وزارة البحث العلمي والتعليم العالي
MINISTERE DE L'ENSEIGNEMENT SUPEREUR ET DE LA RECHERCHE SCIENTIFIQUE
جامعـة عبد الحميد بن باديس مستغانم
Université Abdelhamid Ibn Badis Mostaganem
كـلـيـة الـعـلـوم والـتـكـنـولـوجـيـا
Faculté des Sciences et de la Technologie
DEPARTEMENT DE GENIE DES PROCEDES

N° d'ordre : M2/GPE/2019

MEMOIRE DE FIN D'ETUDES DE MASTER ACADEMIQUE

Filière : Génie des procédés

Option : **Génie des Procédés de l'Environnement**

Thème

Modélisation et simulation de quelques phénomènes de transport

Présenté par :

M^r^. Alfred IYANGWA BOBONGO

Soutenu le 21/07/ 2019 devant le jury composé de :

Président :	H. BOUZID	MCA	Université de Mostaganem
Examinatrice :	N. HADDOU	MCB	Université de Mostaganem
Rapporteur :	M.R. GHEZZAR	Professeur	Université de Mostaganem

Année Universitaire 2018/2019

Remerciements

Je remercie Dieu maître de temps et des circonstances pour son amour, sa bonté et sa grâce continuelle sur moi.

Je tiens à exprimer ma profonde gratitude envers mon encadreur de mémoire, Monsieur M.R. GHEZZAR pour avoir dirigé ce travail et prodigué de nombreux et judicieux conseils.

Je tiens aussi à remercier Dr. H. BOUZID d'avoir accepté de présider notre jury de mémoire de projet de fin d'études. Sans oublier Madame N. HADDOU de l'honneur qu'elle me fait d'examiner ce travail.

Mes remerciements vont également à l'endroit de tous ceux qui m'ont soutenu de de loin ou de près d'une quelconque manière.

Je ne saurais terminer sans témoigner ma vive reconnaissance à tout le corps professoral ayant contribuer à ma formation tout au long de mon cursus universitaire, aussi à mes très chers collègues pour leur sympathie.

Dédicaces

Je dédie naturellement en premier ce modeste travail à mes très chers parents, pour leur amour inconditionnel, leur sacrifice et leur éducation.

C'est grâce à vos conseils et orientations durant toutes ces années que j'ai su maintenir le cap et garder de vue mon objectif.

A ma famille, tous mes frères et sœurs (Gaby, Youyou, Antoinette, Junior, Chicco, Jonathan, Laetitia, Nanou, Lucien, Nixon, Vinny) qui m'ont toujours encouragés et soutenus de mon plus jeune âge jusqu'à présent.

A mes cousins, cousines, nièces et neveux (Pasco, Grace, Archange, Clémence, Jonathan, Laena) pour leur soutien et leur affection à ma personne.

A la Communauté Congolaise d'Algérie et plus particulièrement celle de Mostaganem qui a été ma seconde famille, et a été un appui pour moi pendant le moment de joie comme de tristesse durant mon parcours universitaire.

Liste d'abréviations

- ***Pe : le nombre de Péclet***
- ***Da1 : le premier nombre de Damkohler***
- ***Da2 : le deuxième nombre de Damkohler***
- ***D : le coefficient de diffusivité***
- ***λ : la constante de désintégration***
- ***L : la profondeur***

Listes des tableaux

Listes des figures

Table des matières

Introduction générale

Il existe différents types de modèles mathématiques qui décrivent des processus liés à l'environnement. Par contre, il n'y a pas d'opinion unique sur la présence d'un modèle environnemental unique et/ou général. Chaque processus est par conséquent modélisé par des équations appropriées visant à analyser un processus ou de prédire son comportement dans le temps et l'espace.

Les phénomènes de transport, tels que les transferts de chaleur et de masse, jouent un rôle très important dans la vie humaine. Les gaz et les liquides nous entourent, les flux à l'intérieur de notre corps, et ont une influence profonde sur l'environnement dans lequel nous vivons. Les fluides s'écoulent et produisent des vents, des pluies, des inondations et des ouragans.

Les processus de transport sont responsables des variations de température et du transport des polluants dans l'air, l'eau et le sol.

Dans ce travail nous nous sommes particulièrement intéressés à modéliser des phénomènes en relation avec la dégradation de polluants dans un milieu liquide. Nous avons appliqué l'équation globale de transport avec les deux cas particuliers, à savoir : (i) le régime permanent, (ii) le régime transitoire. Chaque cas renferme des sous états : régime diffusionnel et régime convectionnel.

Ces équations ont été appliqués sur des exemples concrets tels que la désintégration de radio-éléments ou encore la dégradation d'un polluant après son rejet direct dans un cours d'eau.

Ce travail contient deux parties :

- La première partie parle du logiciel MATLAB qui nous a permis de programmer et de simuler les phénomènes de transport
- Et la seconde partie de la modélisation des phénomènes de transport.

Partie 1. Généralités sur le langage de programmation MATLAB

Introduction

Ce premier chapitre présente d'une manière générale les éléments et les notions que nous avons besoin pour effectuer notre travail. Il évoque les généralités, les programmes, les fonctions, les graphiques sur Matlab, etc.

Nous allons commencer par les généralités sur Matlab suivi des autres spécificités du logiciel.

1.2. Généralités sur Matlab

Avant toute chose, il est important de définir ce qu'est MATLAB. Pour certains, c'est un logiciel, un outil, pour d'autres un langage.

En fait, c'est un peu tout cela.

- lorsque l'on parle du logiciel MATLAB, on fait référence à l'outil que l'on utilise, l'interface utilisateur ;

- lorsque l'on parle du langage MATLAB, on désigne la syntaxe spécifique que l'on met en œuvre dans cet outil.

Le nom MATLAB vient de l'anglais MATrixLABoratory. Une traduction littérale nous amène à voir MATLAB comme un laboratoire pour manipuler des matrices. Nous reviendrons sur ce point, qui est un élément fondamental du langage MATLAB : la plupart des fonctions définies dans MATLAB le sont pour des grandeurs matricielles, et par extension, pour des données tabulées.

MATLAB comprend de nombreuses fonctions, de calcul ou de traitements de données, d'affichage, de tracés de courbes, de résolution de systèmes et d'algorithmes de calculs numériques au sens large du terme.

Toutes ces fonctions sont définies par défaut dans MATLAB dans un langage de programmation spécifique que l'on appelle MATLAB !

Ce langage comprend de nombreuses fonctions prédéfinies pour le calcul matriciel, mais pas seulement. De ce fait, les domaines d'application sont extrêmement variés, et l'on peut citer par exemple :

- le calcul numérique dans le corps des réels ou des complexes ;

- le calcul de probabilités ou les statistiques ;
- le calcul intégral ou la dérivation ;
- le traitement du signal ;
- l'optimisation ;
- le traitement d'image ;
- l'automatisme.

Et si en standard, MATLAB propose des fonctions couvrant l'ensemble de ces domaines, si vos développements nécessitent de mettre en œuvre des programmes très poussés, il existe des fonctions plus spécifiques regroupées dans des TOOLBOX (que l'on peut traduire par "boîte à outils"). Ces toolbox sont des extensions évidemment payantes utiles, voire nécessaires, comportant des fonctions dédiées à ces domaines, pour des développements de niveau professionnel. On peut citer les extensions :

- OPTIMIZATION pour l'optimisation ;
- IMAGE PROCESSING pour le traitement d'image.

Pour l'utilisateur débutant, ces toolbox ne sont pas nécessaires dans un premier temps, mais il est important de savoir que le moment venu, si le besoin se fait sentir, vous aurez la possibilité d'exploiter ces toolbox, et leur richesse fonctionnelle.

Par ailleurs, assez vite nous verrons qu'il est possible d'enrichir les possibilités de MATLAB en développant ses propres fonctions, avec le langage MATLAB. La syntaxe de ce langage est accessible aux débutants, parce que tout est fait pour s'affranchir des contraintes qu'imposent la plupart des langages de programmation :

- pas de chaînes de production : édition - définition des liens de bibliothèques - compilation – exécution ;
- pas de nécessité de déclarer et typer les fonctions ;

Nous verrons cependant que la notion de type existe dans ce langage comme dans les autres, et que si cette notion est cachée à l'utilisateur débutant, il peut être utile d'en tenir compte...

De même, s'il est possible d'éviter la phase de compilation, cette possibilité qui facilite l'utilisation de ce langage pour le débutant, cela peut s'avérer être un frein à l'utilisation de

MATLAB pour des développements avancés, qui demanderaient alors d'utiliser un autre langage plus performant. MATLAB, comme tout langage interprété, offre des performances globalement inférieures à d'autres langages compilés. Il faut pourtant reconnaître que les performances de l'interpréteur MATLAB s'améliorent de version en version, ce qui fait que dans bien des cas, les temps de calcul offerts aujourd'hui sont tout à fait honorables.

Notons enfin que MATLAB peut être exécuté sur les plates-formes les plus courantes : linux, mac os X ou Windows. L'absence de compilation rend les programmes écrits sous forme de script, par nature portable : si vous écrivez un programme en utilisant MATLAB sur une plate-forme donnée, vous pouvez diffuser votre code qui pourra être "interprété" par un utilisateur travaillant sur une autre plate-forme.

Matlab est un logiciel de calcul à syntaxe simple, il est optimisé pour le traitement des matrices, d'où est son nom. Matlab est considéré comme un langage de programmation au même titre que Pascal, C ou Basic. Il est adapté pour les problèmes scientifiques. Il permet aussi d'afficher des courbes et des données, de mettre en œuvre des algorithmes, de créer des interfaces utilisateurs, et peut s'interfacer avec d'autres langages comme le C, C++, Java, et Fortran. Matlab est un interpréteur : les instructions sont interprétées et exécutées ligne par ligne.

