HAL
open science

Analyse de sensibilité déterministe pour la simulation numérique du transfert de contaminants

Estelle Marchand

To cite this version:

Estelle Marchand. Analyse de sensibilité déterministe pour la simulation numérique du transfert de contaminants. Mathématiques [math]. Université Paris Dauphine - Paris IX, 2007. Français. NNT : . tel-00271632v2

HAL Id: tel-00271632

https://theses.hal.science/tel-00271632v2

Submitted on 10 Apr 2008

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNIVERSITÉ PARIS DAUPHINE
EDDIMO
INRIA

N° attribué par la bibliothèque

ANALYSE DE SENSIBILITÉ DÉTERMINISTE POUR LA SIMULATION NUMÉRIQUE DU TRANSFERT DE CONTAMINANTS

THÈSE

pour l'obtention du titre de

DOCTEUR EN SCIENCES

Spécialité : Mathématiques Appliquées

(Arrêté du 07 Août 2006)

Présentée et soutenue publiquement par

Estelle MARCHAND

JURY

Directeur de thèse :	**Mme Jean E. ROBERTS** Directeur de recherche à l'INRIA Rocquencourt
Rapporteurs :	**M. Grégoire ALLAIRE** Professeur à l'école Polytechnique
	M. Magne ESPEDAL Professeur à l'Université de Bergen
Président :	**M. Guy CHAVENT** Professeur Émérite à l'Université Paris Dauphine
Suffragants :	**M. François CLÉMENT** Chargé de recherche à l'INRIA Rocquencourt
	M. Guillaume PÉPIN Ingénieur ANDRA

le 12 Décembre 2007

L'université n'entend donner aucune approbation ni improbation aux opinions émises dans les thèses : ces opinions doivent être considérées comme propres à leurs auteurs.

Remerciements

Je remercie bien sûr Grégoire Allaire et Magne Espedal qui ont accepté de consacrer du temps à lire et rapporter sur ma thèse, et Guy Chavent qui a présidé le jury.

Merci à François Clément et Jean Roberts qui ont dirigé mes recherches après m'avoir proposé un sujet passionnant.

Merci à Guillaume Pépin et Laurent Loth qui ont suivi ce travail depuis l'ANDRA et m'ont proposé des applications intéressantes. Merci également à l'ANDRA qui a financé mes trois années de thèse et au GDR MOMAS qui a aussi soutenu ces travaux.

Merci au projet Estime et aux personnes avec qui j'ai eu la chance de pouvoir travailler, à Jérôme Jaffré, à Michel Kern, à Pierre Weis, ...

Merci à Amel Sboui et Vincent Martin dont les travaux de thèse m'ont été très utiles. Merci aussi à Marc Durufflé pour le C++.

Merci à tous d'avoir été attentifs et disponibles.

Ces trois années ont été très enrichissantes. J'ai aussi redécouvert le plaisir de travailler sereinement, c'est inestimable. Merci donc de m'avoir accueillie à l'INRIA et pour l'atmosphère du bâtiment 13.

Merci à Claire, Éric et Ludwig.

Merci à mes parents.

Table des matières

1 Introduction **13**
1.1 Contexte . . . 13
1.1.1 Le stockage en couches géologiques profondes . . . 14
1.1.2 Les fonctions de sûreté . . . 14
1.1.3 Les contaminants . . . 15
1.1.4 Les écoulements en milieux poreux saturés . . . 17
1.1.5 Le transport des radionucléides . . . 18
1.2 Modélisation des écoulements et du transport des contaminants dans les milieux poreux saturés . . . 19
1.3 Analyse de sensibilité . . . 21
1.3.1 Analyse de sensibilité pour le stockage de déchets radioactifs . . . 22
1.3.2 Approche déterministe de l'analyse de sensibilité . . . 23
1.3.3 Identification de paramètres et analyse de sensibilité . . . 24
1.3.4 Sensibilité à discrétisation . . . 25
1.4 Aspects implémentation . . . 25
1.4.1 Outils classiques de calcul scientifique . . . 25
1.4.2 Programmation fonctionnelle . . . 26
1.5 Écoulements diphasiques . . . 28
1.6 Plan du mémoire . . . 28

I Analyse de sensibilité pour un modèle d'écoulement et de transport en milieu poreux **29**

2 Modèles mathématiques et numériques **31**
2.1 Modèle d'écoulement . . . 31
2.1.1 Équation continue . . . 31
2.1.2 Formulation variationnelle . . . 32
2.1.3 Formulation discrète EFMH . . . 40
2.1.4 Implémentation des conditions aux limites . . . 47
2.2 Modèle de transport . . . 48
2.2.1 Équation continue . . . 49
2.2.2 Discrétisation en temps . . . 50

2.2.3 Discrétisation en espace 51
2.2.4 Introduction d'un terme de précipitation 55
2.3 Réduction de dimension du modèle 56

3 Analyse de sensibilité 61
3.1 Analyses probabilistes 61
3.1.1 Formalisme probabiliste 61
3.1.2 Échantillonnage 63
3.1.3 Indicateurs statistiques 63
3.2 Analyses déterministes 68
3.2.1 Analyse d'incertitude locale 68
3.2.2 Analyse de sensibilité 69
3.3 SVD et analyses statistiques 70
3.3.1 Décomposition de variables 70
3.3.2 Utilisation de la SVD en analyse statistique 72
3.3.3 Transformation de Karhunen-Loève (KLT) 73
3.3.4 Analyse en Composantes Principales (PCA) 75
3.3.5 Cas linéaire avec entrées indépendantes 75
3.3.6 Extension au cas non linéaire avec entrées dépendantes 76

4 Calcul de dérivées 79
4.1 Différences divisées 79
4.1.1 Utilisation en vérification 79
4.1.2 Illustrations numériques 80
4.2 Différentiation automatique 83
4.2.1 Différentiation automatique par transformation de source 83
4.2.2 Différentiation automatique par surcharge d'opérateur 85
4.3 Différentiation manuelle 89
4.3.1 Dérivation en mode direct avec application au problème d'écoulement 89
4.3.2 Dérivation en mode inverse 92
4.4 Comparaison de performances 97
4.4.1 Présentation des cas tests 97
4.4.2 Résultats numériques 100
4.4.3 Conclusion 101

5 Décomposition de domaine 103
5.1 Décomposition de domaine sans recouvrement 103
5.1.1 Idée générale 103
5.1.2 Formulation 104
5.1.3 Algorithme de décomposition de domaine 105
5.2 Dérivation de la formulation décomposition de domaine 106
5.2.1 Différentiation en mode direct 107
5.2.2 Différentiation en mode adjoint 107

6 Aspects informatiques **109**
6.1 Algèbre linéaire 109
6.2 Éléments finis : la bibliothèque `LifeV` 109
6.3 Différentiation avec `ADOL-C` 110
6.3.1 Modification des sources 110
6.3.2 Méthodes itératives 112
6.3.3 Remarque sur l'installation de la bibliothèque 113
6.3.4 Utilisation combinée de la différentiation manuelle et de la différentiation automatique 116
6.4 Paradigme pour l'implémentation d'algorithmes complexes 116
6.4.1 Principe 116
6.4.2 Structure générale 117
6.4.3 Parallélisation automatique avec `OCamlP3l` 118
6.5 Plate-forme de couplage de codes 120
6.5.1 Utilisation d'un outil de couplage. 120
6.5.2 Communications pour la décomposition de domaine 120
6.6 Plate-forme d'analyse de sensibilité déterministe 121
6.7 Implémentation `C++` du protocole de communication 123

7 Applications **127**
7.1 Écoulement tridimensionnel : étude des flux aux exutoires et comparaison avec des résultats probabilistes 127
7.1.1 Choix des opérateurs de paramétrisation et d'observation 127
7.1.2 Choix d'une méthode de dérivation 129
7.1.3 Domaine de calcul 130
7.1.4 Données pour les paramètres d'entrée 132
7.1.5 Résultats attendus 132
7.1.6 Résultats probabilistes 133
7.1.7 Résultats déterministes et interprétation 133
7.1.8 Conclusions 139
7.2 Écoulement tridimensionnel : étude déterministe des variations locales . . . 140
7.2.1 Description du cas-test 140
7.2.2 Décomposition en valeurs singulières 140
7.3 Écoulement et transport bidimensionnels : étude des flux aux exutoires et comparaison avec des résultats probabilistes 143
7.3.1 Choix du terme source 143
7.3.2 Définition des fonctions à analyser 143
7.3.3 Résultats probabilistes 145
7.3.4 Résultats déterministes 145

II Sensibilité par rapport à la discrétisation 149

8 De la dérivation par rapport aux paramètres de discrétisation 151
8.1 Sensibilité à la discrétisation en temps 151
8.1.1 Équation différentielle ordinaire 151
8.1.2 Équation de diffusion 154
8.1.3 Équation de diffusion/convection 155
8.2 Sensibilité à la discrétisation en espace 156
8.2.1 Laplacien en dimension 2 avec un maillage à cellules carrées 156
8.2.2 Tentative de construction d'une fonction continue de h et de l'espace 157
8.2.3 Suggestion d'approximation de dérivée 159
8.2.4 Exemples 159

III Écoulements diphasiques eau/air en milieu poreux 161

9 De la prise en compte des variations de la pression en air dans la modélisation des écoulements souterrains eau/air 163
9.1 Les écoulements diphasiques 163
9.1.1 Caractéristiques de l'écoulement 163
9.1.2 Lois de l'écoulement 165
9.2 La formulation pression globale 168
9.2.1 Fonction auxiliaire 169
9.2.2 Définition de la pression globale 169
9.2.3 Propriétés du nouveau système d'équations 171
9.2.4 Conclusion 177
9.3 L'équation de Richards 177
9.4 Discrétisation 179
9.4.1 Un problème simple en dimension 1 - approximation de Richards . . 180
9.4.2 Un problème simple en dimension 1 - modèle diphasique 182
9.4.3 Résultats numériques comparatifs 184
9.5 Conclusion 185

10 Conclusion générale 189

A Quelques éléments techniques sur la dérivation du programme de transport 191
A.1 Principe 191
A.2 Structure du code de transport 192
A.2.1 Paramètres du problème. 193
A.2.2 Paramètres discrets. 193
A.2.3 Variables intermédiaires. 194
A.2.4 Variables intermédiaires discrètes. 194

A.2.5 Déroulement du calcul. 194
A.2.6 Découpage du code : choix des enregistrements. 195
A.3 Enchaînement des pas de temps en mode adjoint 196
A.3.1 Recalculs au cours d'un pas de temps 197

B Résultats exhaustifs pour l'analyse déterministe du problème d'écoulement 201
B.1 Tests avec tous les paramètres sauf un à leur valeur la plus probable 201
B.2 Exploration des sommets de l'ensemble des valeurs possibles pour les paramètres d'entrée 206
B.2.1 Résultats où les deux derniers vecteurs singuliers de l'espace d'entrée sont légèrement déformés 206
B.2.2 Résultats où le premier vecteur singulier de l'espace d'entrée comprend une influence de k_8 211
B.2.3 Résultats où la hiérarchie des influences est modifiée 218
B.2.4 Résultats presque identiques à ceux de la première étude locale . . . 225

Chapitre 1

Introduction

1.1 Contexte

Comme toute activité industrielle, l'industrie nucléaire produit des déchets. Des précautions considérables sont prises pour gérer les déchets radioactifs, quels qu'ils soient, ceux-ci étant, bien entendu, soumis à une réglementation très précise et à des contrôles fréquents et approfondis.

La loi de programme n°2006-739 du 28 juin 2006 relative à la gestion durable des matières et déchets radioactifs affirme et renforce le positionnement de l'ANDRA comme acteur de référence dans la gestion durable de tous les déchets radioactifs produits par la France. Cette loi précise les recherches et études relatives aux déchets de haute activité (HA) et de moyenne activité à vie longue (MA-VL). Elle fait suite à la loi du 30 décembre 1991 qui fixait une période de quinze années au terme de laquelle le gouvernement devait remettre ses conclusions au parlement sur la gestion des déchets HA et MA-VL. Trois axes de recherche y étaient retenus : la séparation poussée-transmutation, le stockage en profondeur et l'entreposage de longue durée.

À partir des études et recherches menées entre 1991 et 2005, l'ANDRA a remis au gouvernement le « Dossier 2005 » qui conclut à la faisabilité du stockage géologique réversible. C'est sur la base des éléments de ce dossier, des conclusions du CEA sur les deux autres axes de recherche et d'un débat public, que la promulgation de la nouvelle loi de 2006 a eu lieu. Dans le cadre de la nouvelle loi, les trois axes de recherche sont confirmés pour la gestion des déchets HA et MA-VL :

- la séparation et la transmutation des éléments radioactifs à vie longue, confié au CEA ;
- le stockage réversible en couche géologique profonde, confié à l'ANDRA. Solution de référence (voir [71]), cet axe doit permettre d'aboutir au choix d'un site et à la conception d'un centre de stockage de sorte qu'en 2015, une demande d'autorisation de création puisse être instruite et qu'en 2025, sous réserve de cette autorisation, le centre de stockage puisse être mis en exploitation. Dans ce cadre, elle exploite le Laboratoire de recherche souterrain de Meuse/Haute-Marne situé à 500 mètres de

profondeur.
- l'entreposage, piloté par l'ANDRA. Cet axe vise d'ici 2015 à la création de nouvelles installations d'entreposage ou à la modification des installations existantes pour répondre aux besoins recensés.

1.1.1 Le stockage en couches géologiques profondes

Le travail de la thèse s'inscrit dans le contexte du projet de stockage en formation géologique profonde de déchets radioactifs haute activité et vie longue (HAVL). L'objectif fondamental de la gestion à long terme des déchets HAVL est de protéger, sur une très grande durée, l'homme et l'environnement des risques associés à ces déchets. La réponse apportée par un stockage consiste à confiner ces déchets dans une formation géologique profonde pour s'opposer à la dissémination des radionucléides qu'ils contiennent. Ce confinement s'effectue sur de grandes échelles de temps (jusqu'à plusieurs centaines de milliers d'années), de manière passive, c'est-à-dire sans nécessiter à très long terme de maintenance ou de surveillance, comme le rappelle la règle fondamentale de sûreté (RFS III.2.f).

Les études inhérentes au stockage de déchets HAVL en formation géologique profonde menées par l'ANDRA s'appuient sur les caractéristiques du site de Meuse/Haute-Marne et s'attachent à évaluer les conditions dans lesquelles on pourrait construire, exploiter, gérer de manière réversible, fermer, surveiller un site de stockage, puis le laisser évoluer sans aucune intervention humaine et sans qu'à aucun moment la sécurité des travailleurs, du public et la protection de l'environnement ne soient compromises. La sûreté est ainsi au centre de la démarche de conception du stockage.

1.1.2 Les fonctions de sûreté

L'objectif fondamental d'un stockage en formation géologique profonde est rappelé par la règle fondamentale de sûreté RFS.III.2.f. : « La protection des personnes et de l'environnement à court et à long terme constitue l'objectif fondamental assigné à un centre de stockage de déchets en formation géologique profonde ».

Cette protection des personnes et de l'environnement est réalisée en s'opposant à la dissémination des radionucléides contenus dans les déchets, sans reposer à terme sur une maintenance ou une surveillance. Pour s'opposer à la dissémination radioactive aux différentes échelles de temps, on confère à une installation de stockage en formation géologique profonde des fonctions de sûreté multiples : en se complétant mutuellement, celles-ci optimisent les performances globales du système ; en offrant une certaine redondance, elles permettent de mieux résister à une défaillance ou à une perturbation externe.

Le principe de sûreté du stockage repose sur trois barrières de confinement : le colis de déchets (les déchets sont immobilisés dans une matrice de verre à dégradation très lente), la barrière ouvragée (béton ou argile) et la barrière géologique (argilites saines du Callovo-Oxfordien).

En profondeur, l'eau souterraine constitue le principal vecteur permettant un transport de radionucléides. Dans ce contexte, le stockage doit être conçu pour (i) s'opposer à la cir-

culation d'eau, (ii) limiter le relâchement des radionucléides et les immobiliser à l'intérieur du stockage lui-même, ce qui renvoie notamment à un besoin de « protéger » les déchets, (iii) retarder et atténuer la migration des radionucléides qui auraient été relâchés par les déchets et (iv) préserver les propriétés favorables du milieu en limitant les perturbations. Ce sont ces fonctions de sûreté qui doivent être remplies par un stockage.

Pour cela, on mobilise le plus largement les propriétés favorables des argilites du Callovo-Oxfordien (faible perméabilité, capacité de rétention, propriétés géochimiques, environnement hydrogéologique). L'âge de la formation, sa stabilité tectonique, la profondeur d'implantation du stockage permettent d'envisager une grande stabilité de ces propriétés favorables, aux échelles de temps étudiées ici (de l'échelle du millier d'années à celle de plusieurs centaines de milliers d'années). Au-delà de la phase d'exploitation et d'observation du stockage, les fonctions de sûreté d'un stockage sont passives, c'est-à-dire qu'elles ne nécessitent aucune intervention humaine.

On s'intéresse dans cette thèse à la barrière géologique, c'est-à-dire au transport des contaminants dans le milieu poreux après rupture de la barrière ouvragée.

Nous allons présenter les différentes composantes de notre modèle : les contaminants eux-mêmes, les milieux poreux (modèle pour la barrière géologique) et l'écoulement de l'eau dans ces milieux, et les paramètres supplémentaires caractérisant l'écoulement des contaminants.

1.1.3 Les contaminants

1.1.3.1 Définitions

Les éléments constitutifs du noyau de l'atome sont les nucléons, la masse de l'atome étant presque proportionnelle à son nombre de nucléons (autour du noyau gravitent des électrons de masse négligeable). On distingue deux sortes de nucléons : les protons, chargés électriquement positivement, et les neutrons, électriquement neutres. Un élément chimique est défini par un nombre de protons. Les isotopes sont des formes d'un même élément chimique, dont les noyaux possèdent un nombre de protons identique et un nombre de neutrons différent. La plupart des éléments ont une forme stable, ses autres formes sont dites instables ou radioactives. Un radionucléide est un isotope radioactif d'un élément. La désintégration est la transformation d'un noyau instable en noyau plus stable, au cours de laquelle le nombre et la nature des nucléons sont modifiés. La radioactivité est la propriété que possèdent certains éléments chimiques de se transformer spontanément en un autre élément par désintégration, en dégageant de l'énergie sous forme de rayonnement. Pendant la désintégration, une quantité importante de chaleur est dégagée. De plus, différents types de rayonnement peuvent être émis en fonction de l'élément d'origine. La radioactivité α se caractérise par l'émission d'un noyau d'Hélium, i.e. deux protons et deux nucléons. La radioactivité β^- est la conversion d'un neutron en proton et est caractérisée par l'émission d'un électron (ou particule β^-) et d'un anti-neutrino. Elle affecte les radionucléides ayant un excès de neutrons. La radioactivité β^+ est la conversion d'un proton en neutron et est caractérisée par l'émission d'un positron (ou particule β^+) et d'un neutrino. Elle affecte

les radionucléides ayant un excès de protons. Les désintégrations α, β^- et β^+ sont accompagnées de l'émission de photons de haute énergie, ou rayons γ, dont les longueurs d'onde sont généralement encore plus courtes que celles des rayons X, étant de l'ordre de 10^{-9}m ou inférieures. Cette émission γ a lieu lors des transitions électroniques à partir de niveaux d'énergie excités (des changements d'orbite des électrons), avec des énergies mises en jeu de l'ordre du MeV.

La demie-vie ou période radioactive $T_{1/2}$ d'un isotope radioactif est la durée au bout de laquelle la moitié des noyaux radioactifs se sont désintégrés. La loi de décroissance radioactive est exponentielle :

$$\frac{\mathrm{d}N(t)}{\mathrm{d}t} = -\lambda N(t)$$

où $N(t)$ est le nombre d'isotopes radioactifs à l'instant t et λ est la constante de décroissance radioactive :

$$\lambda = \frac{\ln 2}{T_{1/2}}.$$

L'unité de mesure d'activité est le Becquerel (Bq), elle correspond à une désintégration par seconde. L'activité totale d'un système fermé décroît comme le nombre de noyaux radioactifs. Pour donner des ordres de grandeur, la radioactivité de l'eau de mer est de 12 Bq.kg^{-1}, l'activité d'un combustible usé dans une centrale nucléaire est de 10^{19} Bq.

1.1.3.2 Les colis de déchets

Les déchets de l'industrie électronucléaire proviennent essentiellement des combustibles usés déchargés des réacteurs de production d'électricité. Actuellement ces combustibles sont retraités par Cogema dans ses usines de La Hague : des résidus sont alors séparés de l'uranium et du plutonium, ce sont les produits de fission et actinides mineurs, ainsi que les structures mécaniques des assemblages de combustible (tronçons de gaines, pièces d'embouts). Pour l'étude du stockage, on considère l'ensemble des déchets engagés par le parc électronucléaire actuel, sur la base d'une hypothèse de durée moyenne de fonctionnement des réacteurs de quarante années.

Les déchets dits de haute activité et à vie longue (ou HAVL) contiennent à la fois des radionucléides à vie courte, généralement en quantité importante (haute activité) et des radionucléides à vie longue en quantité moyennement à très importante. Leur contenu en radionucléides à vie longue renvoie plus particulièrement au risque d'ingestion entraînant une exposition de tissus vivants au rayonnement α ; la période radioactive de certains isotopes dépasse la centaine de milliers d'années. Une grande partie des déchets HAVL présentent aussi une haute activité de rayonnement γ, qui implique de protéger l'homme d'une radio-exposition externe. L'activité $\beta - \gamma$ présente dans les déchets HAVL décroît relativement rapidement dans le temps : ainsi, après quelques dizaines d'années, les combustibles nucléaires ne contiennent plus que quelques pour cent de la radioactivité qu'ils présentaient lors du déchargement du réacteur. L'énergie générée par la radioactivité est convertie essentiellement en chaleur : le rayonnement est absorbé dans la matière même constituant le colis de déchet et, pour une plus faible part, dans la matière située à son

voisinage immédiat. Lorsque la radioactivité $\beta - \gamma$ a très fortement décru (au bout de quelques siècles), l'énergie radioactive résiduelle, associée aux isotopes à longue période, est très faible, et la chaleur produite devient alors non significative. La décroissance de l'activité $\beta - \gamma$ dans le temps suggère, pour les colis les plus actifs, une période intermédiaire d'attente, entre leur production et leur stockage. Cette attente peut être réalisée dans des installations d'entreposage. Elle permet de diminuer la chaleur produite par les déchets, qui intervient sur le dimensionnement des installations de stockage et sur leur emprise dans la formation d'accueil.

1.1.4 Les écoulements en milieux poreux saturés

Un milieu poreux est un milieu composé d'une matrice solide et d'un espace vide pouvant être occupé par un ou plusieurs fluides. De plus sa géométrie doit respecter les propriétés suivantes :

- les espaces vides du milieu poreux doivent être interconnectés (l'écoulement est possible et tous les espaces vides sont susceptibles d'être occupés par les fluides) ;
- les dimensions de l'espace vide sont grandes devant la longueur du trajet libre moyen des molécules fluides ;
- les dimensions de l'espace vide sont suffisamment petites pour que l'écoulement des fluides soit contrôlé par les forces de frottement aux interfaces fluide-solide et les forces de cohésion aux interfaces fluide-fluide.

Les deux dernières conditions permettent de définir la **porosité** du milieu ϕ (fraction de volume occupée par les pores) et le volume élémentaire représentatif. Soit x un point du milieu. Soit Φ la fonction caractéristique de l'espace vide,

$$\Phi(x) = \begin{cases} 0 & \text{si } x \text{ appartient à la matrice solide} \\ 1 & \text{sinon.} \end{cases}$$

Pour $r > 0$ on note $B(x, r)$ la boule de centre x et de rayon r. On définit

$$\Phi_r(x) = \frac{1}{\text{mes}(B(x,r))} \int_{B(x,r)} \Phi(y) \mathrm{d}y.$$

La Figure 1.1 représente l'allure de cette fonction. La valeur retenue pour $\phi(x)$ est Φ. Les rayons compris entre $r_{\min}$ et $r_{\max}$ correspondent aux volumes représentatifs élémentaires possibles.

La notion de **perméabilité** k quantifie l'influence de la géométrie de la matrice solide sur la vitesse d'écoulement des fluides dans la roche. Elle dépend de ϕ, des liaisons entre les pores et de la nature des fluides. Dans le cas général, k est un tenseur.

Un fluide est caractérisé par sa **viscosité** μ, sa **masse volumique** ρ que nous considérerons constantes dans le cas de l'eau, et par sa **pression** p, qui varie a priori dans l'espace.

L'écoulement de l'eau est représenté par la vitesse d'écoulement volumique $\vec{u}$.

Nous nous intéresserons, sauf dans le Chapitre 9, à un écoulement dans un milieu saturé en eau, ce qui signifie que la **teneur en eau** du milieu, i.e. la proportion d'un volume

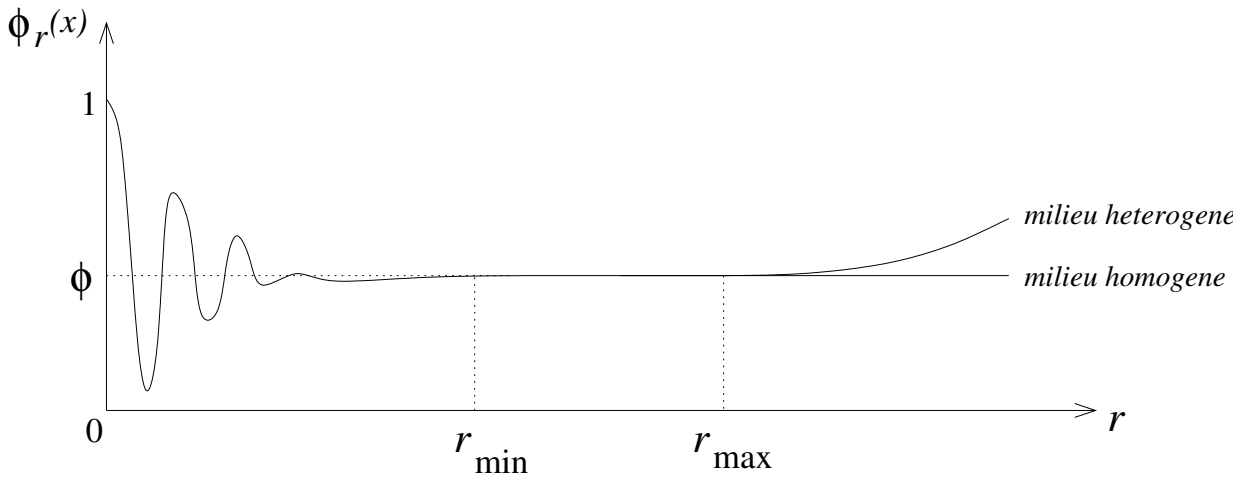


FIG. 1.1 – Porosité et volume représentatif élémentaire. La courbe (fictive) représente la proportion d'espace vide dans la boule de centre x et de rayon r, en fonction du rayon r.

représentatif élémentaire occupée par l'eau, est la plus grande possible en tout point. La plus grande valeur possible pour la teneur en eau est majorée par la porosité.

La **conductivité hydraulique** du milieu poreux est définie comme le rapport (pas forcément scalaire) entre la perméabilité k et la viscosité de l'eau μ. Le gradient de pression $\vec{\nabla} p$ est à l'origine du déplacement des masses d'eau, l'eau se déplaçant des pressions les plus hautes vers les pressions les plus basses. L'intensité de la pesanteur a également un rôle sur ce déplacement, que nous négligerons dans notre étude. Selon la loi de Darcy, le tenseur de conductivité hydraulique est le rapport entre le flux volumique $\vec{u}$ d'écoulement de l'eau, aussi appelé vitesse de Darcy, et le gradient en pression.

1.1.5 Le transport des radionucléides

Les contaminants s'échappant de la barrière ouvragée sont dissous dans l'eau. Leur déplacement dans la barrière géologique est dû d'une part au déplacement global de l'eau, quantifié par la vitesse de Darcy, ou, du point de vue des contaminants, la vitesse de convection, et d'autre part au phénomène de diffusion, qui est la tendance des concentrations dissoutes à devenir homogènes. Le rapport entre la vitesse de diffusion des contaminants et le gradient de concentration est le tenseur de diffusion, qui est la somme d'un tenseur de diffusion moléculaire, scalaire, dépendant de la nature du contaminant considéré, et d'un tenseur de diffusion mécanique, plein, dépendant également de la vitesse de Darcy.

On peut également observer pour certains contaminants un phénomène de précipitation : leur concentration dissoute ne peut pas dépasser un certain seuil. Lorsqu'elle atteint un seuil critique c^{sat}, les radionucléides en surnombre précipitent, c'est-à-dire se fixent sur la matrice rocheuse sous forme solide. Aux points où un contaminant a précipité, sa concentration en phase liquide est localement constante, égale à la concentration de saturation c^{sat}, et on a un échange entre la phase liquide et la phase solide quantifié par un coefficient

d'échange, qui est une constante pour un radionucléide donné.

Notons également que dans le cas le plus général on doit aussi prendre en compte des phénomènes de filiations (un radionucléide "père" peut former, en se désintégrant, d'autres radionucléides moins instables).

1.2 Modélisation des écoulements et du transport des contaminants dans les milieux poreux saturés

À partir des quantités introduites dans la section précédente, nous allons présenter rapidement les équations aux dérivées partielles qui constitueront le modèle mathématique de notre problème.

Le modèle mathématique que l'on considère consiste en un système couplé de plusieurs équations : l'équation de l'écoulement et une équation de transport par radionucléide. Comme l'équation de l'écoulement (1.1) est indépendante du temps et de la concentration, on peut la traiter d'abord et résoudre ensuite de façon séparée les équations de bilan dans les phases liquide et solide. Il est possible que la viscosité μ de l'eau dépende de la concentration de certains contaminants, mais on suppose généralement que les radionucléides sont présents en quantités suffisamment faibles pour avoir une influence négligeable sur la viscosité μ.

Soit Ω le domaine considéré. L'équation de l'écoulement s'écrit

$$\left\{ \begin{array}{rcll} \text{div } \vec{u} & = & \tau & \text{dans } \Omega \\ \vec{u} & = & -\dfrac{k}{\mu}\left(\vec{\nabla} p - \rho \vec{g}\right) & \text{dans } \Omega \end{array} \right. \tag{1.1}$$

où $\vec{u}$ est la vitesse de Darcy, p la pression, k la perméabilité du milieu, ρ la masse volumique du fluide et μ sa viscosité, $\vec{g}$ l'accélération de la pesanteur et τ un terme source (en fait il sera nul). Nous considérerons la conductivité hydraulique $\mathcal{K} = \dfrac{k}{\mu}$.

La conservation de la masse de chaque radionucléide est exprimée de façon séparée pour la partie dissoute (1.2) et pour la partie précipitée (1.3).

L'équation de bilan pour le radionucléide i en phase liquide est une équation de transport de la forme

$$\frac{\partial}{\partial t}(\phi R_i c_i) + \text{div } (c_i \vec{u} - D_i \vec{\nabla} c_i) + \lambda_i \phi R_i c_i = \sum_{j \in \{pères(i)\}} \sigma_{ij} \lambda_j \phi R_j c_j + \phi s_i + \tau_i \qquad \text{dans } \Omega, \tag{1.2}$$

où c_i est la concentration du radionucléide i dissout dans l'eau, R_i le coefficient de retard et λ_i la constante de décroissance radioactive pour ce radionucléide, ϕ est la porosité cinématique du milieu, $\vec{u}$ la vitesse de Darcy, t le temps, D_i le tenseur de dispersion/diffusion, $\{pères(i)\}$ est l'ensemble des pères du radionucléide i, σ_{ij} est la fraction du radionucléide j se désintégrant en le radionucléide i, s_i est le terme d'échange entre les

phases solide et liquide pour le radionucléide i et τ_i le terme source pour le radionucléide i, dépendant du contenu et de l'état de la barrière ouvragée.

L'équation de bilan pour le radionucléide i en phase solide est une équation différentielle en temps de la forme

$$\frac{\partial}{\partial t}[(1-\phi)f_i] + \lambda_i(1-\phi)f_i = \sum_{j\in\{pères(i)\}} \sigma_{ij}\lambda_j(1-\phi)f_j - \phi s_i \qquad \text{dans } \Omega, \tag{1.3}$$

où f_i est la concentration de radionucléide i précipité dans la matrice rocheuse.

Les équations de bilan contiennent des termes d'accumulation $\frac{\partial}{\partial t}(\phi R_i c_i)$ (ou $\frac{\partial}{\partial t}[(1-\phi)f_i]$), de dispersion/diffusion div $(-D_i\vec{\nabla}c_i)$, d'advection div $(c_i\vec{u})$ et de décroissance radioactive $\lambda_i\phi R_i c_i$ (ou $\lambda_i(1-\phi)f_i$). Précisons que le coefficient D_i dans le modèle le plus simple est un scalaire et que seule la diffusion moléculaire est prise en compte. Dans un modèle un peu plus sophistiqué, D_i est une matrice comportant aussi une partie dispersive tensorielle dépendant de la vitesse de Darcy $\vec{u}$.

La loi d'échange entre les phases liquide et solide pour le radionucléide i est de la forme

$$s_i = \omega_i(c_i^{\text{sat}} - c_i)\delta_i \qquad \text{avec } \delta_i = \begin{cases} 0 & \text{si } f_i = 0 \text{ et } c_i < c_i^{\text{sat}} \\ 1 & \text{sinon} \end{cases} \tag{1.4}$$

où ω_i est la constante cinétique de précipitation/dissolution du radionucléide i et c_i^{sat} est la concentration de saturation du radionucléide i.

La loi d'échange entre les deux phases (1.4) exprime la prise en compte de la précipitation par isotope. Elle correspond au couplage entre les équations de bilan via leur terme source. Elle n'est pas différentiable par rapport aux concentrations aux points où $f_i = 0$ et $c_i = c_i^{\text{sat}}$. Un modèle plus sophistiqué qui prendrait en compte la précipitation par élément devrait répartir la quantité de précipité entre les différents isotopes, et donc évaluerait à chaque instant la concentration à saturation de l'isotope i en fonction de la concentration à saturation de l'élément et des concentrations en phase liquide et solide de tous les isotopes de l'élément considéré, par exemple au prorata de la concentration des différents isotopes.

Dans les applications numériques, on supposera que la matrice de diffusion D_i est diagonale, c'est-à-dire que l'on ne prendra pas en compte l'influence de la vitesse de Darcy sur la matrice de diffusion. Le modèle le plus simple consiste également à ne pas tenir compte de la filiation, c'est-à-dire que l'ensemble $\{pères(i)\}$ des pères du radionucléide considéré devient vide, et que la somme correspondante dans les termes source disparaît. On supposera que $\{pères(i)\} = \emptyset$, donc les équations de transport pour les différents radionucléides seront indépendantes. Enfin, on considérera uniquement des radionucléides qui ne précipitent pas. On a mentionné l'équation de transport en phase solide et on donnera une idée de régularisation, mais on ne résoudra pas cette équation.

La fiabilité des résultats dépend de la qualité du modèle choisi et de la fiabilité des valeurs de ses paramètres d'entrée. Intéressons-nous donc à l'analyse de sensibilité, qui entre autre relie la fiabilité des résultats à celle de la mesure des paramètres d'entrée.

1.3 Analyse de sensibilité

Les questions de sûreté et d'incertitudes sont au centre des études de faisabilité d'un stockage souterrain de déchets. Des efforts sont faits dans toutes les disciplines pour répondre à ces questions, et des méthodologies de sûreté sont élaborées. Le calcul scientifique doit participer à cet effort.

Les études de sûreté présentent des incertitudes de différents types : les incertitudes sur les scénarios (par exemple, la date à laquelle la barrière ouvragée va commencer à se dégrader ou l'éventualité d'un séisme dans la zone de stockage), l'incertitude sur les modèles (les lois choisies pour modéliser le transfert de contaminants sont-elles satisfaisantes ?), l'incertitude sur les paramètres de ces lois.

On s'intéresse dans cette thèse au troisième type d'incertitudes (l'incertitude sur les paramètres des équations d'écoulement et de transport). L'incertitude sur les paramètres peut être de nature épistémique, c'est-à-dire due au manque de connaissance ou à des techniques de mesure imparfaites, ou de nature stochastique, c'est-à-dire correspondant à des événements aléatoires, à une variabilité spatiale ou temporelle (les différentes couches géologiques étudiées ne sont pas parfaitement homogènes), ou à des défauts de fabrication (concernant la fabrication des colis de déchets par exemple). Donnons des indications sur l'origine de certaines incertitudes des paramètres de notre étude. La porosité est une propriété directement mesurable. Les incertitudes sur la connaissance de ce paramètre sont donc dues à sa variabilité spatiale (on ne peut la mesurer qu'en un nombre limité de points). La pression est une propriété directement mesurable (à l'aide d'un transducteur de pression), sa mesure associée à celle de la vitesse d'écoulement permet de manière indirecte d'évaluer des conductivités hydrauliques (par des techniques d'identification de paramètres, voir la section 1.3.3).

Un des points importants à prendre en compte est le problème de l'évaluation des incertitudes sur les flux de contaminants à travers les exutoires ou sur d'autres indicateurs de sûreté dues aux incertitudes sur les paramètres d'entrée. Au fur et à mesure que les études s'affinent il sera de plus en plus nécessaire de présenter non pas une courbe de concentration au cours du temps, mais une famille de courbes ou une enveloppe d'incertitude autour d'une courbe afin de représenter les incertitudes sur la quantité calculée (voir Figure 1.2). Il est également important de pouvoir hiérarchiser les différents paramètres en fonction de leur influence sur les indicateurs de sûreté.

Deux aspects différents de la quantification de l'influence des paramètres sur les indicateurs de sûreté sont l'analyse d'incertitude, qui correspond à la quantification des incertitudes concernant les indicateurs (par exemple sous la forme de distributions ou de quantiles) et l'analyse de sensibilité, qui correspond à l'identification des poids des paramètres d'entrée en fonction de leurs influence sur les indicateurs [13, 12].

Ceci peut être obtenu par des méthodes probabilistes [70], comme les méthodes de type Monte-Carlo, et une méthodologie très sophistiquée a été développée aux États-Unis pour les projets Wipp et Yuca Mountain [48]. Ces méthodes sont aussi utilisées dans diverses industries, comme par exemple l'industrie pétrolière. Ces méthodes donnent de bons résultats et sont relativement faciles à mettre en œuvre, mais elles sont coûteuses car

elles nécessitent un grand nombre de simulations (voir par exemple [11]). Ainsi, pour les problèmes de grande taille, ce qui est typiquement le cas pour les calculs tridimensionnels, il faut simplifier les modèles, voire recourir à des approximations par surfaces de réponse. Elles ont vocation à tenir compte de la plage de variation complète des paramètres d'entrée, c'est pourquoi on dit que ce sont des méthodes globales.

Cependant les autorités françaises de sûreté ont des réticences à utiliser de telles méthodes et préfèrent les méthodes déterministes. De toutes façons, du point de vue de la sûreté il est préférable d'avoir à sa disposition plusieurs méthodes de traitement des incertitudes afin de pouvoir confronter les résultats obtenus, ce qui leur donne plus de validité. Enfin la méthode des sensibilités déterministes est nettement moins gourmande en temps de calcul que la méthode probabiliste.

Ce gain en temps de calcul s'accompagne d'une dégradation, par rapport à la méthode probabiliste, de la qualité de l'information—elle ne fournira pas autant d'information que la méthode probabiliste car elle ne permet pas d'associer une densité de probabilité à chacune des courbes de concentration qui pourrait se trouver à l'intérieur du domaine d'incertitude calculé—et d'une complexité plus grande de la mise en œuvre.

Malgré cela la méthode déterministe de calcul des sensibilités, grâce à son coût très inférieur, mérite d'être développée, et c'est l'objet du présent travail.

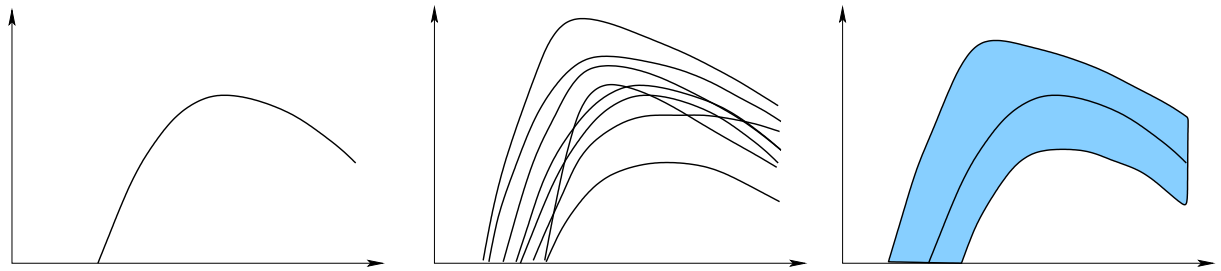


Fig. 1.2 – Concentration d'un contaminant à un exutoire en fonction du temps. À gauche, un résultat pour un jeu de données probable ; au centre, un ensemble de courbes obtenues par la méthode de Monte-Carlo ; à droite, un intervalle d'incertitude, encadrant le résultat calculé pour un jeu de données probable, obtenu par la méthode des sensibilités déterministes.

Le point fort de cette thèse est la confrontation sur un exemple réaliste des résultats fournis par une étude probabiliste de type Monte Carlo à ceux provenant d'une étude déterministe locale.

1.3.1 Analyse de sensibilité pour le stockage de déchets radioactifs

Différents types d'études de sensibilité ont été menées dans le cadre des évaluations de sûreté d'un stockage de déchets HAVL. Elles sont

– de type déterministe mono ou multi-paramétrique, en affectant des jeux de données dédiés correspondant à des situations ou des états particuliers du stockage (valeurs minimales et/ou maximales par exemple)
– de type probabiliste (multi-paramétriques) de type Monte-Carlo, avec comme objectif de quantifier l'incertitude des différents indicateurs de sûreté calculés et d'identifier, en terme de variance, les paramètres dont l'incertitude influe le plus sur l'incertitude du résultat

1.3.2 Approche déterministe de l'analyse de sensibilité

On propose donc dans cette thèse de traiter la question des incertitudes par une méthode déterministe. Cette méthode fournit des sensibilités du premier ordre, donc une information locale : pour des problèmes non linéaires, le résultat obtenu sera correct simplement pour des petites variations des paramètres d'entrée (la signification de "petite" dépend de l'ampleur des non linéarités du modèle mathématique) autour d'un jeu particulier de paramètres. Dans cette méthode, encadrant une courbe de concentrations aux exutoires, on peut produire deux courbes représentant l'incertitude maximale sur ces concentrations correspondant aux incertitudes sur les paramètres physiques de la simulation comme sur la Figure 1.2 à droite. Les paramètres physiques dont il est question sont principalement les perméabilités, les coefficients de retard et de diffusion dans les diverses couches et régions du domaine d'étude. Il s'agit donc d'une analyse de sensibilité multiparamètre [54, 74].

Le vecteur des paramètres étant dénoté par p, on considère l'application $p \longmapsto F(p)$ associant à p un vecteur rassemblant les indicateurs de sûreté $F(p)$. La sensibilité de cette application aux paramètres est mesurée en étudiant la matrice jacobienne $F'(p)$, également appelée matrice de sensibilité. La méthode d'analyse de sensibilité que l'on propose s'appuie sur la décomposition en valeurs singulières (SVD) [7, 28] de la matrice jacobienne $F'(p_0) = USV^T$ au point p_0 correspondant par exemple aux valeurs les plus vraisemblables des paramètres. La SVD est une généralisation de la diagonalisation aux matrices rectangulaires. En développant les incertitudes sur les paramètres δp suivant la base des vecteurs colonnes de V et les incertitudes correspondantes sur les indicateurs de sûreté $F(p_0 + \delta p) - F(p_0)$ suivant celle des vecteurs colonnes de U, on obtient donc une relation simple entre ces dernières et les premières.

On renvoie également à [75] pour l'histoire de la SVD et à [34] pour une description détaillée. La SVD peut aussi être utilisée en analyse statistique (voir [23, 26, 57]).

Différentes méthodes de différentiation peuvent être utilisées pour former la matrice de sensibilité : les différences divisées, la différentiation automatique, la différentiation analytique pour des problèmes éventuellement non linéaires. Un exemple utilisant la différentiation automatique pour un gros code `Fortran` est donné dans [6] et un exemple pour un code `C++` est donné dans [3]. Concernant la différentiation analytique, la théorie est expliquée par exemple dans [46] et un exemple est donné dans [63]. De plus on doit choisir entre construire cette matrice colonne par colonne (chaque colonne correspondant à l'image d'un vecteur de la base canonique de l'espace d'entrée par F'—on dit aussi qu'on différentie en mode direct) ou ligne par ligne (chaque ligne correspondant au gradient d'une

composante de F—on dit aussi qu'on différencie en mode inverse), par exemple en utilisant la méthode de l'état adjoint.

1.3.3 Identification de paramètres et analyse de sensibilité

L'analyse de sensibilité est un outil important pour le calcul d'incertitudes, mais aussi pour l'identification de quantités paramétrant un modèle mathématique. De plus, lors de la mise en œuvre, les calculs de dérivées sont partagés dans les deux cas, surtout si la matrice Jacobienne du modèle est calculée ligne par ligne par une technique adjointe.

On parle d'identification, ou d'estimation, de paramètres, ou encore de problème inverse, lorsque des coefficients, appelés paramètres, intervenants dans un modèle mathématique sont inaccessibles à la mesure et déterminés à partir de mesures des sorties de ce modèle. Le calcul d'incertitude, au contraire, se consacre à la propagation d'incertitudes jusqu'à des quantités qui ne sont pas mesurables directement. L'identification de paramètres et le calcul d'incertitudes ne se feront donc généralement pas sur le même modèle exactement.

Lorsque le modèle mathématique fait intervenir la résolution d'un système d'équations aux dérivées partielles représentée par l'opérateur de modélisation fin $\tilde{F}$, les dimensions des entrées et sorties de cet opérateur fin sont typiquement proportionnelles au nombre de cellules du maillage considéré, ce qui peut être énorme pour des raisons de précision du schéma numérique employé. Le modèle mathématique complet $F = \mathcal{O} \circ \tilde{F} \circ \mathcal{P}$ lui adjoint donc, d'une part, un opérateur de paramétrisation $\mathcal{P}$ qui aura un effet de régularisation sur le problème inverse, et d'autre part, un opérateur de mesure, ou d'observation, $\mathcal{O}$ qui se charge de représenter la réalité des mesures effectivement relevées sur le terrain.

Étant donné un modèle F et un seuil $\varepsilon > 0$, on définit l'indicateur de sensibilité $I(\varepsilon)$ du modèle F comme le nombre de valeurs singulières de F' dans un rapport au plus ε avec la plus grande : si les valeurs singulières sont notées $s_1, ..., s_{\min(n_{ip},n_{op})}$, ordonnées par ordre décroissant,

$$I(\varepsilon) = \max\left\{ i \in 1, \ldots, \min(n_{ip}, n_{op}) | \frac{s_i}{s_1} \geq \varepsilon \right\}.$$

Cet indicateur quantifie le nombre d'entrées du modèle F que l'on peut identifier de façon stable à partir de mesures sur les sorties dont l'incertitude est quantifiée par ε.

L'intérêt de l'analyse de sensibilité déterministe par SVD de la matrice Jacobienne du modèle pour l'identification de paramètres est donc multiple, voir [14, 19]. Le premier intérêt est évident : il est de donner une estimation quantitative du nombre de paramètres que l'on va pouvoir estimer de façon stable pour un dispositif donné, via l'indicateur de sensibilité défini ci-dessus. Mais il permet également de sélectionner parmi une collection d'opérateurs de paramétrisation et/ou d'observation ceux qui seront les plus efficaces pour l'inversion en choisissant ceux associés à l'indicateur de sensibilité maximum. De plus, dans les cas où l'assemblage, et la SVD, de la matrice Jacobienne du modèle fin $\tilde{F}$ est accessible (par exemple, pour des modèles mono-dimensionnels), il est possible d'évaluer le nombre maximum de paramètres que le modèle, et sa discrétisation, permettent d'estimer de façon stable, et donc d'évaluer de façon absolue la qualité d'un dispositif d'inversion

donné : un indicateur de sensibilité très proche de la valeur maximale possible est très satisfaisant ; a contrario, un petit indicateur de sensibilité maximum est très mauvais signe ! Enfin, toujours dans ce cas où l'analyse de sensibilité du modèle fin est accessible, les premiers vecteurs singuliers peuvent directement guider au choix optimum des opérateurs de paramétrisation $\mathcal{P}$ et d'observation $\mathcal{O}$. Notons que le calcul de SVD est local (basé sur un calcul de dérivées en un point), donc il faut au départ avoir une idée au moins grossière des paramètres d'entrée.

Revenons maintenant au problème d'analyse de sensibilité : on dispose d'un jeu de paramètres pour un modèle F à analyser, avec des incertitudes. Certains paramètres peuvent être issus d'un problème d'identification. L'analyse de sensibilité du problème F permet de repérer pour quels paramètres d'entrée on doit diminuer l'incertitude en priorité, afin de diminuer l'incertitude sur les sorties de F. Cette information peut être retournée aux experts d'inversion, afin qu'ils adaptent leur paramétrisation ou leurs techniques de mesure.

Intéressons-nous maintenant à l'inversion proprement dite de F. La solution la plus classique est l'utilisation d'une formulation moindres carrés :

$$\min_m J(m), \quad J(m) = \frac{1}{2} \|d - F(m)\|^2$$

On minimise J par exemple par une méthode de type descente, ce qui demande plusieurs calculs de dérivée de J. Le gradient de la fonction des moindres carrés s'exprime directement à l'aide de la matrice Jacobienne du modèle,

$$\nabla_m J(m) = F'(m)^T (F(m) - d).$$

Ce gradient peut se calculer sans assembler la matrice F'^T, par la méthode de l'état adjoint, qui peut aussi être utilisée en analyse de sensibilité pour le calcul ligne par ligne de la matrice Jacobienne, seuls les seconds membre de l'état adjoint changent.

1.3.4 Sensibilité à discrétisation

On s'est intéressé intéressé principalement dans cette thèse aux paramètres physiques. Les incertitudes sur les résultats sont aussi liées aux approximations numériques. On présentera dans le Chapitre 8 une étude préliminaire concernant la construction d'indicateurs de sensibilité aux paramètres de discrétisation en temps et en espace.

1.4 Aspects implémentation

1.4.1 Outils classiques de calcul scientifique

Pour le développement d'outils de calcul nous utilisons principalement le langage `C++`. On utilise la bibliothèque `LifeV` [42], qui est une bibliothèque `C++` proposant de nombreuses classes pour le calcul éléments finis—allant de la lecture de maillage à des méthodes d'assemblage de matrices éléments finis, en passant par la définition de règles de quadrature.

Une version « locale » intègre des éléments finis composites (voir la thèse [72]) et comporte des outils pour l'utiliser en même temps que la bibliothèque de différentiation automatique `ADOL-C` (voir [43]) (Chapitre 4), et qu'une bibliothèque de communications (Chapitre 6).

On a réutilisé en particulier la classe `DarcySolver` développée par Vincent Martin [59].

Pour le langage `C++` on renvoie à [39, 40] et aux ouvrages [51, 60].

1.4.2 Programmation fonctionnelle

Un autre objectif de la thèse est le développement d'outils génériques, ou plates-formes logicielles, pour l'analyse de sensibilité et pour le couplage de code. Les notions génériques de haut-niveau des applications sont implémentées de façon pleinement générique dans un langage de haut-niveau, et les parties spécifiques, généralement lourdes en calculs, sont fournies par l'utilisateur des plates-formes dans son langage favori. Dans le premier cas (analyse de sensibilité), l'assemblage de la matrice Jacobienne, le déclenchement du calcul de SVD, puis le post-traitement des éléments singuliers constitue la partie générique, alors que le calcul d'une rangée de la matrice est la partie spécifique. Dans le second cas (décomposition de domaine), la partie générique est l'algorithme de décomposition de domaines sans recouvrement avec préconditionnement Neumann-Neumann et équilibrage, alors que la partie spécifique est le solveur de sous-domaine.

Le typage fort des langages fonctionnels tels que Caml [35] et Haskell [41] assure la sécurité de la programmation et est un allié de poids pour la maintenance des codes de grande taille. Beaucoup d'erreurs sont détectées dès la compilation, et les messages d'erreur sont généralement très pertinents. Ils sont de plus très expressifs, et donc particulièrement adaptés à la programmation de la plupart des applications, dont les deux plates-formes évoquées ci-dessus. Cependant, les applications numériques directes, telles que la résolution d'un système d'équations aux dérivées partielles, font appel à de nombreuses bibliothèques d'éléments finis, d'algèbre linéaire ou autre qui existent déjà dans des langages impératifs tels que `Fortran`, `C` ou `C++`, et il n'est pas raisonnable d'envisager de tout réimplémenter à court terme dans un langage fonctionnel, bien que la sûreté de programmation et la généralité auraient grandement à y gagner.

Un point important est que ces langages fonctionnels demandent une façon différente de programmer et constituent une révolution de type copernicien pour la communauté des numériciens. C'est pourquoi on s'est efforcé de développer des plates-formes acceptant que les parties spécifiques des applications fournies par les utilisateurs soient écrites dans l'un des principaux langages impératifs (`Fortran`, `C` ou `C++`).

Caml : un langage fonctionnel fortement typé Le langage `Caml` est distribué par l'INRIA depuis 1985 (voir [35]). Il s'agit d'un langage de programmation de haut niveau, fondé sur une base théorique claire et sur une sémantique formelle. Il est fortement typé, c'est-à-dire que toutes les expressions sont typées, mais le typage est statique et assuré automatiquement par le compilateur. En particulier, il n'y a pas de conversion automatique de type. Ceci peut être déroutant au début car par exemple, on n'utilise pas le même opérateur pour additionner des nombres entiers ou des nombres flottants (+ pour les entiers,

+. pour les flottants) et on ne peut pas additionner sans précautions un entier et un nombre flottant. En contrepartie pour le calcul scientifique, on évite des erreurs qui dans d'autres langages peuvent être difficiles à repérer. Par exemple, que ce soit en `Fortran`, en `C` ou en `C++`, les expressions `2/5*3.14` et `3.14*2/5` n'ont pas du tout la même valeur, même en affectation à des variables de type flottant. En `Caml`, il n'y a jamais d'erreur de segmentation ou de « bus error », ni de fuites de mémoire, et un programme qui compile est souvent presque correct. Avec un langage impératif classique, un dépassement de tableau ne sera pas nécessairement détecté à l'exécution et il faut utiliser des outils spécifiques de débogage de mémoire [36] pour repérer les erreurs. `Caml` est un langage fonctionnel, qui permet une écriture très proche de celle utilisée couramment en mathématiques, et en particulier l'écriture de fonctions prenant en argument des fonctions est immédiate. Par exemple, la composition de deux fonctions s'exprime dans toute sa généralité en une seule ligne de code `Caml` :

```
let compose f g = function x -> g (f x);;
```

l'interpréteur interactif retourne le type de cette fonction

`compose` : $(\alpha \rightarrow \beta) \rightarrow (\beta \rightarrow \gamma) \rightarrow \alpha \rightarrow \gamma$

où α, β, γ sont des noms de type génériques. $\alpha_1 \rightarrow \alpha_2 \rightarrow ... \rightarrow \alpha_n \rightarrow \beta$ est le type d'une fonction à n arguments de types $\alpha_1, \ldots \alpha_n$ et de type de retour β. [1] Le compilateur `Caml` calcule les contraintes sur les arguments de `compose` : le type de retour du premier argument f (noté β) doit être le même que le type d'entrée du deuxième argument g. Le résultat de `compose` est une fonction. Noter qu'il revient exactement au même d'écrire

```
# let compose f g x = g (f x);;
```

L'interpréteur interactif retourne toujours le même type

`compose` : $(\alpha \rightarrow \beta) \rightarrow (\beta \rightarrow \gamma) \rightarrow \alpha \rightarrow \gamma$

et on comprend mieux avec cette commande la règle des parenthèses.

La notion de module, ou de collection de définitions de types, de valeurs et d'exceptions, proche de ce qu'est une bibliothèque pour les langages classiques, et surtout la notion de foncteur, ou de fonctionnelle des modules dans les modules, sont vouées à un brillant avenir dans les applications numériques.

On renvoie pour plus d'éléments sur le langage `Caml` à l'ouvrage [78] et à la thèse [59].

[1] Lorsqu'il y a plusieurs flèches, il y a des parenthèses implicites regroupant les termes à droite : le type $\alpha_1 \rightarrow \alpha_2 \rightarrow \alpha_3$ est équivalent au type $\alpha_1 \rightarrow (\alpha_2 \rightarrow \alpha_3)$, i.e. une fonction de deux variables est équivalente à une fonction d'une variable retournant une fonction d'une variable : il faut donner deux arguments pour obtenir un résultat numérique. Le type $(\alpha_1 \rightarrow \alpha_2) \rightarrow \alpha_3$ n'a en revanche rien à voir avec $\alpha_1 \rightarrow \alpha_2 \rightarrow \alpha_3$. Par exemple si on définit `let valeur_en_zero_de f = f 0 ; ;` alors `valeur_en_zero_de` est du type $(\text{int} \rightarrow \alpha) \rightarrow \alpha$.

1.5 Écoulements diphasiques

On s'est également intéressé aux écoulements non saturés. Lorsque la teneur en eau d'une milieu poreux est très faible, l'écoulement peut être modélisé par l'équation de Richards, qui suppose que la pression en air ne varie pas. Pour les écoulements diphasiques pétroliers, une formulation dite « pression globale » est couramment utilisée. On étudie les limites de ces approximations pour les écoulements diphasiques eau/air partiellement saturés.

1.6 Plan du mémoire

La première partie de ce mémoire est consacrée à l'analyse de sensibilité du problème du transfert de contaminants proprement dit. Nous présentons dans le Chapitre 2 le modèle à analyser. Dans le Chapitre 3 nous présentons les méthodes d'analyse de sensibilité et d'analyse d'incertitude de type Monte Carlo et déterministe, en mettant en évidence les identités possibles entre des indicateurs statistiques et déterministes. Nous nous penchons ensuite dans le Chapitre 4 sur différentes méthodes de calcul de dérivées, qui sont à la base des analyses de sensibilité déterministes. Le Chapitre 5 est consacré à quelques aspects de la décomposition de domaine sans recouvrement pour le problème d'écoulement. Nous présentons dans le Chapitre 6 différents aspects informatiques liés en particulier au couplage de codes. Nous présentons quelques applications de la méthode au problème de l'ANDRA dans le Chapitre 7.

Les parties suivantes présentent des ouvertures possibles de ce travail.

La deuxième partie envisage dans le Chapitre 8 la possibilité d'inclure des paramètres de discrétisation dans l'analyse de sensibilité déterministe.

Enfin, une troisième partie présente dans le Chapitre 9 quelques aspects du traitement des écoulements diphasiques eau/air, dont le modèle d'écoulement de la première partie de la thèse est un des cas limites. Nous comparerons en particulier le modèle diphasique complet avec le modèle de Richards, approximation couramment utilisée en hydrogéologie.

Première partie

Analyse de sensibilité pour un modèle d'écoulement et de transport en milieu poreux

Chapitre 2

Modèles mathématiques et numériques

Nous allons fournir dans ce chapitre une description détaillée des modèles mathématiques et numériques d'écoulement et de transport. La résolution des équations d'écoulement et de transport constitue le "noyau" des fonctions que nous analysons. Un modèle fin prend en entrée les paramètre d'une équation aux dérivées partielles ou d'évolution et retourne sa solution. On doit ensuite composer ce modèle fin en entrée avec une paramétrisation et en sortie avec un opérateur de mesure, pour réduire sa dimension.

2.1 Modèle d'écoulement

Ce modèle fournit la vitesse d'écoulement de l'eau, qui sera la vitesse de convection dans le modèle de transport. Le champ de pression est une inconnue intermédiaire du modèle. On utilise une formulation mixte, fournissant des résultats précis sur les flux en eau.

2.1.1 Équation continue

On s'intéresse d'abord à la modélisation de l'écoulement de l'eau autour d'un site de stockage.

Le domaine de calcul est un ouvert borné Ω de $\mathbb{R}^3$, contenant l'éventuel site de stockage de déchets nucléaires. On suppose que le domaine est saturé en eau (c'est le cas à une profondeur suffisante ; on s'intéresse à ce qui ce passe dans le cas non saturé dans le Chapitre 9 de la thèse) et que l'eau est incompressible. Pour les applications numériques, on supposera que les effets de la gravité peuvent être négligés, ce qui est vrai par exemple lorsque l'extension verticale du domaine de calcul est très faible devant son extension horizontale.

Alors l'écoulement dans le domaine Ω est gouverné par une équation de conservation stationnaire, couplée avec la loi de Darcy (voir [21]). On considère donc le système

d'équations

$$\left\{\begin{array}{rcll} \operatorname{div}\vec{u} & = & \tau & \text{dans } \Omega \quad \text{(loi de conservation)} \\ \vec{u} & = & -\mathcal{K}(\vec{\nabla}p - \rho\vec{g}) & \text{dans } \Omega \quad \text{(loi de Darcy)} \\ p & = & \overline{p} & \text{sur } \Gamma_D \\ \vec{u}\cdot\vec{n} & = & \overline{g} & \text{sur } \Gamma_N. \end{array}\right. \tag{2.1}$$

La frontière de Ω est $\partial\Omega = \overline{\Gamma}_D \cup \overline{\Gamma}_N$ avec $\Gamma_D \cap \Gamma_N = \emptyset$ et $\operatorname{mes}(\Gamma_D) \neq 0$. Les champs $\overline{p}$ et $\overline{g}$ définissent des conditions aux limites éventuellement non homogènes, τ est un terme source et $\rho\vec{g}$ est un terme de gravité, avec ρ la masse volumique de l'eau supposée constante uniforme et $\vec{g}$ le champ de pesanteur. Le produit $\rho\vec{g}$ est appelé le terme constant de l'équation (2.1). Le principal paramètre du problème est $\mathcal{K}$, le champ de conductivité hydraulique. Il est défini en chaque point de Ω par une matrice symétrique définie positive de taille 3×3. On cherche à évaluer le champ $\vec{u}$ de vitesse d'écoulement de l'eau. Le champ p de charge en eau est également inconnu. Le problème est ici mis naturellement sous forme mixte.

2.1.2 Formulation variationnelle

Cette formulation est présentée par exemple dans [10] et [67].

On rappelle d'abord rapidement la définition des espaces standard et quelques résultats classiques. L'ensemble des fonctions définies sur Ω mesurables et de carré intégrable est noté

$$L^2(\Omega) = \left\{ f : \Omega \to \mathbb{R} \,\middle|\, f \text{ mesurable avec } \int_\Omega f^2 < +\infty \right\}.$$

L'espace des fonctions de classe C^∞ à support compact est noté

$$\mathcal{D}(\Omega) = \left\{ f \in C^\infty(\Omega) \,\middle|\, \exists K \subset \Omega \text{ compact }, f_{|\mathbb{R}\setminus K} \equiv 0 \right\}$$

et son dual est

$$\mathcal{D}'(\Omega) = \{ l \ : \mathcal{D}(\Omega) \to \mathbb{R} |\ l \text{ linéaire continue} \}.$$

Les éléments de $\mathcal{D}'(\Omega)$ sont appelés distributions. On identifie $f \in L^2(\Omega)$ avec la distribution

$$g \in \mathcal{D}(\Omega) \mapsto \int_\Omega fg \in \mathbb{R}$$

qui se prolonge en une application linéaire continue sur $L^2(\Omega)$ par

$$g \in L^2(\Omega) \mapsto \int_\Omega fg \in \mathbb{R}.$$

$L^2(\Omega)$ est son propre dual, i.e. toute application linéaire continue sur $L^2(\Omega)$ peut être identifiée avec un élément de $L^2(\Omega)$. Pour $f, g \in L^2(\Omega)$ on notera

$$\langle f, g\rangle_{0,0} = \langle f, g\rangle = \int_\Omega fg.$$

Ceci définit un produit scalaire sur $L^2(\Omega)$, la norme associée est notée $\|.\|_0$.

On définit le gradient et la divergence au sens des distributions : pour $f \in \mathcal{D}'(\Omega)$ on a $\vec{\nabla} f \in (\mathcal{D}'(\Omega))^3$ tel que

$$\forall d \in \mathcal{D}(\Omega)^3, \quad (\vec{\nabla} f)(d) = -f(\text{div } d) \tag{2.2}$$

et pour $f \in \mathcal{D}'(\Omega)^3$ on a div $f \in \mathcal{D}'$ tel que

$$\forall d \in \mathcal{D}(\Omega), \quad (\text{div } f)(d) = -f(\vec{\nabla} d). \tag{2.3}$$

Si f est dérivable au sens classique alors son gradient et sa divergence au sens classique sont bien égaux à son gradient et sa divergence au sens des distributions.

$$H^1(\Omega) = \left\{ q \in L^2(\Omega) | \vec{\nabla} q \in \left(L^2(\Omega)\right)^3 \right\}.$$

C'est un espace de Hilbert, muni de la norme associée au produit scalaire

$$\langle f, g \rangle_{H^1(\Omega)} = \langle f, g \rangle_{0,0} + \sum_{i=1}^{3} \left\langle \frac{\partial f}{\partial x_i}, \frac{\partial g}{\partial x_i} \right\rangle_{0,0}.$$

$$H\left(\text{div}, \Omega\right) = \left\{ \vec{v} \in L^2(\Omega)^3 | \text{div } \vec{v} \in L^2(\Omega) \right\}.$$

C'est un espace de Hilbert, muni de la norme associée au produit scalaire

$$\langle f, g \rangle_{H(\text{div},\Omega)} = \langle f, g \rangle_{0,0} + \langle \text{div } f, \text{div } g \rangle_{0,0}.$$

L'espace $\mathcal{D}(\overline{\Omega})$ est l'ensemble des restrictions à Ω de fonctions de $\mathcal{D}(\mathbb{R}^3)$.

Trace. L'application

$$q \in \mathcal{D}(\overline{\Omega}) \mapsto q_{|\Gamma} \in L^2(\Gamma)$$

linéaire continue se prolonge par continuité en une application linéaire continue nommée trace :

$$\gamma \ : H^1(\Omega) \to L^2(\Gamma).$$

On définit $H^{\frac{1}{2}}(\Gamma) = \text{Im}(\gamma)$. Son dual est noté $H^{-\frac{1}{2}}(\Gamma)$. On a $L^2(\Gamma) \subset H^{-\frac{1}{2}}(\Gamma)$ si on identifie $\overline{q} \in L^2(\Gamma)$ avec $r \mapsto \int_\Gamma \overline{q} r$. Pour $f \in H^{-\frac{1}{2}}(\Gamma)$, $g \in H^{\frac{1}{2}}(\Gamma)$, on note

$$\langle f, g \rangle_{-\frac{1}{2},\frac{1}{2}} = f(g).$$

En particulier,

$$\text{si } f \in L^2(\Gamma) \text{ alors } \langle f, g \rangle_{-\frac{1}{2},\frac{1}{2}} = \int_\Gamma fg. \tag{2.4}$$

Trace normale. L'application

$$\vec{v} \in \mathcal{D}(\overline{\Omega})^3 \mapsto \vec{v}_{|\Gamma} \cdot \vec{n} \in L^2(\Gamma)$$

linéaire continue se prolonge par continuité en une application linéaire continue nommée trace normale :

$$\gamma_{\mathrm{n}} \ : H\left(\mathrm{div}, \Omega\right) \to H^{-\frac{1}{2}}(\Gamma).$$

Cette application est surjective.

Pour plus de précisions sur ces espaces, consulter par exemple [9] (sauf $H\left(\mathrm{div}, \Omega\right)$). On rappellera les autres résultats classiques lorsque l'on en aura besoin.

2.1.2.1 Formulation mixte duale

On s'intéresse dans cette section au cas où les **conditions de Neumann** sont **nulles** dans le sens suivant : on dira que $\vec{v} \in H\left(\mathrm{div}, \Omega\right)$ vérifie les conditions de Neumann si $\vec{v} \in W$ où W est défini par l'équation (2.6). On suppose de plus que les hypothèses suivantes sont vérifiées :

Hypothèses.

1. $\tau \in L^2(\Omega)$
2. $\overline{p} = \lambda_{|\Gamma_D}$ où $\lambda \in H^{\frac{1}{2}}(\Gamma)$.
3. $\mathcal{K}$ est à coefficients bornés et mesurables sur Ω et est uniformément elliptique, i.e. $\exists \alpha > 0$ indépendant de $x \in \Omega$ tel que, pour tout $\vec{\xi} \in \mathbb{R}^3$, $x \in \Omega$,

$$\mathcal{K}(x)\vec{\xi} \cdot \vec{\xi} \geq \alpha \vec{\xi} \cdot \vec{\xi}.$$

4. Ω est ouvert borné, à frontière C^1 par morceaux.
5. $\overset{\circ}{\Gamma}_D \neq \emptyset$.
6. On cherche $\vec{v} \in W$.

Nous allons donc sous ces hypothèses écrire une formulation variationnelle pour le problème (2.1). Pour cela nous devons définir, comme dans [10],

$$H^1_{0,D}(\Omega) = \left\{ q \in H^1(\Omega) | \gamma(q)_{|\Gamma_D} \equiv 0 \right\}, \tag{2.5}$$

puis

$$W = \left\{ \vec{v} \in H\left(\mathrm{div}, \Omega\right) \ | \ \forall q \in H^1_{0,D}(\Omega), \langle \gamma_{\mathrm{n}}(\vec{v}), \gamma(q) \rangle_{-\frac{1}{2}, \frac{1}{2}} = 0 \right\}. \tag{2.6}$$

Soit $(\vec{u}, p) \in W \times H^1(\Omega)$. Supposons que $(\vec{u}, p) \in H\left(\mathrm{div}, \Omega\right) \times H^1(\Omega)$ est solution de l'équation (2.1). Alors, comme comme $(\vec{u}, p)$ vérifie la loi de Darcy, on a

$$\forall \vec{v} \in W, \ \int_\Omega \mathcal{K}^{-1}\vec{u} \cdot \vec{v} + \int_\Omega (\vec{\nabla} p - \rho \vec{g}) \cdot \vec{v} = 0. \tag{2.7}$$

D'après la formule de Green on en déduit

$$\forall \vec{v} \in W, \ \int_\Omega \mathcal{K}^{-1}\vec{u}\cdot\vec{v} - \int_\Omega p\text{div } \vec{v} = -\langle \gamma_{\text{n}}(\vec{v}), \gamma(p)\rangle_{-\frac{1}{2},\frac{1}{2}} + \int_\Omega \rho\vec{g}\cdot\vec{v}. \tag{2.8}$$

Essayons d'exprimer le terme $\langle \gamma_{\text{n}}(\vec{v}), \gamma(p)\rangle_{-\frac{1}{2},\frac{1}{2}}$ en fonction de $\overline{p}$. Soit $p_1, p_2 \in H^1(\Omega)$ tels que $\gamma(p_1)_{|\Gamma_D} = \gamma(p_2)_{|\Gamma_D}$. Alors $(p_1 - p_2) \in H^1_{0,D}$, donc pour tout $\vec{v} \in W$ on a $\langle \gamma_{\text{n}}(\vec{v}), \gamma(p_1 - p_2)\rangle_{-\frac{1}{2},\frac{1}{2}} = 0$ i.e. $\langle \gamma_{\text{n}}(\vec{v}), \gamma(p_1)\rangle_{-\frac{1}{2},\frac{1}{2}} = \langle \gamma_{\text{n}}(\vec{v}), \gamma(p_2)\rangle_{-\frac{1}{2},\frac{1}{2}}$. On peut donc définir pour tout $\vec{v} \in W$ $\langle\langle \gamma_{\text{n}}(\vec{v}), \overline{p}\rangle\rangle$ par

$$\langle\langle \gamma_{\text{n}}(\vec{v}), \overline{p}\rangle\rangle = \langle \gamma_{\text{n}}(\vec{v}), \gamma(q)\rangle_{-\frac{1}{2},\frac{1}{2}} \text{ où } q \in H^1(\Omega) \text{ tel que } \gamma(q)_{|\Gamma_D} = \overline{p}, \tag{2.9}$$

indépendant du choix de $q \in H^1(\Omega)$ tel que $\gamma(q)_{|\Gamma_D} = \overline{p}$.

De plus, comme $\vec{u}$ vérifie la loi de conservation, on a

$$\forall q \in L^2(\Omega), \int_\Omega q\text{div } \vec{u} = \int_\Omega \tau q. \tag{2.10}$$

On introduit les formes bilinéaires classiques $a_{\mathcal{K}}$ définie sur $W \times W$ et b définie sur $W \times L^2(\Omega)$ par

$$a_{\mathcal{K}}(\vec{u}, \vec{v}) = \int_\Omega \mathcal{K}^{-1}\vec{u}\cdot\vec{v}, \tag{2.11}$$

$$b(\vec{v}, p) = -\int_\Omega p\text{div } \vec{v}, \tag{2.12}$$

et les formes linéaires l définie sur W et χ définie sur $L^2(\Omega)$ par

$$l(\vec{v}) = -\langle\langle \gamma_{\text{n}}(\vec{v}), \overline{p}\rangle\rangle + \int_\Omega \rho\vec{g}\cdot\vec{v} \tag{2.13}$$

$$\chi(q) = -\int_\Omega \tau q. \tag{2.14}$$

La formulation variationnelle de l'équation 2.1 s'écrit

$$\begin{array}{l} \text{Trouver } (\vec{u}, p) \in W \times L^2(\Omega) \text{ tels que} \\ \left\{ \begin{array}{lll} \forall \vec{v} \in W, & a_{\mathcal{K}}(\vec{u}, \vec{v}) + b(\vec{v}, p) = l(\vec{v}) & (a) \\ \forall q \in L^2(\Omega), & b(\vec{u}, q) = \chi(q) & (b) \end{array} \right. \end{array} \tag{2.15}$$

On remarque que, pour cette formulation, l'espace $\{(\vec{u}, p) \in W \times L^2(\Omega)\}$ dans lequel on cherche une solution pour la formulation variationnelle (2.15) n'est pas inclus dans l'espace $\{(\vec{u}, p) \in W \times H^1(\Omega)\}$ dans lequel on cherche une solution pour la formulation forte (2.1).

Toute solution de (2.1) est solution de (2.15). On vérifie que la formulation variationnelle (2.15) admet une unique solution $(\vec{u}, p)$. Les points à vérifier sont :

1. continuité et ellipticité de la forme bilinéaire $a_{\mathcal{K}}$ sur le noyau de b défini par

$$\ker(b) = \left\{\vec{v} \in W | \forall q \in L^2(\Omega), b(\vec{v}, q) = 0\right\} = \left\{b \in W | \text{div } \vec{v} \equiv 0\right\}. \tag{2.16}$$

2. continuité et condition inf-sup vérifiée pour la forme bilinéaire b :

$$\inf_{q\in L^2(\Omega)|\|q\|_{L^2(\Omega)}=1} \sup_{\vec{v}\in W|\|\vec{v}\|_{H(\mathrm{div},\Omega)}=1} b(\vec{v},q) > 0. \tag{2.17}$$

3. continuité de l et χ.

Notons que la symétrie de $\mathcal{K}$ n'est pas nécessaire pour que le problème soit bien posé, en revanche elle sera très pratique pour la résolution numérique.

Pour une démonstration de (1. et 2. et 3. $\Rightarrow$ existence et unicité de la solution de la formulation variationnelle), voir par exemple [10], section II.1.1.

On trouvera une démonstration des points 1. et 2. pour le cas d'un problème de Dirichlet pur (resp. de Neumann pur) par exemple dans [67], section 13 (resp. section 14).

Nous allons maintenant démontrer que cette unique solution de l'équation (2.15) est en fait dans $H(\mathrm{div},\Omega)\times H^1(\Omega)$ et qu'elle est solution forte de (2.1).

