I.3. Méthodes de résolution des systèmes d'équations différentielles.

- Calcul des réacteurs continus parfaitement agitées en utilisant Excel ou autres logiciels.
- Calcul d'un réacteur tubulaire isotherme avec logiciel.

CHEMICAL REACTORS

FORMULATION MATHÉMATIQUE DE PROBLÈMES DE RÉACTEURS

Une équation différentielle pour une fonction qui ne dépend que d'une variable est appelée équation différentielle ordinaire. La variable indépendante est souvent le temps, t, mais pour les réacteurs, elle peut également être en aval d'un réacteur à écoulement piston. Un exemple d'équation différentielle ordinaire est :

$$\frac{dy(t)}{dt} = f(y), y(0) = y_0 \qquad (8.1)$$

Example: un réacteur à écoulement piston **and Batch Reactor**

$$dF = r\,dV \quad \text{or} \quad \frac{dF}{dV} = r \qquad (8.2)$$

Le débit molaire peut être lié à la vitesse dans le tube:

$$F = uAc \qquad (8.3)$$

Dans de nombreux cas, la vitesse et la section transversale du tube sont constantes. Ensuite, vous pouvez écrire

$$dF = u\,A dc, dV = A dz \qquad (8.4)$$

Si vous les mettez dans l'égaliseur. (8.2) et divisez par dz vous obtenez

$$u\frac{dc}{dz} = r, \text{ plug flow reactor} \qquad (8.5)$$

$$c(0) = c_0 \qquad (8.6)$$

Si le réacteur contient un catalyseur, alors la vitesse de réaction est exprimée comme la vitesse molaire de changement par masse de catalyseur, r. Il faut ensuite le multiplier par la masse de catalyseur par unité de volume, mais sinon la situation est la même:

$$dF = \rho_B r'\,dV \quad \text{and} \quad u\frac{dc}{dz} = \rho_B r', \text{ catalytic plug flow reactor} \qquad (8.7)$$

Dans un réacteur discontinu, soit N les moles d'un produit chimique; le changement molaire d'un produit chimique est la vitesse de réaction (changement molaire par unité de temps par unité de volume) multipliée par le volume et le temps:

$$dN = r\,V dt \qquad (8.8)$$

Cela peut également être mis dans une équation différentielle:

$$\frac{dN}{dt} = rV \qquad (8.9)$$

Si le volume du réacteur discontinu est constant, alors

$$c = \frac{N}{V} \qquad (8.10)$$

et l'équation devient

$$\frac{dc}{dt} = r, \quad \text{batch reactor} \qquad (8.11)$$

Si la réaction est

$$aA + bB \rightarrow cC + dD \qquad (8.12)$$

Exemple: réacteur à cuve à agitation continue(CSTR)

L'équilibre molaire sur ce réacteur est

$$F_{out} - F_{in} = r\,V \qquad (8.13)$$

oùV est le volume du réacteur. Les débits molaires sont liés aux débits volumétriques, Q, comme suit:

$$F = Qc \qquad (8.14)$$

Lorsque les débits volumétriques sont constants, Eq. (8.13) devient

$$Q(c_{out} - c_{in}) = r\,V \text{ ,CSTR} \qquad (8.15)$$

Dans un réacteur bien mélangé (CSTR), la concentration sortant du réacteur est la même que la concentration dans le réacteur. Puisque l'expression du taux est fonction de c, elle est fonction de cout. Ensuite, Eq. (8.15) devient une équation algébrique dans une variable (ici) pour cout. Vous pouvez résoudre ce problème en utilisant Excel ou MATLAB.

UTILISATION DE MATLAB POUR RÉSOUDRE DES ÉQUATIONS DIFFÉRENTIELLES ORDINAIRES

Exemple simple

Dans MATLAB, vous définissez le problème au moyen d'une fonction, appelée m-file. Vous dites ensuite à MATLAB de résoudre l'équation différentielle. Ainsi, vous devez préparer un fichier m qui définit l'équation, puis appeler le sous-programme «ode45» pour effectuer l'intégration. Cette méthode peut sembler mystérieuse au premier abord, car vous appelez un sous-programme, qui à son tour appelle votre fichier m.