• Fenêtre Commande : Dans cette fenêtre, l'usage donne les instructions et Matlab retourne les résultats.

• Fenêtre Graphique : Matlab trace les graphiques dans ces fenêtres

• Fichiers M : Ce sont des programmes en langage Matlab.

• Toolboxes : Ce sont des collections de fichiers .M développés pour des domaines d'application spécifiques (signal Processing, Toolbox, System identification Toolbox, Control System Toolbox, u-Synthesis and Analyis Toolbox, Robust Control Toolbox, optimization Toolbox, Neural Network Toolbox, Chemometrics Toolbox, Fuzzy Logic Toolbox, etc.)

• Simulink : C'est l'extension graphique de Matlab permettant de travailler avec des diagrammes en blocs.

• Blocksets : Ce sont des collections de blocs Simulink développés pour des domaines d'application spécifique (DSP Blockset, Power System Blockset, etc.)

Il existe deux modes de fonctionnement sur Matlab :

Mode interactif : Matlab exécute les instructions au fur et à mesure qu'elles sont données par l'usager, c'est-à-dire travaillé sur la fenêtre d'édition de Matlab et on risque de perdre tout s'il y a problème.

Mode exécutif : Matlab exécute ligne par ligne un « fichier M » (programme en langage Matlab).

1.3. Programme Matlab

Afin d'éviter de devoir retaper une série de commandes, il est possible de créer un programme Matlab, connu sous le nom de « Fichier M » (« M-file »), le nom provenant de la terminaison « .m » de ces fichiers. Il s'agit, à l'aide de l'éditeur de Matlab (Menu « File→ New→ M-file ») ou d'un éditeur de texte, de créer un fichier en format texte qui contient une série de commandes Matlab.

Une fois le fichier sauvegardé (sous le nomdefichier.m par exemple), il s'agit de l'appeler dans Matlab à l'aide de commande.

>> Nom de fichier

Les commandes qui y sont stockées seront alors exécutées. Si vous devez apporter une modification à votre série de commandes, nous avons qu'à modifier la ligne du fichier-M en question et réexécuter le fichier-M en entrant le nom du fichier dans Matlab à nouveau (essayez la touche ↑). Cette procédure évite de retaper une série de commandes à répétition. C'est la procédure recommandée pour nos travaux pratiques. Il est possible de programmer des boucles et branchements dans les fichiers-M :

```
For i=1 :10                    % boucle for
  If i<5                       % branchement
  X(i)=i ;                     % n'oubliez pas votre point-virgule, sinon
   Else
     X(i)=0
     End                       % on n'oublie pas de terminer le branchement
     End                       % et la boucle
```

Le point-virgule à la fin d'une ligne signale à Matlab de ne pas afficher le résultat de l'opération à l'écran.

Une pratique courante est de mettre des « ; » à la fin de toutes les lignes et d'enlever certains de ceux-ci lorsque quelque chose ne tourne pas rond dans un programme, afin de voir ce qui se passe.

En général, il est préférable d'éviter de tel programme puisque nous utilisons un langage interactif, ce bout de code fait un appel au logiciel à chaque itération de la boucle et à chaque évaluation de la condition.

De plus, le vecteur change de taille à chaque itération Matlab doit donc faire une demande au système d'exploitation pour avoir plus de mémoire à chaque itération. Ce n'est pas important pour un programme de petite taille, mais si l'on a un programme qui fait une quantité importante de calculs importante de calculs, un programme optimisé peut être accéléré par un facteur 1000!

Les programmeurs expérimentés prennent l'habitude de débuter leurs programmes Matlab avec une série de leurs commandes préférées qui ont pour but d'éviter les mauvaises surprises par exemple :

```
>> clear all ;          % on efface toutes les variables, fonctions
>> Format compact ;     % suppression des sauts des lignes superflus
>> Format short e ;     % éviter qu'une petite quantité s'affiche 0.000
>> dbstop               % facilite l'impression d'une variable en cas d'erreur
```

1.4. Fonctions Matlab

En plus des fonctionnalités de base de MATLAB, une vaste bibliothèque de fonctions (les « toolbox » en langage MATLAB) est à votre disposition. Pour avoir une liste des familles de fonctions qui sont disponibles, entrez la commande help. Pour voir la liste des fonctions d'une famille de fonctions, on peut entrer help matfun par exemple, afin de voir la liste des fonctions matricielles. Pour obtenir de l'information sur une fonction en particulier, il s'agit d'utiliser la commande help avec le nom de la fonction, soit help cond pour avoir de l'information sur la fonction cond. Si la fonction n'a pas été compilée afin de gagner de la vitesse d'exécution, il est possible de voir le code source en entrant type cond, par exemple. Il est aussi possible de créer ses propres fonctions MATLAB. Le concept de fonction en MATLAB est similaire aux fonctions avec d'autres langages de programmation, i.e. une fonction prend un/des argument

(s) en entrée et produit un/des argument(s) en sortie. La procédure est simple. Il s'agit de créer un fichier-M, nommons-le mafonction.m. Ce qui différentie un fichier-M d'une fonction est que la première ligne de la fonction contient le mot clef function, suivi de la définition des arguments en entrée et en sortie. Par exemple, voici une fonction qui interverti l'ordre de deux nombres :

```
Function [y1, y2]= mafonction (x1,x2)
% définition de la fonction « ma fonction » :
% arguments en entrée : x1 et x2
% arguments en sortie : y1 et y2
% y1=y2 ; x1=x2 ;
```

Il s'agit en suite d'appeler votre fonction à l'invite Matlab :

```
>> [a, b]= ma fonction(1,2)
A=2
B=1
```

Si l'on veut définir une fonction pour calculer x^2, on écrira :

```
Fonction y= carre(x)
%
% définition de ma fonction x^2
% y=x*x ;
```

En entrant help carre, vous verrez les lignes de commentaires qui suivent immédiatement le mot clef fonction :

>> help carre

Définition de ma fonction x^2

Utilisez cette fonctionnalité pour vos fonctions. Les communications entre les fonctions et les programmes se font donc à l'aide des arguments en entrée et en sortie. La portée des variables définie à l'intérieur d'une fonction est donc limitée à cette fonction.

1.5. Graphiques

La fonction de base pour tracer un graphique avec MATLAB est la commande plot qui prend comme arguments une série de points donnés sous la forme de 2 vecteurs, qu'elle relie de segments de droites. C'est la responsabilité de l'utilisateur de générer assez de points pour qu'une courbe régulière paraisse régulière à l'écran. De plus, il est possible de donner des arguments additionnels, ou d'utiliser d'autres fonctions, pour contrôler l'apparence d'un graphique :

```
>> X= 0 :0.1 :2*pi                          %l'ordonnée
>> plot (x, sin(x), 'b-o,'cos(x) ; 'm--+') ;   % le graphe du sinus et du cosinus
>> axis ([ 0 :2*pi -1.1 1.1]) ;             % définition des axes.
>> title ('le titre du graphique') ;
>> xlabel ('l'axe des x') ;
>> ylabel ('l'axe des y') ;
Legend ('sinus','cosinus') ;
```

```
>> Clf reset                                % on réinitialise l'environnement graphique
>> hold on
>> plot (x, sin(x), 'b—o ') ;               % un premier graphique
>> plot (   x, cos(x), 'm--+') ;            % on superpose un deuxième graphique
>> hold on
```

La commande hold évite que le premier graphique soit écrasé par le deuxième. L'opérateur « : » n'est pas idéal pour créer le vecteur x dans ce contexte. Comme vous pouvez le voir sur la figure précédente, x(end) n'est pas égal à 2π. Dans ce cas, on préfère utiliser la commande linspace pour générer le nombre de points voulu dans un intervalle donné, alors que l'opérateur « : » nous donne plutôt le contrôle sur la distance entre les points.

```
>> x1= 0 :1 :2*pi                    % combien a-t-on des points entre 0 et 2*pi
X1=
      0     1     2     3     4     5     6
>> x2= linspace (0,2*pi, 7)          % ici l'on sait quel que l'on en a 7, allant de 0 à 2*pi
X2=
      0    1,0472   2,0944   3,1416   4,1888   5,2360   6,2832
```

1.6. Opérations matricielles avec les vecteurs et les matrices

Matlab est vraiment performant pour la manipulation des matrices, des vecteurs, ou des tableaux en général.

Pour la clarté de l'exposé, nous noterons par la suite les vecteurs avec des minuscules et les matrices par des majuscules. La multiplication des vecteurs et des matrices se fait en utilisant la notation '*'. Par exemple

>> C= [3 2 4 ;5 2 1 ;3 1* 2]

>> b= [3 1 2] ; >> v= [1 3 4] ; >> v*b' >> A*C

Notez bien évidemment que cette multiplication n'est pas commutative, comme il se doit. Pour les matrices carrées, il existe aussi des fonctions spécifiques extrêmement utiles :

1. Inv(A) calcule l'inverse de A
2. Det(A) calcule le déterminant de A
3. Diag(A) retourne la diagonale

4. [V,E]=eig(A) retourne deux matrices V et E. V est une matrice dont les colonnes contiennent les vecteurs propres de A et E est une matrice dont la diagonale contient les valeurs propres.**[6]**

1.7. Résolution des équations différentielles

Soient $I = [t_0, t_1]$ un intervalle de $\mathbb{R}$ et *a, b, f* des applications de *I* dans $\mathbb{R}$ continues.