Soit $(\vec{u},p)\in H(\mathrm{div},\Omega)\times L^2(\Omega)$ l'unique solution de la formulation variationnelle (2.15). Alors

1. *Selon l'égalité (2.15)-(b), pour tout $q\in L^2(\Omega)$, $\langle \mathrm{div}\ \vec{u},q\rangle_{0,0}=\langle\tau,q\rangle_{0,0}$ donc $\mathrm{div}\ \vec{u}=\tau$.*
2. *Les conditions aux limites de Neumann sont vérifiées dans le sens $\vec{u}\in W$.*
3. *Soit $\vec{v}\in\mathcal{D}(\Omega)$. Alors en particulier, $\vec{v}\in W$ donc l'égalité (2.15)-(a) est vérifiée, et la trace normale $\gamma_{\mathrm{n}}(\vec{v})$ est nulle. On en déduit $\int_\Omega p\mathrm{div}\ \vec{v}=-\int_\Omega\left(\mathcal{K}^{-1}\vec{u}-\rho\vec{g}\right)\cdot\vec{v}$, donc, par définition de la dérivation (au sens des distributions), $\vec{\nabla}p=-\mathcal{K}^{-1}\vec{u}+\rho\vec{g}$, et en particulier $\vec{\nabla}p\in L^2(\Omega)^3$ donc $p\in H^1(\Omega)$.*
4. *Il reste à démontrer que les conditions aux limites de Dirichlet sont bien vérifiées. On utilise le résultat que l'on vient d'obtenir : dans l'égalité (2.15)-(a), on peut maintenant remplacer $\mathcal{K}^{-1}\vec{u}$ par $-\left(\vec{\nabla}p-\rho\vec{g}\right)$. On obtient*

$$\forall\vec{v}\in W,\quad -\int_\Omega\vec{\nabla}p\cdot\vec{v}+\int_\Omega\rho\vec{g}\cdot\vec{v}-\int_\Omega p\mathrm{div}\ \vec{v}=-\langle\langle\gamma_{\mathrm{n}}(\vec{v}),\overline{p}\rangle\rangle+\int_\Omega\rho\vec{g}\cdot\vec{v}$$

donc, d'après la formule de Green,

$$\forall\vec{v}\in W,\ \langle\gamma_{\mathrm{n}}(\vec{v}),\gamma(p)\rangle_{-\frac{1}{2},\frac{1}{2}}=\langle\langle\gamma_{\mathrm{n}}(\vec{v}),\overline{p}\rangle\rangle\,. \tag{2.18}$$

   (a) *Pour un problème de Dirichlet pur, on a $W=H(\mathrm{div},\Omega)$, donc $\mathrm{Im}(\gamma_{n|W})=\mathrm{Im}(\gamma_n)=H^{-\frac{1}{2}}(\Gamma)$, donc on en déduit $\gamma(p)=\overline{p}$.*

   (b) *Sinon, c'est à dire pour $\Gamma=\Gamma_N\cup\Gamma_D$ avec $\overset{\circ}{\Gamma}_N\neq\emptyset$ (et on rappelle que par hypothèse, $\overset{\circ}{\Gamma}_D\neq\emptyset$)*
   soit

$$p_2\in H^1(\Omega)\text{ tel que }\gamma(p_2)_{|\Gamma_D}=\overline{p}.$$

   On définit

$$\lambda=\gamma(p-p_2).$$

Montrons que $\lambda_{|\Gamma_D} \equiv 0$.
Par définition de $\langle\langle .,. \rangle\rangle$ *(équation (2.9)),*

$$\forall \vec{v} \in W,\ \langle \gamma_{\mathrm{n}}(\vec{v}), \gamma(p_2)\rangle_{-\frac{1}{2},\frac{1}{2}} = \langle\langle \gamma_{\mathrm{n}}(\vec{v}), \overline{p}\rangle\rangle\,. \tag{2.19}$$

On a, d'après les égalités (2.18) et (2.19),

$$\forall \vec{v} \in W,\ \langle \gamma_{\mathrm{n}}(\vec{v}), \lambda\rangle_{-\frac{1}{2},\frac{1}{2}} = \langle \gamma_{\mathrm{n}}(\vec{v}), \gamma(p)\rangle_{-\frac{1}{2},\frac{1}{2}} - \langle \gamma_{\mathrm{n}}(\vec{v}), \gamma(p_2)\rangle_{-\frac{1}{2},\frac{1}{2}} = 0. \tag{2.20}$$

Rappelons que $H^{\frac{1}{2}}(\Gamma) \subset L^2(\Gamma) \subset H^{-\frac{1}{2}}(\Gamma)$.
Nous allons raisonner par l'absurde :
supposons que $\lambda_{|\Gamma_D} \not\equiv 0$.
Alors il existe $\lambda_D \in L^2(\Gamma_D)$ *tel que* $\int_{\Gamma_D} \lambda_D \lambda \neq 0$. *On peut prolonger* λ_D *par 0 sur* Γ_N *pour former la fonction* $\lambda_1 \in L^2(\Gamma)$.
La trace normale est surjective de $H(\mathrm{div}, \Omega)$ *dans* $H^{-\frac{1}{2}}(\Gamma)$.
Soit donc $\vec{v}_1 \in H(\mathrm{div}, \Omega)$ *tel que* $\gamma_{\mathrm{n}}(\vec{v}_1) = \lambda_1$.
Alors $\vec{v}_1 \in W$ *avec* $\langle \gamma_{\mathrm{n}}(\vec{v}_1), \lambda\rangle_{-\frac{1}{2},\frac{1}{2}} = \int_\Gamma \lambda_D \lambda = \int_{\Gamma_D} \lambda_D \lambda \neq 0$ *(on se trouve dans le cas particulier (2.4)), ce qui est contradictoire avec l'équation (2.20).*
Donc $\lambda_{|\Gamma_D} \equiv 0$, *i.e.* $\gamma(p - p_2)_{|\Gamma_D} \equiv 0$, *i.e.* $\gamma(p)_{|\Gamma_D} \equiv \gamma(p_2)_{|\Gamma_D}$, *i.e.* $\gamma(p)_{|\Gamma_D} = \overline{p}$.
Finalement, $(\vec{u}, p) \in H(\mathrm{div}, \Omega) \times H^1(\Omega)$ *et* $(\vec{u}, p)$ *est solution forte de l'équation (2.1).*

Solutions approchées. Soit une famille d'espaces d'approximation $W_h, M_h,\ h > 0$, $W_h \subset W$, munis de la norme de $H(\mathrm{div}, \Omega)$, de dimension finie, et $M_h \subset L^2(\Omega)$, muni de la norme de $L^2(\Omega)$, de dimension finie, et compatibles au sens de la condition inf-sup discrète :

$$\forall h,\ \beta_h = \inf_{q_h \in M_h | \|q_h\|_{L^2(\Omega)}=1} \ \sup_{\vec{v}_h \in W | \|\vec{v}_h\|_{H(\mathrm{div},\Omega)}=1} b(\vec{v}_h, q_h) > 0. \tag{2.21}$$

On définit les problèmes variationnels approchés :

Trouver $(\vec{u}_h, p_h) \in W_h \times M_h$ tels que

$$\begin{cases} \forall \vec{v}_h \in W_h, \quad a_{\mathcal{K}}(\vec{u}_h, \vec{v}_h) + b(\vec{v}_h, p_h) = -\langle \gamma_{\mathrm{n}}(\vec{v}_h), \overline{p}\rangle_{-\frac{1}{2},\frac{1}{2}} + \displaystyle\int_\Omega \vec{v}_h \cdot \rho \vec{g} & (a) \\ \forall q_h \in M_h, \quad b(\vec{u}_h, q_h) = -\displaystyle\int_\Omega \tau q_h. & (b) \end{cases} \tag{2.22}$$

Alors le problème (2.22) admet une unique solution. Si de plus, la condition inf-sup discrète est vérifiée uniformément, c'est-à-dire si $\inf_h \beta_h > 0$, et si

$$\ker\left(b_{|W_h \times M_h}\right) \subset \ker(b), \tag{2.23}$$

c'est-à-dire si

$$\left(\vec{v}_h \in W_h \text{ et } \forall q_h \in M_h,\ \int_\Omega q_h \mathrm{div}\ \vec{v}_h = 0\right) \Rightarrow \mathrm{div}\ \vec{v}_h \equiv 0, \tag{2.24}$$

alors il existe $C > 0$, indépendant de h, tel que

$$\|\vec{v} - \vec{v}_h\|_{H(\mathrm{div},\Omega)} + \|p - p_h\|_{L^2(\Omega)} \leq C \left(\inf_{\vec{v}_h \in W_h} \|\vec{v} - \vec{v}_h\|_{H(\mathrm{div},\Omega)} + \inf_{q_h \in M_h} \|p - p_h\|_{L^2(\Omega)} \right). \tag{2.25}$$

Un condition suffisante pour vérifier (2.24) est

$$\mathrm{div}\ W_h \subset M_h. \tag{2.26}$$

Dans [67] on explique comment construire une famille d'espaces d'approximation vérifiant les propriétés (2.21) et (2.24) et on donne des majorations du second membre de l'égalité (2.25). Dans ce travail, nous avons utilisé des éléments de Raviart-Thomas-Nedelec de plus bas degré et des éléments composites.

Cas des conditions de Neumann non nulles. On considère maintenant que $\overline{g} \in L^2(\Gamma_N)$ avec $\overline{g} \neq 0$.

Pour écrire une formulation variationnelle lorsque les conditions de Neumann sont non nulles, on cherche à se ramener au cas précédent (Neumann homogène).

On suppose que $(\vec{u}^*, p^*) \in H(\mathrm{div}, \Omega) \times H^1(\Omega)$ tels que $\vec{u}^* \cdot \vec{n} = \overline{g}$ sur Γ_N et $\vec{u}^* = -\mathcal{K}\vec{\nabla} p^*$ sur Ω (i.e. $(\vec{u}^*, p^*)$ est une solution particulière d'un problème constitué de la loi de Darcy, posé sans conditions de Dirichlet et sans loi de conservation). Il suffit alors d'écrire le problème vérifié par la différence $(\vec{u}^0, p^0) = (\vec{u} - \vec{u}^*, p - p^*)$. On cherche finalement $(\vec{u}^0, p^0) \in H(\mathrm{div}, \Omega) \times H^1(\Omega)$ tels que

$$\left\{ \begin{array}{rcll} \mathrm{div}\ \vec{u}^0 & = & \tau - \mathrm{div}\ \vec{u}^* & \text{dans } \Omega \\ \vec{u}^0 & = & -\mathcal{K}\vec{\nabla} p^0 & \text{dans } \Omega \\ p^0 & = & -p^* + \overline{p} & \text{sur } \Gamma_D \\ \vec{u}^0 \cdot \vec{n} & = & 0 & \text{sur } \Gamma_N. \end{array} \right. \tag{2.27}$$

où "$\vec{u}^0 \cdot \vec{n} = 0$ sur Γ_N" est un abus de notation pour "$\vec{u}^0 \in W$". En fait la condition de Neumann

$$\vec{u} \cdot \vec{n} = \overline{g} \text{ sur } \Gamma_N$$

est à remplacer par

$$\forall q \in H^1_{0,D}(\Omega),\ \langle \gamma_{\mathrm{n}}(\vec{u}), \gamma(q) \rangle_{-\frac{1}{2},\frac{1}{2}} = \langle\langle \overline{g}, \gamma(q) \rangle\rangle\,,$$

où $\langle\langle \overline{g}, \gamma(q) \rangle\rangle = \langle \overline{g}_1, \gamma(q) \rangle_{-\frac{1}{2},\frac{1}{2}}$ pour $\overline{g}_1$ un prolongement L^2 quelconque de $\overline{g}$ sur Γ (on vérifie que la définition est indépendante du prolongement choisi).

Vérifions qu'une solution particulière $(\vec{u}^*, p^*)$ existe. On considère par exemple le problème : trouver $(\vec{u}^*, p^*) \in H(\mathrm{div}, \Omega) \times H^1(\Omega)$ tels que

$$\left\{ \begin{array}{rcll} \mathrm{div}\ \vec{u}^* & = & 0 & \text{dans } \Omega \quad \text{(loi de conservation)} \\ \vec{u}^* & = & -\mathcal{K}\vec{\nabla} p^* & \text{dans } \Omega \quad \text{(loi de Darcy)} \\ \vec{u}^* \cdot \vec{n} & = & \overline{g}_1 & \text{sur } \Gamma \end{array} \right. \tag{2.28}$$

où $\overline{g}_1 \in L^2(\Gamma)$ tel que $\overline{g}_{1|\Gamma_N} = \overline{g}$. Ce problème admet une unique solution $(\vec{u}^*, p^*)$ dans $H(\mathrm{div}, \Omega) \times H^1(\Omega)/\mathbb{R}$ à condition que $\overline{g}_1$ vérifie la condition de compatibilité

$$\int_\Gamma \overline{g}_1 = 0.$$

Dans la pratique, W_h sera toujours tel que $\forall \vec{v} \in W_h, \gamma_n(\vec{v}) \in L^2(\Gamma)$. Pour construire les solutions approchées, il est alors possible de ne pas passer par une solution particulière. Une première possibilité est de ne pas inclure de condition sur Γ_N dans la définition de W_h, et d'écrire des équations supplémentaires pour imposer les conditions aux limites de Neumann. On présente cette solution pour une discrétisation EFMH dans la section 2.1.3.3. Une autre possibilité est d'écrire une équation discrète vérifiée pour un problème de Neumann nul, et de corriger ensuite le second membre pour prendre en compte les conditions aux limites. On présente cette solution pour une discrétisation EFMH dans la section 2.1.3.4, et pour une formulation mixte dans la section 2.2.3.2.

2.1.2.2 Formulation mixte-hybride

On utilise une formulation mixte hybride plutôt qu'une formulation mixte classique pour avoir à résoudre une équation impliquant une matrice symétrique définie positive. Cette idée est présentée par exemple dans l'article [2].

Cette formulation dépend de la définition d'une triangulation, et est totalement équivalente à la formulation mixte. On ajoute des degrés de liberté (multiplicateurs de Lagrange) pour rendre l'équation à résoudre symétrique définie positive.

On considère une triangulation $\mathcal{T}_h$, pour l'instant quelconque, de Ω : $\Omega = \cup_{T\in\mathcal{T}_h}\overline{T}$ et pour $T_1, T_2 \in \mathcal{T}_h$, $\overset{\circ}{T_1} \cap \overset{\circ}{T_2} = \emptyset$. On note $\partial\mathcal{T}_h = \Pi_{T\in\mathcal{T}_h}\partial T$ et $L(\partial\mathcal{T}_h) = \Pi_{T\in\mathcal{T}_h}L^2(\partial T)$. Si $\lambda = (\lambda_T)_{T\in\mathcal{T}_h} \in L(\partial\mathcal{T}_h)$ est tel que

$$\text{pour tout } T_1, T_2 \in \mathcal{T}_h \text{ tels que } \overline{T_1}\cap\overline{T_2} \neq \emptyset, \text{ pour tout } r \in L^2(\overline{T_1}\cap\overline{T_2}), \int_{\overline{T_1}\cap\overline{T_2}} r\lambda_{T_1} = \int_{\overline{T_1}\cap\overline{T_2}} r\lambda_{T_2} \tag{2.29}$$

et

$$\text{pour tout } T \in \mathcal{T}_h \text{ tels que } \overline{T}\cap\Gamma_D \neq \emptyset, \text{ pour tout } r \in L^2(\overline{T_1}\cap\overline{T_2}), \int_{\overline{T}\cap\Gamma_D} r\lambda_T = \int_{\overline{T}\cap\Gamma_D} r\overline{p} \tag{2.30}$$

alors la solution $(\vec{u}_h, p_h)$ de l'équation (2.22) vérifie également

$$\forall \vec{v} \in W,\ a_{\mathcal{K}}(\vec{u}, \vec{v}) + b(\vec{v}, p) + \sum_{T\in\mathcal{T}_h} \int_{\partial T} \lambda_T \gamma_{\mathrm{n}T}(\vec{v}) = \int_\Omega \vec{v}\cdot\rho\vec{g}. \tag{2.31}$$

Dans les formulations mixtes hybrides, l'équation (2.22)-(a) mixte est remplacée par une équation de la forme de (2.31) et des conditions supplémentaires assurent l'unicité de $(\vec{u}_h, p_h, \lambda_h)$.

En écrivant des égalités vérifiées cellule par cellule, on verra que λ peut être interprété comme une valeur de solution approchée p_h sur $\partial\mathcal{T}_h$. Rigoureusement, ceci n'a pas de sens car on cherche une solution approchée $L^2(\Omega)$, donc sa trace sur $\partial\mathcal{T}_h$ n'est pas définie.

Dans la suite, on construira une formulation éléments finis mixte hybride (EFMH) en sommant des égalités vérifiées sur des cellules $T \in \mathcal{T}_h$.

2.1.3 Formulation discrète EFMH

2.1.3.1 Triangulation

On considère une discrétisation éléments finis $\mathcal{T}_h = (T_i)_{i\in I}$ du domaine Ω à N_c éléments : l'ensemble des indices des cellules du maillage est $I = \{1, 2, \ldots, N_c\}$, T_i pour $i \in I$ sont des ouverts de Ω tels que $\Omega = \cup_{i\in I} T_i$ et $T_i \cap T_j = \emptyset$ pour $i \neq j$. La triangulation $\mathcal{T}_h$ du domaine Ω peut être constituée de tétraèdres, d'hexaèdres convexes ou d'un mélange des deux.

L'ensemble des faces de la triangulation est noté $\mathcal{S}_h = (E_j)_{j\in J}$, $J = \{1, \ldots, N\}$. Chaque face E_j est orientée par le choix d'un vecteur normal unitaire $\vec{\mathrm{n}}_j$. Les faces frontières sont orientées vers l'extérieur du domaine de calcul Ω et les faces intérieures sont orientées arbitrairement. On suppose que chaque face de la frontière est soit incluse dans Γ_N, soit incluse dans Γ_D. On décompose alors $J = \overset{\circ}{J} \cup J_N \cup J_D$, où $\overset{\circ}{J}$ est l'ensemble des indices des faces intérieures, J_N est l'ensemble des indices des faces incluses dans Γ_N et J_D est l'ensemble des indices des faces incluses dans Γ_D. On note $\#\overset{\circ}{J} = \overset{\circ}{N}$, $\#J_N = N_N$ et $\#J_D = N_D$ où $\#$ désigne la cardinalité.

La cellule adjacente à une face E_j, $j \in J_N \cup J_D$ de la frontière du domaine de calcul est $T_{i_1(j)}$. Les cellules adjacentes à une face E_j, $j \in \overset{\circ}{N}$ de l'intérieur du domaine de calcul sont $T_{i_1(j)}$ et $T_{i_2(j)}$, de sorte que $\vec{\mathrm{n}}_j$ soit dirigé de $T_{i_1(j)}$ vers $T_{i_2(j)}$ (voir la Figure 2.1). Dans la pratique, le sens de $\vec{\mathrm{n}}_j$ est défini par le choix de $i_1(j)$ et $i_2(j)$.

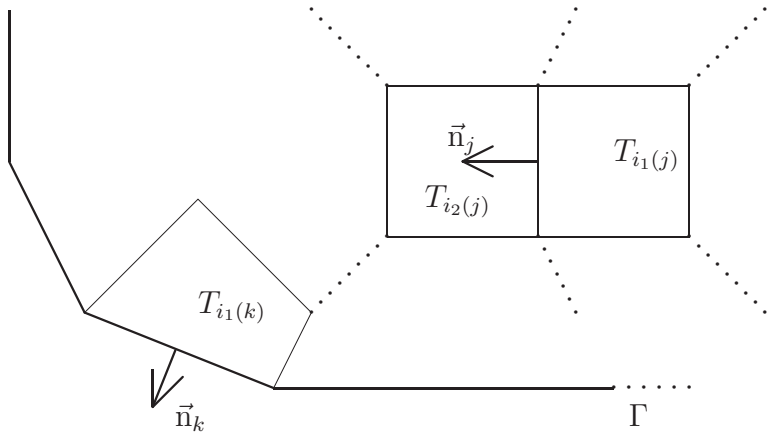


FIG. 2.1 – Orientation des faces de la triangulation.

Les propriétés suivantes aident à la construction d'espaces d'approximation (voir par exemple [67]) :

$$\left\{\begin{array}{l} q \in L^2(\Omega) \\ \forall T \in \mathcal{T}_h, \quad q_{|T} \in H^1(T) \\ \forall T_1, T_2 \in \mathcal{T}_h, \quad \gamma\left(q_{|T_1}\right)_{|T_1 \cap T_2} = \gamma\left(q_{|T_2}\right)_{|T_1 \cap T_2} \end{array}\right. \Rightarrow q \in H^1(\Omega).$$

On définit

$$\mathcal{H}(\mathrm{div}, \Omega) = \left\{\vec{v} \in H\left(\mathrm{div}, \Omega\right) | \gamma_{\mathrm{n}}(\vec{v}) \in L^2(\partial\Omega)\right\}.$$

Alors

$$\left\{\begin{array}{l} \vec{v} \in L^2(\Omega)^3 \\ \forall T \in \mathcal{T}_h, \quad \vec{v}_{|T} \in \mathcal{H}(\mathrm{div}, \Omega) \\ \forall T_1, T_2 \in \mathcal{T}_h, \quad \gamma_{\mathrm{n}}\left(\vec{v}_{|T_1}\right)_{|T_1 \cap T_2} + \gamma_{\mathrm{n}}\left(\vec{v}_{|T_2}\right)_{|T_1 \cap T_2} = 0 \end{array}\right. \Rightarrow \vec{v} \in \mathcal{H}(\mathrm{div}, \Omega).$$

2.1.3.2 Inventaire des degrés de liberté et fonctions de base

On utilise des éléments de Raviart-Thomas-Nedelec de plus bas degré dans le cas de cellules tétraédriques ou parallélépipédiques [66, 62, 16], et les éléments composites définis dans [72], inspirés des éléments de Kusnetzov et Repin [52] pour des maillages hexahédriques généraux. En effet, pour des maillages hexahédriques non parallélépipédiques, l'espace d'approximation W_h pour les éléments de Raviart-Thomas-Nedelec de plus bas degré ne contient pas les fonctions constantes, et par conséquent le terme

$$\inf_{\vec{v}_h \in W_h} \|\vec{v} - \vec{v}_h\|_{H(\mathrm{div},\Omega)}$$

dans le second membre de l'équation (2.25) n'est pas un petit $\circ$ de h. Pour ces formulations, les degrés de liberté sont

- Les pressions : pour chaque cellule T_i, une valeur correspondant à une approximation de la moyenne de la charge en eau sur la cellule. Ces degrés de liberté sont stockés dans un vecteur P de taille N_c.
- Les traces de pression : pour chaque face E_j de $\overset{\circ}{J} \cup J_N \cup J_D$, une valeur correspondant à une approximation de la moyenne de la charge en eau sur la face. Ces degrés de liberté sont stockés dans un vecteur L de taille N.
- Les flux : pour chaque cellule T_i et pour chaque face E_j de T_i, une valeur correspondant à une approximation du flux sortant de T_i à travers E_j. Nous avons ainsi une inconnue par face frontière et deux inconnues par face intérieure. Contrairement à ce qui se passe pour la formulation mixte classique, la continuité du flux n'est pas imposée par le choix des inconnues, mais par des équations supplémentaires, obtenues en utilisant des multiplicateurs de Lagrange associés aux traces de pression correspondant à des faces intérieures. De même, les conditions aux limites de Neumann sont imposées par l'intermédiaire des multiplicateurs de Lagrange associés aux faces de la frontière de Neumann. Ces degrés de liberté sont stockés dans un vecteur U de taille $2\,\overset{\circ}{N} + N_N + N_D$.

On décompose le champ de vitesse d'écoulement approché suivant une base de $N_N + N_D + 2\ \overset{\circ}{N}$ éléments, notés $(\vec{u}_j)_{j=1,\ldots,N_N+N_D+2\overset{\circ}{N}}$. On rappelle que $J = \left\{1, \ldots, N_N + N_D + \overset{\circ}{N}\right\}$ et on note $J' = \left\{N_N + N_D + \overset{\circ}{N} + 1, \ldots, N_N + N_D + 2\ \overset{\circ}{N}\right\}$. Les fonctions de cette base vérifient les propriétés suivantes :

– Pour $j \in J$, $\vec{u}_j$ est nulle hors de $\overline{T_{i_1(j)}}$ et est continue à divergence constante sur $\overline{T_{i_1(j)}}$ et on a

$$\left\{\begin{array}{l} \displaystyle\int_{T_{i_1(j)}} \text{div } \vec{u}_j = \int_{E_j} \vec{u}_j \cdot \vec{\text{n}}_j = 1, \\ \displaystyle\int_{E_k} \vec{u}_j \cdot \vec{\text{n}}_k = 0 \text{ si } k \neq j, \\ \displaystyle\int_{T_i} \text{div } \vec{u}_j = 0 \text{ si } i \neq i_1(j). \end{array}\right. \tag{2.32}$$

– à chaque $j \in \overset{\circ}{J}$, on associe un unique $j' \in J'$. Alors $\vec{u}_{j'}$ est nulle hors de $\overline{T_{i_2(j)}}$ et est continue à divergence constante sur $\overline{T_{i_2(j)}}$ et on a

$$\left\{\begin{array}{l} \displaystyle\int_{T_{i_2(j)}} \text{div } \vec{u}_{j'} = \int_{E_j} \vec{u}_{j'} \cdot (-\vec{\text{n}}_j) = 1, \\ \displaystyle\int_{E_j} \vec{u}_{j'} \cdot (-\vec{\text{n}}_k) = 0 \text{ si } k \neq j, \\ \displaystyle\int_{T_i} \text{div } \vec{u}_{j'} = 0 \text{ si } i \neq i_2(j). \end{array}\right. \tag{2.33}$$

L'expression exacte de ces fonctions de base dépend des éléments choisis. Avec les éléments finis RTN de plus bas degré, les restrictions des fonctions de base à une cellule T_i sont polynomiales. Avec des éléments finis composites, ces restrictions sont polynomiales par morceaux et continues à divergence constante. Les fonctions de base sont donc à divergence constante par cellule. Elles ne sont ni continues, ni à composante normale continue.

L'espace d'approximation $W_h \subset H\,(\text{div}, \Omega)$ pour le champ de vitesse est

$$W_h = \text{Vec}\left(\{\vec{u}_j | j \in J \cup J'\}\right) \cap H\,(\text{div}, \Omega) = \left\{\sum_{j \in J \cap J'} V_j \vec{u}_j | \forall j \in \overset{\circ}{J}, V_j + V_{j'} = 0\right\}. \tag{2.34}$$

Les fonctions de base n'appartiennent pas à W_h. Par conséquent, une équation supplémentaire imposera la condition div $\vec{u}_h \in L^2(\Omega)$.

La charge en eau est décomposée sur une base canonique de fonctions constantes par morceaux et $M_h \subset L^2(\Omega)$ est l'ensemble des fonctions constantes par cellule.

Il n'y a pas de fonctions de base associées aux traces de pression. Les traces de pression sont en fait les multiplicateurs de Lagrange associés à la contrainte div $\vec{u}_h \in L^2(\Omega)$.

Le nombre total de degrés de liberté est

$$n_{dof} = N_N + N_D + 2\ \overset{\circ}{N} + N_c + N. \tag{2.35}$$

2.1.3.3 Équations du problème discret

Soit $i \in I$, $j \in J \cup J'$. Alors la solution forte $\vec{u}, p$ de la formulation (2.1) vérifie

$$\int_{T_i} \mathcal{K}^{-1}\vec{u}\cdot\vec{u}_j + \int_{T_i} \vec{\nabla}p\cdot\vec{u}_j = \int_{T_i} \mathcal{K}^{-1}\vec{u}\cdot\vec{u}_j + \int_{\partial T_i} p\vec{u}_j\cdot\vec{n}^i - \int_{T_i} p\mathrm{div}\ \vec{u}_j = \int_{T_i} \rho\vec{g}\cdot\vec{u}_j \quad (2.36)$$

ce qui suggère de chercher $\vec{u}_h, p_h$ et L tels que pour tout $i \in I$, $j \in J \cup J'$,

$$\int_{T_i} \mathcal{K}^{-1}\vec{u}_h\cdot\vec{u}_j + \sum_{k\in J|E_k\subset\overline{T}_i} \int_{E_k} L_k\vec{u}_j\cdot\vec{n}^i - \int_{T_i} p_h\mathrm{div}\ \vec{u}_j = \int_{T_i} \rho\vec{g}\cdot\vec{u}_j. \quad (2.37)$$

Ainsi, $\vec{u}_h, p_h$ vérifiera l'équation (2.22)-(a) à condition que

$$L_j = \int_{E_j} \overline{p}\vec{u}_j\cdot\vec{\mathrm{n}}_j \quad (2.38)$$

pour $E_j \subset \Gamma_D$. Alors, d'après les propriétés des fonctions de base (2.32) et (2.33), et comme $\vec{u}_j$ est nulle hors de $T_{i_1(j)}$ et $\vec{u}_{j'}$ est nulle hors de $T_{i_2(j)}$ la loi de comportement (loi de Darcy) s'écrit sous forme discrète

$$\left\{\begin{array}{l} \forall j \in J \text{ (équation (2.37) pour } i = i_1(j)) \\ \quad \displaystyle\sum_{k\in J\cup J'} a_{\mathcal{K}}(\vec{u}_j,\vec{u}_k)U_k + L_j - P_{i_1(j)} = a_{\mathbf{I}}(\rho\vec{g},\vec{u}_j). \\ \forall j \in \overset{\circ}{J}, \text{ i.e. } j' \in J' \text{ existe }, \quad \text{(équation (2.37) pour } i = i_2(j)) \\ \quad \displaystyle\sum_{k\in J\cup J'} a_{\mathcal{K}}(\vec{u}_{j'},\vec{u}_k)U_k + L_j - P_{i_2(j)} = a_{\mathbf{I}}(\rho\vec{g},\vec{u}_{j'}). \end{array}\right. \quad (2.39)$$

où $a_{\mathcal{K}}$ est défini dans l'équation (2.11). Les intégrales peuvent être calculées comme combinaison linéaire des valeurs des fonctions à intégrer en certains points bien choisis (les points de quadrature - voir par exemple [10]). Le nombre de points nécessaires pour que les intégrales soit exactes dépend du degré des polynômes à intégrer.

Pour $i \in I$, soit $J_{MH}(i)$ l'ensemble des indices des fonctions de base pour le champ de vitesse d'écoulement non nulles sur T_i, c'est-à-dire l'ensemble des éléments j de J tels que $i = i_1(j)$ et des éléments j' de J' tels que $i = i_2(j)$. Les $J_{MH}(i)$ forment une partition de $J \cup J'$. Pour $j, k \in J \cup J'$, si $a_{\mathcal{K}}(\vec{u}_j, \vec{u}_k)$ est non nul alors il existe $i \in I$ tel que $j \in J_{MH}(i)$ et $k \in J_{MH}(i)$.

L'équation de conservation (2.22)-(b) s'écrit

$$\forall i \in I, \quad \sum_{j\in J_{MH}(i)} U_j = \mathrm{mes}(T_i)\tau_i. \quad (2.40)$$

où τ_i est la valeur moyenne du flux volumique sur la cellule T_i : $\displaystyle\int_{T_i} \tau = \mathrm{mes}(T_i)\tau_i$. La continuité du flux (contrainte W_h (2.34)) s'écrit

$$\forall j \in \overset{\circ}{J}, \quad U_j + U_{j'} = 0. \quad (2.41)$$

Les conditions aux limites de Neumann imposent U_j pour $j \in J_N$:

$$U_j = \int_{E_j} \overline{g}\vec{u}_j \cdot \vec{\mathrm{n}}_j \tag{2.42}$$

pour $j \in J_N$. Les conditions aux limites de Dirichlet imposent L_j pour $j \in J_D$ selon l'équation (2.38).

On doit finalement résoudre une équation de la forme

$$\begin{pmatrix} A_{\mathcal{K}} & B^T & C^T \\ B & \mathbf{0} & \mathbf{0} \\ C & \mathbf{0} & D \end{pmatrix} \begin{pmatrix} U \\ P \\ L \end{pmatrix} = \begin{pmatrix} R_1 \\ R_2 \\ R_3 \end{pmatrix}. \tag{2.43}$$

La première équation $A_{\mathcal{K}}U + B^T P + C^T L = R_1$ correspond à la loi de Darcy. La deuxième équation $BU = R_2$ correspond à la loi de conservation. La troisième équation $CU + DL = R_3$ exprime selon les lignes la continuité du flux ou les conditions aux limites de Dirichlet.

La matrice de masse $A_{\mathcal{K}}$ est carrée de taille $(N+\overset{\circ}{N}) \times (N+\overset{\circ}{N})$. À une renumérotation près, consistant à regrouper les composantes de U selon la partition $(J_{MH}(i))_{i\in I}$, la matrice $A_{\mathcal{K}}$ est diagonale par blocs. Chaque bloc de la diagonale $A_{\mathcal{K},i}$, $i \in I$, est la matrice de masse "élémentaire" intervenant dans la résolution d'un problème aux limites local posé sur la cellule T_i. Cette matrice est indépendante de la nature des conditions aux limites. Les valeurs de $A_{\mathcal{K},i}$ dépendent des fonctions de base choisies.

La matrice B est de taille $N_c \times (N+\overset{\circ}{N})$ et ne dépend pas non plus des conditions aux limites. On a tout simplement

$$\begin{cases} B_{i,j} = -1_{j\in J_{MH}(i)} \\ R_{2_i} = -\mathrm{mes}(T_i)\tau_i. \end{cases} \tag{2.44}$$

Si on permute les colonnes de B suivant la partition de $J \cup J'$, pour garder la cohérence avec les permutations effectuées sur la matrice $A_{\mathcal{K}}$, B devient diagonale par blocs ($N_c \times N_c$ blocs non carrés). Le $i^{ème}$ bloc de la diagonale est noté B_i. C'est une matrice ligne contenant uniquement la valeur -1, de longueur le nombre de faces de la cellule T_i.

L'organisation des matrices C et D dépend de la nature des conditions aux limites. La matrice C est de taille $N \times (N+\overset{\circ}{N})$. La matrice D est diagonale de taille $N \times N$. R_1, R_2 et R_3 sont des vecteurs de longueur N dépendant des conditions aux limites, du terme source et de la gravité.

Nous allons maintenant expliquer l'utilisation de ces matrices A et B dans chaque ligne du système (2.43), et expliciter ainsi les coefficients de C et D et le second membre de (2.43). Les lignes des équations de (2.43)

$$A_{\mathcal{K}}U + B^T P + C^T L = R_1 \tag{2.45}$$

$$CU + DL = R_3 \tag{2.46}$$

sont associées à des éléments de J (2.46) ou de $J \cup J'$ (2.45) et nous allons expliquer les $j^{ème}$ lignes de (2.45) et (2.46) dans les cas 1. $j \in J_D$; 2. $j \in J_N$; et 3. $j \in \overset{\circ}{J}$.

1. Pour $j \in J_D$, la $j^{ème}$ ligne de C est nulle et le $j^{ème}$ terme diagonal de D est égal à 1. Le terme L_j de l'équation $\sum_{k \in J \cup J'} a_{\mathcal{K}}(\vec{u}_j, \vec{u}_k) U_k + L_j - P_{i_1(j)} = a_{\mathbf{I}}(\rho \vec{g}, \vec{u}_j)$ (voir l'équation (2.39)), équation correspondant à la $j^{ème}$ ligne de $A_{\mathcal{K}} U + B^T P + C^T L = R_1$, se retrouve dans le second membre : $R_{1_j} = a_{\mathbf{I}}(\rho \vec{g}, \vec{u}_j) - L_{D_j}$, où L_{D_j} est proche de la valeur moyenne de la condition aux limites de Dirichlet sur E_j puisque $L_{D_j} = \int_{D_j} \overline{p} \vec{u}_j \cdot \vec{n}_j$. La $j^{ème}$ ligne de $CU + DL = R_3$ impose tout simplement la valeur de L_j, on a $R_{3_j} = L_{D_j}$.
2. Pour $j \in J_N$, la $j^{ème}$ ligne de C a exactement un terme non nul $C_{j,j} = 1$, $D_{j,j}$ est nul et la $j^{ème}$ ligne de l'équation $CU + DL = R_3$ impose seulement la valeur de U_j : on a donc $R_{3_j} = U_{N_j}$, où U_{N_j} est la valeur moyenne de la condition aux limites de Neumann $\overline{g}$ sur E_j, et $R_{1_j} = a_{\mathbf{I}}(\rho \vec{g}, \vec{u}_j)$.
3. Sinon, c'est à dire pour $j \in \overset{\circ}{J}$, la $j^{ème}$ ligne de C a exactement deux termes non nuls, $C_{j,j} = C_{j,j'} = 1$ et $D_{j,j} = 0$. On a $R_{1_j} = R_{1_{j'}} = a_{\mathbf{I}}(\rho \vec{g}, \vec{u}_j)$ et la $j^{ème}$ ligne de l'équation $CU + DL = R_3$ exprime la continuité du flux à travers E_j, on a donc $R_{3_j} = 0$.

Finalement, chaque colonne de C a au plus un terme non nul (égal à 1). On peut également permuter les colonnes de C selon la partition de $J \cup J'$. On peut ensuite réordonner ses lignes de façon à isoler celles correspondant à une condition aux limites de Dirichlet, c'est-à-dire que l'on réordonne les composantes de L. C est alors constituée de $2 \times N_c$ blocs. Le bloc d'indice $(1, i)$ est noté C_i, il est de taille $(N - N_D) \times$le nombre de faces de la cellule T_i. Chaque ligne de C_i a 1 ou deux termes non nuls et chaque colonne de C_i a au plus 1 terme non nul. Les blocs d'indice $(2, i)$ sont de taille $N_D \times$le nombre de faces de la cellule T_i et sont nuls. La matrice D réordonnée est diagonale par blocs, avec un bloc diagonal nul de taille $(N - N_D) \times (N - N_D)$ et un bloc diagonal identité de taille $N_D \times N_D$.

Une autre approche complètement équivalente consiste à considérer que les valeurs de pression sur la frontière de Dirichlet ne sont pas des degrés de liberté. Le multiplicateur de Lagrange utilisé est $\tilde{L}$, de taille $N - N_D = \overset{\circ}{N} + N_N$, qui n'a pas de composante sur Γ_D. Le système d'équations équivalent à (2.43) s'écrit sous la forme

$$\begin{pmatrix} A_{\mathcal{K}} & B^T & \tilde{C}^T \\ B & \mathbf{0} & \mathbf{0} \\ \tilde{C} & \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} U \\ P \\ \tilde{L} \end{pmatrix} = \begin{pmatrix} R_1 \\ R_2 \\ \tilde{R}_3 \end{pmatrix}. \tag{2.47}$$

La matrice $\tilde{C}$ est de taille $(N_N + \overset{\circ}{N}) \times (N_N + 2\overset{\circ}{N})$. Le vecteur $\tilde{R}_3$ n'exprime plus les conditions limites de Dirichlet. Les conditions de Dirichlet interviennent toujours, comme précédemment, dans R_1. Les matrices $A_{\mathcal{K},i}$, B_i, C_i, pour $i \in I$, restent définies comme précédemment (mêmes tailles, mêmes valeurs).

Il existe une unique solution à l'équation (2.43), qui par construction fournit les coefficients des décompositions de $\vec{u}_h$ et p_h, solutions de la formulation variationnelle approchée

sur les espace W_h et M_h décrits dans la section 2.1.3.2, selon les bases mentionnées dans la section 2.1.3.2.

2.1.3.4 Résolution

La matrice du système d'équations (2.43) n'est jamais assemblée et nous ne résolvons jamais un système d'équations de taille $(N+\overset{\circ}{N}) \times (N+\overset{\circ}{N})$. On commence par éliminer les variables U et P pour construire et résoudre une équation satisfaite par l'inconnue L. Comme $A_{\mathcal{K}}$, B et C sont des matrices blocs, avec des petits blocs, on peut éliminer U et P en inversant seulement des matrices locales, de petite taille. Une fois L connu, P et U peuvent être reconstitués par des produits matrice-vecteur locaux. Plus précisément, on définit

$$\left\{\begin{array}{lcl} M & = & CA_{\mathcal{K}}^{-1}\left(A_{\mathcal{K}} - B^T\left(BA_{\mathcal{K}}^{-1}B^T\right)^{-1}B\right)A_{\mathcal{K}}^{-1}C^T - D \\ R & = & CA_{\mathcal{K}}^{-1}\left(A_{\mathcal{K}} - B^T\left(BA_{\mathcal{K}}^{-1}B^T\right)^{-1}B\right)A_{\mathcal{K}}^{-1}R_1 + CA_{\mathcal{K}}^{-1}B^T\left(BA_{\mathcal{K}}^{-1}B^T\right)^{-1}R_2 - R_3. \end{array}\right. \tag{2.48}$$

ou

$$\left\{\begin{array}{lcl} \tilde{M} & = & \tilde{C}A_{\mathcal{K}}^{-1}\left(A_{\mathcal{K}} - B^T\left(BA_{\mathcal{K}}^{-1}B^T\right)^{-1}B\right)A_{\mathcal{K}}^{-1}\tilde{C}^T \\ \tilde{R} & = & \tilde{C}A_{\mathcal{K}}^{-1}\left(A_{\mathcal{K}} - B^T\left(BA_{\mathcal{K}}^{-1}B^T\right)^{-1}B\right)A_{\mathcal{K}}^{-1}R_1 + \tilde{C}A_{\mathcal{K}}^{-1}B^T\left(BA_{\mathcal{K}}^{-1}B^T\right)^{-1}R_2 - \tilde{R}_3. \end{array}\right. \tag{2.49}$$

On n'a pas à inverser $A_{\mathcal{K}}$ pour assembler $\tilde{M}$. En effet, $\tilde{M}$ est la somme des contributions des matrices locales, chaque contribution $\tilde{M}_i$ a bien la taille de $\tilde{M}$ (nombre de lignes de C_i) : on a $\tilde{M} = \sum_{i=1}^{N_c} \tilde{M}_i$ et $\tilde{R} = \sum_{i=1}^{N_c} \tilde{R}_i$ où

$$\left\{\begin{array}{lcl} \tilde{M}_i & = & C_iA_{\mathcal{K},i}^{-1}\left(A_{\mathcal{K},i} - B_i^T\left(B_iA_{\mathcal{K},i}^{-1}B_i^T\right)^{-1}B_i\right)A_{\mathcal{K},i}^{-1}C_i^T \\ \tilde{R}_i & = & C_iA_{\mathcal{K},i}^{-1}\left(A_{\mathcal{K},i} - B_i^T\left(B_iA_{\mathcal{K},i}^{-1}B_i^T\right)^{-1}B_i\right)A_{\mathcal{K},i}^{-1}R_{1,i} + C_iA_{\mathcal{K},i}^{-1}B_i^T\left(B_iA_{\mathcal{K},i}^{-1}B_i^T\right)^{-1}R_{2,i} - \tilde{R}_{3,i}. \end{array}\right. \tag{2.50}$$

On n'assemble pas non plus les matrices C_i, mais on utilise directement les fonctions linéaires associées :

- Soit M_{el} une matrice carrée dont le nombre de lignes est le nombre de faces de la cellule T_i ($i \in I$). Les faces adjacentes à T_i non incluses dans Γ_D sont $E_{j_1}, \ldots, E_{j_{n_i}}$. Alors pour $k, l \in i, \ldots, n_i$ on a $\left[C_iM_{el}C_i^T\right]_{j_k,j_l} = [M_{el}]_{k,l}$ et pour $k, l \notin j_1, \ldots, j_{n_i}$ $\left[C_iM_{el}C_i^T\right]_{k,l} = 0$.
- Soit V_{el} un vecteur de longueur le nombre de faces de la cellule T_i ($i \in I$). Alors pour $k \in i, \ldots, n_i$ on a $[C_iV_{el}]_{j_k} = [V_{el}]_k$ et pour $k \notin j_1, \ldots, j_{n_i}$ $[C_iV_{el}]_k = 0$.

Les équations suivantes sont vérifiées par les multiplicateurs de Lagrange :

$$ML = R \tag{2.51}$$

et

$$\tilde{M}\tilde{L} = \tilde{R}. \tag{2.52}$$

La matrice M est symétrique définie positive. On peut reconstruire P et U, toujours par blocs, suivant

$$\left\{ \begin{array}{rcl} P & = & \left(BA_{\mathcal{K}}B^T\right)^{-1}\left(BA_{\mathcal{K}}^{-1}(R_1 - \tilde{C}\tilde{L}) - R_2\right) \\ U & = & A_{\mathcal{K}}^{-1}\left(R_1 - B^T P - \tilde{C}^T\tilde{L}\right). \end{array} \right. \tag{2.53}$$

2.1.4 Implémentation des conditions aux limites

Dans la pratique, pour des raisons de flexibilité des codes de calcul, il est préférable que la taille des équations à résoudre ne dépende pas des conditions aux limites. Il est utile de comprendre cette mise en œuvre pour pouvoir implémenter facilement la dérivation.

Introduisons d'abord quelques définitions pour simplifier les notations :

$$\left\{ \begin{array}{rcl} M_1 & = & A_{\mathcal{K}}^{-1}\left(A_{\mathcal{K}} - B^T\left(BA_{\mathcal{K}}^{-1}B^T\right)^{-1}B\right)A_{\mathcal{K}}^{-1} \\ M_2 & = & A_{\mathcal{K}}^{-1}B^T\left(BA_{\mathcal{K}}^{-1}B^T\right)^{-1} \end{array} \right. \tag{2.54}$$

de sorte que l'équation (2.51) se réécrit

$$(CM_1C^T - D)L = CM_1R_1 + CM_2R_2 - R_3 \tag{2.55}$$

et l'équation (2.52) se réécrit

$$\tilde{C}M_1\tilde{C}^T\tilde{L} = \tilde{C}M_1R_1 + \tilde{C}M_2R_2 - \tilde{R}_3. \tag{2.56}$$

Notons que M_1 et M_2 ne dépendent pas des conditions aux limites.

On décrit ici l'algorithme utilisé dans la bibliothèque `LifeV` ([42]) : on construit d'abord l'équation $(\mathcal{E}_0)$ correspondant à un problème de Neumann pur homogène :

$$(\mathcal{E}_0) : \qquad C_0M_1C_0^T = C_0M_1R_{1,0} + C_0M_2R_2 - R_{3,0}.$$

- comme il n'y a pas de frontière de Dirichlet pour le problème décrit par $(\mathcal{E}_0)$, $R_{1,0}$ ne contient qu'une contribution du terme constant $\rho\vec{g}$. On peut décomposer dans le cas général $R_1 = R_{1,0} - L_D$, où L_D représente la contribution des conditions aux limites de Dirichlet.
- R_2 est indépendant des conditions aux limites puisqu'il représente le terme source, donc $R_{2,0} = R_2$;
- $R_{3,0}$ est nul car on considère des conditions de Neumann nulles. Pour faciliter ultérieurement la dérivation, on va considérer que $R_3 = U_N + L_D$ peut prendre la forme plus générale $R_3 = U_N + L_D + R_{3,0}$, où les composantes de $R_{3,0}$ correspondant à des faces frontières sont nulles. Choisir $R_{3,0}$ non nul revient à imposer une discontinuité du flux de $\vec{u}$.

La matrice $M_0 = C_0 M_1 C_0^T$ est construite en additionnant les contributions de matrices locales correspondant à chaque cellule.

On construit ensuite M à partir de M_0 et R à partir de R_0 en prenant en compte les conditions aux limites pas forcément de Neumann nulles. On impose d'abord les conditions de Dirichlet en "diagonalisant" partiellement l'équation, c'est-à-dire que l'on annule les termes non diagonaux des lignes et colonnes de M correspondant à des composantes de la frontière de Dirichlet. On modifie le second membre pour tenir compte des conditions de Dirichlet et compenser les termes annulés. On donne dans le Tableau 2.1 l'algorithme de diagonalisation.

$M \leftarrow M_0$;
$R \leftarrow R_0$;
pour $j \in J_D$
$\quad M_{j,j} = 1$;
$\quad R_j = L_{D_j}$;
$\quad$ pour $k \in J, k \neq j$
$\quad\quad R_k = R_k - M_{k,j} L_{D_j}$;
$\quad\quad M_{j,k} = M_{k,j} = 0.$

TAB. 2.1 – Algorithme de diagonalisation.

Les conditions de Neumann n'interviennent que dans le second membre : il reste à prendre en compte la modification R_3. On présente dans le Tableau 2.2 l'algorithme de prise en compte des conditions de Neumann.

pour $j \in J_N$
$\quad R_j = R_j - U_{lim\,j}.$

TAB. 2.2 – Algorithme de prise en compte des conditions de Neumann.

La reconstruction des inconnues U et P s'effectue aussi par blocs, l'écriture de l'algorithme de reconstruction est indépendante des conditions aux limites (les valeurs des conditions aux limites de Neumann interviennent dans le calcul de L mais pas dans la reconstruction ; les conditions aux limites de Dirichlet se retrouvent dans les valeurs de L) :

$$P = M_2^T (R_{1,0} - C_0^T L) - \left(B A_{\mathcal{K}} B^T\right)^{-1} R_2 \ ;$$
$$U = A_{\mathcal{K}}^{-1} \left(R_{1,0} - B^T P - C_0^T L\right).$$

2.2 Modèle de transport

Ce modèle décrit le transport des déchets à partir du site de stockage après la fuite des colis de déchets et de la barrière ouvragée, dû à l'écoulement de l'eau (vitesse de Darcy)

et au phénomène de diffusion dépendant des propriétés du sol.

2.2.1 Équation continue

On s'intéresse d'abord au transport de radionucléides qui ne précipitent pas, comme l'Iode 29 (dans les conditions considérées pour notre application). On considère donc l'équation linéaire, pour des contaminants indicés par i,

$$\begin{cases} \dfrac{\partial\left(\Phi R_i c_i\right)}{\partial t} + \operatorname{div}\left(c_i\vec{u} - D_i\vec{\nabla}c_i\right) + \lambda_i \Phi R_i c_i = \displaystyle\sum_{j\in I} \sigma_{ij}\lambda_j \Phi R_j c_j + \tau_i \text{ dans } \Omega, \\ \text{conditions aux limites sur } \partial\Omega\ . \end{cases} \tag{2.57}$$

Sous les hypothèses de l'absence de couplage entre les différents radionucléides, i.e. $\{pères(i)\} = \emptyset$, on obtient

$$\begin{cases} \Phi R_i \dfrac{\partial c_i}{\partial t} + \operatorname{div}\left(c_i\vec{u} - D_i\vec{\nabla}c_i\right) + \lambda_i \Phi R_i c_i = \tau_i, \\ \text{conditions aux limites sur } \partial\Omega. \end{cases} \tag{2.58}$$

Les équations concernant les différents radionucléides sont alors découplées. On introduit pour simplifier les changements de notations suivants :

$$\omega = R_i\Phi, \tag{2.59}$$
$$\lambda = \lambda_i, \tag{2.60}$$
$$D = D_i, \tag{2.61}$$
$$c = c_i, \tag{2.62}$$
$$\tau = \tau_i. \tag{2.63}$$

On va s'intéresser à la discrétisation en temps et en espace de l'équation de convection-diffusion linéaire suivante :

$$\begin{cases} \omega\dfrac{\partial c}{\partial t} + \operatorname{div}\left(c\vec{u} - D\vec{\nabla}c\right) + \omega\lambda c = \tau & \text{dans } \Omega, \\ c = c_D & \text{sur } \Gamma_D \text{ (conditions de Dirichlet)}, \\ \vec{\nabla}c\cdot\vec{n} = g_N & \text{sur } \Gamma_N \text{ (conditions de Neumann)}, \\ \left(c\vec{u} - D\vec{\nabla}c\right)\cdot\vec{n} = g_F & \text{sur } \Gamma_F \text{ (conditions de flux)}. \end{cases} \tag{2.64}$$

On rappelle que $\vec{u}$ est le champ de vitesse d'écoulement de l'eau, solution de l'équation (2.1). Le tenseur de diffusion $D = D_i$ peut aussi dépendre de $\vec{u}$ selon

$$D = d_e\mathbf{I} + |\vec{u}|\left[\alpha_l\varepsilon + \alpha_t\left(\mathbf{I} - \varepsilon\right)\right] \quad \text{où} \quad \varepsilon_{k,l} = \frac{\vec{u}_k\vec{u}_l}{|\vec{u}|^2}, \quad k,l = 1,\ldots,3 \tag{2.65}$$

où $\mathbf{I}$ est la matrice identité de taille 3×3. La partition $\partial\Omega = \overline{\Gamma}_D \cup \overline{\Gamma}_N \cup \overline{\Gamma}_F$ de la frontière du domaine de calcul est indépendante de celle utilisée pour le calcul d'écoulement. Pour

éviter des difficultés, on va supposer que

$$\vec{u} \cdot \vec{n} \geq 0 \text{ sur } \Gamma_N \cup \Gamma_F, \tag{2.66}$$

c'est-à-dire que l'on doit imposer des conditions de Dirichlet sur la partie de la frontière par laquelle le flux de Darcy est entrant.

2.2.2 Discrétisation en temps

On souhaite utiliser un schéma implicite pour le terme de diffusion, pour des raisons de stabilité (par exemple [76], chapitre 4.) et explicite pour les termes de convection, pour des raisons de précision (par exemple [76] chapitre 5.). Le choix d'un pas de temps doit satisfaire la condition de Courant-Friedrichs-Levy (condition CFL) [56, 76], déterminée par le maillage en espace et la vitesse de convection $\vec{u}$. Les coefficients de diffusion n'imposent pas de borne supérieure au pas de temps, en revanche un faible pas de temps demande de résoudre de nombreuses équations implicites (systèmes linéaires qui peuvent être de grande taille). Le compromis choisi entre ces contraintes est l'utilisation d'une méthode de splitting ne résolvant pas le même nombre d'équations de convection (explicites) que d'équations de diffusion (implicites). Cette méthode est développée dans [72].

Les paramètres de discrétisation en temps sont

- Δt, nombre réel strictement positif : pas de temps,
- M, nombre entier strictement positif : paramètre du splitting,
- $\Delta t_{conv} = \dfrac{\Delta t}{M}$, nombre réel strictement positif : pseudo-pas de temps de convection.

On note Δt_{CFL} le plus grand pas de temps vérifiant la condition CFL, dont l'expression est précisée par l'équation (2.76). Pour Δt fixé, les paramètres du splitting optimaux sont

$$M = \mathrm{E}\left(\frac{\Delta t}{\Delta t_{CFL}}\right) + 1, \quad \Delta t_{conv} = \frac{\Delta t}{M}, \tag{2.67}$$

où E désigne la partie entière.

Notre objectif est d'étudier l'évolution de la distribution de concentration c depuis l'instant $t_0 = 0$ jusqu'à l'instant $T = N\Delta t$.

On suppose que l'équation de convection d'inconnue c et de condition initiale $\underline{c}$

$$\left\{ \begin{array}{rcll} \omega \dfrac{c - \underline{c}}{\Delta t_{conv}} + \mathrm{div}\ (c\vec{u}) & = & \tau & \text{dans } \Omega \\ c & = & c_D & \text{sur } \Gamma_D \end{array} \right. \tag{2.68}$$

se discrétise sous la forme

$$\mathcal{E}_{conv}\left(C, \underline{C}\right) = 0, \tag{2.69}$$

où C (resp. $\underline{C}$) est une discrétisation en espace de c (resp. $\underline{c}$) à l'instant $\underline{t} + \Delta t_{conv}$ (resp.

$\underline{t}$), et que l'équation "de diffusion" d'inconnue c et de condition initiale $\underline{c}$

$$\left\{\begin{array}{rcll} \omega\dfrac{c-\underline{c}}{\Delta t}+\text{div}\left(-D\vec{\nabla}c\right)+\omega\lambda c & = & 0 & \text{dans } \Omega \\ c & = & c_D & \text{sur } \Gamma_D \\ \vec{\nabla}c\cdot\vec{n} & = & g_N & \text{sur } \Gamma_N \\ \vec{\nabla}c\cdot\vec{n} & = & g_F-c^0\vec{u}\cdot\vec{n} & \text{sur } \Gamma_F \end{array}\right. \tag{2.70}$$

se discrétise sous la forme

$$\mathcal{E}_{diff}\left(C,\underline{C}\right)=0. \tag{2.71}$$

On note $C^0, C^1, \ldots, C^N$ les discrétisations des approximations du champ de concentrations (vecteurs) aux instants $0, \Delta t, \ldots, N\Delta t$. On utilise également, pour $n = 0, \ldots, N-1$, les variables intermédiaires $C^{n,0}, C^{n,1}, C^{n,2}, \ldots, C^{n,M}$ (vecteurs de même taille que $C^0, C^1, \ldots, C^N$). L'algorithme de splitting utilisé est résumé dans le Tableau 2.3.

1. Faire, pour $n = 0, \ldots, N-1$,
2. $\quad C^{n,0} \leftarrow C^n$;
3. $\quad$ résoudre, pour $m = 0, \ldots, M-1$,
4. $\quad\quad \mathcal{E}_{conv}\left(C^{n,m+1}, C^{n,m}\right) = 0$;
5. $\quad$ résoudre
6. $\quad\quad \mathcal{E}_{diff}\left(C^{n+1}, C^{n,M}\right) = 0.$

TAB. 2.3 – Algorithme pour la méthode de splitting

2.2.3 Discrétisation en espace

On souhaite maintenant expliciter les opérateurs $\mathcal{E}_{conv}$ et $\mathcal{E}_{diff}$ des équations (2.69) et (2.71).

La triangulation utilisée est la même que pour l'équation de Darcy, donnée en section (2.1.3.1). Ainsi, nous n'aurons par d'interpolation à faire pour le flux de Darcy. Cette fois, chaque face frontière est incluse soit dans Γ_N, soit dans Γ_F, soit dans Γ_D et l'ensemble J est décomposé $J = \overset{\circ}{J} \cup J_N \cup J_F \cup J_D$, où J_F est l'ensemble des indices des faces incluses dans Γ_F et $\#J_F = N_F$. On utilise des éléments finis mixtes pour l'équation de diffusion et des volumes finis pour l'équation de convection.

2.2.3.1 Équation de convection

On discrétise l'équation (2.68) par un schéma volumes finis décentré amont. La composante C_i (resp. $\underline{C}_i$) du vecteur C (resp. $\underline{C}$) correspond à une approximation de la moyenne

du champ c (resp. $\underline{c}$) sur la cellule T_i. On définit, pour $i \in I$, $J_M(i)$ comme l'ensemble des indices des faces adjacentes à la cellule T_i puis, pour $i \in I$ et $j \in J_M(i)$,

$$\left\{ \begin{array}{rcrl} \mathrm{sgn}(i,j) & = & 1 & \text{si } i = i_1(j) \\ & & -1 & \text{si } i = i_2(j). \end{array} \right. \tag{2.72}$$

Enfin, on définit $\underline{C}_{i,j}$ comme la concentration amont à la face j, définie par le sens de traversée de l'approximation de $\vec{u}$ à travers la face E_j :

$$\left\{ \begin{array}{rcll} \underline{C}_{i,j} & = & \underline{C}_{i_1(j)} & \text{si } U_j \geq 0 \\ & & \underline{C}_{i_2(j)} & \text{si } U_j < 0 \text{ et } j \in \mathring{J} \\ & & C_{D_j} & \text{si } U_j < 0 \text{ et } j \in J_D. \end{array} \right. \tag{2.73}$$

Pour $j \notin J_M(i)$, on définit $\mathrm{sgn}(i,j) = 0$ et $\underline{C}_{i,j} = 0$. Pour tout $i \in I$ (indice de cellule), on obtient, en intégrant l'équation (2.68) sur la cellule T_i,

$$\frac{\omega}{\Delta t_{conv}} \mathrm{mes}(T_i)(C_i - \underline{C}_i) + \sum_{j \in J_M(i)} \underline{C}_{i,j} \mathrm{sgn}(i,j) U_j = \mathrm{mes}(T_i)\tau_i. \tag{2.74}$$

C'est une équation explicite : on obtient directement

$$C_i = \underline{C}_i + \frac{\Delta t_{conv}}{\omega}\tau_i - \frac{\Delta t_{conv}}{\omega \mathrm{mes}(T_i)} \sum_{j \in J_M(i)} \underline{C}_{i,j} \mathrm{sgn}(i,j) U_j. \tag{2.75}$$

La condition de stabilité CFL, qui traduit que le schéma de discrétisation est stable si on respecte l'idée qu'un front de concentration ne peut pas se déplacer de plus d'une cellule par pas de temps, s'explicite

$$\Delta t_{conv} \leq \Delta t_{CFL} = \min_{i \in I} \frac{\omega \mathrm{mes}\,(T_i)}{\displaystyle\sum_{j \in J_M(i)} \max\left(-\mathrm{sgn}(i,j) U_j, 0\right)}. \tag{2.76}$$

Le terme $\displaystyle\sum_{j \in J_M(i)} \max\left(-\mathrm{sgn}(i,j) U_j, 0\right)$ correspond au flux d'écoulement entrant dans la cellule T_i.

2.2.3.2 Équation de diffusion

L'équation (2.70) s'écrit sous forme mixte

$$\left\{ \begin{array}{rcll} \mathrm{div}\ \vec{v} + \omega \left(\dfrac{1}{\Delta t} + \lambda \right) c & = & \dfrac{\omega}{\Delta t} \underline{c} & \text{dans } \Omega, \\ \vec{v} & = & -D \vec{\nabla} c. & \text{dans } \Omega, \\ c & = & c_D & \text{sur } \Gamma_D, \\ \vec{v} \cdot \vec{n} & = & g & \text{sur } \Gamma_N \cup \Gamma_F \end{array} \right. \tag{2.77}$$

où

$$\left\{ \begin{array}{lcll} g & = & g_N & \text{sur } \Gamma_N, \\ & & g_F - \underline{c}\vec{u}\cdot\vec{n} & \text{sur } \Gamma_F. \end{array} \right. \tag{2.78}$$

(voir les similitudes avec l'équation (2.1)).

Grâce au terme d'évolution $\omega\left(\dfrac{1}{\Delta t}+\lambda\right)c$, on verra que l'équation à résoudre pour une formulation mixte comprendra une matrice symétrique définie positive. De plus, quelle que soit la formulation utilisée, une factorisation de la matrice à inverser peut être stockée et réutilisée pour tous les pas de temps. Si on utilise une formulation mixte hybride, l'étape de reconstruction de U et P à partir de L dans la formulation mixte hybride n'est donc plus négligeable en temps de calcul devant la résolution du système linéaire. Par conséquent nous discrétisons le problème sous forme mixte et pas sous forme mixte hybride.

Les fonctions de base pour le champ de vitesse (ici $\vec{v}$) dans une formulation mixte utilisant des éléments finis de plus bas degré sont continues et à divergence constante par morceaux, polynomiales par morceaux sur Ω. On a N fonction de bases $(\vec{v}_j)_{j\in J}$ vérifiant en particulier

$$\left\{ \begin{array}{ll} \forall j\in J, & \displaystyle\int_{T_{i_1(j)}} \text{div } \vec{v}_j = \int_{E_j} \vec{v}_j\cdot\vec{\mathrm{n}}_j = 1, \\ \forall j\in \overset{\circ}{J}, & \displaystyle\int_{T_{i_2(j)}} \text{div } \vec{v}_j = -1. \end{array} \right. \tag{2.79}$$

Elles sont reliées aux fonctions de base utilisées dans la formulation EFMH par

$$\left\{ \begin{array}{lcll} \vec{v}_j & = & \vec{u}_j & \text{pour } j\in J_N\cup J_F\cup J_D, \\ \vec{v}_j & \underset{p.p.}{=} & \vec{u}_j - \vec{u}_{j'} & \text{pour } j\in \overset{\circ}{J}, \end{array} \right. \tag{2.80}$$

où $\vec{u}_j$, pour $j\in J$ et $\vec{u}_{j'}$, pour $j\in\overset{\circ}{J}$ sont les fonctions de base caractérisées par les équations (2.32) et (2.33). Le signe $\underset{p.p.}{=}$ signifie qu'on a égalité presque partout, i.e. sauf sur une partie de Ω de mesure nulle. En effet, pour $j\in\overset{\circ}{J}$, $\vec{v}_j=\vec{u}_j$ sur $T_{i_1(j)}$, $-\vec{u}_{j'}$ sur $T_{i_2(j)}$, 0 sur T_i pour $i\notin\{i_1(j),i_2(j)\}$, et $\vec{v}_j$ est continue : l'égalité $\vec{v}_j=\vec{u}_j-\vec{u}_{j'}$ est vraie partout sauf sur E_j.

Les composantes de la décomposition de $\vec{v}$ selon cette base sont notées $(V_j)_{j\in J}$. La concentration c est approchée par une fonction constante par cellule : pour $i\in I$, C_i correspond à une approximation de la valeur moyenne de c sur T_i.

L'intégration sur le domaine de calcul Ω du produit scalaire de l'équation $\vec{v}+D\vec{\nabla}c=0$ (voir (2.77)) par les fonctions de base pour $\vec{v}$ permet d'obtenir, via les formules de Green :

1. Pour tout $j\in\overset{\circ}{J}$

$$\int_{T_{i_1(j)}\cup T_{i_2(j)}} D^{-1}\vec{v}\cdot\vec{v}_j - C_{i_1(j)} + C_{i_2(j)} = 0\ ; \tag{2.81}$$

2. Pour tout $j\in\Gamma_D$

$$\int_{T_{i_1(j)}} D^{-1}\vec{v}\cdot\vec{v}_j - C_{i_1(j)} = -C_{D_j}\ ; \tag{2.82}$$

3. Pour tout $j \in \Gamma_N \cup \Gamma_F$

$$V_j = V_{N_j} \tag{2.83}$$

(cette dernière équation n'est pas obtenue par intégration).

avec $V_{N_j} = \int_{E_j} g_N$ si $j \in J_N$ et $V_{N_j} = \int_{E_j} g_F - \underline{C}_{i_1(j)} \int_{E_j} \vec{u} \cdot \vec{\mathrm{n}}_j$ si $j \in J_F$. L'équation de conservation (première ligne de (2.77)) devient, par intégration sur chaque cellule,

$$\forall i \in I, \sum_{j \in J_M(i)} \mathrm{sgn}(i,j) V_j + \omega \left(\frac{1}{\Delta t} + \lambda \right) C_i = \frac{\omega}{\Delta t} \underline{C}_i. \tag{2.84}$$

Pour résoudre ces équations, on construit un système linéaire de taille $N \times N$, d'inconnue V, en remplaçant $C_{i_1(j)}$ et $C_{i_2(j)}$ dans les équations (2.81) et (2.82) par leur expression en fonction des composantes de V donnée par l'équation (2.84). Plus précisément, on définit la matrice de masse éléments finis mixtes A_D^M, de taille $N \times N$,

$$\left\{ \begin{array}{lll} A^M_{D_{j,k}} = \int_\Omega D^{-1} \vec{v}_j \cdot \vec{v}_k = & A_{D_{j,k}} + A_{D_{j',k'}} - A_{D_{j,k'}} - A_{D_{j',k}} & \text{si } j,k \in \mathring{J} \\ & A_{D_{j,k}} - A_{D_{j,k'}} & \text{si } j \in \mathring{J} \text{ et } k \in J_N \cup J_D \\ & A_{D_{j,k}} - A_{D_{j',k}} & \text{si } j \in J_N \cup J_D \text{ et } k \in \mathring{J} \\ & A_{D_{j,k}} & \text{si } j,k \in J_N \cup J_D \end{array} \right. \tag{2.85}$$

où A_D est la matrice de masse EFMH définie dans 2.1.3.3. La matrice A_D^M est construite à partir de l'assemblage des matrices élémentaire $A_{D,i}$ définies comme dans la section 2.1.3.3. La matrice A_D^M est symétrique définie positive.

L'équation (2.84) nous permet d'obtenir, pour $j \in \mathring{J}$,

$$-C_{i_1(j)} + C_{i_2(j)} = \frac{\Delta t}{\omega (1 + \lambda \Delta t)} \left(\sum_{k \in J} \left(\mathrm{sgn}(i_1(j),k) - \mathrm{sgn}(i_2(j),k) \right) V_k \right) + \frac{1}{1 + \lambda \Delta t} \left(-\underline{C}_{i_1(j)} + \underline{C}_{i_2(j)} \right) \tag{2.86}$$

et pour $j \in \Gamma_D$

$$-C_{i_1(j)} = \sum_{k \in J} (\mathrm{sgn}(i_1(j),k) V_k - \frac{1}{1 + \lambda \Delta t} \underline{C}_{i_1(j)}. \tag{2.87}$$

Soit B^{evol} la matrice de taille $N \times N$, symétrique, définie par

$$\left\{ \begin{array}{rcrl} B^{evol}_{j,k} & = & 2 & \text{si } j = k, \\ & & 1 & \text{si } i_1(j) = i_1(k) \text{ ou } i_2(j) = i_2(k), \\ & & -1 & \text{si } i_1(j) = i_2(k) \text{ ou } i_2(j) = i_1(k), \\ & & 0 & \text{sinon.} \end{array} \right. \tag{2.88}$$

B^{evol} est une matrice symétrique définie positive.

Pour un problème de Dirichlet pur, l'équation à résoudre est

$$\left(A^M + \frac{\Delta t}{\omega\,(1+\lambda\Delta t)} B^{evol}\right) V = -R_D + \frac{1}{1+\lambda\Delta t} R_0 \tag{2.89}$$

où R_D contient les valeurs des conditions aux limites de Dirichlet et $R_{0_j} = \underline{C}_{i_1(j)} - \underline{C}_{i_2(j)}$ ou $\underline{C}_{i_1(j)}$.

Pour un problème quelconque, on diagonalise l'équation pour prendre en compte les conditions aux limites de Neumann, en utilisant un algorithme similaire à celui du Tableau 2.1. Remarquons que dans la formulation EFMH, où les inconnues de l'équation étaient des valeurs de charge en eau et non des flux, on diagonalisait pour appliquer des conditions aux limites de Dirichlet et non de Neumann.

On termine cette étape en calculant C à partir de l'expression (2.84).

2.2.4 Introduction d'un terme de précipitation

Lorsque les radionucléides peuvent précipiter, on doit considérer le système d'équations suivant, qui modélise le transport des phases liquide (concentration c_i) et solide (concentration f_i) du radionucléide i et l'échange entre les phases liquide et solide :

$$\left\{\begin{array}{l} \dfrac{\partial\,(\Phi R_i c_i)}{\partial t} + \mathrm{div}\,\left(c_i\vec{u} - D_i\vec{\nabla}c_i\right) + \lambda_i \Phi R_i c_i = \displaystyle\sum_{j\in I} \sigma_{ij}\lambda_j \Phi R_j c_j + \Phi s_i + \tau_i \\ \dfrac{\partial\,((1-\Phi) f_i)}{\partial t} + \lambda_i (1-\Phi) f_i = \displaystyle\sum_{j\in I} \sigma_{ij}\lambda_j (1-\Phi) f_j - \Phi s_i, \end{array}\right. \tag{2.90}$$

avec

$$\left\{\begin{array}{l} s_i = \omega_i (c_i^{sat} - c_i)\delta_i, \\ \delta_i = 0 \text{ si } f_i = 0 \text{ et } c_i < c_i^{sat}, \\ \delta_i = 1 \text{ sinon.} \end{array}\right. \tag{2.91}$$

La première ligne de l'équation (2.90) correspond à l'équation de transport linéaire (2.57) avec une source supplémentaire s_i représentant une non-linéarité. Le terme d'échange s_i entre les phases liquide et solide est représenté sur la Figure 2.2. On peut toujours utiliser une méthode de séparation d'opérateur, pour résoudre séparément, à chaque pas de temps,

1. une équation en phase solide, en supposant que la concentration non précipitée c_i est fixée. Cette équation comporte un terme de précipitation discontinu en $f_i = 0$. Elle demande de résoudre uniquement des équations locales (scalaires). Cette résolution nous fournit en particulier un terme "source" s_i pour l'étape suivante.

2. une équation de transport linéaire pour la phase liquide, en supposant que le terme source $\tau = \Phi s_i + \tau_i$ est fixé. Cette équation est résolue suivant les étapes 2. à 6. de l'algorithme donné dans le Tableau 2.3.

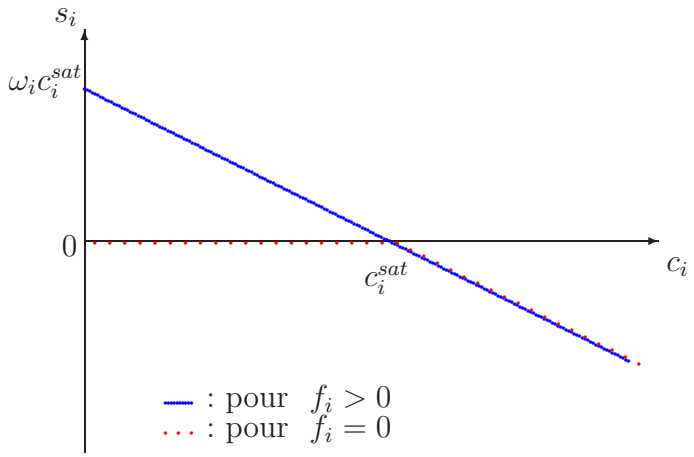


FIG. 2.2 – Terme d'échange entre les phases solide et liquide utilisé dans l'équation (2.90).

Régularisation. La fonction s_i définie par (2.91) n'est pas différentiable par rapport aux concentrations aux points où $f_i = 0$ et $c_i = c_i^{\text{sat}}$. Pour une éventuelle prise en compte du phénomène de précipitation dans une analyse de sensibilité déterministe, reposant sur un calcul de dérivée, on propose de remplacer s_i par une fonction plus régulière s_i^r dépendant d'un paramètre supplémentaire f_i^{lim}

$$\begin{array}{rcll} s_i^r & = & \omega_i \left(c_i^{sat} - c_i\right) & \text{si} \quad c_i \geq c_i^{sat} \\ & & \omega_i \left(c_i^{sat} - c_i\right) \min\left(\dfrac{f_i}{f_i^{lim}}, 1\right) & \text{si} \quad c_i \leq c_i^{sat}. \end{array}$$

2.3 Réduction de dimension du modèle

Pour l'analyse de sensibilité, qu'elle soit déterministe ou probabiliste, on devra choisir une fonction à analyser. Une fonction analysable numériquement prend en entrée un vecteur de nombres réels et retourne un vecteur de nombre réels.

Pour chaque étude numérique, on devra choisir un certain nombre n_{ip} de paramètres d'entrée et un opérateur de paramétrisation $\mathcal{P}$ défini sur $\mathbb{R}^{n_{ip}}$ à valeurs dans l'espace des paramètres physiques du problème, et choisir un certain nombre n_o d'observations et définir un opérateur d'observation $\mathcal{O}$ défini sur l'espace des degrés de liberté du problème et à valeurs dans $\mathbb{R}^{n_o}$.

On analysera

$$F = \mathcal{O} \circ \tilde{F} \circ \mathcal{P}, \tag{2.92}$$

où $\tilde{F}$ est définie implicitement par une équation d'état ou d'évolution.

Pour le problème d'écoulement,

$$\tilde{F} = \tilde{F}_{ec} : \mathcal{K} \mapsto \begin{pmatrix} U \\ P \\ L \end{pmatrix}. \tag{2.93}$$

On pourrait considérer les conditions aux limites comme des entrées de $\tilde{F}_{ec}$, au même titre que $\mathcal{K}$.

Pour le problème de transport,

$$\tilde{F} = \tilde{F}_{tr} : (\Phi, R_i, D_i, \lambda_i, \tau_i; U) \mapsto \begin{pmatrix} C^0, & \dots & , C^n \\ V^0, & \dots & , V^n \end{pmatrix}. \tag{2.94}$$

Dans les applications, on ne s'intéressera pas aux paramètres ω_i et c_i^{sat}. Là aussi, les conditions aux limites et éventuellement la condition initiale pourraient être des entrées de $\tilde{F}_{tr}$.

Le modèle complet est

$$\tilde{F} : (\Phi, R_i, D_i, \lambda_i, \tau_i; \mathcal{K}) \mapsto \tilde{F}_{tr}\left(\Phi, R_i, D_i, \lambda_i, \sigma_{ii}, \tau_i, \omega_i, c_i^{sat}; \tilde{F}_{ec_{1,\dots,n}}(\mathcal{K})\right). \tag{2.95}$$

Lorsque les espaces de départ et d'arrivée de $\tilde{F}$ ne sont pas de grande dimension, l'analyse de sensibilité de $\tilde{F}$ est également intéressante, en particulier pour guider les choix de $\mathcal{P}$ et $\mathcal{O}$ (voir [19, 1]). Pour les exemples que nous étudions, ceci n'est pas possible. En effet nous résolvons des problèmes à plusieurs centaines de milliers de degrés de liberté, l'analyse de la matrice Jacobienne correspondante $\tilde{F}'$ serait trop coûteuse. On a donc besoin d'informations extérieures (expertises) pour choisir les paramétrages $\mathcal{P}$. On donne des exemples de choix pour $\mathcal{P}$ et $\mathcal{O}$ dans la section 4.4.1.1 et dans le Chapitre 7. Pour les applications que nous considérons, $\mathcal{O}$ consiste à reconstituer des flux.

On propose aussi ici un exemple de paramétrisation ne consistant pas à faire une hypothèse de régularité sur les paramètres d'entrée. On considère l'équation monodimensionnelle

$$\begin{cases} u = -\mathcal{K}(x)\dfrac{\partial p}{\partial x} \quad \text{sur } [0, L] \\ \dfrac{\partial u}{\partial x} = 0 \quad \text{sur } [0, L] \\ p(0) = p_0, p(L) = p_L \end{cases}$$

de solution exacte

$$u = \frac{p_0 - p_L}{\displaystyle\int_0^L \left(\frac{1}{\mathcal{K}(x)}\mathrm{d}x\right)}.$$

L'intervalle $[0, L]$ est divisé en n_z couches de hauteurs L_i, $i = 1, ..., n_z$, $\sum_i L_i = L$. On décompose $\mathcal{K}(x)$ sous la forme

$$\text{pour } x \in [x_i, x_i + L_i], \quad \mathcal{K}(x) = K_i \exp\left(\sigma_i^2 \alpha(x)\right) \tag{2.96}$$

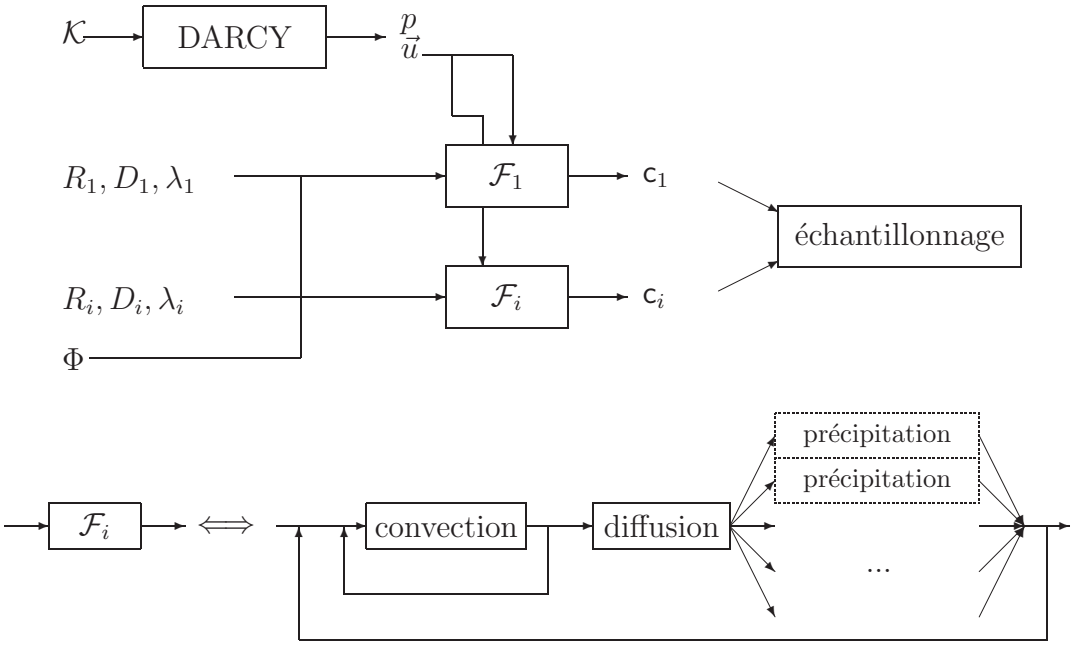


FIG. 2.3 – Schéma récapitulatif du modèle complet, sous les hypothèses de l'absence de filiation et de l'absence de précipitation. On a un bloc "$\mathcal{F}_i$" par radionucléide et, pour chaque radionucléide, un bloc "précipitation" par cellule de la triangulation.

avec

$$\int_{x_i}^{x_i+L_i} \alpha(x)\mathrm{d}x = 0, \quad \frac{1}{L_i}\int_{x_i}^{x_i+L_i} \alpha^2(x)\mathrm{d}x = 1. \tag{2.97}$$

On définit pour $n \in \mathbb{N}$, $i = 1, ..., n_z$ les moments

$$M_{n_i} = \frac{1}{L_i}\int_{x_i}^{x_i+L_i} \alpha^n(x)\mathrm{d}x \tag{2.98}$$

puis

$$\eta_i = \sum_{n=1}^{+\infty} M_{n_i}\frac{\sigma_i^{2n}}{n!} > 0. \tag{2.99}$$

Alors

$$u = \frac{p_0 - p_L}{\displaystyle\sum_i \frac{L_i\,(1+\eta_i)}{K_i}}. \tag{2.100}$$

Une paramétrisation en (K_i, η_i) a l'intérêt de permettre de quantifier l'influence des hétérogénéités (les moments d'ordre> 1) sans imposer leur forme.

Chapitre 3

Analyse de sensibilité

Dans ce chapitre, nous nous intéressons à différents aspects de l'analyse de sensibilité d'un modèle général F, c'est à dire une application régulière

$$F : \Omega_{ip} \subset \mathbb{R}^{n_{ip}} \to \Omega_{op} \subset \mathbb{R}^{n_{op}}, \tag{3.1}$$

et à l'analyse d'incertitude des valeurs de retour du modèle F, qui dépend de l'incertitude sur ses paramètres de départ. Par « analyse de sensibilité », nous entendons la quantification et l'analyse des dépendances entre les entrées et les sorties du modèle mathématique. La première section est consacrée aux analyses de type probabiliste Monte Carlo, dites globales car elles prennent en compte la totalité de la plage de variations possible pour les paramètres d'entrée. La seconde section est consacrée à l'analyse déterministe, locale lorsque F est non linéaire. La décomposition en valeurs singulières est au centre de ces deux types d'approche : c'est l'outil principal en analyse déterministe. Mais cette décomposition est aussi très utile en analyse statistique, comme nous le verrons dans la troisième section.

3.1 Analyses probabilistes

Nous allons nous intéresser aux méthodes statistiques de type Monte-Carlo, en nous concentrant sur les outils qui ont été utilisés pour l'analyse d'incertitude et de sensibilité du problème d'écoulement à l'ANDRA et qui sont mentionnés dans le Chapitre 7. Ce n'est qu'un exemple, couramment utilisé étant donné sa simplicité de mise en œuvre, des outils statistiques existant.

3.1.1 Formalisme probabiliste

Les paramètres de départ du modèle F sont considérés comme des variables aléatoires. L'incertitude sur les paramètres de départ se traduit par le choix des lois de probabilité suivies par ces variables. Les lois peuvent prendre en compte différentes sortes d'incertitudes, provenant des variabilités spatiale ou temporelle, d'une homogénéisation, ou

de l'imprécision des appareils de mesure. Les contraintes physiques reliant différents paramètres se traduisent par des relations de dépendances entre ces lois. On peut représenter analytiquement la dépendance entre deux variables aléatoires X_1 et X_2 par exemple en définissant une loi de probabilité $\mathcal{P}_1$ suivie par X_1 et une loi de probabilité $\mathcal{P}_R$ suivie par la variable aléatoire $R = \dfrac{X_1}{X_2}$, en considérant que les variables X_1 et R sont indépendantes.