Vous n'appelez pas directement votre fichier m lors de la résolution de l'équation différentielle. Ce processus est illustré à l'aide d'une seule équation différentielle simple:

$$\frac{dy}{dt} = -10y, \quad y(0) = 1 \qquad (8.16)$$

Intégrez cette équation de t = 0 à t = 1. La solution exacte peut être trouvée par quadrature et est

$$y(t) = e^{-10t} \qquad (8.17)$$

Étape 1 Pour utiliser MATLAB, vous devez d'abord construire un fichier m qui définit l'équation.

```
% filenamerhs.m
 Function ydot=rhs(t,y)  %compute the rhs of eqns.forany input
                          %values t, y
ydot = -10*y            %computerhs
```

Étape 2 Ensuite, pour tester la fonction, lancez la commande

```
q = rhs(0.2,3)
ydot = -30
q = -30
```

Étape 3 Ensuite, créez un fichier m qui fournit le script pour exécuter le problème. Ces commandes peuvent également être saisies dans la fenêtre de commande, mais lorsque vous avez plusieurs commandes que vous utiliserez encore et encore, il est plus pratique de créer un script qui peut exécuter toutes les commandes avec un seul message de votre part.

```
% filenamesimple.m
y0(1)=1.;               % set the initial condition to 1.
tspan=[0 1];            % integratefrom t = 0 to t = 1
[t,y]=ode45(@rhs,tspan,y0)      % call the routine ode45
plot(t,y,'r')           % plot the solution
xlabel('t')                      % label the x-axis
ylabel('y')                      % label the y-axis
```

Les résultats donnent un tableau de valeurs pour t et y et la figure 8.1.

```
t  = 0   0.0050  0.0100  0.0157 ... 0.9475 0.9738        1.0
y  = 1.0 0.9510  0.9044  0.8601 ... 0.000076   0.000059  0.000046
```

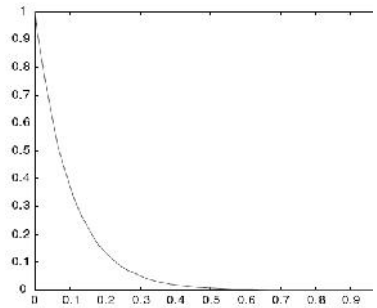

Figure 8.1. Solution to Eq. (8.16) using MATLAB.

Étape 4 La validité de votre solution dépend de plusieurs choses. Tout d'abord, vous vérifiez le fichier m «rhs» et déterminez qu'il donne le résultat correct lorsque vous insérez des valeurs pour t et y. Vous devez toujours faire cette vérification pour montrer que votre travail est correct. Vous comptez sur MATLAB pour faire son travail correctement. Vous pouvez facilement vérifier qu'il utilise correctement les heures initiale et de fin à partir de la sortie et voir que la condition initiale pour y est également correcte. Vous pouvez toujours résoudre un problème pour lequel vous connaissez la solution, et vous essayez de le rendre aussi semblable que possible à votre problème. Dans ce cas, vous n'avez pas besoin d'utiliser des méthodes numériques pour le test, car vous avez une solution exacte. Ainsi, vous pouvez comparer le résultat numérique au résultat analytique (ils sont les mêmes) et déterminer que MATLAB a fait son travail correctement. Dans les cas sans solution analytique, vous vérifiez votre fichier m «rhs» puis vous dépendez de MATLAB (et des contrôles de précision). Notez que dans les valeurs tabulées présentées précédemment, les incréments entre les points temporels sont de 0,005 au début du calcul et de 0,0272 à la fin du calcul. La routine d'intégration ajuste le pas de temps pour garantir qu'une précision spécifiée est atteinte. Vous pouvez également diminuer l'erreur numérique en définissant un paramètre (tolérance d'erreur) pour améliorer le résultat (voir l'annexe B).

Utilisation de la commande «global»

Dans l'exemple susmentionné, vous avez résolu le problème lorsque le côté droit était à -10y et vous avez simplement utilisé la valeur 10 dans le programme. Il existe deux autres façons de définir cette valeur, en utilisant la commande «global» ou en passant k

comme paramètre. Vous pourriez vouloir écrire ceci comme −ky et résoudre le problème pour plusieurs valeurs de k. Étant donné que la fonction ne sait que ce qui est défini à l'intérieur de la fonction ou ce qui y est transféré, vous devez apprendre à transférer des informations de l'espace de travail vers la fonction. Vous devrez apporter ces modifications au code utilisé précédemment. Si vous utilisez la commande «global», placez le nom de la variable à la fois dans l'espace de travail et dans le fichier m et attribuez sa valeur à un endroit ou à l'autre. Ensuite, cette valeur est accessible à partir du fichier m ou de l'espace de travail.