Considérons l'équation différentielle ordinaire du second ordre :

$(\mathcal{E})$ $\quad y''(x) + a(x)y'(x) + b(x)y(x) = f(x) \forall x \in I$

Avec pour condition initiales :$y(t_0) = \propto_0, y'(t_0) = \propto_1$.

1.7.1. Mise sous forme d'un système différentiel du premier ordre

MATLAB ne sait résoudre que des systèmes différentiels du premier ordre. Une étape préliminaire à la résolution par MATLAB de l'équation différentielle est donc mise de celle-ci sous forme d'un système du premier ordre. Pour ce faire, désignons par y_1 la fonction inconnue *y* et par y_2 sa dérivée *y'*. On a bien entendu :

$y_1'(x) = y_2(x)$ pour tout $x \in I$. (1.1)

Par ailleurs, l'équation (Ɛ) s'écrit encore :

$$y_2'(x) = y''(x) = f(x) - a(x)y'(x) - b(x)y(x) = f(x) - a(x)y_2(x) - b(x)y_1(x). \quad (1.2)$$

Introduisons la fonction inconnues Y : $x \in I \rightarrow \begin{pmatrix} y_1(x) \\ y_2(x) \end{pmatrix} \in \mathbb{R}^2$. Compte-tenu des équations (1.1) et (1.2) on a

$$Y'(x) = \begin{pmatrix} y_1'(x) \\ y_2'(x) \end{pmatrix} = \begin{pmatrix} y_2(x) \\ f(x) - a(x)y_2(x) - b(x)y_1(x) \end{pmatrix} = F(x, Y(x)) \quad (1.3)$$

Où *F* est la fonction de $\mathbb{R}^3$ dans $\mathbb{R}^2$ définie par

$$F(x, y_1, y_2) = \begin{pmatrix} y_2 \\ f(x) - a(x)y_2 - b(x)y_1 \end{pmatrix}$$

Ainsi,

$$Y'(x) = \begin{pmatrix} F_1(x, y_{1,y_2}) \\ F_2(x, y_1, y_2) \end{pmatrix}$$

avec $F_1: (x, y_1, y_2) \rightarrow y_2$ et $F_2: (x, y_1, y_2) \rightarrow f(x) - a(x)y_2 - b(x)y_1$

De la même manière, une équation différentielle d'ordre 3 peut s'écrire sous la forme d'un système différentiel du premier ordre de dimension 3 et plus généralement, toute équation différentielle d'ordre *N* est équivalente à un système différentiel du premier ordre de dimension *N*.

1.7.2. Créer la fonction MATLAB décrivant le système différentiel

La seconde étape consiste à écrire une fonction MATLAB décrivant le système différentiel.

La fonction doit être de la forme

functiondY = edofct (x,Y)

où**edofct**est le nom (arbitraire) de la fonction MATLAB codant la fonction mathématique F.

le résultat **dY** est un vecteur colonne les composantes F_1 et F_2 de la fonction F.

Remarque : Même si F ne dépend pas explicitement de x, la présence des deux paramètres d'entrée de la fonction est obligatoire.

Exemple

```
functiondY = linedo (x,Y)
% definition de l'edo d'ordre n comme système du 1er ordre de la forme
% Y'=F(x,Y(x))
% F tant une application de R^{n+1} dans R^n
dY(1, :) =Y(2) ;
dY(2, :) =f(x)-a(x)*Y(2)-b(x)*Y(1) ;
```

Les fonctions a, b et f sont définies dans des fichiers séparés.

```
function val=a(x)
val = 1 ;
function val=b(x)
val = 1 ;
function val=f(x)
val = sin(x) ;
```

1.7.3. Résolution du système différentiel

La troisième étape consiste à choisir le solveur MATLAB devant résoudre le problème. Il existe deux types e solveur : les solveurs pours les problèmes classiques et ceux pour problème dits « raides ». Les solveurs pour problèmes classiques sont les suivants :

ode45 utilise une méthode de Runge-Kutta explicite à un pas. C'est le solveur à utiliser en premier choix pour la plupart des problèmes.

ode23 utilise également une méthode de Runge-Kutta explicite à un pas. Il peu être efficace que **ode45** dans certains cas.

ode113 utilise uneméthode de Adams-Bashfort-Moulton. C'est un solveur multi-pas. Les solveurs pour problèmes « raides » sont au nombre de quatre : **ode15s**, **ode23**s, **ode23t**, et **ode23tb.**

La syntaxe d'appel du solveur est :

[t,Y] = odexx(@edofct,[t0 t1], Y0)

où

- **odexx**désigne un des solveurs listés ci-dessus ;
- **edofct**désigne le nom de la fonction MATLAB décrivant la fonction F associée au système différentiel ;
- **[t0 t1]** est l'intervalle de temps sur lequel on cherche à calculer la solution ;
- **Y0** est le vecteur colonne contenant les données initiales : $Y_0 = \big(y(t_0), y'(t_0)\big)$.

Les paramètres de sortie du solveur sont :

- **t :** un vecteur colonne contenant les nœuds de l'intervalle [t0, t1] où a été calculée la solution ;
- **Y :** une matrice contenant les valeurs de la solution et des dérivées aux nœuds de l'intervalle [t0, t1] spécifiés dans le vecteur t. la i^{e}ligne de Y contient les valeurs de la dérivéé i-1^{e} de la solution aux nœuds de l'intervalle [t0, t1]. La première ligna contient la solution *y* de l'équation différentielle de départ (Ɛ).

Remarque : Noter la présence du caractère @ devant **edofct** dans les paramètres de la fonction. **[3]**

Partie 2. Modélisation et simulation de quelques phénomènes de transport liés à l'environnement

Introduction :

Dans cette partie du mémoire, nous détaillons des modèles mathématiques en relation avec l'équation globale de transport avec différents cas particuliers. Des exemples concrets ont été appliqués pour vérifier l'authenticité des résultats de calculs.

2.1. Décomposition et dégradation

La matière organique et les substances organiques sont sujettes à la dégradation. Les processus de **dégradation** sont de nature biochimique car ils sont véhiculés par des bactéries. Les détails de ces processus sont généralement assez complexes. Pour leur activité, les différentes cultures de bactéries dépendent fortement non seulement de la biogéochimie de l'environnement mais aussi sur les conditions de température et de pression. Une condition cruciale, par exemple, est la disponibilité en oxygène.

En milieu aérobie les bactéries dominent et consomment de l'oxygène en dehors de la matière organique. Ces composants sont transformés en divers produits qui incluent toujours le gaz carbonique. Dans un environnement anaérobie, lorsque l'oxygène disponible est consommé, d'autres bactéries prennent le rôle de contributeurs majeurs à la dégradation de la matière organique. D'autres accepteurs d'électrons deviennent plus importants, manganèse et fer dans la phase solide, ou azote et sulfate dans la phase fluide.

Le terme **décomposition** est généralement utilisé pour désigner des processus physiques ou chimiques qui provoquent une perte de substance. Le terme est bien connu à propos de désintégration radioactif pour la transformation des radionucléides en produits filles.

L'uranium U^{238} se désintègre avec une demi-vie de $4{,}5 \cdot 10^6$ ans, c'est-à-dire après la moitié de la masse initiale est toujours présente pendant que l'autre moitié est transformée en Th^{234}, si aucun autre processus n'est impliqué. Comme le produit fille Th^{234} a une demi-vie de 2,1 jours seulement, la plupart se décomposent en Pa^{234}, ce qui a encore une plus courte durée avec une demi-vie de seulement 12 minutes. Le prochain produit fille U^{234} a une longue demi-vie de $2{,}5 \cdot 10^5$ ans.

Il existe une formulation mathématique de base couramment utilisée pour décrire la décomposition et dégradation. Une approche générale reconnaît les pertes q proportionnelle à une puissance de la concentration c :

$$q = -\lambda c^n \qquad (2.1)$$

Où l'entier n est l'ordre de dégradation. λ est la constante de dégradation qui dépend généralement de toutes les variables de l'environnement du système mis en équation. La décomposition du deuxième ordre est proportionnelle au carré de la concentration. L'unité physique de λ dépend de l'exposant n. Pour le cas important avec n = 1, l'unité de λ est [1/T].

Le terme **q**, donné par (2.1), est substitué dans l'équation de transport de masse général. Il faut noter que les deux termes, q et l'équation différentielle ont les mêmes unités physiques (M/T/ L^3).

Si on veut être rigoureux, nous devons appliquer l'équation générale de transport de masse :

$$\frac{\partial c}{\partial t} = \nabla . D\nabla c - \nabla . vc - \lambda c^n \qquad (2.2)$$

Avec cette approche, les processus de décomposition et de dégradation sont inclus dans l'équation de transport, et il devient possible de traiter le transport, la décomposition et dégradation simultanément.

Avant de donner des solutions et des stratégies de solution pour la situation générale, les cas spéciaux seront traités en premier.

Si aucun autre processus n'est pertinent l'équation dévient :

$$\frac{\partial c}{\partial t} = -\lambda c^n \qquad (2.3)$$

C'est une équation différentielle ordinaire pour la variable indépendante t.

Le plus important est la décroissance ou la dégradation du 1er ordre, c'est-à-dire le cas n = 1, où les pertes sont proportionnelles à la concentration. Puis la solution du différentiel l'équation (3.3) peut être notée directement :

$$c = c_0 \exp(-\lambda t) \qquad (2.4)$$

Qui vaut pour la condition initiale $c\ (t = 0) = c_0$.