En pratique, on considérera uniquement des variables aléatoires absolument continues, c'est-à-dire que les lois suivies peuvent être représentées par des densités de probabilité. Plus précisément, une variable aléatoire X prend ses valeurs dans un intervalle I_X de $\mathbb{R}$. La fonction de répartition de X est une fonction croissante sur I_X, définie par

$$\begin{array}{rcl} I_X & \rightarrow & [0,1], \\ x & \mapsto & P(X \leq x). \end{array}$$

Si X est absolument continue, alors sa fonction de répartition est dérivable, à dérivée continue par morceaux. On peut alors définir la fonction densité de probabilité f_X de X comme la dérivée de sa fonction de répartition. Les variables aléatoires X_1 et X_2 sont indépendantes si $P(X_1 \leq x_1 \text{ et } X_2 \leq x_2) = P(X_1 \leq x_1)P(X_2 \leq x_2)$. On définit à partir des densités de probabilité l'espérance $E(X)$ de la variable aléatoire X par

$$E(X) = \int_{I_X} x f_X(x) \mathrm{d}x, \tag{3.2}$$

qui vérifie la propriété

$$E(F(X)) = \int_{I_X} F(x) f_X(x) \mathrm{d}x, \tag{3.3}$$

sa variance

$$V(X) = E((X - E(X))^2) = E(X^2) - (E(X))^2, \tag{3.4}$$

et la covariance $Cov(X_1, X_2)$ de deux variables aléatoires X_1 et X_2

$$Cov(X_1, X_2) = E\left((X_1 - E(X_1))(X_2 - E(X_2))\right) = E(X_1 X_2) - E(X_1)E(X_2). \tag{3.5}$$

Les valeurs de retour de F sont elles aussi des variables aléatoires. L'analyse d'incertitude probabiliste du modèle consiste à évaluer les lois suivies par les valeurs de retour de F (par exemple en analysant des diagrammes de dispersion) ou certaines propriétés de ces lois (leurs écarts types par exemple). L'analyse de sensibilité consiste à analyser les dépendances entre les paramètres de départ et les variables d'arrivée. Ces dépendances peuvent être quantifiées par différents indicateurs, dont nous donnerons quelques exemples dans la section 3.1.3.

En revanche, on considère qu'il n'y a pas d'incertitude sur le modèle F lui-même.

Les principales étapes d'une analyse de type Monte-Carlo sont la génération d'un échantillon raisonnable de paramètres de départ, l'application du modèle F à tous les éléments de l'échantillon, et enfin l'analyse de l'échantillon de résultats en relation avec l'échantillon de départ. Ce type d'analyse est dit global car il couvre l'ensemble du spectre de variations possible pour les paramètres de départ.

3.1.2 Échantillonnage

On considère donc un vecteur aléatoire $X = \left(X_1, ..., X_{n_{ip}}\right)^T$ prenant ses valeurs dans Ω_{ip}. On veut constituer un échantillon représentatif de jeux de paramètres de départ :

$$\left(x^1, ..., x^N\right) \in \Omega_{ip}^N, \text{ et pour } n = 1, ..., N, \ x^n = \left(x_1^n, ..., x_{n_{ip}}^n\right)^T.$$

La solution la plus simple est d'utiliser un échantillonnage aléatoire, c'est-à-dire de générer indépendamment N jeux de paramètres $x^1, ..., x^N$ suivant la loi de X. Selon la loi des grands nombres, les propriétés empiriques de cet échantillon seront proches des propriétés théoriques de la loi de X pour N suffisamment grand. Cependant, lorsque les fonctions de densités de probabilité ne sont pas uniformes (lois normales par exemple), la taille d'échantillon N qui permettra de reproduire correctement les têtes et queues de distribution peut être très grande. Des méthodes d'échantillonnage spécifiques permettent de forcer certaines propriétés empiriques de l'échantillon, par exemple elles peuvent imposer qu'il contienne un nombre fixé de représentants dans certains sous-ensembles de Ω_{ip}

La méthode la plus précise est l'échantillonnage stratifié. Pour construire un échantillon de taille N, on commence par diviser l'ensemble Ω_{ip} en N sous ensemble équiprobables : $\Omega_{ip} = \cup_{n=1,...,N} \Omega_{ip_n}$. On choisit ensuite aléatoirement, pour chaque $n = 1, ..., N$, x^n suivant la loi de X conditionnée par l'appartenance au sous-ensemble Ω_{ip_n}. Une des principales difficultés de cette méthode est la mise en œuvre du découpage en sous-ensembles équiprobables.

Le compromis utilisé dans les calculs de sûreté de l'ANDRA est l'échantillonnage hypercube latin (LHS). Pour chaque paramètre de départ représenté par la variable aléatoire X_i $(i = 1, ..., n_{ip})$, ou variable intermédiaire indépendante R_i $(i = 1, ..., n_{ip})$ dans le cas où tous les paramètres de départ de F ne sont pas indépendants, on divise l'ensemble des valeurs possibles pour X_i ou R_i en N intervalles équiprobables, et on choisit aléatoirement, suivant la loi de X_i ou R_i conditionnée par l'appartenance à ces intervalles, une valeur x_i^n ou r_i^n, $n = 1, ..., N$, dans chacun des N intervalles. Ces valeurs sont ensuite combinées aléatoirement pour former N jeux de paramètres de départ, de sorte que chaque valeur x_i^n ou r_i^n soit utilisée exactement une fois dans l'échantillon. Un inconvénient de cette méthode est que l'on ne peut pas augmenter simplement la taille d'un échantillon.

Ces méthodes d'échantillonnage sont détaillées par exemple dans [32].

3.1.3 Indicateurs statistiques

Un échantillon de valeurs d'arrivée de taille N est généré en appliquant F à chaque élément de l'échantillon de jeux de paramètres : $y = \left(y^1, ..., y^N\right) \in \Omega_{op}^N$, et pour $n = 1, ..., N$, $y^n = F(x^n) = \left(y_1^n, ..., y_{n_{op}}^n\right)^T$. Les dépendances entre les paramètres de départ et les valeurs de retour de F peuvent être quantifiées (mais pas représentées exhaustivement) par différents indicateurs statistiques.

Nous allons définir ici les indicateurs qui seront utilisés pour les applications numériques du Chapitre 7. Ils sont introduits par exemple dans [47].

Tous ces indicateurs prennent leur valeur dans l'intervalle $[-1, 1]$. Plus la valeur absolue d'un indicateur est proche de 1, plus la dépendance {paramètre/résultat} quantifiée par cet indicateur est forte. Le signe de l'indicateur dépend du sens de variation de la relation entre paramètre et résultat (l'indicateur est positif si cette relation est croissante). Des valeurs absolues plus petites qu'un seuil, dépendant de l'indicateur considéré, indiquent une dépendance très faible. Un point crucial est le choix d'un indicateur bien adapté au problème étudié. Enfin, nous n'avons abordé le choix d'indicateurs pertinents pour des modèles non monotones (par exemple, nous de proposons pas d'indicateur pertinent pour $x \mapsto x^2$).

Ces coefficients permettent de classer les paramètres de départ de F en fonction de leur degré d'influence. Des précisions sur ces outils sont données par exemple dans [33].

3.1.3.1 Coefficient de Pearson

Le coefficient de corrélation, ou coefficient de Pearson, est un premier indicateur permettant de quantifier le degré de dépendance entre deux variables aléatoires. Dans le milieu des probabilités, cet indicateur, comme tous ceux que nous mentionnerons dans la suite, est dit déterministe, c'est-à-dire qu'il est défini pour un échantillon donné : une fois l'échantillon fixé, l'aspect probabiliste disparaît. On peut le relier aux covariances et aux variances empiriques : si $x_i = (x_i^1, ..., x_i^N)$ est un échantillon de la variable aléatoire X_i et $y_j = (y_j^1, ..., y_j^N)$ est un échantillon de la variable aléatoire Y_j, alors le coefficient de corrélation entre x_i et y_j est

$$cor(x_i, y_j) = \frac{\sum_{n=1}^{N} (x_i^n - \overline{x}_i)\left(y_j^n - \overline{y}_j\right)}{\sqrt{\sum_{n=1}^{N} (x_i^n - \overline{x}_i)^2}\sqrt{\sum_{n=1}^{N} \left(y_j^n - \overline{y}_j\right)^2}} \tag{3.6}$$

où $\overline{x}_i$ désigne la moyenne empirique : $\overline{x}_i = \dfrac{x_i^1 + ... + x_i^N}{N}$.

Lorsque les variables X_i et Y_j sont indépendantes alors la covariance

$$Cov(X_i, Y_j) = E\left((X_i - E(X_i))\left(Y_j - E(Y_j)\right)\right) \tag{3.7}$$

est nulle et le coefficient de corrélation $cor(x_i, x_j)$ sera presque nul pour un échantillon suffisamment grand. Notons que la réciproque est fausse : des variables aléatoires non indépendantes peuvent avoir une covariance nulle. Si X_i et Y_j sont proportionnels alors $cor(x_i, y_j) = \pm 1$ et le signe du coefficient de corrélation est le signe du coefficient de proportionnalité entre x_i et y_j.

Dans le cas où F est linéaire, on écrit, pour $n = 1, ..., N$, $y^n = Ax^n$, où A est une matrice non aléatoire de taille $n_{ip} \times n_{op}$. Le coefficient de corrélation (3.6) peut alors s'exprimer à

partir des propriétés de l'échantillon x et de la matrice A :

$$cor\left(x_i,(Ax)_j\right)=\frac{\displaystyle\sum_{k=1}^{n_{ip}}A_{jk}cor(x_i,x_k)\hat{\sigma}(x_k)}{\sqrt{\displaystyle\sum_{k=1}^{n_{ip}}\sum_{l=1}^{n_{ip}}A_{jk}A_{jl}cor(x_k,x_l)\hat{\sigma}(x_k)\hat{\sigma}(x_l)}} \tag{3.8}$$

où $\hat{\sigma}$ désigne un estimateur de l'écart type :

$$\hat{\sigma}(x_k)=\sqrt{\frac{1}{N-1}\sum_{n=1}^{N}\left(x_k^n-\overline{x}_k\right)^2}. \tag{3.9}$$

Les coefficients de Pearson sont donc proportionnels aux coefficients des relations linéaires entre les paramètres de départ et d'arrivée. Nous attirons l'attention sur l'absence de coefficients de y dans la relation 3.8. Elle rapproche aussi les coefficients de corrélation pour un problème linéarisé des coefficients de la matrice de Jacobienne de F. La donnée de tous les coefficients de Pearson donne l'ensemble des dépendances entre les entrées et les sorties du modèles, mais pas de manière condensée (on a autant de coefficients de corrélation que de coefficients dans la matrice Jacobienne de F).

Si F est fortement non linéaire, alors l'utilisation de cet indicateur devient moins pertinente (il est possible d'obtenir une corrélation nulle entre des variables dépendantes). On obtient par exemple, si X suit une loi centrée symétrique (loi normale centrée par exemple), et Fp est une fonction paire (la fonction carrée par exemple), étant donné que tous les moments d'ordre impair de la loi de X sont nuls, $Cov(X,Fp(X))=0$.

Pour quantifier des dépendances dans les cas non linéaires, on peut avoir recours à des indicateurs de rang.

3.1.3.2 Coefficient de corrélation de Spearman

Lorsque les relations entre les paramètres de départ et d'arrivée de F sont fortement non linéaires, mais monotones, il peut être intéressant de remplacer les valeurs des éléments des échantillons de départ et d'arrivée par leur rang dans l'échantillon. On ne mesurera plus un coefficient de linéarité entre des variables, mais un degré de monotonicité.

Le coefficient de corrélation de Spearman est le coefficient de Pearson calculé pour ces rangs : on définit

$$\text{Pour } i=1,...,n_{ip},\ n=1,...,N,\quad r_{x_i}^n=\#\left(\{m\in 1,...,N|x_i^m\leq x_i^n\}\right)\ ; \tag{3.10}$$

$$\text{Pour } i=1,...,n_{op},\ n=1,...,N,\quad r_{y_i}^n=\#\left(\{m\in 1,...,N|y_i^m\leq y_i^n\}\right). \tag{3.11}$$

Alors le coefficient de corrélation de Spearman SPEAR(x_i,y_j) est défini par

$$\text{SPEAR}(x_i,y_j)=cor\left(r_{x_i},r_{y_j}\right). \tag{3.12}$$

3.1.3.3 Coefficient de corrélation partiel (PCC) et coefficient de corrélation de rang partiel (PRCC)

Lorsque les paramètres de départ de F ne sont pas indépendants, le coefficient de corrélation de Pearson ou de Spearman entre une entrée et une sortie particulières dépend du modèle F mais aussi des corrélations entre les paramètres de départ. Ceci est clairement exprimé pour le cas linéaire dans l'expression (3.8). Ces coefficients peuvent être corrigés pour réduire l'effet des autres paramètres de départ lorsqu'on quantifie la dépendance entre un paramètre de départ et un paramètre d'arrivée particuliers. Lorsqu'on calcule le PCC de x_i et y_j, on efface non seulement l'effet des dépendances entre x_i et x_k pour $k \neq i$, mais aussi l'influence directe de x_k sur y_j par l'intermédiaire du modèle F, pour $k \neq i$.

Pour calculer le coefficient de corrélation partiel $\mathrm{PCC}(x_i, y_j)$, $i = 1, ..., n_{ip}$, $j = 1, ..., n_{op}$, on considère les modèles de régression linéaire

$$\hat{x}_i = c_0 + \sum_{k=1, k\neq i}^{n_{ip}} c_k x_k \tag{3.13}$$

et

$$\hat{y}_{j|i} = b_{0|i} + \sum_{k=1, k\neq i}^{n_{ip}} b_{k|i} x_k \tag{3.14}$$

où les coefficients c_k, $b_{k|i}$, $k = 1, ..., i-1, i+1, ..., n_{ip}$, dépendant de i et j, sont choisis pour minimiser les erreurs quadratiques sur les échantillons $r^2(x_i) = \sum_{n=1}^{N} (x_i^n - \hat{x}_i^n)^2$ et $r^2(y_{j|i}) = \sum_{n=1}^{N} \left(y_j^n - \hat{y}_{j|i}^n\right)^2$. Alors

$$\mathrm{PCC}(x_i, y_j) = cor(x_i - \hat{x}_i, y_j - \hat{y}_{j|i}). \tag{3.15}$$

Dans le cas linéaire, on obtient

$$\begin{aligned} \mathrm{PCC}\left(x_i, (Ax)_j\right) &= 0 && \text{si } A_{ji} = 0 \\ &= \mathrm{sgn}(A_{ji}) && \text{sinon} \end{aligned} \tag{3.16}$$

Dans ce cas, le PCC renseigne donc uniquement sur le signe de la relation directe entre x_i et y_j.

Vérifions le résultat (3.16).

On suppose qu'on a construit la régression linéaire

$$\hat{x}_i = c_0 + \sum_{k\neq i} c_k x_k$$

et on cherche b_0, $b_{k|i}$, $k \neq i$ minimisant

$$r^2(y_{j|i}) = \sum_{n=1}^{N} \left(y_j^n - b_{0|i} - \sum_{k\neq i} b_{k|i} x_k^n \right)^2,$$

où

$$y_j = (Ax)_j.$$

Si $A_{ji} = 0$, *alors* $y_j = \sum_{k \neq i} A_{jk} x_k$, *et* $r^2(y_{j|i})$ *s'annule pour* $b_{0|i} = 0$, $b_{k|i} = A_{jk}$, $k \neq i$, *donc* $y_j - \hat{y}_j = 0$, *donc* $cor(x_i - \hat{x}_i, y_j - \hat{y}_{j|i}) = 0$, *i.e.* $\mathrm{PCC}(x_i, y_j) = 0$.
Sinon, i.e. si $A_{ji} \neq 0$ *alors, pour* $n = 1, ..., N$,

$$y_j^n - b_{0|i} - \sum_{k \neq i} b_{k|i} x_k^n = A_{ji} \left(x_i^n - \frac{b_{0|i}}{A_{ji}} - \sum_{k \neq i} \left(\frac{b_{k|i}}{A_{ji}} - \frac{A_{jk}}{A_{ji}} \right) x_k \right)$$

donc par définition de la régression $\hat{x}_i$, *on minimise* $r^2(y_{j|i})$ *pour* $\frac{b_{0|i}}{A_{ji}} = c_0$ *et* $\frac{b_{k|i}}{A_{ji}} - \frac{A_{jk}}{A_{ji}} = c_k$, $k \neq i$ *i.e.* $b_{0|i} = c_0 A_{ji}$ *et* $b_{k|i} = c_k A_{ji} + A_{jk}$, $k \neq i$. *On en déduit* $y_j - \hat{y}_{j|i} = A_{ji}(x_i - \hat{x}_i)$. $(y_j - \hat{y}_j)$ *est donc proportionnel à* $(x_i - \hat{x}_i)$, *donc* $cor(y_j - \hat{y}_j, x_i - \hat{x}_i) = \pm 1$ *et* $\mathrm{PCC}(x_i, y_j) = \mathrm{sgn}(A_{ji})$.

Comme le coefficient de Spearman, le PCC peut être calculé pour les rangs des valeurs dans leur échantillon : on obtient le coefficient de corrélation de rang partiel (PRCC)

$$\mathrm{PRCC}(x_i, y_j) = \mathrm{PCC}\left(r_{x_i}, r_{y_j}\right). \tag{3.17}$$

3.1.3.4 Coefficient de régression de rang standardisé (SRRC)

Le coefficient de régression standardisé (SRC) quantifie l'effet de la variation d'un paramètre de départ d'une fraction fixée de son écart type. On considère le modèle de régression linéaire, non partiel,

$$\hat{y}_j = b_0 + \sum_{k=1}^{n_{ip}} b_k x_k \tag{3.18}$$

où les coefficients b_k, $k = 1, ..., n_{ip}$, dépendant de j, sont choisis pour minimiser l'erreur quadratiques sur l'échantillon $\sum_{n=1}^{N} \left(y_j^n - \hat{y}_j^n\right)^2$. Le SRC est défini par

$$\mathrm{SRC}(x_i, y_j) = \frac{b_i \hat{\sigma}(x_i)}{\hat{\sigma}(y_j)} \tag{3.19}$$

où $\hat{\sigma}$ est défini dans l'équation (3.9).

C'est le coefficient de corrélation calculé après centrage et réduction des variables.

Si F est linéaire et pour un échantillon suffisamment grand, alors $b_0 = 0$ et $b_k = A_{jk}$, $k = 1, ..., n_{ip}$. On obtient

$$\begin{aligned} \mathrm{SRC}\left(x_i, (Ax)_j\right) &= \frac{A_{ji} \hat{\sigma}(x_i)}{\hat{\sigma}((Ax)_j)} \\ &= \frac{A_{ji} \hat{\sigma}(x_i)}{\sqrt{\sum_{k=1}^{n_{ip}} \sum_{l=1}^{n_{ip}} A_{jk} A_{jl} cor(x_k, x_l) \hat{\sigma}(x_k) \hat{\sigma}(x_l)}}. \end{aligned} \tag{3.20}$$

On peut également décomposer

$$cor\left(x_i,(Ax)_j\right)=\sum_{k=1}^{n_{ip}} cor(x_i,x_k)\mathrm{SRC}\left(x_k,(Ax)_j\right). \tag{3.21}$$

Le coefficient de régression de rang standardisé (SRRC) est le SRC calculé pour les rangs dans les échantillons :

$$\mathrm{SRRC}(x_i,y_j)=\mathrm{SRC}\left(r_{x_i},r_{y_j}\right). \tag{3.22}$$

Plus les dépendances entre les paramètres de départ sont fortes, plus il est difficile d'isoler l'influence de F.

3.2 Analyses déterministes

Nous allons maintenant nous intéresser aux méthodes d'analyse de sensibilité déterministes, locales, plus difficile à mettre en œuvre, mais plus économiques en temps de calcul.

3.2.1 Analyse d'incertitude locale

L'analyse d'incertitude déterministe, locale, est basée sur une approximation du premier ordre : on considère le modèle linéarisé

$$\mathrm{d}y=F'\left(x\right)\mathrm{d}x,$$

qui associe des variations approchées des valeurs d'arrivée $\mathrm{d}y$ à des petites variations des paramètres de départ $\mathrm{d}x$ autour d'un jeu de paramètres fixé x. Dans le cas où F est affine, un majorant de l'incertitude sur les sorties y_i, $i=1,...,n_{op}$, est

$$\sum_{j=1}^{n_{ip}}\left|F'(x)_{ij}\right|\left|\Delta x_j\right|$$

où la valeur positive $|\Delta x_j|$ quantifie l'incertitude sur le paramètre de départ x_j. Les dépendances entre les paramètres de départ ne sont pas prises en compte dans cette majoration. On peut les prendre en compte par un changement de variable amenant à considérer des paramètres de départ indépendants (on réduit ainsi le majorant). Des coefficients de corrélation ou de régression pour le problème linéarisé peuvent également être calculés, sans appliquer le modèle F à tous les éléments d'un échantillon de départ, et en remplaçant éventuellement les moments empiriques par les moments théoriques, selon les égalités (3.8), (3.16) et (3.20). Dans le cas général (non linéaire), l'arithmétique d'intervalle peut être utilisée pour évaluer les incertitudes sous la forme d'intervalles contenant l'image par F d'intervalles de départ (voir [68]).

3.2.2 Analyse de sensibilité

L'analyse de sensibilité déterministe, analyse locale, est basée sur la décomposition en valeurs singulières (SVD) de la matrice Jacobienne $F'(x)$ pour un paramètre de départ fixé x.

La notion de valeurs singulières est une généralisation pour les matrices quelconques de la notion de valeurs propres pour les matrices symétriques positives. La SVD de la matrice rectangulaire $F'(x)$ est donnée par

$$F'(x) = USV^T$$

où U et V sont des matrices orthogonales et S est une matrice diagonale de la même taille que $F'(x)$. Les colonnes de V sont les vecteurs singuliers dans l'espace de départ et les colonnes de U sont les vecteurs singuliers dans l'espace d'arrivée. Les termes diagonaux de S sont les valeurs singulières de $F'(x)$, c'est-à-dire les racines carrées des valeurs propres de la matrice carrée symétrique positive $F'(x)^T F'(x)$. Ce sont donc des valeurs positives. Elles sont ordonnées par ordre décroissant. Cette décomposition existe toujours et la matrice des valeurs singulières S est déterminée de manière unique. On a unicité de la décomposition USV^T à une multiplication éventuelle de certains vecteurs singuliers par -1 près dans le cas où les valeurs singulières sont distinctes (voir par exemple [28], p. 17).

La SVD réalise un changement de base orthogonal simultanément dans les espaces de départ et d'arrivée. Si on note u_k (respectivement v_k) la $k^{ème}$ colonne de U (respectivement de V) et s_k le $k^{ème}$ terme diagonal de S, on a

- Pour $k \leq \min(n_{ip}, n_{op})$

$$F'(x)\, v_k = s_k u_k, \tag{3.23}$$

  c'est-à-dire que la variation des valeurs d'arrivée dans la direction u_k dépend exclusivement de la variation des paramètres de départ dans la direction v_k, et cette dépendance est quantifiée par la valeur singulière s_k (éventuellement nulle).

Si F' n'est pas carrée alors l'un des deux autres cas suivants peut aussi se présenter :

- Si $n_{op} < n_{ip}$ alors on a, pour $n_{op} < k \leq n_{ip}$, $F'(x)\, v_k = \mathbf{0}$, c'est-à-dire que la direction v_k dans l'espace des paramètres de départ n'a pas d'influence sur les variables d'arrivée, elle se trouve dans le noyau de $F'(x)$. Par conséquent, si on écrit une variation des paramètres de départ $\mathrm{d}x$ comme une combinaison linéaire des éléments de la base, $\mathrm{d}x = \sum_{k=1}^{n_{ip}} \alpha_k v_k$, la variation des variables d'arrivée associée est $\mathrm{d}y = \sum_{k=1}^{n_{op}} s_k \alpha_k u_k$.
- Si $n_{ip} < n_{op}$ alors, pour $n_{ip} < k \leq n_{op}$, les variables de sortie sont invariantes dans la direction u_k, qui se trouve dans le complémentaire orthogonal de l'image de $F'(x)$. Par conséquent, pour une variation de départ $\mathrm{d}x = \sum_{k=1}^{n_{ip}} \alpha_k v_k$, la variation des variables

de sorties associée est $\mathrm{d}y = \sum_{k=1}^{n_{ip}} s_k \alpha_k u_k$.

Ainsi, le taux de décroissance des valeurs singulières fournit une classification hiérarchique de l'influence des directions v_k dans l'espace de départ sur les directions u_k dans l'espace d'arrivée.

La matrice Jacobienne et les résultats de sa décomposition en valeurs singulières peuvent être utilisés, par exemple, pour décider quels paramètres de départ doivent être le mieux explorés, en fonction des variables d'arrivée sur lesquelles on veut diminuer l'incertitude. Dans un autre contexte, lorsque les variables d'arrivée peuvent être mesurées et doivent être contrôlées, les résultats de SVD permettent parfois d'identifier facilement quels paramètres de départ doivent être modifiés, et quelles dépendances sur ces modifications doivent être respectées, pour modifier une variable de sortie particulière sans perturber les autres. Ces possibilités seront illustrées dans la section 7.1.7.4.

La SVD condense en quelques relations linéaires la totalité des informations contenues dans la matrice Jacobienne F'. Elle fournit pour le problème linéarisé une liste exhaustive et la plus courte possible des dépendances entre variables de départ et variables d'arrivée dues directement au problème (et pas à des dépendances entre variables de départ).

3.3 SVD et analyses statistiques

Nous allons d'abord dans cette section écrire la SVD utilisée en analyse déterministe sous une forme ressemblant aux décompositions utilisées en analyse statistique, afin d'amorcer une comparaison et un rapprochement entre les décompositions utilisées dans les analyses déterministes et statistiques. Nous allons présenter quelques exemples de décompositions de variables aléatoires utilisées en statistiques faisant appel à des décompositions en valeur singulière. Nous présenterons un cas particulier dans lequel les décompositions utilisées dans les deux domaines sont strictement équivalentes. Enfin nous verrons comment la SVD permet de condenser les relations de dépendances obtenues par une analyse statistique, comme c'est fait pour les analyses déterministes.

3.3.1 Décomposition de variables

Analyse déterministe. Dans l'analyse de sensibilité déterministe locale, on s'intéresse aux variations linéarisées des valeurs d'arrivée d'une fonction F :

$$\mathrm{d}y = F'(x)\mathrm{d}x$$

dépendant de n_{ip} paramètres $x_1, ..., x_{n_{ip}}$. La décomposition en valeurs singulières de $F'(x)$ conduit en particulier à décomposer $\mathrm{d}y$ sur la base orthogonale $U = [u_1, ..., u_{n_{op}}]$ sous la forme

$$\mathrm{d}y = \sum_{k=1}^{\min(n_{ip},n_{op})} s_k \alpha_k u_k = US\alpha, \tag{3.24}$$

si $\mathrm{d}x$ est décomposé sur la base $V = [v_1, ..., v_{n_{ip}}]$ sous la forme

$$\mathrm{d}x = \sum_{k=1}^{n_{ip}} \alpha_k v_k. \tag{3.25}$$

(Comme V est orthogonale, $\alpha = V^T \mathrm{d}x$.) Les valeurs singulières $s_1 \geq ... \geq s_{\min(n_{ip},n_{op})} \geq 0$ sont rangées par ordre décroissant de sorte que l'on obtient une approximation de $y = F(x + \mathrm{d}x)$ en tronquant cette somme :

$$l < \min(n_{ip}, n_{op}), \quad y \approx F(x) + \sum_{k=1}^{l} s_k \alpha_k u_k.$$

Remarque 1 *Nous notons ici $y = F(x + \mathrm{d}x)$ pour pouvoir faire le lien avec les décompositions statistiques qui suivront, et pas $y + \mathrm{d}y = F(x + \mathrm{d}x)$, comme c'est souvent fait en calcul différentiel.*

Décomposition propre orthogonale. En analyse de données probabiliste, par l'utilisation des décompositions propres orthogonales, on cherche à décomposer le vecteur aléatoire Y sur une base orthogonale de vecteurs constants, *« de telle sorte que les échantillons dans l'espace d'échantillonnage puissent être exprimés de manière optimale en utilisant les l premiers vecteurs de base »* ([57]). L'optimalité signifie que la variance de Y est concentrée dans les premiers termes de la décomposition. Plus formellement, si $\Phi = [\phi_1, ..., \phi_{n_{op}}]$ est une base orthogonale non aléatoire de $\mathbb{R}^{n_{op}}$,

$$Y = \sum_{k=1}^{n_{op}} Z_k \phi_k = \Phi Z$$

où pour $k = 1, ..., n_{op}$, $Z_k = \phi_k^T Y$. Les variables $Z_1, ..., Z_{n_{op}}$ sont des variables aléatoires. Pour $l \leq n_{op}$ et pour $k_1, ..., k_l$ des entiers arbitraires distincts de $1, ..., n_{op}$, on définit

$$Y(l) = \sum_{k=1}^{l} Z_k \phi_k \tag{3.26}$$

et

$$\hat{Y}(l) = \sum_{k=k_1,...,k_l} Z_k \phi_k.$$

Dans le cas où $E(Y) = \mathbf{0}$, on a également $E(Z_k) = 0, E(Y(l)) = \mathbf{0}$ et $E(\hat{Y}(l)) = \mathbf{0}$, et on souhaite que

$$E\left(\|Y - Y(l)\|^2\right) \leq E\left(\left\|Y - \hat{Y}(l)\right\|^2\right).$$

On fait le lien avec les égalités (3.24) et (3.25) en remplaçant Φ par U et Z_k par $s_k \alpha_k = u_k^T y$.

La matrice analysée en analyse probabiliste est souvent la matrice de covariance

$$\Sigma_Y = E\left([Y - E(Y)]\,[Y - E(Y)]^T\right),$$

ou sa version empirique. On fera le lien avec l'analyse de $F'F'^T = USS^TU^T$ considérée dans l'analyse déterministe.

Nous allons donc présenter quelques exemples de décompositions propres orthogonales. Des décompositions sont aussi présentées dans [26, 18] et une application dans [63]. Dans les sections 3.3.2 à 3.3.4, il ne sera pas question de modèle mathématique F.

3.3.2 Utilisation de la SVD en analyse statistique

On résume ici l'analyse utilisée dans l'article [57].

Elle est définie en termes **déterministes**, c'est-à-dire pour un échantillon donné.

On considère un échantillon y de taille n du vecteur aléatoire Y

$$y = [y_1, ..., y_n],\ \text{pour } i = 1, ..., n,\ y_i \in \mathbb{R}^{n_{op}}.$$

Soit $\Phi = [\phi_1, ..., \phi_{n_{op}}]$ une base arbitraire de $\mathbb{R}^{n_{op}}$. On veut approcher y et Y en conservant les l premiers termes de la décomposition suivant la base Φ. On considère l'erreur sur l'échantillon ($y_1, ..., y_n$ sont fixés)

$$\epsilon^2(l) = \sum_{i=1}^{n} \|y_i - y_i(l)\|^2,$$

où $y_i(l)$ est défini similairement à l'égalité (3.26) :

$$y_i(l) = \sum_{k=1}^{l} z_k \phi_k,\ \text{avec, pour } k = 1, ..., l,\ z_k = \phi_k^T y_i. \tag{3.27}$$

Dans [57] il est vérifié (égalité 47) que

$$\epsilon^2(l) = \left\|y^T \Phi_{n_{op}-l}\right\|_F^2,$$

où $\Phi_{n_{op}-l} = [\phi_{l+1}, ..., \phi_{n_{op}}]$ et $\|.\|_F$ est la norme de Frobenius : $\|(a_1, ..., a_n)^T\|_F^2 = \|a_1\|^2 + ... + \|a_n\|^2$. Trouver les vecteurs de base optimaux revient à résoudre le problème de minimisation

$$\min_{\phi_j} \epsilon^2(l) \text{ tels que } \phi_i^T \phi_j = \delta_{ij}.$$

Dans [57] il est montré, en utilisant un Lagrangien, que l'optimalité est atteinte quand la base choisie est la matrice de vecteurs singuliers d'arrivée de y et que l'erreur minimum pour la troncature à l termes est la somme des carrés des $n_{op} - l$ dernières valeurs singulières

de y : si $yy^T = U\Lambda U^T$ où U est une matrice orthogonale et Λ est une matrice diagonale dont les termes diagonaux sont positifs et ordonnés dans l'ordre décroissant,

$$\Phi_{opt} = U = [u_1, ..., u_{n_{op}}]$$

et

$$\epsilon^2_{opt}(l) = \sum_{k=l+1}^{n_{op}} \lambda_k.$$

3.3.3 Transformation de Karhunen-Loève (KLT)

Cette transformation est utilisée pour la compression de données.

3.3.3.1 Version continue

Développement donné dans [23]. On considère un processus aléatoire du second ordre $Y(t)$. On peut décomposer Y sous la forme

$$Y(t) = \sum_{k=1}^{+\infty} (s_k V_k) u_k(t)$$

où s_k, $k = 1, ...$ sont des constantes positives telles que $\sum_{k=1}^{+\infty} s_k^2 < +\infty$, V_k, $k = 1, ...$ sont des variables aléatoires indépendantes telles que $E(V_k V_l) = \delta_{kl}$ et u_k, $k = 1, ...$ sont des fonctions scalaires continues telles que $\int_T u_k(t) u_l(t) \mathrm{d}t = \delta_{kl}$.

3.3.3.2 Version discrète

Développement donné dans [23]. Si Y_i, $i = 1, ..., n$, sont des vecteurs aléatoires discrets, on peut décomposer $Y = [Y_1, ..., Y_n]$ sous la forme

$$Y_i = \sum_{k=1}^{n_{op}} (s_k V_k) u_{k_i}$$

où s_k, $k = 1, ..., n_{op}$ sont des constantes positives, V_k, $k = 1, ..., n_{op}$ sont des vecteurs aléatoires indépendants tels que $E(V_k^T V_l) = \delta_{kl}$ et u_k, $k = 1, ..., n_{op}$ sont des vecteurs constants tels que $u_k^T u_l = \delta_{kl}$.

Définition (comme dans [57]). On se place dans le cas $n = 1$. Un vecteur aléatoire Y dans $\mathbb{R}^{n_{op}}$ se décompose selon une base orthogonale $\Phi = [\phi_1, ..., \phi_{n_{op}}]$ de vecteurs constants sous la forme

$$Y = \sum_{k=1}^{n_{op}} Z_k \phi_k = \Phi Z$$

où Z_k, $k = 1, ..., n_{op}$ sont des variables aléatoires et $Z = (Z_1, ..., Z_{n_{op}})^T$. On a $Z_k = \phi_k^T Y$. Soit $l \leq n_{op}$, $b_{l+1}, ..., b_{n_{op}}$ des constantes et

$$Y(l) = \sum_{k=1}^{l} Z_k \phi_k + \sum_{k=l+1}^{n_{op}} b_k \phi_k.$$

Les vecteurs $Y(l)$ et $\Delta Y(l) = Y - Y(l)$ sont des vecteurs aléatoires. Pour que $Y(l)$ soit une bonne approximation de Y, on cherche à minimiser

$$\epsilon^2(l) = E\left(\|\Delta Y(l)\|^2\right) = E\left(\sum_{k=l+1}^{m} (Z_k - b_k)^2\right).$$

On veut résoudre le problème de minimisation

$$\min_{\phi_k} \epsilon^2(l) \text{ tel que } \phi_k^T \phi_{k_1} = \delta_{kk_1}.$$

Il est immédiat que pour ϕ_k, $k = 1, ..., n_{op}$ fixés, on minimise $\epsilon^2(l)$ en prenant $b_k = E(Z_k)$. Il est montré dans [57] que **des** ϕ_i optimaux sont les vecteurs propres de Σ_Y , dans l'ordre décroissant des valeurs propres associées, et que le $\epsilon^2(l)$ correspondant est la somme des $n_{op} - l$ plus petites valeurs propres (positives) de Σ_Y. On peut écrire les Z_k correspondant sous la forme $Z_{k_{opt}} = s_k V_k$ où s_k est la racine carrée de la $k^{\text{ème}}$ valeur propre de Σ_Y et $E(V_k^2) = 1$.

Relation avec la SVD. *« The discrete Karhunen-Loève expansion is algebraically equivalent to the singular-value decomposition of a rectangular matrix »* ([23]). Dans cet article, l'équivalence entre ces décompositions est établie par l'intermédiaire de la décomposition des fonctions continues de 2 variables par séparation des variables :

1. Décomposition en valeurs singulières d'une matrice y :

$$y_{ij} = \sum_{k=1}^{\min(m,n)} S_{kk} U_{ik} V_{jk}$$

   où U et V sont des matrices orthogonales et S une matrice diagonale de la taille de y.

2. Séparation des variables d'une fonction continue de 2 variables :

$$y(t, w) = \sum_{k=1}^{+\infty} s_k u_k(t) v_k(w)$$

   où $\int_t u_k(t) u_l(t) \mathrm{d}t = \delta_{kl}$ et $\int_w v_k(w) v_l(w) \mathrm{d}w = \delta_{kl}$.

3. K-L expansion pour un processus aléatoire $Y(t)$:

$$Y(t) = \sum_{k=1}^{+\infty} s_k V_k u_k(t)$$

où $E(V_k V_l) = \delta_{kl}$ et $\int_t u_k(t) u_l(t) \mathrm{d}t = \delta_{kl}$.

4. K-L expansion pour un processus aléatoire discret $Y = (Y_1, ..., Y_n)$:

$$Y_i = \sum_{k=1}^{n_{op}} s_k \left(u_k\right)_i V_k$$

où $u_k^T u_l = \delta_{kl}$ et $E(V_k^T V_l) = \delta_{kl}$.

3.3.4 Analyse en Composantes Principales (PCA)

Il s'agit d'une technique de réduction de dimension en analyse statistique multivariable.

Définition de [57]. L'objectif est de réduire la dimension d'un jeu de données constitué d'un grand nombre de variables dépendantes, en conservant le plus possible les variations du jeu de données. Ceci est réalisé par changement de variables. Les nouvelles variables sont indépendantes et les premières d'entre elles conservent la plus grande partie des variations des variables originales.

Y est toujours un vecteur aléatoire de $\mathbb{R}^{n_{op}}$. Ses composantes principales Z_i, $i = 1, ..., n_{op}$ s'écrivent sous la forme

$$Z_i = \sum_{k=1}^{n_{op}} \alpha_{i_k} Y_k = \alpha_i^T Y$$

où les α_i sont des vecteurs constants. La variance de Z_i est

$$s_i^2 = \alpha_i^T \Sigma_Y \alpha_i$$

où Σ_Y est la matrice de covariance de Y. On impose une normalisation : $\alpha_i^T \alpha_i = 1$. Pour que les Z_i soient indépendants il faut également que $\alpha_i^T \alpha_j = 0$ pour $i \neq j$.

On maximise successivement les variances des Z_i, $i = 1, 2, ..., n_{op}$, si α_i, $i = 1, ..., n_{op}$ sont les vecteurs propres de Σ_Y correspondant aux valeurs propres rangées dans l'ordre décroissant. Ce choix minimise également $\epsilon^2(l) = E\left(\left\|Y - \sum_{k=1}^{l} Z_i \alpha_i\right\|^2\right)$.

3.3.5 Cas linéaire avec entrées indépendantes

On suppose que F est une fonction linéaire, dont les paramètres de départ $X_1, ..., X_{n_{ip}}$ sont des variables aléatoires indépendantes centrées telles que $E(X_k X_l) = \delta_{k,l}$, i.e. $E(XX^T) =$

I_n. On considère le vecteur aléatoire $Y = F(X)$, combinaison linéaire de $X_1, ..., X_{n_{ip}}$. La matrice Jacobienne F' de F est constante avec $F' = F$. Soit $F' = USV^T$ sa SVD. Y se décompose de manière exacte sous la forme

$$Y = \sum_{k=1}^{n} s_k \alpha_k u_k = US\alpha$$

où $\alpha = V^T X$. On a $\alpha\alpha^T = V^T X X^T V$ et $E(\alpha\alpha^T) = V^T E(XX^T)V = I_n$. La matrice de covariance Σ_Y de Y vérifie

$$\begin{aligned}
\Sigma_Y &= E(YY^T) \\
&= E(US\alpha\alpha^T S^T U^T) \\
&= USE(\alpha\alpha^T)S^T U^T \\
&= USS^T U^T \\
&= F'F'^T.
\end{aligned}$$

L'analyse de sensibilité de F par décomposition en valeurs singulières de sa matrice Jacobienne est ici globale et est équivalente à la décomposition de Y par KLT et PCA. On a également

$$\Sigma_{Y,X} = E(YX^T) = E(US\alpha\alpha^T V^T) = USV^T = F'.$$

On peut rapprocher ce résultat de l'égalité (3.8), qui se simplifie dans notre cas particulier :

$$cor(x_i, y_j) = \frac{A_{ji}}{\sqrt{\sum_{k=1}^{n_{ip}} A_{jk}^2}} = \frac{(\Sigma_{Y,X})_{ji}}{\sqrt{\sum_{k=1}^{n_{ip}} (\Sigma_{Y,X})_{jk}^2}}.$$

3.3.6 Extension au cas non linéaire avec entrées dépendantes

Dans le cas où F est linéaire ou affine et où ses paramètres d'entrée ne sont pas indépendants, on a

$$\Sigma_{F(X),X} = F'\Sigma_{X,X}. \tag{3.28}$$

Dans le cas général, on peut définir

$$G = \Sigma_{F(X),X} \left(\Sigma_{X,X}\right)^{-1}. \tag{3.29}$$

On peut décomposer

$$Cov\left(X_i, (F(X))_j\right) = G_{ji} Cov(X_i, X_i) + \sum_{k=1, k\neq i}^{n_{ip}} G_{jk} Cov(X_i, X_k) \tag{3.30}$$

et si on décompose G en valeurs singulières sous la forme $G = USV^T$ on obtient, pour $i = 1, ..., n_{ip}$, $k = 1, ..., \min(n_{ip}, n_{op})$,

$$\sum_{j=1}^{n_{op}} (u_k)_j \, Cov(X_i, (F(X))_j) = \sum_{j=1}^{n_{ip}} s_k \, (v_k)_l \, Cov(X_i, X_j). \tag{3.31}$$

c'est-à-dire

$$Cov\left(X_i, \sum_{j=1}^{n_{op}} (u_k)_j \, (F(X))_j\right) = s_k Cov\left(X_i, \sum_{j=1}^{n_{ip}} (v_k)_j \, X_j\right). \tag{3.32}$$

Pour $n_{ip} < k \leq n_{op}$, $Cov\left(X_i, \sum_{j=1}^{n_{op}} (u_k)_j \, (F(X))_j\right) = 0$.

Enfin on effectue les changements de variable, pour $k = 1, ..., \min(n_{ip}, n_{op})$,

$$\tilde{X}_k = \sum_{i=1}^{n_{ip}} (v_k)_i \, X_i \tag{3.33}$$

et

$$\tilde{Y}_k = \sum_{j=1}^{n_{op}} (u_k)_j \, (F(x))_j \, . \tag{3.34}$$

Alors on obtient à partir de l'égalité (3.32), comme la base v est orthogonale,

$$Cov(\tilde{X}_k, \tilde{Y}_k) = s_k Cov(\tilde{X}_k, \tilde{X}_k) \tag{3.35}$$

et pour $k \neq l$,

$$Cov(\tilde{X}_k, \tilde{Y}_l) = s_k Cov\left(\tilde{X}_k, \tilde{X}_l\right) = s_k \sum_{i=1}^{n_{ip}} \sum_{j=1, j\neq i}^{n_{ip}} (v_k)_i \, (v_l)_j \, Cov\left(X_i, X_j\right) + s_k \sum_{i=1}^{n_{ip}} (v_k)_i \, (v_l)_i \, Cov\left(X_i, X_i\right). \tag{3.36}$$

En particulier, si les variables d'entrée sont indépendantes et normalisées, alors $Cov(\tilde{X}_k, \tilde{X}_l) = 0$ et $Cov(\tilde{X}_k, \tilde{Y}_l) = 0$ pour $k \neq l$.

Dans la pratique, on peut utiliser les covariances empiriques et écrire des relations similaires pour les coefficients de corrélation de Pearson :

– Pour $k = 1, ..., \min(n_{ip}, n_{op})$

$$\hat{\sigma}(\tilde{y}_k) cor(\tilde{x}_k, \tilde{y}_k) = s_k \hat{\sigma}(\tilde{x}_k). \tag{3.37}$$

– Pour $k = 1, ..., \min(n_{ip}, n_{op})$, $l = 1, ..., \min(n_{ip}, n_{op})$, $k \neq l$

$$\hat{\sigma}(\tilde{y}_l) cor(\tilde{x}_k, \tilde{y}_l) = s_k \hat{\sigma}(\tilde{x}_l) cor(\tilde{x}_k, \tilde{x}_l). \tag{3.38}$$

– Pour $i = 1, ..., n_{ip}$, $k = n_{ip} + 1, ..., n_{op}$,

$$cor(x_i, \tilde{y}_k) = 0. \tag{3.39}$$

On n'a pas construit de matrice de covariance comme dans le cas linéaire, mais elle s'estime.

Dans le cas où les variables d'entrées sont indépendantes et normalisées, on a fourni un changement sur les variables d'entrée et de sortie tel que les nouvelles variables d'entrée soient aussi indépendantes, et tel que chaque nouvelle variable de sortie soit corrélée avec au plus une nouvelle variable d'entrée.

Chapitre 4

Calcul de dérivées

L'analyse de sensibilité déterministe, basée sur la décomposition en valeurs singulières d'une matrice Jacobienne, demande donc le calcul de toutes les dérivée partielles d'un modèle donné. Nous allons expliquer dans ce chapitre comment mettre en place la dérivation d'une fonction calculée par un code `C++` par différences divisées, par différentiation automatique et « à la main », en mode direct ou en mode inverse, comparer les performances de ces méthodes et voir comment les faire cohabiter dans un même programme.

4.1 Différences divisées

C'est une méthode simple à mettre en place, et la seule possible lorsque les codes à différentier sont des boîtes noires. Nous allons voir qu'elle doit être utilisée avec précaution.

4.1.1 Utilisation en vérification

Soit F une application définie sur $\mathbb{R}^{n_{ip}}$ à valeurs dans $\mathbb{R}^{n_{op}}$. Pour $y \in \mathbb{R}^{n_{op}}$ on définit

$$\begin{array}{rcl} F_y : \mathbb{R}^{n_{op}} & \rightarrow & \mathbb{R} \\ x & \mapsto & \langle y, F(x) \rangle. \end{array}$$

On a $\vec{\nabla} F_y(x) = F'(x)^T y$. Soit A le résultat d'un calcul de la matrice Jacobienne de F en x. Il faut vérifier par exemple que, pour toute direction d de $\mathbb{R}^{n_{ip}}$ et y de $\mathbb{R}^{n_{op}}$,

$$\lim_{h \to 0} \frac{F_y(x + hd) - F_y(x)}{h} = \langle y, Ad \rangle.$$

(On pourrait aussi considérer des différences divisées centrées ou décentrées à gauche.) On calcule $\langle y, Ad \rangle$ et $\frac{F_y(x+hd)-F_y(x)}{h}$ pour des valeurs de h de plus en plus faibles. Le nombre de décimales communes entre les deux quantités est d'abord croissant (on s'approche de la limite) puis décroissant (h est trop faible étant donnée la précision du calcul). En général, un maximum de 6 à 8 décimales communes en double précision est bon signe. Un pas supplémentaire consiste à vérifier que ce nombre de décimales augmente significativement

si l'on passe à la quadruple précision, mais tous les compilateurs ne le permettent pas automatiquement.

Soit $h_1, h_2, \ldots, h_N$ des réels strictement positifs, par exemple $h_1, \ldots, h_N = 10^{-1}, \ldots, 10^{-N}$. Pour $y \in \mathbb{R}^{n_{op}}$ et $d \in \mathbb{R}^{n_{ip}}$, on note

$$\mathcal{R}(y, d) = \left(\frac{F_y(x + h_n d) - F_y(x)}{h_n} \right)_{n=1,\ldots,N}.$$

Soit $(y_i)_{i \in \{1,\ldots,n_{op}\}}$ la base canonique de $\mathbb{R}^{n_{op}}$ et $(d_i)_{i \in \{1,\ldots,n_{ip}\}}$ la base canonique de l'espace de départ. En calculant $\mathcal{R}(y_i, d_j)$ pour $i = 1, \ldots, n_{op}$, $j = 1, \ldots, n_{ip}$, on vérifie le calcul de tous les termes de la matrice Jacobienne A. Une autre possibilité est de faire uniquement le calcul de $\mathcal{R}(y_i, A^T y_i)$ pour $i = 1, \ldots, n_{op}$, ce qui revient à donner une importance plus faible à la justesse des petites valeurs de gradient : on vérifie alors en effet les valeurs de $\left\| A^T y_i \right\|^2$ pour $i = 1, \ldots, n_{op}$, c'est à dire les normes des gradients de chaque composante de F.

4.1.2 Illustrations numériques

Pour les exemples présentés sur les Figures 4.1 (resp. 4.2), on observe un maximum de 10 (resp. 8) décimales communes entre le calcul du taux d'accroissement centré pour la fonction sinus en 1. (resp. en 1.57) et le calcul de dérivée cos(1) (resp. cos(1.57)). Pour les taux décentrés à gauche ou à droite en 1. et en 1.57, ce nombre de décimales est plus faible et est atteint pour une valeur plus faible de h.

Si on veut utiliser les taux d'accroissement pour évaluer une dérivée, et pas pour la vérifier, on ne dispose que des images de la première ligne de chaque figure. Les courbes présentent à chaque fois un palier, mais il est difficile de savoir quel point de ce palier est optimal. De plus, il est difficile de prévoir où se trouvera ce palier sans avoir calculé plusieurs taux d'accroissement. On note que ces courbes rappellent beaucoup celles qui servent à définir des volumes représentatifs élémentaires, comme celle de la Figure 1.1.

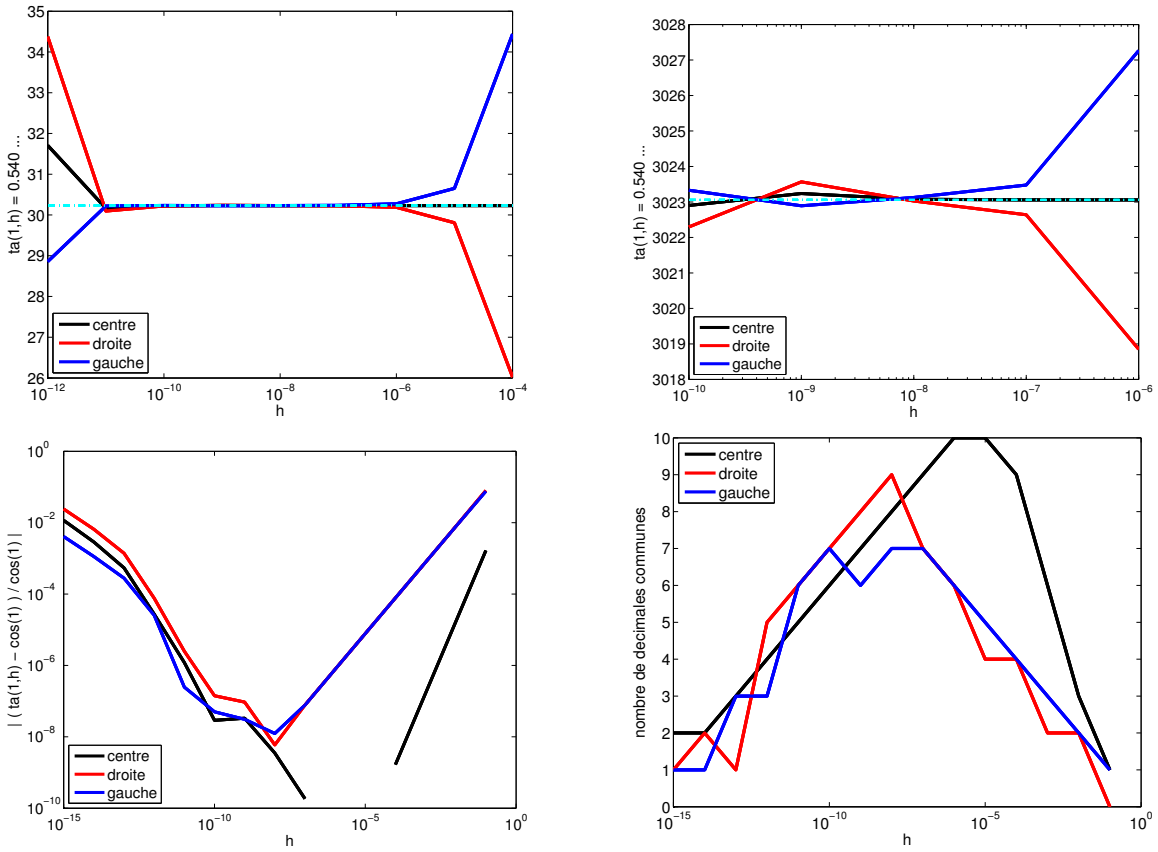


FIG. 4.1 – En haut : taux d'accroissement (ta) calculé en double précision en fonction du pas des différences divisées, pour la fonction sinus, autour du point $(1, \sin(1))$. L'axe des ordonnées donne les chiffres après la virgule à partir du quatrième, i.e. pour la figure de gauche $10^5\,(\mathrm{ta}(1,\mathrm{h}) - 0.540)$ et pour la figure de droite $10^7\,(\mathrm{ta}(1,\mathrm{h}) - 0.540)$. En bas, à gauche : valeur absolue de la différence relative entre le taux d'accroissement calculé et la dérivée. En bas, à droite : nombre de décimales communes entre le taux d'accroissement calculé et la dérivée.

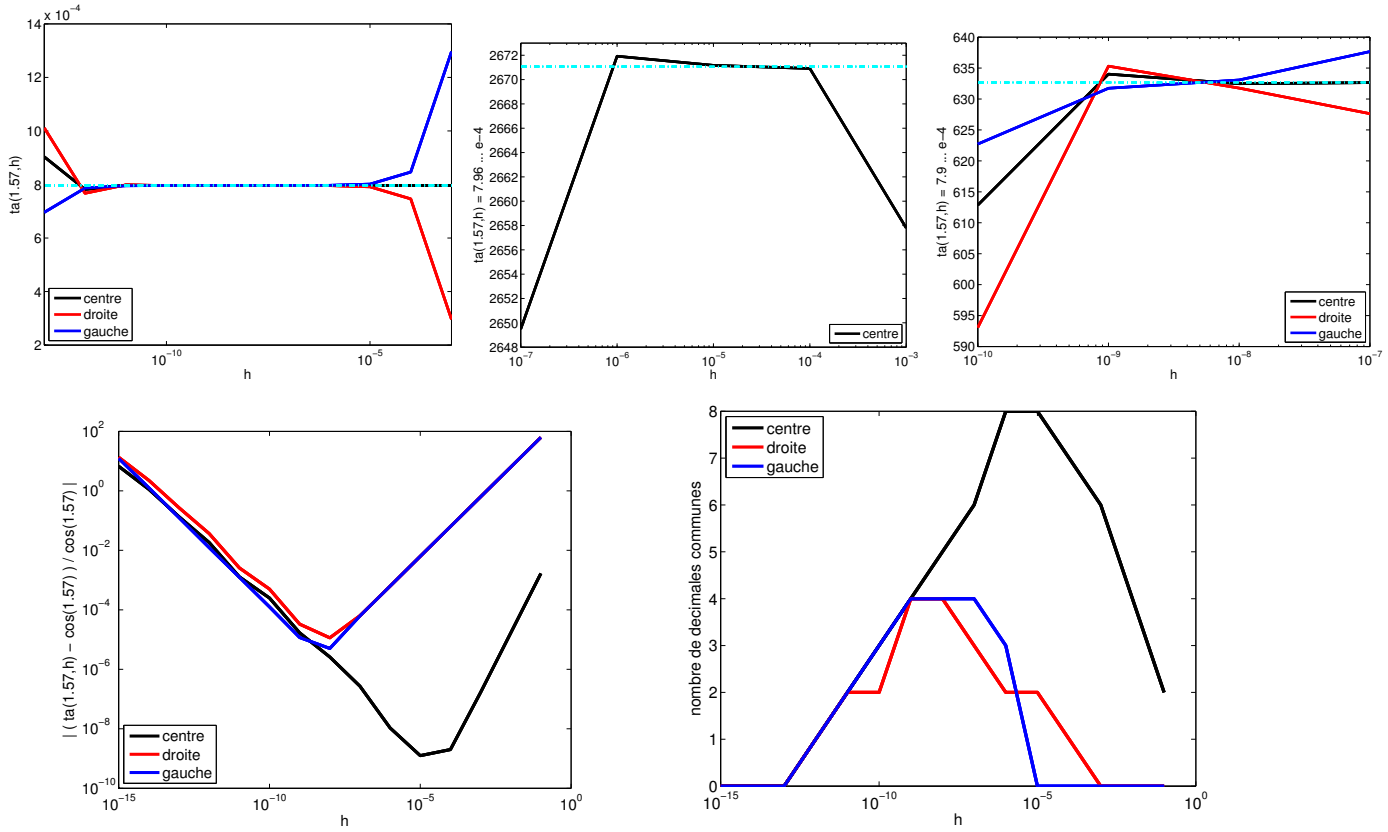


FIG. 4.2 – En haut : taux d'accroissement (ta) calculé en double précision en fonction du pas des différences divisées, pour la fonction sinus, autour du point $(1.57, \sin(1.57))$. Pour la deuxième figure, l'axe des ordonnées donne les chiffres après la virgule à partir du septième, i.e. $10^6\,(\mathrm{ta}(1,\mathrm{h}) - 7.96\mathrm{e}-4)$. Pour la troisième figure, l'axe des ordonnées donne les chiffres après la virgule à partir du sixième, i.e. $10^4\,(\mathrm{ta}(1,\mathrm{h}) - 7.9\mathrm{e}-4)$. En bas, à gauche : valeur absolue de la différence relative entre le taux d'accroissement calculé et la dérivée. En bas, à droite : nombre de décimales communes entre le taux d'accroissement calculé et la dérivée.

4.2 Différentiation automatique

Les outils de différentiation automatique peuvent utiliser soit la transformation de source, soit la surcharge d'opérateur, c'est-à-dire le prolongement de la définition de fonctions ou de routines à d'autres types. En transformation de source, l'utilisateur fournit des fichiers sources à un logiciel, qui fabrique de nouveaux fichiers sources permettant le calcul de la dérivée de la fonction calculée par le programme de départ. Un outil de différentiation automatique par surcharge d'opérateurs est une bibliothèque. L'utilisateur de cette bibliothèque doit modifier légèrement ses sources pour s'y adapter, et la dérivation a lieu pendant l'exécution.

Les exécutions de programmes obtenus par ces deux méthodes sont équivalentes. Dans les deux cas, l'outil est basé sur les dérivations des opérations élémentaires – c'est à dire des opérations binaires $+$, $-$, $\times$, $\div$ et des fonctions « standard » telles que exp, sin, log – et sur les règles de composition (« chain rule », illustration en Figure 4.3). La dérivation est possible en mode direct, dans l'objectif par exemple de calculer une matrice Jacobienne colonne par colonne, et, selon les outils, en mode inverse, pour un calcul de matrice Jacobienne ligne par ligne, ce qui est intéressant lorsqu'il y a moins de lignes que de colonnes.

Une liste d'outils est disponible sur la page [37]. Les outils par surcharge d'opérateur sont plus faciles à implémenter mais plus complexes à utiliser. Étant donné que nous souhaitons pouvoir dériver des programmes `C++` en mode inverse, nous nous sommes intéressés à l'outil de différentiation automatique par surcharge d'opérateurs `ADOL-C`, disponible sur la page [43]. Ces méthodes sont bien détaillées dans les ouvrages [29] et [54]. On pourra également consulter [65].

Notons que l'outil `ADOL-C`, dont nous allons principalement parler ici, permet également la dérivation à des ordres supérieurs à 1 et permet de définir des règles de quadrature, c'est-à-dire d'imposer la manière de dériver certaines fonctions. Nous ne développerons pas ces deux points.

4.2.1 Différentiation automatique par transformation de source

Nous nous appuyons sur l'exemple très simple d'un programme devant calculer et retourner la somme de ses deux arguments.

4.2.1.1 Mode direct

Un exemple est donné dans le Tableau 4.1. Pour chaque variable `x` déclarée dans le code de départ, deux variables sont déclarées dans le code d'arrivée : une « valeur » `x` et sa « variation » `dx`.

Avant chaque opération élémentaire sur des valeurs, on effectue une opération élémentaire sur les variations. L'ordre « opération sur les variations **puis** opération sur les valeurs » est primordial lorsque des noms de variables sont réutilisés. Par exemple le morceau de code suivant ne calcule pas la dérivée souhaitée :

pseudo-code initial	code retourné par le logiciel de différentiation automatique
real x1, x2, v, y ; x1=arg1 ; x2=arg2 ; v=x1+x2 ; y=v ;	real dx1, x1, dx2, x2, dv, v, dy, y ; dx1=darg1 ; x1=arg1 ; dx2=darg2 ; x2=arg2 ; dv=dx1+dx2 ; v=x1+x2 ; dy=dv ; y=v ;

TAB. 4.1 – Différentiation automatique par transformation de source en mode direct.

```
y = x*y
dy = x*dy + y*dx
```

En effet, si y_0 est la valeur initiale de y et y_1 est sa valeur finale, on obtient

$$y_1 = x*y_0$$
$$dy_1 = x*dy_0 + x*y_0*dx.$$

En revanche celui-ci est correct :

```
dy = x*dy + y*dx
y = y + x.
```

Si y_0 est la valeur initiale de y et y_1 est sa valeur finale, on obtient

$$y_1 = x*y_0$$
$$dy_1 = x*dy_0 + y_0*dx.$$

4.2.1.2 Mode inverse

Pseudo-code initial	code retourné par un logiciel en mode inverse
real x1, x2, v, y ; x1=arg1 ; x2=arg2 ; v=x1+x2 ; y=v ;	real g_x1, x1, g_x2, x2, g_v, v, g_y, y ; g_x1=0 ;g_x2=0 ; g_v=0 ; g_y=0 ; x1=arg1 ; x2=arg2 ; v=x1+x2 ; y=v ; g_y=gz ; g_v=g_y ; g_x2 += g_v ; g_x1 += g_v ; g_arg2=g_x2 ; g_arg1=g_x1 ;

TAB. 4.2 – Différentiation automatique par transformation de source en mode inverse.

Le Tableau 4.2 présente le résultat d'une transformation de source en mode inverse. Pour chaque variable x declarée dans le pseudo-code initial, deux variables x et g_x sont déclarées dans le pseudo-code différentié, la variable g_x représentant un gradient par rapport à x. Les gradients sont d'abord initialisés à zéro. Ensuite on effectue les mêmes opérations que

dans le code initial. Ces opérations sont enfin redéroulées à l'envers, chaque opération élémentaire entraînant une opération élémentaire sur un gradient. Les opérations binaires doivent être décomposées :

```
v=x1+x2 ;
```

est équivalent à

```
v=0 ; v+=x1 ; v+=x2 ;
```

4.2.2 Différentiation automatique par surcharge d'opérateur

4.2.2.1 Principe

La bibliothèque de différentiation automatique (`ADOL-C` par exemple) fournit de nouvelles fonctionnalités que l'utilisateur doit intégrer à bon escient dans son code. Il faut déterminer la partie du code à différentier, en définir précisément les entrées et les sorties, et demander explicitement le calcul de dérivation suivant le mode choisi en utilisant la solution appropriée de la bibliothèque. Le programme recalcule alors les valeurs ainsi que les dérivées.

4.2.2.2 Nouvelles notions

Notions théoriques On appelle **variables actives** les variables représentant des quantités par rapport auxquelles on souhaite dériver ou que l'on doit dériver. Elles sont réparties entre

- les variables actives **indépendantes**, non modifiables entre leur initialisation et leur destruction. Elles représentent les arguments de la fonction à différentier.
- les variables **dépendantes** : toutes les autres, c'est-à-dire les variables « intermédiaires » et les valeurs de retour de la fonction à différentier.

Implémentation du mode direct Pour fonctionner en mode direct, une bibliothèque de différentiation automatique par surcharge d'opérateurs définit une classe (ou plus simplement, un type de données) `a`(ctive)`double` qui contient au moins :

- Les attributs ou variables de classe « `x` » et « `dx` », représentant respectivement une valeur et une variation élémentaire. (On peut dire qu'`adouble` est un type produit cartésien, ses attributs sont en gros les facteurs de ce type.)
- Une surcharge des opérateurs binaires (`+, -, *, /`) et une surcharge des fonctions de flottants de la bibliothèque standard mathématique `C` (`sin, ln, pow, ...`), c'est-à-dire que ces opérateurs et fonctions pourront être utilisés avec des `adouble` en argument, et que la bibliothèque de différentiation définit ce que doit être le résultat de l'application de ces opérateurs ou de ces fonctions à des `adouble`.
- Des opérateurs d'assignement et des convertisseurs. La bibliothèque `ADOL-C` définit l'opérateur <<=, dont le premier membre est de type `adouble` et le second membre

de type `double`, qui permet d'indiquer que le premier membre de l'opérateur est une variable indépendante, et qui donne à son attribut « `x` » la valeur du second membre. Elle définit l'opérateur >>=, dont le premier membre est de type `adouble` et le second membre de type `double`, qui donne au second membre la valeur de l'attribut « `x` » du premier membre, et indique qu'on a affaire à une variable de sortie de la fonction à dériver. Un opérateur d'assignement est en général une surcharge de l'opérateur « = ». Un convertisseur permet de fabriquer une variable d'un type à partir d'une variable d'un autre type. Les convertisseurs sont souvent appelés automatiquement (par exemple, si on demande d'additionner un `double x` et un `adouble y`, `x` est d'abord « transformé » localement en `adouble`, puis deux `adouble` sont additionnés). Les deux morceaux de code suivants sont équivalents :

1. ```
   double x=1 ;
   adouble y=1 ;
   adouble z=x+y ;
   ```
2. ```
   double x=1 ;
   adouble y=1 ;
   adouble x2 = x ; // = est l'opérateur d'assignement
   // x2 est défini tel que x2.« x » = x et x2.« dx » = 0
   adouble z=x2+y ;
   ```

Dans `ADOL-C`, les choses se passent en fait un peu différemment. La surcharge d'opérateur entraîne l'enregistrement dans un tableau ou dans un fichier de la liste de toutes les opérations effectuées sur des variables actives dans la partie du code identifiée comme à dériver. Cet enregistrement est utilisé lors de l'appel d'une routine particulière. L'appel d'une routine de calcul de dérivée en mode direct recalcule les valeurs en même temps que les dérivées. En contrepartie, la différentiation peut être demandée, sous certaines conditions (voir la section 4.2.2.2), pour d'autres variables de départ que celles pour lesquelles l'enregistrement a été fabriqué, et en particulier on peut réutiliser le même enregistrement pour plusieurs variables de départ.

Implémentation du mode inverse En mode inverse, il faut garder une trace des opérations effectuées pendant l'exécution. Les éléments suivants sont indispensables pour une différentiation par surcharge d'opérateurs [29], dans `ADOL-C` ils sont invisibles pour l'utilisateur :

- la définition d'une structure de données `trace` et d'un type de données `adouble` où :
  - un élément de `trace` correspond à une opération élémentaire
  - un `adouble` correspond à un indice dans `trace`
  - un élément contient :
    - des attributs v_i (« valeur ») et $\overline{v}_i$ (« gradient par rapport à v_i »),
    - l'opération élémentaire qui a produit v_i
    - les indices des éléments correspondant aux arguments de cette opération élémentaire
- La définition des opérations arithmétiques sur les `adouble`, dont les assignements entre des `adouble`s et leurs initialisations à partir de `double`.

– La définition d'une routine `return_sweep` qui parcourt `trace` à l'envers et calcule les gradients $\overline{v}_i$.
– La définition d'un mécanisme pour extraire la valeur flottante courante d'un `adouble`, pour initialiser les composantes adjointes au début du parcours à l'envers et pour extraire les valeurs des composantes du gradient à la fin.

Dans `ADOL-C`, le tableau en question est stocké en mémoire ou, au-delà d'une certaine taille choisie par l'utilisateur, dans un fichier binaire. On utilise ce tableau également en mode direct. Une des limites de la différentiation automatique serait la taille des fichiers qu'elle génère (voir la section 4.4.2). Des outils existent cependant pour optimiser les choix « enregistrements ou répétition de calculs » [31, 44]).

Valeurs de retour pour les routines `ADOL-C` L'enregistrement est écrit à partir d'une certaine valeur des paramètres d'entrée que nous noterons x_0. Il contient la suite d'instructions élémentaires ayant permis de passer de x_0 à la valeur de retour $y_0 = f(x_0)$.

Dans le programme principal, on peut utiliser directement cet enregistrement en partant d'une nouvelle valeur d'entrée x_1. Parfois, la lecture de l'enregistrement ne peut pas fournir la valeur de $f(x_1)$, en particulier lorsque la routine permettant de calculer f fait intervenir des structures conditionnelles (`if(...)`). L'outil de différentiation automatique doit permettre de repérer ces situations. Il doit permettre également de donner des informations locales sur la régularité de f.

L'appel d'une fonction utilisant un enregistrement retourne un entier indiquant quelles situations critiques ont été rencontrées (régularité locale - validité ou non de la suite d'opérations élémentaires enregistrée pour la nouvelle valeur d'entrée).

Ces situations peuvent se produire dans les cas de calcul d'une valeur absolue, d'un minimum discret ou d'un maximum discret, ou d'un choix d'opération dépendant d'une comparaison.

La bibliothèque `ADOL-C` utilise 6 valeurs de retour ([30]) :

3 : La fonction est localement analytique : en cas de structures conditionnelles, les mêmes branches ont été utilisées que pour le point qui a servi à créer l'enregistrement et on n'a pas atteint de valeur limite i.e. le résultat d'une comparaison n'est jamais une égalité et une valeur absolue n'est jamais nulle.

2 : La fonction est localement analytique mais les choix de min ou max peuvent avoir varié selon la valeur d'entrée x_0 ou x_1.

1 : La fonction est Lipschitzienne mais éventuellement non différentiable en x : on a rencontré une valeur absolue nulle ou on s'est trouvé dans un cas limite pour la détermination d'un minimum ou d'un maximum discret.

0 : La fonction peut être discontinue : on s'est trouvé dans un cas limite pour des comparaisons faisant intervenir des variables actives, i.e. égalité.

-1 : La suite d'instructions enregistrée n'est pas valable au point où l'on a tenté de différentier (structure `if`).

-2 : Utilisation illicite d'une règle de quadrature

La bibliothèque `ADOL-C` définit l'opération `condassign`, qui permet d'enregistrer différentes

opérations à effectuer selon les valeurs de leurs arguments, ce qui permet de passer de la valeur de retour 2 à la valeur de retour -1. L'opération `condassign` est calquée sur la structure du langage `C` « ...? ... : ... ». Les trois lignes de code suivantes sont presque équivalentes :

1. `if(a>0) x=b ; else x=c ;`
2. `x = (a>0)? b : c ;`
3. `condassign(x,a,b,c).`

Pour les deux premières lignes, un enregistrement écrit avec une valeur d'entrée conduisant à une valeur de `a` positive ne sera pas valable en relecture pour une valeur d'entrée conduisant à une valeur de `a` strictement négative. Si on utilise la troisième ligne, l'enregistrement restera valable quelle que soit la valeur de `a`.

Un événement impliquant une valeur de retour strictement négative arrête la lecture de l'enregistrement. Lorsque la valeur de retour d'une routine est strictement négative, on doit refaire l'enregistrement.

4.2.2.3 Exemple

code initial	code adapté par l'utilisateur de la bibliothèque

`double x1, x2, v, y ;`
`x1=arg1 ; x2=arg2 ;`
`v=x1+x2 ;`
`y=v ;`
	`adouble x1, x2, v ; double y ;`
	`x1<<=arg1 ; x2<<=arg2 ;`
	`v=x1+x2 ;`
	`v>>=y ;`

Tab. 4.3 – Différentiation automatique par surcharge d'opérateur.

Nous reprenons l'exemple précédent de la somme de deux arguments flottants. La modification du code source est donnée dans le Tableau 4.3. Nous présentons ici les modifications à effectuer sur le programme principal.

Modification de sources - programme principal [30] La partie du code à différentier est délimitée par des mots-clés. La structure du programme modifié est la suivante :

- Mot clé d'entrée.
- Déclaration des variables actives.
- Initialisation des variables indépendantes : à partir de variables non actives (déclarées avant le mot clé), en utilisant l'opérateur <<=.
- Instructions : inchangées.
- Définition des valeurs de retour, en utilisant l'opérateur >>=.
- Destruction des variables actives.
- Mot clé de sortie.

Il faut bien repérer l'ordre d'affectation des variables indépendantes et l'ordre de définition des variables de sortie, et les dénombrer (le nombre de variables d'entrées doit être donné explicitement, et elles sont indicées automatiquement selon l'ordre de l'utilisation de l'opérateur <<=).

Nous avons vu que la bibliothèque `ADOL-C` utilise un système d'enregistrement : si on exécute le code obtenu après les modifications énumérées ci-dessus, le résultat visible de l'exécution sera le même que pour le code non modifié, mais la liste des opérations impliquant des variables actives effectuées entre le mot clé d'entrée et le mot clé de sortie aura été écrite, sous une forme définie par la bibliothèque, dans un tableau ou un fichier. Pour que la différentiation en mode direct se déroule effectivement, il faut également initialiser les variables indépendantes et leurs variations élémentaires, invoquer une routine (par exemple `forward(...)` pour `ADOL-C`) et demander un post-traitement.

D'autres routines sont disponibles, comme `jac_vec(...)`, dont l'appel revient à initialiser les variables indépendantes pour former un vecteur argument de cette routine puis à appeler la routine `forward`, et `jacobian(...)`, qui revient à appeler successivement la routine `jac_vec` pour chaque vecteur de la base canonique de l'espace d'entrée, pour former une matrice Jacobienne.

Pour le mode adjoint il faut également

- Initialiser les directions adjointes
- Avant extraction des valeurs du gradient, invoquer par exemple la routine `reverse`
- Demander un post-traitement.

D'autres routines que `reverse` sont également disponibles, comme `vec_jac(...)`.

Un exemple de modification de code plus complexe est donné par les Tableaux 6.1 et 6.2.

4.3 Différentiation manuelle

Par « différentiation manuelle », on entend simplement la programmation des dérivées obtenues sur le papier. Elle peut aussi être effectuée en mode direct ou en mode inverse, par exemple en utilisant la méthode de l'état adjoint. Nous allons principalement nous appuyer dans cette section sur l'exemple du problème d'écoulement décrit dans la section 2.1. La méthode de l'état adjoint sera aussi expliquée dans le cas général.

4.3.1 Dérivation en mode direct avec application au problème d'écoulement

Comme on l'a vu dans la section 2.1.3.1, l'équation de Darcy se met sous la forme

$$\mathcal{E}_h(\mathcal{K},\mathcal{U}) = \vec{0} \tag{4.1}$$

où

$$\mathcal{U} = \begin{pmatrix} U \\ P \\ \tilde{L} \end{pmatrix} \; ; \tag{4.2}$$

$$\mathcal{E}_h(\mathcal{K},\mathcal{U}) = \begin{pmatrix} A_{\mathcal{K}}U + B^T P + \tilde{C}^T \tilde{L} - R_1 \\ BU - R_2 \\ \tilde{C}U - R_3 \end{pmatrix}. \tag{4.3}$$

On va considérer que $\mathcal{K}$ est constant sur chaque cellule T_i de la discrétisation $\mathcal{T}_h$, avec $\mathcal{K}_{|T_i} \equiv \mathcal{K}_i$. On note $\mathcal{M}^+_{3\times3}$ l'ensemble des matrices symétriques positives de taille 3×3. La notation $\mathcal{K}$ représentera indifféremment le champs de conductivités hydrauliques ou le vecteur de ${\mathcal{M}^+_{3\times3}}^{N_c}$ dont la $i^{ème}$ composante est égale à $\mathcal{K}_i$. On considère tout d'abord la fonction

$$\begin{array}{rlcl} \tilde{F}: & {\mathcal{M}^+_{3\times3}}^{N_c} & \rightarrow & \mathbb{R}^{n_{dof}} \\ & \mathcal{K} & \mapsto & \mathcal{U} \text{ tel que } \mathcal{E}_h(\mathcal{K},\mathcal{U}) = \vec{0}. \end{array} \tag{4.4}$$

La dérivée de $\tilde{F}$ est obtenue en mode direct en annulant des variations élémentaires de l'opérateur $\mathcal{E}_h$, c'est-à-dire en résolvant des équations de la forme

$$\text{trouver } \mathrm{d}\mathcal{U} \text{ tel que } \frac{\partial \mathcal{E}_h}{\partial \mathcal{K}}\mathrm{d}\mathcal{K} + \frac{\partial \mathcal{E}_h}{\partial \mathcal{U}}\mathrm{d}\mathcal{U} = \vec{0}. \tag{4.5}$$

Intéressons nous au seul terme dépendant de $\mathcal{K}$

$$\frac{\partial\left(A_{\mathcal{K}}U\right)}{\partial \mathcal{K}}\mathrm{d}\mathcal{K}. \tag{4.6}$$

On définit la fonction $\mathcal{A} : {\mathcal{M}^+_{3\times3}}^{N_c} \rightarrow \mathcal{M}_{N\times N}$ par

$$\text{Pour tout } i,j = 1,\ldots,N, \quad \left(\mathcal{A}(\mathcal{R})\right)_{i,j} = \int_\Omega \mathcal{R}\vec{u}_i \cdot \vec{u}_j, \tag{4.7}$$

de sorte que

$$\mathcal{A}(\mathcal{K}^{-1}) = A_{\mathcal{K}}. \tag{4.8}$$

La fonction à intégrer dans l'égalité (4.7) est non nulle sur au plus une cellule T_e. Les intégrales sont calculés par l'emploi règles de quadrature. Pour la cellule e, on note :

- N_q^e : nombre de points de quadrature
- $x_1^e, \ \ldots \ , x_{N_q^e}^e$ sont des points de $\overline{T_e}$
- $P_{e,n}$: poids associé au $n^{\text{ième}}$ point de quadrature
- $\vec{u}_{i,n}^e$: valeur de la fonction de base $\vec{u}_i$ en x_n^e

On a supposé $\mathcal{K}$ uniforme sur chaque élément :

$$\left(A_{\mathcal{K}}\right)_{i,j} = \sum_{n=1}^{N_q^e} P_{e,n}\mathcal{K}_e^{-1}\vec{u}_{i,n} \cdot \vec{u}_{j,n}. \tag{4.9}$$

On note $\vec{w}_{i,n} = \mathcal{K}_e^{-1}\vec{u}_{i,n}$:

$$(A_{\mathcal{K}})_{i,j} = \sum_{n=1}^{N_q^e} P_{e,n}\vec{w}_{i,n} \cdot \vec{u}_{j,n}.$$

Pour tout i, n, on a

$$\mathrm{d}\left(\mathcal{K}_e\vec{w}_{i,n}\right) = \vec{0}.$$

On développe le produit :

$$\left(\mathrm{d}\mathcal{K}_e\right)\vec{w}_{i,n} + \mathcal{K}_e\mathrm{d}\vec{w}_{i,n} = \vec{0}$$

d'où

$$\mathrm{d}\vec{w}_{i,n} = -\mathcal{K}_e^{-1}\left(\mathrm{d}\mathcal{K}_e\right)\vec{w}_{i,n}.$$

En sommant sur n pour $n = 1, \ldots, N_q^e$, on obtient

$$\mathrm{d}\left(A_{\mathcal{K}}\right)_{i,j} = -\left(\mathcal{A}\left(\mathcal{K}^{-1}\left(\mathrm{d}\mathcal{K}\right)\mathcal{K}^{-1}\right)\right)_{i,j}. \tag{4.10}$$

En particulier, dans le cas scalaire $\mathcal{K} \equiv K\mathbf{I}$, on obtient

$$\mathrm{d}\left(A_{\mathcal{K}}\right) = -\mathcal{A}\left(\frac{\mathrm{d}K}{K^2}\mathbf{I}\right) = -\frac{\mathrm{d}K}{K}A_{\mathcal{K}}.$$

On utilise de plus le développement

$$\left(\frac{\partial\left(A_{\mathcal{K}}U\right)}{\partial\mathcal{K}}\mathrm{d}\mathcal{K}\right)_i = \sum_j \mathrm{d}\left(A_{\mathcal{K}}\right)_{i,j}U_j. \tag{4.11}$$

On obtient

$$\begin{pmatrix} A_{\mathcal{K}} & B^T & \tilde{C}^T \\ B & \mathbf{0} & \mathbf{0} \\ \tilde{C} & \mathbf{0} & \mathbf{0} \end{pmatrix}\begin{pmatrix} \mathrm{d}U \\ \mathrm{d}P \\ \mathrm{d}\tilde{L} \end{pmatrix} = \begin{pmatrix} \mathcal{A}\left(\mathcal{K}^{-1}\left(\mathrm{d}\mathcal{K}\right)\mathcal{K}^{-1}\right)U \\ 0 \\ 0 \end{pmatrix}. \tag{4.12}$$

Le premier membre provient de $\frac{\partial\mathcal{E}_h}{\partial\mathcal{U}}\mathrm{d}\mathcal{U}$ et le second membre provient de $\frac{\partial\mathcal{E}_h}{\partial\mathcal{K}}\mathrm{d}\mathcal{K}$. Cette équation correspond à un problème aux limites et est résolue par une méthode similaire à celle utilisée pour le problème direct (voir la section 2.1.3.4). L'équation (4.12) correspond à un problème aux limites, dont les caractéristiques sont résumées dans le Tableau 4.4.