Étape 1 Ici, vous définissez la valeur de k dans l'espace de travail ou le programme appelant, puis elle sera disponible dans le fichier m.

```
% % filename simple2.m %compute the rhs of Eq.(8.16)
global k
k=10; % set k to 10; itisnowavailable to any
                    % m-file that has the
                    % global k statement in it.
y0=1; % set the initial condition to 1
tspan=[0 1]; % set the initial time to zero and the
                    % final time to 1
                    % call the routine ode45; use the m-file
                    % called rhs2,
                    % startwith y=y0 attspan(1) and
                    % integrateuntiltspan(2)
[t,y]=ode45(@rhs2,tspan,y0)
plot(t,y)           % plot the solution
```

Étape 2 Ensuite, ajoutez la commande globale au fichier m, appelée rhs2. (Notez que vous avez ajouté un identifiant numérique au programme appelant et au fichier m pour les aider à rester droits.)

```
% filename rhs2.m
functionydot=rhs(t,y)
                 % compute the rhs of Eq. (8.16) for any input
                 % values t, y
global k               % The m-file canaccess the numerical
                       % value of k, if
                       % it has been set to global and given a
                       % value elsewhere
disp(k)              % display the numerical value of k
                     % use thisonlywhentesting the function rhs2
ydot = -k*y;   % computerhs
```

Naturellement, vous attendez les mêmes résultats, que vous pouvez vérifier en exécutant le nouveau code.

Étape 3 Vérifiez le code comme précédemment, avec un ajout. Il est utile la première fois que vous exécutez le code d'imprimer les variables à l'intérieur de la fonction pour vous assurer qu'elles sont accessibles. Sinon, MATLAB vous indiquera que la variable n'est

pas définie. Vous pouvez également utiliser les techniques de débogage décrites dans l'annexe B.

Passer des paramètres

Une autre façon d'introduire k dans la fonction consiste à l'utiliser comme paramètre dans l'argument appelant.

```
% filename simple3.m
k=10;          % set k to 10
y0=1;          % set the initial condition to 1
tspan=[0 1];   % integratefrom t = 0 to t =  1
options = []   % the OPTIONS = [] is a place holder
               % The symbol [] is made as [] without a space.
                              % call the routine ode45;
                              % use the m-file called rhs3,
[t,y]=ode45(@rhs3,tspan,y0,options,k)
plot(t,y) % plot the solution
```

Maintenant, la fonction est

```
% filename rhs3.m
functionydot = rhs(t,y,k)
            %compute the rhs of eqns. for any input
            %values t, y
disp(k)         % display the numerical value of k; use only for
                % testing of rhs3
ydot = -k*y;    % computerhs
```

Ce programme donne les mêmes résultats que ceux donnés par simple1.m et simple2.m.

Exemple: réacteur isotherme à écoulement piston

prenez un réacteur dans lequel les composants A et C sont alimentés en quantités équimolaires, et la réaction suivante a lieu:

$$2A+C \rightarrow B \tag{8.18}$$

Vous supposez que la réaction a lieu en phase liquide et que le débit volumétrique reste constant même en cas de réaction. Les équations sont Eq. (8.5) pour chaque espèce:

$$u\frac{dC_j}{dz} = r_j, \quad j = 1, \ldots, 3 \tag{8.19}$$

oùCj est la concentration molaire de la jème espèce, u est la vitesse, z est la distance dans le tube de l'entrée et rj est la vitesse de réaction de la jème espèce, en moles par volume-temps. Ici, la vitesse de réaction est prise comme deuxième ordre:

$$\text{Vitesse de formation de B} = kC_A^2 \tag{8.20}$$

où les unités de k sont le volume par mole de temps. Les équations pour les trois espèces sont alors

$$u\frac{dC_A}{dz} = -2kC_A^2, \quad u\frac{dC_B}{dz} = +kC_A^2, \quad u\frac{dC_C}{dz} = 0 \tag{8.21}$$

A l'entrée vous prenez

$$C_A(0)=2 \text{ kmol/m}^3, \quad C_B(0)=0, \quad C_C(0)=2 \text{ kmol/m}^3 \tag{8.22}$$

And we take u= 0.5 m/s, k= 0.3 m^3/kmol s, et la longueur totale du réacteur z= 2.4 m.