La fonction exponentielle est évidemment la solution pour un composant avec décroissance de premier ordre, qui explique la notation décroissance exponentielle. La demi-vie $t_{1/2}$ est la période au cours de laquelle la concentration en composants diminue jusqu'à la moitié de la valeur initiale. Donc selon (3.4) la demi-vie $t_{1/2}$ est caractérisé par la condition :

$$1/2 = \exp(-\lambda t_{1/2}) \qquad (2.5)$$

Ce qui équivaut à la condition $t_{1/2}=ln(2)/\lambda$ C'est la réciproque relation entre constante de désintégration et demi-vie. Avec$t_{1/2}$ décroissance exponentielle peut être noté sous forme sans dimension comme :

$$\boxed{c/c_0 = \exp(-\ln(2)\, t/t_{1/2})} \qquad (2.6)$$

Pour les variables sans dimension c/c_0 et $t/t_{1/2}$. Pour la période de cinq ans la fonction demi-vie est décrite par le programme MATLAB suivant :

```
Plot ([0 :0.1 :5], exp (-log (2) *[0 :0.1 :5])) ; grid ;

xlabel ('time t / t_{1/2}') ; ylabel ('c / c_0') ;
```

Nous obtenons le graphique de la figure suivante :

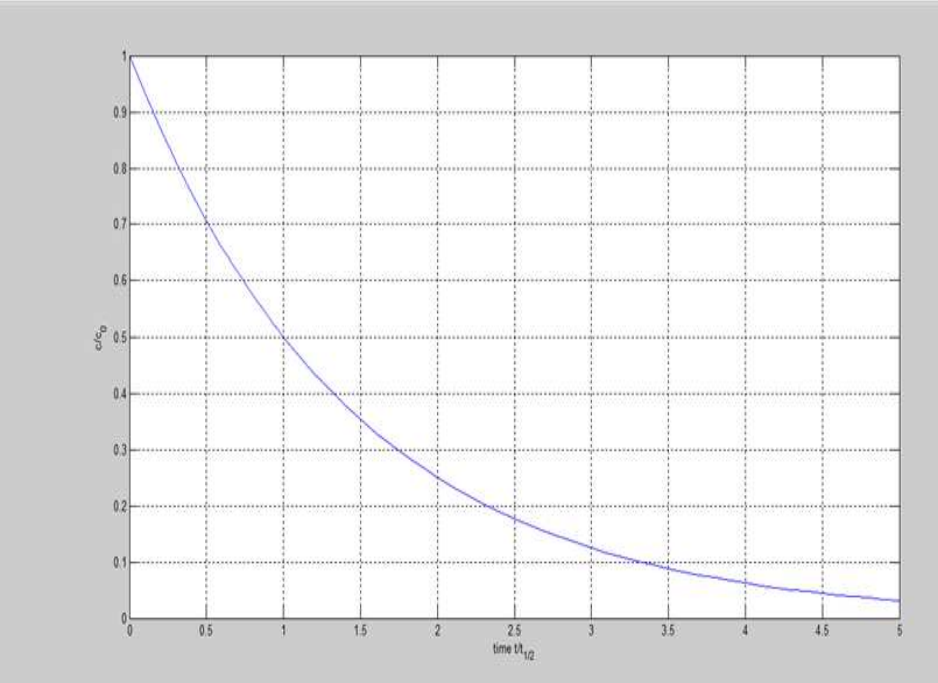


Figure 2.1. *Décroissance exponentielle par les variables adimensionnelles*

Du point de vue mathématique, la principale différence entre la décroissance des substances radioactives et les autres formes de pertes de premier ordre résident dans la constante de décroissance. La demi-vie de radionucléides est constante dans toutes les conditions connues. Le taux de dégradation exponentiel d'un nucléide n'est influencé par aucune condition environnementale, ni par la température, ni par la pression, ni par l'environnement biogéochimique. En revanche, les taux chimiques et biochimiques sont fortement influencés par les facteurs environnementaux.

Nous avons par la suite calculé les demi-vies de quelques radionucléides, dont nous en résumons l'essentiel dans le tableau suivant :

Tableau 2.1. Demi-vies de radionucléides sélectionnés

Radionucléides	Demie vie	Radionucléides	Demie vie
U-238	$4.5 .10^9$ ans	H-3	12.35 ans
U-235	32500 ans	R-228	5.8 ans
Ra-226	1600 ans	Th-22	1.91 ans
Am-241	432.2 ans	Gd-153	242 ans
Pu-238	87.74 ans	Po-210	138 ans
Cs-137	30 ans	Sr-89	50.5 ans
Pb-210	22.3 ans	Th-234	24.1 ans

En fait, la loi de décomposition peut souvent être comprise comme une règle la plus simplifiée, dans laquelle l'interaction de plusieurs processus complexes est rassemblés et où λ est un paramètre forfaitaire. Clairement, dans un environnement modifié le paramètre est différent. Les biodégradations plus complexes sont traitées en utilisant la cinétique de Michaelis-Menten ou de Monod. **[4]**

2.2. Etat stationnaire (régime permanent)

Comme mentionné ci-dessus, l'équation différentielle pour l'état stationnaire est obtenue en mettant à zéro les dérivées temporelles de l'équation de transport (2.2). Le côté droit de l'équation de transport a donc été mis égal à zéro. L'équation (2.2) deviendra :

$$\frac{\partial}{\partial x} D \frac{\partial c}{\partial x} - v \frac{\partial c}{\partial x} - \lambda c = 0 \qquad (2.7)$$

C'est une équation différentielle ordinaire pour la variable indépendante x.

Avec MATLAB, les équations différentielles ordinaires peuvent être résolues numériquement. Ici, une solution analytique, qui fournit la solution dans une formule explicite, est une alternative si les coefficients sont des constantes, c'est-à-dire indépendantes de *x* et *t*. Afin de résoudre l'équation différentielle (2.7) analytiquement, il convient de la noter sous une forme différente :

$$\left(\frac{\partial}{\partial x} - \mu_1\right)\left(\frac{\partial}{\partial x} - \mu_2\right) c = 0 \qquad (2.8)$$

Les paramètres μ_1 et μ_2 peuvent être obtenus par comparaison de coefficients dans (2.7) et (2.8) :

$$\mu_1 + \mu_2 = v/D \qquad \mu_1.\mu_2 = -\lambda/D \qquad (2.9)$$

Il en résulte une équation quadratique qui a les solutions suivantes :

$$\mu_{1,2} = \frac{1}{2D}\left(v \pm \sqrt{v^2 + 4\lambda D}\right) \qquad (2.10)$$

L'équation (2.8) peut maintenant être résolue en deux étapes. D'abord la solution $\bar{c}$ de l'équation :

$$\left(\frac{\partial}{\partial x} - \mu_1\right)\bar{c} = 0 \qquad (2.11)$$

est déterminée, ce qui est donné par $\bar{c} = C_0 \exp(\mu_1 x)$.

C_0 est une constante d'intégration qui est déterminé ci-dessous afin de remplir les conditions aux limites. Dans une seconde étape, *c* se trouve comme solution de l'équation différentielle

$$\left(\frac{\partial}{\partial x} - \mu_2\right) c = \bar{c} \qquad (2.12)$$

On obtient une formule pour la solution générale :

$$\boxed{c(x) = \exp(\mu_2 x)\,(C_1 + C_0 \int \exp(\mu_1 x)\exp(-\mu_2 x)\,dx) = C_1 \exp(\mu_2 x) + \frac{C_0}{\mu_1 - \mu_2}\exp(\mu_1 x)} \qquad (2.13)$$

Où C_1 est la deuxième constante d'intégration. C_0et C_1 sont déterminés par les conditions aux limites. La condition habituelle à l'entrée $c(x=0) = c_{in}$ donne la condition : $C_1 + C_0/(\mu_1-\mu_2) = c_{in}$ *ou* $C_1 = c_{in} - C_0/(\mu_1-\mu_2)$.

Notez que μ_1 etμ_2, donnés par (2.10), ont des signes opposés. Comme μ_1 est positif, le premier terme en (2.13) diminue avec la profondeur, tandis que le second augmente avec la profondeur. Pour cette raison, les solutions s'approchent de l'infini pour tous valeur $C_0 \neq 0$. Inversement, la

fonction avec $C_0 = 0$ la seule solution qui garantit une concentration finie pour des valeurs arbitraires élevées de x. C'est cette propriété qui rend le choix $C_0 = 0$ favorable dans les études, où il y a aucune information concernant la condition aux limites en aval (Anderson et al. 1988 ; Henderson et al. 1999). La solution se lit alors simplement :

$$c(x) = c_{in}\exp(\mu_2 x) \qquad (2.14)$$

Lorsque la deuxième condition aux limites nécessite un gradient de concentration nul à la profondeur L, c'est-à-dire $(\partial c/\partial x)(L) = 0$, la deuxième équation pour C_0 et C_1 est donné par :

$$C_1\mu_2 \exp(\mu_2 L) + \frac{C_0\mu_1}{\mu_1-\mu_2}\exp(\mu_1 L) = 0 \qquad (2.15)$$

Ce qui conduit à la solution :

$$\boxed{C_0 = \frac{c_{in}\mu_2(\mu_2-\mu_1)\exp(\mu_2 L)}{\mu_1 \exp(\mu_1 L)-\mu_2\exp(\mu_2 L)}} \quad \text{et} \quad \boxed{C_1 = \frac{c_{in}\exp(\mu_1 L)}{\mu_1 \exp(\mu_1 L)-\mu_2\exp(\mu_2 L)}} \qquad (2.16)$$