Source	nulle
Conditions aux limites	de Dirichlet sur Γ_D, nulle de Neumann sur Γ_N, nulle
Terme constant	$\mathcal{A}(\mathcal{K}^{-1}\left(\mathrm{d}\mathcal{K}\right)\mathcal{K}^{-1})U$

Tab. 4.4 – Caractéristiques du problème aux limites correspondant au calcul de variations (4.5).

4.3.2 Dérivation en mode inverse

4.3.2.1 Méthode de l'état adjoint

On souhaite évaluer le gradient d'une fonction scalaire g de la forme

$$\begin{aligned} g : \Omega_{ip} &\rightarrow \mathbb{R}, \\ m &\mapsto G\left(\mathcal{O} \circ \tilde{F} \circ \mathcal{P}(m)\right), \end{aligned}$$

où Ω_{ip} est un ouvert de $\mathbb{R}^{n_{ip}}$, $\tilde{F}$ est définie implicitement par

$$\mathcal{U} = \tilde{F}(\mathcal{K}) \Leftrightarrow \mathcal{E}_h(\mathcal{K},\mathcal{U}) = \mathbf{0}.$$

L'équation $\mathcal{E}_h(\mathcal{K},\mathcal{U}) = \mathbf{0}$ est appelée équation d'état. La fonction $\mathcal{O} : \mathbb{R}^{n_{dof}} \rightarrow \Omega_{op}$ est appelée opérateur d'observation ou opérateur de mesure. La fonction $\mathcal{P} : \Omega_{ip} \rightarrow \ldots$ est appelée opérateur de paramétrisation. Les opérateurs $\mathcal{P}$, $\mathcal{O}$ et G sont des fonctions différentiables.

L'équation d'état $\mathcal{E}_h(\mathcal{K},\mathcal{U}) = \mathbf{0}$ définissant $\tilde{F}$ est considérée comme une contrainte additionnelle, par l'introduction du Lagrangien

$$\begin{aligned} \mathcal{L} : \Omega_{ip} \times \mathbb{R}^{n_{dof}} \times \mathbb{R}^{n_{dof}} &\rightarrow \mathbb{R}, \\ (m;\mathcal{U},\lambda) &\mapsto G\left(\mathcal{O}\left(\mathcal{U}\right)\right) + \left\langle \mathcal{E}_h\left(\mathcal{P}(m),\mathcal{U}\right), \lambda \right\rangle. \end{aligned}$$

Le Lagrangien est traditionnellement utilisé dans les problèmes d'optimisation, G étant une fonction coût à minimiser sous la contrainte $\mathcal{E}_h = 0$. Nous allons voir que le Lagrangien peut être utile, avec un autre choix de G, pour calculer la matrice Jacobienne de fonctions définies implicitement. Pour un paramètre donné m, on note $\mathcal{U}_m$ la variable d'état solution de l'équation $\mathcal{E}_h(\mathcal{P}(m),\mathcal{U}) = \mathbf{0}$. La variable d'état $\mathcal{U}_m$ est aussi caractérisée par l'équation

$$\forall \delta\lambda \in \Lambda, \quad \frac{\partial \mathcal{L}}{\partial \lambda}\left(m;\mathcal{U}_m,\lambda\right)\delta\lambda = \mathbf{0}.$$

Cette condition est indépendante de la variable λ. D'une façon similaire, si $\mathcal{O}$, $\mathcal{E}_h$ et G sont suffisamment régulières, l'état adjoint λ_m est défini par l'équation

$$\forall \delta\mathcal{U} \in \mathbb{R}^{n_{dof}}, \quad \frac{\partial \mathcal{L}}{\partial \mathcal{U}}\left(m;\mathcal{U}_m,\lambda_m\right)\delta\mathcal{U} = \mathbf{0}.$$

Ainsi, λ_m est la solution de l'équation linéaire

$$\left[\frac{\partial \mathcal{E}_h}{\partial \mathcal{U}}(\mathcal{P}(m),\mathcal{U}_m)\right]^T \lambda_m = -\left[\left(G \circ \mathcal{O}\right)'\left(\mathcal{U}_m\right)\right]^T. \tag{4.13}$$

De plus on a, $\forall m \in \Omega_{ip}, \quad g(m) = \mathcal{L}(m;X_m,\lambda_m)$ car $\mathcal{U}_m$ vérifie l'équation $\mathcal{E}_h(\mathcal{P}(m),\mathcal{U}_m) = \mathbf{0}$. Par conséquent, si $\mathcal{E}_h$ est suffisamment régulière,

$$\begin{aligned} \langle \nabla g, \delta m \rangle &= \frac{\partial \mathcal{L}}{\partial m}(m;\mathcal{U}_m,\lambda_m)\delta m + \frac{\partial \mathcal{L}}{\partial \mathcal{U}}(m;\mathcal{U}_m,\lambda_m)\frac{\partial \mathcal{U}_m}{\partial m}\delta m + \frac{\partial \mathcal{L}}{\partial \lambda}(m;\mathcal{U}_m,\lambda_m)\frac{\partial \lambda_m}{\partial m}\delta m \\ &= \frac{\partial \mathcal{L}}{\partial m}(m;\mathcal{U}_m,\lambda_m)\delta m. \end{aligned}$$

Si $\tilde{F}$, $\mathcal{O}$, $\mathcal{P}$ et G sont suffisamment régulières alors g est différentiable et son gradient est donné par

$$\nabla g = \left[\frac{\partial \mathcal{E}_h}{\partial m}(m, X_m)\right]^T \lambda_m. \tag{4.14}$$

Lorsque $G(v) = \frac{1}{2}\|v - d\|^2$, g est la fonction des moindres carrés associée à la résolution d'un problème inverse. La résolution de ce problème inverse par une méthode itérative nécessite d'évaluer à chaque itération le gradient de la fonction coût g. La méthode de l'état adjoint évite le calcul de la matrice Jacobienne de $\tilde{F}$. Le calcul de gradient se décompose en

- Calcul de la variable d'état $\mathcal{U}_m$ par la définition de $\mathcal{E}_h$
- Détermination de l'état adjoint λ_m par (4.13)
- Détermination de $\vec{\nabla} g(m)$ par (4.14)

Lorsque $G(\Phi) = \langle \Phi, v \rangle$ pour un vecteur de réels de taille n_{op} donné v, alors

$$\nabla g = [\mathcal{P}'(m)]^T \left[\tilde{F}'(\mathcal{P}(m))\right]^T [\mathcal{O}'(\mathcal{U}_m)]^T v. \tag{4.15}$$

Lorsque $v = e_i$ (vecteur de la base canonique de $\mathbb{R}^{n_{op}}$), l'équation 4.15 donne la $i^{ème}$ ligne de la matrice Jacobienne $\left(\mathcal{O} \circ \tilde{F} \circ \mathcal{P}\right)'(m)$.

Présentons maintenant un exemple soulignant l'intérêt de la méthode de l'état adjoint.

Soit $f : (A, b) \mapsto g(X) \in \mathbb{R}$ tel que $AX = b$ où A est une matrice $n \times n$ inversible et b un vecteur de taille n. On notera

$$\begin{aligned} g_X &= \vec{\nabla}_X g, \\ g_A &= \vec{\nabla}_A f, \\ g_b &= \vec{\nabla}_b f, \\ \text{pour i, j} = 1, ..., \text{n} \quad g_{A_{ij}} &= \vec{\nabla}_{A_{ij}} f. \end{aligned}$$

Pour dériver f en mode inverse signifie on doit expliciter la fonction

$$g_X \mapsto (g_A, g_b).$$

On peut utiliser la méthode de l'état adjoint selon

$$AX = b \tag{4.16}$$

$$A^T \lambda = -g_X \tag{4.17}$$

$$g_b = -\lambda^T \tag{4.18}$$

$$\text{Pour } i, j = 1, ..., n \quad g_{A_{ij}} = X_j \lambda_i. \tag{4.19}$$

Si on ne connaît pas la méthode de l'état adjoint, ce qui est le cas des différentiateurs automatiques, on effectue une dérivation opération par opération sans introduire de variable intermédiaire λ : on considère donc

$$(A, b) \mapsto X \text{ tel que } AX = b.$$

Selon la règle de composition on a

$$g_A = g_X \frac{\partial X}{\partial A} \ ; \ g_b = g_X \frac{\partial X}{\partial b}.$$

On obtient immédiatement

$$g_b = g_X A^{-1}. \tag{4.20}$$

L'expression (4.20) est bien équivalente aux expressions (4.17),(4.18). Intéressons-nous maintenant au terme g_A.

$$\frac{\partial X}{\partial A} \mathrm{d}A = -A^{-1}(\mathrm{d}A)X.$$

$$\left(A^{-1}(\mathrm{d}A)X\right)_i = \sum_{j,k=1}^{n} \left(A^{-1}\right)_{ij} (\mathrm{d}A)_{jk} X_k = \sum_{j,k=1}^{n} \left(A^{-1}\right)_{ij} X_k (\mathrm{d}A)_{jk}$$

donc

$$\frac{\partial X_i}{\partial A_{jk}} = -\left(A^{-1}\right)_{ij} X_k.$$

$$g_X \frac{\partial X}{\partial A_{jk}} = \sum_{i=1}^{n} g_{X_i} \frac{\partial X_i}{\partial A_{jk}} = -\sum_{i=1}^{n} g_{X_i} \left(A^{-1}\right)_{ij} X_k$$

avec

$$X_k = \left(A^{-1}b\right)_k = \sum_{l=1}^{n} \left(A^{-1}\right)_{kl} b_l.$$

Finalement

$$g_X \frac{\partial X}{\partial A_{jk}} = -\sum_{i=1}^{n} \sum_{l=1}^{n} g_{X_i} \left(A^{-1}\right)_{ij} \left(A^{-1}\right)_{kl} b_l,$$

i.e.

$$g_{A_{jk}} = g_X H_{j,k} b \tag{4.21}$$

où $H_{j,k}$ est une matrice de taille $n \times n$ définie par

$$(H_{j,k})_{il} = -\left(A^{-1}\right)_{ij} \left(A^{-1}\right)_{kl},$$

On vérifie que l'expression (4.21) est équivalente aux expressions (4.17),(4.19). L'utilisation de l'égalité (4.21) permet de calculer tous les gradients voulus sans recalcul, une fois stockée la matrice H. Dans le cas où A est une grande matrice creuse, ceci peut être rédibitoire (son inverse est sous forme de matrice pleine). La méthode de l'état adjoint peut se présenter comme un compromis entre recalcul et stockage.

4.3.2.2 Application au problème d'écoulement

Calcul de l'état adjoint. On veut déduire du gradient d'une fonction scalaire z par rapport à $\mathcal{U}$ le gradient de cette même fonction par rapport à une composante de $\mathcal{K}$. On utilise l'état adjoint λ, solution de l'équation

$$\left[\frac{\partial \mathcal{E}_h}{\partial \mathcal{U}}\right]^T \lambda = -\vec{\nabla}_{\mathcal{U}} z \tag{4.22}$$

soit

$$\begin{pmatrix} A_{\mathcal{K}} & B^T & \tilde{C}^T \\ B & \mathbf{0} & \mathbf{0} \\ \tilde{C} & \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \lambda_U \\ \lambda_P \\ \lambda_{\tilde{L}} \end{pmatrix} = \begin{pmatrix} -\vec{\nabla}_U z \\ -\vec{\nabla}_P z \\ -\vec{\nabla}_{\tilde{L}} z \end{pmatrix} \tag{4.23}$$

$$\lambda_{L^D} = -\vec{\nabla}_{L^D} z. \tag{4.24}$$

Cette équation correspond à un problème aux limites et est résolue par une méthode similaire à celle utilisée pour le problème direct (voir la section 2.1.3.4). Les caractéristiques du problème aux limites correspondant sont résumées dans le Tableau 4.5.

Source	$-\vec{\nabla}_P z$
Conditions aux limites	de Dirichlet sur Γ_D, valeur $-\vec{\nabla}_{L^D} z$ de Neumann sur Γ_N, valeur $-\vec{\nabla}_{\tilde{L}} z$
Terme constant	$-\vec{\nabla}_U z - \vec{\nabla}_{L^D} z$

TAB. 4.5 – Caractéristiques du problème aux limites correspondant au calcul de l'état adjoint (4.22).

Calcul des gradients. On peut ensuite en déduire $\vec{\nabla}_{\mathcal{K}} z$:

$$\vec{\nabla}_{\mathcal{K}} z = \left[\frac{\partial \mathcal{E}_h}{\partial \mathcal{K}}\right]^T \lambda \tag{4.25}$$

soit :

1. si $\mathcal{K}$ est scalaire et uniforme, on calcule un gradient par rapport à la valeur réelle des coefficients non nuls de $\mathcal{K}$, notée K :

$$\vec{\nabla}_K z = {}^t U \begin{pmatrix} -\frac{A_{\mathcal{K},1}}{K} & \dots & -\frac{A_{\mathcal{K},l}}{K} & \dots & -\frac{A_{\mathcal{K},N_c}}{K} \end{pmatrix} \lambda_U \tag{4.26}$$

2. Si $\mathcal{K}$ est scalaire non uniforme, on calcule un gradient par rapport à chaque valeur scalaire : $\mathcal{K} \equiv K_i \mathrm{Id}$ sur le sous-domaine i noté SD_i

$$\vec{\nabla}_{K_i} z \;=\; \sum_{e\in I, T_e\subset SD_i} {}^tU^e \left(-\frac{A_{\mathcal{K},e}}{K_i}\right) \lambda_U{}^e \tag{4.27}$$

(Les vecteurs U et λ_U peuvent se décomposer en blocs, chaque bloc correspondant à une cellule du maillage $\mathcal{T}_h$. Les blocs élémentaires U^e et $\lambda_U{}^e$ correspondent à la cellule T_e.)

3. Sinon on a besoin de connaître explicitement la matrice $\dfrac{\partial \mathcal{E}_h}{\partial \mathcal{K}}$ et en particulier la matrice $\dfrac{\partial\left(A_{\mathcal{K}}U\right)}{\partial \mathcal{K}}$.
D'après l'équation (4.5),

$$\frac{\partial\left(A_{\mathcal{K}}U\right)}{\partial \mathcal{K}}\mathrm{d}\mathcal{K} = -\mathcal{A}\left(\mathcal{K}^{-1}\left(\mathrm{d}\mathcal{K}\right)\mathcal{K}^{-1}\right)U. \tag{4.28}$$

On note $\mathcal{R} = \mathcal{K}^{-1}$ et $\mathrm{d}\mathcal{R} = \mathcal{R}\left(\mathrm{d}\mathcal{K}\right)\mathcal{R}$. Pour $p = 1..3,\ q = 1..3$,

$$\left(\mathrm{d}\mathcal{R}\right)_{i,j} = \sum_{k=1}^{3}\sum_{l=1}^{3} \mathcal{R}_{p,k}\mathcal{R}_{l,q}\mathrm{d}\mathcal{K}_{k,l}. \tag{4.29}$$

D'après la définition des règles de quadratures donnée dans la section 4.3.1, pour i, j indices de degrés de liberté,

$$\mathcal{A}\left(\mathrm{d}R\right)_{i,j} = -\sum_{n=1}^{N_q^e} P_{e,n} \sum_{p,q=1..3}\left(\sum_{k,l=1..3} \mathcal{R}_{p,k}\mathcal{R}_{l,q}\mathrm{d}\mathcal{K}_{k,l}\right) u^e_{i,n_p} u^e_{j,n_q}. \tag{4.30}$$

$$\left[\mathcal{A}\left(\mathrm{d}\mathcal{R}\right)U\right]_i = -\sum_{k,l}\left(\sum_j\sum_n P_{e,n}\sum_{p,q}\mathcal{R}_{p,k}\mathcal{R}_{l,q}u^e_{i,n_p}u^e_{j,n_q}U_j\right)\mathrm{d}\mathcal{K}_{k,l} \tag{4.31}$$

La composante de la $i^{ème}$ ligne de la matrice jacobienne correspondant à la valeur de la composante d'indice (k, l) de $\mathcal{K}$ sur l'élément e est

$$\left[\frac{\partial\left(A_{\mathcal{K}}U\right)}{\partial \mathcal{K}}\right]_{i,(e,(k,l))} = -\sum_j\left(\sum_n P_{e,n}\sum_{p,q}\mathcal{R}_{p,k}\mathcal{R}_{l,q}u_{i,n_p}u_{j,n_q}\right)U_j. \tag{4.32}$$

Finalement,

$$\left[\frac{\partial\left(A_{\mathcal{K}}U\right)}{\partial \mathcal{K}}\right]_{.,(e,(k,l))} = \mathcal{A}\left(H^{(k,l)}\right)U \tag{4.33}$$

où, pour $p, q = 1..3$

$$H^{(k,l)}_{p,q} = -\mathcal{R}_{p,k}\mathcal{R}_{l,q}. \tag{4.34}$$

On calcule un gradient par rapport à la valeur de la composante d'indice (k, l), $k \geq l$ de $\mathcal{K}$ sur le sous-domaine i, notée $k^{(i)}_{k,l}$, en imposant la symétrie de $\mathrm{d}\mathcal{K}$:

$$
\begin{array}{ccccccccccccc}
m & \rightarrow & \boxed{\mathcal{P}} & \rightarrow & \mathcal{K} & \rightarrow & \boxed{\tilde{F}} & \rightarrow & \mathcal{U} & \rightarrow & \boxed{\mathcal{O}} & \rightarrow & z \\
1 & \rightarrow & \boxed{\mathcal{P}'(m)} & \rightarrow & \dfrac{\partial \mathcal{K}}{\partial m} & \rightarrow & \boxed{\tilde{F}'(\mathcal{K})} & \rightarrow & \dfrac{\partial \mathcal{U}}{\partial m} & \rightarrow & \boxed{\mathcal{O}'(\mathcal{U})} & \rightarrow & \dfrac{\partial z}{\partial m} \\
\delta m & \rightarrow & \boxed{\mathcal{P}'(m)} & \rightarrow & \delta \mathcal{K} & \rightarrow & \boxed{\tilde{F}'(\mathcal{K})} & \rightarrow & \delta \mathcal{U} & \rightarrow & \boxed{\mathcal{O}'(\mathcal{U})} & \rightarrow & \delta z \\
\vec{\nabla}_m z & \leftarrow & \boxed{\mathcal{P}'(m)^T} & \leftarrow & \vec{\nabla}_{\mathcal{K}} z & \leftarrow & \boxed{\tilde{F}'(\mathcal{K})^T} & \leftarrow & \vec{\nabla}_{\mathcal{U}} z & \leftarrow & \boxed{\mathcal{O}'(\mathcal{U})^T} & \leftarrow & 1 \\
g_m & \leftarrow & \boxed{\mathcal{P}'(m)^T} & \leftarrow & g_{\mathcal{K}} & \leftarrow & \boxed{\tilde{F}'(\mathcal{K})^T} & \leftarrow & g_{\mathcal{U}} & \leftarrow & \boxed{\mathcal{O}'(\mathcal{U})^T} & \leftarrow & g_z
\end{array}
$$

FIG. 4.3 – « Chain rule » illustrée en mode direct et en mode inverse : $\left(\mathcal{O} \circ \tilde{F} \circ \mathcal{P}\right)' = \left(\mathcal{O}' \circ \tilde{F} \circ \mathcal{P}\right)\left(\tilde{F}' \circ \mathcal{P}\right)\mathcal{P}'$ (cette illustration suppose que $\mathcal{P}$ est définie sur une partie de $\mathbb{R}$ et que $\mathcal{O}$ est à valeurs dans $\mathbb{R}$).

– si $k = l$

$$\vec{\nabla}_{k_{k,k}^{(i)}} z = \sum_{e \in I, T_e \subset SD_i} {}^t U^e \left(\mathcal{A}_e \left(H^{(k,k)}\right)\right) \lambda_U{}^e \ ; \tag{4.35}$$

– sinon

$$\vec{\nabla}_{k_{k,l}^{(i)}} z = \sum_{e \in I, T_e \subset SD_i} {}^t U^e \left(\mathcal{A}_e \left(H^{(k,l)} + H^{(l,k)}\right)\right) \lambda_U^e \ . \tag{4.36}$$

Pour évaluer le gradient d'une fonction dépendant à la fois de K et U : $z = Z(U, K)$, il faudrait ajouter la quantité calculée comme ci-dessus à $\vec{\nabla}_K Z$ pour obtenir $\vec{\nabla}_K z$.

4.4 Comparaison de performances

Nous allons comparer les différentiations manuelle et automatique en termes de contraintes sur les programmes à dériver, de durée de mise en œuvre, de place mémoire et de temps de calcul, sur des cas tests représentatifs.

4.4.1 Présentation des cas tests

4.4.1.1 Problème modèle simplifié

Pour les comparaisons de performance nous avons supposé la perméabilité scalaire et uniforme et différentié par rapport à la perméabilité K scalaire les champs de pression et de vitesse discrets. Autrement dit, nous utilisons l'opérateur de paramétrisation

$$\begin{array}{rccl} \mathcal{P} : & \mathbb{R} & \rightarrow & \mathcal{M}_{3\times 3}^{+}{}^{N_c} \\ & K & \mapsto & \mathcal{K} \equiv K\mathbf{I}. \end{array} \tag{4.37}$$

Nous utilisons deux cas tests : nous avons choisi un domaine cylindrique découpé en 3200 hexaèdres et nous imposons un écoulement monodimensionnel selon son axe principal (conditions de flux nul sur la face latérale). Pour le test que nous appellerons « Neumann-Neumann » nous imposons un flux uniforme identique sur les deux faces de base, pour le test que nous appellerons « Dirichlet-Dirichlet » nous imposons une différence de pression. Dans ces deux cas, les dérivées des problèmes continus (2.1) s'écrivent simplement :

1. Cas test « Neumann-Neumann » (pas de solution unique, mais on peut obtenir une solution particulière, et $\vec{u}$ et $\vec{\nabla} p$ sont uniques) :

$$\frac{\partial \vec{u}}{\partial K} = \vec{0} \; ; \; \frac{\partial \left(\vec{\nabla} p\right)}{\partial K} = -\frac{\vec{\nabla} p}{K} \tag{4.38}$$

   et les variations élémentaires vérifient

$$\delta \vec{u} = \vec{0} \; ; \; \delta \left(\vec{\nabla} p\right) = -\frac{\delta K}{K} \vec{\nabla} p. \tag{4.39}$$

2. Cas-test « Dirichlet-Dirichlet » :

$$\frac{\partial \vec{u}}{\partial K} = \frac{\vec{u}}{K} \; ; \; \frac{\partial p}{\partial K} = 0 \; ; \; \frac{\partial \left(\vec{\nabla} p\right)}{\partial K} = \vec{0} \tag{4.40}$$

   et les variations élémentaires vérifient

$$\delta \vec{u} = \frac{\delta K}{K} \vec{u} \; ; \delta p = 0 \; ; \delta \left(\vec{\nabla} p\right) = \vec{0}. \tag{4.41}$$

Ce cas simple permet donc un première vérification du calcul des dérivées.

Pour l'évaluation des performances de la différentiation automatique en mode inverse, on choisit comme mesure l'une des composantes de U discret : autrement dit on choisit l'opérateur d'observation ou de mesure

$$\mathcal{O} \; : \mathcal{U} \in \mathbb{R}^{n_{dof}} \rightarrow U_0 \in \mathbb{R}. \tag{4.42}$$

Pour ces tests nous avons programmé et utilisé un algorithme de gradient conjugué générique, c'est-à-dire fonctionnant pour des vecteurs de scalaires quelconques (vecteurs de `double`, d'`adouble` ou de tout type pour lesquels les opérateurs binaires auront été surchargés), ce qui n'est pas forcément prévu dans les bibliothèques de solveurs telles que `AZTEC` ou `UMFPACK`. Pour le calcul automatique, nous différentions cet algorithme itération par itération, ce qui peut poser problème (voir la section 6.3.2). Si le calcul de dérivée à la main demande moins d'itérations de gradient conjugué que le calcul de valeurs, alors on effectue des opérations inutiles en différentiation automatique. Dans le cas contraire, le calcul de dérivées en différentiation automatique n'est pas assez précis. De plus, le conditionnement de la matrice à inverser est très mauvais pour ce test, il faudrait utiliser une autre méthode itérative ou un préconditionneur (on aurait besoin d'une version générique de `AZTEC`). En mode adjoint automatique, nous n'arrivons pas au bout du calcul des 10000 dérivées demandées (mais presque) pour des raisons de place mémoire. Pour des domaines de calcul plus petits, on retrouve les résultats du mode direct.

4.4.1.2 Test Neumann-Neumann (direct)

Nous avons représenté sur la Figure 4.4 les champs de pression et de vitesse et leurs variations élémentaires pour $K = 7$, $\delta K = 1$. On observe une variation du gradient de pression proportionnelle au gradient de pression, avec un facteur de proportionnalité égal à $-\dfrac{5\ 10^{-2}}{3.5\ 10^{-1}} \approx -\dfrac{1}{7}$ (les échelles de couleurs automatiques des légendes vont de la valeur minimale à la valeur maximale), en mode manuel et en mode automatique. La variation du champ de vitesse est négligeable devant le champ de vitesse. Ceci est en accord avec l'équation (4.39) correspondant au problème continu. Le résultat du calcul de variation de vitesse est plus proche de zéro avec la différentiation automatique qu'avec la différentiation manuelle, ce qui est lié au « dépassement (dans ce cas) du critère d'arrêt » pour la résolution des systèmes linéaires en mode automatique (en mode manuel, le calcul de variations demande moins d'itérations de gradient conjugué que le calcul de valeurs).

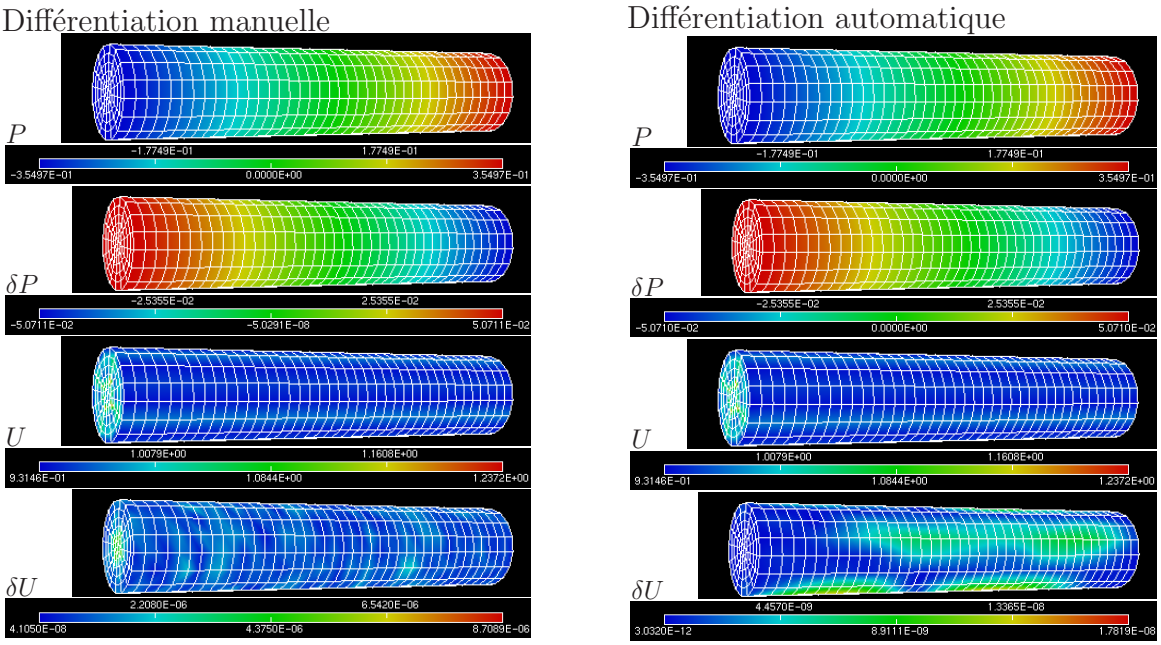


FIG. 4.4 – Calcul de variation en mode direct, test « Neumann-Neumann ».

4.4.1.3 Test Dirichlet-Dirichlet (direct)

Nous avons représenté sur la Figure 4.5 les champs de pression et de vitesse et leurs variations élémentaires pour $K = 7$, $\delta K = 1$. On observe une variation de champ de pression négligeable devant le champ de pression. La variation de champ vitesse est proportionnelle au champ de vitesse, avec un facteur de proportionnalité égal à $\dfrac{0.214}{1.49} \approx \dfrac{1}{7}$ (les échelles de couleurs automatiques des légendes vont de la valeur minimale à la valeur maximale), en

mode manuel et en mode automatique. Ceci est en accord avec l'équation (4.41) correspondant au problème continu. Le résultat du calcul de variation de pression est plus proche de zéro avec la différentiation automatique qu'avec la différentiation manuelle, ce qui est lié au « dépassement (dans ce cas) du critère d'arrêt » pour la résolution des systèmes linéaires en mode automatique (en mode manuel, le calcul de variations demande moins d'itérations de gradient conjugué que le calcul de valeurs).

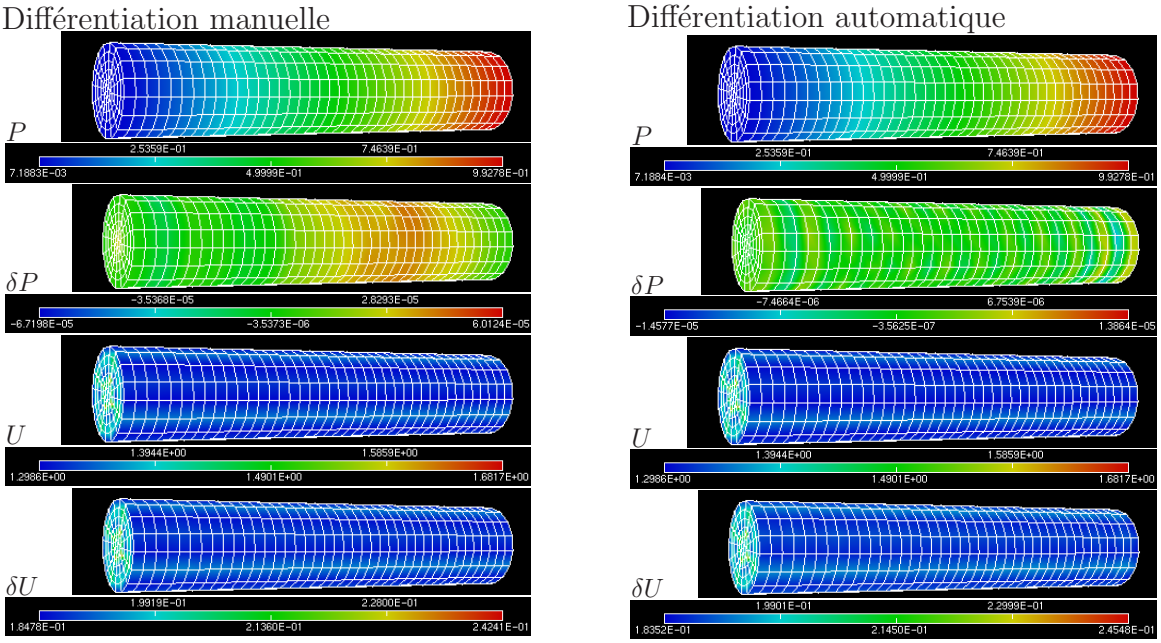


FIG. 4.5 – Calcul de variations en mode direct, test « Dirichlet-Dirichlet ».

4.4.2 Résultats numériques

Temps de développement. Pour cette première expérience, le temps de développement a été environ deux fois plus long pour la différentiation manuelle que pour la différentiation automatique. Pour les expériences suivantes, le temps de développement avec `ADOL-C` diminue.

Temps d'exécution. Afin de comparer les temps d'exécution, nous avons décomposé le temps d'exécution du code différentié manuellement en 4 parties :

1. init : temps nécessaire pour « construire » le problème (principalement la lecture du maillage) ;
2. values : calcul de la solution de l'équation de Darcy ;
3. direct : calcul de dérivée en mode direct ;
4. reverse : calcul d'une composante de dérivée en mode inverse.

Nous avons décomposé le temps d'exécution pour la différentiation automatique en 4 parties :

1. init : temps nécessaire pour « construire » le problème ;
2. taping : première résolution de l'équation de Darcy, avec fabrication d'un enregistrement ;
3. `forward` : relecture de l'enregistrement avec calcul de dérivée en mode direct ;
4. `vec_jac` : relecture de l'enregistrement « à l'envers » (des opérations préliminaires sont nécessaires) pour le calcul d'une composante de dérivée en mode inverse.

Notons que dans ce cas particulier, avec une seule valeur d'entrée, il n'est pas raisonnable d'utiliser le mode inverse pour calculer la matrice Jacobienne ligne par ligne alors que c'est un vecteur colonne.

Les résultats de cette comparaison sont reportés dans le Tableau 4.6.

Neumann-Neumann				Dirichlet-Dirichlet			
manuel		automatique		manuel		automatique	
init	0.5 s	1.0 s	init	init	0.5 s	1.0 s	init
values	7.3 s	12.7 s	taping	values	7.3 s	15.1 s	taping
direct	2.8 s	2.8 s	`forward`	direct	2.8 s	3.5 s	`forward`
1×reverse	5.7 s	6.2 s	1×`vec_jac`	1×reverse	5.7 s	7.4 s	1×`vec_jac`

Tab. 4.6 – Comparaison des temps d'exécution des deux codes dérivés.

Ainsi, pour un tel problème, le temps d'exécution ne semble pas être une limite à l'utilisation de la différentiation automatique.

De la manière où nous l'utilisons ici, la différentiation automatique crée plusieurs fichiers d'enregistrement de quelques centaines de Méga-octets chacun. Nous avons encore des difficultés pour la gestion de la place mémoire en différentiation automatique. Cependant, elles devraient pouvoir être surmontées en décomposant le problème en sous-problèmes suffisamment petits et en utilisant les méthodes décrites dans [31].

En différentiation automatique il est relativement simple de modifier le choix des arguments ou des résultats (de $\mathcal{P}$ et $\mathcal{O}$).

4.4.3 Conclusion

Le programme fonctionnant par différentiation automatique est limité à la résolution de problèmes de petite taille. En revanche, il peut considérer des conditions aux limites comme des paramètres d'entrée.

Le programme fonctionnant en mode manuel a pu être utilisé sur un maillage de 300000 mailles, décomposé en 14 sous-domaines dont 2 avec diffusion tensorielle (maillage utilisé dans [72]). En mode direct, nous avons calculé la matrice Jacobienne complète. En mode

adjoint, nous avons calculé la matrice Jacobienne de fonctions retournant quelques composantes de flux ou le flux d'eau à travers des exutoires (voir le Chapitre 7).

La différentiation automatique s'est révélée être un outil intéressant pour vérifier une différentiation manuelle. En effet, la généralisation d'une différentiation automatique, en ce qui concerne le choix des paramètres d'entrée (i.e. la définition de $\mathcal{P}$) par exemple, est plus simple à implémenter que celle d'une vérification par différences divisées.

Nous suggérons d'utiliser une différentiation manuelle pour des programmes faisant appel à des bibliothèques extérieures et pour des programmes utilisant des méthodes itératives ; et d'utiliser la différentiation automatique pour les programmes dans lesquels la liste des opérations réalisées pendant l'exécution dépend beaucoup des paramètres (schémas décentrés par exemple) mais pas itérativement.

Il est possible de combiner les deux (voir l'Annexe A), un solution pouvant être de faire communiquer, au moyen d'une bibliothèque d'échange de données, des exécutables différentiés manuellement et des exécutables différentiés automatiquement (voir les sections 6.5.2 et 6.7).

Notons enfin qu'`ADOL-C` s'utilise plus facilement sur des programmes utilisant le plus possible les fonctionnalités de `C++` (par opposition aux programmes en langage `C`) telles que les `template`s. Les modifications à effectuer sur les codes sources pour différencier avec `ADOL-C` des programmes élaborés sont détaillées en section 6.3.

Chapitre 5

Décomposition de domaine

5.1 Décomposition de domaine sans recouvrement

L'intérêt de la décomposition de domaine est double : d'une part, cela permet de répartir les besoins en place mémoire sur plusieurs processeurs. D'autre part, l'algorithme résultant a un meilleur taux de convergence (même utilisé sur un seul processeur).

Nous allons nous intéresser à une méthode de décomposition de domaine sans recouvrement, présentée par exemple dans [55]. Pour optimiser la convergence de l'algorithme, il est important d'utiliser un préconditionnement, par exemple de type Neumann-Neumann avec équilibrage [58, 20].

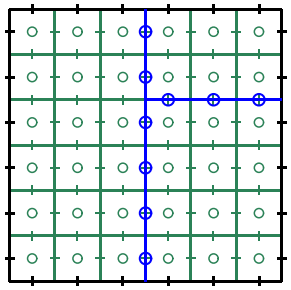
L'objectif de ce chapitre est d'évaluer l'impact de la décomposition de domaine sur les calculs de dérivée. Pour simplifier les notations, nous ne nous intéressons pas dans ce chapitre au préconditionnement.

5.1.1 Idée générale

On rappelle l'équation de Darcy

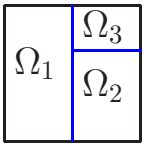
$$\mathcal{P}\left\{\begin{array}{rcll} \text{div } \vec{u} & = & q & \Omega \\ \vec{u} & = & -\mathcal{K}\vec{\nabla}p & \Omega \\ p & = & \overline{p} & \Gamma_D \\ \vec{u}\cdot\vec{n} & = & \overline{g} & \Gamma_N \end{array}\right.$$

Ω



$\Omega = \Omega_1 \cup \Omega_2 \cup \Omega_3$

$\Sigma = \Sigma_{12} \cup \Sigma_{13} \cup \Sigma_{23}$



$\Gamma = \Gamma_D \cup \Gamma_N$

Les degrés de liberté pour la formulation sans décomposition de domaine décrite dans la section 2.1 sont une valeur de pression par maille (aux signes $\{\circ\}$) et les flux de $\vec{u}$ à travers les faces des mailles (aux signes $\{+\}$, $\{+\}$ et $\{\oplus\}$). La formulation de décomposition de domaine choisie consiste à écrire des problèmes aux limites sur différents sous-domaines Ω_i séparés par l'interface Σ. Les conditions aux limites sur les frontières des sous-domaines non incluses dans Γ sont de type Dirichlet, leur valeur est à déterminer : c'est une nouvelle

inconnue du problème. Une équation supplémentaire impose la continuité du flux de $\vec{u}$ à travers les interfaces. Les degrés de liberté pour cette formulation seront donc ceux de la formulation "classique" (sans décomposition de domaine) auxquels on ajoute une valeur de pression à chaque face de l'interface Σ (aux signes $\{\oplus\}$).

5.1.2 Formulation

Le domaine de calcul Ω est décomposé en n_{sd} sous-domaines. Nous introduisons les notations suivantes :

- $\overline{\Omega} = \cup_{i=1,...,n_{sd}} \overline{\Omega}_i$, $\Omega_i \cap \Omega_j \neq \emptyset \Leftrightarrow i = j$: décomposition du domaine de calcul. Ω et les Ω_i pour $i = 1, ..., n_{sd}$ sont des ouverts. De plus, on suppose que pour toute cellule T_e du maillage $\mathcal{T}_h$, il existe $i \in 1, ..., n_{sd}$ tel que $T_e \subset \Omega_i$.
- Σ : réunion des frontières des sous-domaines. On a $\Sigma \cap \Gamma_D = \emptyset$ et $\Sigma \cap \Gamma_N = \emptyset$.
- Pour $i \in 1, ..., n_{sd}$ on définit $\Sigma_i = \Sigma \cap \overline{\Omega}_i$. Par conséquent $\Sigma = \cup_{i=1,...,n_{sd}} \Sigma_i$.
- Pour $i < j$ on définit $\Sigma_{ij} = \Sigma_i \cap \Sigma_j$. L'interface $\Sigma_{i,j}$ est de codimension 1 lorsque les domaines Ω_i et Ω_j sont voisins.
- Pour toute application x définie sur Ω, on note x_i la restriction de x au sous-domaine Ω_i. Pour toute application y définie sur Σ, on note y_i sa restriction à Σ_i.
- $\vec{\mathrm{n}}_i$ est la normale unitaire extérieure à Ω_i.

La formulation décomposition de domaine s'écrit

Trouver $(\vec{u}_i)_{i=1,...,n_{sd}}$, $(p_i)_{i\in 1,...,n_{sd}}$, λ, tels que

$$\begin{cases} \text{pour } i = 1, ..., n_{sd} \ (p_i, \vec{u}_i) \text{ est solution de } \mathcal{P}_i(\lambda) : \begin{cases} \mathrm{div}\vec{u}_i = q_i & \Omega_i \\ \vec{u}_i + \mathcal{K}_i \vec{\nabla} p_i = 0 & \Omega_i \\ p_i = \lambda_i & \Sigma_i \\ p_i = \overline{p}_i & \Gamma_D \cap \overline{\Omega}_i \\ \vec{u}_i \cdot \vec{\mathrm{n}}_i = \overline{g}_i & \Gamma_N \cap \overline{\Omega}_i \end{cases} \\ \\ \text{pour } i, j = 1, ..., n_{sd} \text{ tels que } i < j \text{ et } \Sigma_{ij} \neq \emptyset, \qquad \vec{u}_{i|\Sigma_{ij}} \cdot \vec{\mathrm{n}}_i + \vec{u}_{j|\Sigma_{ij}} \cdot \vec{\mathrm{n}}_j \equiv 0. \end{cases} \tag{5.1}$$

Pour λ donné, on note pour $i = 1, ..., n_{sd}$, $(p_i(\lambda), \vec{u}_i(\lambda))$ la solution du problème $\mathcal{P}_i(\lambda)$.

Nous définissons, pour $i, j = 1, ..., n_{sd}$ $\mu_{i,j}(\lambda) = \vec{u}_i(\lambda)_{|\Sigma_{ij}} \cdot \vec{\mathrm{n}}_i$; $\mu_{j,i}(\lambda) = \vec{u}_j(\lambda)_{|\Sigma_{ij}} \cdot \vec{\mathrm{n}}_j$; $\mu(\lambda) = \{\mu_{i,j}(\lambda) + \mu_{j,i}(\lambda)\}_{i<j}$. Nous devons donc trouver λ tel que $\mu(\lambda) = 0$.

L'application $\lambda \mapsto \mu(\lambda)$ est affine. Par conséquent nous décomposons

$$\mu(\lambda) = \mu^* + \underbrace{\mu^0(\lambda)}_{S\lambda}$$

où $S : \lambda \to \mu^0(\lambda)$ est linéaire. S est l'opérateur de Steklov-Poincaré. Plus précisément, on décompose $\vec{u} = \vec{u}^* + \vec{u}^0$; $p = p^* + p^0$ où

1. $(\vec{u}_i^*, p_i^*)$ pour $i = 1, ..., n_{sd}$ sont solutions des problèmes locaux $\mathcal{P}_i^*$ définis par

$$\mathcal{P}_i^* \begin{cases} p_i^* = 0 \text{ sur } \Sigma \ ; \\ \text{terme source et conditions aux limites sur } \Gamma \text{ du problème } \mathcal{P} \ ; \\ \text{Pas de conditions de transmission à vérifier.} \end{cases} \tag{5.2}$$

2. $\vec{u}^0, p^0$ est solution du problème couplé $\mathcal{P}^0$ défini par

$$\mathcal{P}^0 \begin{cases} p_i^0 = \lambda \text{ sur } \Sigma \text{ ;} \\ \text{terme source nul et conditions aux limites nulles homogènes sur } \Gamma \text{ ;} \\ \text{Conditions de transmission } \mu_{i,j}^0 + \mu_{j,i}^0 = -\mu_{i,j}^* - \mu_{j,i}^*. \end{cases} \tag{5.3}$$

5.1.3 Algorithme de décomposition de domaine

1. **Calculer le second membre** μ^* en résolvant les problèmes locaux $\mathcal{P}_i^*$
2. **Trouver les valeurs aux interfaces** : par un algorithme de gradient conjugué préconditionné, résoudre
$$S\lambda = -\mu^*.$$
L'opérateur S peut être décomposé sous la forme
$$S = \sum_{i=1}^{n_{sd}} R_i^T S_i R_i$$
où S_i résout $\mathcal{P}_i^0$ et retourne les flux aux interfaces locales et R_i est la restriction de Σ à Σ_i.
On notera $\lambda_i = R_i\lambda$ et $\mu_i^0 = S_i\lambda_i$. μ_i^0 n'est pas une restriction de μ^0, λ_i est bien une restriction de λ.
3. **Calculer la solution** : $\vec{u}_i, p_i$ sont obtenus en résolvant $\mathcal{P}_i(\lambda)$ (λ est alors connu).

On peut écrire l'opérateur S, qui réalise un produit matrice-vecteur sans former la matrice, sous forme "fonctionnelle" (voir la Figure 5.1).

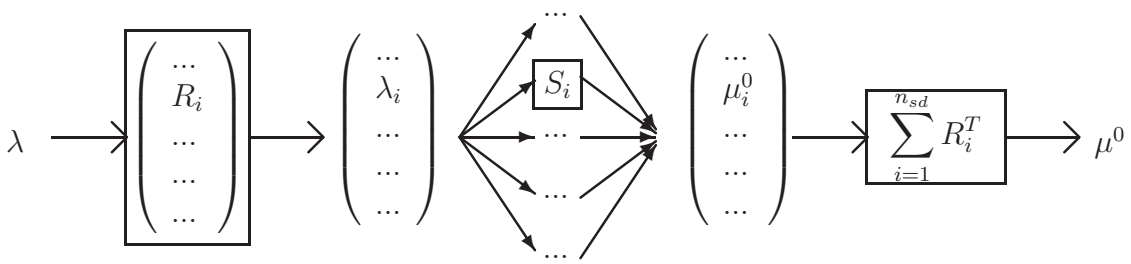


FIG. 5.1 – L'opérateur S est un argument de l'algorithme de Gradient Conjugué Préconditionné. Cette illustration permet de repérer quelles fonctions génériques d'une bibliothèque de calcul parallèle devront être utilisées pour implémenter S.

La décomposition de domaine peut aussi être utilisée dans la résolution des équations de transport, mais elle demande un plus fort couplage entre les sous-domaine lorsqu'on utilise des schémas décentrés.

5.2 Dérivation de la formulation décomposition de domaine

On s'intéresse à la dérivation de la formulation décomposition de domaine de l'équation de Darcy donnée dans l'équation (5.1).

On sait déjà résoudre et différentier le problème $\mathcal{P}_i^*$ de l'équation (5.2), selon la méthode présentée dans la section 2.1.2.2. Pour $\mathcal{P}^0$ (équation (5.3)) on utilise également une formulation mixte hybride. Les degrés de liberté sont $U_i^0, P_i^0, \tilde{L}_i$ pour $i = 1, ..., n_{sd}$. Les matrices $A_{i_\mathcal{K}}$, B_i, C_i, correspondent aux matrices $A_\mathcal{K}$, B, C de la section 2.1.3.3 pour le problème aux limites d'écoulement posé sur Ω_i avec le champ de conductivité hydraulique $\mathcal{K}_{|\Omega_i}$, des conditions aux limites de Dirichlet sur $\partial\Omega_i \cap \Gamma \cup \Gamma_D$ et des conditions aux limites de Neumann sur $\partial\Omega_i \cup \Gamma_N$. Pour le second membre, on définit U_i^* le vecteur dont les composantes correspondent à une approximation de la valeur moyenne du flux de la solution du problème local $\vec{u}_i^*$ sortant de Ω_i à travers une face de $\partial\Omega_i \cup \Gamma$.

L'opérateur $\mathcal{R}_i$ construit les conditions de Dirichlet : il forme le second membre R pour le problème aux limites posé sur Ω_i avec conditions aux limites de Dirichlet éventuellement non nulles sur $\partial\Omega_i \cap \Gamma \cup \Gamma_D$ de valeur λ_{Σ_i}.

$$\mathcal{R}_i \ : \lambda_\Sigma \to \lambda_{\Sigma_i}$$

est un opérateur de restriction : il extrait les composantes de λ_Σ correspondant à des degrés de liberté sur Σ_i.

La formulation mixte hybride pour le problème couplé s'écrit

$$\left\{\begin{array}{l} \text{pour } i = 1, ..., n_{sd},\ A_{i_\mathcal{K}} U_i^0 + B_i^T P_i^0 + \tilde{C}_i^T \tilde{L}_i = \mathcal{R}_i \lambda_\Sigma \\ \text{pour } i = 1, ..., n_{sd},\ B_i U_i^0 = 0 \\ \text{pour } i = 1, ..., n_{sd},\ \tilde{C}_i U_i^0 = 0 \\ \displaystyle\sum_{i=1}^{n_{sd}} \left(\mathcal{R}_i^T U_i^0\right) = -\sum_{i=1}^{n_{sd}} \left(\mathcal{R}_i^T U_i^*\right). \end{array}\right. \tag{5.4}$$

Les colonnes de $\mathcal{R}_i^T$ non identiquement nulles correspondent à des degrés de liberté sur Σ_i.

Les paramètres de la fonction à différentier sont $\mathcal{K}$ **et** $\{U_i^*\}_{i=1,...,n_{sd}}$.

On définit

$$\mathcal{C}\left(\mathcal{K}, \{U_i^*\}_{i=1,...,n_{sd}}; \{U_i^0\}_{i=1,...,n_{sd}}, \{P_i^0\}_{i=1,...,n_{sd}}, \left\{\tilde{L}_i\right\}_{i=1,...,n_{sd}}, \lambda_\Sigma\right) = \left(\begin{array}{l} \left(A_{i_\mathcal{K}} U_i^0 + B_i^T P_i^0 + \tilde{C}_i^T \tilde{L}_i - \mathcal{R}_i \lambda_\Sigma\right)_{i=1,...,n_{sd}} \\ \left(B_i U_i^0\right)_{i=1,...,n_{sd}} \\ \left(\tilde{C}_i U_i^0\right)_{i=1,...,n_{sd}} \\ \displaystyle\sum_{i=1}^{n_{sd}} \left(\mathcal{R}_i^T U_i^0\right) + \sum_{i=1}^{n_{sd}} \left(\mathcal{R}_i^T U_i^*\right) \end{array}\right). \tag{5.5}$$

5.2.1 Différentiation en mode direct

La dérivée en mode direct est obtenue par

$$\text{trouver } \{\delta U_i^0\}_{i=1,\dots,n_{sd}},\ \{\delta P_i^0\}_{i=1,\dots,n_{sd}},\ \left\{\tilde{L}_i^0\right\}_{i=1,\dots,n_{sd}},\ \delta\lambda_\Sigma \text{ tels que}$$
$$\frac{\partial \mathcal{C}}{\partial \mathcal{K}}\delta\mathcal{K} + \sum_{i=1,\dots,n_{sd}} \left[\frac{\partial \mathcal{C}}{\partial U_i^*}\delta U_i^* + \frac{\partial \mathcal{C}}{\partial U_i^0}\delta U_i^0 + \frac{\partial \mathcal{C}}{\partial P_i^0}\delta P_i^0 + \frac{\partial \mathcal{C}}{\partial \tilde{L}_i^0}\delta P_i^0 + \frac{\partial \mathcal{C}}{\partial \lambda_\Sigma}\delta\lambda_\Sigma\right] = 0, \tag{5.6}$$

c'est-à-dire

$$\begin{cases} \text{pour } i=1,\dots,n_{sd},\ A_{i_\mathcal{K}}\delta U_i^0 + B_i^T\delta P_i^0 + \tilde{C}_i^T\delta\tilde{L}_i^0 - \mathcal{R}_i\delta\lambda_\Sigma = \mathcal{A}_i(\mathcal{K}^{-1}(\mathrm{d}\mathcal{K})\mathcal{K}^{-1})U_i^0 \\ \text{pour } i=1,\dots,n_{sd},\ B_i\delta U_i^0 = 0 \\ \text{pour } i=1,\dots,n_{sd},\ \tilde{C}_i\delta U_i^0 = 0 \\ \displaystyle\sum_{i=1}^{n_{sd}}\left(\mathcal{R}_i^T\delta U_i^0\right) = -\sum_{i=1}^{n_{sd}}\left(\mathcal{R}_i^T\delta U_i^*\right). \end{cases}$$

On peut décomposer le problème dérivé pour retrouver un problème de décomposition de domaine standard : $\delta U^0 = \delta U^1 + \delta U^2$

1. Problèmes locaux :

$$\text{pour } i=1,\dots,n_{sd},\ \begin{cases} A_{i_\mathcal{K}}\delta U_i^1 + B_i^T\delta P_i^1 + \tilde{C}_i^T\delta\tilde{L}_i^1 = \mathcal{A}_i\left(\mathcal{K}^{-1}(\mathrm{d}\mathcal{K})\mathcal{K}^{-1}\right)U_i^0 \\ B_i\delta U_i^1 = 0 \\ C_i\delta\tilde{L}_i^1 = 0. \end{cases} \tag{5.7}$$

   Les problèmes locaux se résolvent comme l'équation (4.12).

2. Transmission :

$$\begin{cases} \text{pour } i=1,\dots,n_{sd},\ A_{i_\mathcal{K}}\delta U_i^2 + B_i^T\delta P_i^2 + \tilde{C}_i^T\delta\tilde{L}_i = \mathcal{R}_i\delta\lambda_\Sigma \\ \text{pour } i=1,\dots,n_{sd},\ B_i\delta U_i^2 = 0 \\ \text{pour } i=1,\dots,n_{sd},\ \tilde{C}_i\delta U_i^2 = 0 \\ \displaystyle\sum_{i=1}^{n_{sd}}\left(\mathcal{R}_i^T\delta U_i^2\right) = -\sum_{i=1}^{n_{sd}}\left(\mathcal{R}_i^T\delta U_i^*\right) - \sum_{i=1}^{n_{sd}}\left(\mathcal{R}_i^T\delta U_i^1\right). \end{cases} \tag{5.8}$$

5.2.2 Différentiation en mode adjoint

On calcule d'abord l'état adjoint qui vérifie

$$\left[\frac{\partial \mathcal{C}}{\partial\left(\mathcal{K}, \{U_i^*\}_{i=1,\dots,n_{sd}}\right)}\right]^T \lambda = -\vec{\nabla}_{\{U_i^0\}_{i=1,\dots,n_{sd}}\{P_i^0\}_{i=1,\dots,n_{sd}},\{\tilde{L}_i\}_{i=1,\dots,n_{sd}},\lambda_\Sigma} z \tag{5.9}$$

c'est-à-dire

$$\begin{cases} \text{pour } i=1,\dots,n_{sd},\ A_{i_\mathcal{K}}\lambda_U^i + B_i^T\lambda_P^i + \tilde{C}_i^T\lambda_{\tilde{L}}^i + \mathcal{R}_i\lambda_{\lambda_\Sigma} = -\vec{\nabla}_{U_i} z \\ \text{pour } i=1,\dots,n_{sd},\ B_i\delta\lambda_U^i = -\vec{\nabla}_{P_i} z \\ \text{pour } i=1,\dots,n_{sd},\ \tilde{C}_i\delta\lambda_U^i = -\vec{\nabla}_{\tilde{L}_i} z \\ \displaystyle\sum_{i=1}^{n_{sd}}\left(-\mathcal{R}_i^T\lambda_U^i\right) = -\vec{\nabla}_{\lambda_\Sigma} z. \end{cases} \tag{5.10}$$

Les gradients sont ensuite mis à jour par

$$\text{pour } i = 1, ..., n_{sd},\ \vec{\nabla}_{\mathcal{K}} z \leftarrow \vec{\nabla}_{\mathcal{K}} z +^t U_i \left(-\mathcal{A}_i \left(\mathcal{K}^1(\mathrm{d}\mathcal{K})\mathcal{K}^{-1}\right) \lambda_U^i\right) \ ; \tag{5.11}$$

$$\text{pour } i = 1, ..., n_{sd},\ \vec{\nabla}_{U_i^*} z \leftarrow \vec{\nabla}_{U_i^*} z - \mathcal{R}_i^T \lambda_U^i. \tag{5.12}$$

On décompose le calcul des λ pour revenir à un problème de décomposition de domaine classique : $\lambda = \lambda^* + \lambda^0$

1. Problèmes locaux :

$$\text{pour } i = 1, ..., n_{sd},\ \left\{ \begin{array}{l} A_{i_{\mathcal{K}}} \lambda_U^{i^*} + B_i^T \lambda_P^{i^*} + \tilde{C}_i^T \lambda_{\tilde{L}}^{i^*} = -\vec{\nabla}_{U_i} z \\ B_i \lambda_U^{i^*} = -\vec{\nabla}_{P_i} z \\ \tilde{C}_i \lambda_U^{i^*} = -\vec{\nabla}_{\tilde{L}_i} z \end{array} \right. \tag{5.13}$$

   Les problèmes locaux se résolvent comme l'équation (4.24).

2. Transmission :

$$\left\{ \begin{array}{l} \text{pour } i = 1, ..., n_{sd},\ A_{i_{\mathcal{K}}} \lambda_U^{i^0} + B_i^T \lambda_P^{i^0} + \tilde{C}_i^T \lambda_{\tilde{L}}^{i^0} = \mathcal{R}_i^T \lambda_{\lambda_\Sigma} \\ \text{pour } i = 1, ..., n_{sd},\ B_i \delta \lambda_U^{i^0} = 0 \\ \text{pour } i = 1, ..., n_{sd},\ \tilde{C}_i \lambda_U^{i^0} = 0 \\ \displaystyle\sum_{i=1}^{n_{sd}} \left(\mathcal{R}_i^T \lambda_U^{i^0}\right) = \vec{\nabla}_{\lambda_\Sigma} z - \sum_{i=1}^{n_{sd}} \left(\mathcal{R}_i^T \lambda_U^{i^*}\right) \end{array} \right. \tag{5.14}$$

(définition d'une décomposition de domaine adjointe).

Chapitre 6

Aspects informatiques

Nous allons présenter dans ce chapitre les principaux outils informatiques que nous avons utilisés ou développés.

6.1 Algèbre linéaire

Pour la décomposition en valeurs singulières et pour les opérations d'algèbre linéaire sur des matrices pleines, nous utilisons la bibliothèque d'algèbre linéaire `Lapack` (Linear Algebra PACKage). C'est une bibliothèque `Fortran` pour laquelle une interface `C` est disponible.

Pour la résolution des grands systèmes linéaires nous utilisons une des trois méthodes de résolution suivantes :

1. méthode itérative gradient conjugué générique, qui avait été utilisé pour la comparaison de performances avec le différentiateur automatique ;
2. bibliothèque `AZTEC` (méthodes itératives) ;
3. solveur direct creux de `UMFPACK`. La bibliothèque `AZTEC` propose aussi un solveur direct, mais `UMFPACK` permet de stocker et réutiliser la factorisation de la matrice, ce qui sera utile en particulier pour la décomposition de domaine.

6.2 Éléments finis : la bibliothèque `LifeV`

La bibliothèque `LifeV` est une bibliothèque éléments finis fournissant des implémentations de méthodes mathématiques et numériques modernes. C'est à la fois une bibliothèque industrielle et une bibliothèque de recherche. Elle a déjà été utilisée pour des applications médicales (écoulements sanguins) et industrielles. C'est un travail de coopération entre trois institutions : l'École Polytechnique Fédérale de Lausanne en Suisse, le Politecnico di Milano en Italie, et l'INRIA (projet BANG).

Elle a des applications en particulier en dynamique des fluides, en dynamique des structures, en modélisation des transferts de chaleur, des interactions fluide-structure et

du transport en milieu poreux.

Cette bibliothèque est écrite en `C++` et utilise un environnement de programmation moderne et des outils qui ont prouvé leur efficacité (elle utilise la bibliothèque `boost` [38], qui offre de nombreuses possibilités de haut niveau pour le `C++` : parmi d'autres, pointeurs partagés, aide à la construction de fonctions du second ordre, tableaux et matrices performants).

6.3 Différentiation avec `ADOL-C`

6.3.1 Modification des sources

Nous nous intéressons ici aux adaptations des fichiers définissant les sous-programmes appelés par le programme principal. (Les modifications à apporter à un programme principal pour utiliser `ADOL-C` ont été abordées dans la section 4.2.2.3.) L'exemple considéré fait appel à la bibliothèque d'éléments finis `LifeV`.

La structure présentée dans la section 4.2.2.3 est valable pour un programme constitué uniquement de la fonction principale `int main(...)`. Si le programme est plus élaboré, on peut décider de modifier les types de toutes les sous fonctions appelées entre les mots clés d'entrée et de sortie, et également celui des attributs et méthodes de classes utilisées entre les mots clés d'entrée et de sortie. On peut aussi créer des versions génériques des fonctions ou des classes, c'est à dire des versions paramétrées par un type de données, de façon à ce qu'elles puissent être utilisées indifféremment avec des variables actives ou passives. Cependant, ces modifications et les recompilations qui en découlent peuvent être interminables lorsqu'on fait appel à de nombreuses bibliothèques ; l'utilisation de variables actives n'est pas anodine concernant la consommation de mémoire, il est préférable de la réduire au minimum ; enfin, une modification systématique des types n'est pas toujours possible. Nous avons finalement suivi les idées suivantes.

Concernant les modifications de type.

1. Les variables d'espace ne seront jamais pour nous des variables actives, leur type n'est donc pas modifié. Une conséquence est qu'aucune fonction de la bibliothèque d'éléments finis définissant des règles de quadrature n'a dû être redéfinie. On rencontrera donc de nombreuses fonctions utilisant à la fois des variables actives et des variables passives. Par exemple, les fonctions construisant des matrices de masse font appel à des règles de quadrature, passives, et utilisent des coefficients de conductivité hydraulique, qui sont des variables actives indépendantes.
2. La meilleure solution n'est pas toujours d'utiliser le moins possible de variables actives. Il faut parfois faire un compromis entre la mémoire et la taille des enregistrements. Dans les enregistrements, les variables passives sont toujours remplacées par leurs valeurs, donc si une boucle sur un indice `i` fait intervenir `A[i]` où `A` est un tableau de variables passives, alors tous les pas de la boucle seront détaillés dans l'enregistrement. En revanche, si seuls des tableau actifs sont utilisés dans la boucle,

la boucle sera enregistrée d'une manière compacte, similaire au code source. Par exemple, si on écrit

```
adouble x = 1.;
double a[10000];
...
for(int i=0; i<100000; i++){
  x *= a[i];
}
```

alors on aura une seule variable active, mais dans l'enregistrement les `a[i]` sont remplacés par leurs valeurs et les 100000 multiplications seront écrites explicitement. Par contre si on écrit

```
adouble x = 1.;
adouble a[100000];
...
for(int i=0; i<100000; i++){
  x *= a[i];
}
```

alors on aura 100001 variables actives mais les 100000 multiplications seront condensées dans l'enregistrement de la même manière que dans le code source.

Modification de l'ordre des opérations ou des opérations.

1. Les opérations mélangeant des variables actives et des variables passives sont la plupart du temps ambiguës pour le compilateur. Il faut alors les décomposer en opérations élémentaires, quitte à définir des variables intermédiaires, et utiliser de préférence les opérateurs `+= -= *= /=`.
2. Il faut regrouper les opérations sur les variables passives pour réduire la taille des enregistrements. Si on écrit

   ```
   adouble a; a << ... ;
   double b = ... , c = ... ;
   adouble d = a * b * c;
   ```

   alors `b` et `c` sont remplacés individuellement par leurs valeurs et on enregistre deux multiplications par des variables passives, comme si on avait développé (`d = a ; d *=b ; d *= c ;`). En revanche, si on écrit

   ```
   adouble a; a << ... ;
   double b = ... , c = ... ;
   adouble d = a * ( b * c ) ;
   ```

   ou

   ```
   adouble a; a << ... ;
   double b = ... , c = ... ;
   adouble d = b * c * a ;
   ```

alors `( b * c )` est remplacé par sa valeur et on enregistre une seule multiplication.

3. Nous avons constaté que les divisions prenaient beaucoup de place dans les enregistrements. On peut réduire la taille des enregistrements en remplaçant par exemple `1./sqrt(a)` par `pow(a,-0.5)`, ou
`for(i=0 ; i<n ; i++) b[i] /= a ;`
par `inva = 1./a ; for(i=0 ; i<n ; i++) b[i] *= inva`.
On sait que la deuxième version est moins coûteuse en temps de calcul, mais c'est aussi un compromis avec la précision.

Structures conditionnelles. Nous devons utiliser un opérateur spécifique pour certaines structures `if` : la procédure `condassign`, calquée sur la structure `C ...=... ?... :...`, permet que les enregistrements restent valables pour une plus grande plage de valeurs d'entrée (voir la section 4.2.2.2).

Bibliothèques extérieures. Nous utilisons la bibliothèque d'algèbre linéaire `Lapack` (solveurs linéaires non itératifs pour les matrices pleines). C'est une bibliothèque `Fortran`. La façon classique pour l'utiliser dans les programmes `C` est de définir une interface pour le `C / C++` avec les objets compilés en `Fortran`. Ceci interdit l'utiliser les variables actives : la bibliothèque est compilée en `Fortran` pour fonctionner uniquement avec des scalaires de type `float` ou `double`. Il existe une traduction `C` de cette bibliothèque (`clapack`) dans laquelle nous avons prélevé et adapté uniquement les fonctions que nous utilisions.

Codes `C`. Notons enfin qu'il peut être plus compliqué de recycler pour `ADOL-C` des programmes `C` que des programme `C++`. Par exemple, `ADOL-C` surcharge les fonctions `std : :cin, std : :cout, std : :cerr` (lecture sur l'entrée standard, écriture sur la sortie standard). Leurs équivalents `scanf, print, printf` de la bibliothèque standard `C`, demandant une spécification de format, ne sont pas surchargés.

6.3.2 Méthodes itératives

La validité de la différentiation de méthodes itératives, intervenant par exemple dans la résolution de grands systèmes linéaires, n'est pas une évidence. En particulier, le nombre d'itérations correspondant à la « résolution automatique » des systèmes adjoints ou dérivés sera toujours le même que pour le calcul direct : nous ne pouvons en effet pas utiliser de structure comme `condassign` pour les boucles `while`. Si on décide tout de même de la mettre en œuvre, on rencontre des difficultés : une nouvelle implémentation générique ou typée avec `adouble` est envisageable pour des méthodes simples (gradient conjugué pour `DarcySolver`, mais il ne conviendra pas pour le programme de transport où l'on doit inverser des matrices non symétriques définies positives). On peut tenter une recompilation des bibliothèques avec `adouble` (ceci n'a pas abouti pour `Aztec`). Il faut donc tenter d'isoler de telles opérations pour les différentier manuellement.

```
void main(){
 GetPot data_file(``data'');
 DarcySolver pb(data_file);
 pb.computeHybridMatrix();
 pb.applyBC();
 pb.solveDarcy();
 pb.computePresFlux();
 for(UInt i=0;i<pb.dimVdof;i++)
  pb.globalFlux.vec()(i) *= (-pb.diff_coef);
 pb.postProcess();
}
```

TAB. 6.1 – Code source simplifié pour la résolution de l'équation de Darcy.

La résolution du système linéaire de l'équation (2.51) modifie une variable de classe `L` représentant l'inconnue L et utilise les variables de classe `M` représentant M et `R` représentant R. Elle est est effectuée par une méthode `solveDarcy`. On résume dans le Tableau 6.1 le code source permettant de résoudre l'équation de Darcy, sans dérivation. On résume dans le Tableau 6.2 le code source permettant de résoudre l'équation de Darcy et de calculer une dérivée avec `ADOL-C`.

On résume dans le Tableau 6.3 les modifications à effectuer pour dériver à la main le système linéaire, et dériver tout le reste avec `ADOL-C` (mode direct) : on doit alors partager le programme principal. Il s'agit donc de combiner judicieusement différentiations manuelle et automatique. Ceci demande d'abord une bonne organisation des variables intermédiaires.

6.3.3 Remarque sur l'installation de la bibliothèque

Si des programmes utilisent à la fois les bibliothèques `ADOL-C` et `Lapack`, il faut modifier, avant compilation d'`ADOL-C`, dans les fichiers
`adolc-*/examples/additional_examples/taylor/trigger.cpp`
`adolc-*/adolc/drivers/taylor.c`
`adolc-*/adolc/drivers/taylor.h`
le nom de la fonction `address`, qui existe à la fois dans `ADOL-C` et dans `Lapack` (ces deux bibliothèque n'utilisent malheureusement pas de `namespace`).

```
void main(){
  GetPot data_file(``data'')
  DarcyHandler pb0(data_file);
  int m=pb0.dimPdof+pb0.dimVdof;//nombre de valeurs de retour
  int n=1;//nombre de variables independantes
  int p=n;//plus grand degre de derivation
  //Declarations des variables non actives :
  double* x = new double[n];//pour arguments
  double* y = new double[m];//pour valeurs de retour
  double** XF;//derivees des arguments
  int i;
  XF = new double* [n];for(i=0;i<n;i++) XF[i] = new double [p];
  double** YF;//derivees des valeurs de retour (mode direct)
  YF = new double* [m];for(i=0;i<m;i++) YF[i] = new double [p];
  short int tag=0;//identificateur de l'enregistrement
  trace_on(tag);//mot cle de debut d'enregistrement
  adouble PERM;//declaration d'une variable active
  PERM<<=perm;//initialisation des variables independantes
  //instructions (!! des adouble sont declares dans cette partie) :
   DarcySolver pb(data_file,PERM);  pb.computeHybridMatrix();
   pb.applyBC();  pb.solveDarcy();  pb.computePresFlux();
   for(i=0;i<pb.dimVdof;i++) pb.globalFlux.vec()(i) *= (-PERM);
   //definition des valeurs de retour :
  for(i=0;i<pb.dimPdof;i++) pb.globalP.vec()(i)>>=y[i];
  for(i=0;i<pb.dimVdof;i++) pb.globalFlux.vec()(i)>>=y[i+pb.dimPdof];
  trace_off(); //mot cle de fin d'enregistrement
  //initialisation des variables independantes (forme inactive) :
  x[0]=perm;
  double deltaperm=data_file("physics/delta_diffusion_coef",1.);
  XF[0][0]=deltaperm;
  //appel des routines adolC, mode direct (par exemple) :
  int FORWARD=forward(tag,m,n,p,x,XF,y,YF);
  //initialisation appel des routines adolC, mode adjoint (par exemple) :
  double** z=new double*[m];
  for(int mm=0;mm<m;mm++) z[mm]=new double[n];
  for(int mm=0;mm<m;mm++){
    double* u=new double[m];
    for(int k=0;k<m;k++) u[k]=0.;u[mm]=1.;
     vec_jac(0,m,n,repeat,x,u,z[mm]);
  }
```

TAB. 6.2 – Code source simplifié pour la résolution de l'équation de Darcy et une dérivation par ADOL-C.

1. `trace_on(0)`
   `PERM<<= x[0]`
   `...`
   `...>>=y0[i]`
   `...`
   `trace_off(0)`
2. extraire `R`, `dR`, `M`, `dM` de `y0`,dy0
3. $L = M^{-1}R$

4. construire `y1` et `dy1`
5. `trace_on(1)`
   `...`
   `...<<= y1[i]`
   `...`
   `...>>=U[i]`
   `...`
   `trace_off(1)`
6. `forward(` *0,m0,n,p,x,dx,y0,dy0*)
7. extraire `R`, `dR`, `M`, `dM` de `y0`,dy0
8. $L = M^{-1}R$
   $\mathrm{d}L = M^{-1}\left(\mathrm{d}R - (\mathrm{d}M)L\right)$
9. construire `y1` et `dy1`
10. `forward(` *1,m,m1,p,y1,dy1,y,dy*)

TAB. 6.3 – Code source simplifié pour la résolution de l'équation de Darcy et une dérivation partiellement par `ADOL-C`.

6.3.4 Utilisation combinée de la différentiation manuelle et de la différentiation automatique

Il peut être utile de combiner l'utilisation d'un outil de différentiation automatique et de la différentiation à la main pour deux raisons.

Nous avons vu que certaines routines, comme des routines de résolution de systèmes linéaires, ne doivent pas ou ne peuvent pas être différentiées automatiquement.

On peut utiliser la différentiation automatique comme outil de vérification d'une dérivation à la main. Pour un programme de grande taille, on ne souhaite pas simplement savoir si le résultat d'un calcul est faux : en cas d'erreurs, il faut savoir dans quelles parties du programme les rechercher. Il est donc intéressant de pouvoir vérifier des sous-routines manuelles une par une, les autres routines utilisées dans un programme étant soit déjà vérifiées soient dérivées par différentiation automatique. Une version différentoée totalement automatiquement est vérifiée par comparaison des résultats des calculs de valeurs avec ceux du code non différentié de départ, sur plusieurs exemples, sous l'hypothèse qu'un code automatique qui sait calculer des valeurs correctement sait aussi calculer les dérivées. Les versions manuelles des fonctions ou méthodes peuvent être vérifiées une par une par comparaison des résultats avec ceux obtenus par la version automatique.

Dans l'Annexe A, nous précisons comment nous avons mis en place une différentiation combinée pour le programme de résolution de l'équation de transport.

6.4 Paradigme pour l'implémentation d'algorithmes complexes

Un programme informatique peut souvent être séparé en une partie algorithmique générique, éventuellement très complexe à mettre en œuvre, et une partie calculatoire spécifique, éventuellement lourde mais proposée en grande partie dans des bibliothèques de calcul scientifique. Nous proposons dans cette section un paradigme pour ne pas avoir à reprogrammer les parties génériques pour toutes les applications spécifiques.

L'idée générale est d'utiliser les capacités d'expressivité et la sécurité de programmation apportées par les langages fonctionnels tels que Caml. Ce langage de haut niveau repose sur des bases théoriques sûres, il offre par exemple l'accès à l'automatisation sûre de la parallélisation d'un programme.

6.4.1 Principe

1. On utilise le langage fonctionnel Caml pour implémenter de façon pleinement générique des algorithmes complexes faisant intervenir des notions de haut niveau.
2. Les parties calculatoires spécifiques lourdes sont effectuées par des programmes extérieurs (des travailleurs ou *worker*s) qui peuvent être implémentés dans n'importe quel langage (pour lequel une bibliothèque de communication est disponible, voir le point 3). En général un langage impératif classique (tel que `Fortran`, `C`, `C++`) est utilisé

pour des raisons d'efficacité, mais surtout pour réutiliser des codes existants. Dans l'exemple de la décomposition de domaine, le calcul des $\mu_i^0 = S_i(\lambda_i)$ est effectué par des travailleurs `C++` (nous réutilisons le solveur classique de l'équation de Darcy, voir la section 6.5.2).

3. Les communications entre le pilote et les programmes extérieurs (communications non parallèles, à l'intérieur d'un même processeur, les communications liées à la parallélisation étant gérées par `OCamlP3l`) suivent un protocole simple et sûr utilisant les canaux d'entrée-sortie standards. On peut fabriquer des *worker*s à partir de programmes préexistant en les encapsulant à l'aide de la bibliothèque `PIO`.
4. Le parallélisme éventuel est géré par `OCamlP3l`(en particulier, le déploiement des processus sur les différents processeurs ainsi que les échanges de données entre processeurs).

Des exemples de l'utilisation de ce paradigme sont une plate-forme de couplage de code par décomposition de domaine (autocouplage) et une plate-forme d'analyse de sensibilité déterministe. Il est utilisé dans la thèse [59] pour la décomposition de domaine suivant l'algorithme de Robin non-conforme.

On peut utiliser un couplage à plusieurs niveaux. Par exemple, dans un programme de couplage qui assemble une matrice Jacobienne, le calcul de chaque colonne de matrice Jacobienne peut lui-même utiliser la plate-forme de couplage par décomposition de domaine.

Bibliothèque polyglotte de communication Le protocole d'échange de données est implémenté dans différents langages dans la bibliothèque PIO (pour « Polyglot Input/Output »). Il existe actuellement des versions pour les langages `Caml`, `C++`, `C` et `Fortran`.

6.4.2 Structure générale

La structure générale d'une application utilisant une plate-forme générique est la suivante, illustrée par la Figure 6.1 pour le cas d'un travailleur `C++`.

1. Le programme maître (pilote, ou driver) est écrit en Caml. Il implémente la partie générique de l'application, i.e. la plate-forme. Il se charge du lancement de l'exécution du programme esclave, puis il utilise la version `Caml` de la bibliothèque `PIO` pour envoyer des ordres au travailleur et en recevoir les réponses. Seuls les développeurs de la plate-forme ont à y intervenir.
2. Le programme esclave (travailleur, ou worker) est écrit dans l'un des langages prévus. Il implémente la partie calculatoire spécifique de l'application. Il se présente sous la forme d'un serveur de calculs, c'est-à-dire qu'il entre dans une boucle infinie en attente de stimuli. Il utilise la version appropriée de la bibliothèque `PIO` pour recevoir les ordres du pilote et lui renvoyer ses réponses. Pour une plate-forme donnée, le programme principal du travailleur est lui-même générique et peut être fourni par les développeurs de la plate-forme. Au final, l'utilisateur de la plate-forme n'a plus à fournir que la bibliothèque de calculs spécifiques écrite dans son langage favori.

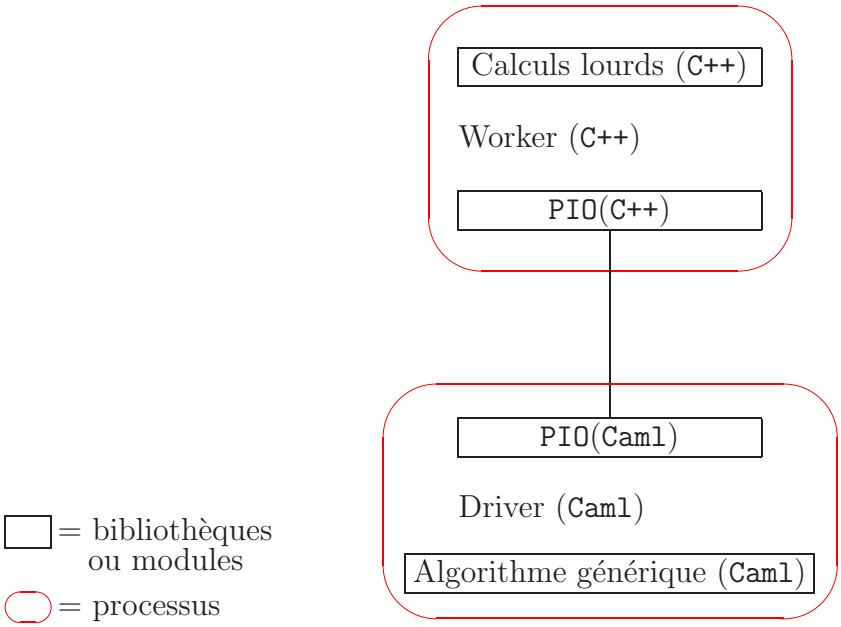


FIG. 6.1 – Couplage de codes simple pour C++.