Étape 1 Le programme MATLAB vous oblige à écrire une fonction qui définit le côté droit. Les paramètres d'entrée de la fonction sont les concentrations de toutes les espèces. (a) Ainsi, pour résoudre le problème, vous utilisez les variables

$$y_1 = C_A,\ y_2 = C_B,\ y_3 = C_C \tag{8.23}$$

(b) La fonction a également besoin de la vitesse, u, et de la constante de vitesse, k. La distance de l'entrée, z, tient lieu de temps et est la variable indépendante. (c) Les vitesses de réaction sont ensuite évaluées et la fonction renvoie la valeur numérique du côté droit. Le code de la fonction est

```
% rate1.m
% This functiongives the right-hand side for a simple
% reactorproblemwhichisisothermal.
Function ydot=rate1(VR,y)
% y(1) is CA, y(2) is CB, y(3) is CC
% k = 0.3 and u = 0.5
CA=y(1);
rate = 0.3*CA*CA;
ydot(1) = - 2.*rate/0.5;
ydot(2) = + rate/0.5;
ydot(3) = 0.;
ydot = ydot';
```

Étape 2 Vous testez ce fichier m en l'appelant avec des valeurs spécifiques pour yj, j = 1, 2, 3 pour vous assurer qu'il est correct. À l'aide de y (j) pour yj, exécutez les commandes suivantes.

```
y(1) = 0.2; y(2) = 0.3; y(3) = 0.4; rate1(0.1,y)
```

Vous recevez

```
ans = -0.048, 0.024, 0
```

qui est d'accord avec les calculs manuels. Il s'agit d'une étape très importante car c'est là que vous ajoutez de la valeur. MATLAB intégrera toutes les équations que vous lui donnez, bonnes ou mauvaises, et vous seul pouvez-vous assurer que le programme a résolu les bonnes équations.

Étape 3 Ensuite, écrivez un code qui sert de pilote. Ce code doit (a) définir toutes les constantes (ici, elles sont simplement mises dans la fonction rate1 pour plus de simplicité), (b) définir les conditions initiales et la longueur totale du réacteur, et (c) appeler le solveur d'ode.

```
% run_rate1.m
% This is the driver program to solve the simple flow
reactorexample.
y0 = [2 0 2]            % set the initial conditions
zspan = [0 2.4]         % set the total volume of the reactor
[z y] = ode45(@rate1,zspan,y0)
% call the ode solver
plot(z,y(:,1),'*-',z,y(:,2),'+-',z,y(:,3),'x-')
% plot the result
xlabel('length (m)')
ylabel('concentrations (kgmol/m^3)')
legend('A', 'B', 'C')
```

Étape 4 Chaque courbe est étiquetée avec un symbole différent dans la commande de tracé, et les courbes sont faciles à distinguer (voir figure 8.2). La commande de tracé donnée précédemment donnera aux trois courbes des couleurs différentes, mais celles-ci ne sont pas visibles dans cette version en noir et blanc. Maintenant que vous avez un code validé, vous pouvez faire varier les paramètres pour voir leur effet.

```
plot(z,y(:,1),'*-',z,y(:,2),'+-',z,y(:,3),'x-')
```

```
plot(z,y(:,1),'*-',z,y(:,2),'+-',z,y(:,3),'x-')
```

Maintenant, chaque courbe est étiquetée avec un symbole différent, et les courbes sont faciles à distinguer(voir figure 8.3).

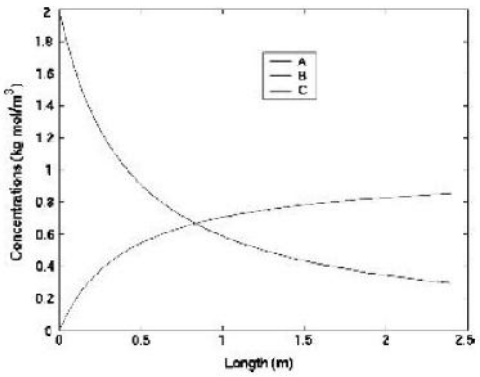

Figure 8.2. Solution to problem posed by Eqs. (8.21)–(8.22).