Avec ces formules, la solution est peut-être calculée avec MATLAB. Dans tous les cas les constantes libres C_0 et C_1 dans une formule pour la solution générale, comme dans (2.13), doivent être déterminés pour remplir les conditions. Pour les conditions aux limites de Dirichlet sur les deux côtés $C(0) = c_{in}$ et $C(L) = c_0$ on obtient :

$$\boxed{C_0 = \frac{c_{in}(\mu_2-\mu_1)\exp(\mu_2 L)}{\exp(\mu_1 L)-\exp(\mu_2 L)}} \quad \text{et} \quad \boxed{C_1 = c_{in} - \frac{C_0}{\mu_1-\mu_2}} \qquad (2.17)$$

Une dégradation d'ordre plus élevée est généralement beaucoup plus difficile à gérer que la dégradation de premier ordre. Afin de traiter des formulations plus complexes, MATLAB offre la possibilité d'utiliser des méthodes numériques pour les équations différentielles ordinaires. **[5]**

2.3. Formulation adimensionnelle

Dans la forme adimensionnelle, la solution (2.13) peut être écrite ainsi :

$$\frac{c(\overline{x})}{c_{in}} = \bar{C}_0 \exp(\bar{\mu}_1\bar{x}) + (1-\bar{C}_0)\exp(\bar{\mu}_2\bar{x}) \qquad (2.18)$$

Avec sans dimension de profondeur $\bar{x} = x/L$, $\bar{\mu}_1 = \mu_1 L$, $\bar{\mu}_2 = \mu_2 L$ et une constante d'intégration $\bar{C}_0$.

Les paramètres $\bar{\mu}_1$ et $\bar{\mu}_2$ peuvent être exprimés en fonction du nombre sans dimension de Péclet $Pe = vL/D$ et le 2ème nombre adimensionnel de Damkohler $Da_2 = \lambda L^2/D$:

$$\bar{\mu}_{1,2} = \frac{1}{2} Pe \pm \sqrt{\frac{1}{4} Pe^2 + Da_2} \qquad (2.19)$$

Si le premier nombre de Damkohler pour la phase fluide est défini par $Da_1 = \lambda L/v$, les deux valeurs de μ peuvent également être exprimées en fonction de *Pe* et Da_1.

En utilisant l'identité $Da_2 = Pe/Da_1$, on obtient $\bar{\mu}_1$ et $\bar{\mu}_2$ en fonction de *Pe* et Da_1:

$$\bar{\mu}_{1,2} = \frac{1}{2} Pe \pm \sqrt{\frac{1}{4} Pe^2 + \frac{Pe}{Da_1}} \qquad (2.20)$$

La solution avec disparition du gradient de concentration à la profondeur L peut être exprimé par la formule :

$$\frac{c(\bar{x})}{c_{in}} = \frac{\bar{\mu}_2 \exp(\bar{\mu}_2) \exp(\bar{\mu}_1 \bar{x}) - \bar{\mu}_1 \exp(\bar{\mu}_1) \exp(\bar{\mu}_2 \bar{x})}{\bar{\mu}_2 \exp(\bar{\mu}_2) - \bar{\mu}_1 \exp(\bar{\mu}_1)} \qquad (2.21)$$

Un cas particulier de (2.21) est

$$\frac{c(\bar{x})}{c_{in}} = \frac{\exp(\sqrt{Da_2}) \exp(-\sqrt{Da_2}\bar{x}) + \exp(-\sqrt{Da_2}) \exp(\sqrt{Da_2}\bar{x})}{\exp(\sqrt{Da_2}) + \exp(-\sqrt{Da_2})} \qquad (2.22)$$

Qui est obtenu pour $Pe = 0$, c'est-à-dire sans advection. Les graphiques de fonctions (2.22) pour différentes valeurs du deuxième nombre de Damkohler sont présentés dans Fig.2.2

Le code MATLAB suivant est utilisé pour le tracé. Les marqueurs en lignes ont été ajoutés par post-traitement avec l'éditeur de figures MATLAB. Le résultat de calcul est présenté par la figure 2.2.

```
x = [0: 0.01: 1];

Da2 = 8 ; mu1 = sqrt (Da2); mu2 = -mu1;

s = mu2 * exp (mu2) -mu1 * exp (mu1);

c = (mu2 * exp (mu2) * exp (mu1 * x) -mu1 * exp (mu1) * exp (mu2 * x)) ./ s
```

```
for Da2 = [4 2 1 0.5 0.25 0.125] ;

s = sqrt (Da2); mu1 = s; mu2 = -s;
```

```
s = mu2 * exp (mu2) -mu1 * exp (mu1);

c = [c; (mu2 * exp (mu2) * exp (mu1 * x) -mu1 * exp (mu1) * exp (mu2 * x)) ./ s];

end

plot (x, c);

legend ('Da 2 = 8', 'Da 2 = 4', 'Da 2 = 2', 'Da 2 = 1', 'Da 2 = 0,5','Da 2 = 0,25', 'Da 2 = 0,125');

xlabel ('x / L [-]') ; ylabel ('c / c {in} [-]');
```

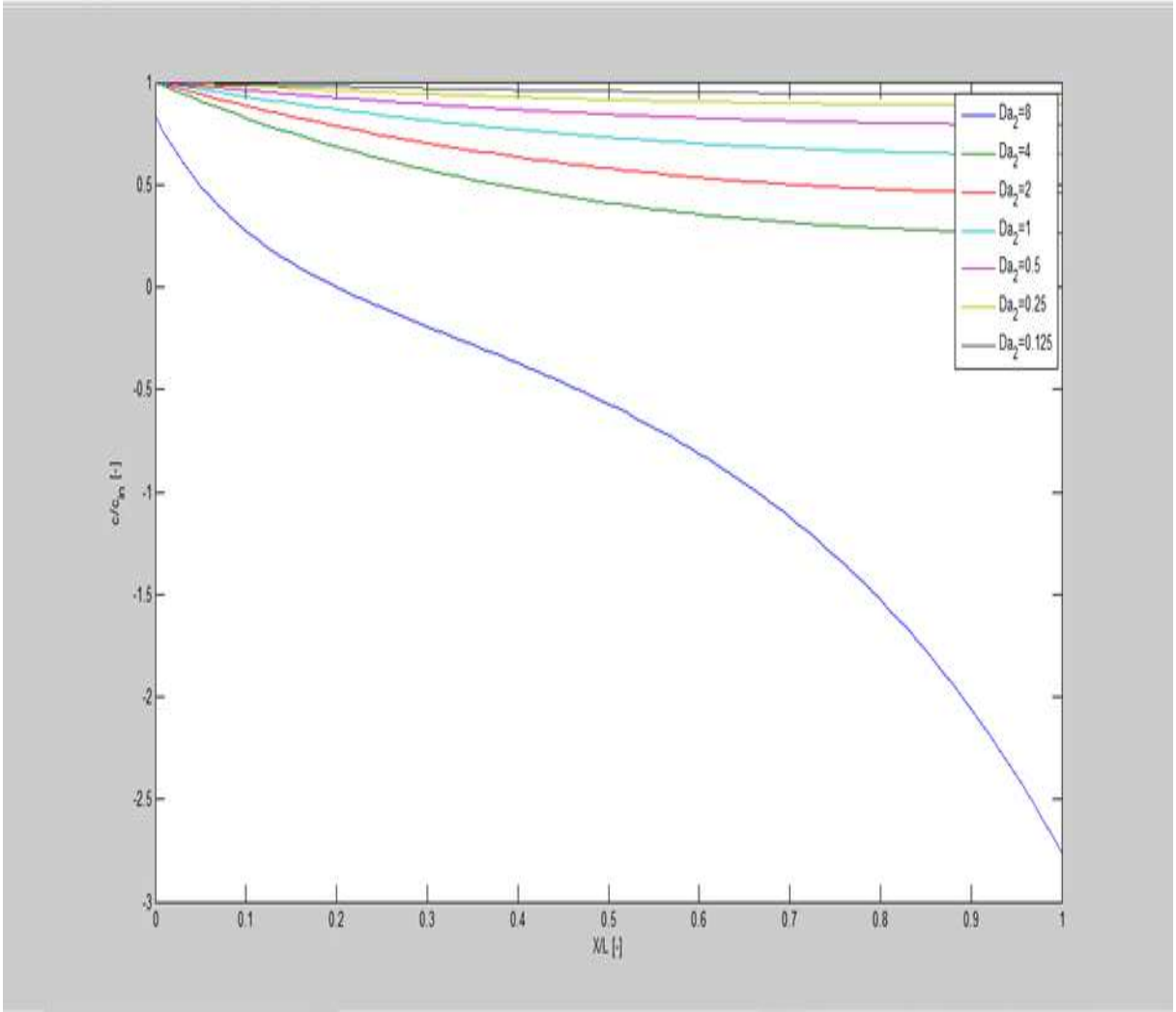


Figure 2.2. Profils de concentration en cas de diffusion et de décroissance à constante paramètres D et λ ; en fonction du 2ème nombre sans dimension de Damkohler Da2.