6.4.3 Parallélisation automatique avec OCamlP3l

Paralléliser un programme séquentiel est particulièrement difficile car le processus de parallélisation introduit des désynchronisations dans les calculs. L'ordre de leur exécution peut être décalé, entraînant des résultats imprévisibles, différents à chaque exécution et différents des résultats obtenus par le programme séquentiel d'origine. La mise au point des programmes parallèles est de ce fait un véritable cauchemar. Face à cette difficulté, le système OCamlP3l offre une solution élégante et théoriquement fondée : une sémantique séquentielle qui permet d'écrire un programme dont la version parallèle, générée automatiquement, donnera le même résultat que la version séquentielle. Le programmeur dispose de briques de base du parallélisme, les squelettes, dont le comportement de passage du séquentiel au parallélisme est parfaitement maîtrisé. Une dizaine de squelettes, combinés comme on l'entend à l'intérieur d'un cadre contraint, suffisent pour exprimer l'essentiel du parallélisme utilisé en pratique. Le logiciel OCamlP3l utilise le langage Objective Caml et tous les outils de ce langage fonctionnel de haut niveau sont à la disposition du programmeur. Cette méthode de parallélisation, qui paraît quasi miraculeuse aux utilisateurs habitués aux mois de travail de mise au point des versions parallèles, est le résultat d'une dizaine d'années de travail, d'abord sur le langage Caml dès 1985 puis OCamlP3l développé avec l'université Paris 7 et l'université de Pise.

La spécificité du système OCamlP3l est d'offrir à l'utilisateur (ici le développeur d'une plate-forme implémentant un algorithme générique donné) la capacité de déployer automatiquement à l'exécution différents processus (Caml) sur différents processeurs en suivant le schéma spécifié par les squelettes de parallélisme employés, ainsi que les échanges de données entre ces processus (Caml).

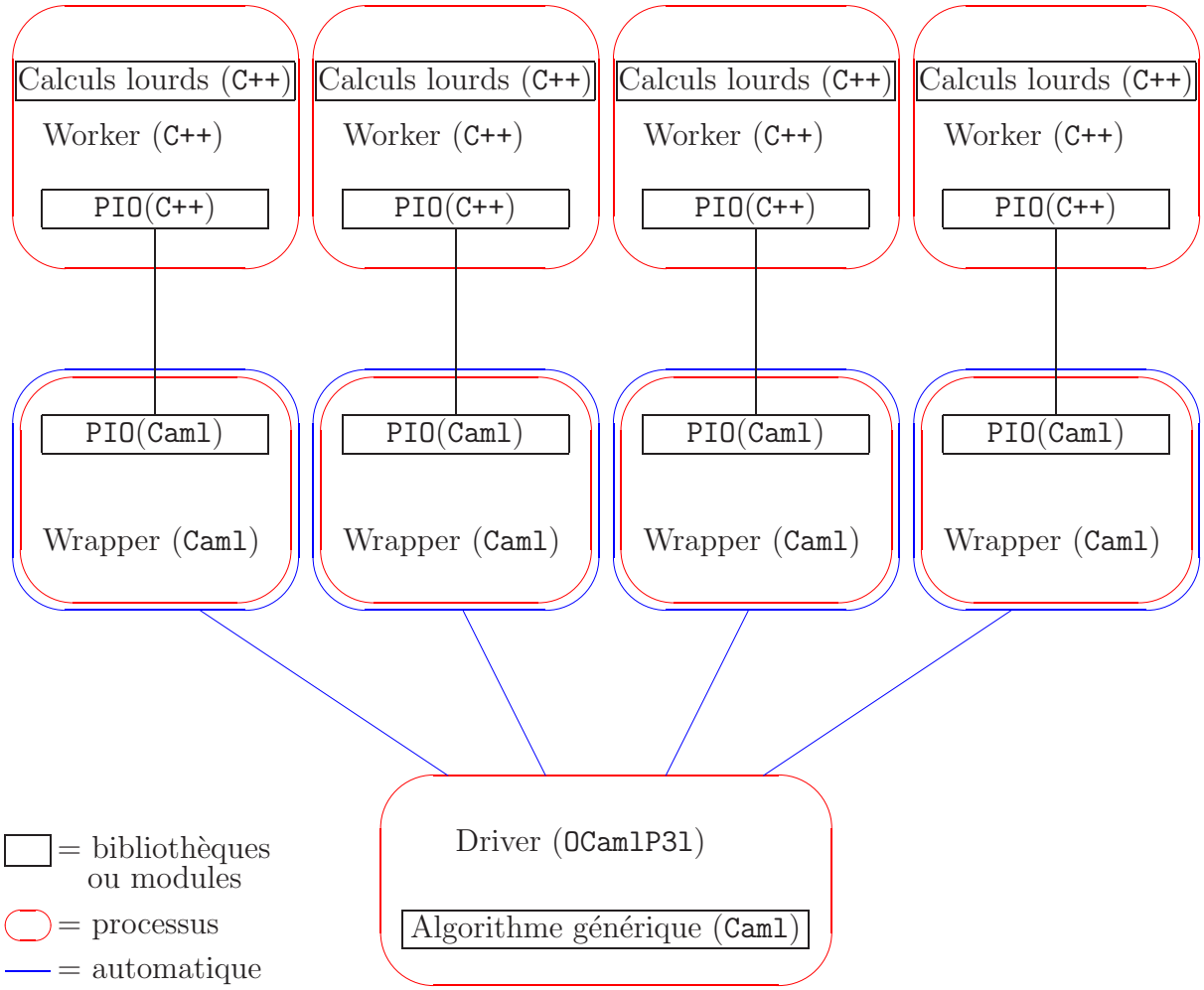


FIG. 6.2 – Couplage de codes parallèle pour C++.

Un squelette est défini comme une fonction

- du **second ordre**, c'est-à-dire une fonction de fonctions,
- **générique**, c'est-à-dire paramétrée par d'autres morceaux de codes,

le point important étant que les squelettes forment une algèbre **compositionnelle**, c'est-à-dire que les squelettes peuvent se combiner entre eux.

La structure générale d'une application parallèle utilisant une plate-forme générique est la suivante, illustrée par la Figure 6.2 pour le C++. Elle peut être vue comme une généralisation du précédent cas simple.

1. Le programme maître (pilote, ou driver) est écrit en Caml. Il implémente la partie générique de l'application, i.e. la plate-forme, au moyen des squelettes de parallélisme d'OCamlP3l. Il se charge de déployer automatiquement les processus « esclaves » (au sens de la parallélisation) sur les différents processeurs disponibles pour le calcul. Seuls les développeurs de la plate-forme ont à y intervenir.

2. Les processus esclaves (appelés ici capsules, ou wrappers) sont écrits en `Caml`. En fait, ils sont obtenus par copie du pilote `Caml`. Ils se chargent du lancement de l'exécution des travailleurs (sur le même processeur qu'eux), puis ils utilisent la version `Caml`de la bibliothèque `PIO`pour envoyer des ordres aux travailleurs et en recevoir les réponses. Encore une fois, seuls les développeurs de la plate-forme ont à y intervenir.
3. Les programmes esclaves (travailleurs, ou workers) sont écrits dans l'un des langages prévus. Ils implémentent la partie calculatoire spécifique de l'application. Ils se présentent sous la forme de serveurs de calculs, c'est-à-dire qu'ils entrent dans une boucle infinie en attente de stimuli. Ils utilisent la version appropriée de la bibliothèque `PIO`pour recevoir les ordres des capsules (wrappers) et lui renvoyer leurs réponses. Pour une plate-forme donnée, les programmes principaux des travailleurs sont eux-même génériques et peuvent être fournis par les développeurs de la plate-forme. Au final, l'utilisateur de la plate-forme n'a plus à fournir que la bibliothèque de calculs spécifiques écrite dans son langage favori.

6.5 Plate-forme de couplage de codes

Nous allons expliquer ici des techniques permettant, d'une part de réutiliser des codes existants pour former des programmes plus complexes (dans notre étude, nous avons un couplage simple entre un code d'écoulement et un code de transport, qui consiste juste à transmettre les résultats du premier code à l'autre), et d'autre part de faciliter la parallélisation, par exemple pour la décomposition de domaine.

6.5.1 Utilisation d'un outil de couplage.

L'utilisateur d'un outil de couplage couple ses codes facilement. Il travaille avec des exécutables indépendants. Son ou ses programmes extérieurs utilisent n'importe quel langage de programmation (pour lequel une version de la bibliothèque de communication existe). Seul son programme principal doit suivre une spécification particulière : c'est un serveur, c'est-à-dire une boucle infinie, qui attend des ordres et retourne des résultats en utilisant la bibliothèque de communication. Il doit être capable de répondre à une liste donnée de tâches, pour être compatible avec l'outil de couplage. Des exemples de serveur sont proposés pour chaque langage. De plus, il n'est pas nécessaire de connaître le langage `Caml` pour utiliser un tel outil de couplage.

6.5.2 Communications pour la décomposition de domaine

Le pilote `Caml` envoie à un travailleur une communication de type `Task`, c'est-à-dire le nom d'une fonction et ses arguments. Il détermine la liste des tâches possibles. Le travailleur reçoit la communication et après calcul renvoie au pilote soit une communication de type `Result`, c'est-à-dire des valeurs de retour, soit une communication de type `Error`, qui demandera au pilote de lancer une exception.

On donne dans le Tableau 6.4 un exemple d'encapsulation de programme `C++`, pour transformer le solveur de l'équation de Darcy en solveur de sous-domaine. n_{sd} versions de ce programme sont lancées par l'outil de couplage dans le cadre d'une décomposition de domaine. On utilise pour résoudre l'équation de Darcy sans décomposition de domaine une classe `DarcySolver`, paramétrée en particulier par des valeurs de conditions aux limites. La classe `SubDomainSolver` dérive de `DarcySolver`. Sa particularité est que ses conditions aux limites sont variables (par contre leur nature, Neumann ou Dirichlet, ne l'est pas).

On a principalement défini une fonction `loop`, qui entre dans une boucle infinie et attend des tâches. La classe `Communication` est définie dans la bibliothèque `PIO` dont on explique l'implémentation dans la section 6.7. L'appel de la méthode `to_task()` provoque le lancement d'une exception de type défini dans la bibliothèque `PIO` si la communication reçue n'est pas de type `Task`. Cette exception doit être rattrapée par le programme principal du travailleur de manière à envoyer une communication de type `Error` sur la sortie standard (elle sera donc reçue par le pilote). Un objet de type `Task` possède comme attribut une chaîne de caractères, renvoyée par la méthode `std : :string name()`, le « nom » de la tâche. Dans cet exemple, trois noms sont acceptés (sinon on envoie une communication de type `Error`). La méthode `args` de la classe `Communication` retourne un objet de type `std : :vector<Typed_value>` contenant les différents arguments de la fonction à appeler : le type `Typed_value` est défini dans la bibliothèque `PIO` et est destiné à représenter des variables de type très général (voir la section 6.7). Les arguments de `iterate` et de `final` représentent des « λ », c'est-à-dire des morceaux de conditions aux limites de Dirichlet. Des convertisseurs sont proposés des types de `PIO` vers des types de la bibliothèque standard `C++` et des types classiques, ce qui permet d'implémenter facilement `init` (initialisation et résolution du problème $\mathcal{P}_i^*$), `iterate` (implémentation de S_i), `final` (résolution de $\mathcal{P}_i$ connaissant λ) à partir des méthodes déjà fournies par la classe `DarcySolver`. La méthode `send_interface_fluxes`, appelée par `init` et `iterate`, provoque l'envoi sur la sortie standard d'une communication de type `Result` contenant des valeurs de flux aux interfaces (μ_i^* pour `init` ou μ_i^0 pour `iterate`). Les vérifications de taille d'argument nécessaires dans ces opérations sont facilitées par les définitions des classes de `PIO`.

6.6 Plate-forme d'analyse de sensibilité déterministe

Nous avons implémenté un pilote d'analyse de sensibilité. On décompose un calcul d'analyse de sensibilité de la manière suivante :

- Initialiser : lire du maillage ; si on veut utiliser le mode inverse, faire un premier calcul de valeur ; retourner la taille $m \times n$ de la matrice Jacobienne
- Calculer m lignes ou n colonnes de la matrice Jacobienne
- Assembler la matrice Jacobienne (fabriquer une matrice à partir de m ou n vecteurs)
- Paramétrer : effectuer un changement de variable sur les entrées et/ou les sorties. En pratique, il s'agit de calculer des produits matrice-vecteur. Les matrices définissant les changements de variables peuvent être données sous forme creuse.
- Calculer la SVD

```
void loop( SubDomainSolver* pb, istream& ib, ostream& ob ) {
  while(true){
           Communication com=receive(ib);
           string fun_name=com.to_task().name();
           if(fun_name == "init" ) pb->init( ob ) ;
           else if(fun_name == "iter") pb->iterate( ob, com.args() );
           else if(fun_name == "final") pb->final( ob, com.args() );
           else throw(Unbound_task(fun_name));
    }
}

SubDomainSolver::iterate( ostream& ob,
vector<Typed_value> args ) {
   build_BC(args);
   applyBC();
   solve();
   compute_fluxes();
   send_interface_fluxes(ob);
}
/*
 preexisting functions    new functions   data types provided by the wrapper
*/
```

Tab. 6.4 – Adaptation d'un programme extérieur.

Le programme principal du pilote a pour argument le nom du programme définissant la fonction que l'on souhaite analyser. Il lance ce programme en redirigeant ses entrées et sorties standards (le langage `Caml` permet d'effectuer ces redirections facilement) puis envoie la tâche `init`, qui devra retourner la taille de la matrice Jacobienne du modèle. Il envoie ensuite les tâches demandant le calcul de lignes ou de colonnes, selon l'option choisie, qui devront retourner des vecteurs. Les différentes lignes ou colonnes peuvent être calculées en parallèle. Il effectue éventuellement un changement de variable. Il assemble la matrice Jacobienne puis demande le calcul de sa décomposition en valeurs singulières, déléguée à un autre programme extérieur, faisant appel à la bibliothèque `Lapack`, selon le même processus.

Le travailleur, un programme `C++` par exemple, doit donc fournir des méthodes de calcul de ligne et/ou de colonne de matrice Jacobienne. En revanche il ne s'occupe pas de son assemblage. Cependant les lignes et les colonnes doivent être numérotées, ce qui n'est pas toujours très naturel (par exemple dans le programme de résolution de l'équation de Darcy qui n'utilise pas cette fonctionnalité, la conductivité hydraulique est supposée constante par zone, et les composantes du vecteur d'entrée de F sont repérées en interne par les numéros de zones et un indice dans la matrice de conductivité hydraulique locale). Pour

la différentiation automatique, cette numérotation existe déjà.

6.7 Implémentation C++ du protocole de communication

Le protocole choisi exige que l'on puisse, dans le cas de la réception, appeler des fonctions avant de connaître la nature des données qui vont être reçues. Il s'agit donc de fonctions abstraites. Le langage C++ est bien approprié à la définition de fonctions génériques. Il permet de définir des types de données (classes) abstraits et des méthodes abstraites sur ces classes, mais par définition les classes abstraites, ou classes *virtuelles*, ne peuvent pas être instanciées : par conséquent quand une fonction est déclarée avec des arguments abstraits, elle est utilisée avec des arguments concrets. On ne peut pas non plus récupérer le résultat d'une fontion déclarée avec un type de retour abstrait sans connaître le type particulier de retour. Avec ce principe, on peut disposer de fonctions et de types de données très généraux, mais il faut encore savoir dans quel cas particulier on se trouve **avant** de les utiliser. Toutes les fonctions doivent **connaître** concrètement leur type au moment où elles sont appelées.

Avec notre protocole de communication, les informations sur la nature des informations (tâche ou donnée, donnée scalaire ou vectorielle, type paramétré scalaire...) reçues sont connues au fur et à mesure de la réception, c'est-à dire après l'appel de la fonction de réception. Nous avons donc besoin pour notre implémentation d'un type général pouvant être instancié : il s'agit d'un type somme général (correspondant en mathématiques à une union d'ensembles). Le langage C fournit un constructeur de type somme appelé `union`, dont on donne un exemple d'utilisation dans le tableau 6.5. Son principe est le suivant : il

```
union union-name {
  public-members-list;
  private:
  private-members-list;
} object-list; // \cite{\urlcppreference}
```

Par exemple :

```
union real {
  int i;
  double d;
  float f;
}
```

Tab. 6.5 – Type somme du langage C.

s'agit d'une structure particulière dont un seul des champs est défini à un instant donné.

Tous les champs partagent le même espace mémoire. La mémoire allouée à l'instanciation d'une `union` est celle nécessaire pour le champ le plus gros. Il faut prévoir hors de l'union une variable indiquant quel champ est actif. L'utilisation des union `C` est réservée aux types simples (entiers, caractères, flottants simple et double précision) : en effet l'appel à des constructeurs à l'instantiation d'une `union` est impossible, car le compilateur doit connaître la taille des champs de l'union. On peut contourner la difficulté en utilisant des `union`s de pointeurs, c'est-à-dire d'entiers, mais on perd alors des fonctionnalités de gestion automatique et d'allocation dynamique de mémoire de la bibliothèque standard `C++`.

Nous avons donc simulé les type sommes généraux fournis par le langage `Caml` (tableau 6.6) en nous appuyant sur la bibliothèque standard `C++`. Un préalable est la simulation

```
type real =
| Int of int
| Float of float ;;
```

Tab. 6.6 – Exemple de type somme en langage `Caml`.

du type pointeur libre. En langage `Caml`, les pointeurs libres ont le type paramétré α `option`. Un `int option` peut prendre par exemple la valeur `None`, ce qui signifie que le pointeur libre contient « rien », ou la valeur `Some 113`, ce qui signifie que le pointeur libre contient un entier de valeur 113. Nous avons donc implémenté une classe `Elem`, dont l'entête est présentée dans le tableau 6.7, qui contient principalement comme attributs privés un vecteur pouvant prendre les tailles 0 ou 1 et un identificateur de l'**état** de l'objet, c'est à dire vide, lorsque vecteur est de taille 0, ou plein. Cet identificateur est redondant avec la taille du vecteur. Des méthodes publiques permettent de lire ou de modifier l'**état** de l'objet :
`bool is()` pour la lecture,
`void Some()`,
`void None()` pour la modification.
Les méthodes de lecture et de modification du **contenu** du vecteur (`... data(...)`) envoient des exceptions si leur appel n'est pas cohérent avec l'**état** de l'objet, c'est-à-dire si `is()==false`. Le constructeur par défaut construit un élément vide.

Une classe représentant un type somme comprend des attributs de type `Elem<...>` paramétrés par les types composant la somme. On doit s'assurer qu'au plus un champ de type `Elem<...>` est non vide à la fois. Un attribut supplémentaire identifie l'éventuel champ actif ou indique que la somme est vide. Les méthodes d'une classe somme auront un comportement particulier dans le cas où la somme est vide. Le constructeur par défaut construit une somme vide qui n'aura pas d'intérêt pratique puisque nous préférons ne pas donner la possibilité de modifier le choix du champ actif d'un objet somme. Nous devons également proposer un constructeur à argument par attribut de type `Elem<...>`. Une classe somme doit aussi proposer des convertisseurs vers des types classiques, qui renvoient des exceptions lorsqu'on appelle un convertisseur incompatible avec le champ actif.

```
template<class dataType> class Elem{
  bool _is;
  vector<dataType> _data;
public:
  Elem(const dataType& x);
  Elem();
  bool is();
  void Some();
  void None();
  void data(dataType x);
  dataType & data();
  const dataType & data();
  virtual ~Elem();
}
```

TAB. 6.7 – Simulation de pointeur libre.

Dans le déroulement d'une fonction de réception de communication, l'acquisition d'informations sur l'objet en cours de réception se traduit par l'appel de constructeurs particuliers d'objets somme (une fonction de réception de communication appelée sans aucune information préalable fabriquera un objet de type somme `Communication`). L'utilisation des données contenues dans une `Communication` demande l'utilisation des convertisseurs. On peut alors soit faire des hypothèses sur la nature des informations contenues dans la communication (dans les cas pratiques, on a souvent une idée), le convertisseur nous avertira si ces hypothèses étaient fausses ; soit conditionner les opérations de conversion et les suivantes par le résultat de la lecture des identificateurs de champs actifs.

La bibliothèque de communication doit aussi permettre de transmettre les données flottantes sous format binaire ou ascii : le format binaire recopie exactement la représentation du flottant en mémoire et empêche donc une perte de précision au cours des communications, tandis que le format ascii est lisible par un humain mais représente une approximation du flottant. Le choix entre ces formats est déterminé par des variable globales. La variable utilisée en écriture est à valeurs modifiables directement par l'utilisateur de la bibliothèque. Le format ascii est normalement à réserver au débogage, donc la valeur par défaut de la variable est `binary`. En général, cette variable est modifiée au plus une fois, au début du programme principal. La variable utilisée en lecture n'est pas modifiable directement par l'utilisateur de la bibliothèque et est modifiée automatiquement selon les informations reçues dans une communication. La technologie utilisée est celle des *singleton pattern*, décrite par exemple dans [24] et [60], P222. Une variable globale est définie à partir de

1. une fonction non exportée, sans argument, définissant une variable statique[1] et re-

[1]Une variable statique est une variable dont il existe toujours exactement une instance pendant

tournant son adresse. Si on écrit

```
int & entier_global(){static int eg=113 ; return eg ;}
```

113 sera simplement la valeur par défaut pour la variable globale ;

2. une fonction sans argument, éventuellement exportée, retournant la valeur de la variable statique. Par exemple

```
int get_entier_global(){return entier_global() ;}
```

3. une fonction éventuellement exportée permettant de modifier la valeur de la variable statique. Par exemple

```
void set_entier_global(int i){entier_global()=i ;}
```

Les point 2 et 3 peuvent être modifiés selon l'utilisation souhaitée. Par exemple le choix ascii/binaire est déterminé par un booléen global et on exporte les fonctions `void set_binary()`, `void set_ascii()` et `bool get_binary()`.

Autres utilisation de la bibliothèque de communication. Cette bibliothèque facilite le découpage d'un code en phase de développement : on peut écrire différents programmes principaux pour différentes briques d'un problème et les deboguer séparément. On utilise la bibliothèque de communication pour enregistrer les sorties d'une brique dans un fichier et faire lire ce fichier par la brique suivante. Ce système fournit la sécurité de la transmission des informations : comme les types et les tailles de données sont systématiquement vérifiés, on évite les erreurs dues à des décalages dans la lecture de fichiers de données.

l'exécution du programme. Une variable statique déclarée dans la définition d'une fonction conserve sa valeur entre les différents appels de la fonction.

Chapitre 7

Applications

7.1 Écoulement tridimensionnel : étude des flux aux exutoires et comparaison avec des résultats probabilistes

7.1.1 Choix des opérateurs de paramétrisation et d'observation

On considère le modèle d'écoulement décrit dans la section 2.1. On s'intéresse à l'effet des incertitudes sur le champ de conductivité hydraulique $\mathcal{K}$ sur le flux de $\vec{u}$ à travers n_{oc} surfaces de $\overline{\Omega}$, ou exutoires S_i, $i = 1, \ldots, n_{oc}$. Ces n_{oc} exutoires étant données, et orientées par les champs de vecteurs normaux $\vec{n}^i$, $i = 1, \ldots, n_{oc}$, les indicateurs de sûreté, ou observations, ou mesures, sont les flux Φ_i, $i = 1, \ldots, n_{oc}$, définis par

$$\text{pour } i = 1, \ldots, n_{oc}, \quad \Phi_i = \int_{S_i} \vec{u} \cdot \vec{n}^i. \tag{7.1}$$

On suppose que le champ de matrices de conductivité hydraulique est uniforme par morceaux. On définit donc n_z zones Ω_α, $\alpha = 1, \ldots, n_z$, ou sous-domaines de Ω, formant une partition de Ω : $\overline{\Omega} = \cup_{\alpha=1}^{n_z} \overline{\Omega}_\alpha$, sur chacune desquelles $\mathcal{K}$ est uniforme : $\mathcal{K}_{|\Omega_\alpha} \equiv \mathbf{K}_\alpha$. On considérera le vecteur de matrice symétriques définies positives $\mathbf{K}$, $\mathbf{K} = (\mathbf{K}_\alpha)_{\alpha=1,\ldots,n_z}$.

Ensuite, on a besoin de choisir une paramétrisation de $\mathbf{K}$, c'est-à-dire de définir un certain nombre de paramètres d'entrée pour la fonction que nous analyserons. Pour tout α, $\alpha = 1, \ldots, n_z$, $\mathbf{K}_\alpha$ est une matrice symétrique de taille 3×3, de sorte que le champ de tenseurs $\mathcal{K}$ est entièrement déterminé par $6n_z$ paramètres, et un choix naturel pour ces paramètres serait les coefficients des moitiés inférieures des n_z matrices. Le nombre de paramètres est réduit si nous faisons des hypothèses supplémentaires sur la forme des matrices. Par exemple, si nous supposons que toutes les matrices sont diagonales, on a besoin de $3n_z$ paramètres, par exemple les termes diagonaux des n_z matrices. Le nombre de paramètres est encore réduit si nous supposons que la conductivité hydraulique est parfaitement connue dans certaines zones : les coefficients de $\mathbf{K}_\alpha$ dans de telles zones seront des paramètres constants et pas des paramètres d'entrée.

Pour le problème considéré ici, on suppose que, pour chaque zone Ω_α, $\mathbf{K}_\alpha$ est soit diagonal soit scalaire, donc déterminé par au plus 3 paramètres. Le nombre total de paramètres d'entrée indépendants n_{ip} vérifie $n_{ip} \leq 3_{nz}$. On note K, $K = (K_j)_{j=1,\ldots,n_{ip}}$, le vecteur contenant ces paramètres. Comme les coefficients de K peuvent varier de plusieurs ordres de grandeur, on utilise une paramétrisation logarithmique. Par conséquent, on choisit comme paramètres d'entrée les logarithmes des K_j et on note κ, $\kappa = (\kappa_j)_{j=1,\ldots,n_{ip}}$, le vecteur de nombres réels défini par

$$\text{pour } j = 1, \ldots, n_{ip}, \quad \kappa_j = \log K_j. \tag{7.2}$$

Nous nous intéressons à la fonction $\mathcal{F}$ qui associe les paramètres de sortie $(\Phi_i)_{i=1,\ldots,n_{oc}}$ aux paramètres d'entrée κ, une fonction de $\mathbb{R}^{n_{ip}}$ dans $\mathbb{R}^{n_{oc}}$. Nous allons en fait analyser sa version discrétisée F qui associe les approximations des flux à travers les exutoires $\tilde{\Phi} = \left(\tilde{\Phi}_i\right)_{i=1,\ldots,n_{oc}}$ aux paramètres d'entrée κ :

$$F = \begin{pmatrix} F_1 \\ \vdots \\ F_{n_{oc}} \end{pmatrix} \tag{7.3}$$

avec, pour $i = 1, \ldots, n_{oc}$,

$$\begin{array}{llll} F_i : & \mathbb{R}^{n_{ip}} & \rightarrow & \mathbb{R}, \\ & \kappa & \mapsto & \tilde{\Phi}_i = \displaystyle\sum_{j \in J | E_j \subset S_i} \vec{\mathrm{n}}_j \cdot \vec{n}^i U_j \end{array} \tag{7.4}$$

où $\vec{\mathrm{n}}_j \cdot \vec{n}^i = \pm 1$ selon que $\vec{\mathrm{n}}_j = \vec{n}^i$ ou $\vec{\mathrm{n}}_j = -\vec{n}^i$. La fonction F se décompose sous la forme

$$F = \mathcal{O} \circ \tilde{F} \circ \mathcal{P} \tag{7.5}$$

où $\mathcal{P}$ est l'opérateur de paramétrisation :

$$\mathcal{P} : \kappa \in \mathbb{R}^{n_{ip}} \mapsto \mathcal{K} \in {\mathcal{M}_{3\times3}^{+}}^{n_c} \tag{7.6}$$

$\mathcal{O}$ est l'opérateur d'observation :

$$\mathcal{O} : \mathcal{U} \in \mathbb{R}^{n_{dof}} \mapsto \tilde{\Phi} \in \mathbb{R}^{n_{oc}} \tag{7.7}$$

et

$$\tilde{F} : \mathcal{K} \in {\mathcal{M}_{3\times3}^{+}}^{n_c} \mapsto \mathcal{U} \in \mathbb{R}^{n_{dof}} \tag{7.8}$$

où $\mathcal{U}$ est solution de l'équation d'état

$$\mathcal{E}_h\left(\mathcal{K}, \mathcal{U}\right) = \vec{0} \tag{7.9}$$

définie par les équations (4.1) à (4.3).

7.1.2 Choix d'une méthode de dérivation

Pour notre exemple nous aurons plus de paramètres d'entrée (i.e. de composantes pour κ) que de paramètres de sortie (i.e. que d'exutoires). Nous choisissons donc une différentiation en mode inverse. Le nombre de variables intermédiaires est trop important pour utiliser la différentiation automatique en mode inverse avec `ADOL-C`. Nous utilisons donc un code différentié manuellement, par la méthode de l'état adjoint.

Dérivation en mode direct. La section 4.3.1 explique comment dériver $\tilde{F}$ en mode direct. Pour calculer F', on utilise la règle de composition, mais on n'assemble pas $\tilde{F}'$. La matrice Jacobienne est calculée colonne par colonne en annulant des variations élémentaires de l'opérateur $\mathcal{E}_h$. On note, pour $j, j' \in \{1, \ldots, n_{ip}\}$, $(\mathrm{d}\kappa^j)_{j'} = \delta_{j,j'}$ et

$$\mathrm{d}\mathcal{K}^j = \frac{\partial \mathcal{K}}{\partial \kappa} \mathrm{d}\kappa^j. \tag{7.10}$$

La $j^{ème}$ colonne de F' est calculée à partir de $\mathrm{d}\mathcal{U}^j$ tel que

$$\frac{\partial \mathcal{E}_h}{\partial \mathcal{K}} \mathrm{d}\mathcal{K}^j + \frac{\partial \mathcal{E}_h}{\partial \mathcal{U}} \mathrm{d}\mathcal{U}^j = \mathbf{0}. \tag{7.11}$$

La $i^{ème}$ composante de cette colonne est $F'_{ij} = \mathrm{d}\tilde{\Phi}_i^j = \sum_{k \in J | E_k \subset S_i} \vec{\mathrm{n}}_k \cdot \vec{n}^i \mathrm{d}\mathcal{U}_k^j$. Ainsi, indépendamment du nombre d'exutoires n_{oc}, il y a n_{ip} systèmes linéaires à résoudre pour calculer la matrice Jacobienne F' de F. La taille des systèmes linéaires est indépendante du nombre de paramètres d'entrée et de sortie (pour un opérateur aux dérivées partielles discret $\mathcal{E}_h$ donné).

Dérivation en mode inverse. La section 4.3.2.2 explique comment dériver F en mode inverse. La matrice Jacobienne est calculée ligne par ligne en utilisant la méthode de l ;état adjoint. Pour $i = 1, \ldots, n_{oc}$, on note par $g_{\tilde{\Phi}}^i$ le vecteur de longueur n_{dof} (la longueur du vecteur $\mathcal{U}$) dont la $k^{ème}$ composante est égale à $\vec{\mathrm{n}}_k \cdot \vec{n}^i$ si E_k existe avec $E_k \subset S_i$ et est égale à 0 sinon. Pour calculer la $i^{ème}$ ligne de F', on utilise les équations (4.13) et (4.14) dans le cas particulier (4.15) avec $[\mathcal{O}'(\mathcal{U}_m)]^T v = g_{\tilde{\Phi}}^i$: nous devons d'abord déterminer l'état adjoint correspondant λ^i en résolvant, selon l'équation (4.23),

$$\begin{pmatrix} A_{\mathcal{K}} & B^T & \tilde{C}^T \\ B & \mathbf{0} & \mathbf{0} \\ \tilde{C} & \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \lambda_U^i \\ \lambda_P^i \\ \lambda_{\tilde{L}}^i \end{pmatrix} = -g_{\tilde{\Phi}}^i. \tag{7.12}$$

Ensuite, la $i^{ème}$ ligne de la matrice Jacobienne de F est

$$g_\kappa^i = \left[\frac{\partial \mathcal{E}_h}{\partial \mathcal{K}} \frac{\partial \mathcal{K}}{\partial \kappa} \right]^T \lambda^i. \tag{7.13}$$

Ainsi, nous devons résoudre un système linéaire par exutoire pour former la matrice Jacobienne de F. Le nombre de systèmes linéaires à résoudre est indépendant de la longueur du vecteur κ. La taille des systèmes à résoudre est la même que pour le mode direct.

Puisqu'il est possible que nous voulions modifier le choix des paramètres d'entrée et de sortie, il est avantageux d'être capable de dériver notre modèle à la fois en mode direct et en mode inverse, et d'utiliser le mode direct lorsque les paramètres de sortie sont les plus nombreux, et le mode inverse lorsque les paramètres d'entrée sont les plus nombreux.

7.1.3 Domaine de calcul

Le modèle est un modèle hydraulique tridimensionnel simplifié, principalement constitué de couches planes parallèles. Le domaine de calcul Ω a une profondeur d'environ 500 mètres et une extension horizontale d'environ 40 kilomètres par 40 kilomètres. Il est divisé en 12 zones Ω_α, $\alpha = 1, \ldots, 12$, chaque zone étant contenue dans une couche. Nous utilisons un maillage d'environ 300 000 cellules, chaque cellule étant incluse dans une zone. Les zones sont représentées sur les deux coupes de la Figure 7.1. Le site de stockage est contenu dans la zone Ω_1 et est représenté en rouge sur la figure. Sur toutes les figures, les dimensions verticales ont été multipliées par 30. La conductivité hydraulique est constante sur

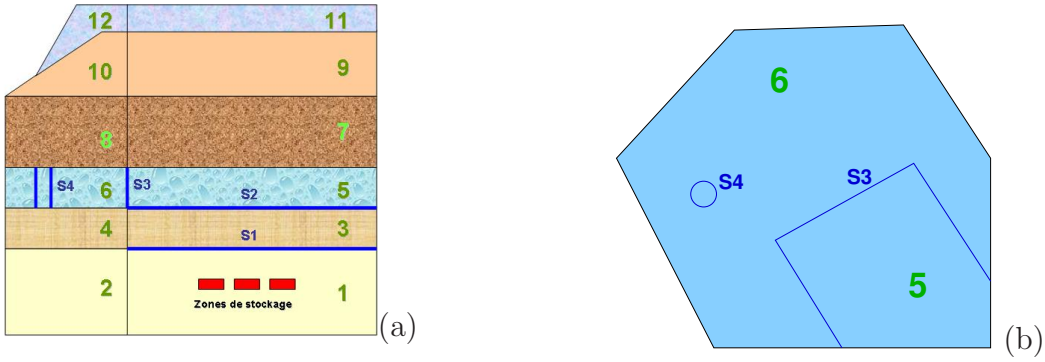


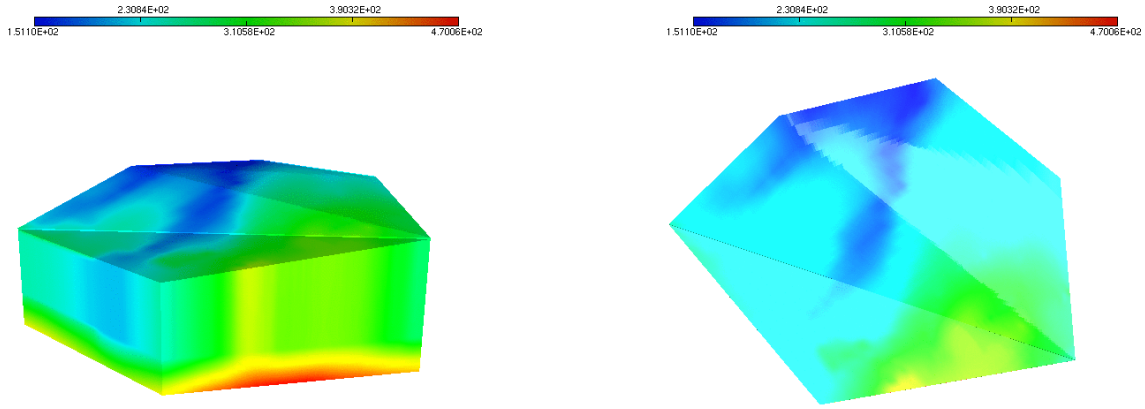
Fig. 7.1 – Section 2D verticale (a) et section 2D horizontale (b) du modèle hydraulique 3D.

chaque zone. La conductivité hydraulique dans les zones Ω_1 et Ω_2 est la même, $\mathbf{K}_1 = \mathbf{K}_2$. Cette matrice est diagonale, avec une composante verticale k_v et deux composantes horizontales égales k_h. Dans les zones Ω_3 à Ω_{12}, la conductivité hydraulique est scalaire : pour $\alpha = 3, \ldots, 12$, $\mathbf{K}_\alpha = k_\alpha \mathbf{I}$, où $\mathbf{I}$ est la matrice identité de taille 3×3. Le vecteur K est de longueur $n_{ip} = 12$ avec $K_1 = k_h$, $K_2 = k_v$ et pour $\alpha = 3, \ldots, 12$, $K_\alpha = k_\alpha$. Sur les figures et les tableaux qui suivront, les deux premières composantes seront indexées par h et v, au lieu d'être indexées par 1 et 2.

Il y a quatre exutoires $(S_i)_{i=1,\ldots,4}$, également représentées, en bleu foncé, sur la figure 7.1. Les exutoires S_1 et S_2 sont des surfaces planes horizontales, coïncidant avec des interfaces entre des zones : $S_1 = \overline{\Omega}_1 \cap \overline{\Omega}_3$ et $S_2 = \overline{\Omega}_3 \cap \overline{\Omega}_5$. L'exutoire S_3 est constituée de 3 surfaces

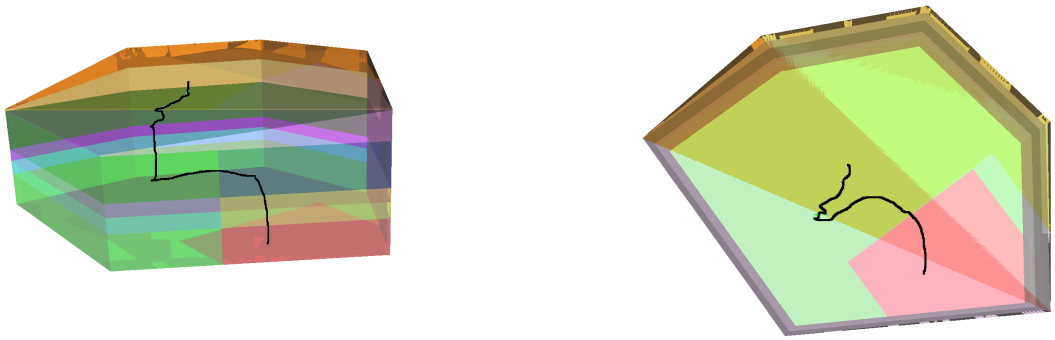
planes verticales, avec $S_3 = \overline{\Omega}_5 \cap \overline{\Omega}_6$. L'exutoire S_4 est la face latérale d'un cylindre à axe vertical, avec $S_4 \subset \Omega_6$.

La Figure 7.2 montre, selon deux perspectives différentes, la distribution de pressions d'eau correspondant à un jeu de conductivités hydrauliques très probable (« best estimate »), dont les valeurs sont données dans les Tableaux 7.2 et 7.3. L'eau s'écoule globalement des zones à plus haute pression (représentées en rouge) vers les zones à plus basse pression (représentées en bleu). La Figure 7.3 représente, pour les mêmes conductivités hydrauliques et selon les mêmes perspectives, une seule ligne de courant partant du site de stockage. Elle passe à travers les exutoires S_1, S_2, S_3, et le long de l'axe du cylindre définissant S_4.



(a) (b)

FIG. 7.2 – Champ de pression pour un jeu de paramètres très probable. (a) : vue de côté ; (b) : vue de dessus.



(a) (b)

FIG. 7.3 – Ligne de courant de la vitesse de l'eau pour un jeu de paramètres très probable. (a) : vue de côté ; (b) : vue de dessus. Le domaine de calcul est transparent et les différentes couleurs de fond correspondent à différentes zones.

7.1.4 Données pour les paramètres d'entrée

On suppose que les conductivités hydrauliques dans les zones 7 à 12 sont précisément connues. Leurs valeurs $k_7, \ldots, k_{12}$ sont données dans le Tableau 7.2. Les autres variables $k_h, k_v, k_3, \ldots, k_6$ sont définies par les lois de probabilité données dans le Tableau 7.1, on voit que certaines sont corrélées. Les valeurs de ces paramètres considérées comme les plus probables sont données dans le Tableau 7.3. Le choix d'une paramétrisation logarithmique pour l'étude déterministe est cohérente avec la description des densités de probabilité par des lois log-normales.

Variable	Vmin	Vref	Vmax	cm	cM
k_v (m/s)	10^{-14}	10^{-13}	10^{-12}	1	99
$\frac{k_h}{k_v}$	10^0	10^1	10^2	5	95
$\frac{k_4}{k_3}$	10^1	10^2	10^3	0.1	0.99
k_5 (m/s)	10^{-9}	10^{-8}	10^{-7}	5	95
$\frac{k_6}{k_5}$	10^1	10^2	10^3	0.1	99.9

(a)

Variable	Vmin	Vmax
$\frac{k_3}{k_v}$	10	1000

(b)

Tab. 7.1 – Conductivités hydrauliques variables dans l'étude probabiliste. (a) : lois log-normales tronquées aux centiles cm et cM, de moyenne Vref, d'écart type tel que la loi charge [Vmin, Vmax] ; (b) : loi uniforme de minimum Vmin, et de maximum Vmax.

k_7	k_8	k_9	k_{10}	k_{11}	k_{12}
10^{-9}	$2\ 10^{-7}$	10^{-12}	10^{-11}	$3\ 10^{-4}$	$3\ 10^{-5}$

Tab. 7.2 – Conductivités hydrauliques invariantes (m/s).

k_h	k_v	k_3	k_4	k_5	k_6
10^{-11}	10^{-13}	10^{-11}	10^{-9}	$8\ 10^{-9}$	$6\ 10^{-7}$

$\tilde{\Phi}_1$	$\tilde{\Phi}_2$	$\tilde{\Phi}_3$	$\tilde{\Phi}_4$
$2.48\ 10^{-5}$	$2.46\ 10^{-5}$	$1.44\ 10^{-4}$	$5.51\ 10^{-3}$

Tab. 7.3 – Valeurs considérées comme les plus probables pour les paramètres variables (m/s) et flux en eau correspondant (m^3/s).

7.1.5 Résultats attendus

Le flux moyen est principalement déterminé par les valeurs des conductivités hydrauliques dans les zones les moins perméables, qui peuvent être considérées comme des bou-

chons, et par les conditions aux limites, dont l'influence n'est pas étudiée ici (voir par exemple pour cela l'article [53]). Les hétérogénéités de la conductivité hydraulique ont une influence sur les variations spatiales du flux : l'eau est en effet attirée par les zones les plus perméables, qui se comportent comme des éponges.

7.1.6 Résultats probabilistes

L'échantillon représentatif de paramètres d'entrée généré pour cette étude effectuée par l'ANDRA est de taille 1000.

La sensibilité de $\tilde{\Phi}_2$ n'a pas été étudiée ici, car il a été d'abord observé que la valeur de $\tilde{\Phi}_2$ était toujours très proche de celle de $\tilde{\Phi}_1$ (on a conservation su flux entre S_1 et S_2). L'indicateur présenté ici est la moyenne des coefficients de monotonie de Spearman (section 3.1.3.2), du PRCC (section 3.1.3.3) et du SRRC (section 3.1.3.4). Les résultats sont donnés dans le Tableau 7.4 : nous observons que la variation de $\tilde{\Phi}_1$ dépend exclusivement de la variation de k_v, la variation de $\tilde{\Phi}_3$ dépend principalement de la variation de k_5 et la variation de $\tilde{\Phi}_4$ dépend exclusivement de la variation de k_6.

$\tilde{\Phi}_1$	indicateur	$\tilde{\Phi}_3$	indicateur	$\tilde{\Phi}_4$	indicateur
(k_h)	0.10	(k_h)	0.03	(k_h)	0.01
$(\mathbf{k_v})$	**1.00**	(k_v)	0.05	(k_v)	0.01
(k_3)	0.08	(k_3)	0.00	(k_3)	0.03
(k_4)	0.02	(k_4)	0.01	(k_4)	0.02
(k_5)	0.12	$(\mathbf{k_5})$	**0.93**	(k_5)	0.14
(k_6)	0.08	$(\mathbf{k_6})$	**0.47**	$(\mathbf{k_6})$	**1.00**

TAB. 7.4 – Indicateur de corrélation statistique entre les flux d'eau et les paramètres d'entrée en régime permanent.

7.1.7 Résultats déterministes et interprétation

La matrice Jacobienne a été calculée selon la méthode de l'état adjoint puisque les sorties sont moins nombreuses que les entrées, en utilisant un code différentié exactement à la main.

L'algorithme de SVD est largement utilisé et plusieurs implémentations existent. Nous utilisons la routine `dgesvd` fournie par la bibliothèque d'algèbre linéaire `Lapack`.

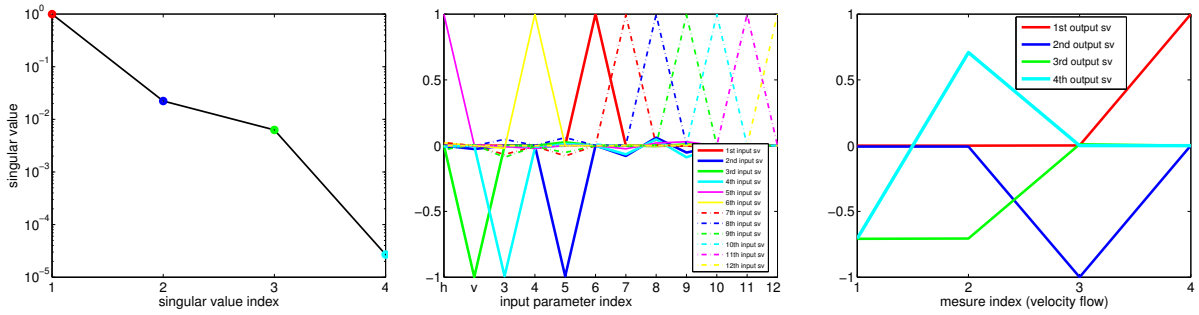
7.1.7.1 Lire les résultats d'analyse de sensibilité

Sur chaque groupe de trois images (Figure 7.4 à Figure 7.8), la première image (a) donne les valeurs singulières, normalisées par rapport à la première. Les coefficients de normalisation sont donnés entre crochets. Nous avons $n_{oc} = 4$ paramètres de sortie (un par exutoire) et $n_{oc} < n_{ip}$ (n_{ip} est le nombre de paramètres d'entrée) donc nous avons 4 valeurs

singulières $\{s_k\}_{k=1,\ldots,4}$, chacune étant représentée par une bille de couleur sur l'image (a). La seconde image (b) donne les vecteurs singuliers dans l'espace d'entrée représentés dans la base canonique correspondant aux composantes de K. Nous avons $n_{ip} = 12$ paramètres d'entrée donc 12 vecteur singuliers $\{v_k\}_{k=1,\ldots,12}$ à 12 composantes dans l'espace d'entrée. Chacune des 12 courbes correspond à un vecteur singulier et chacune des 4 premières courbes est associée avec la bille de la même couleur sur la première image (a). Pour $k > n_{oc} = 4$, une petite variation des paramètres d'entrée selon la direction v_k n'a pas d'influence sur les sorties. Un tel vecteur est dans le noyau de la matrice Jacobienne. La troisième image (c) donne les vecteurs singuliers de l'espace de sortie, représentés dans la base canonique correspondant aux composantes de $\tilde{\Phi}$. Nous avons 4 paramètres de sortie donc 4 vecteurs singuliers $\{u_k\}_{k=1,\ldots,4}$ à 4 composantes dans l'espace de sortie. Chaque courbe correspond à un vecteur singulier et est associée avec la bille de la même couleur sur la première image (a) et avec la courbe de la même couleur sur la deuxième image (b), selon l'équation (3.23).

7.1.7.2 Une première étude locale

Les résultats représentés sur la Figure 7.4 sont obtenus avec les paramètres considérés comme très probables, donnés dans les Tableaux 7.2 et 7.3. Autour de ce jeu de paramètres,



(a) [5.53 10^{-3} m^3/s] (b) (c)

FIG. 7.4 – Résultats de SVD pour les paramètres les plus probables (voir les Tableaux 7.2 et 7.3).

l'influence principale (représentée par la bille et les courbes en rouge) est celle de la conductivité hydraulique dans la zone numéro 6 (la seule composante non nulle de v_1 est la sixième) sur le flux $\tilde{\Phi}_4$ à travers la surface S_4 (la seule composante non nulle de u_1 est la quatrième). Ensuite nous avons (représentée par la bille et les courbes en bleu foncé) l'influence de la conductivité hydraulique dans la zone numéro 5 (la seule composante non nulle de v_2 est la cinquième) sur $\tilde{\Phi}_3$ (la seule composante non nulle de u_3 est la troisième). Ensuite nous avons (représentée par la bille et les courbes en vert) l'influence de la conductivité hydraulique verticale dans la zone de base (la seule composante non nulle de v_3 est la deuxième) sur la somme, ou la moyenne, de $\tilde{\Phi}_1$ et $\tilde{\Phi}_2$ (u_3 a exactement deux composantes non nulles, égales, la première et la deuxième). Enfin nous avons (représentée par la bille et les courbes en

bleu clair) l'influence de la conductivité hydraulique dans la zone 3 (la seule composante non nulle de v_4 est la troisième) sur la différence entre les flux $\tilde{\Phi}_1$ et $\tilde{\Phi}_2$ (en effet, u_4 a exactement deux composantes non nulles, opposées, la première et la deuxième). Puisque la dernière valeur singulière s_4 est négligeable devant les autres, on peut dire que $\tilde{\Phi}_2 - \tilde{\Phi}_2$ est presque invariant pour de petites variations des paramètres d'entrée autour du jeu de paramètres considéré. Les autres paramètres d'entrée ont une influence négligeable sur les flux considérés.

Finalement, on arrive aux mêmes conclusions que par l'étude probabiliste. Cependant, une seule étude locale n'est pas suffisante pour justifier une conclusion, par conséquent nous avons effectué la même étude pour d'autres jeux de paramètres d'entrée.

7.1.7.3 Variabilité des résultats d'analyse de sensibilité lorsque l'on fait varier le jeu de paramètres d'entrée

Les autres résultats de SVD présentés correspondent à des jeux de paramètres beaucoup moins probables. Nous avons d'abord calculé 12 matrices Jacobiennes supplémentaires ainsi que les décompositions en valeurs singulières correspondantes, en choisissant la valeur la plus probable pour tous les paramètres sauf un, pour lequel nous choisissons la valeur la plus petite ou la plus grande possible (en respectant les corrélations entre paramètres). Ensuite, nous avons exploré les sommets de l'ensemble des valeurs possibles pour les conductivités hydrauliques (cet ensemble est un polyèdre), c'est-à-dire que nous avons choisi pour chaque paramètre variable la plus petite ou la plus grande valeur possible, en vérifiant les corrélations, données dans le Tableau 7.5. Ceci signifie $2^6 = 64$ calculs de matrices Jacobiennes et décompositions en valeurs singulières supplémentaires. Quelques

k_v	$\in$	$\{10^{-14}, 10^{-12}\}$
k_h	$\in$	$\{k_v, 10^2 k_v\}$
k_3	$\in$	$\{10^{-12}, 10^{-10}\}$
k_4	$\in$	$\{10k_3, 10^3 k_3\}$
k_5	$\in$	$\{10^{-9}, 10^{-7}\}$
k_6	$\in$	$\{10k_5, 10^3 k_5\}$

TAB. 7.5 – Choix des paramètres variables pour les analyses déterministes.

échantillons des résultat obtenus sont donnés dans les Figures 7.5 à 7.8. Nous résumons ici les principales tendances observées (**liste exhaustive**). Tous les résultats des 12 + 64 tests sont donnés en Annexes B.1 et B.2.

1. Dans la plupart des cas, les résultats sont similaires à ceux obtenus pour la première étude locale, c'est-à-dire que la liste hiérarchisée des influences locales est

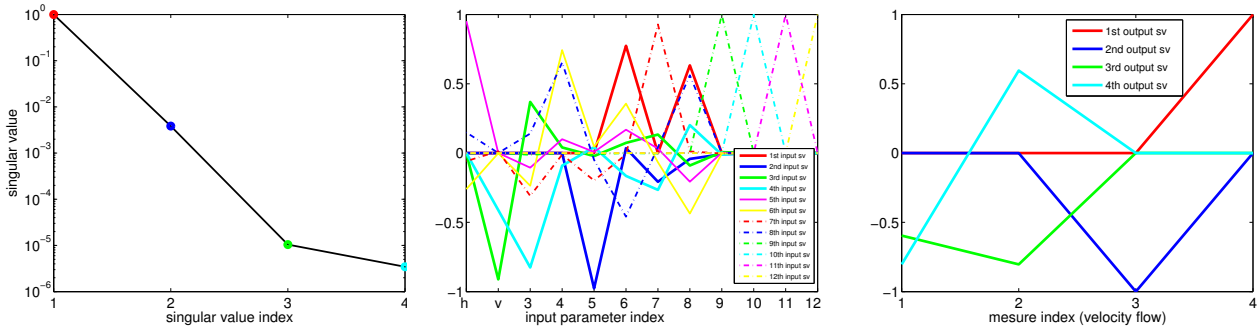
$$(\log k_6 \text{ sur } \tilde{\Phi}_4; \log k_5 \text{ sur } \tilde{\Phi}_3; \log k_v \text{ sur } \tilde{\Phi}_1 + \tilde{\Phi}_2; \log k_3 \text{ sur } \tilde{\Phi}_1 - \tilde{\Phi}_2).$$

2. L'influence locale de la conductivité hydraulique dans la zone de base $\Omega_1 \cup \Omega_2$ dans les résultats de sensibilité semble rester presque nulle pour tous les jeux de paramètres possibles.
3. Dans certains cas, on observe une modification de la hiérarchie des influences locales :
   (a) Chaque fois que $\frac{k_5}{k_v} = 10^3$ (c'est la plus petite valeur possible), l'influence locale de $\log k_v$ sur $\tilde{\Phi}_1 + \tilde{\Phi}_2$ est plus grande que l'influence locale de $\log k_5$ sur $\tilde{\Phi}_3$. Voir les Figures 7.7 et 7.8.
   (b) Chaque fois que $\frac{k_6}{k_v} = 10^4$ (c'est la plus petite valeur possible), l'influence locale de $\log k_v$ sur $\tilde{\Phi}_1 + \tilde{\Phi}_2$ est plus grande que l'influence locale de $\log k_6$ sur $\tilde{\Phi}_4$. Voir la Figure 7.8.
4. Dans certains cas, on observe une modification de la composition des vecteurs singuliers :
   (a) Chaque fois que $k_6 = 10^{-4}$ (c'est la plus grande valeur possible), $\log k_6$ et $\log k_8$ ont tous les deux une influence locale sur $\tilde{\Phi}_4$. Voir les Figures 7.5 et 7.6.
   (b) Chaque fois que $\frac{k_3}{k_v} = 1$ (c'est la plus petite valeur possible), nous observons une influence de $\log k_v + \alpha \log k_3$ sur $\tilde{\Phi}_1 + \tilde{\Phi}_2$, où $\alpha \approx 0.5$, au lieu d'observer une influence de $\log k_v$ sur $\tilde{\Phi}_1 + \tilde{\Phi}_2$. Voir les Figures 7.6 et 7.8.
   (c) Chaque fois que $\frac{k_3}{k_v} = 10^4$ (c'est la plus grande valeur possible), on observe une influence de $\log k_v - \alpha \log k_3$ sur $\tilde{\Phi}_1 + \tilde{\Phi}_2$, où $\alpha \approx 0.5$, au lieu d'observer une influence de $\log k_v$ sur $\tilde{\Phi}_1 + \tilde{\Phi}_2$. Voir la Figure 7.5.
   (d) Le paramètre k_4 a une (très faible) influence lorsque $\frac{k_4}{k_3}$ et $\frac{k_4}{k_v}$ sont tous les deux les plus petits possibles. Voir la Figure 7.8.

7.1.7.4 Utiliser les résultats d'analyse de sensibilité

Nous considérons la linéarisation de notre modèle autour du jeu de paramètres le plus probable. Si les flux à travers chacune des exutoires ont la même importance, les paramètres d'entrée les plus importants à bien déterminer (pour réduire la variance des paramètres de sortie par exemple) sont $\log k_v$, $\log k_5$ et $\log k_6$. Le paramètre de sortie $\tilde{\Phi}_3$ (resp. $\tilde{\Phi}_4$) pourrait être contrôlé localement, sans modifier les autres paramètres de sortie, en modifiant simplement la valeur de $\log k_5$ (resp. $\log k_6$). Le paramètre de sortie $\tilde{\Phi}_1$ pourrait être contrôlé localement, sans modifier les autres paramètres de sortie, en modifiant les valeurs de $\log k_v$ et de $\log k_3$ avec le même signe. Le rapport entre ces modifications dépend des valeurs singulières suivant l'équation $s_3 \Delta(\log k_v) - s_4 \Delta(\log k_3) = 0$, de manière à imposer $\Delta \tilde{\Phi}_2 = 0$. Le paramètre de sortie $\tilde{\Phi}_2$ pourrait être contrôlé localement, sans modifier les autres paramètres de sortie, en modifiant les valeurs de $\log k_v$ et $\log k_3$ selon des signes opposés, en

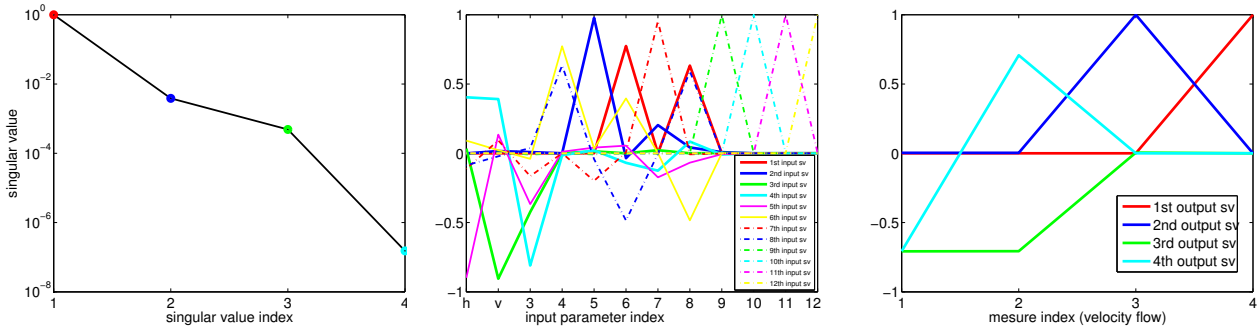
zone n^o	1, 2	3	4	5	6	7	8	9	10	11	12
k_α or $\mathbf{K}_\alpha$(m/s)	$k_h = 10^{-14}, k_v = 10^{-14}$	10^{-10}	10^{-9}	10^{-7}	10^{-4}	10^{-9}	$2\ 10^{-7}$	10^{-12}	10^{-11}	$3\ 10^{-4}$	$3\ 10^{-5}$



(a) [3.78 10^{-1} m^3/s] (b) (c)

FIG. 7.5 – Exemple de résultats de SVD, avec k_6 et $\dfrac{k_3}{k_v}$ prenant leur plus grande valeur possible.

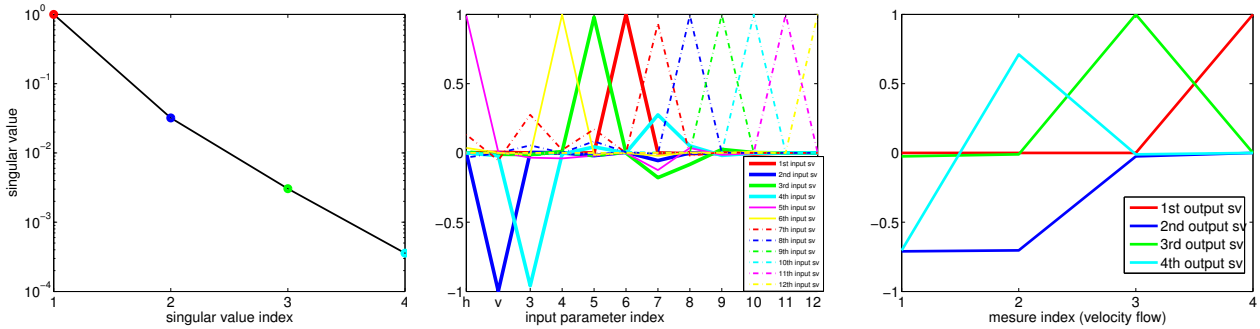
zone n^o	1, 2	3	4	5	6	7	8	9	10	11	12
k_α or $\mathbf{K}_\alpha$(m/s)	$k_h = 10^{-10}, k_v = 10^{-12}$	10^{-12}	10^{-9}	10^{-7}	10^{-4}	10^{-9}	$2\ 10^{-7}$	10^{-12}	10^{-11}	$3\ 10^{-4}$	$3\ 10^{-5}$



(a) [3.78 10^{-1} m^3/s] (b) (c)

FIG. 7.6 – Exemple de résultats de SVD, avec $\dfrac{k_3}{k_v}$ prenant sa plus petite valeur possible et k_6 prenant sa plus grande valeur possible.

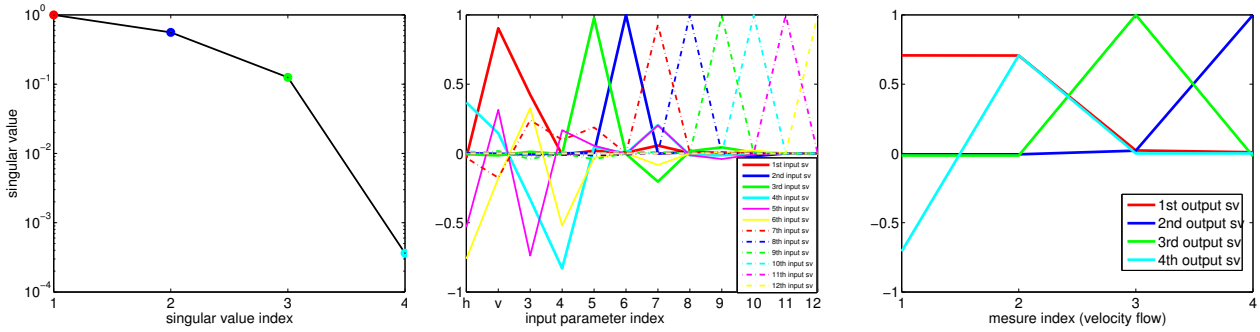
zone n^o	1, 2	3	4	5	6	7	8	9	10	11	12
k_α or $\mathbf{K}_\alpha$(m/s)	$k_h = 10^{-12}, k_v = 10^{-12}$	10^{-10}	10^{-7}	10^{-9}	10^{-6}	10^{-9}	$2\ 10^{-7}$	10^{-12}	10^{-11}	$3\ 10^{-4}$	$3\ 10^{-5}$



(a) [9.29 10^{-3} m^3/s] (b) (c)

FIG. 7.7 – Exemple de résultats de SVD, avec $\dfrac{k_5}{k_v}$ prenant sa plus petite valeur possible.

zone n^o	1, 2	3	4	5	6	7	8	9	10	11	12
k_α or $\mathbf{K}_\alpha$(m/s)	$k_h = 10^{-10}, k_v = 10^{-12}$	10^{-12}	10^{-11}	10^{-9}	10^{-8}	10^{-9}	$2\ 10^{-7}$	10^{-12}	10^{-11}	$3\ 10^{-4}$	$3\ 10^{-5}$



(a) [1.62 10^{-4} m^3/s] (b) (c)

FIG. 7.8 – Exemple de résultats de SVD, avec $\dfrac{k_3}{k_v}$, $\dfrac{k_5}{k_v}$, $\dfrac{k_6}{k_v}$, $\dfrac{k_4}{k_v}$ et $\dfrac{k_4}{k_3}$ prenant leur plus petite valeur possible.

respectant l'égalité $s_3\Delta(\log k_v) + s_4\Delta(\log k_3) = 0$, de manière à imposer $\Delta\tilde{\Phi}_1 = 0$. Puisque s_4 est beaucoup plus petit que s_3, il serait très difficile de contrôler indépendamment $\tilde{\Phi}_1$ et $\tilde{\Phi}_2$.

7.1.8 Conclusions

Plusieurs études déterministes locales ont été réalisées, autour de jeux de paramètres plus ou moins probables. Pour cet exemple, on observe une faible variabilité des influences locales lorsque le choix des paramètres d'entrée varie dans l'ensemble des paramètres possibles. Ceci est dû à la faible non linéarité du modèle, assurée par le choix d'une paramétrisation logarithmique. Les résultats ont été comparés avec ceux obtenus par une étude probabiliste de type Monte-Carlo. Les études probabiliste et déterministe fournissent des résultats similaires, et qui sont cohérents avec ceux attendus pour ce cas-test (section 7.1.5). Notons que dans les cas ou les résultats d'analyse déterministe locale varient beaucoup suivant les paramètres d'entrée, il faudrait aussi se méfier des résultats probabilistes.

7.2 Écoulement tridimensionnel : étude déterministe des variations locales

Nous effectuons des analyses de sensibilité locales, dans le sens où les résultats que nous obtenons sont pertinents à condition que les plages de variation des paramètres soient faibles. Nous employons le terme local dans le titre de cette section dans un autre sens : dans la section 7.1, nous avons étudié la sensibilité du flux total de la vitesse de l'eau à travers des exutoires. Maintenant nous nous intéressons aux variations spatiales, à l'intérieur de petites surfaces, de la vitesse d'écoulement de l'eau.

7.2.1 Description du cas-test

On considère le même problème d'écoulement que dans la section 7.1, mais cette fois ci les sorties de notre fonction sont les flux de $\vec{u}$ à travers certaines faces du maillage. Nous n'avons pas utilisé de paramétrisation logarithmique, en revanche nous avons considéré comme entrée de notre fonction toutes les composantes de la moitié inférieure des tenseurs de perméabilité, même celles extradiagonales (nulles). On peut ainsi mesurer l'influence de l'hypothèse « tenseur diagonal ». Sur toutes les figures, les dimensions verticales sont grossies 100 fois. La Figure 7.9-(a) montre le domaine de calcul avec les conditions aux limites : la pression est imposée sur la partie rouge et la partie verte est imperméable. L'emplacement des composantes de flux choisies est aussi montré sur la figure 7.9-(a).

zone number	1, 2, 14	3	4	5	6	7	8	9	10	11	12	13
permeability m.s^{-1}	$\begin{pmatrix} 10^{-12} & & \\ 0 & 10^{-12} & \\ 0 & 0 & 10^{-14} \end{pmatrix}$	10^{-12}	10^{-10}	8.10^{-9}	6.10^{-7}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}	3.10^{-5}

Tab. 7.6 – Distribution de perméabilité.

La figure 7.9-(b) montre la distribution de pressions correspondante. L'eau s'écoule globalement des hautes pressions (en rouge sur la figure) vers les basses pressions (en bleu sur la figure).

7.2.2 Décomposition en valeurs singulières

Nous avons différentié 12 composantes de flux en fonction de la distribution de perméabilité. Ces 12 composantes sont situées à la surface du domaine, dans la zone numéro 11 qui est la plus perméable. Elles sont représentées en beige sur la Figure 7.9-(a).

La décomposition en valeurs singulières de la matrice Jacobienne a été effectuée en utilisant la méthode de l'état adjoint puisque le problème a moins de sorties que d'entrées et par le code manuel. Trop de place mémoire était nécessaire pour utiliser le code automatique. La figure 7.10 donne les valeurs singulières. Les vecteurs singuliers correspondant sont donnés sur la figure 7.11.

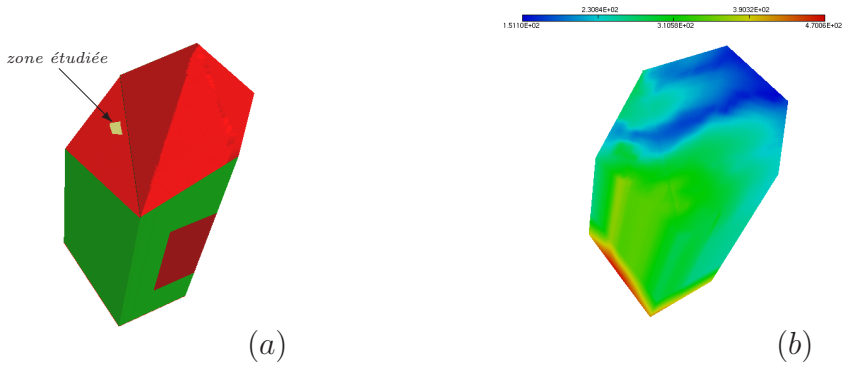


FIG. 7.9 – Domaine de calcul (a) et distribution des pressions, en mètres (b).

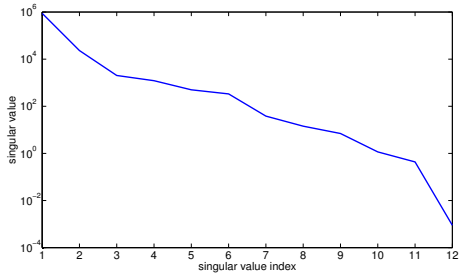
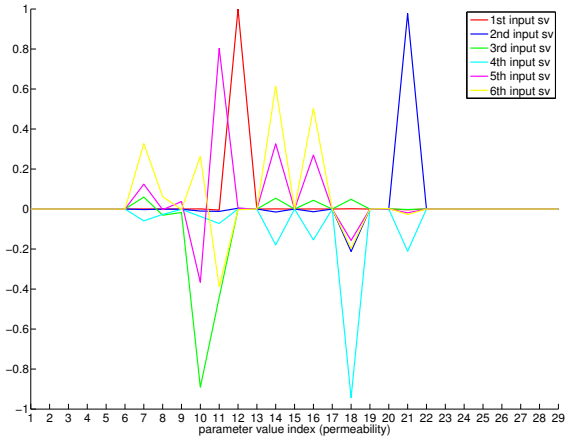


FIG. 7.10 – Les 12 valeurs singulières. Les 6 premières sont dans un rapport de 10^4.

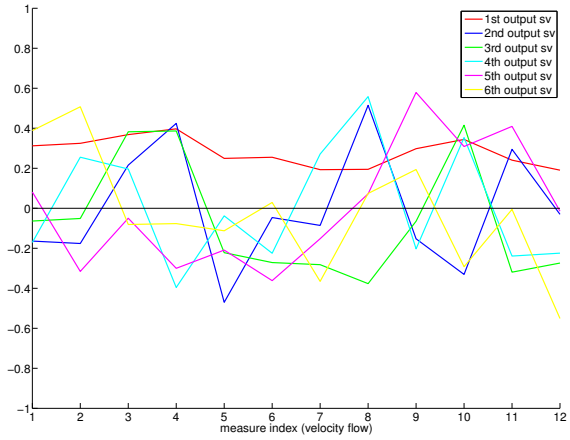
Le premier vecteur singulier de l'espace d'entrée correspond principalement à la 12ème composante dans le vecteur d'entrée, c'est-à-dire à la composante verticale du tenseur de perméabilité dans la zone 2. Cette zone est la plus profonde du domaine et recouvre toute sa base. Il est associé à un vecteur singulier de sortie dont toutes les composantes sont du même ordre : la perméabilité verticale à la base du domaine a une influence sur le flux moyen dans la zone étudiée.

Le second vecteur singulier correspond principalement à la 21ème composante du vecteur d'entrée, c'est-à-dire à la perméabilité scalaire dans la zone 11, où les composantes étudiées sont situées. Les composantes du vecteur singulier de sortie ont des signe variables. La perméabilité locale influence les variations locales de flux.

Les sensibilités par rapport aux perméabilités des zones intermédiaires, beaucoup plus perméables que la zone de base, sont beaucoup plus faibles. Les vitesses de sortie semblent principalement déterminées par les pressions imposées et par la perméabilité de la zone la moins perméable.



(a)



(b)

FIG. 7.11 – Les 6 premiers vecteurs singuliers des espaces d'entrée (a) et de sortie (b).

7.3 Écoulement et transport bidimensionnels : étude des flux aux exutoires et comparaison avec des résultats probabilistes

7.3.1 Choix du terme source

On considère le taux de relâchement de la matrice de verre des colis de déchets τ_r, commun à tous les radionucléides. Il indique la proportion des radionucléides relâchés par la matrice par unité de temps. On considère le stockage comme une source volumique de radionucléides. Le terme source initial de notre modèle pour un radionucléide i considéré écrit

$$q_i = \frac{n_{\text{colis}} \times \text{A}_\text{i} \times (\text{Bq} \to \text{mol})_\text{i}}{V} \tau_r$$

où A_i est l'activité de chaque colis de déchets concernant le radionucléide i, n_{colis} le nombre de colis et V est le volume total du site de stockage et $(\text{Bq} \to \text{mol})_i$ est un coefficient de conversion. La source décroît exponentiellement en temps pour chaque radionucléide considéré, selon le coefficient de décroissance radioactive.

7.3.2 Définition des fonctions à analyser

La fonction à valeurs réelles $\mathcal{F}_i^t$ associe le flux Ψ_i^t calculé selon le modèle de la section 2.2.1 à travers les exutoires S_i à l'instant t aux (logarithmes des) paramètres d'entrée K, Φ, D et q pondérés selon les écarts types probabilistes. On considère aussi sa version discrétisée F_i^t.

$$\mathcal{F}_i^t : \text{paramètres d'entrée} \to \Psi_i^t = \int_{S_i} \left(c\vec{u} - D\vec{\nabla} c \right)^t \cdot \vec{n}^i. \tag{7.14}$$

Notons que

- $\vec{u}$ est la vitesse de Darcy de l'équation d'écoulement
- les paramètres sont supposés constants en temps, constants par morceaux en espace
- les équations sont résolues suivant le schéma décrit dans les sections 2.2.2 et 2.2.3.

On réalise un calcul bidimensionnel sur une coupe non plane des zones numéros 6 à 11 du domaine représenté sur la Figure 7.1, avec 11 paramètres incertains :

- q (terme source)
- Φ constant sur $\Omega_1 \cup \Omega_3 \cup \Omega_4$, noté Φ_{134}
- Φ constant sur $\Omega_5 \cup \Omega_6$, noté Φ_{56}
- D constant $\Omega_1 \cup \Omega_3 \cup \Omega_4$, noté D_{134}
- D constant sur $\Omega_5 \cup \Omega_6$, noté D_{56}.
- $k_h, k_v, k_3, k_4, k_5, k_6$.

On utilise un changement de variable supplémentaire $\mathcal{P}_1$, consistant en une normalisation et éventuellement une transformation logarithmique, présenté dans le Tableau 7.7.

p_1	p_2	donnée probabiliste
$q = 1.5e - 9$	$\dfrac{\ln q}{\sigma(\ln \tau_r)}$	$q = 2.04\ 10^{-5}\tau_r$ avec $\tau_r \rightsquigarrow \ln-\text{normale}(-9.53, 1.07)$
$\Phi_{1347}0.105$	$\dfrac{\Phi_{1347}}{\sigma(\Phi_{1347})}$	loi uniforme (0.01 - 0.2)
$\Phi_{56} = 0.112$	$\dfrac{\Phi_{56}}{\sigma(\Phi_{56})}$	loi uniforme (0.0368 - 0.184)
$D_{1347} = 4\ 10^{-12}$	$\dfrac{D_{1347}}{\sigma(D_{1347})}$	loi normale, corrélation avec $\Phi_{1347} = 0.7$
$D_{56} = 2\ 10^{-9}$	$2\ 10^{-9}$	valeur connue
$k_h = 10^{-12}$	$\dfrac{\ln k_h}{\sigma(\ln k_h)}$	$k_v \times \ln-\text{normale}(\mu = 2.30, \sigma = 1.40)$
$k_v = 10^{-13}$	$\dfrac{\ln k_v}{\sigma(\ln k_v)}$	loi ln-normale ($\mu = -29.93, \sigma = 0.988$)
$k_3 = 4.5\ 10^{-11}$	$\dfrac{\ln k_3}{\sigma(\ln k_3)}$	$k_v \times \text{unif}(\min = 10, \max = 1000)$
k_4=4 10^{-9}	$\dfrac{\ln k_4}{\sigma(\ln k_4)}$	$k_3 \times \ln-\text{normale}(\mu = 4.605, \sigma = 0.743)$
$k_5 = 10^{-8}$	$\dfrac{\ln k_5}{\sigma(\ln k_5)}$	loi ln-normale ($\mu = -18.42, \sigma = 1.354$)
k_6= 2.2 10^{-6}	$\dfrac{\ln k_6}{\sigma(\ln k_6)}$	proba : $k_5 \times \ln-\text{normale}(\mu = 4.605, \sigma = 0.743)$

La constante de décroissance radioactive est fixée, $\lambda = \dfrac{\ln 2}{T} = 4.41\ 10^{-8}$

TAB. 7.7 – Données les plus vraisemblables pour le programme de transport.

On considère pour cinq dates $t_1 = 5\ 10^4$ ans, $t_2 = 10^5$ ans, $t_3 = 3\ 10^5$ ans, $t_4 = 5\ 10^5$ ans et $t_6 = 10^6$ ans, la fonction F^{t_n} qui retourne les quatre flux de contaminants à travers les exutoires $\Psi_1^{t_n}, \Psi_2^{t_n}, \Psi_3^{t_n}, \Psi_4^{t_n}$,

$$F^{t_n} \ : \text{paramètres d'entrée} \rightarrow \left(\Psi_1^{t_n}, \Psi_2^{t_n}, \Psi_3^{t_n}, \Psi_4^{t_n}\right)^T,$$

puis, pour chacun des exutoires S_i, $i = 1, \ldots, 4$, la fonction F_i qui retourne les cinq flux de contaminants $\Psi_i^{t_1}, \Psi_i^{t_2}, \ldots$, à travers l'exutoire S_i aux dates $t_1, \ldots, t_5$,

$$F_i \ : \text{paramètres d'entrée} \rightarrow \left(\Psi_i^{t_1}, \Psi_i^{t_2}, \Psi_i^{t_3}, \Psi_i^{t_4}, \Psi_i^{t_5}\right)^T.$$

Les valeurs des paramètres utilisées pour ces tests sont données dans le Tableau 7.7. L'opérateur de paramétrisation $\mathcal{P}$ consiste à passer des données p_2 aux données p_1 puis aux paramètres continus. On peut décomposer $\mathcal{P} = \mathcal{P}_1 \circ \mathcal{P}_2$ où $\mathcal{P}_2 \ : p_2 \rightarrow p_1$.

