i] = (ca[0][0]-ca[0][i])/ca[0][0]; // chi
21    }

22    for (int i = 0; i < npt; ++i)
23       if (r_max != 0)
24          ca[4][i] = ca[2][i]/r_max; // r_norm
25       else
26          ca[4][i] = -1; // code d'erreur
27 }
28 //------------------------------------------------------
```

Explications

Nous avons choisi, là encore, de n'utiliser qu'une équation différentielle (ligne 6), pour calculer A, et de calculer B par l'équation de conservation (ligne 16).

Comme dans l'exercice 3, nous avons déclaré une variable globale S (ligne 2), à l'extérieur de et avant toute fonction qui l'utilise. Elle contiendra B_0+A_0 (ligne 11).

Dans la première boucle `for` (ligne 14), nous calculons, en plus de la concentration B, la vitesse non normalisée, r (ligne 17), et l'avancement normalisé, `chi` (ligne 20). Il n'est pas possible de calculer à ce stade la vitesse normalisée, car nous ne connaissons pas encore le facteur de normalisation. Pour cela, nous déclarons **avant la boucle** (ligne 13), une variable locale `r_max`, que nous initialisons à zéro. Au cours du déroulement de la boucle, r (`ca[2][i]`) est comparé à `r_max` (ligne 18). Si il est supérieur, r_max prend cette nouvelle valeur (ligne 19). Ainsi, en fin de boucle, `r_max` contiendra bien la vitesse maximale et, comme elle a été déclarée en dehors de la boucle, elle sera encore accessible.

Ce n'est pas le cas, par exemple de la variable `i`, qui est déclarée **dans** la boucle. En fin de boucle, `i` n'existe plus. Cela explique que l'on peut redéclarer une variable `i` dans la boucle suivante. Si l'on avait besoin de récupérer la valeur de `i` après la boucle (cela peut arriver), il faudrait écrire, par exemple :

```
int i;
for (i = ... ; ... ; ...)
{
   ...;
}
quelquechose = une fonction de i;
```

On refait donc une deuxième boucle (ligne 22) pour diviser la vitesse r par `r_max`. Il y a peu de chances ici que `r_max` soit nul, cependant il faut toujours se protéger d'une éventuelle **division par zéro**. C'est pourquoi `r_max` est testé ligne 23 : si elle est différente de zéro, on calcule la vitesse normalisée (ligne 24), sinon on lui affecte une valeur -1 (ligne 26). Le choix de cette valeur est arbitraire bien sûr, mais comme les vitesses ne peuvent pas être négatives, cela nous avertira, le cas échéant, qu'il y a eu un problème sur le calcul de `r_max`.

N'oubliez pas d'affecter 1 à `n_diff` et 5 à `nv_mod`. Compilez, etc.

Simulations

A- Refaites les simulations avec les paramètres du tableau 1. Vous pouvez maintenant compléter votre exploration en traçant dans chaque cas la vitesse en fonction du temps, et la vitesse (normalisée ou non) en fonction de l'avancement normalisé, par exemple.

B- Revenez aux valeurs de la première ligne du tableau 1. Puis mettez $k = 10^{-3}$ $mol^{-1}.L.s^{-1}$. C'est une valeur plausible.

Que se passe-t-il ? Apparemment rien. Et vous êtes conforté dans cette idée si vous regardez le fichier des données calculées (bouton **d** de la fenêtre principale de Sa) : la concentration de A n'a pas changé d'un seul chiffre après le point ! Apparemment, oui.

Refermez la fenêtre ouverte avec **d**. Dans l'onglet *Général*, volet *Fichier de commande*, cochez la case *Ultra-précision*. Ce mode ne change pas la précision des calculs, mais permet simplement d'afficher et de récupérer toute la précision des données. Relancez la simulation et regardez de nouveau les données calculées : vous devez voir que A commence à varier au niveau de la 9ème décimale. Il s'est donc bien passé quelque chose. Mais c'est très lent, c'est la période d'induction de l'autocatalyse : on a l'impression qu'il ne se passe rien. Augmentez progressivement la valeur de *Fin* (et de *Incrément*, en accord), par exemple par puissances de 10, jusqu'à retrouver une courbe qui ressemble à la courbe initiale, à l'échelle de temps près.

C- Revenez aux valeurs de la première ligne du tableau 1 et faites l'expérience inverse : k = 10^6 $mol^{-1}.L.s^{-1}$ par exemple. Vous devez observer une curieuse marche d'escalier. Si vous regardez les données, vous vous apercevez que A passe d'un seul coup d'une valeur très proche de sa valeur initiale à pratiquement zéro, sans aucune valeur intermédiaire : cela s'appelle une discontinuité et révèle sans doute un problème d'intégration numérique. Pour vous en assurer, divisez *Incrément* par 2, par exemple, en gardant toujours le pas d'intégration en mode *Auto*. Vous devez observer un déplacement de la marche d'escalier : c'est le signe évident d'un problème d'intégration numérique.

L'intégrateur `srkvi` ajuste en interne le pas d'intégration en fonction de la difficulté ou de la facilité d'intégration, sans pouvoir dépasser la valeur de *pas max*. Si cette valeur est trop grande, et l'intégration difficile (c'est le cas de l'autocatalyse), il ne peut plus régler correctement le pas réel et l'intégration est incorrecte. La valeur du *pas max* est le paramètre d'intégration le plus important. Ce n'est pas l'intégrateur qui est en cause.

Le mode *Auto* met la valeur de *pas max* automatiquement à la valeur de *Incrément*. Cela permet dans la plupart des cas d'obtenir une intégration à la fois correcte et la plus rapide possible, mais on doit y renoncer dans les cas difficiles.

pas max est enregistré dans le fichier .sac sous le nom `h0`, ainsi que *tolérance* (`tol`) .Ce sont aussi les noms des paramètres correspondants qui figurent dans les arguments de `srkvi` :

```
srkvi(n_diff, &ca[first_var], ind, npt, h0, tol, iset, jacob, h_compt,
c_min);
```

Pour y remédier, il faut diminuer le *pas max*, mais aussi *Fin* et *Incrément*, car on doit s'attendre évidement à une échelle de temps beaucoup plus courte. Puisqu'on a multiplié k par 10^3, on doit s'attendre à une échelle de temps divisée par 10^3. Essayons donc les valeurs :

Fin = 2×10^{-2} s

Incrément = 2×10^{-4} s

pas max = 2×10^{-4} s

La courbe obtenue semble plus correcte. Pour s'en assurer, divisons encore *pas max* par 2, par exemple. On constate que la courbe ne change plus : l'intégration est correcte. On peut d'ailleurs maintenant remettre le mode *Auto* sans problème.

Comme toute méthode numérique, l'intégration a des limites et doit être correctement paramétrée, en fonction du problème traité. Dans les problèmes connus à priori comme difficiles, ou en cas de doute, il faut toujours vérifier la validité des résultats obtenus.

La première règle à appliquer est de choisir un pas maximum d'intégration compatible avec l'**échelle de temps** du problème traité. Certains problèmes, comme le cas de l'autocatalyse, peuvent présenter **plusieurs échelles de temps très différentes** (période d'induction très lente par rapport à la phase active de la réaction). Il faut alors choisir le *pas max* en fonction de la phase la plus rapide.

Complément : intégration numérique