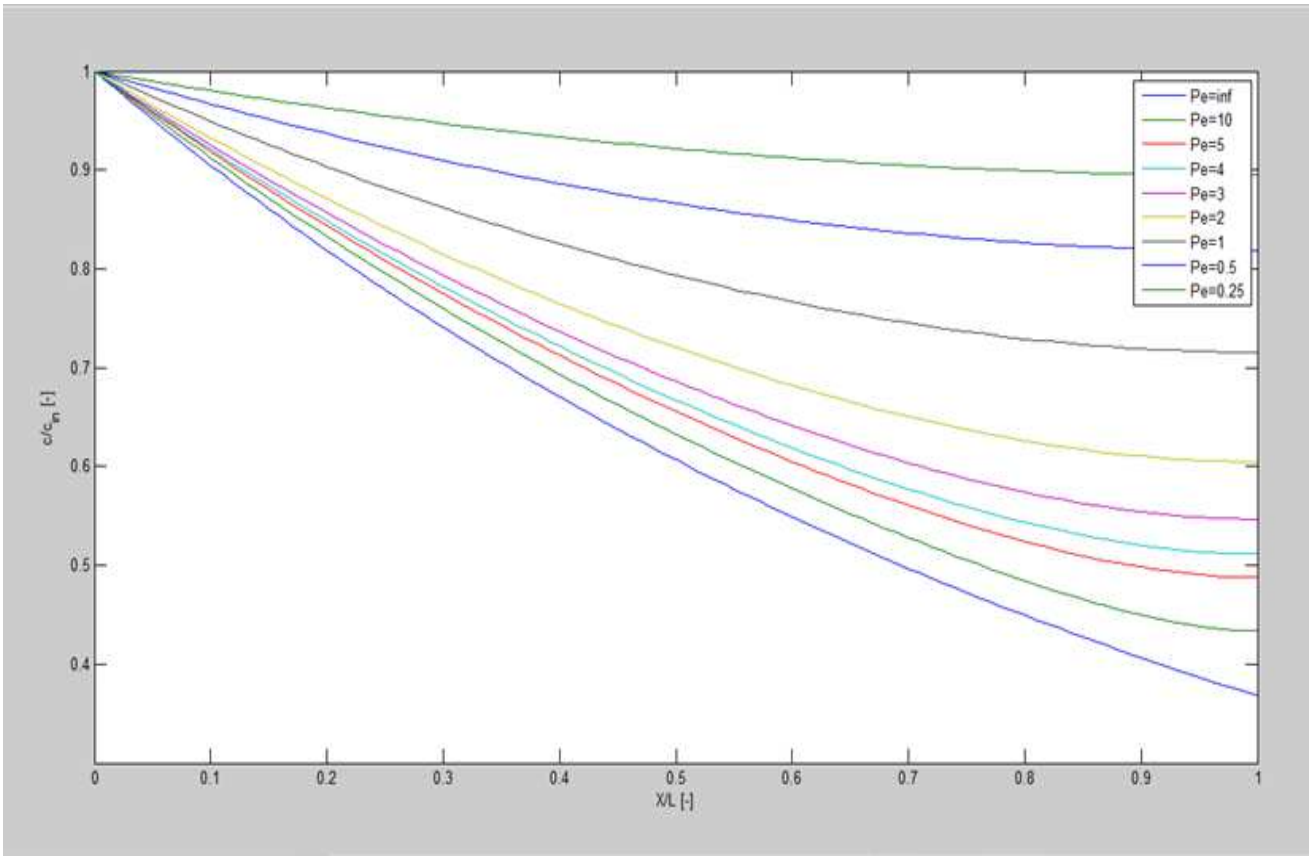


Figure 2.3. Profil pour la concentration de la phase fluide en cas de transport à constante paramètres pour Da1=1 en fonction du nombre Péclet Pe.

La courbe de la Fig.2.3 montre le profil de l'advection et de la dégradation seulement, c'est-à-dire sans diffusion, calculé comme une solution du premier ordre simple équation différentielle ordinaire :

$$-v\frac{\partial c}{\partial x} - \lambda c = 0 \qquad \text{Ou} \qquad \frac{\partial c}{\partial x} = -\frac{\lambda}{v}c \qquad (2.23)$$

La solution est :

$$c(x) = c_{in}\exp(-\frac{v}{\lambda}x) \qquad (2.24)$$

Ou sous la forme adimensionnelle (pour la concentration adimensionnelle, avec paramètre et variable indépendante adimensionnelle)

$$c/c_{in} = \exp(-Da_1\bar{x}) \qquad (2.25)$$

Formellement, on peut représenter le cas de non-diffusion par $Pe = \infty$. Comme on pourrait pour des valeurs croissantes de Pe, la fonction de (2.25) est approchée comme asymptote. Mais la convergence est assez lente. Pour la valeur deDa_1= 1, la valeur du nombre de Péclet doit être nettement supérieure à 10 afin d'obtenir une bonne correspondance entre la distribution de concentration et l'asymptote. Pour une bonne correspondance près du point de sortie (x près de L), *Pe* devrait être 100 ou plus. Pour cela, on exécute le programme suivant :

```
x = [0: 0,01: 1];

Da1 = 1;

c = exp (-Da1 * x);

for Pe = [10 5 4 3 2 1 0,5 0,25];

s = sqrt (0,25 * Pe * Pe + Pe / Da1);

mu1 = 0,5 * Pe + s; mu2 = 0,5 * Pe-s;

s = mu2 * exp (mu2) -mu1 * exp (mu1);

c = [c; (mu2 * exp (mu2) * exp (mu1 * x) -mu1 * exp (mu1) * exp (mu2 * x)) ./ s];

end

plot (x, c);

legend ('Pe = Inf', 'Pe = 10', 'Pe = 5', 'Pe = 4', 'Pe = 3', 'Pe = 2', 'Pe = 1'

'Pe = 0,5', 'Pe = 0,25');

xlabel ('x / L [-]'); ylabel ('c / c {in} [-]');
```

La figure 2.4 montre des graphiques de fonctions selon (2.21) pour une valeur fixe du nombre Péclet *Pe* = 1 et des valeurs sélectionnées du premier nombre de DamkohlerDa_1.

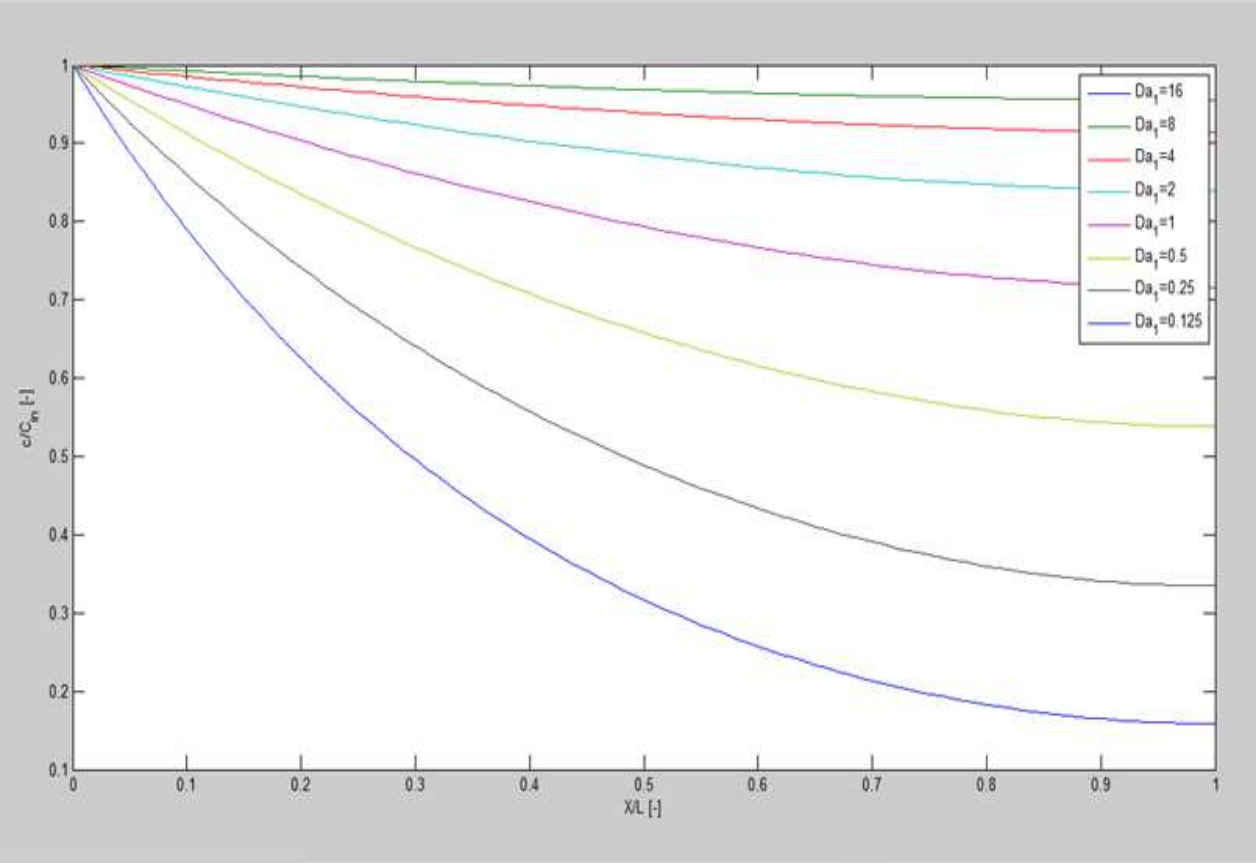


Figure 2.4. Profils pour la concentration de la phase fluide en cas de transport à paramètres constantes pour le nombre de Pe en fonction du premier nombre de Damkohler.

La valeur constante de *Pe* garantit une relation constante entre la diffusion et advection, lorsque que les valeurs de Da_1 décomposition ou de dégradation changent les processus changent leurs importances relatives. Pour les valeurs élevées du Damkohler la décroissance numérique est relativement importante et donc les concentrations sont significativement réduit de l'entrée ($x = 0$) à la sortie ($x = L$). Avec décroissance pour Da_1, la décroissance devient moins pertinente. Puis le profil pour l'advection et la diffusion est approchée, cet approchement est la ligne droite représentant la concentration constante $c = c_0$ (pour les conditions aux limites données).