7.3.3 Résultats probabilistes

Les courbes de la Figure 7.12 donnent une évaluation de la valeur absolue du PRCC et du SRRC en fonction du temps pour les surface S_1 à S_3. Pour chaque surface, les tendances sont globalement identiques entre PRCC et SRRC : elle permet ainsi de confirmer que les 3 paramètres ϕ_{134}, D_{134} et k_v sont les paramètres dont l'incertitude influe le plus sur l'incertitude du résultat. Pour S_1 et S_2, on remarque une faible influence des paramètres Hp1-Hp4 (i.e. zones numéros 5 et 6) dans le calcul des PRCC, l'influence du coefficient de diffusion diminue en fonction du temps (la part diffusive du flux total de radionucléides diminue avec le temps, au bénéfice de la convection). Pour S_3, on voit logiquement apparaître une influence plus marquée des paramètres de Hp1-Hp4 (notamment pour PRCC) sur les résultats de transfert de solutés, mais cette influence reste limitée au regard des variations des paramètres de la couche d'argilites (zones 1 à 3). Ces résultats sont présentés dans les documents [27, 69].

7.3.4 Résultats déterministes

7.3.4.1 Sortie = flux $(\Psi_1^t, \Psi_2^t, \Psi_3^t, \Psi_4^t)^T$, à une date t donnée

Les résultats sont donnés sur les Figures 7.13 à 7.17.

On retouve les 3 influences identifiées dans l'étude probabiliste : Φ_{134}, D_{134}, et k_v sur les flux à travers toutes les exutoires. On trouve deux autres influences qui semblent non physiques : k_4 et k_5. Il faudrait revoir le choix des pondérations pour des données hétérogènes. De plus contrairement à ce qui se passait pour le problème d'écoulement, les non-linéarités ne sont pas de même nature pour tous les paramètres. Pour remédier à cette difficulté, on pourrait commencer par mener des analyses partielles concernant les paramètres d'entrée de même nature. Ceci n'est cependant sûrement pas suffisant pour expliquer ces résultats, qui sont à considérer avec prudence pour le moment.

7.3.4.2 Sortie = flux $\left(\Psi_i^{t_1}, \Psi_i^{t_2}, \ldots\right)^T$, à travers un exutoire S_i donnée

Les résultats sont donnés sur les Figures 7.18 à 7.21.

La tendance générale observée à exutoire fixé est une influence des paramètres de l'équation de diffusion sur ce qui se passe aux premiers instants et des paramètres de l'équation de Darcy (conductivités hydrauliques) aux dates plus tardives.

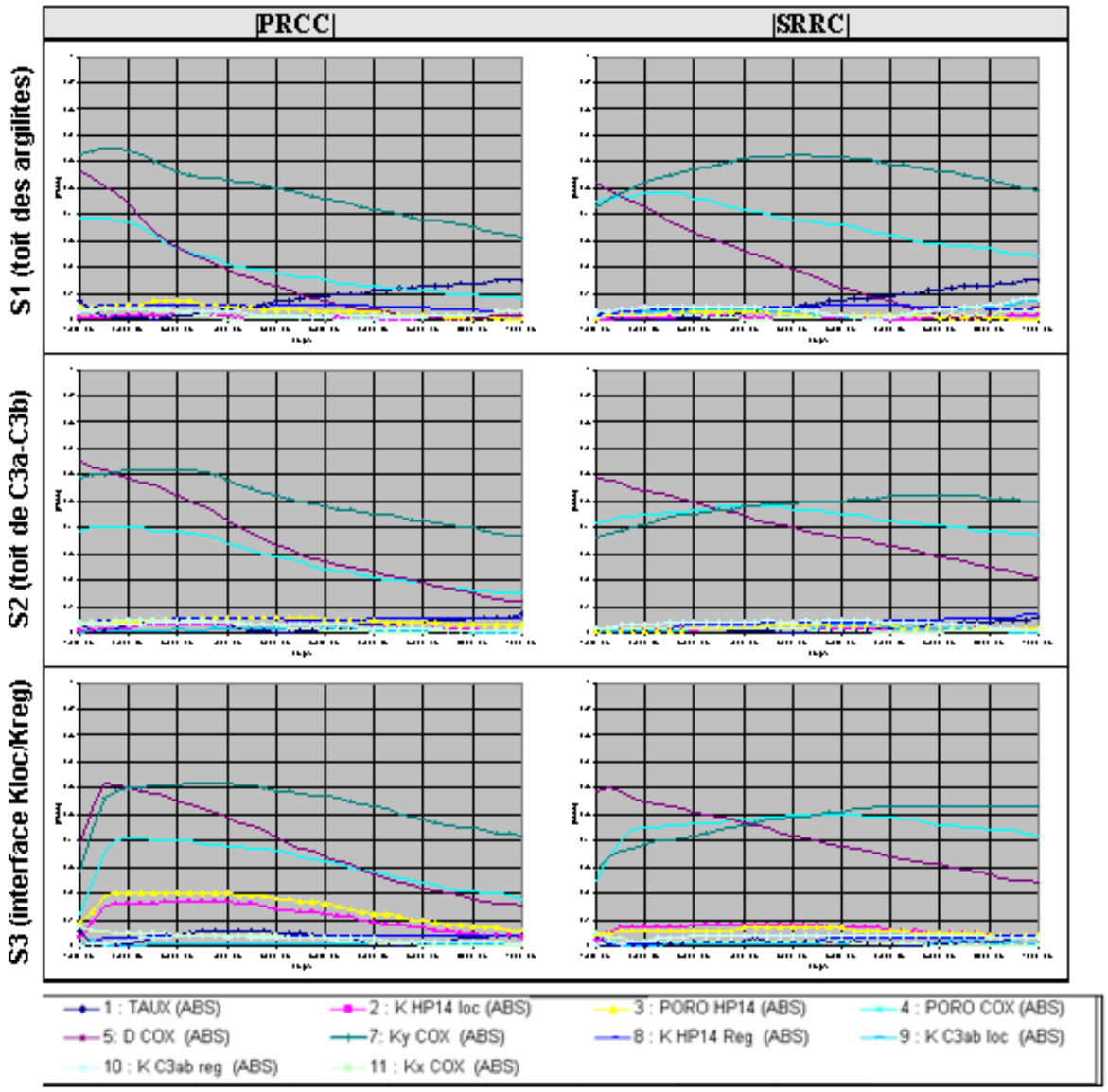


FIG. 7.12 – Résultats d'analyse de sensibilité probabiliste. La couche HP1-HP4 locale est la zone numéro 5, la couche HP1-HP4 est la zone numéro 6, le COX correspond aux zones numéro 1 à 3.

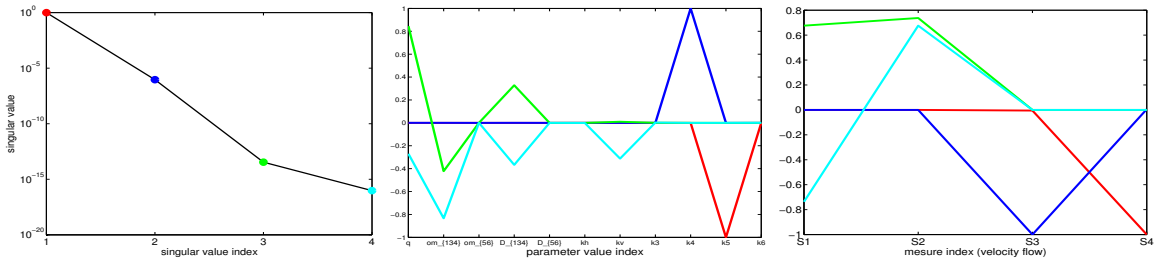


FIG. 7.13 – Date $t_1 = 50\ 000$ ans - maillage fin

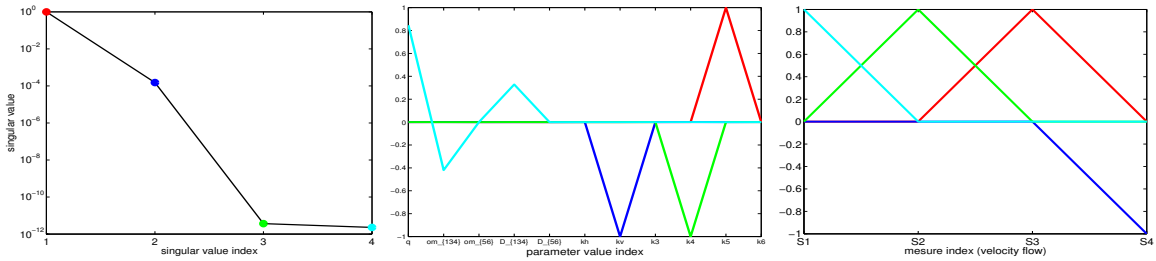


FIG. 7.14 – Date $t_2 = 100\ 000$ ans - maillage fin

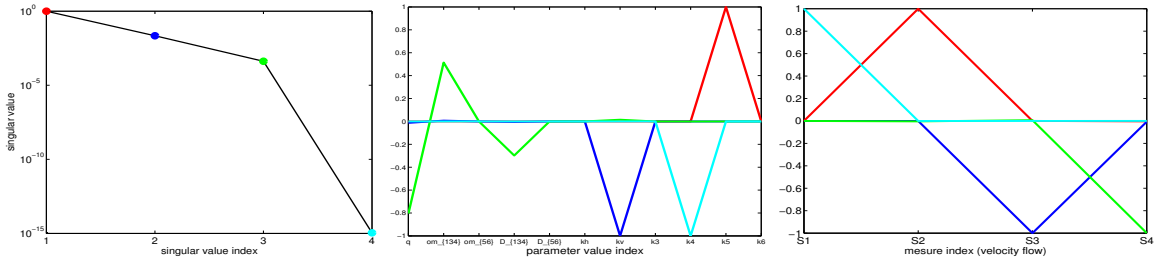


FIG. 7.15 – Date $t_3 = 300\ 000$ ans - maillage fin

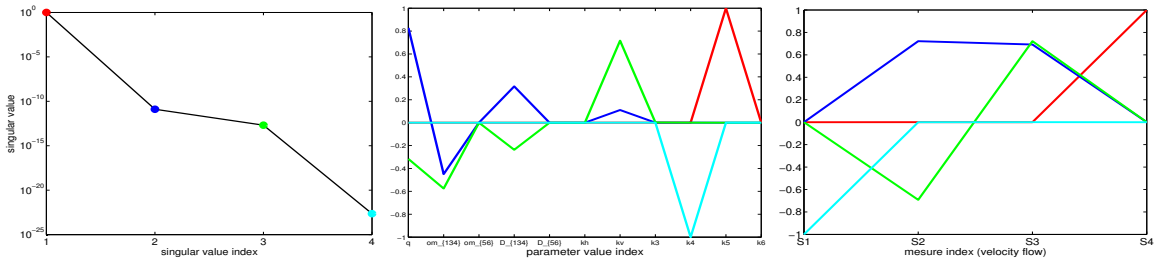


FIG. 7.16 – Date $t_4 = 500\ 000$ ans - maillage fin

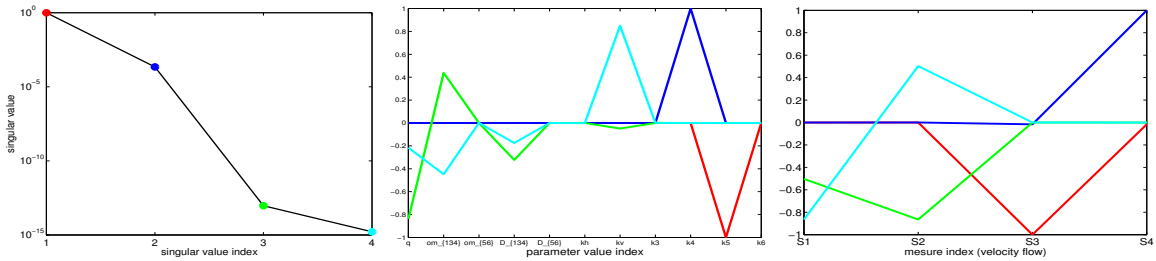


FIG. 7.17 – Date $t_5 = 1\ 000\ 000$ ans - maillage fin

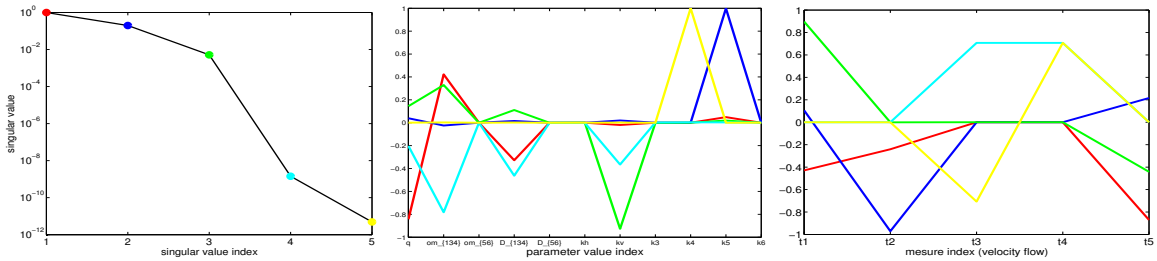


FIG. 7.18 – S_1 - maillage fin

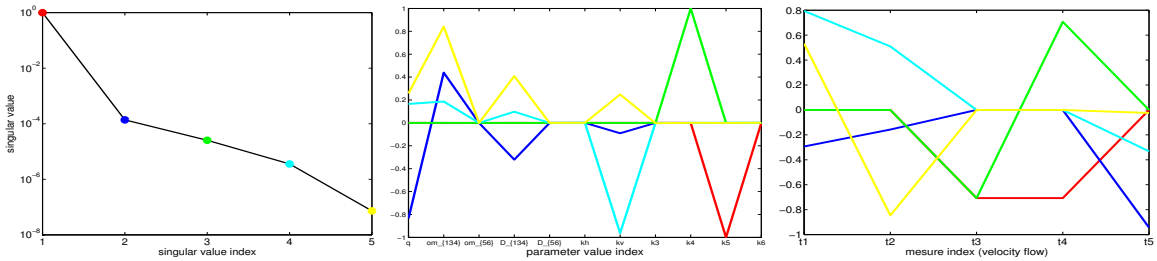


FIG. 7.19 – S_2 - maillage fin

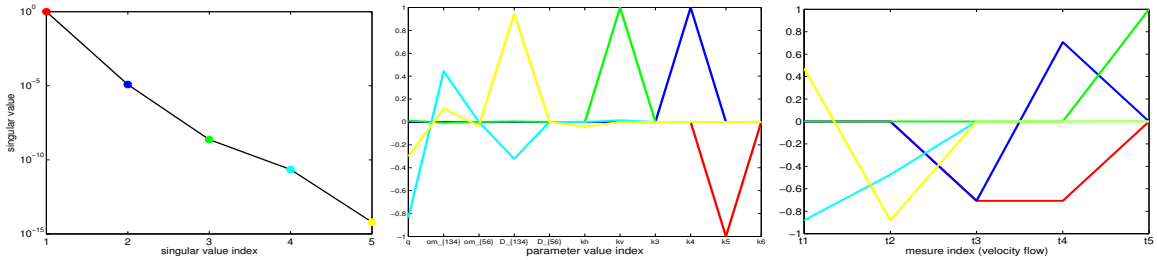


FIG. 7.20 – S_3 - maillage fin

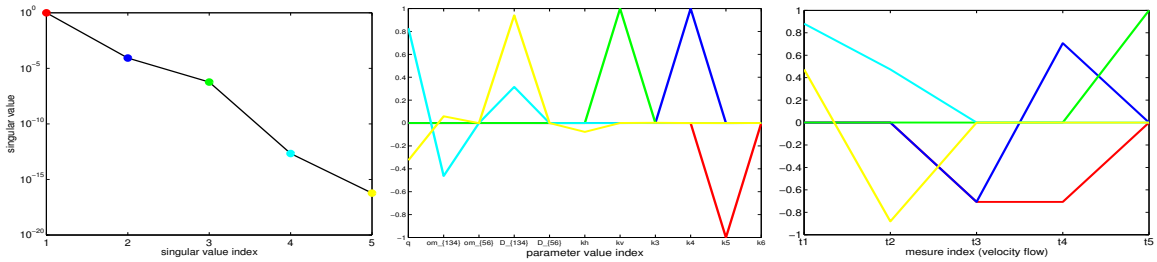


FIG. 7.21 – S_4 - maillage fin

Deuxième partie

Sensibilité par rapport à la discrétisation

Chapitre 8

De la dérivation par rapport aux paramètres de discrétisation

L'idée est de construire un indicateur de sensibilité par rapport aux paramètres de discrétisation, en particulier par rapport au pas de temps et/ou à des paramètres du maillage, en se basant toujours sur des approximations du premier ordre. Ils pourront être intégrés dans la matrice de sensibilité F', afin par exemple de déterminer s'il est plus utile, pour réduire l'incertitude sur les indicateurs de sûreté, de réduire l'incertitude sur les paramètres d'entrée physiques du problème, ou d'effectuer de nouveaux calculs avec une discrétisation plus fine.

Nous allons d'abord nous intéresser au cas très simple d'une équation différentielle ordinaire pour laquelle le seul paramètre de discrétisation est un pas de temps, puis étendre la construction de la dérivée par rapport au pas de temps aux cas des équations aux dérivées partielles d'évolution, enfin nous nous intéresserons aux paramètres de discrétisation en espace.

8.1 Sensibilité à la discrétisation en temps

8.1.1 Équation différentielle ordinaire

8.1.1.1 Équation différentielle linéaire du premier ordre à coefficient constant

On considère l'équation différentielle ordinaire (EDO)

$$\begin{cases} \dfrac{\mathrm{d}x}{\mathrm{d}t} = a_0 x, \\ x(0) = x_0, \end{cases} \tag{8.1}$$

de solution exacte $x(t) = e^{a_0 t} x_0$. Une solution approchée de (8.1) peut être construite suivant un schéma de la forme

$$\begin{cases} x_0 \text{ est donné (condition initiale)} \\ \text{pour } n = 0, 1, ..., \ \dfrac{x_{n+1} - x_n}{k} = a_0(r x_n + s x_{n+1}) \\ \text{où } k \in I = \left]0, \dfrac{1}{|a_0|}\right[, \ r \geq 0, \ s \geq 0, \ r + s = 1. \end{cases} \tag{8.2}$$

On reconnaît un schéma d'Euler explicite si $r = 1$, un schéma d'Euler implicite si $r = 0$, le schéma de Crank-Nicolson si $r = \dfrac{1}{2}$.

Notons tout d'abord que la sensibilité de l'approximation de $x(t)$ par rapport à k n'a rien à voir avec $\dfrac{\partial x_n}{\partial k}$. En effet, x_n est une approximation de $x(nk)$ et considérer x_n calculé avec deux pas de temps différents, disons k et $k + \delta k$, revient à évaluer $x(nk)$ et $x(nk + n\delta k)$, avec des schémas de discrétisation légèrement différents. La dérivée $\dfrac{\partial x_n}{\partial k}$ approcherait donc plutôt, au facteur n près, $\dfrac{\mathrm{d}x(t)}{\mathrm{d}t} = x'(t)$, où $t = nk$.

On définit alors pour $n \in \mathbb{N}$

$$\tilde{x}(k, nk) = x_n = (u(k))^n x_0, \tag{8.3}$$

où u est définie sur $\{0\} \cup I$ par $u(k) = \dfrac{1 + r a_0 k}{1 - s a_0 k}$. Comme $u(k)$ reste strictement positif sur I, la fonction $\tilde{x}$ peut se prolonger de manière C^∞ et monotone sur $I \times \mathbb{R}^+$ sous la forme

$$\tilde{x}(k, t) = x_0 u(k)^{\frac{t}{k}} = x_0 \exp\left[\frac{t}{k} \ln(u(k))\right]. \tag{8.4}$$

$\tilde{x}(k, t)$ est une approximation de $x(t)$ dépendant d'un paramètre k et égale à x_n lorsque $t = nk$. Elle est définie sur un convexe et très régulière. On peut étudier la sensibilité du premier ordre de cette approximation par rapport au paramètre k.

Dans le cas général, on cherchera à construire un prolongement sur $I \times [0, T]$, le plus régulier possible, d'une fonction définie sur $\{(k, nk), \ k \in I, \ n \in \mathbb{N}, \ nk \in [0, T]\}$, et c'est la sensibilité de ce prolongement par rapport au pas de temps k et à d'éventuels paramètres (par exemple a_0 pour l'équation (8.1)) que l'on étudiera. On pourrait aussi étudier la sensibilité par rapport aux autres paramètres de discrétisation (r, s pour l'équation (8.1), pour mesurer si cela vaut le coup de prendre le risque de se rapprocher de l'instabilité).

Résultats. On obtient si u est de classe C^1 avec $u(0) = 1$

$$\lim_{k \to 0} \tilde{x}(k, t) = e^{u'(0)t} x_0. \tag{8.5}$$

On a pour les discrétisations considérées $u'(0) = a_0$ donc $\lim_{k \to 0} \tilde{x}(k, t) = x(t)$. C'est un résultat attendu car ces schémas classiques sont convergents.

$$\frac{\partial \tilde{x}(k,t)}{\partial k} = \frac{t}{k} u(k)^{\frac{t}{k}} \left(\frac{u'(k)}{u(k)} - \frac{\ln(u(k))}{k} \right). \tag{8.6}$$

Si u est de classe C^2 avec $u(0) = 1$ alors on a également

$$\lim_{k \to 0} \frac{\partial \tilde{x}(k,t)}{\partial k} = e^{tu'(0)} \left(-\frac{t}{2} \left(u'(0)\right)^2 + \frac{t}{2} u''(0) \right) x_0. \tag{8.7}$$

En particulier, pour le schéma d'**Euler explicite**, on a $u''(0) = 0$ et par conséquent

$$\lim_{k \to 0} \frac{\partial \tilde{x}(k,t)}{\partial k} = -\frac{1}{2} a_0^2 x_0 t e^{a_0 t} \neq 0 \text{ si } t \neq 0.$$

Pour le schéma d'**Euler implicite**, $u''(0) \neq 2a_0^2$ et par conséquent

$$\lim_{k \to 0} \frac{\partial \tilde{x}(k,t)}{\partial k} = \frac{1}{2} a_0^2 x_0 t e^{a_0 t} \neq 0 \text{ si } t \neq 0.$$

En revanche, pour le schéma de **Crank-Nicolson**, $u''(0) = a_0^2$ et par conséquent

$$\lim_{k \to 0} \frac{\partial \tilde{x}(k,t)}{\partial k} = 0 \ \forall t \in \mathbb{R}^+.$$

On peut rapprocher ces résultats du fait que la convergence du schéma de Crank-Nicolson est d'ordre 2, alors que les convergences des schémas d'Euler explicite et implicite sont d'ordre 1.

8.1.1.2 Système différentiel linéaire à coefficients constants

Un peu plus généralement, on considère un système différentiel linéaire à coefficients constants

$$\begin{cases} \dfrac{\mathrm{d}X}{\mathrm{d}t} = AX + B, \\ X(0) = X_0, \end{cases} \tag{8.8}$$

de solution exacte $X(t) = \exp(tA)R_1 + R_2$ où $R_1 = X_0 + A^{-1}B$ et $R_2 = -A^{-1}B$. On discrétise l'équation (8.8) suivant un schéma d'Euler de paramètres $k \in I,\ r,\ s$. On note $\mathbf{I}$ la matrice identité de la même taille que A. On suppose que les paramètres choisis sont tels que $\mathbf{I} - skA$ soit inversible. Soit U définie sur $\{0\} \cup I$ par

$$U(k) = (\mathbf{I} - skA)^{-1}(\mathbf{I} + rkA). \tag{8.9}$$

Similairement à ce qui est fait dans 8.1.1.1, on définit pour $k \in I$ et $n \in \mathbb{N}$

$$\tilde{X}(k, nk) = U(k)^n R_1 + R_2.$$

Si $U(k)$ est diagonalisable à valeurs propres strictement positives (en particulier si elle est symétrique définie positive), alors on peut prolonger $\tilde{X}$ sur $I \times \mathbb{R}^+$ par

$$\tilde{X}(k,t) = \exp\left(\frac{t}{k}\ln(U(k))\right) R_1 + R_2. \tag{8.10}$$

On a pour U suffisamment régulière

$$\frac{\partial \tilde{X}(k,t)}{\partial k} = \exp\left[\frac{t}{k}\ln\left(U(k)\right)\right]\left[\frac{t}{k}\left(U(k)\right)^{-1}U'(k) - \frac{t}{k^2}\ln\left(U(k)\right)\right] R_1. \tag{8.11}$$

Résultats. On obtient alors, comme $U(0) = \mathbf{I}$ et U est C^2 (par exemple pour la norme quadratique)

$$\lim_{k\to 0} \tilde{X}(k,t) = \exp(tU'(0))R_1 + R_2 = \exp(tA)R_1 + R_2 \tag{8.12}$$

et

$$\lim_{k\to 0} \frac{\partial \tilde{X}(k,t)}{\partial k} = \exp(tU'(0))\left(-\frac{t}{2}\left(U'(0)\right)^2 + \frac{t}{2}U''(0)\right) R_1 = \exp(tA)\left(-\frac{t}{2}A^2 + \frac{t}{2}U''(0)\right) R_1.$$

Le terme $U''(0)$ dépend de la discrétisation choisie.

Note. On rappelle que si la matrice carrée M se diagonalise sous la forme

$$M = P^{-1}DP$$

où D est diagonale alors on utilise pour les calculs

$$\exp(M) = P^{-1}\exp(D)P$$

($\exp(D)$ est diagonale avec $\exp(D)_{i,i} = \exp(D_{i,i})$), et si de plus D est à termes diagonaux strictement positifs,

$$\ln(M) = P^{-1}\ln(D)P$$

($\ln(D)$ est diagonale avec $\ln(D)_{i,i} = \ln(D_{i,i})$).

8.1.2 Équation de diffusion

L'équation d'évolution

$$\begin{cases} \dfrac{\partial p}{\partial t} + \operatorname{div}\left(-K\vec{\nabla}p\right) = 0 \\ \text{conditions initiales} \\ \text{conditions aux limites,} \end{cases} \tag{8.13}$$

où K est supposé constant en temps, se discrétise en temps et en espace, si on choisit un schéma implicite, sous la forme

$$\frac{P^{n+1} - P^n}{k} + AP^{n+1} = R \tag{8.14}$$

où A est une matrice symétrique définie positive dépendant de K et de la discrétisation en espace mais pas du pas de temps k, les tailles de R et P^n, P^{n+1} dépendent de la discrétisation en espace, pour une discrétisation de l'espace donnée R dépend des conditions aux limites. La matrice carrée $\mathbf{I} + kA$ est symétrique définie positive, donc on peut poursuivre : P^n s'écrit sous la forme

$$P^n = (\mathbf{I} + kA)^{-n} \left(P^0 - A^{-1}R\right) + A^{-1}R. \tag{8.15}$$

Soit $I = \mathbb{R}^{+*}$. On définit $\tilde{P}$ sur $I \times \mathbb{R}^+$ par

$$\tilde{P}(k,t) = \exp\left[-\frac{t}{k}\ln(\mathbf{I} + kA)\right]\left(P_0 - A^{-1}R\right) + A^{-1}R. \tag{8.16}$$

On est en fait dans la situation de 8.1.1.2 avec $U(k) = (\mathbf{I} + kA)^{-1}$, $R_1 = P_0 - A^{-1}R$, $R_2 = A^{-1}R$ car l'équation à résoudre est $\frac{\mathrm{d}P}{\mathrm{d}t} = -AP + R$, par un schéma implicite, i.e. avec $r = 0$.

8.1.3 Équation de diffusion/convection

On considère la discrétisation de l'équation d'évolution

$$\begin{cases} \dfrac{\partial c}{\partial t} + \mathrm{div}(c\vec{v} - D\vec{\nabla}c) = 0 \\ \text{conditions initiale} \\ \text{conditions aux limites} \end{cases} \tag{8.17}$$

suivant une méthode de splitting, explicite pour la convection, implicite pour la diffusion, avec des pas de temps éventuellement différents pour la convection et la diffusion. On suppose que $\vec{v}$ et D sont constants en temps. Chaque étape de convection ($m = 0, ..., M-1$) s'écrit sous la forme

$$\frac{C^{n,m+1} - C^{n,m}}{k/M} = HC^n + R_{conv} \tag{8.18}$$

avec $C^{n,0} = C^n$, où H est une matrice carrée dépendant de $\vec{v}$ pour une discrétisation de l'espace donnée, indépendante du pas de temps k, et R_{conv} est un vecteur dépendant des conditions initiales pour une discrétisation en espace donnée, indépendant du pas de temps k. L'étape de diffusion s'écrit sous la forme

$$\frac{C^{n+1} - C^{n,M}}{k} + AC^{n+1} = R_{diff} \tag{8.19}$$

où A est une matrice symétrique définie positive dépendant de la discrétisation en espace et de K, et le vecteur R_{diff} dépend de la discrétisation en espace et des conditions aux limites. En sommant ces équations ((8.19)$+\frac{1}{M}\sum$(8.18)) on obtient

$$\frac{C^{n+1} - C^n}{k} + AC^{n+1} = R_{diff} + R_{conv} + \frac{1}{M}H\sum_{m=0}^{M-1} C^{n,m}. \tag{8.20}$$

On se place pour l'instant dans le cas $M = 1$. Dans ce cas particulier on peut écrire

$$C^n = \left((\mathbf{I} + kA)^{-1} (\mathbf{I} + kH)\right)^n R_1 + R_2 \tag{8.21}$$

avec

$$\begin{cases} (A - H)R_1 = (A - H)C^0 - (R_{conv} + R_{diff}) \,, \\ (A - H)R_2 = (R_{conv} + R_{diff}) \,. \end{cases} \tag{8.22}$$

Si k vérifie les conditions CFL (voir l'équation (2.76)), alors $\mathbf{I} + kH$ est définie positive donc diagonalisable. Soit I un intervalle de $\mathbb{R}^{+*}$ tel que tout k dans I satisfasse la condition CFL. On peut définir $\tilde{C}$ sur $I \times \mathbb{R}^+$ par

$$\tilde{C}(k, t) = \exp\left[\frac{t}{k} \ln\left((\mathbf{I} + kA)^{-1}(\mathbf{I} + kH)\right)\right] R_1 + R_2. \tag{8.23}$$

On est en fait dans une situation similaire à celle de 8.1.1.2 avec

$$U\ (k) = (\mathbf{I} + kA)^{-1}(\mathbf{I} + kH).$$

Cependant, l'équation

$$\frac{\mathrm{d}C}{\mathrm{d}t} = (-A + H)C + R_{conv} + R_{diff}$$

n'est pas discrétisée avec un schéma d'Euler.

8.2 Sensibilité à la discrétisation en espace

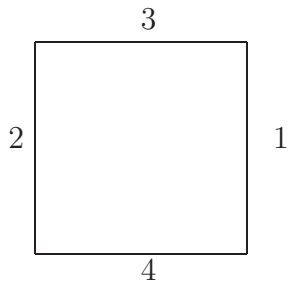
On présente ici une tentative pour quantifier une sensibilité au premier ordre par rapport à des paramètres de discrétisation en espace, pour des problèmes faisant intervenir des équations aux dérivées partielles.

8.2.1 Laplacien en dimension 2 avec un maillage à cellules carrées

On considère l'équation aux dérivées partielles d'écoulement

$$\begin{cases} \vec{u} = -K\vec{\nabla}p \\ \text{div } \vec{u} = 0 \\ \text{conditions aux limites de Dirichlet.} \end{cases} \tag{8.24}$$

On suppose K uniforme. Le domaine de calcul est de dimension 2, carré de côté 1. On utilise un maillage carré dont les cellules sont de côté h. Les arêtes des cellules sont numérotées localement de la façon suivante :



Le maillage a $N_c = \dfrac{1}{h^2}$ cellules et $N_e = \dfrac{2}{h}\left(\dfrac{1}{h} - 1\right)$ arêtes intérieures. On rappelle la forme des équation locales sur chaque cellule T_i :

$$\mathcal{E}_i \left\{ \begin{array}{l} AU_i - \left(B^T\right) P_i + \left(C_i^T\right) L = -L_{D_i} \\ BU_i = 0. \end{array} \right. \tag{8.25}$$

où

- Le vecteur U_i de taille 4 contient les flux de $\vec{u}$ à travers chaque arête de T_i orientée vers l'extérieur. Comme K est uniforme et le maillage est régulier, la matrice A ne dépend pas de la cellule. Comme on est en dimension 2, elle ne dépend pas non plus du pas d'espace h (en dimension 3, elle serait proportionnelle à h^{-1}) et elle est de taille 4×4.
- Le scalaire P_i représente la pression au centre de la cellule i. La matrice B est une matrice ligne, $B = (1\ 1\ 1\ 1)$.
- Le vecteur L de taille N_e contient les traces de pression. La matrice C_i est de taille $4 \times N_e$ et contient au plus 1 terme non nul par ligne, égal à 1.
- Le vecteur L_{D_i} de taille 4 contient les conditions aux limites de Dirichlet (les composantes éventuellement non nulles correspondent aux lignes nulles de C_i).

La continuité du flux de $\vec{u}$ est exprimée par l'équation supplémentaire

$$\sum_{i=1}^{N_e} C_i U_i = 0. \tag{8.26}$$

En éliminant les inconnues U_i puis P_i à l'aide des équations (8.25), l'équation (8.26) se réécrit sous la forme

$$\sum_{i=1}^{N_c} C_i M({}^t C_i L + L_{D_i}) = 0. \tag{8.27}$$

On ne rappelle pas ici l'expression de M, on indique juste qu'elle est symétrique définie positive de taille 4×4 et qu'elle est elle aussi indépendante du pas d'espace.

8.2.2 Tentative de construction d'une fonction continue de h et de l'espace

On va s'intéresser au champ de pression approché $\tilde{p}(h, ., .)$, que l'on va supposer être construit à partir des traces de pression (composantes de L) et des conditions aux limites

uniquement. Pour passer à une version continue en espace de cette équation, l'idée est de jongler entre les indices de cellules (resp. de traces) et les coordonnées de leurs centres (resp. des milieux des arêtes correspondantes). On définit d'abord $\tilde{p}(h,x,y)$ pour les h inverses d'entiers pour $(x,y) \in \left(\frac{1}{h}\mathbb{Z} \times \frac{1}{h}\mathbb{Z}\right) \cap ([0,1] \times [0,1])$. En particulier on peut écrire, si les coordonnées du centre de la cellule T_i sont (x_i, y_i),

$$\begin{pmatrix} p(h, x_i + \frac{h}{2}, y_i) \\ p(h, x_i - \frac{h}{2}, y_i) \\ p(h, x_i, y_i + \frac{h}{2}) \\ p(h, x_i, y_i - \frac{h}{2}) \end{pmatrix} =^t C_i L + L_{D_i}. \tag{8.28}$$

La matrice $M = \begin{pmatrix} M_1 \\ M_2 \\ M_3 \\ M_4 \end{pmatrix}$ correspond à une fonction de 4 variables p_1, p_2, p_3, p_4. Pour tout point (x,y) milieu d'une arête on vérifie l'équation

$$\sum_{i=1}^{N_c} f_i(x,y) = 0 \tag{8.29}$$

où f_i est définie en chaque milieu d'arête de la manière suivante : on note, pour simplifier les écritures,

$$\begin{aligned} p_{i_1} &= p(h, x_i + \tfrac{h}{2}, y_i), \\ p_{i_2} &= p(h, x_i - \tfrac{h}{2}, y_i), \\ p_{i_3} &= p(h, x_i, y_i + \tfrac{h}{2}), \\ p_{i_4} &= p(h, x_i, y_i - \tfrac{h}{2}). \end{aligned} \tag{8.30}$$

Alors, si (x_i, y_i) sont les coordonnées du centre de la cellule T_i,

$$\begin{aligned} &f_i(x_i + \tfrac{h}{2}, y_i) = M_1(p_{i_1}, p_{i_2}, p_{i_3}, p_{i_4}), \\ &f_i(x_i - \tfrac{h}{2}, y_i) = M_2(p_{i_1}, p_{i_2}, p_{i_3}, p_{i_4}), \\ &f_i(x_i, y_i + \tfrac{h}{2}) = M_3(p_{i_1}, p_{i_2}, p_{i_3}, p_{i_4}), \\ &f_i(x_i, y_i - \tfrac{h}{2}) = M_4(p_{i_1}, p_{i_2}, p_{i_3}, p_{i_4}), \\ &f_i(x,y) = 0 \text{ pour } (x,y) \text{ autre milieu d'arête.} \end{aligned} \tag{8.31}$$

Si le point (x,y) est le milieu d'une arête verticale alors l'équation (8.29) s'explicite :

$$\begin{aligned} M_1\left(p(h,x,y), p(h,x-h,y), p(h,x-\tfrac{h}{2},y+\tfrac{h}{2}), p(h,x-\tfrac{h}{2}y-\tfrac{h}{2})\right) + \\ M_2\left(p(h,x+h,y), p(h,x,y), p(h,x+\tfrac{h}{2},y+\tfrac{h}{2}), p(h,x+\tfrac{h}{2},y-\tfrac{h}{2})\right) = 0. \end{aligned} \tag{8.32}$$

Si le point (x,y) est le milieu d'une arête horizontale alors l'équation (8.29) s'explicite :

$$\begin{aligned} M_3\left(p(h,x+\tfrac{h}{2},y-\tfrac{h}{2}), p(h,x-\tfrac{h}{2},y-\tfrac{h}{2}), p(h,x,y), p(h,x,y-h)\right) + \\ M_4\left(p(h,x+\tfrac{h}{2},y+\tfrac{h}{2}), p(h,x-\tfrac{h}{2},y+\tfrac{h}{2}), p(h,x,y+h), p(h,x,y)\right) = 0. \end{aligned} \tag{8.33}$$

On espère que, pour tout $h > 0$ et assez petit, on peut définir $p(h, ., .)$ en tout point de $[0, 1] \times [0, 1]$ par les relations (8.32) et (8.33). On a besoin de conditions aux limites et de préciser sur quel domaine les équation (8.32) et (8.33) sont vérifiées :

$$\begin{cases} (8.32) \text{ pour } \{(x-h, y), (x+h, y), (x, y-\frac{h}{2}), (x, y+\frac{h}{2})\} \subset [0, 1] \times [0, 1], \\ (8.33) \text{ pour } \{(x, y-h), (x, y+h), (x-\frac{h}{2}, y), (x+\frac{h}{2}, y)\} \subset [0, 1] \times [0, 1], \\ p(h, x, y) \text{ donné pour } x = 0 \text{ ou } 1 \text{ ou } y = 0 \text{ ou } 1. \end{cases} \tag{8.34}$$

Si plusieurs fonctions vérifient ces équations, des critères de régularité et monotonicité pourraient permettre d'en retenir une. On donne des exemples pour des cas monodimensionels en section 8.2.4.

8.2.3 Suggestion d'approximation de dérivée

On dérive (8.32) puis (8.33) par rapport à h :

On note

$$\gamma_H = \frac{\partial p}{\partial H},\ \gamma_X = \frac{\partial p}{\partial X},\ \gamma_Y = \frac{\partial p}{\partial Y}. \tag{8.35}$$

On espère que l'on peut obtenir une approximation locale de γ_H en supposant que γ_H est localement constant par rapport aux variables d'espace X, Y et que γ_X et γ_Y sont localement linéaires par rapport aux variables d'espace :

$$\begin{aligned} \gamma_X(x+\delta_x, y+\delta_y) &= \gamma_X(x, y) + \Delta_X \delta_x + \Delta_{XY} \delta_y, \\ \gamma_Y(x+\delta_x, y+\delta_y) &= \gamma_Y(x, y) + \Delta_{XY} \delta_x + \Delta_Y \delta_y. \end{aligned} \tag{8.36}$$

Si c'est le cas, on obtient, en notant S la somme de tous les coefficients de M (on rappelle aussi que M est symétrique),

$$\begin{aligned} S\gamma_H + \left(M_{11} + M_{12} + \frac{M_{13} + M_{14} + M_{23} + M_{24}}{2}\right) h\Delta_X + \\ \left(M_{34} + M_{44} + \frac{M_{13} + M_{14} + M_{23} + M_{24}}{2}\right) h\Delta_Y + \\ (M_{14} - M_{13} + M_{23} - M_{24})\, h\Delta_{XY} \approx 0. \end{aligned} \tag{8.37}$$

En remplaçant M par sa valeur, le terme avec les dérivées croisées disparaît et on obtient

$$\gamma_H = \frac{h}{4}\left(\Delta_X + \Delta_Y\right). \tag{8.38}$$

8.2.4 Exemples

8.2.4.1 Équation du premier ordre en dimension 1

On considère le problème suivant, construit par exemple pour résoudre l'équation $\mathrm{div} u = 0$:

$$\begin{aligned} &\text{trouver } \tilde{u} : \mathbb{R}^{+*} \times \mathbb{R}^{+} \to \mathbb{R} \text{ telle que} \\ &\begin{cases} \tilde{u}(h, x+h) - \tilde{u}(h, x) = 0\ \forall h > 0, x \geq 0 \\ \tilde{u}(h, 0) = a. \end{cases} \end{aligned} \tag{8.39}$$

Toutes le fonctions $\tilde{u}$ telles que $\tilde{u}(h,.)$ soit périodique de période h avec $\tilde{u}(h,0) = a$ conviennent ; la fonction constante a convient et est monotone.

8.2.4.2 Équation du deuxième ordre en dimension 1

On considère le problème suivant, construit par exemple pour résoudre l'équation $\Delta p = 0$:

$$\begin{array}{l} \text{trouver } \tilde{p} :]0, \frac{1}{2}] \times [0,1] \to \mathbb{R} \text{ telle que} \\ \left\{ \begin{array}{l} \tilde{p}(h, x+h) - \tilde{p}(h,x) = \tilde{p}(h,x) - \tilde{p}(h, x-h) \ \forall x \in [h, 1-h], \\ \tilde{p}(h,0) = a, \\ \tilde{p}(h,1) = b. \end{array} \right. \end{array} \tag{8.40}$$

Les solutions vérifient, pour tout $n \in \mathbb{N}$, $\frac{1}{2} \geq h > 0$ tels que $(n+1)h \leq 1$ (par récurrence immédiate),

$$\left\{ \begin{array}{l} \tilde{p}(h, (n+1)h) = (n+1)\tilde{p}(h,h) - n\tilde{p}(h,0), \\ \tilde{p}(h, 1-(n+1)h) = (n+1)\tilde{p}(h, 1-h) - n\tilde{p}(h,1). \end{array} \right. \tag{8.41}$$

Si $\dfrac{1}{h}$ est entier alors les valeurs de $\tilde{p}(h,h)$ et $\tilde{p}(h, 1-h)$ sont imposées par l'égalité des deux lignes de (8.41) et on obtient

$$\tilde{p}(h, nh) = nhb + (1-nh)a. \tag{8.42}$$

On peut sans problème étendre cette définition pour $1/h$ rationnel puis quelconque, par passage à la limite, et pour x quelconque dans [0,1] :

$$\tilde{p}(h,x) = xb + (1-x)a. \tag{8.43}$$

On se trouve dans un cas pas très intéressant où la solution approchée ne dépend pas du pas de discrétisation. On peut tout de même noter qu'ici, on a $\frac{\partial \tilde{p}}{\partial h} = 0$ et $\frac{\partial^2 \tilde{p}}{\partial x^2} = 0$, donc l'égalité (8.37) est bien vérifiée.

Troisième partie

Écoulements diphasiques eau/air en milieu poreux

Chapitre 9

De la prise en compte des variations de la pression en air dans la modélisation des écoulements souterrains eau/air

L'équation de Richards est largement utilisée en hydrogéologie. C'est une simplification du modèle général diphasique, car elle suppose que la pression en air ne varie pas. Elle ne peut donc pas tenir compte des effets de la présence d'air qui peut être piégé dans le sous-sol. Afin de mettre en évidence les effets de cette simplification, nous allons présenter ces deux modèles et les comparer numériquement. [1]

9.1 Les écoulements diphasiques

9.1.1 Caractéristiques de l'écoulement

Une **phase** est une partie chimiquement homogène d'un système, séparée des autres parties ou phases par une frontière définie physiquement. Dans un système diphasique, l'espace vide est rempli par au moins deux fluides immiscibles, c'est-à-dire qui maintiennent une frontière distincte entre eux. Plusieurs fluides miscibles entre eux peuvent former une seule phase (par exemple, les gaz sont toujours parfaitement miscibles entre eux). Dans notre système, on suppose que l'on reste dans des conditions de température et de pression telles que l'eau et l'air soient parfaitement immiscibles, et chacun de ces deux fluides constitue une phase.

Les écoulements diphasiques en milieux poreux sont caractérisés par les paramètres de la section 1.1.4, et par les paramètres suivants, définis en fonction du même volume représentatif élémentaire que la porosité.

[1] Ce travail a été mené dans le cadre de l'Action de Recherche Coopérative DYNAS (http ://www-rocq.inria.fr/estime/DYNAS/).

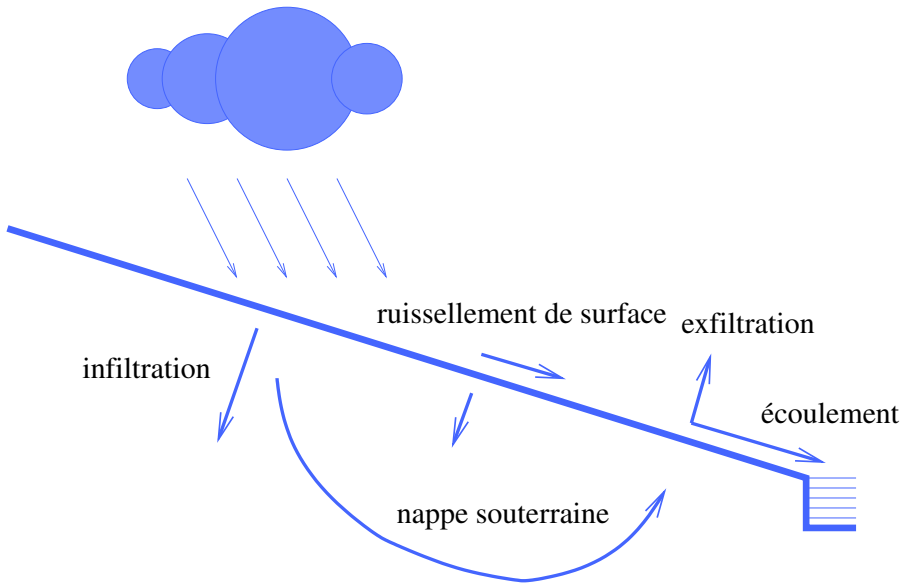


FIG. 9.1 – Phénomènes hydrogéologiques.

La **saturation** s_i est la proportion du volume de pore occupée par le fluide i dans un volume représentatif élémentaire.

La **perméabilité** pour le fluide i est le produit de la perméabilité intrinsèque et de la perméabilité relative pour ce fluide. La **perméabilité intrinsèque**, notée K, est indépendante des fluides considérés. La **perméabilité relative** k_{r_i} pour le fluide i est un coefficient ne dépendant que de sa saturation. Cette perméabilité relative est une fonction non linéaire de la saturation. Les fonctions $k_{r_i}(s_i)$ ainsi définies sont toujours croissantes et prennent des valeurs comprises entre 0 et 1. Elles sont déterminées expérimentalement et répertoriées dans des tables. On peut également trouver des modèles formels de ces fonctions comme les modèles de van Genuchten, présenté en 1980 dans [77], ou de Brooks-Corey. Elles sont reprises par exemple dans [4]. Les liens avec d'autres modèles sont présentés dans [61].

Le fluide i est également caractérisé par sa **viscosité** μ_i, sa **masse volumique** ρ_i et sa **pression** p_i, qui varient a priori dans l'espace.

L'écoulement du fluide i est représenté par le vecteur **flux volumique** $\vec{\varphi}_i$.

Dans un écoulement diphasique on appelle **fluide mouillant** celui qui a la plus faible pression. On peut alors définir la **pression capillaire** par

$$p_c = p_{\text{fluide non mouillant}} - p_{\text{fluide mouillant}}.$$

Dans les cas que nous considérerons, c'est une fonction de la saturation du fluide mouillant, que nous noterons simplement s, décroissante et toujours positive par définition. Une im-

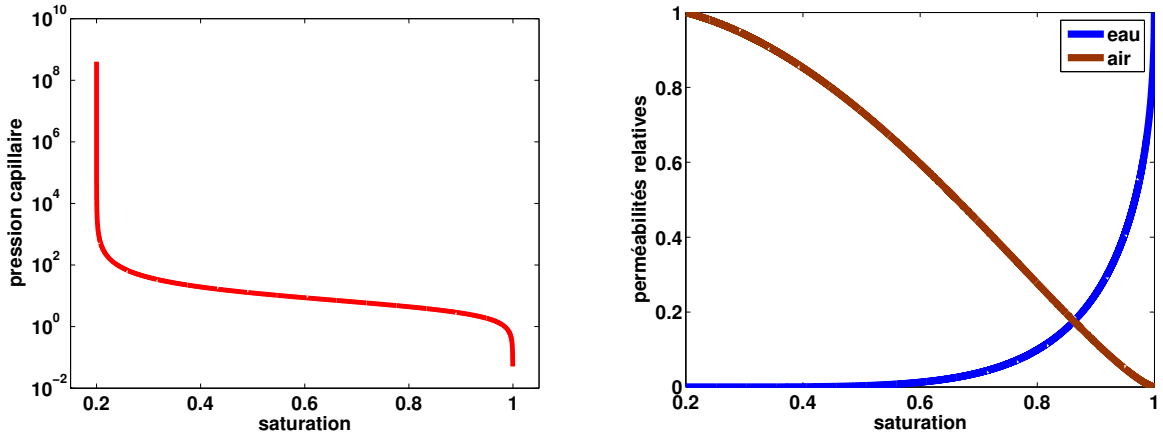


FIG. 9.2 – Modèle de van Genuchten avec $n = 2$ et $s_r = 0.2$.

mobilisation est possible, c'est-à-dire que l'on peut avoir

$$\begin{array}{ll} \lim\limits_{\substack{s \to s_{\min} \\ s > s_{\min}}} p_c(s) = +\infty & \text{avec } s_{\min} > 0 \\ p_c(s_{\max}) = 0 & \text{avec } s_{\max} < 1. \end{array}$$

On pourra également en trouver une description plus précise dans [4].

Certains paramètres sont constants ou dépendent uniquement de la position. Ce sont soit des propriétés du sous-sol (la porosité ϕ et la perméabilité intrinsèque K), soit des propriétés des fluides (leur viscosité dynamique μ, leur masse volumique ρ pour les fluides incompressibles ou leur coefficient de compressibilité c_a pour les fluides compressibles). La compressibilité de l'air introduira des termes quadratiques dans certaines des équations que nous nommerons dans la suite « équations en pression ».

D'autres paramètres sont également fonction de la saturation. Ceci est la source de non linéarités dans les équations que nous nommerons dans la suite « équations en saturation ». Il s'agit de la pression capillaire, qui représente la différence $p_c(s)$ entre les pressions des deux fluides de l'écoulement – dans notre cas la pression en air est toujours supérieure à la pression en eau et on définit $p_c(s) = p_a - p_w > 0$ – et des perméabilités relatives k_{ri} de chaque fluide, fonctions croissantes de leurs saturations respectives et bornées par 0 et 1.

Pour un écoulement eau / air le fluide non mouillant est l'air (indice a) et le fluide mouillant est l'eau (indice w).

9.1.2 Lois de l'écoulement

On suppose que :

– les phases occupent tous les volumes de pore, c'est-à-dire que

$$s_a + s_w = 1. \tag{9.1}$$

On peut donc définir les saturations également de la manière suivante :

$$s_i = \frac{\text{volume de pore du fluide } i}{\text{volume de pore de l'eau } + \text{ volume de pore de l'air}}$$

dans un volume représentatif élémentaire ;

- l'eau est incompressible — i.e. sa masse volumique est constante — et l'air est faiblement compressible, c'est-à-dire que la loi reliant sa masse volumique à sa pression est linéaire (l'air est donc considéré comme un gaz parfait) :

$$\rho_w = cste, \quad \rho_a = c_a\, p_a \tag{9.2}$$

  où c_a est une constante positive ;
- les viscosités μ_a et μ_w sont des constantes ;
- la pression capillaire ne dépend que de la saturation :

$$p_a - p_w = p_c(s_w). \tag{9.3}$$

Les écoulements diphasiques sont régis par les lois de conservation de chaque fluide :

$$\phi \frac{\partial(\rho_i s_i)}{\partial t} + \text{div } (\rho_i \vec{\varphi}_i) = 0 \quad \text{pour} \quad i \in \{a, w\} \tag{9.4}$$

où $\vec{\varphi}_i$ représente le flux volumique du fluide i. Il est explicité à partir de la loi de Darcy ([21]) :

$$\vec{\varphi}_i = -K \frac{k_{r_i}}{\mu_i} \left(\vec{\nabla} p_i - \rho_i\, \vec{g} \right) \quad \text{pour} \quad i \in \{a, w\}. \tag{9.5}$$

On pourra trouver des explications sur l'origine de cette loi dans [5], chapitre 2.

On remarque que les lois de conservation sont des lois massiques alors que la loi de Darcy est une loi volumique, ce qui introduit une difficulté supplémentaire lorsque l'on tient compte de la compressibilité d'un fluide par rapport au cas incompressible : on ne peut alors pas obtenir d'équation stationnaire par combinaison linéaire des deux lois de conservation.

On note $s = s_w$. On a $s_a = 1 - s$, ainsi k_{r_a} et k_{r_w} peuvent s'exprimer comme des fonctions de s. On définit les flux massiques :

$$\vec{u}_i = \rho_i \vec{\varphi}_i \quad \text{pour} \quad i \in \{a, w\}, \tag{9.6}$$

et le flux massique total

$$\vec{u} = \vec{u}_a + \vec{u}_w. \tag{9.7}$$

On va réduire ces équations en un système de quatre équations en inconnues $s,\ \vec{u}_w,\ p_a, \vec{u}$.

On introduit, pour simplifier les écritures, les mobilités, fonctions non linéaires de la saturation,

$$k_i(s) = \frac{k_{r_i}(s)}{\mu_i} \quad \text{pour} \quad i \in \{a, w\}, \tag{9.8}$$

Le modèle présentée dans la section 2.1 est le cas limite du modèle diphasique pour $s \equiv 1$ ($\mathcal{K} = \dfrac{K}{\mu_w}$).

9.1.2.1 Équations en saturation

On reprend l'équation de conservation de l'eau,

$$\phi \frac{\partial (\rho_w s)}{\partial t} + \text{div } \vec{u}_w = 0. \tag{9.9}$$

La loi de Darcy (9.5) nous permet d'obtenir pour le flux massique (9.6), en utilisant la définition de la pression capillaire (9.3),

$$\vec{u}_w = -\rho_w K k_w(s) \left(\vec{\nabla} p_a - p_c'(s) \vec{\nabla} s - \rho_w \vec{g} \right). \tag{9.10}$$

En effet, comme p_c ne dépend que de s, on a $\vec{\nabla} p_w - \vec{\nabla} p_a = -p_c'(s) \vec{\nabla} s$. Pour éliminer le gradient de pression en air, on considère également la loi de Darcy pour l'air :

$$\vec{u}_a = -K \rho_a(p_a) k_a(s) \left(\vec{\nabla} p_a - \rho_a(p_a) \vec{g} \right). \tag{9.11}$$

Pour éliminer le gradient de pression en air, on combine donc
$\rho_a(p_a) k_a(s) \times (9.10) - \rho_w k_w(s) \times (9.11)$: on obtient

$$\begin{aligned} &\rho_a(p_a) k_a(s) \vec{u}_w - \rho_w k_w(s) \left(\vec{u} - \vec{u}_w \right) = \\ &-K \rho_w k_w(s) \rho_a(p_a) k_a(s) \left(-p_c'(s) \vec{\nabla} s - \rho_w \vec{g} \right) + K \rho_w k_w(s) \rho_a(p_a) k_a(s) \left(-\rho_a(p_a) \vec{g} \right). \end{aligned}$$

On définit la mobilité massique par

$$d(s, p_a) = \rho_w k_w(s) + \rho_a(p_a) k_a(s), \tag{9.12}$$

et la mobilité massique fractionnaire de l'eau par

$$b_0(s, p_a) = \frac{\rho_w k_w(s)}{d(s, p_a)}. \tag{9.13}$$

Alors

$$\begin{aligned} d(s, p_a) \vec{u}_w - \rho_w k_w(s) \vec{u} &= -K d^2(s, p_a) b_0(s, p_a)(1 - b_0(s, p_a)) \left(-p_c'(s) \vec{\nabla} s \right) \\ &\quad + K \left(\rho_w^2 \rho_a(p_a) k_w(s) k_a(s) - \rho_w \rho_a^2(p_a) k_w(s) k_a(s) \right) \vec{g}. \end{aligned}$$

On introduit enfin, pour simplifier les notations, les quantités (non nommées)

$$\begin{aligned} b_\rho(s, p_a) &= (1 - b_0(s, p_a)) b_0(s, p_a) d(s, p_a) (\rho_w - \rho_a(p_a)) \\ &= \frac{\rho_w k_w(s) \rho_a(p_a) k_a(s)}{s(s, p_a)} (\rho_w - \rho_a(p_a)), \end{aligned} \tag{9.14}$$

$$a(s, p_a) = (1 - b_0(s, p_a)) b_0(s, p_a) d(s, p_a) \left(-p_c'(s) \right). \tag{9.15}$$

On utilise les définitions (9.14) et (9.15) :

$$d(s, p_a) \vec{u}_w = d(s, p_a) b_0(s, p_a) \vec{u} - K d(s, p_a) a(s, p_a) \vec{\nabla} s + K d(s, p_a) b_\rho(s, p_a) \vec{g}.$$

On en déduit finalement que

$$\vec{u}_w = -K a(s, p_a) \vec{\nabla} s + b_0(s, p_a) \vec{u} + K b_\rho(s, p_a) \vec{g}. \tag{9.16}$$

9.1.2.2 Équations en pression

En sommant les deux équations de conservation de la masse, on obtient

$$\phi \frac{\partial}{\partial t} \left(c_a(1-s)p_a + \rho_w s \right) + \text{div } \vec{u} = 0. \tag{9.17}$$

On peut expliciter $\vec{u}$ en fonction des grandeurs définies précédemment. Pour cela on utilise d'abord les lois de Darcy : on a

$$\begin{aligned} \vec{u} &= \vec{u}_a + \vec{u}_w \\ &= -\rho_a(p_a) K k_a(s) \left(\vec{\nabla} p_a - \rho_a(p_a) \vec{g} \right) - K \rho_a(p_a) k_a(s) \vec{\nabla} p_a - K \rho_w k_w(s) \left(\vec{\nabla} p_a - p_c'(s) \vec{\nabla} s \right). \end{aligned}$$

On définit

$$\gamma_\rho(s, p_a) = \frac{\rho_w^2 k_w(s) + \rho_a^2 k_a(s)}{d(s, p_a)}. \tag{9.18}$$

Alors

$$\vec{u} = K \gamma_\rho(s, p_a) d(s, p_a) \vec{g} - K d(s, p_a) \vec{\nabla} p_a + K b_0(s, p_a) d(s, p_a) p_c'(s) \vec{\nabla} s.$$

Finalement,

$$\vec{u} = -K d(s, p_a) \left(\vec{\nabla} p_a - b_0(s, p_a) p_c\prime(s) \vec{\nabla} s - \gamma_\rho(s, p_a) \vec{g} \right). \tag{9.19}$$

En formulation « pression en air », nous chercherons à résoudre le système { (9.9)-(9.16),(9.17)-(9.19)}.

On peut remarquer la présence d'un terme de diffusion en saturation dans l'équation en pression. Ceci représente un couplage fort entre les deux groupes d'équations.

9.2 La formulation pression globale

Cette formulation, très efficace pour la modélisation des écoulements pétroliers, a été introduite par Guy Chavent pour le cas compressible en 1981 ([25] et [15]). L'objectif est d'éliminer le terme de diffusion en saturation dans l'équation en pression (9.19). Pour y parvenir on cherche à définir une nouvelle variable artificielle, homogène a une pression, nommée « pression globale », telle qu'on puisse expliciter une fonction f vérifiant

$$\vec{\nabla} p_a - b_0(s, p_a) p_c'(s) \vec{\nabla} s = f(s, p) \vec{\nabla} p. \tag{9.20}$$

On construit la pression globale pour le cas qui nous intéresse, c'est-à-dire eau incompressible et air faiblement compressible.

La pression globale sera du même ordre de grandeur que p_a et p_w. On suppose que p vérifie une égalité de la forme

$$p = \frac{1}{2}(p_a + p_w) + \tilde{\gamma}(s, p).$$

Cherchons à quelles conditions sur $\tilde{\gamma}$ et f l'égalité (9.20) est vérifiée. On a

$$\vec{\nabla} p = \frac{1}{2}\left(2\vec{\nabla} p_a - p_c'(s)\vec{\nabla} s\right) + \frac{\partial \tilde{\gamma}}{\partial s}\vec{\nabla} s + \frac{\partial \tilde{\gamma}}{\partial p}\vec{\nabla} p$$

i.e.

$$\left(1 - \frac{\partial \tilde{\gamma}}{\partial p}\right)\vec{\nabla} p = \vec{\nabla} p_a + \left(\frac{\partial \tilde{\gamma}}{\partial s} - \frac{1}{2}p_c'(s)\right)\vec{\nabla} s.$$

L'égalité (9.20) sera donc vérifiée si

$$\frac{\partial \tilde{\gamma}}{\partial s} = \left(\frac{1}{2} - b_0(s, p_a)\right) p_c'(s) \tag{9.21}$$

$$f(p) = 1 - \frac{\partial \tilde{\gamma}}{\partial p}. \tag{9.22}$$

9.2.1 Fonction auxiliaire

On introduit une fonction auxiliaire γ.
Soit $s_c \in [0, 1]$ caractérisé par

$$p_c(s_c) = 0 \text{ ou } [p_c(1) > 0 \text{ et } s_c = 1]. \tag{9.23}$$

On va supposer dans la suite que $s_c = 1$, ce qui ne change rien à la généralité des résultats qui seront obtenus. On définit alors la fonction auxiliaire γ par

$$\forall s \in]0, 1] \quad \forall p \in [p_w, p_w + p_c(s)] \quad \gamma(s, p) = \int_{s_c}^{s} \left(b_0(\sigma, p) - \frac{1}{2}\right)(-p_c'(\sigma))\, \mathrm{d}\sigma. \tag{9.24}$$

On a $\dfrac{\partial \gamma}{\partial s} = \left(\dfrac{1}{2} - b_0(s, p)\right) p_c'(s)$. L'égalité (9.21) est vérifiée de manière approchée pour $\tilde{\gamma} = \gamma$ à condition que $b_0(s, p)$ reste proche de $b_0(s, p_a)$.

9.2.2 Définition de la pression globale

La pression globale est alors définie de façon implicite :

$$\begin{cases} p_w \leq p_{globale} \leq p_a, \\ p_{globale} = \dfrac{1}{2}(p_a + p_w) + \gamma(s, p_{globale}). \end{cases} \tag{9.25}$$

La validité de sa définition est donc liée à un théorème de point fixe. Intéressons-nous aux conditions d'existence de ce point fixe.

Majoration en norme de γ. Par définition, la fonction b_0 est bornée par 0 et 1. De plus p_c est monotone décroissante. On en déduit

$$\forall s, p \quad |\gamma(s, p)| \leq \frac{1}{2}\left(p_c(s) - p_c(1)\right). \tag{9.26}$$

Majoration en norme de $\frac{\partial\gamma}{\partial p}$. Comme p_c est décroissante positive, on a

$$\left|\frac{\partial\gamma}{\partial p}\right| = \left|\int_1^s \frac{\partial b_0}{\partial p}\left|-\frac{\mathrm{d}p_c}{\mathrm{d}\sigma}\right|\mathrm{d}\sigma\right| \leq \max\left(\left|\frac{\partial b_0}{\partial p}\right|\right) p_c(s)$$

On rappelle que $b_0 = \frac{\rho_w k_w}{\rho_w k_w + \rho_a k_a}$, ρ_w =cste, $k_w = k_w(s)$ et on a $\frac{\partial \rho_a}{\partial p} = c_a$ d'où

$$\frac{\partial b_0}{\partial p} = -\frac{db_0}{d^2}\left(c_a k_a\right) = -\frac{b_0}{d} c_a \frac{(1-b_0)d}{\rho_a} = -\frac{b_0(1-b_0)}{p}.$$

Or $0 \leq b_0 \leq 1$ donc $0 \leq b_0(1-b_0) \leq \frac{1}{4}$. Finalement,

$$\forall s, p \qquad \left|\frac{\partial\gamma}{\partial p}\right| \leq \frac{1}{4}\frac{p_c(s)}{p} \tag{9.27}$$

Cas où la pression capillaire est faible. Pour tous s, p_w, l'application

$$p \mapsto \frac{1}{2}\left(p_a + p_w\right) + \gamma(s,p)$$

est à valeurs dans $[p_a, p_w]$ grâce à l'inégalité (9.26). Si $p_c(s)$ n'est pas trop grand alors elle est contractante grâce à l'inégalité (9.27). On en déduit par un théorème du point fixe l'existence d'un unique réel p_0 tel que

$$p_w \leq p_0 \leq p_a,$$

$$p_0 = \frac{1}{2}\left(p_a + p_w\right) + \gamma(s, p_0).$$

Cette égalité caractérise la pression globale que nous la noterons tout simplement p.

On va supposer, sans pour le moment préciser quels sont les ordres de grandeur à conserver, que la pression capillaire est suffisamment faible pour que l'on puisse remplacer p_a par p dans la plupart des fonctions définies dans la section 9.1.2. On peut alors écrire

$$\frac{\partial \vec{u}}{\partial t} = \frac{\partial}{\partial t}\left[(c_a p - \rho_w)(1-s)\right].$$

On a de plus

$$\vec{\nabla} p = \frac{1}{2}(\vec{\nabla} p_a + \vec{\nabla} p_w) + \frac{\partial\gamma(s,p)}{\partial s}\vec{\nabla} s + \frac{\partial\gamma(s,p)}{\partial p}\vec{\nabla} p,$$

donc

$$\begin{aligned}\left(1 - \frac{\partial\gamma(s,p)}{\partial p}\right)\vec{\nabla} &= \frac{1}{2}(\vec{\nabla} p_a + \vec{\nabla} p_w) + (b_0 - \frac{1}{2})(-p_c'(s))\vec{\nabla} s \\ &= \vec{\nabla} p_a - b_0 p_c'(s)\vec{\nabla} s.\end{aligned}$$

Finalement, l'équation en pression devient

$$\phi \frac{\partial}{\partial t} \left[(c_a p - \rho_w)(1-s) \right] + \text{div } \vec{u} = 0 \tag{9.28}$$

où

$$\vec{u} = -Kd(s,p) \left[\left(1 + \frac{b_0(1-b_0)}{p} (p_c(s) - p_c(0)) \right) \vec{\nabla} p - \gamma_\rho \vec{g} \right]. \tag{9.29}$$

On a donc éliminé le terme de diffusion en saturation.

L'équation en saturation n'a pas été modifiée, elle reste de la forme (9.9), (9.16) :

$$\phi \frac{\partial \rho_w s}{\partial t} + \text{div } \ \vec{u}_w = 0 \tag{9.30}$$

où

$$\vec{u}_w = -Ka(s,p_g)\vec{\nabla} s + b_0(s,p_g)\vec{u} + Kb_\rho(s,p_g)\vec{g}. \tag{9.31}$$

On a simplement supposé que les égalités (9.9) et (9.16) étaient toujours vérifiées en remplaçant p_a par p_g.

Applications numériques. À partir du modèle de van Genuchten, explicité dans le Tableau 9.1, pour plusieurs valeurs des paramètres, nous avons représenté l'étendue des valeurs que pouvaient prendre les fonctions définissant les équations diphasiques en formulation pression globale, pour des pressions variant dans un intervalle d'amplitude $p_c(s)$, pour chaque valeur de saturation $s \in [0,1]$ (Figures 9.3 - 9.4 - 9.5 - 9.6). Les courbes « pertinence de la définition de la pression globale » représentent le minimum atteint par l'application $p \to \|p - \frac{1}{2}(p + p_w) + \gamma(s,p)\|$ pour p variant de p_w à $p_w + p_c(s)$.

9.2.3 Propriétés du nouveau système d'équations

Guy Chavent a pu établir des théorèmes d'existence de solution pour cette formulation. De plus, la pression globale représentée comme une fonction de la saturation en eau est plus régulière que la pression en eau ou la pression en air ; par conséquent, on mettra plus facilement en place pour cette formulation des schémas numériques stables. Enfin, des observations numériques montrent que la formulation pression globale sera a priori d'autant plus valable qu'on est en présence de fortes saturations, alors que l'équation de Richards est valable pour des milieux faiblement saturés : leurs domaines de validité sont complémentaires.

L'existence du point fixe définissant la pression globale est liée aux valeurs de la pression capillaire : en effet, l'application γ est contractante à condition que la pression capillaire reste faible devant la pression en eau. Or la pression capillaire prend des valeurs beaucoup plus fortes pour les écoulements compressibles eau/air que pour les écoulements pétroliers, moins compressibles.

De plus dans notre cas, les valeurs de la pression globale sont souvent beaucoup plus proches des valeurs de la pression en eau que de celles de la pression en air. Par conséquent, pour des fortes valeurs de p_c, l'approximation $p_a \approx p_g$ ne sera plus valable.

Paramètres de base

Paramètres communs à la pression capillaire et aux perméabilités relatives :
$\theta_s \in]0,1[$, $\theta_r \in]0,1[$ (valeurs minimum et maximum autorisées pour s) et $n \in [1,+\infty[$;
pour définir la pression capillaire : $\alpha \in]0,1[$;
pour définir les perméabilités relatives : $\epsilon \in]0,1[$ et $\gamma \in]0,1[$.

Variables intermédiaires

$$\overline{s}_w = \frac{s-\theta_r}{\theta_s-\theta_r}$$
$$\overline{s}_a = \frac{\theta_s-s}{\theta_s-\theta_r}$$
$$m = 1-\frac{1}{n}$$

Équation de la pression capillaire

$$p_c(s) = \frac{1}{\alpha}\left(\overline{s}_w^{-\frac{1}{m}}-1\right)^{\frac{1}{n}} \quad ;$$

Équation de la perméabilité relative de l'eau

$$k_{r_w}(s) = \overline{s}_w^{\epsilon}\left(1-\left(1-\overline{s}_w^{\frac{n}{n-1}}\right)^{\frac{n-1}{n}}\right)^2 \quad ;$$

Équation de la perméabilité relative de l'air

$$k_{r_a}(s) = \overline{s}_a^{\gamma}\left(1-(1-\overline{s}_a)^{\frac{n}{n-1}}\right)^{\frac{2(n-1)}{n}} \quad .$$

Tab. 9.1 – Modèle de van Genuchten

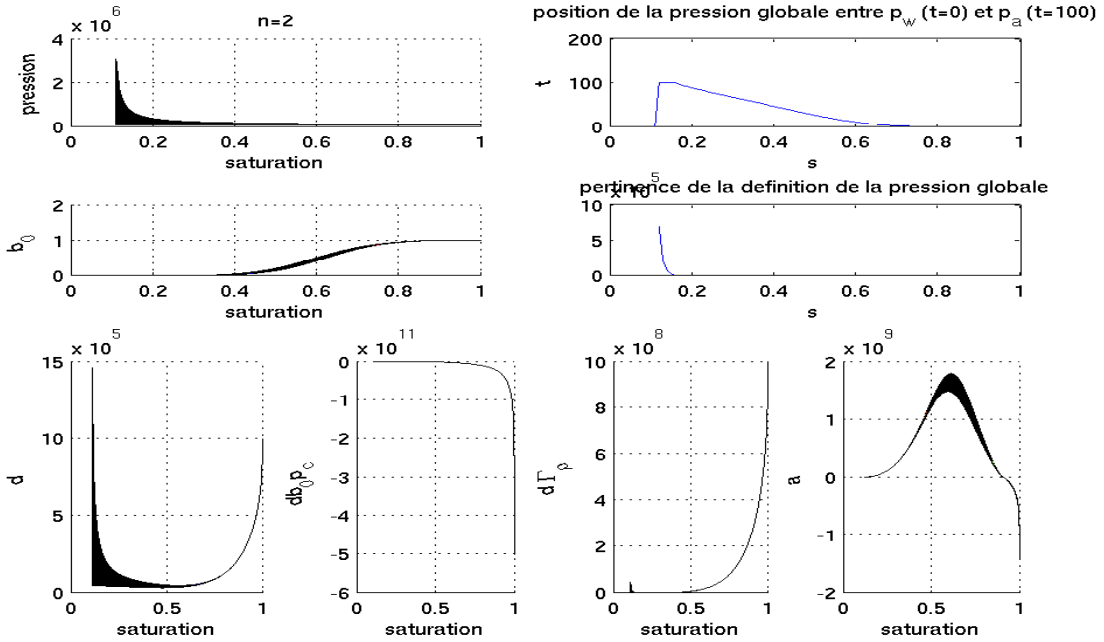


FIG. 9.3 – Courbes noires : valeurs des coefficients intervenant dans les équations diphasiques { (9.9)-(9.16),(9.17)-(9.19) } lorsque leur deuxième argument p_a varie entre p_w et $p_w+p_c(s)$ (il s'agit d'une analyse de sensibilité globale, manuelle). Les courbes représentent, de gauche à droite et de haut en bas, en fonction de la saturation en eau, (i) $p-p_w$; (ii) $100\dfrac{p_g-p_w}{p_a-p_w}$; (iii) $b_0(s,p)$ (9.13) ; (iv) $\min_{p\in[p_w,p_w+p_c(s)]}\left\|p-\frac{1}{2}(p+p_w)+\gamma(s,p)\right\|$ (le point fixe définissant la pression globale existe lorsque ce minimum est nul) ; (v) $d(s,p)$ (9.12) ; (vi) $d(s,p)b_0(s,p)p_s(s)$; (vii) $d(s,p)\gamma_\rho(s,p)$ (9.18) ; (viii) $a(s,p)$ (9.15). Le modèle de van Genuchten 9.1 est utilisé avec $n=2$.

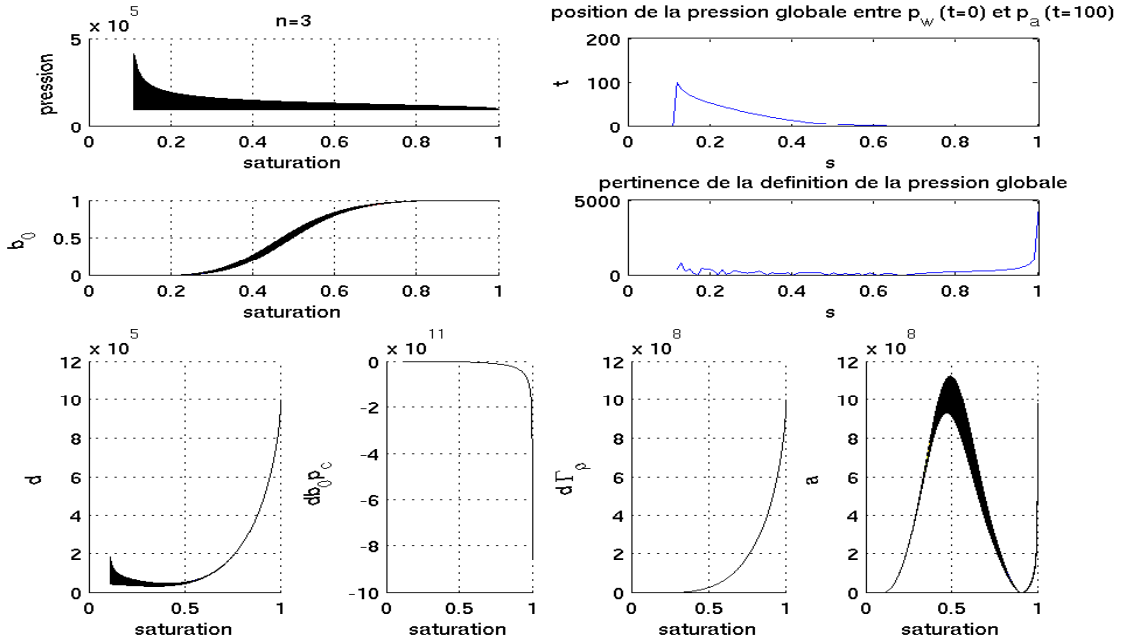


FIG. 9.4 – Courbes noires : valeurs des coefficients intervenant dans les équations diphasiques { (9.9)-(9.16),(9.17)-(9.19) } lorsque leur deuxième argument p_a varie entre p_w et $p_w+p_c(s)$ (il s'agit d'une analyse de sensibilité globale, manuelle). Les courbes représentent, de gauche à droite et de haut en bas, en fonction de la saturation en eau, (i) $p-p_w$; (ii) $100\dfrac{p_g-p_w}{p_a-p_w}$; (iii) $b_0(s,p)$ (9.13) ; (iv) $\min_{p\in[p_w,p_w+p_c(s)]}\left\|p-\frac{1}{2}(p+p_w)+\gamma(s,p)\right\|$ (le point fixe définissant la pression globale existe lorsque ce minimum est nul) ; (v) $d(s,p)$ (9.12) ; (vi) $d(s,p)b_0(s,p)p_s(s)$; (vii) $d(s,p)\gamma_\rho(s,p)$ (9.18) ; (viii) $a(s,p)$ (9.15). Le modèle de van Genuchten 9.1 est utilisé avec $n=3$.

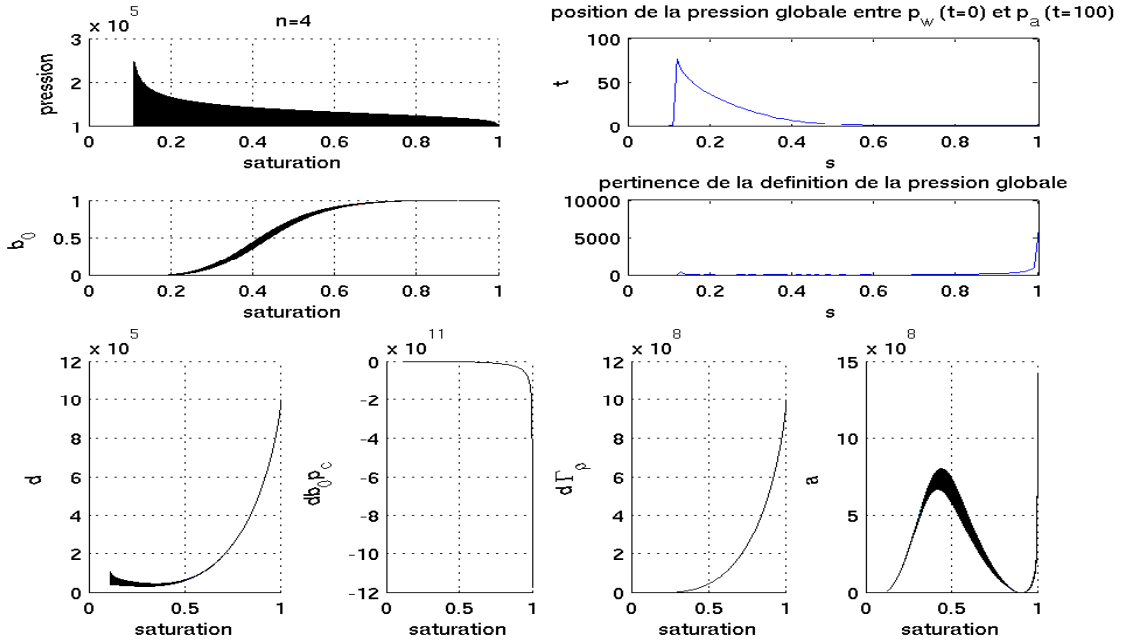


FIG. 9.5 – Courbes noires : valeurs des coefficients intervenant dans les équations diphasiques { (9.9)-(9.16),(9.17)-(9.19) } lorsque leur deuxième argument p_a varie entre p_w et $p_w+p_c(s)$ (il s'agit d'une analyse de sensibilité globale, manuelle). Les courbes représentent, de gauche à droite et de haut en bas, en fonction de la saturation en eau, (i) $p-p_w$; (ii) $100\dfrac{p_g-p_w}{p_a-p_w}$; (iii) $b_0(s,p)$ (9.13) ; (iv) $\min_{p\in[p_w,p_w+p_c(s)]}\left\|p-\frac{1}{2}(p+p_w)+\gamma(s,p)\right\|$ (le point fixe définissant la pression globale existe lorsque ce minimum est nul) ; (v) $d(s,p)$ (9.12) ; (vi) $d(s,p)b_0(s,p)p_s(s)$; (vii) $d(s,p)\gamma_\rho(s,p)$ (9.18) ; (viii) $a(s,p)$ (9.15). Le modèle de van Genuchten 9.1 est utilisé avec $n=4$.