La figure est produite dans MATLAB à l'aide du code suivant :

```
x = [0: 0,01: 1];

Pe = 1 ; Da1 = 16;

s = sqrt (0,25 * Pe * Pe + Pe / Da1);

mu1 = 0,5 * Pe + s; mu2 = 0,5 * Pe-s;
```

```
s = mu2 * exp (mu2) -mu1 * exp (mu1);
c = (mu2 * exp (mu2) * exp (mu1 * x) -mu1 * exp (mu1) * exp (mu2 * x)) ./ s;
for  Da1 = [8 4 2 1 0,5 0,25 0,125];
s = sqrt (0,25 * Pe * Pe + Pe / Da1);
mu1 = 0,5 * Pe + s; mu2 = 0,5 * Pe-s;
s = mu2 * exp (mu2) -mu1 * exp (mu1);
c = [c; (mu2 * exp (mu2) * exp (mu1 * x) -mu1 * exp (mu1) * exp (mu2 * x)) ./ s];
end
plot (x, c);
legend ('Da_1 = 16', 'Da_1 = 8', 'Da_1 = 4', 'Da_1 = 2', 'Da_1 = 1', 'Da_1 = 0,5',
'Da_1 = 0,25', 'Da_1 = 0,125');
xlabel ('x / L [-]'); ylabel ('c / c {in} [-]');
```

La figure 2.5 représente les solutions pour augmenter l'advection si la relation entre diffusion et décroissance reste égale. Pour ce graphe, on suppose que les deux processus sont d'importance égale, ce qui est exprimé par la 2^{e} nombre Damkohler Da_2=1. Avec l'augmentation du nombre de Péclet, l'advection devient plus pertinente, le front peut pénétrer plus loin, et les gradients de concentration deviennent moins raides. On appliquera le code suivant :

```
 x = [0: 0,01: 1];
Da2 = 1;
mu1 = sqrt (Da2); mu2 = -mu1;
s = mu2 * exp (mu2) -mu1 * exp (mu1);
c = (mu2 * exp (mu2) * exp (mu1 * x) -mu1 * exp (mu1) * exp (mu2 * x)) ./ s;
for Pe = [0,0625 0,125 0,25 0,5 1 2 4 8 16];
```

```
s = sqrt (0,25 * Pe * Pe + Da2);

mu1 = 0,5 * Pe + s; mu2 = 0,5 * Pe-s;

s = mu2 * exp (mu2) -mu1 * exp (mu1);

c = [c; (mu2 * exp (mu2) * exp (mu1 * x) -mu1 * exp (mu1) * exp (mu2 * x)) ./ s];

end

plot (x, c);

legend ('Pe = 0', 'Pe = 0,0625', 'Pe = 0,125', 'Pe = 0,25', 'Pe = 0,5',

'Pe = 1', 'Pe = 2', 'Pe = 4', 'Pe = 8', 'Pe = 16');

xlabel ('x / L [-]'); ylabel ('c / c {in} [-]');
```

L'exécution donne la courbe de la figure suivante :

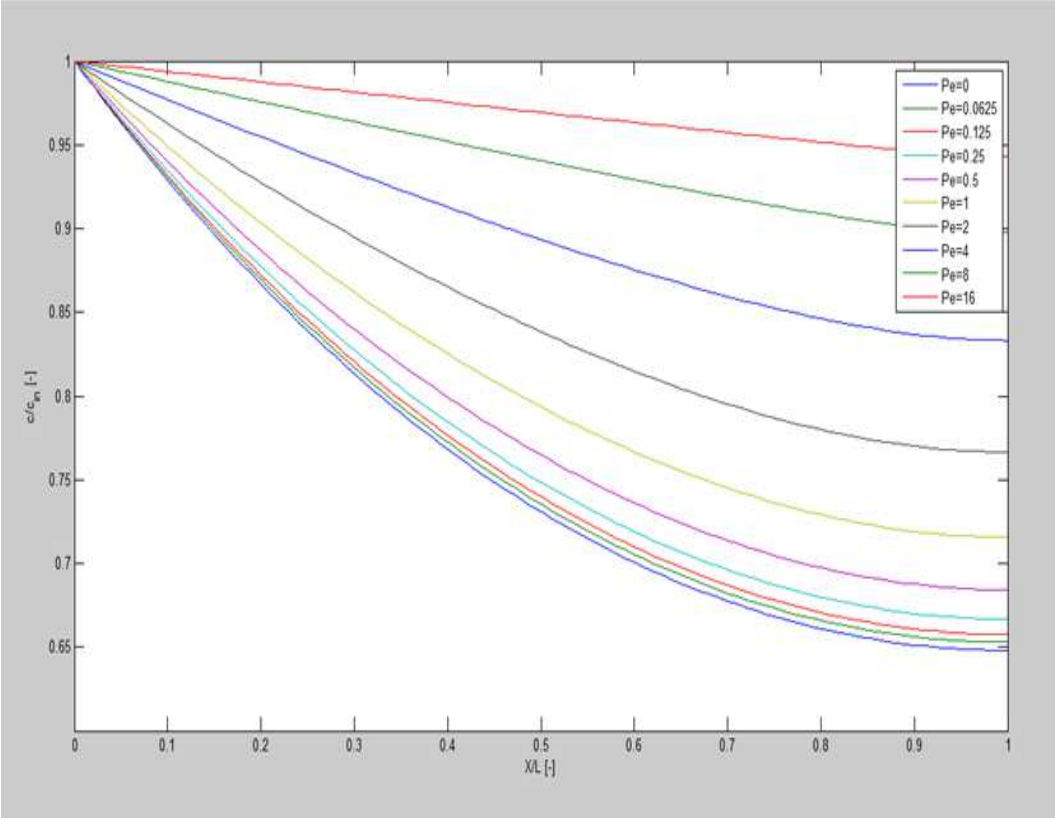


Figure 2.5. Profils pour la concentration de la phase fluide en cas de transport à paramètres constantes pour le 2ème nombre de Damkohler Da2= 1 en fonction du Pe.

La limite de la solution pour $Pe = 0$ a été obtenue à partir de l'équation différentielle

En absence de l'advection :

$$\frac{\partial}{\partial x} D \frac{\partial c}{\partial x} - \lambda c = 0 \qquad \text{Ou} \qquad \frac{\partial^2 c}{\partial x^2} = \frac{\lambda}{D} c \quad (2.26)$$

Avec la solution générale :

$$c(x) = C_0 \exp\left(\sqrt{\frac{v}{D}} x\right) + C_1 \exp\left(-\sqrt{\frac{v}{D}} x\right) \quad (2.27)$$

Dans les paramètres et variables adimensionnels l'équation dévient :

$$c(x) = C_0 \exp(Da_2 \bar{x}) + C_1 \exp(-Da_2 \bar{x}) \quad (2.28)$$

Les constantes libres C_0et C_1sont encore obtenues pour les conditions aux limites. **[4]**

2.4. Régime transitoire

La solution analytique pour l'entrée d'un front avec concentration C_{in} dans la région avec concentration C_0 est donnée par :

$$c(x,t) = c_0 \exp(-\lambda t)\left(1 - \frac{1}{2} erfc\left(\frac{x - vt}{2\sqrt{Dt}}\right) - \frac{1}{2}\exp\left(\frac{vx}{D}\right) erfc\left(\frac{x + vt}{2\sqrt{Dt}}\right)\right) \ldots$$
$$+ \frac{c_{in}}{2}\left(\exp\left(\frac{v - u}{2D} x\right) erfc\left(\frac{x - ut}{2\sqrt{Dt}}\right) + \exp\left(\frac{v + u}{2D} x\right) erfc\left(\frac{x + ut}{2\sqrt{Dt}}\right)\right)$$

(2.29)

Avec $u = \sqrt{v^2 + 4\lambda D}$ (Wexler 1992). La solution consiste en deux parties : la première décrit d'abord le déclin de la concentration initiale C_0 et le second le changement de la concentration d'entrée c_{in} dans la configuration stationnaire. **[7]**

Dans MATLAB, la solution doit être incluse dans le fichier M "analtrans.m". Le paramètre de désintégration λ est ajouté dans la partie entrée du module :

```
lambda = 0,1;  % decayrate
```

Ensuite, le paramètre auxiliaire u est calculé par :

```
u = sqrt (v * v + 4 * lambda * D);
```

Et la formule (29) est programmée par le long terme :

```
c = [c;  c0*exp (-lambda * t (i)) * (e-0,5 * erfc (h * (x-e * v * t (i)))
-0,5 * exp ((v / D) * x). * Erfc (h * (x + e * v * t (i))))
+ ... (cin-c0) * 0.5 * (exp ((v-u) / (D + D) * x). * erfc (h * (x-e * u * t (i)))
+ exp ((v + u) / (D + D) * x). * erfc (h * (x + e * u * t (i))))] ;
```

En outre, le modèle ‘simpletrans.m’ peut être étendu facilement en introduisant décomposition en tant que processus supplémentaire. Écrivez un autre sous-module, nommé ‘kinetics.m’ qui comprend une seule commande :

```
c1 = exp (-lambda * dtdiff) * c1; %  simple first orderkinetic
```

Dans le module principal, ‘simpletrans.m ’, on l’appelle la commande cinétique avant diffusion.

Dans la formulation adimensionnelle, la solution est la suivante :

$$c(\bar{x},\bar{t}) = c_0 \exp(-Da_2\bar{t})\left(1-\frac{1}{2}erfc\left(\frac{\bar{x}-\bar{t}}{2\sqrt{\bar{t}/Da_2}}\right)-\frac{1}{2}\exp(Pe.\bar{x})\,erfc\left(\frac{\bar{x}+\bar{t}}{2\sqrt{\bar{t}/Da_2}}\right)\right)$$

$$+\frac{c_{in}}{2}\left(\exp\left(Pe\frac{1-u}{2}\bar{x}\right)erfc\left(\frac{\bar{x}-u\bar{t}}{2\sqrt{\bar{t}/Da_2}}\right)+\exp\left(Pe\frac{1+u}{2}\bar{x}\right)erfc\left(\frac{\bar{x}+u\bar{t}}{2\sqrt{\bar{t}/Da_2}}\right)\right)$$

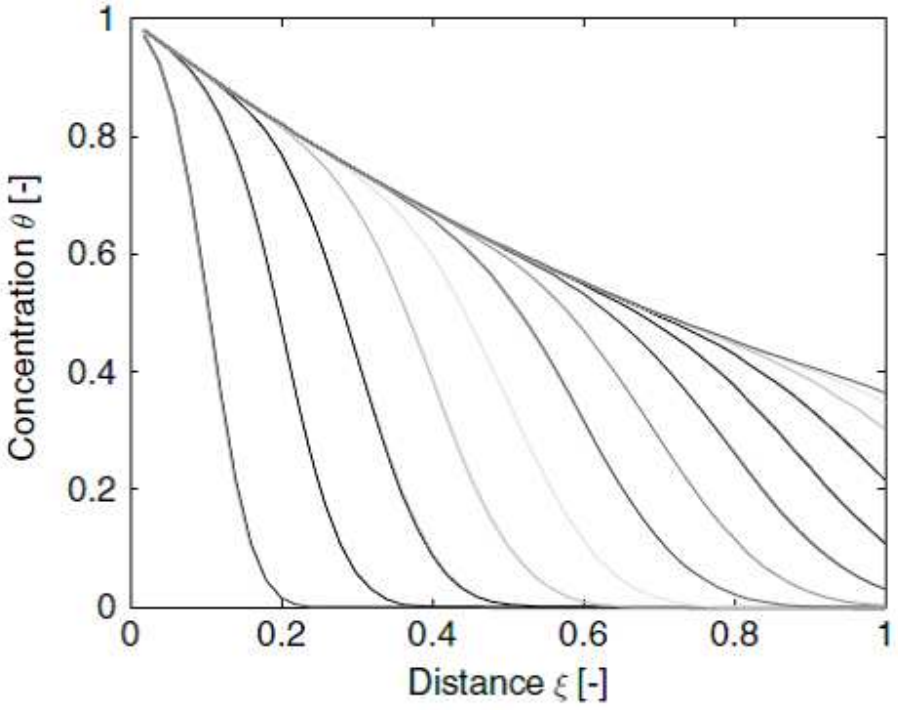
(2.30)

Avec $u = \sqrt{1+4\,Da_2/Pe}$. La figure 6 illustre la solution pour un nombre de Peclet élevé Pe = 100 et un deuxième nombre de Damkohler modéré. Le front procède dans la direction x positive de gauche à droite. Le temps de procéder d'une ligne de front à la suivante équivaut à la 10ème partie du temps moyen qu'un traceur aurait besoin de migrer à travers le système entier.

Avec le front intrusif la concentration est réduite.

En raison du nombre élevé de Péclet, la ligne de front reste relativement raide, ce qui a l’effet supplémentaire que la déviation de l'état d'équilibre est relativement modérée mais en augmentation. La parcelle a été obtenue en utilisant le fichier M "analtransnodim.m" avec les paramètres appropriés.

Les figures (2.6 et 2.7) montrent un comportement typique lorsque C_0 n'est pas égal à zéro. Dans ce cas, les deux termes des formules (3.29) et (3.30) doivent être pris en compte.



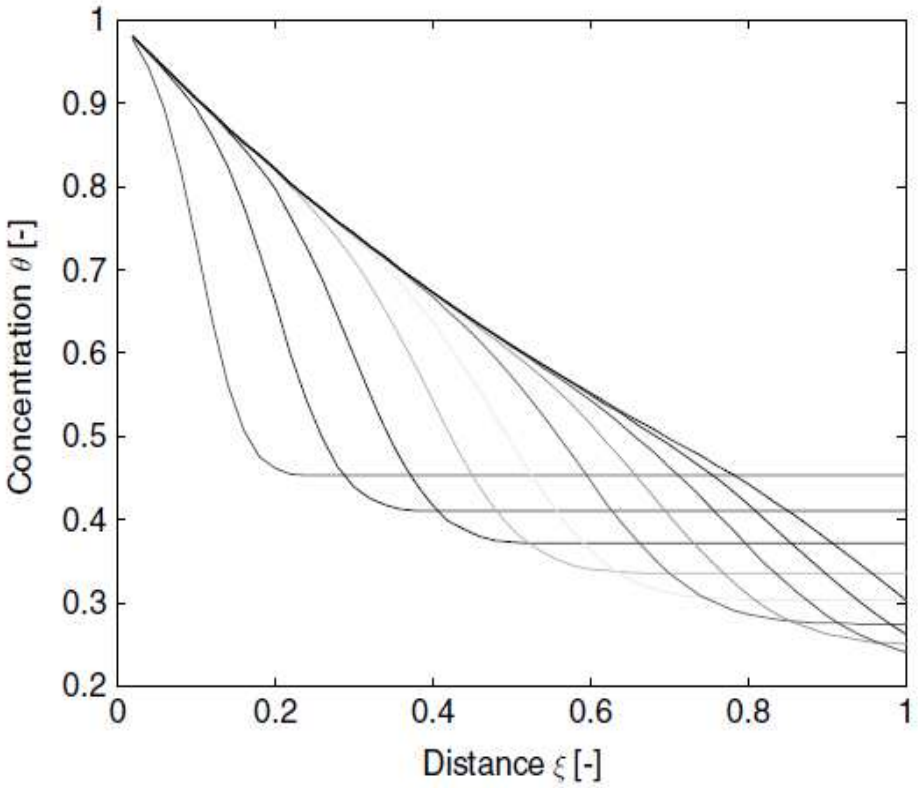


Figure 2.7. Solution en régime transitoire pour l'équation de transport avec dégradation : Pe = 100 et Da_2 = 1 pour $C_0 = C_{in}/2 = 0.5$

Le déclin des concentrations d'une ligne à l'autre du côté droit de la figure illustre la dégradation du matériau, initialement présent dans le système. Les concentrations croissantes du côté gauche proviennent de la progression de face. **[2]**

Conclusion

Ce travail s'inscrit dans le cadre global de la modélisation et la simulation des grands processus de transport de matière pour la dégradation de polluant ou d'éléments radioactifs se dissipant des milieux fluides : eau ou atmosphère. Les mécanismes choisis sont la diffusion, la convection et la réaction, simultanément ou séparément.

Ainsi nous avons pour le cas général :

Pour le régime permanent :

$$c(x) = \exp(\mu_2 x)\,(C_1 + C_0 \int \exp(\mu_1 x)\exp(-\mu_2 x)\,dx) = C_1 \exp(\mu_2 x) + \frac{C_0}{\mu_1 - \mu_2}\exp(\mu_1 x)$$

Pour le régime stationnaire :

$$c(x,t) = c_0 \exp(-\lambda t)\left(1 - \frac{1}{2} erfc\left(\frac{x - vt}{2\sqrt{Dt}}\right) - \frac{1}{2}\exp\left(\frac{vx}{D}\right) erfc\left(\frac{x + vt}{2\sqrt{Dt}}\right)\right) \ldots$$

$$+ \frac{c_{in}}{2}\left(\exp\left(\frac{v - u}{2D}x\right) erfc\left(\frac{x - ut}{2\sqrt{Dt}}\right) + \exp\left(\frac{v + u}{2D}x\right) erfc\left(\frac{x + ut}{2\sqrt{Dt}}\right)\right)$$

Ces deux modèles sont valables et applicables pour des tests en laboratoires.

Bibliographies

[1] : ANDERSON R.F./BOPP R.F./BUESSLER K.O/BISCAYE P.E, Mixing of particles and organic constituents in sediments from the continental shelf and shape off cape cod : SEEP-1 results, cont. Shelf res., vol.8, N_0 5-7, 925-946, 1998.

[2] : APPELO C.A/POSTMA D., Geochemistry, groundwater and pollution, Balkema, Rotterdam, 536p, 1993.

[3] : BALZAC Stephan/STURM Fréderic, 2003, Algèbre et Analyse cours de première année, Lausanne, sciences appliqués de l'INSA Lyon.

[4] : EKKERHARD O. Holzbecher, Environmental modelig using MATLAB, Springer, 393 pages, 2007.

[5] : HENDERSON G.M/ LINDSAY F.N/ SLAWEY N.C, Variation in bioturbation with water depth on marine shapes : a study on the little bahams bank, Marine geology, vol.160, 105-118, 1999.

[6] : MUSOMONI MUTAMBA, Christian, 2018, Développement de calcul en élasticité linéaire sous code MATLAB, projet de fin cycle de Licence, GENIE MECANIQUE, Mostaganem, Université de Mostaganem, page 4-11

[7] : WEXLER E.J, Analytical solutions for one, two, three dimensionnal solute transport in ground water systems with uniforms flow, Techniques of water ressources investigations of united states geological survey, Book3, Chapter B7, 190p, 1992.