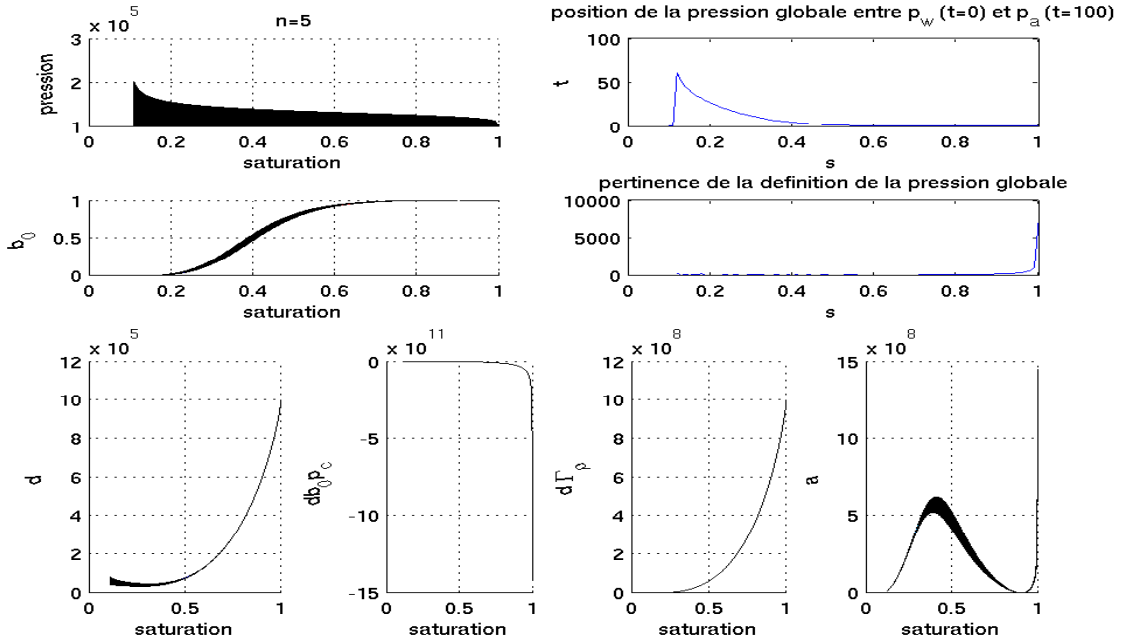


FIG. 9.6 – Courbes noires : valeurs des coefficients intervenant dans les équations diphasiques { (9.9)-(9.16),(9.17)-(9.19) } lorsque leur deuxième argument p_a varie entre p_w et $p_w+p_c(s)$ (il s'agit d'une analyse de sensibilité globale, manuelle). Les courbes représentent, de gauche à droite et de haut en bas, en fonction de la saturation en eau, (i) $p-p_w$; (ii) $100\dfrac{p_g-p_w}{p_a-p_w}$; (iii) $b_0(s,p)$ (9.13) ; (iv) $\min_{p\in[p_w,p_w+p_c(s)]}\left\|p-\frac{1}{2}(p+p_w)+\gamma(s,p)\right\|$ (le point fixe définissant la pression globale existe lorsque ce minimum est nul) ; (v) $d(s,p)$ (9.12) ; (vi) $d(s,p)b_0(s,p)p_s(s)$; (vii) $d(s,p)\gamma_\rho(s,p)$ (9.18) ; (viii) $a(s,p)$ (9.15). Le modèle de van Genuchten 9.1 est utilisé avec $n=5$.

Les hypothèses permettant d'utiliser la formulation pression globale ne sont donc pas vérifiées.

9.2.4 Conclusion

La pression globale est une pression artificielle, définie implicitement comme un point fixe et sa valeur est encadrée par celles de la pression en eau et de la pression en air,

$$p_w \leq p_g \leq p_a.$$

On forme les équations en pression gloabale en remplaçant la variable p_a par la variable p_g comme argument dans les fonctions non linéaires.

Des théorèmes d'existence ont été établis par Chavent pour les nouvelles équations ainsi obtenues.

Cependant l'existence du point fixe définisant la pression globale dépend des valeurs de $p_c = p_a = p_w$ et le passage la la formulation pression globale suppose que $p_a \approx p_g$. Or pour les écoulements eau-air on a en général

$$|p_g - p_a| > |p_g - p_w|$$

Le formulation pression globale sera donc pertinente pour les écoulements eau-air uniquement lorsque le milieu est presque saturé en eau. Son domaine de validité est complémentaire de celui de l'équation de Richards présentée dans la section 9.3.

9.3 L'équation de Richards

Les hydrogéologues utilisent souvent le modèle de Richards, qui est une approximation de modèle diphasique eau-air. Dans ce modèle, on suppose la pression en air constante et uniforme :

$$p_a = \text{ constante } = \text{ pression atmosphérique } \Longrightarrow \vec{\nabla} p_w = -\vec{\nabla} p_c$$

et l'équation de conservation en air disparaît. Ceci est vrai pour les milieux faiblement saturés. En effet, pour ces milieux, p_a est négligeable devant $p_c(s)$ et $\vec{\nabla} p_a$ est négligeable devant $p_c'(s)\vec{\nabla} s$ (voir la Figure 9.2.)

Le système se réduit à l'équation de conservation en eau :

$$\begin{aligned} &\phi \frac{\partial s}{\partial t} + \text{div } \vec{\varphi}_w = 0, \\ &\vec{\varphi}_w = -K k_w(s)\, \vec{\nabla}(-p_c(s) - \rho_w g z). \end{aligned} \tag{9.32}$$

Dans cette équation on a

– un terme de diffusion capillaire div $\Big(-K k_w(s)\, \vec{\nabla}(-p_c(s))\Big)$

– un terme d'advection par gravité div $\left(Kk_w(s)\,\vec{\nabla}(\rho_w g z)\right)$.

Le formalisme que nous avons utilisé jusqu'à maintenant est celui de la mécanique des fluides. Les hydrogéologues utilisent un formalisme différent et regroupent en particulier toutes les propriétés du sol et du fluide i dans un seul paramètre noté K_i. De plus ils ne quantifient pas la composition de l'écoulement par les saturations mais par la teneur en eau θ. Les pressions ne sont plus exprimées en Pascal mais en hauteur d'eau.

On réécrit l'équation de conservation de l'eau sous la forme

$$\frac{\partial \overbrace{\phi s}^{\theta}}{\partial t} + \text{div}\ \vec{\varphi}_w = 0, \qquad \vec{\varphi}_w = -\underbrace{Kk_w\rho_w g}_{K_w}\ \vec{\nabla}\left(\underbrace{\frac{p_w}{\rho_w g}}_{h} - z\right).$$

– θ, teneur en eau (water content) volumique
– h, potentiel matriciel (pressure head)
– K_w, conductivité hydraulique
– $\vec{g} = g\vec{\nabla}z$

Nous indiquons donc ici les correspondances entre ces formalismes :

$$\begin{array}{rclcrcl} h & = & \dfrac{-p_c}{\rho_w g} & \qquad & K_i & = & K_{si}k_{ri} \\ K_{si} & = & \dfrac{\rho_i}{\mu_i} g K & & \theta & = & \phi s. \end{array}$$

Avec ce nouveau formalisme et sous l'hypothèse que p_a est constant et uniforme, on obtient une nouvelle équation en saturation :

$$\frac{\partial \theta}{\partial t} = \text{div}\ \left(K_w \vec{\nabla}\varphi\right), \quad \varphi = h - z. \tag{9.33}$$

On remarque en particulier que les paramètres liés aux propriétés de l'air ont disparu. Cette équation est donc celle d'un écoulement monophasique (on a supposé que l'équation en pression devenait $\dfrac{\partial p}{\partial t} = 0$). Cette équation est étudiée en particulier dans [15, 45]. Les applications $h \to \theta$ et $h \to K_w$ ont des propriétés mathématiques plus intéressantes que $\theta \to h$ et $\theta \to K_w$. C'est pourquoi on résout en général l'équation (9.33) par rapport à l'inconnue h. Malheureusement, pour des raisons de simplification, on ne peut pas choisir cette inconnue dans le cas diphasique.

Concernant le terme de diffusion capillaire, on notera que dans le modèle de van Genuchten que l'on utilise (voir [77]) en fixant ϵ=0.5

$$p_c(s) = \frac{1}{\alpha}\left(\left(\frac{s - s_r}{1 - s_r}\right)^{-\frac{1}{m}} - 1\right)^{\frac{1}{n}}, \ n = 2, \ m = 1 - \frac{1}{n} = \frac{1}{2}, \ s_r = 0.2, \tag{9.34}$$

la dérivée de la pression capillaire en $s = 1$, donc le coefficient de diffusion, est infinie. Les courbes de pression capillaire et de perméabilité relative pour ce modèle sont représentées sur la Figure 9.2.

9.4 Discrétisation

On choisit d'implémenter la formulation diphasique « classique » et l'équation de Richards afin de les comparer.

Pour l'implémentation diphasique, on utilise une variante de la méthode IMPES (« implicit pressure explicit saturation », efficace pour les écoulements pétroliers, voir par exemple [17]) : on découple la résolution des équations en pression et en saturation, en utilisant éventuellement des pas de temps différents pour chaque type d'équation, pour s'adapter aux vitesses d'évolution de chaque variable.

Cette méthode est expliquée dans [17] dans le cas de deux fluides incompressibles. On suppose choisie une discrétisation en espace. On décompose l'intervalle de temps de simulation en N pas de temps limités par $t_0 < ... < t_N$. On dispose d'une condition initiale en saturation s^0. Dans le cas incompressible on verra que la pression est imposée par la saturation ; dans le cas général on a également besoin d'une condition initiale en pression.

On suppose connue la distribution de saturation s^n au temps t^n. Dans le cas de deux fluides incompressibles, on obtient par la loi de conservation de la masse div $(\vec{\varphi}_a + \vec{\varphi}_w) = 0$. La distribution de pression p_a^n est obtenue de manière implicite par cette égalité. Dans le cas général on doit également connaître la distribution de pression p_a^{n-1} et la distribution de saturation s^{n-1} et résoudre

$$\Phi \frac{c_a(1-s^n)p_a^n + \rho_w s^n - c_a(1-s^{n-1})p_a^{n-1} - \rho_w s^{n-1}}{\Delta t} + \text{div } \vec{u}(s^n, p_a^n) = 0. \tag{9.35}$$

On en déduit la distribution de $u^n = u(s^n, p^n)$.

On calcule ensuite s^{n+1} par le schéma explicite

$$\Phi \frac{\rho_w s^{n+1} - \rho_w s^n}{\delta t} + \text{div } \vec{u}_w(s^n, p^n) = 0. \tag{9.36}$$

Dans le cas compressible où la pression initiale est connue on commence par cette étape, il n'y a donc pas de problème à l'instant initial. Cette méthode est suggérée dans [73].

Il a également été observé dans le cas des fluides incompressibles que la résolution implicite de l'équation en pression est beaucoup plus longue que la résolution explicite de l'équation en saturation et que le schéma explicite pour l'équation en saturation n'est stable que pour des pas de temps très courts. Pour gagner en temps de calcul Chen, Huan et Li proposent dans [17] d'utiliser deux pas de temps et de résoudre successivement 1 fois l'équation de pression pendant un temps δt et M fois l'équation de saturation pendant un temps $\frac{\Delta t}{M}$.

On utilise une méthode de Newton (voir par exemple [8], [64], [22] ou [50]) pour résoudre les équations discrétisées, non linéaires.

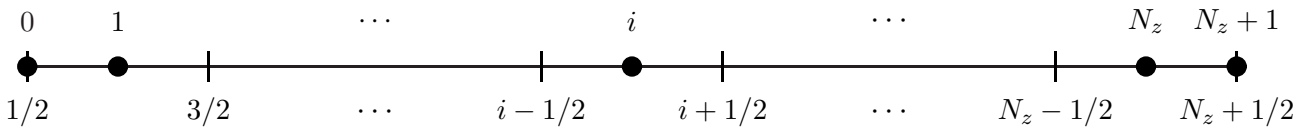


FIG. 9.7 – Discrétisation en espace 1D.

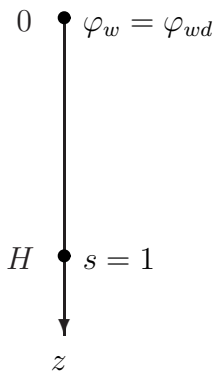
9.4.1 Un problème simple en dimension 1 - approximation de Richards

9.4.1.1 Problème modèle

L'équation en dimension 1, le long d'un axe z orienté vers le bas, s'écrit

$$\phi\frac{\partial s}{\partial t} + \frac{\partial}{\partial z}\left(-Kk_w(s)\frac{\partial}{\partial z}(-p_c(s) - \rho_w g z)\right) = 0. \tag{9.37}$$

On s'intéresse à un écoulement monodimensionnel dans une colonne homogène verticale ouverte de hauteur H.



La condition aux limites φ_{wd} correspond à une intensité de pluie.
L'état initial est l'état stationnaire correspondant à $\varphi_{wd} = 0$.
Il vérifie

$$\text{Condition initiale } : p_c(s) = \rho_w g(H - z). \tag{9.38}$$

9.4.1.2 Discrétisation

On utilise un schéma d'Euler implicite avec décentrage amont pour l'advection (voir la Figure 9.7) :

$$\phi\frac{s_i^{n+1} - s_i^n}{\Delta t} + \frac{\varphi_{w,i+1/2}^{n*} - \varphi_{w,i-1/2}^{n*}}{h} = 0, \quad i = 1, \cdots, N_z \tag{9.39}$$

avec

$$\varphi_{w,i+1/2}^{n*} = k_{i+1/2}^{*,n+1}\,\frac{p_c(s_{i+1}^{n+1}) - p_c(s_i^{n+1})}{h} + \rho_w g K k_w(s_i^{n+1}),\ i = 1, \cdots, N_z. \tag{9.40}$$

Nous avons effectué des essais en définissant $k_{i+1/2}^{*,n+1}$ comme la moyenne harmonique entre $Kk_w(s_i^{n+1})$ et $Kk_w(s_{i+1}^{n+1})$ puis comme la valeur amont.

À la surface, le flux entrant est l'intensité de pluie si elle peut être absorbée, le flux maximal pouvant être absorbé sinon. Précisons ce qu'est le flux maximal pouvant être absorbé. La saturation s_0 ne fait pas partie des degrés de liberté du problème mais le flux entrant vérifie

$$\varphi_{w,1/2}^{n*} = k^*(s_0^{n+1}, s_1^{n+1}) \frac{p_c(s_1^{n+1}) - p_c(s_0^{n+1})}{h/2} + \rho_w g K k_w(s_0^{n+1}). \tag{9.41}$$

Pour s_1^{n+1} fixé, ce flux croît avec s_0 qui prend au maximum la valeur 1. Cette condition s'écrit

$$\varphi_{w,1/2}^{n*} = \min\left(\varphi_{wd}\ ,\ k^*(s=1, s_1^{n+1}) \frac{p_c(s_1^{n+1})}{h/2} + \rho_w g K k_w(s=1)\right). \tag{9.42}$$

Influence de la définition des perméabilités aux interfaces entre les cellules. On choisit d'abord la condition aux limites $\varphi_{wd} = 0$. La condition initiale est la discrétisation de l'état stationnaire. Si on résout exactement l'équation, alors la solution est constante. Sinon on tend asymptotiquement vers un nouvel état stationnaire discret.

On choisit comme condition initiale une approximation constante par cellule de l'état stationnaire, c'est à dire de la solution de l'équation

$$\frac{\partial}{\partial z}\left(-K k_w(s) \frac{\partial}{\partial z}(-p_c(s) - \rho_w g z)\right) = 0. \tag{9.43}$$

Cette condition initiale vérifie

$$p_c(s_i) = \rho_w g (H - z_i), \quad i = 1, \cdots, N_z. \tag{9.44}$$

L'état stationnaire pour l'équation discrétisée est la solution de

$$\varphi_{w,i+1/2}^* - \varphi_{w,i-1/2}^* = 0, \quad i = 1, \cdots, N_z \tag{9.45}$$

avec

$$\varphi_{w,i+1/2}^* = k_{i+1/2}^* \frac{p_c(s_{i+1}^n) - p_c(s_i^n)}{h} + \rho_w g K k_w(s_i^n),\ i = 1, \cdots, N_z. \tag{9.46}$$

Le résultat numérique est donné sur la Figure 9.8 : si on utilise la moyenne harmonique

$$k_{i+1/2}^{*,n+1} = \frac{2K k_w(s_i^{n+1}) k_w(s_{i+1}^{n+1})}{k_w(s_i^{n+1}) + k_w(s_{i+1}^{n+1})}. \tag{9.47}$$

la solution de l'équation (9.44) est différente de la solution de (9.45). En revanche si on utilise

$$k_{i+1/2}^{*,n+1} = K k_w(s_i^{n+1}). \tag{9.48}$$

les solutions de (9.44) et (9.45) se superposent. En effet, l'état stationnaire pour l'équation discrétisée vérifie

$$\varphi_{w,i+\frac{1}{2}}^* - \varphi_{w,i-1/2}^* = 0, \quad i = 1, \ldots, N_x$$

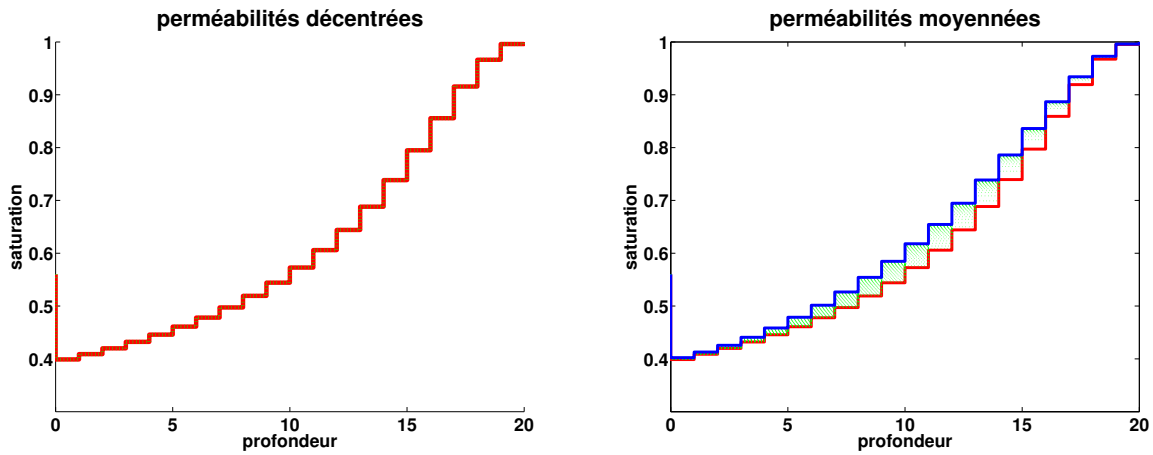


FIG. 9.8 – Influence de la définition des valeurs de perméabilité aux interfaces.

i.e.

$$p_c(s_{i+1}) - p_c(s_i) \quad = \quad -h\rho_w g \frac{K k_w(s_i^n)}{k^*_{i+1/2}}, \quad i = 2, \ldots, N_x - 1.$$

Donc la linéarité de p_c (équation (9.44)) est respectée uniquement si on choisit la valeur amont pour les coefficients non linéaires. L'ampleur de la différence entre les deux résultats est dûe à la grande courbure de la fonction k_w : pour s compris entre s_1 et s_2 avec $s_1 < s_2$, pour $\|s - s_1\| \leq \|s_2 - s\|$ on aura $\|k_w(s) - k_w(s_1\| \ll \|k_w(s) - k_w(s_2)\|$.

Des résultats numériques pour $\varphi_{wd} \neq 0$ sont donnés dans la section 9.4.3

9.4.2 Un problème simple en dimension 1 - modèle diphasique

9.4.2.1 Problème modèle

Les équations diphasiques en dimension 1 s'écrivent

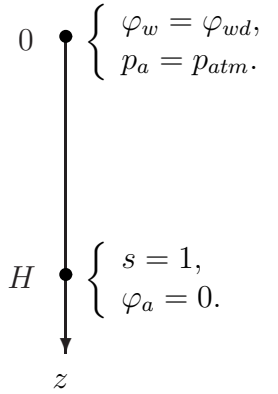
$$\left\{ \begin{array}{l} \phi \dfrac{\partial s}{\partial t} + \dfrac{\partial}{\partial z}\left[-K k_w(s) \left(\dfrac{\partial}{\partial z}(-p_c(s) + p_a) - \rho_w g \right) \right] = 0 \\ \phi \dfrac{\partial \left(c_a p_a (1-s) \right)}{\partial t} + \dfrac{\partial}{\partial z}\left[-K k_a(s) c_a p_a \left(\dfrac{\partial}{\partial z} p_a - c_a p_a g \right) \right] = 0 \end{array} \right. \tag{9.49}$$

- $s^{n,0} = s^n$
- Pour $k = 0, \cdots, n_s - 1$

$$\begin{cases} \phi \dfrac{s^{n,k+1} - s^{n,k}}{\Delta t_s} + \text{div } \vec{\varphi}_w = 0, \\ \vec{\varphi}_w = -K k_w^{n,k+1} \left(\vec{\nabla} p_a^n - \vec{\nabla} p_c^{n,k+1} - \rho_w \vec{g} \right) \end{cases}$$

- $s^{n+1} = s^{n,n_s}$
- $$\begin{cases} \phi c_a \dfrac{s_a^{n+1} p_a^{n+1} - s_a^n p_a^n}{\Delta t_p} + \text{div } \vec{\varphi}_a = 0, \\ \vec{\varphi}_a = -K k_a^{n+1} c_a p_a^{n+1} \left(\vec{\nabla} p_a^{n+1} - c_a p_a^{n+1} \vec{g} \right) \end{cases}$$

TAB. 9.2 – Discrétisation en temps pour la modélisation des écoulements diphasiques.



L'état initial est l'état stationnaire correspondant à $\varphi_{wd} = 0$.
Pour l'air on vérifie $\vec{\nabla} p_a - c_a p_a \vec{g} = \vec{0}$, i.e.

$$\text{Condition initiale pour l'air : } p_a = p_{atm} e^{c_a g z}. \tag{9.50}$$

Pour l'eau on vérifie $\vec{\nabla} p_w - \rho_w \vec{g} = \vec{0}$, i.e.

$$\begin{aligned} &\text{Condition initiale pour l'eau :} \\ &p_c(s) = \rho_w g (H - z) - p_a(H) + p_a(z). \end{aligned} \tag{9.51}$$

9.4.2.2 Discrétisation en temps

On note $\Delta t_p = \Delta t$ le pas de temps, n_s le nombre de résolutions d'équations en saturation par résolution d'équation en pression et $\Delta t_s = \Delta t / n_s$.

On liste dans le Tableau 9.2 les opération permettant de passer des champs de saturation en eau s^n et de pression en air p_a^n à l'instant $t^n = n\Delta t$ aux champs de saturation en eau s^{n+1} et de pression en air p_a^{n+1} à l'instant $t^{n+1} = (n+1)\Delta t$.

9.4.2.3 Discrétisation en espace pour l'équation en saturation

On utilise un schéma d'Euler implicite avec décentrage amont pour l'advection (voir la Figure 9.7) :

$$\phi \frac{s_i^{n+1} - s_i^n}{\Delta t} + \frac{\varphi_{w,i+1/2}^{n*} - \varphi_{w,i-1/2}^{n*}}{h} = 0, \quad i = 1, \cdots, N_z \tag{9.52}$$

avec

$$\varphi_{w,i+1/2}^{n*} = k_{i+1/2}^{*,n+1} \frac{p_c(s_{i+1}^{n+1}) - p_{a,i+1}^n - p_c(s_i^{n+1}) + p_{a,i}^n}{h} + \rho_w g K k_w(s_i^{n+1}), \ i = 1, \cdots, N_z. \tag{9.53}$$

$k^{*,n+1}_{i+1/2}$ peut être la moyenne harmonique entre $Kk_w(s_i^{n+1})$ et $Kk_w(s_{i+1}^{n+1})$ ou la valeur amont.

Le flux entrant est l'intensité de pluie si elle peut être absorbée, le flux maximal pouvant être absorbé sinon, i.e.

$$\varphi^{n*}_{w,1/2} = \min\left(\varphi_{wd}\ ,\ k^*(s=1, s_1^{n+1})\frac{p_c(s_1^{n+1}) - p^n_{a,1} + p_{atm}}{h/2} + \rho_w g K k_w(s=1)\right). \quad (9.54)$$

9.4.2.4 Discrétisation en espace pour l'équation en pression

On utilise un schéma d'Euler implicite avec décentrage amont pour l'advection (voir la Figure 9.7) :

$$\phi c_a \frac{s^{n+1}_{a,i} p^{n+1}_{a,i} - s^n_{a,i} p^n_i}{\Delta t_p} + \frac{\varphi^*_{a,i+1/2} - \varphi^*_{a,i-1/2}}{h} = 0, i = 1, \cdots, N_z \quad (9.55)$$

avec

$$\varphi^*_{a,i+1/2} = -Kk^{H,n+1}_{a,i+1/2} c_a p^{n+1}_{a,i+1/2} \frac{p^{n+1}_{a,i+1} - p^{n+1}_{a,i}}{h} + c_a K g k^{n+1}_{a,i+1} p^{n+1}_{a,i+1}. \quad (9.56)$$

$k^{H,n+1}_{a,i+1/2}$ est la moyenne harmonique entre $k_a(s^{n+1}_{w,i})$ et $k_a(s^{n+1}_{w,i+1})$

9.4.3 Résultats numériques comparatifs

9.4.3.1 Évolution de la saturation - cas 1

On impose une intensité de pluie pouvant être absorbée, c'est à dire telle que $\varphi^{n*}_{w,1/2} = \phi_{wd}$ lorsque $s_1^{n+1} = 1$ dans les équations (9.42) et (9.54). Les test sont normalisés, de sorte que l'intensité maximale pouvant être absorbée est $\varphi_{wd} = 1$. La Figure 9.9 représente l'évolution de la saturation suivant les modèles des sections 9.4.1 et 9.4.2. Les résultats ne diffèrent pas suivant le modèle choisi. On donne la différence entre les résultats fournis par les deux modèles sur la Figure 9.10.

9.4.3.2 Évolution de la saturation - cas 2

On impose maintenant une intensité de pluie légèrement supérieure à l'intensité maximale pouvant être absorbée, c'est à dire telle que dans les équations (9.42) et (9.54), $\varphi^{n*}_{w,1/2}$ est un peu inférieur à ϕ_{wd} lorsque $s_1^{n+1} = 1$. Les résultats sont donnés à la Figure 9.11. Les deux modèles donnent des résultats identiques jusqu'à ce que la surface soit saturée. Ensuite, le modèle diphasique est « en retard » sur le modèle de Richards : l'augmentation de la pression en air empêche l'eau de pénétrer dans le sol (l'air est piégé dans le sol).

9.4.3.3 Évolution de la saturation - cas 3

On impose maintenant une intensité de pluie beaucoup plus forte que l'intensité maximale pouvant être absorbée, c'est à dire telle que $\varphi^{n*}_{w,1/2} << \phi_{wd}$ lorsque $s_1^{n+1} = 1$ dans les équations (9.42) et (9.54). Les résultats sont donnés à la Figure 9.12. On observe la même tendance que précédemment, le retard du modèle diphasique est encore plus marqué.

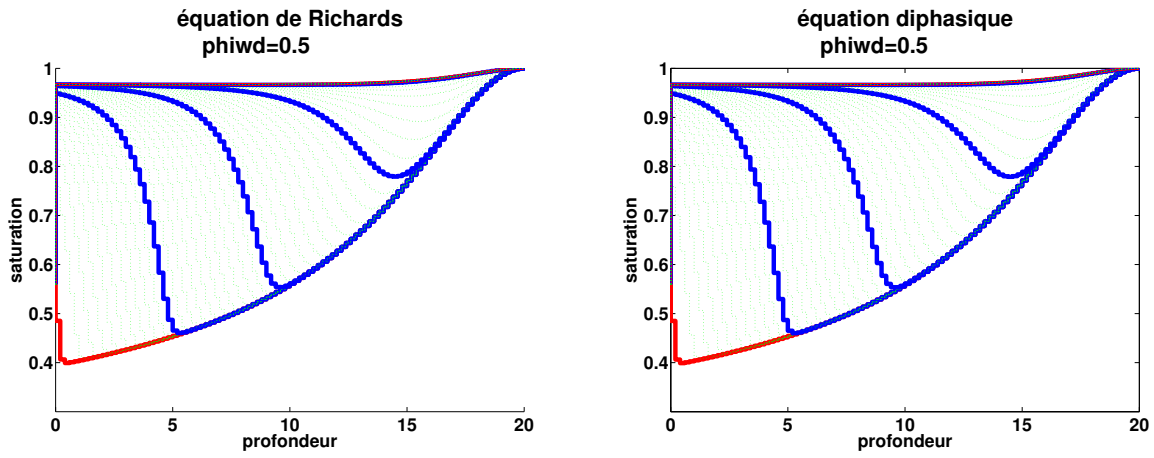


FIG. 9.9 – Évolution de la saturation en eau, suivant le modèle de Richards et le modèle diphasique, pour une faible intensité de pluie (pas de ruissellement). Chaque courbe (rouge, verte ou bleue) correspond à un instant et la saturation en chaque point augmente avec le temps. Les courbes correspondant à des instants identiques sont représentées de la même couleur sur les deux images.

9.4.3.4 Pressions en air

La Figure 9.13 représente l'évolution de la pression en air suivant le modèle diphasique (pressions en mètres de colonne d'eau).

9.5 Conclusion

L'influence de l'air est importante quand la surface est saturée.

On obtiendrait de plus grosses différences avec une colonne fermée, ce qui demande des tests au moins en deux dimensions.

On doit adapter les schémas aux grandes pentes des coefficients de diffusion (i.e. dans notre cas, utiliser (9.48) au lieu de (9.47)).

Il est prévu d'effectuer une comparaison similaire en deux dimensions puis en trois dimensions, par une formulation éléments finis mixtes en diffusion et volumes finis [49] en convection avec convection explicite et diffusion implicite. Nous avons programmé un code diphasique tridimensionnels à l'aide de la bibliothèque `LifeV`, mais il ne respecte pas encore toutes les spécificités dont l'importance est mise en évidence dans ce chapitre, comme le décentrage en diffusion.

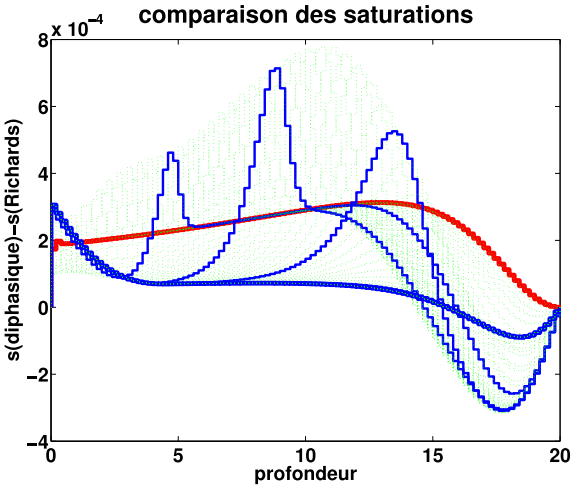


FIG. 9.10 – Différence entre les saturations en eau calculés par le modèle diphasique et le modèle de Richards. Chaque courbe (rouge, verte ou bleue) correspond à un instant.

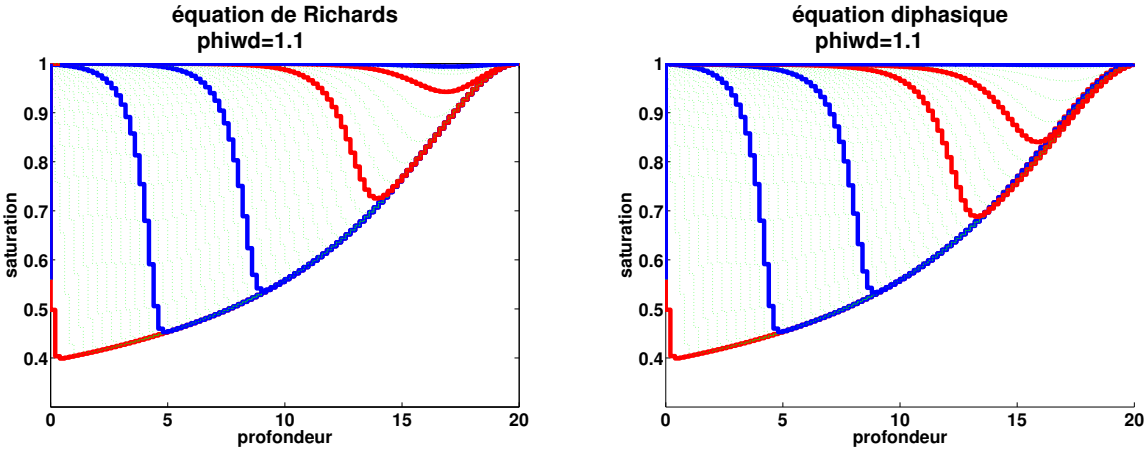


FIG. 9.11 – Évolution de la saturation en eau, suivant le modèle de Richards et le modèle diphasique. Chaque courbe (rouge, verte ou bleue) correspond à un instant et la saturation en chaque point augmente avec le temps. Les courbes correspondant à des instants identiques sont représentées de la même couleur sur les deux images.

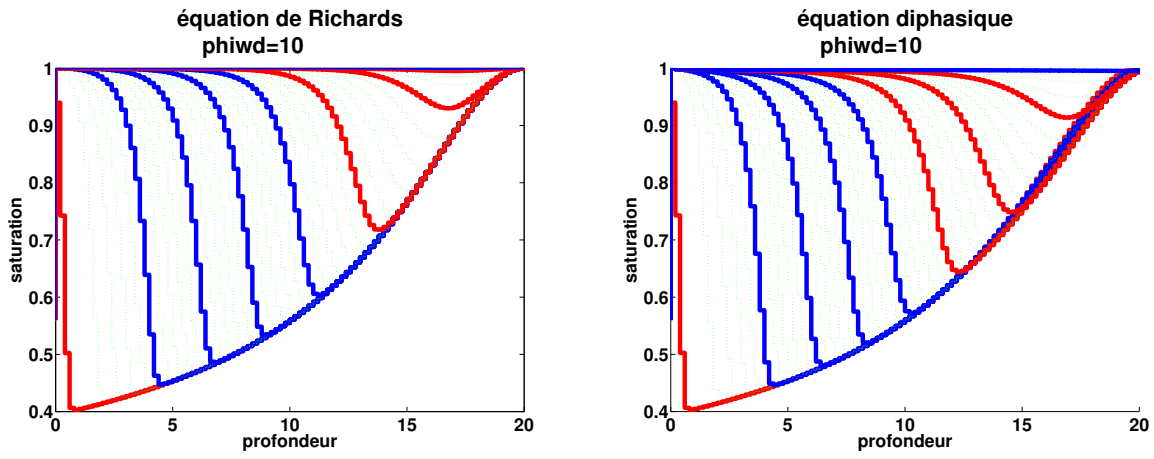


FIG. 9.12 – Évolution de la saturation en eau, suivant le modèle de Richards et le modèle diphasique, pour une forte intensité de pluie. Chaque courbe (rouge, verte ou bleue) correspond à un instant. Les courbes correspondant à des instants identiques sont représentées de la même couleur sur les deux images.

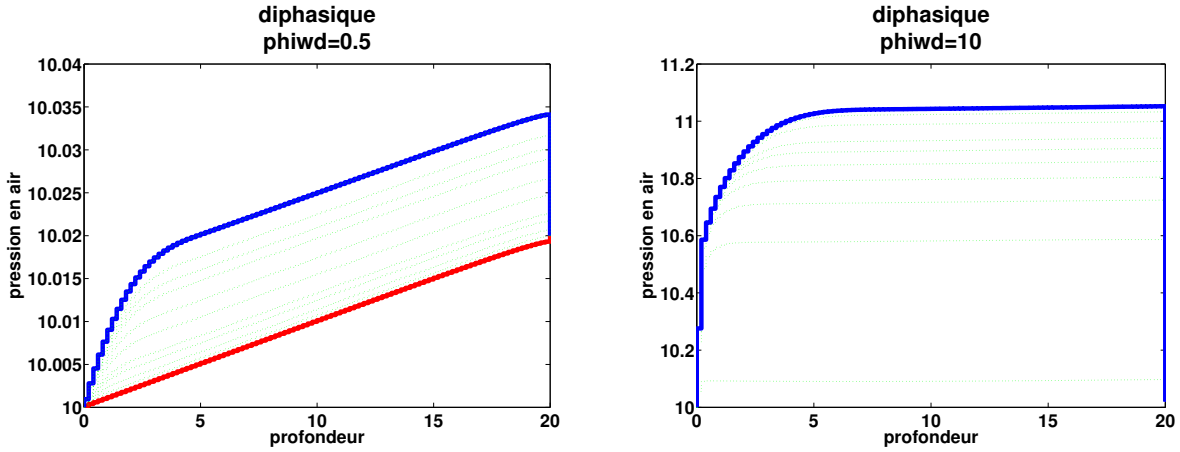


FIG. 9.13 – Évolution de la pression en air, suivant le modèle diphasique, pour deux intensités de pluie différentes.

Chapitre 10

Conclusion générale

Cette thèse s'inscrit dans le contexte des calculs de sûreté pour un éventuel stockage souterrain de déchets nucléaires.

Nous avons d'abord fourni une description détaillée d'un modèle mathématique et de son implémentation, afin de permettre l'implémentation d'un code dérivé le plus simplement possible et de façon, dans la mesure du possible, modulaire, c'est-à-dire avec le plus de parties possibles réutilisables. Dans notre présentation du modèle mathématique, nous avons insisté en particulier sur la prise en compte des conditions aux limites.

Nous avons donné une présentation générale des méthodes d'analyse de sensibilité probabiliste de type Monte-Carlo, puis d'analyse déterministe, en mettant en évidence les similarités entre les indicateurs fournis par ces deux types d'approches.

Nous avons présenté diverses méthodes classiques de calcul de dérivées. Nous avons en particulier mis en évidence la limite des méthodes différences divisées, détaillé la mise en place d'une différentiation automatique par `ADOL-C` sur un exemple conséquent, comparé les performances de la dérivation par le différentiateur automatique `ADOL-C` avec celles de la dérivation à la main. La bibliothèque `ADOL-C` se révèle compétitive concernant les temps de calcul et surtout les temps de développement, mais ne peut pas être utilisée telle quelle pour les programmes très gourmands en place mémoire. En revanche c'est un bon outil de vérification, et il est plus simple de l'utiliser que de mettre en place une vérification par différences divisées — à condition que l'on ait déjà adapté les bibliothèques de calcul scientifique nécessaires. Nous avons rappelé la méthode de l'état adjoint et appuyé notre présentation sur l'exemple du problème d'écoulement. Enfin nous avons vu comment il est possible de combiner dans un même programme la différentiation manuelle et la différentiation automatique, ce qui est utile pour vérifier un calcul de dérivée sous-fonction par sous-fonction.

Nous avons présenté quelques aspects informatiques, en particulier liés à la programmation fonctionnelle, qui est parfaitement adaptée au développement de plate-formes génériques logicielles, par exemple de couplage de code. Nous avons développé en particulier une plate-forme d'analyse de sensibilité et une bibliothèque de communication `C++`, utilisée avec des plate-formes génériques ou pour l'échange sécurisé de données.

Nous avons appliqué la méthode d'analyse de sensibilité déterministe à un modèle

d'écoulement tridimensionnel et à un modèle écoulement-transport bidimensionnel. Pour le modèle d'écoulement, peu non-linéaire et où les paramètres d'entrée considérés sont homogènes (ce sont tous des composantes de conductivité hydraulique), les résultats d'analyse de sensibilité déterministe sont peu dépendants des valeurs des paramètres d'entrée et fournissent les mêmes conclusions qu'une analyse probabiliste de type Monte-Carlo. Pour le modèle de transport, faisant intervenir différents types de non linéarités et des paramètres de dimensions différentes (coefficients de diffusion ou conductivités hydrauliques), il est plus difficile de comparer les deux approches.

Nous avons donné des pistes pour la construction d'un indicateur de sensibilité par rapport à des paramètres de discrétisation. Elles peuvent permettre de comparer l'incertitude d'un résultat due à l'incertitude sur des paramètres physiques avec l'incertitude due à la discrétisation, et donc de mesurer l'utilité ou non d'effectuer des calculs précis. Cette contruction est possible pour des paramètres de discrétisation en temps. Elle semble plus difficile pour des paramètres de discrétisation en espace.

Enfin nous nous sommes intéressés aux écoulements diphasiques : nous avons mis en évidence l'insuffisance du modèle de Richards pour des écoulements saturés en certains points, et donné des éléments sur le calcul des coefficients de diffusion non constants dans les méthodes mixtes.

Une perspective est l'implémentation de la décomposition de domaine avec sa dérivation pour le problème d'écoulement puis de transport. On a vu que la dérivation en décomposition de domaine ne posait pas de problème théorique pour l'écoulement. Les calculs sur les écoulements diphasiques peuvent être élargis en deux puis trois dimensions.

Annexe A

Quelques éléments techniques sur la dérivation du programme de transport

A.1 Principe

Nous avons différentié individuellement chaque fonction ou méthode.

Le programme principal est clairement divisé entre initialisation, boucle en temps, finalisation, éventuellement appel des fonctions de calcul de ligne ou de colonne de la matrice Jacobienne. La boucle en temps est elle-même découpée selon l'algorithme du Tableau 2.3. On rappelle que l'équation de transport est résolue par une méthode de splitting. Dans la résolution de l'équation de diffusion, nous avons isolé la résolution des systèmes linéaires, comme suggéré dans la section 6.3.2.

Nous avons au départ programmé deux versions de code dérivé. La première version est dérivée uniquement en mode direct. Elle utilise au choix une dérivation avec `ADOL-C` ou à la main, ou combine ces deux méthodes, c'est-à-dire que certaines parties du programme sont différentiées avec `ADOL-C` et d'autres à la main. La deuxième version utilise uniquement la différentiation à la main pour des raisons de place mémoire. Le mode adjoint est en cours d'implémentation dans cette deuxième version.

Pour faciliter l'identification des variables d'entrée et de sortie de chaque fonction ou méthode, pour la différentiation à la main, les attributs "actifs" des classes ne servent plus qu'à l'initialisation : les variables actives seront passées en argument aux méthodes (structure moins "objet"). Autrement dit, les méthodes ne modifieront pas des attributs mais des imitations passées en argument.

Dans la première version du programme dérivé, pour chaque fonction ou méthode du programme non différentié faisant intervenir des variables actives, nous avons trois fonctions ou méthodes dans le programme différentié : une version servant à fabriquer les enregistrements avec `ADOL-C`, une version effectuant un calcul de valeurs, une version effectuant à la fois un calcul de valeurs et un calcul de variations. (Les deux dernières versions

servent uniquement pour la différentiation “à la main”.) On a ajouté une fonction d’initialisation générale : en mode automatique, elle fabrique tous les enregistrements nécessaires, dont certains sont utilisés plusieurs fois dans un seul calcul de variation (dans la boucle en temps), comme c’est mentionné dans la section 4.2.2.2 ; en version manuelle, elle calcule ce qui est indépendant des variations d’entrée (par exemple les variables “discrètes”, comme les pas de temps, ou le nombre d’itérations de convection par itération de diffusion dans le splitting). On a ajouté une fonction initialisant les vecteurs de variables : elle calcule leurs tailles, les initialise éventuellement avec des valeurs lues dans des fichiers ou calculées à l’initialisation ; on utilise exactement la même fonction pour les versions manuelle et automatique. On a ajouté une fonction qui permet d’assembler tous les morceaux pour un calcul de variations. Elle prend en particulier en argument un vecteur de types énumérés indiquant quels morceaux doivent être différentiés automatiquement et quels morceaux doivent être différentiés à la main. Nous avons fourni une version automatique pour la résolution de systèmes linéaires mais il est très déconseillé de l’utiliser (elle fonctionne à nombre d’itérations constant, en particulier pour tous les pas de temps), c’est donc pour cette partie que la version manuelle a été vérifiée en premier.

La version totalement automatique est vérifiée par comparaison avec le code non différentié de départ, sur plusieurs exemples, sous l’hypothèse qu’un code automatique qui sait calculer des valeurs correctement sait aussi calculer les dérivées. Les versions manuelles des fonctions ou méthodes peuvent être vérifiées une par une par comparaison des résultats avec ceux obtenus par la version automatique.

Dans la deuxième version de code dérivé, qui ne fonctionne pas en mode automatique, on a une quatrième version de chaque fonction ou méthode permettant la dérivation en mode inverse avec les recalculs strictement nécessaires. La méthode de l’état adjoint est utilisée en particulier pour la résolution de systèmes linéaires. Les versions de fonctions “calcul de valeur” servent pour la dérivation en mode inverse. Ce code n’est pas encore validé (voir A.3).

A.2 Structure du code de transport

Le premier travail pour dériver le programme de transport était de dresser un inventaire des variables présentes dans le code, en séparant en particulier les paramètres constants de ceux qui peuvent être modifiés. Parmi ceux qui peuvent être modifiés, on distingue les variables d’entrée du programme (appelées variables indépendantes en différentiation automatique), celles que l’on doit retourner, et les variables intermédiaires.

On ne dérive pas par rapport aux variables discrètes. Les variables intermédiaires discrètes peuvent dépendre de paramètres variables, mais ne sont pas dérivées. Des modifications de leurs valeurs correspondent à des discontinuités de la fonction à différentier.

On découpe ensuite le déroulement du calcul en étapes de taille raisonnable, et en mettant en évidence quelles variables interviennent au cours de chaque étape.

On rappelle l'équation de transport :

$$\frac{\partial\left(\Phi R_i c_i\right)}{\partial t} + \operatorname{div}\ \left(c_i\vec{u} - D_i\vec{\nabla}c_i\right) + \lambda_i \Phi R_i c_i = \sum_{j\in\{pères(i)\}} \sigma_{ij}\lambda_j \Phi R_j c_j + \Phi s_i + q_i. \tag{A.1}$$

Sous les hypothèses de l'absence de couplage entre les différents radionucléides, i.e. $\{pères(i)\} = \emptyset$, on obtient :

$$\Phi R_i \frac{\partial c_i}{\partial t} + \operatorname{div}\ \left(c_i\vec{u} - D_i\vec{\nabla}c_i\right) + \lambda_i \Phi R_i (1-\sigma_i) c_i = \Phi s_i + q_i \tag{A.2}$$

Notre programme de transport par splitting résout l'équation suivante :

$$\omega\frac{\partial c}{\partial t} + \operatorname{div}\ \left(c\vec{u} - D\vec{\nabla}c\right) + \omega\lambda c = 0. \tag{A.3}$$

On effectue donc d'abord les changements de variable :

$$\omega = R_i\Phi\ ; \tag{A.4}$$

$$\lambda = (1-\sigma_i)\,\lambda_i. \tag{A.5}$$

Le terme source est à ajouter. Les coefficients de diffusion sont à calculer après la résolution de l'équation d'écoulement :

$$D = d_e\mathbf{I} + |\vec{u}|\left[\alpha_l E + \alpha_t\left(\mathbf{I} - E\right)\right] \quad \text{où} \quad E_{i,j} = \frac{u_i u_j}{|\vec{u}|^2}. \tag{A.6}$$

A.2.1 Paramètres du problème.

1. λ constant en temps et en espace
2. ω constant par morceaux en espace, constant en temps
3. vitesse de Darcy `FluxDarcy`
4. tenseur de diffusion par cellule (construit à partir de la vitesse de Darcy ou, pour des essais simplement, constant par morceaux)

On renvoie à la Figure 2.3.

A.2.2 Paramètres discrets.

1. Pas de temps de diffusion
2. `dimXdof` : nombre de cellules du maillage
3. `dimWdof` : nombre de faces du maillage
4. Maillage

A.2.3 Variables intermédiaires.

1. vecteur `D_evol` de taille `dimXdof` : termes diagonaux de la matrice de convection
2. matrice `jac` de taille `dimWdof`×`dimWdof`, stockée dans un vecteur de taille nzA ≈ 6×`dimWdof`
3. vecteur `sourceVector` de taille `dimWdof` : second membre du système linéaire de l'équation de diffusion
4. vecteur `Flux_Diff` de taille `dimWdof` : inconnue du système linéaire
5. vecteur `conc0` de taille `dimXdof` : champ de concentrations à t = $n\Delta t$
6. vecteur `conc1` de taille `dimXdof` : champ de concentrations à t = $(n+1)\Delta t$

A.2.4 Variables intermédiaires discrètes.

1. liste des sous-domaines pris en compte dans le calcul
2. CFL locales
3. nombre de divisions du pas de temps de diffusion et pas de temps de convection
4. vecteur `iamont` : pour chaque face F_j du maillage, indice de la cellule amont, définie par le sens du flux de $\vec{u}$ à travers F_j (la résolution de l'équation de convection utilise un schéma décentré amont).

A.2.5 Déroulement du calcul.

1. Initialisation
   (a) Construction de `D_evol` à partir de tous les paramètres du problème sauf le tenseur de diffusion par cellule
   (b) Construction de `jac` à partir de `D_evol` et du tenseur de diffusion par cellule
2. Boucle en temps
   (a) Initialisation de la boucle en temps (1 fois) : calcul des variables intermédiaires discrètes
   (b) Les pas de temps de convection : calcul de `conc1` à partir de `conc1`, `FluxDarcy`, `phi_coef` et des variables discrètes
   (c) Le pas de temps de diffusion
      i. calcul de `conc0` et `sourceVector` à partir de λ, `conc1` et des variables discrètes
      ii. calcul de `Flux_Diff` à partir de `jac` et `sourceVector`
      iii. calcul de `conc1` à partir de λ, `D_evol`, `Flux_Diff` et `conc0`

A.2.6 Découpage du code : choix des enregistrements.

Lors de la lecture d'un enregistrement avec `ADOL-C`, toutes les entrées de la fonction représentée par l'enregistrement doivent être regroupées dans un seul tableau. Il en est de même pour les sorties. On liste ici les différents tableaux utilisés, avec leurs tailles. Chaque mise à jour de la variable `tag` annonce un nouvel enregistrement dont on précise les tableaux d'entrée et de sortie (`X_...` et `Y_...`). Certains tableaux peuvent se recouvrir, quand les mêmes variables sont utilisées par différents enregistrements. Des données doivent parfois être dupliquées, pour des raisons de contiguïté.

"14" est le nombre de sous-domaines du maillage, qui est utilisé pour la définition des paramètres constantes par morceaux, comme le tenseur de conductivité ou éventuellement le tenseur de diffusion D_i. Dans le cas général, le tenseur est relié au champ de vitesse de l'eau par la relation (A.6) donc il est simplement considéré constant par cellule. Il sera constant par zone si les coefficients α_l et α_t de l'équation (A.6) sont négligeables devant d_e, avec d_e constant par zone, dans toutes les zones. La variable `diff_tensor_per_sbd` est utilisée uniquement lorsque l'on fait l'hypothèse d'un tenseur de diffusion constant par morceaux. La variable `diff_coef` est utilisée uniquement lorsque l'on fait l'hypothèse d'un tenseur de diffusion constant par morceaux et diagonal. "autres diffusions" désigne le champ de tenseurs de diffusion calculé à partir du flux de Darcy, il est calculé avant la phase d'initialisation. En mode non automatique, il est stocké et lu dans un fichier ; l'argument "autres diffusions" est alors remplacé par un nom de fichier, c'est-à-dire une chaîne de caractères.

1. Pour l'initialisation (lecture des données, calcul de `D_evol` et de la matrice de diffusion) :
   `X_init` = [λ ; (`diff_coef`,`phi_coef`) ; `diff_tensor_per_sbd` ; `conc_init` ; FluxDarcy] ;
   λ : 1 ; (`diff_coef`,`phi_coef`) : 14×2 ; `diff_tensor_per_sbd` : 14×9 ; `conc_init` : `dimXdof` ; FluxDarcy : `dimWdof`
   `Y_init` = [`conc1` ; `D_evol`] (taille $2 \times$ `dimXdof`) ;
   - `tag = 100`
     Lecture : `X_init`$\longrightarrow$`Y_init` mais avec les tailles (`size_X_init`)$\longrightarrow$(`dimXdof`) (retourne juste `conc1`).
   - `tag = 101`
     Calcul de `D_evol` : `X_init`$\longrightarrow$`Y_init+dimXdof` avec les tailles ($1+2\times14$)$\longrightarrow$(`dimXdof`) (calcul de `D_evol` à partir de λ et `phi_coef`).
   - `tag = 11`
     `X_matrix` = [`D_evol` ; `diff_coef` ; `diff_tensor_per_sbd` ; autres diffusions] (taille `dimXdof` + 14 + 9 $\times$ 14 + 9 $\times$ `dimXdof`) ;
     `Y_matrix` = [`jac`] (taille nzA) ; stocké dans `X_31`.
2. Pour les pas de temps de convection :
   - `tag = 20`
     `X_conv` = [`conc1` ; `FluxDarcy` ; `phi_coef`] (taille `dimXdof` + `dimWdof` + 14) ;
     `Y_conv` = [`conc1`] (taille `dimXdof`) ; stocké dans `Y_loop` ; `Y_loop` = `X_30` ;
3. Pour les pas de temps de diffusion :

– `tag = 30`
  `X_30` = [`conc1` ; λ] (taille `dimXdof`+1) ;
  `Y_30` = [`conc0` ; `sourceVector`] (taille `dimXdof` + `dimWdof`) ;
– `tag = 31`
  `X_31` = [`jac` ; `sourceVector`] (taille nzA + `dimWdof`) ;
  `Y_31` = [`Flux_Diff`] (taille `dimWdof`) ; `Y_31` = `X_32 + 2 dimXdof + 1` ;
– `tag = 32`
  `X_32` = [`conc0` ; λ ; `D_evol` ; `Flux_Diff`] (taille 2×`dimXdof` + `dimWdof` + 1) ;
  `Y_32` = [`conc1`] (taille `dimXdof`) ; stocké dans `Y_loop`.

Bilan du pas de temps de diffusion :

[`conc1` ; `jac` ; λ ; `D_evol`] (taille nzA + 2 × `dimXdof` + 1) $\longrightarrow$ [`conc1`] (taille `dimXdof`).

A.3 Enchaînement des pas de temps en mode adjoint

On suppose que les valeurs du champ de concentration au début et à la fin de chaque pas de temps ont déjà été calculées, et stockées dans le tableau `all_X_conv`. Le vecteur `(vector<double>) grad` contient les gradients des indicateurs de sûreté par rapport au champ de concentration à l'instant final.

Examinons une implémentation :

```
 ...

// reading of the output gradients:

std::copy(grad.begin(),grad.end(),grad_conc);

// backward computation:

for(int locnstep=pb->nstep_diff; locnstep>=1; locnstep--){
  double* X_conv_0 = all_X_conv[locnstep-1];
  double* X_conv_1 = all_X_conv[locnstep];
  backwardTimeStep( pb, grad_conc, size_X_conv, X_conv_0, X_conv_1,
                    X_30, grad_X_30,
                    size_X_31, X_31, grad_X_31,
                    X_32, grad_X_32,
                    Y_30, grad_Y_30, size_Y_31, Y_31, grad_Y_31,
                    size_Y_loop, Y_loop, grad_Y_loop,
                    nzA);
}
std::copy(grad_conc, grad_conc+size_X_conv, grad_X_conv);
```

Gradient par rapport au tenseur de diffusion, si on l'a choisi constant par morceaux :

```
std::copy(grad_X_32 + pb->dimXdof + 1,
          grad_X_32 + pb->dimXdof + 1 + pb->dimXdof,
          grad_Y_init + pb->dimXdof);
```

Gradient par rapport à λ :

```
std::copy(grad_X_conv, grad_X_conv + pb->dimXdof, grad_Y_init);

grad_X_init[0] = grad_X_30[pb->dimXdof];

//phi_coef:
for(int i=0; i<14; i++)
  grad_X_init[2+2*i] = grad_X_conv[pb->dimXdof+pb->dimWdof+i];
//FluxDarcy:
std::copy(grad_X_conv + pb->dimXdof,
          grad_X_conv + pb->dimXdof+pb->dimWdof,
          grad_X_init + 1 + 2*14 + 9*14 + pb->dimXdof);
//conc1:
std::copy(grad_X_conv, grad_X_conv + pb->dimXdof, grad_Y_init);

double* grad_Y_matrix = grad_X_31;
```

Dans le cas où le tenseur de diffusion dépend de la vitesse de Darcy, il a été stocké dans le fichier `fileName`. On écrit les gradients correspondant dans le fichier `grad_fileName` :

```
backwardComputeMatrix(pb, X_matrix, grad_X_matrix,
                      grad_Y_matrix, fileName, grad_fileName);

double* grad_Y_D_evol = grad_Y_init + pb->dimXdof;
double* grad_X_D_evol = grad_X_init;
double* grad_X_diffusion = grad_X_init + (size_X_init - pb->dimWdof);

std::copy(grad_X_matrix + pb->dimXdof + 14,
          grad_X_matrix + pb->dimXdof + 14 + 9*14,
          grad_X_init + 1 + 28);
backwardComputeD_evol(pb, X_D_evol, grad_X_D_evol, Y_D_evol, grad_Y_D_evol);
if(!hom)
  backwardComputeDiffusionTensor(pb, X_diffusion, grad_X_diffusion,
                                     fileName, grad_fileName);
backwardReadProblem( pb, X_init, grad_X_init, Y_init, grad_Y_init, true );
```

A.3.1 Recalculs au cours d'un pas de temps

On détaille ici la fonction `backwardTimeStep` : on doit recalculer les valeurs intermédiaires du champ de concentration au différentes étapes du pas de temps (après chaque itération

de convection). Les trois étapes

`valueDiffusionStep_begin`
`valueDiffusionStep_middle`
`valueDiffusionStep_end`

correspondent à l'isolation de la résolution du système linéaire, selon un principe similaire à ce qui est décrit pour le problème d'écoulement dans le Tableau 6.3.
La méthode `valueDiffusionStep_begin` calcule le second membre de l'équation (2.89), `valueDiffusionStep_middle` résout l'équation (2.89)
et `valueDiffusionStep_end` calcule C suivant l'équation (2.84).

Les 3 étapes

`backwardDiffusionStep_end`
`backwardDiffusionStep_middle`
`backwardDiffusionStep_begin`

calculent les adjoints correspondant.

La méthode `backwardDiffusionStep_middle` utilise le résultat suivant.

Soit $f : (A, b) \mapsto g(X) \in \mathbb{R}$ tel que $AX = b$ où A est une matrice $n \times n$ inversible et b un vecteur de taille n. On peut utiliser la méthode de l'état adjoint pour dériver f selon

$$\begin{aligned}
&AX = b\\
&A^T\lambda = -\vec{\nabla}_X g\\
&\vec{\nabla}_b f = -\lambda^T\\
&\text{Pour } i, j = 1, ..., n\\
&\quad \vec{\nabla}_{A_{ij}} f = X_j \lambda_i.
\end{aligned}$$

```
// recalculation with storage of all the intermediary values:
// ----------------------------------------------------------
double X_conv[(int)pb->current_nstep_conv+1][(int)size_X_conv];
std::copy(X_conv_0, X_conv_0+size_X_conv, X_conv[0]);
for(int ms=1; ms<=pb->current_nstep_conv; ms++){
  double* X_conv_temp = X_conv[ms-1];
  valueConvectionStep( pb, X_conv_temp, Y_loop);
  std::copy(Y_loop, Y_loop+pb->dimXdof, X_conv[ms]);
}
valueDiffusionStep_begin( pb, X_30, Y_30 );
std::copy(Y_30, Y_30+pb->dimXdof, X_32);
std::copy(Y_30+pb->dimXdof, Y_30+pb->dimXdof+pb->dimWdof, X_31+nzA);
valueDiffusionStep_middle( pb, X_31, Y_31, nzA);
valueDiffusionStep_end( pb, X_32, Y_loop);
std::copy(Y_loop, Y_loop+pb->dimXdof, X_conv_1);

// backward computation:
// ---------------------
```

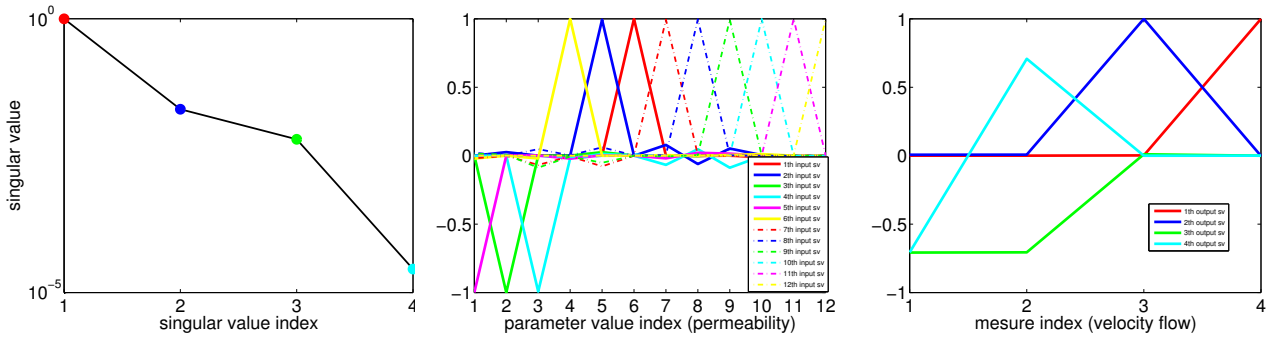
```
std::copy(grad_conc, grad_conc+pb->dimXdof, grad_Y_loop);
backwardDiffusionStep_end( pb, X_32, grad_X_32, Y_loop, grad_Y_loop);
backwardDiffusionStep_middle( pb, size_X_31, X_31, grad_X_31,
                              size_Y_31, Y_31, grad_Y_31, nzA, true);
std::copy(grad_X_31+nzA,
          grad_X_31+nzA+pb->dimWdof,
          grad_Y_30+pb->dimXdof);
std::copy(grad_X_32, grad_X_32+pb->dimXdof, grad_Y_30);
backwardDiffusionStep_begin(pb, X_30, grad_X_30, Y_30, grad_Y_30);
for(int ms=pb->current_nstep_conv; ms>=1; ms--){
  std::copy(grad_conc, grad_conc+size_Y_loop, grad_Y_loop);
  std::copy(X_conv[ms], X_conv[ms]+size_Y_loop, Y_loop);
  double* X_conv_temp = X_conv[ms-1];
  backwardConvectionStep( pb, X_conv_temp, grad_conc, Y_loop, grad_Y_loop);
}
```

Annexe B

Résultats exhaustifs pour l'analyse déterministe du problème d'écoulement

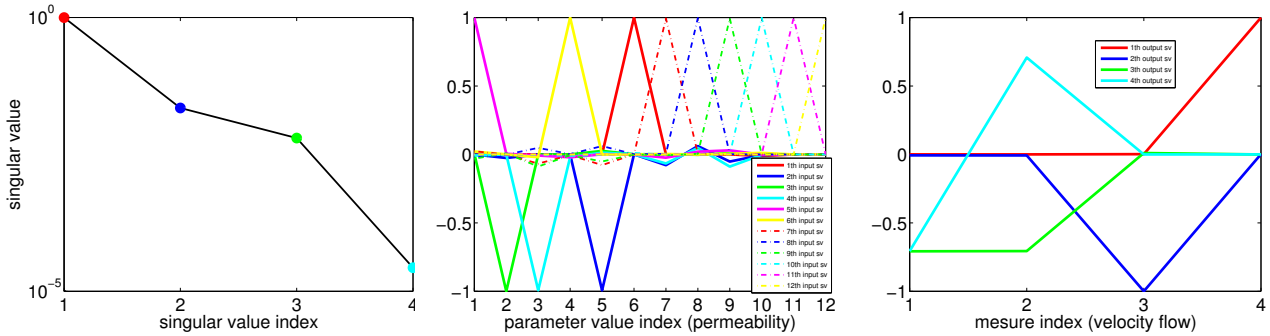
B.1 Tests avec tous les paramètres sauf un à leur valeur la plus probable

Les données utilisées pour les résultats présentés dans les Figures B.1 à B.12 sont données dans les Tableaux 7.1 et 7.2.



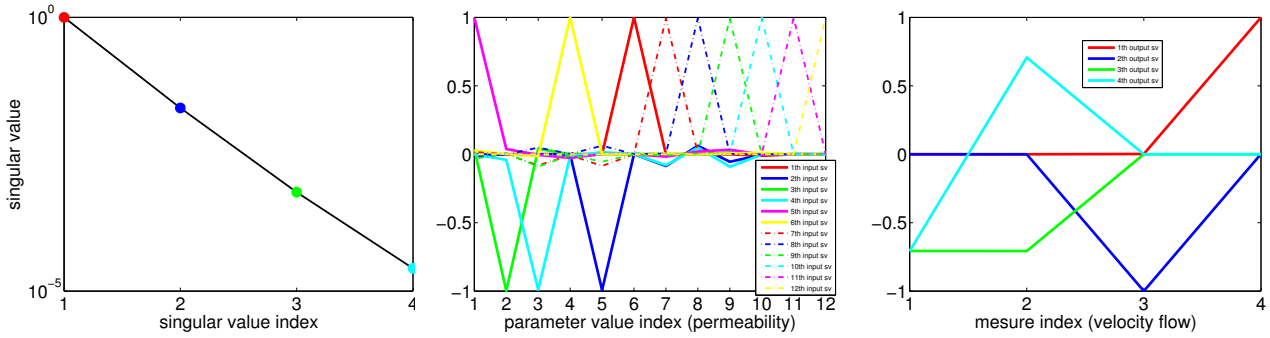
(a) [5.53 10^{-3} m^3/s] (b) (c)

FIG. B.1 – Résultats de SVD avec la plus petite valeur possible pour $\dfrac{k_h}{k_v}$.



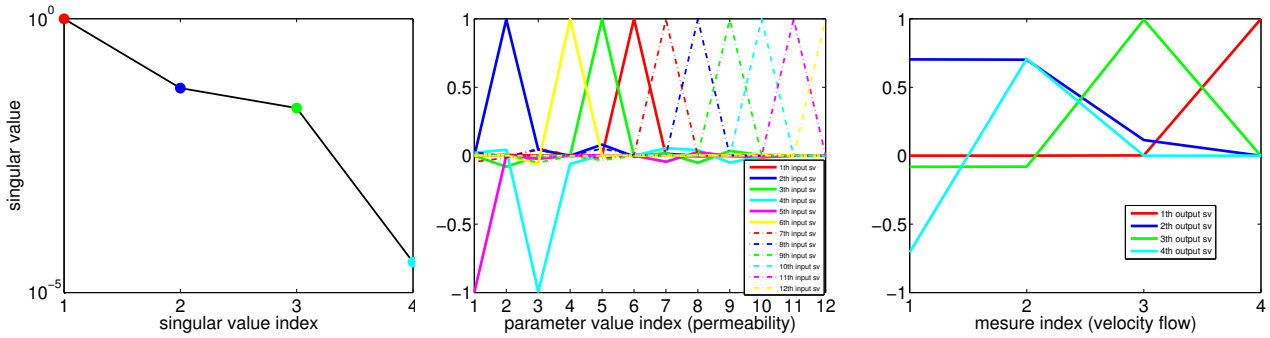
(a) [5.53 10^{-3} m^3/s] (b) (c)

FIG. B.2 – Résultats de SVD avec la plus grande valeur possible pour $\dfrac{k_h}{k_v}$.



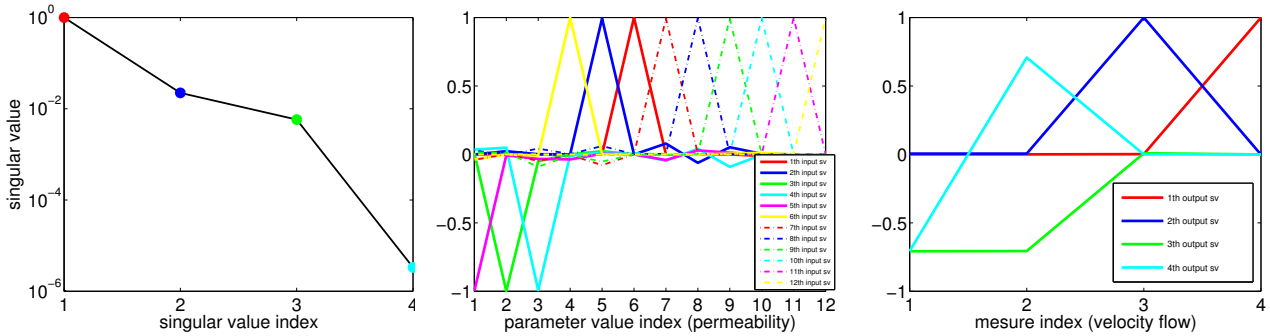
(a) [5.53 10^{-3} m^3/s] (b) (c)

FIG. B.3 – Résultats de SVD avec la plus petite valeur possible pour k_v.



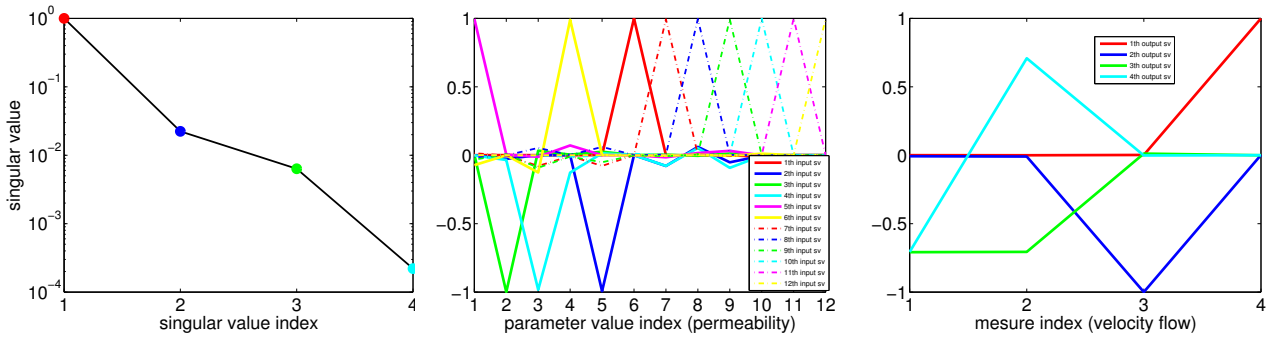
(a) [5.55 10^{-3} m^3/s] (b) (c)

FIG. B.4 – Résultats de SVD avec la plus grande valeur possible pour k_v.



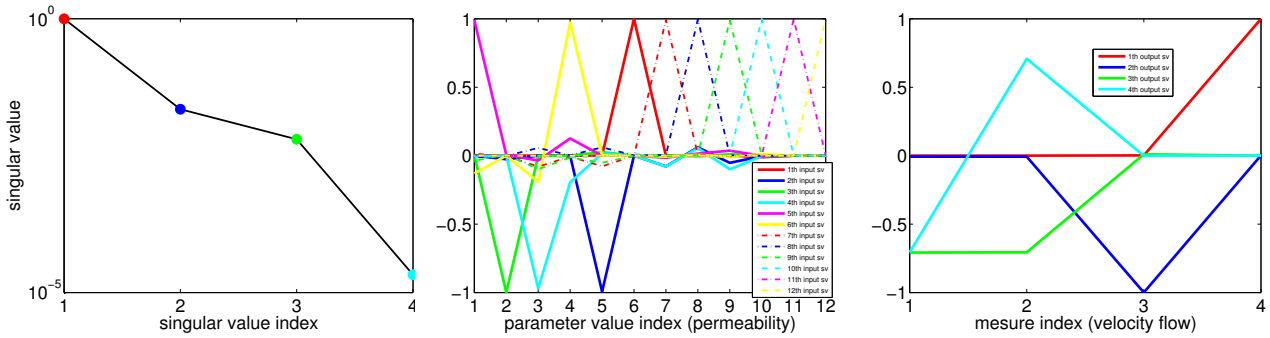
(a) [5.53 10^{-3} m^3/s] (b) (c)

FIG. B.5 – Résultats de SVD avec la plus petite valeur possible pour $\frac{k_3}{k_v}$.



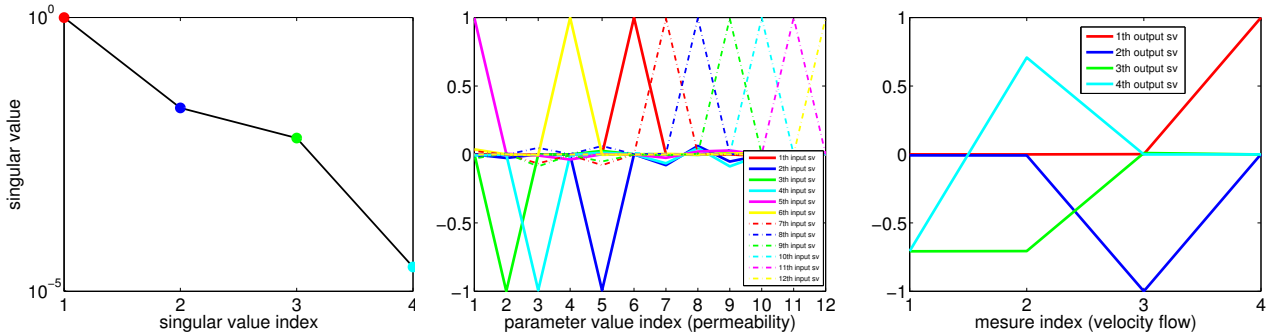
(a) [5.53 10^{-3} m^3/s] (b) (c)

FIG. B.6 – Résultats de SVD avec la plus grande valeur possible pour $\frac{k_3}{k_v}$.



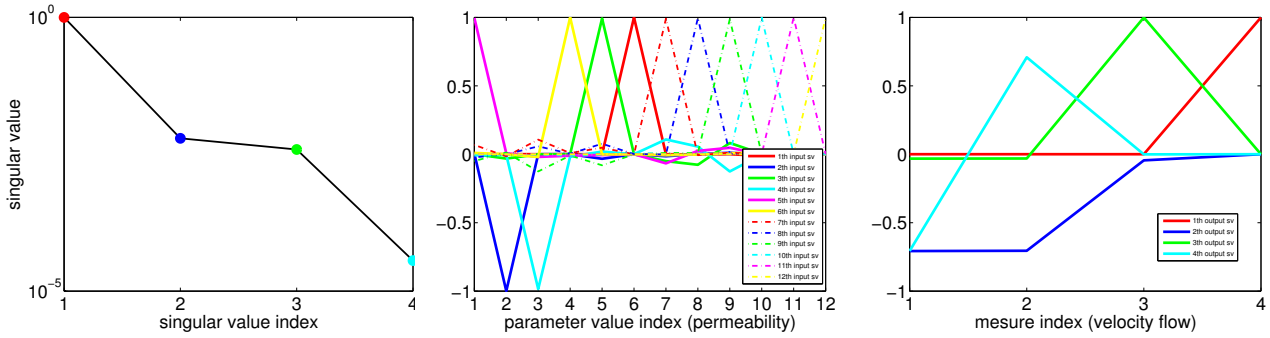
(a) [5.53 10^{-3} m^3/s] (b) (c)

FIG. B.7 – Résultats de SVD avec la plus petite valeur possible pour $\dfrac{k_4}{k_3}$.



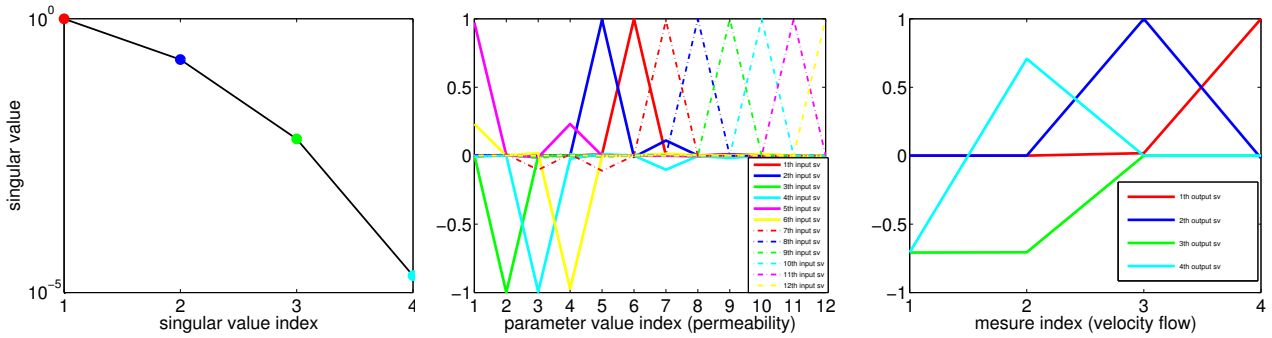
(a) [5.53 10^{-3} m^3/s] (b) (c)

FIG. B.8 – Résultats de SVD avec la plus grande valeur possible pour $\dfrac{k_4}{k_3}$.



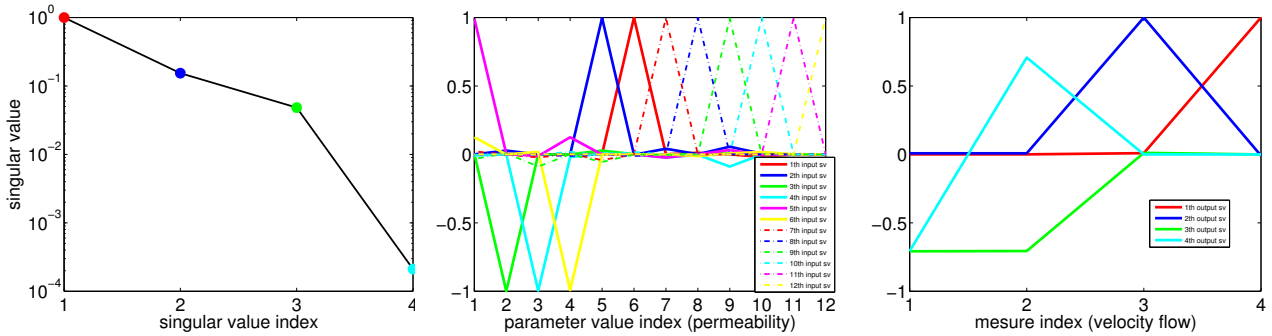
(a) [5.53 10^{-3} m^3/s] (b) (c)

FIG. B.9 – Résultats de SVD avec la plus petite valeur possible pour k_5.



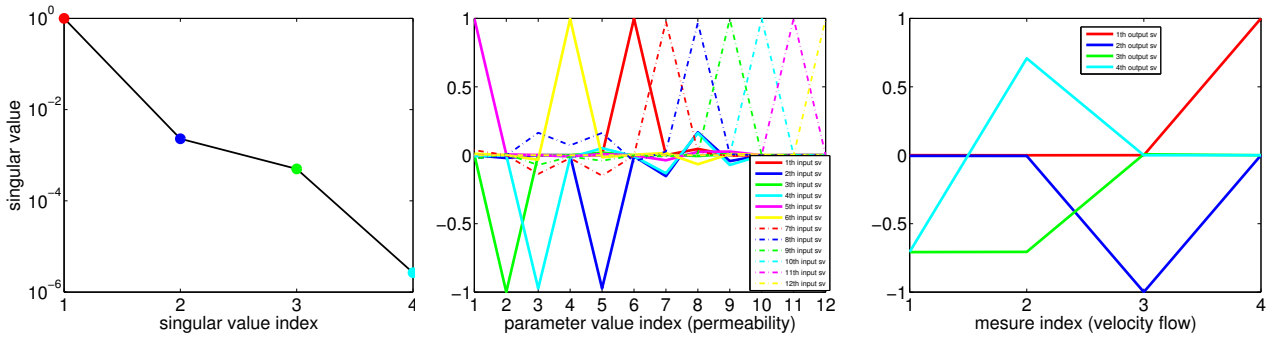
(a) [5.56 10^{-3} m^3/s] (b) (c)

FIG. B.10 – Résultats de SVD avec la plus grande valeur possible pour k_5.



(a) [7.21 10^{-4} m^3/s] (b) (c)

FIG. B.11 – Résultats de SVD avec la plus petite valeur possible pour $\frac{k_6}{k_5}$.



(a) [6.96 10^{-2} m^3/s] (b) (c)

FIG. B.12 – Résultats de SVD avec la plus grande valeur possible pour $\frac{k_6}{k_5}$.

B.2 Exploration des sommets de l'ensemble des valeurs possibles pour les paramètres d'entrée

Les valeurs de paramètres utilisées pour les résultats présentés dans les figures suivantes sont données dans les Tableaux 7.5 et7.2.

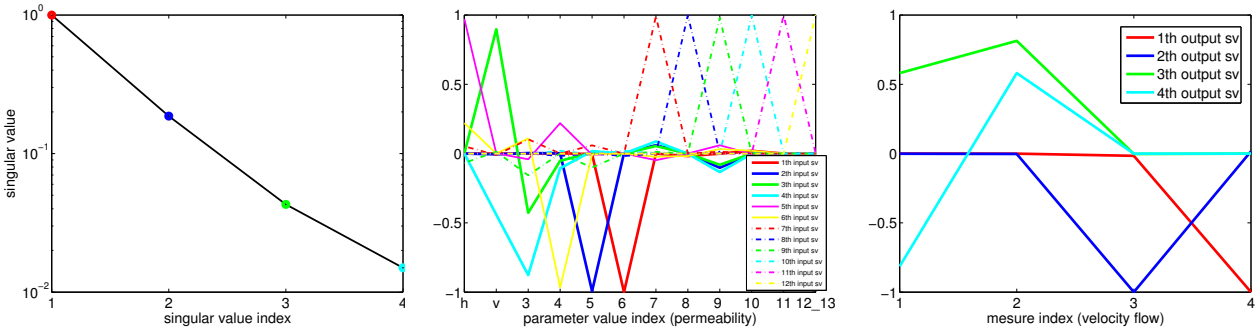
Ces résultats ont été triés suivant les résultats de la SVD, i.e. suivant la composition des vecteurs singuliers et leur hiérarchie. Ils sont comparés avec les résultats obtenus pour les valeurs de paramètres les plus vraisemblables, donnés en Figure 7.4. On remarque que ce tri correspond à un tri sur les valeurs des paramètres d'entrée (voir la section 7.1.7.3).

B.2.1 Résultats où les deux derniers vecteurs singuliers de l'espace d'entrée sont légèrement déformés

Avec la plus grande valeur possible pour $\frac{k_3}{k_v}$ (10^4). Figures B.13 à B.20 : $\ln(k_v) - \alpha \ln(k_3) \rightsquigarrow \Phi_1 + \Phi_2$ and $\ln(k_3) + \alpha \ln(k_v) \rightsquigarrow \Phi_1 - \Phi_2$ avec $\alpha \approx \frac{1}{2}$, les 2ème et 3ème vecteurs singuliers de l'espace d'entrée sont légèrement déformés par rapport à ceux de la première étude locale.

Avec la plus petite valeur possible pour $\frac{k_3}{k_v}$ (10^0). Figures B.21 et B.22 : $\ln(k_v) + \alpha \ln(k_3) \rightsquigarrow \Phi_1 + \Phi_2$ et $\ln(k_3) - \beta(\ln(k_v) + \ln(k_h)) \rightsquigarrow \Phi_1 - \Phi_2$, $\alpha \approx \frac{1}{2}$ (les vecteurs singuliers dans l'espace d'entrée sont légèrement déformés, comparés à ceux de la première étude locale). Figure B.23 : $\ln(k_v) + \alpha \ln(k_3) \rightsquigarrow \Phi_1 + \Phi_2$; $(k_3, k_4) \rightsquigarrow \Phi_1 - \Phi_2$; rôle non négligeable de $k_4 \leftarrow$ petites valeurs à la fois pour $\frac{k_4}{k_3}$ et pour $\frac{k_3}{k_v}$ (voir 7.1.7.3). Figure B.24 : résultats proches de ceux de la Figure B.23, la différence avec ceux de la première étude locale est encore plus forte ($\Phi_1 - \Phi_2$ est locally influencé principalement par k_4). k_h a une influence non nulle.

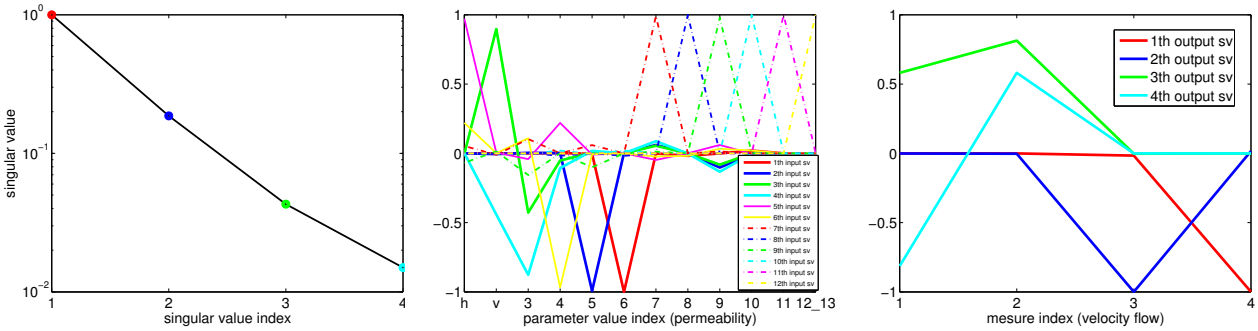
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-14}, k_v = 10^{-14}$	10^{-10}	10^{-9}	10^{-9}	10^{-8}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [8.93 10^{-5} m^3/s] (b) (c)

FIG. B.13 – Résultats d'analyse de sensibilité avec données.

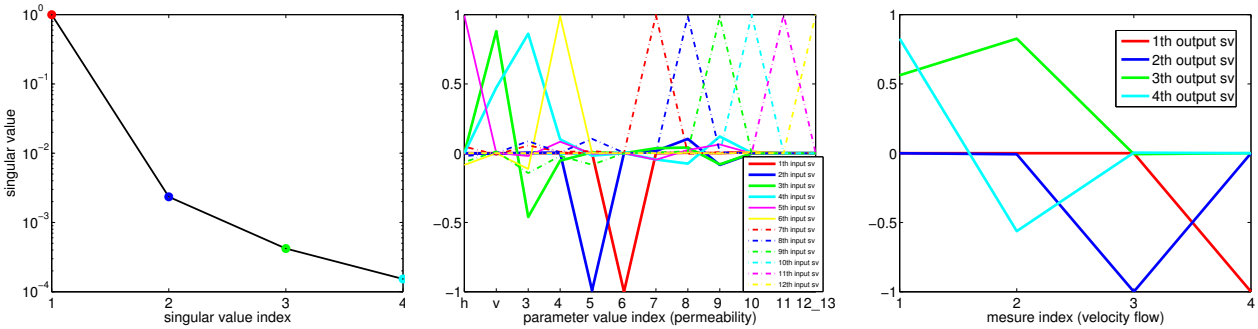
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-12}, k_v = 10^{-14}$	10^{-10}	10^{-9}	10^{-9}	10^{-8}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [8.93 10^{-5} m^3/s] (b) (c)

FIG. B.14 – Résultats d'analyse de sensibilité avec données.

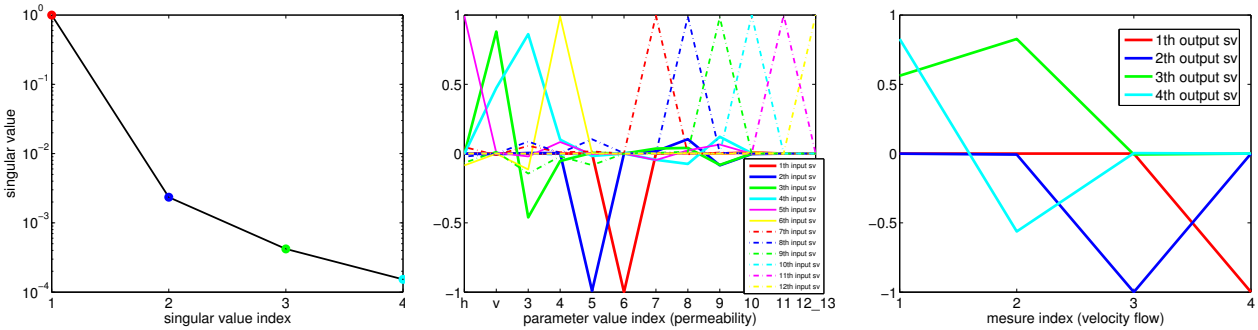
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-14}, k_v = 10^{-14}$	10^{-10}	10^{-9}	10^{-9}	10^{-6}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [9.27 10^{-3} m^3/s] (b) (c)

FIG. B.15 – Résultats d'analyse de sensibilité avec données.

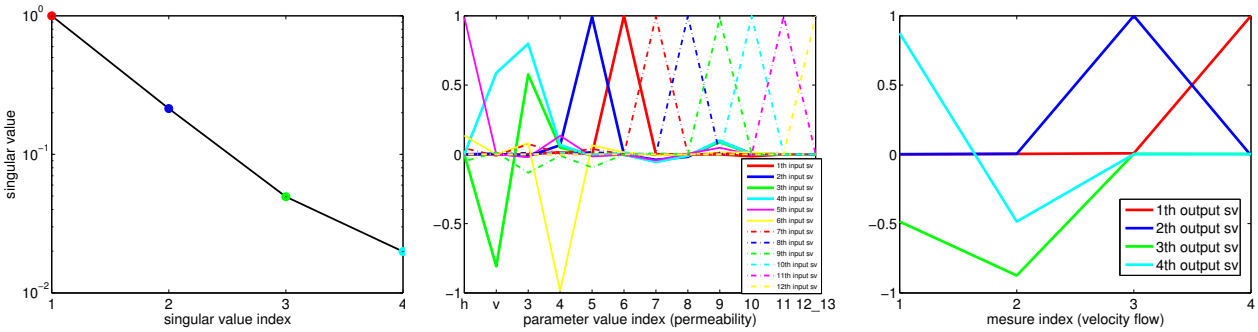
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-12}, k_v = 10^{-14}$	10^{-10}	10^{-9}	10^{-9}	10^{-6}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [9.27 10^{-3} m^3/s] (b) (c)

FIG. B.16 – Résultats d'analyse de sensibilité avec données.

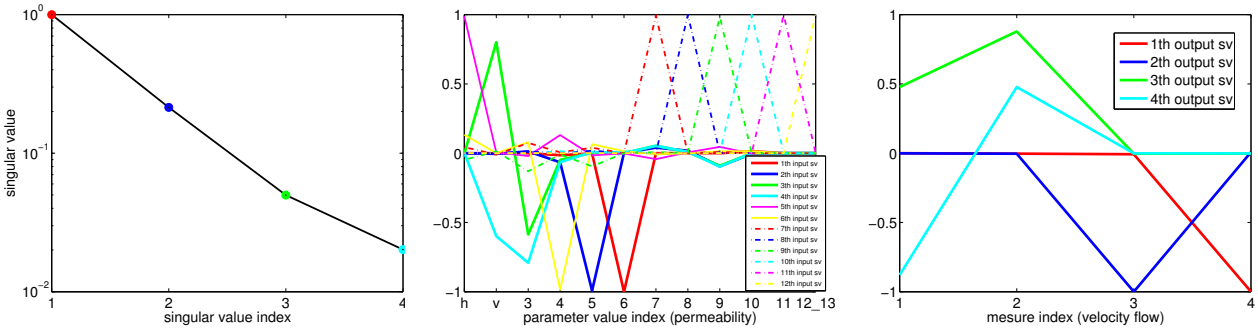
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-14}, k_v = 10^{-14}$	10^{-10}	10^{-7}	10^{-9}	10^{-8}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [8.37 10^{-5} m^3/s] (b) (c)

FIG. B.17 – Résultats d'analyse de sensibilité avec données.

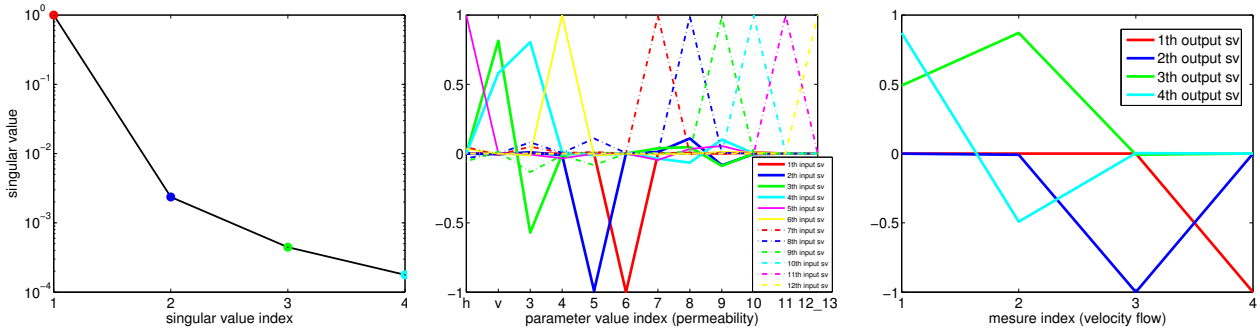
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-12}, k_v = 10^{-14}$	10^{-10}	10^{-7}	10^{-9}	10^{-8}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [8.37 10^{-5} m^3/s] (b) (c)

FIG. B.18 – Résultats d'analyse de sensibilité avec données.

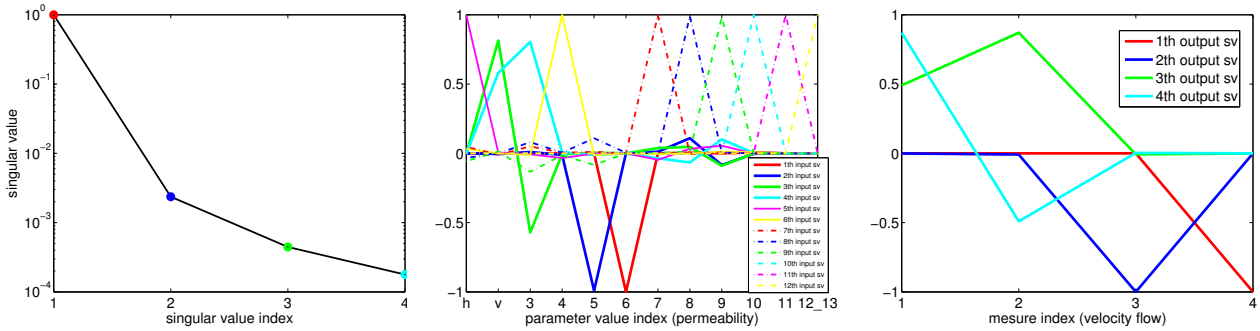
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-14}, k_v = 10^{-14}$	10^{-10}	10^{-7}	10^{-9}	10^{-6}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [9.27 10^{-3} m^3/s] (b) (c)

FIG. B.19 – Résultats d'analyse de sensibilité avec données.

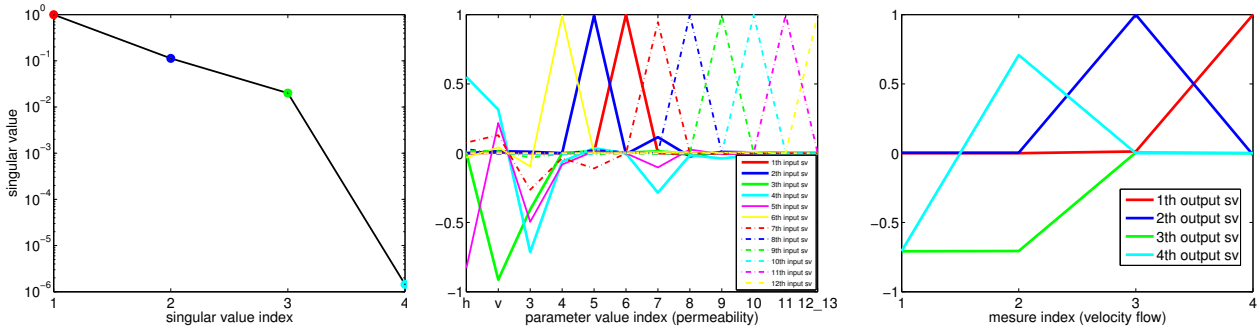
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-12}, k_v = 10^{-14}$	10^{-10}	10^{-7}	10^{-9}	10^{-6}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [9.27 10^{-3} m^3/s] (b) (c)

FIG. B.20 – Résultats d'analyse de sensibilité avec données.

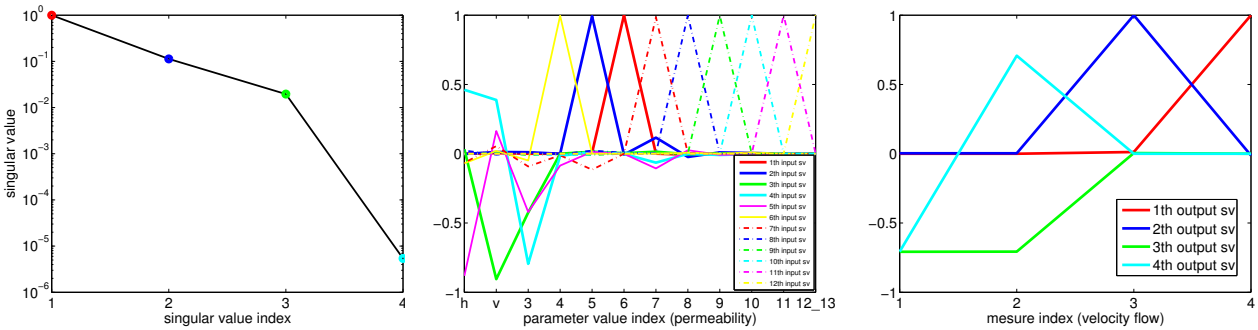
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-12}, k_v = 10^{-12}$	10^{-12}	10^{-9}	10^{-7}	10^{-6}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [9.31 10^{-3} m^3/s] (b) (c)

FIG. B.21 – Résultats d'analyse de sensibilité avec données.

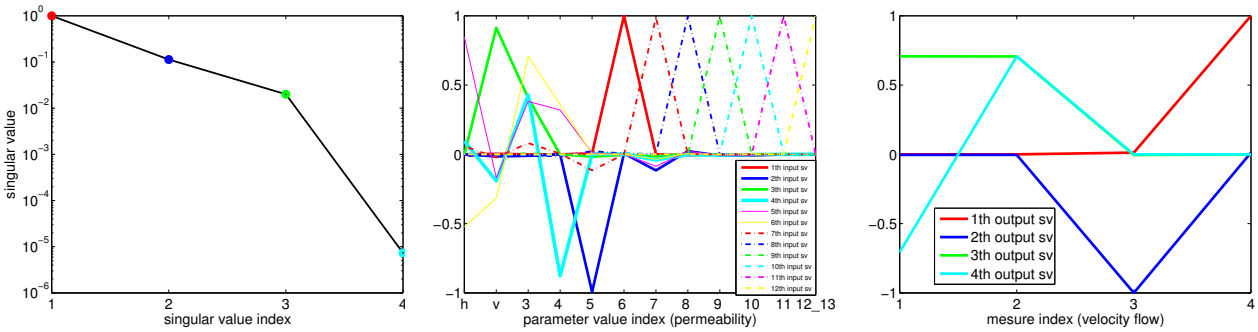
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-10}, k_v = 10^{-12}$	10^{-12}	10^{-9}	10^{-7}	10^{-6}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [9.31 10^{-3} m^3/s] (b) (c)

FIG. B.22 – Résultats d'analyse de sensibilité avec données.

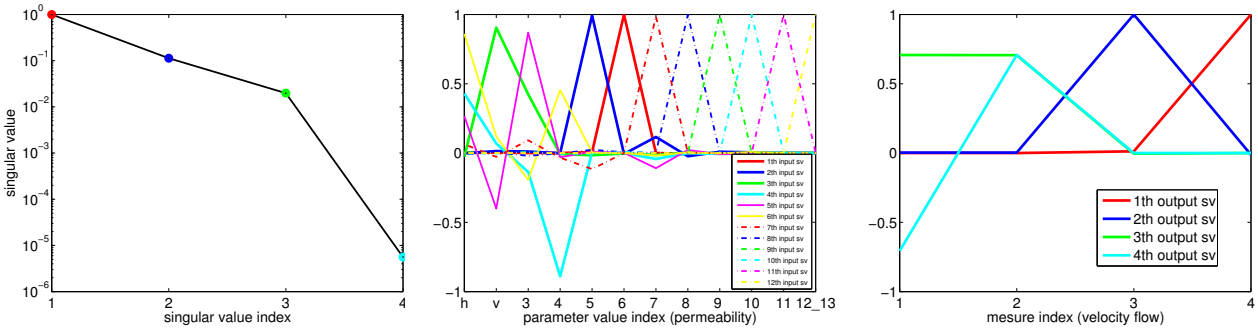
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-12}, k_v = 10^{-12}$	10^{-12}	10^{-11}	10^{-7}	10^{-6}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [2.96 10^{-4} m^3/s] (b) (c)

FIG. B.23 – Résultats d'analyse de sensibilité avec données.

zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-10}, k_v = 10^{-12}$	10^{-12}	10^{-11}	10^{-7}	10^{-6}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [2.96 10^{-4} m^3/s] (b) (c)

FIG. B.24 – Résultats d'analyse de sensibilité avec données.

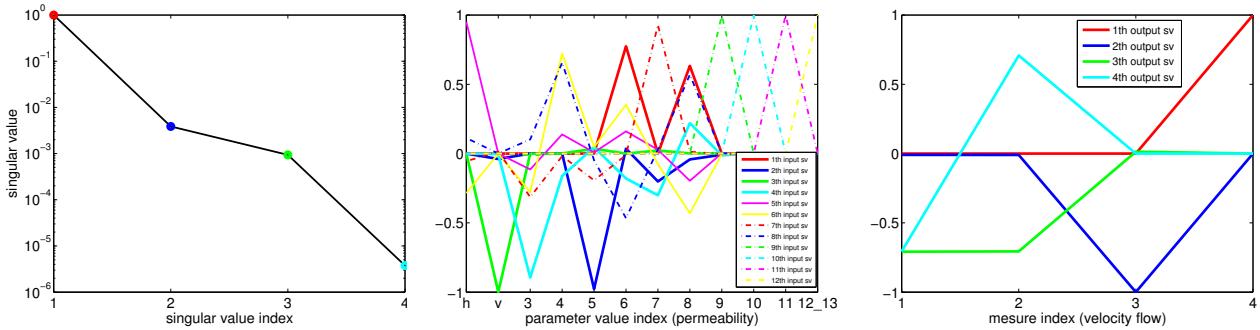
B.2.2 Résultats où le premier vecteur singulier de l'espace d'entrée comprend une influence de k_8

Avec la plus grande valeur possible pour k_6 (10^{-4}). Figure B.25 et Figure B.32 : $(\ln(k_6), \ln(k_8)) \rightsquigarrow \Phi_4$.

Avec la plus grande valeur possible pour k_6 (10^{-4}) et la plus grande valeur possible pour $\frac{k_3}{k_v}$ (10^4). Figures B.33 à B.36 : les deux derniers vecteurs singuliers de l'espace d'entrée sont aussi modifiés.

Avec la plus grande valeur possible pour k_6 (10^{-4}) et la plus petite valeur possible pour $\frac{k_3}{k_v}$ (10^0). Sur les Figures B.37 à B.38, les deux derniers vecteurs singuliers de l'espace d'entrée sont déformés. Figure B.39 et Figure B.40 : voir les commentaires au sujet des Figures B.23 et B.24.

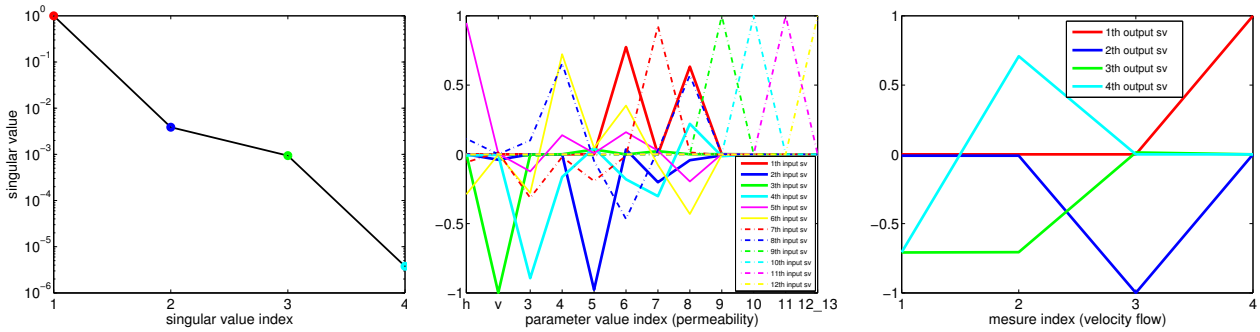
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-12}, k_v = 10^{-12}$	10^{-10}	10^{-9}	10^{-7}	10^{-4}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [3.78 10^{-1} m^3/s] (b) (c)

FIG. B.25 – Résultats d'analyse de sensibilité avec données.

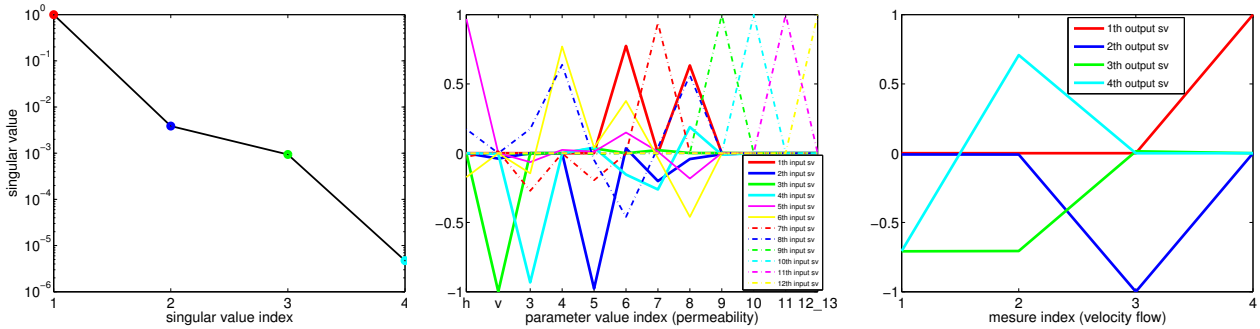
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-10}, k_v = 10^{-12}$	10^{-10}	10^{-9}	10^{-7}	10^{-4}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [3.78 10^{-1} m^3/s] (b) (c)

FIG. B.26 – Résultats d'analyse de sensibilité avec données.

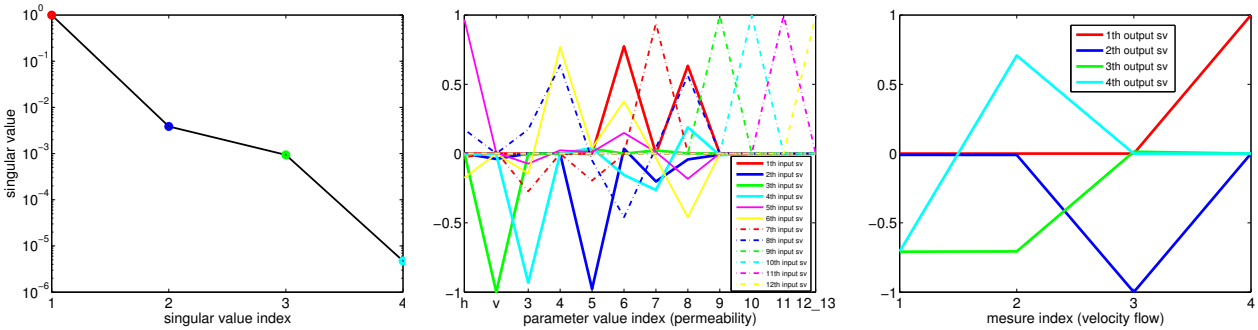
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-12}, k_v = 10^{-12}$	10^{-10}	10^{-7}	10^{-7}	10^{-4}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [3.78 10^{-1} m^3/s] (b) (c)

FIG. B.27 – Résultats d'analyse de sensibilité avec données.

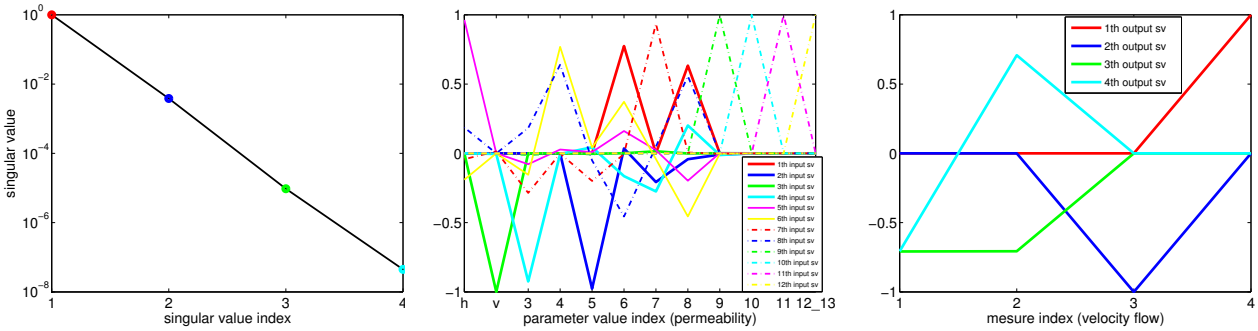
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-10}, k_v = 10^{-12}$	10^{-10}	10^{-7}	10^{-7}	10^{-4}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [3.78 10^{-1} m^3/s] (b) (c)

FIG. B.28 – Résultats d'analyse de sensibilité avec données.

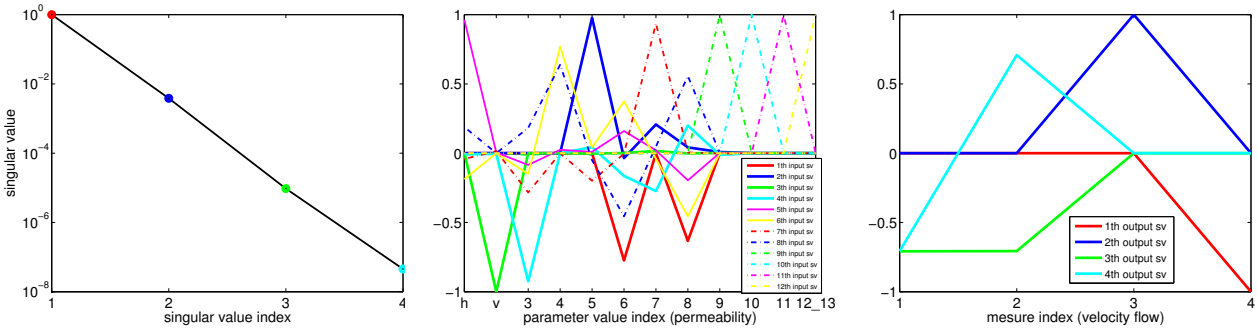
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-14}, k_v = 10^{-14}$	10^{-12}	10^{-9}	10^{-7}	10^{-4}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [3.78 10^{-1} m^3/s] (b) (c)

FIG. B.29 – Résultats d'analyse de sensibilité avec données.

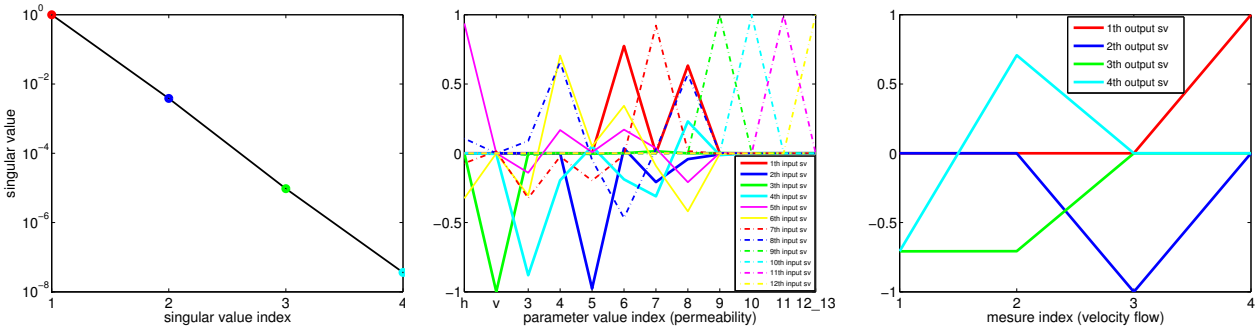
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-12}, k_v = 10^{-14}$	10^{-12}	10^{-9}	10^{-7}	10^{-4}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [3.78 10^{-1} m^3/s] (b) (c)

FIG. B.30 – Résultats d'analyse de sensibilité avec données.

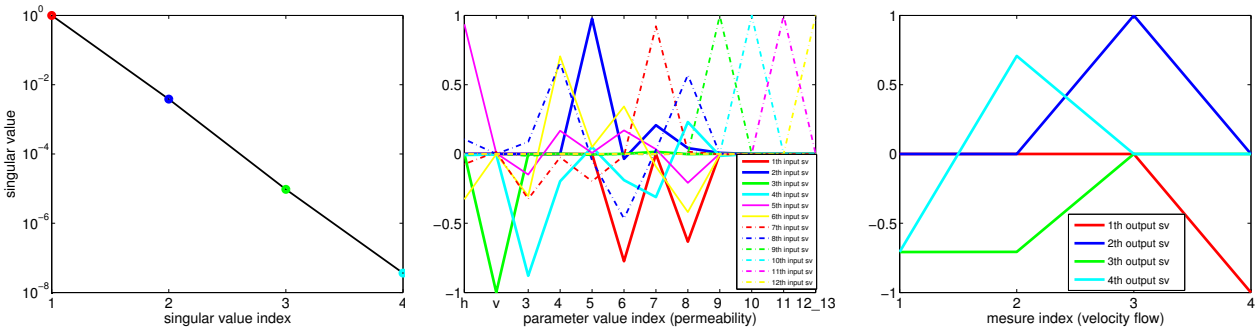
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-14}, k_v = 10^{-14}$	10^{-12}	10^{-11}	10^{-7}	10^{-4}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [3.78 10^{-1} m^3/s] (b) (c)

FIG. B.31 – Résultats d'analyse de sensibilité avec données.

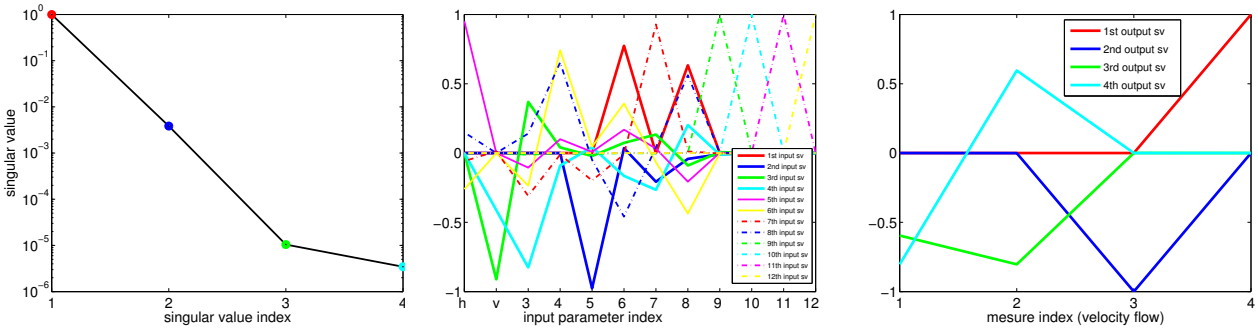
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-12}, k_v = 10^{-14}$	10^{-12}	10^{-11}	10^{-7}	10^{-4}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [3.78 10^{-1} m^3/s] (b) (c)

FIG. B.32 – Résultats d'analyse de sensibilité avec données.

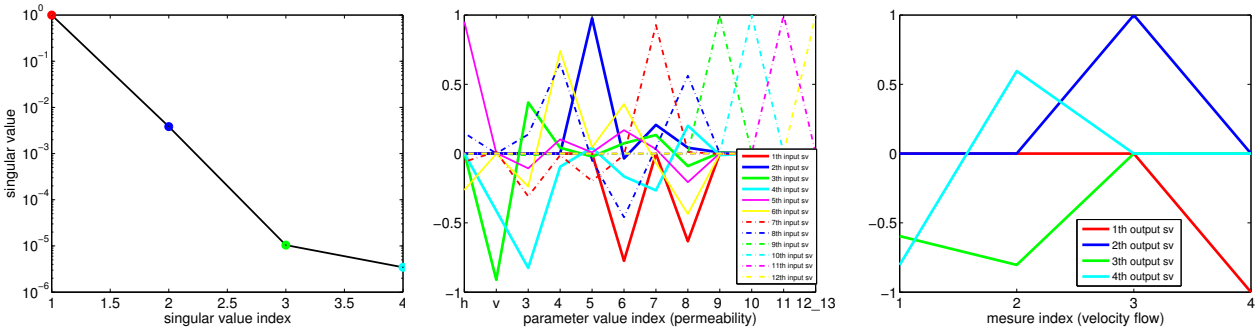
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-14}, k_v = 10^{-14}$	10^{-10}	10^{-9}	10^{-7}	10^{-4}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [3.78 10^{-1} m^3/s] (b) (c)

FIG. B.33 – Résultats d'analyse de sensibilité avec données.

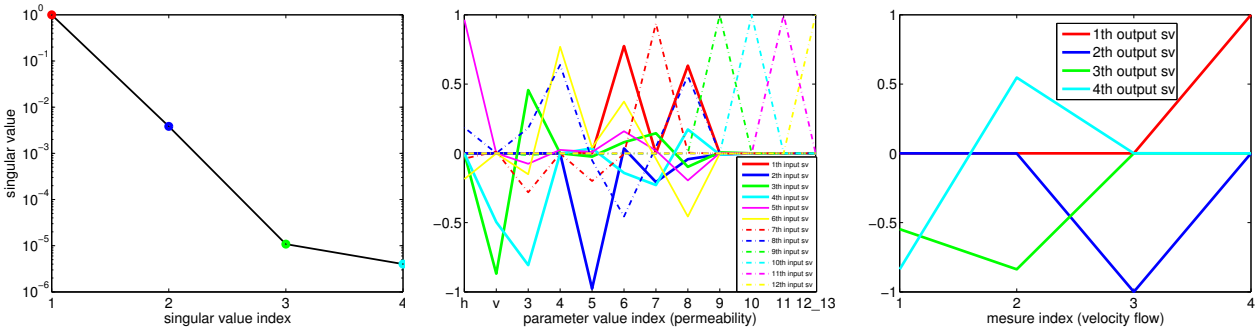
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-12}, k_v = 10^{-14}$	10^{-10}	10^{-9}	10^{-7}	10^{-4}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [3.78 10^{-1} m^3/s] (b) (c)

FIG. B.34 – Résultats d'analyse de sensibilité avec données.

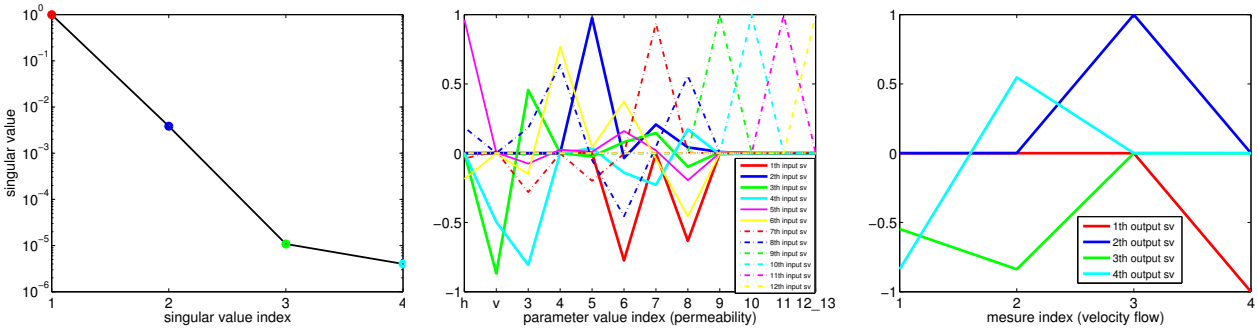
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-14}, k_v = 10^{-14}$	10^{-10}	10^{-7}	10^{-7}	10^{-4}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [3.78 10^{-1} m^3/s] (b) (c)

FIG. B.35 – Résultats d'analyse de sensibilité avec données.

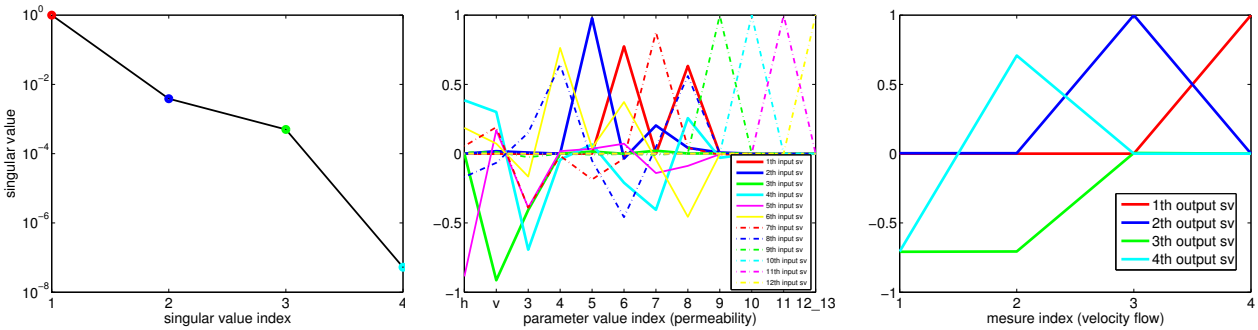
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-12}, k_v = 10^{-14}$	10^{-10}	10^{-7}	10^{-7}	10^{-4}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [3.78 10^{-1} m^3/s] (b) (c)

FIG. B.36 – Résultats d'analyse de sensibilité avec données.

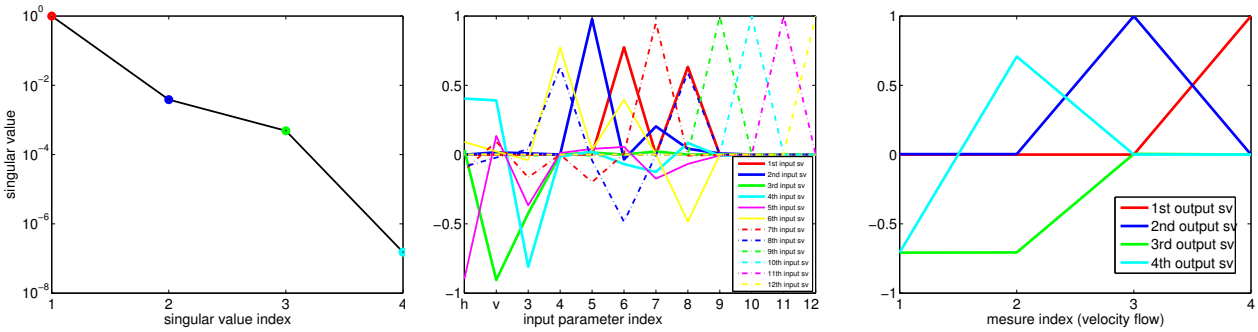
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-12}, k_v = 10^{-12}$	10^{-12}	10^{-9}	10^{-7}	10^{-4}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [3.78 10^{-1} m^3/s] (b) (c)

FIG. B.37 – Résultats d'analyse de sensibilité avec données.

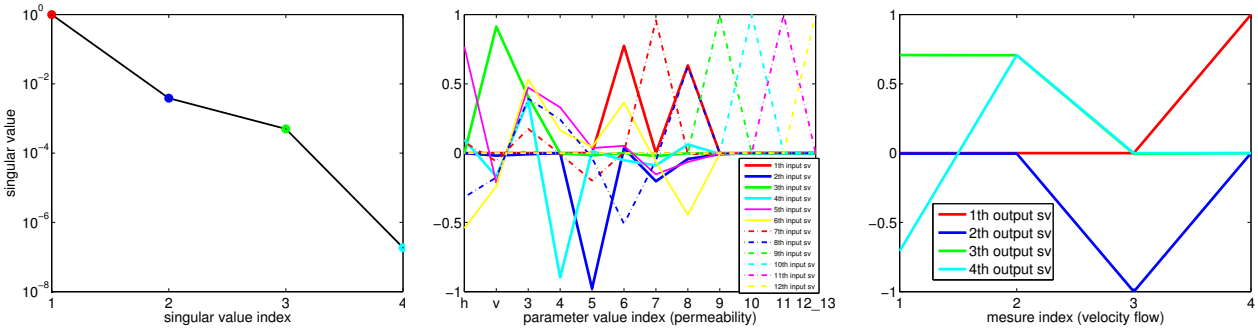
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-10}, k_v = 10^{-12}$	10^{-12}	10^{-9}	10^{-7}	10^{-4}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [3.78 10^{-1} m^3/s] (b) (c)

FIG. B.38 – Résultats d'analyse de sensibilité avec données.

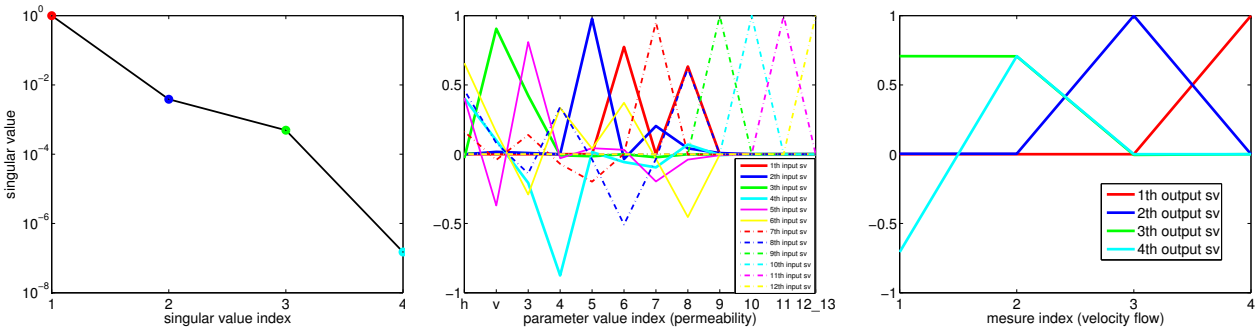
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-12}, k_v = 10^{-12}$	10^{-12}	10^{-11}	10^{-7}	10^{-4}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [3.78 10^{-1} m^3/s] (b) (c)

FIG. B.39 – Résultats d'analyse de sensibilité avec données.

zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-10}, k_v = 10^{-12}$	10^{-12}	10^{-11}	10^{-7}	10^{-4}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [3.78 10^{-1} m^3/s] (b) (c)

FIG. B.40 – Résultats d'analyse de sensibilité avec données.

B.2.3 Résultats où la hiérarchie des influences est modifiée

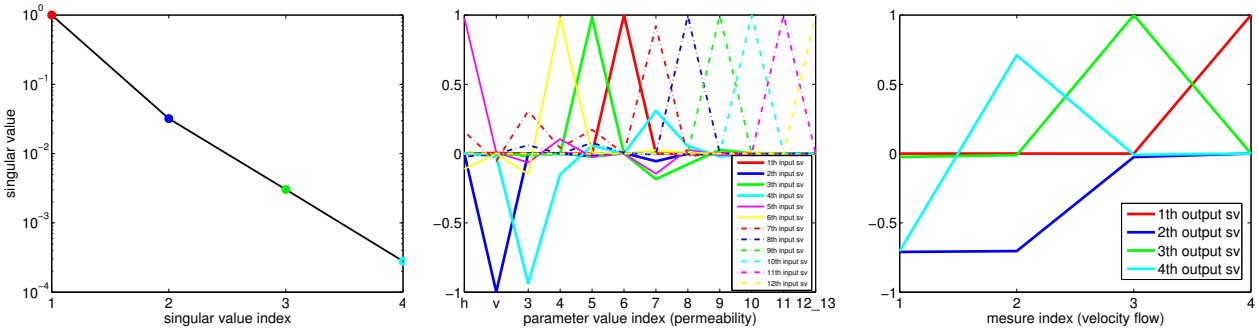
Avec la plus petite valeur possible pour $\frac{k_5}{k_v}$ (10^3). Figure B.41 et Figure B.44 : voir les explications en section 7.1.7.3.

Avec la plus petite valeur possible pour $\frac{k_5}{k_v}$ (10^3) et la plus petite valeur possible pour $\frac{k_3}{k_v}$ (10^0). Figure B.45 et Figure B.46 : voir les explications en section 7.1.7.3. Figure B.47 et Figure B.48 : voir les explications en section 7.1.7.3.

Avec la plus petite valeur possible pour $\frac{k_5}{k_v}$ (10^3) et la plus petite valeur possible pour $\frac{k_6}{k_v}$ (10^4). Figure B.49 et Figure B.52 : voir les explications en section 7.1.7.3.

Avec la plus petite valeur possible pour $\frac{k_5}{k_v}$ (10^3), la plus petite valeur possible pour $\frac{k_6}{k_v}$ (10^4) et la plus petite valeur possible pour $\frac{k_3}{k_v}$ minimal (10^0). Figure B.53 et Figure B.54 : voir les explications en section 7.1.7.3. Figure B.55 et Figure B.56 : voir les explications en section 7.1.7.3.

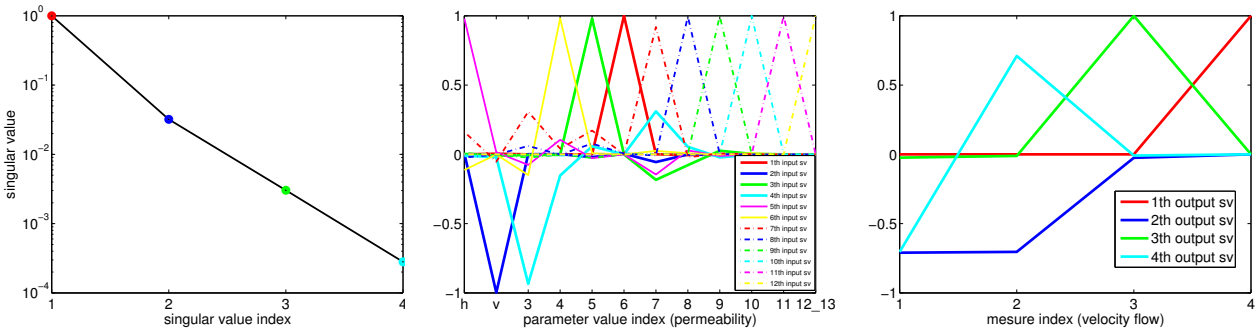
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-12}, k_v = 10^{-12}$	10^{-10}	10^{-9}	10^{-9}	10^{-6}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [9.29 10^{-3} m^3/s] (b) (c)

FIG. B.41 – Résultats d'analyse de sensibilité avec données.

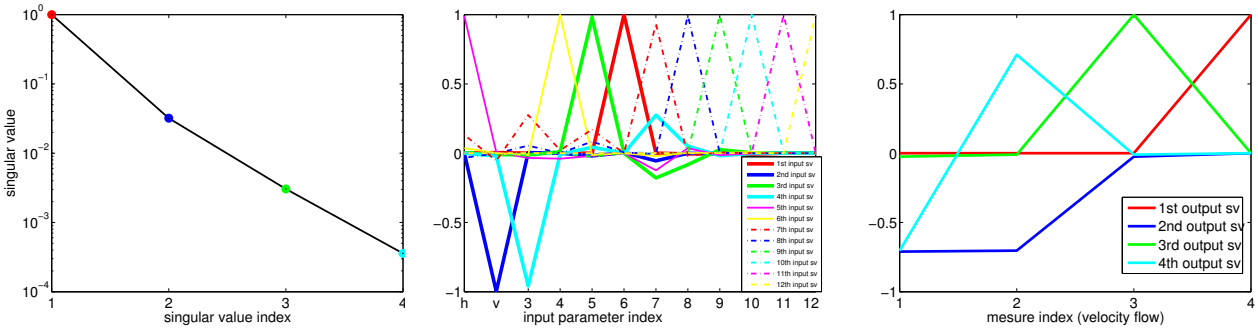
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-10}, k_v = 10^{-12}$	10^{-10}	10^{-9}	10^{-9}	10^{-6}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [9.29 10^{-3} m^3/s] (b) (c)

FIG. B.42 – Résultats d'analyse de sensibilité avec données.

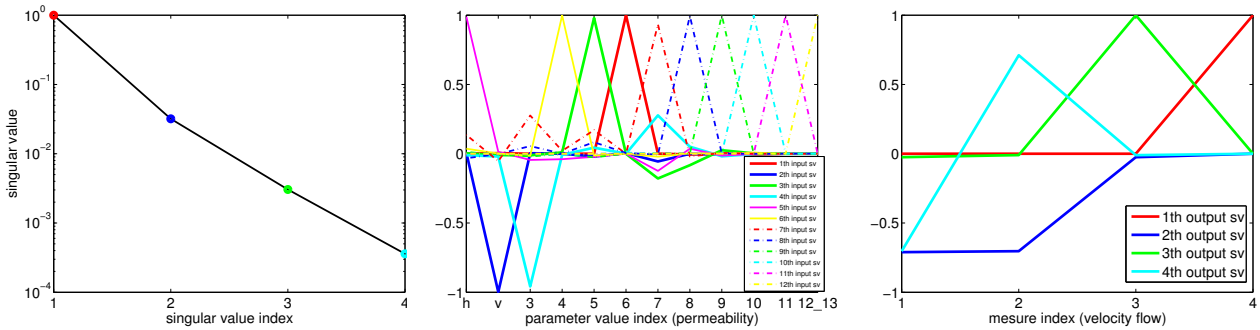
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-12}, k_v = 10^{-12}$	10^{-10}	10^{-7}	10^{-9}	10^{-6}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [9.29 10^{-3} m^3/s] (b) (c)

FIG. B.43 – Résultats d'analyse de sensibilité avec données.

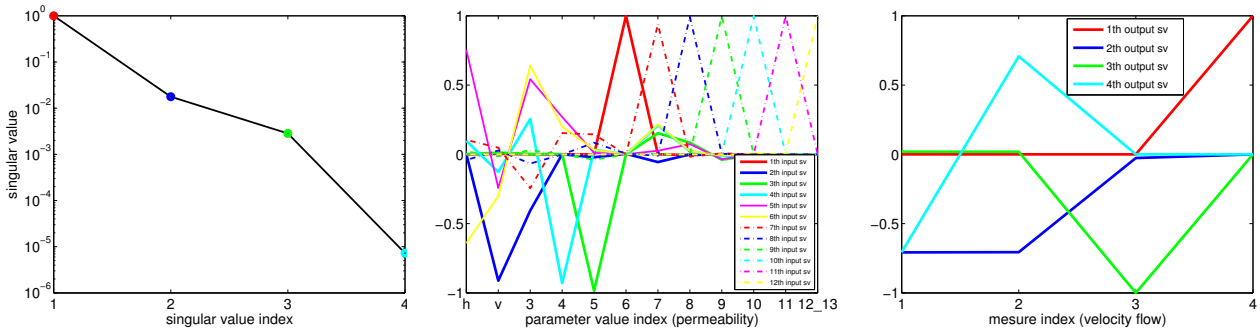
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-10}, k_v = 10^{-12}$	10^{-10}	10^{-7}	10^{-9}	10^{-6}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [9.29 10^{-3} m^3/s] (b) (c)

FIG. B.44 – Résultats d'analyse de sensibilité avec données.

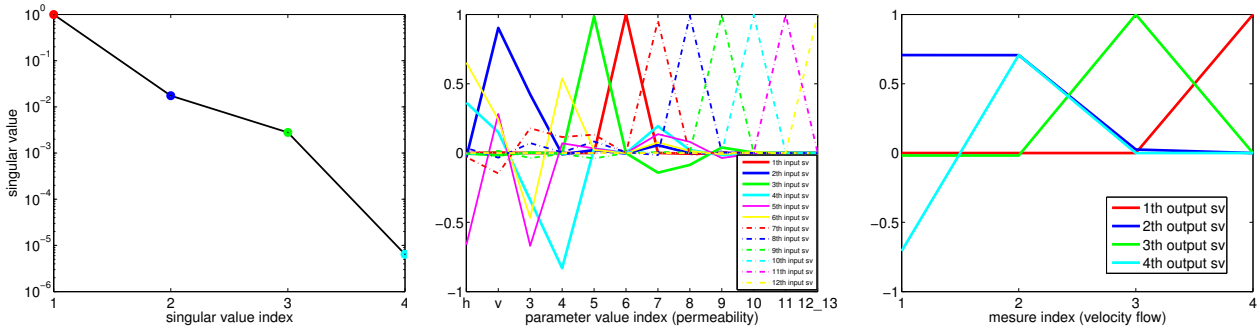
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-12}, k_v = 10^{-12}$	10^{-12}	10^{-11}	10^{-9}	10^{-6}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [9.28 10^{-3} m^3/s] (b) (c)

FIG. B.45 – Résultats d'analyse de sensibilité avec données.

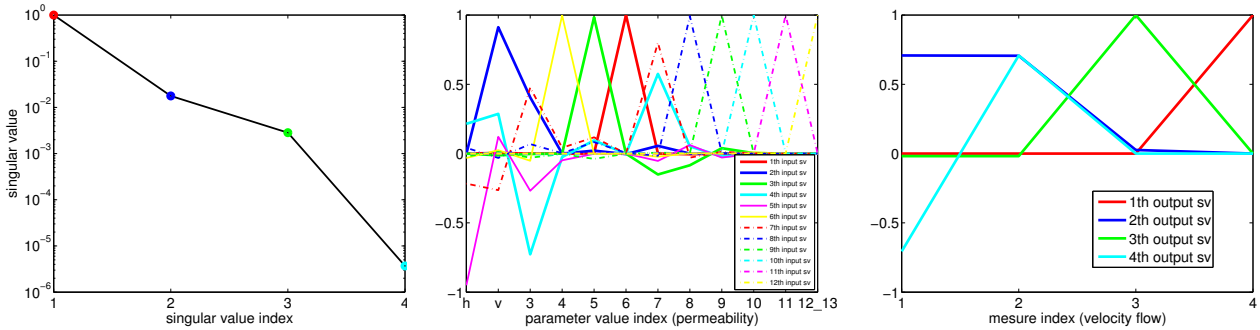
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-10}, k_v = 10^{-12}$	10^{-12}	10^{-11}	10^{-9}	10^{-6}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [9.28 10^{-3} m^3/s] (b) (c)

FIG. B.46 – Résultats d'analyse de sensibilité avec données.

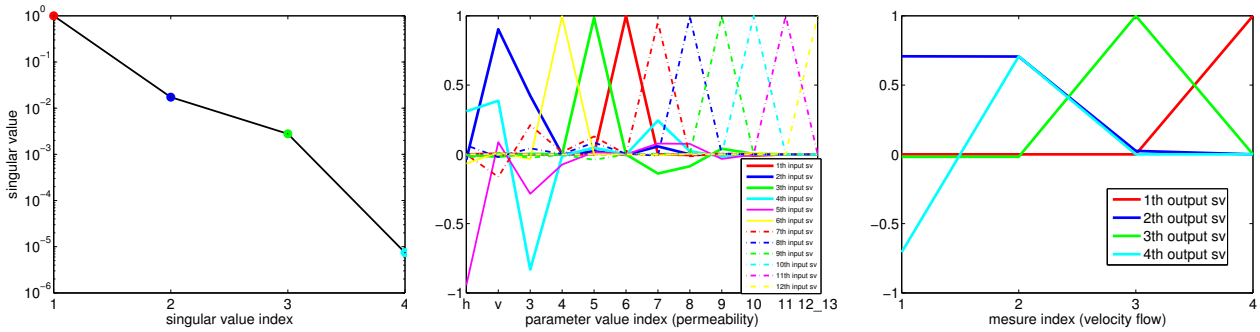
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-12}, k_v = 10^{-12}$	10^{-12}	10^{-9}	10^{-9}	10^{-6}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [9.29 10^{-3} m^3/s] (b) (c)

FIG. B.47 – Résultats d'analyse de sensibilité avec données.

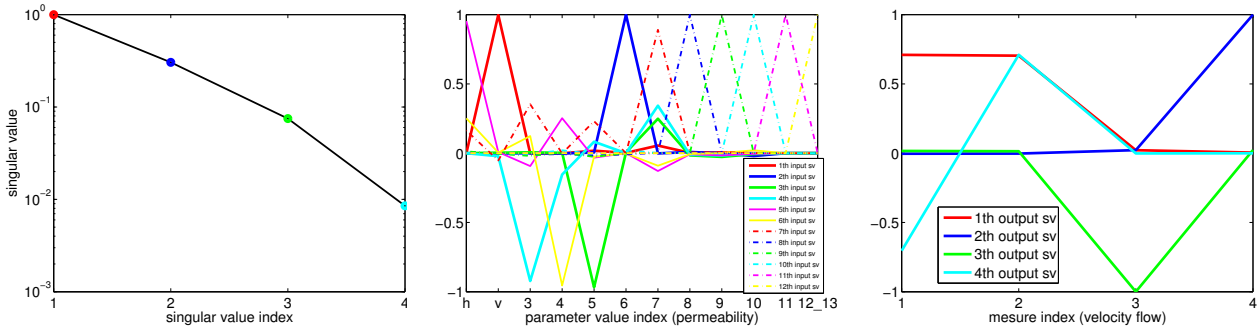
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-10}, k_v = 10^{-12}$	10^{-12}	10^{-9}	10^{-9}	10^{-6}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [9.29 10^{-3} m^3/s] (b) (c)

FIG. B.48 – Résultats d'analyse de sensibilité avec données.

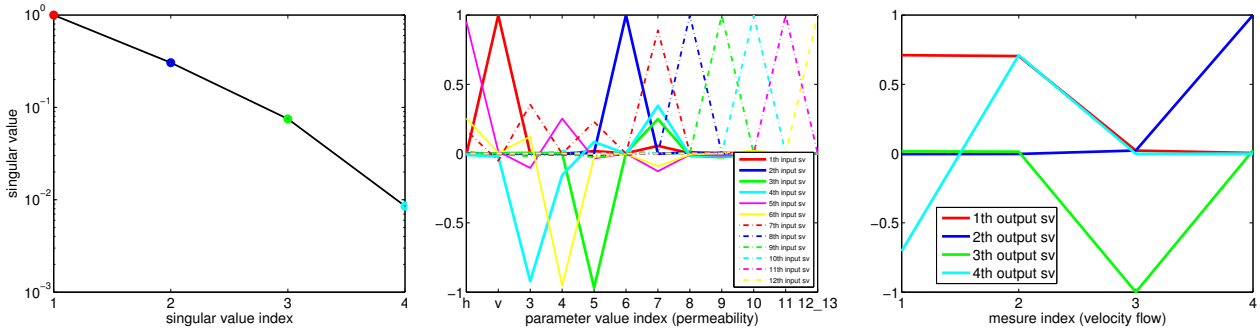
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-12}, k_v = 10^{-12}$	10^{-10}	10^{-9}	10^{-9}	10^{-8}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [2.96 10^{-4} m^3/s] (b) (c)

FIG. B.49 – Résultats d'analyse de sensibilité avec données.

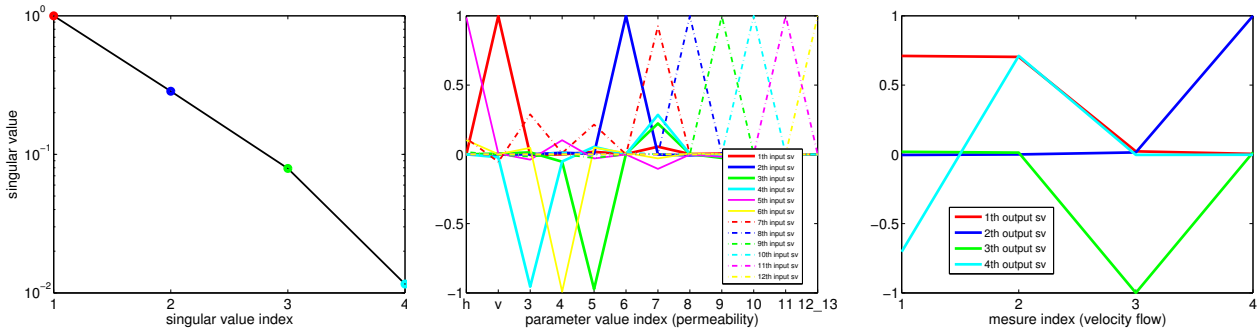
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-10}, k_v = 10^{-12}$	10^{-10}	10^{-9}	10^{-9}	10^{-8}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [2.96 10^{-4} m^3/s] (b) (c)

FIG. B.50 – Résultats d'analyse de sensibilité avec données.

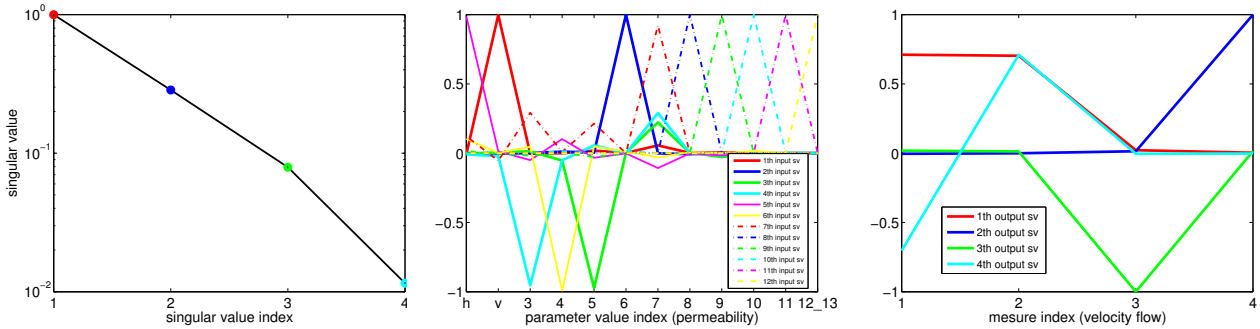
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-12}, k_v = 10^{-12}$	10^{-10}	10^{-7}	10^{-9}	10^{-8}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [2.96 10^{-4} m^3/s] (b) (c)

FIG. B.51 – Résultats d'analyse de sensibilité avec données.

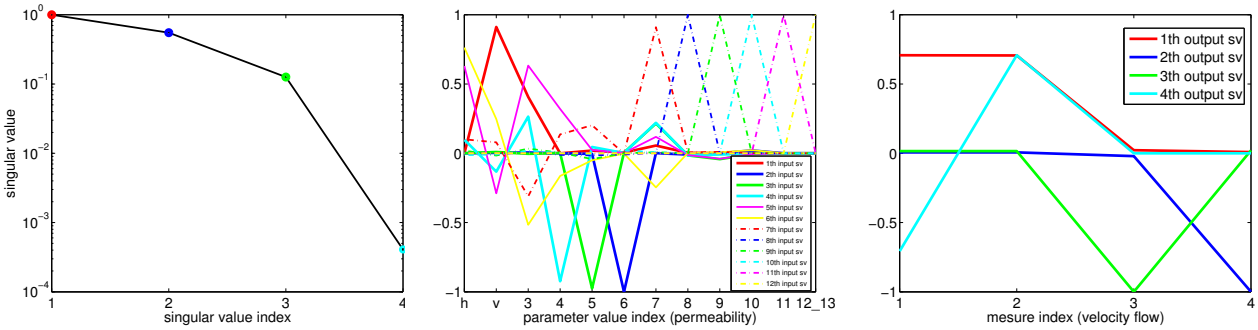
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-10}, k_v = 10^{-12}$	10^{-10}	10^{-7}	10^{-9}	10^{-8}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [2.96 10^{-4} m^3/s] (b) (c)

FIG. B.52 – Résultats d'analyse de sensibilité avec données.

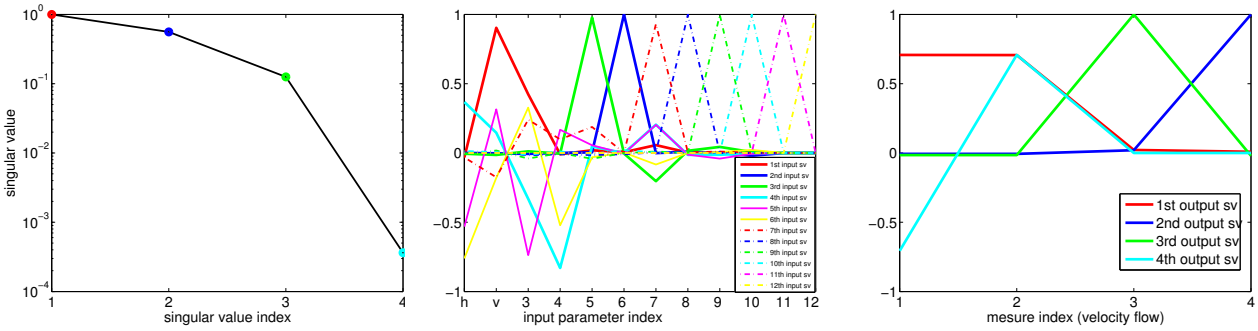
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-12}, k_v = 10^{-12}$	10^{-12}	10^{-11}	10^{-9}	10^{-8}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [1.64 10^{-4} m^3/s] (b) (c)

FIG. B.53 – Résultats d'analyse de sensibilité avec données.

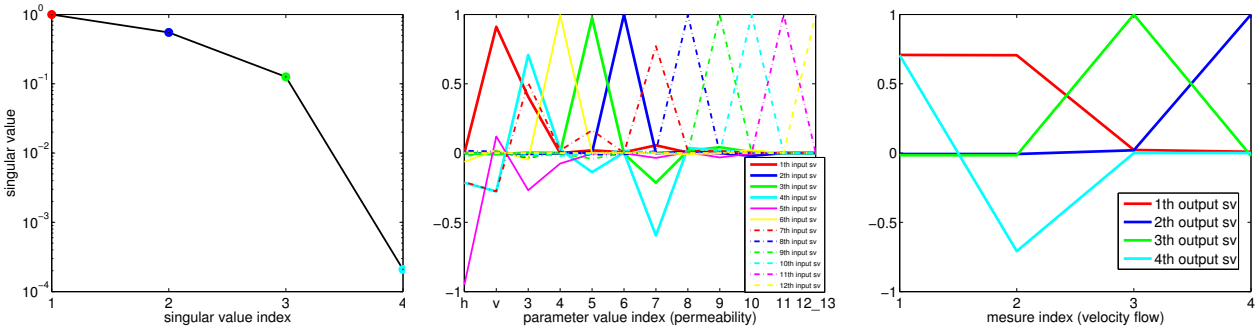
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-10}, k_v = 10^{-12}$	10^{-12}	10^{-11}	10^{-9}	10^{-8}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [1.62 10^{-4} m^3/s] (b) (c)

FIG. B.54 – Résultats d'analyse de sensibilité avec données.

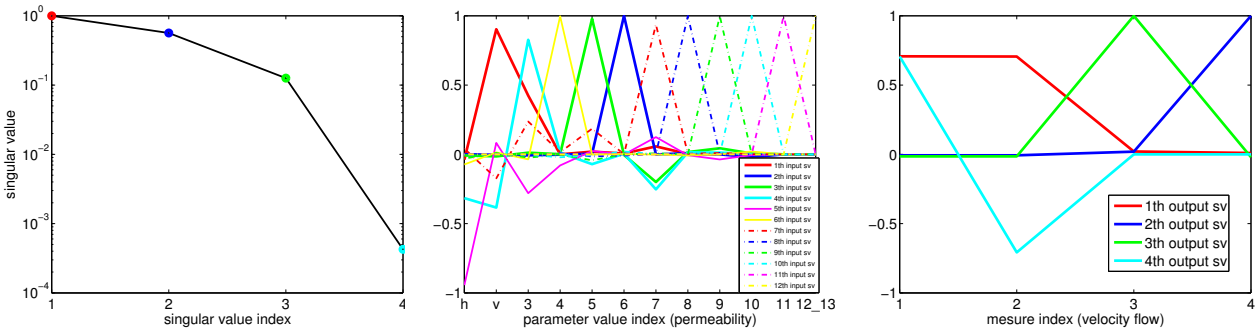
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-12}, k_v = 10^{-12}$	10^{-12}	10^{-9}	10^{-9}	10^{-8}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [1.64 10^{-4} m^3/s] (b) (c)

FIG. B.55 – Résultats d'analyse de sensibilité avec données.

zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-10}, k_v = 10^{-12}$	10^{-12}	10^{-9}	10^{-9}	10^{-8}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [1.60 10^{-4} m^3/s] (b) (c)

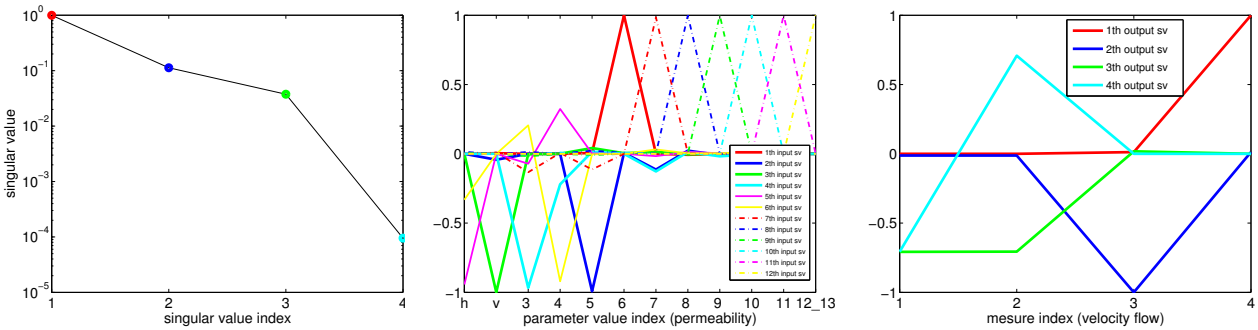
FIG. B.56 – Résultats d'analyse de sensibilité avec données.

B.2.4 Résultats presque identiques à ceux de la première étude locale

Rien à signaler pour les Figures B.57 à B.72.

Les résultats des Figures B.73 à B.76 sont des résultats intermédiaires : on commence à voir apparaître un couplage entre les influences de k_v et k_3 sur Φ_1 et Φ_2.

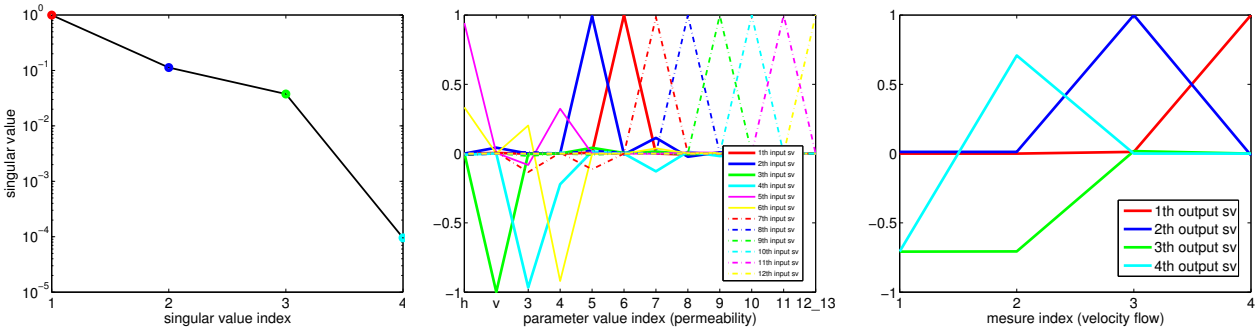
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-12}, k_v = 10^{-12}$	10^{-10}	10^{-9}	10^{-7}	10^{-6}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [9.31 10^{-3} m^3/s] (b) (c)

FIG. B.57 – Résultats d'analyse de sensibilité avec données.

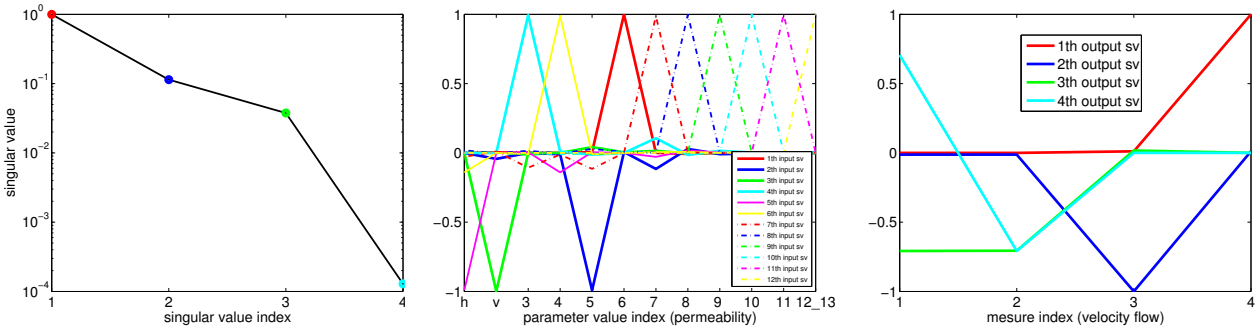
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or $K^\alpha(rmm.s^{-1})$	$k_h = 10^{-10}, k_v = 10^{-12}$	10^{-10}	10^{-9}	10^{-7}	10^{-6}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [9.31 10^{-3} m^3/s] (b) (c)

FIG. B.58 – Résultats d'analyse de sensibilité avec données.

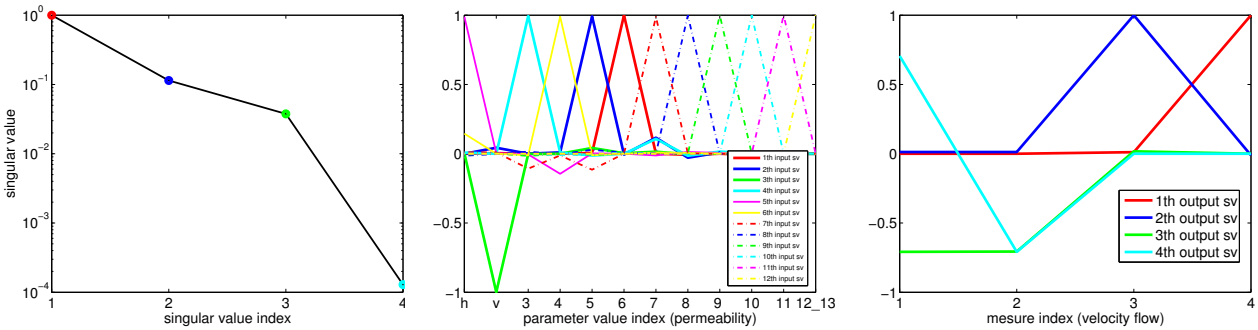
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-10}, k_v = 10^{-10}$	10^{-10}	10^{-7}	10^{-7}	10^{-6}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [9.31 10^{-3} m^3/s] (b) (c)

FIG. B.59 – Résultats d'analyse de sensibilité avec données.

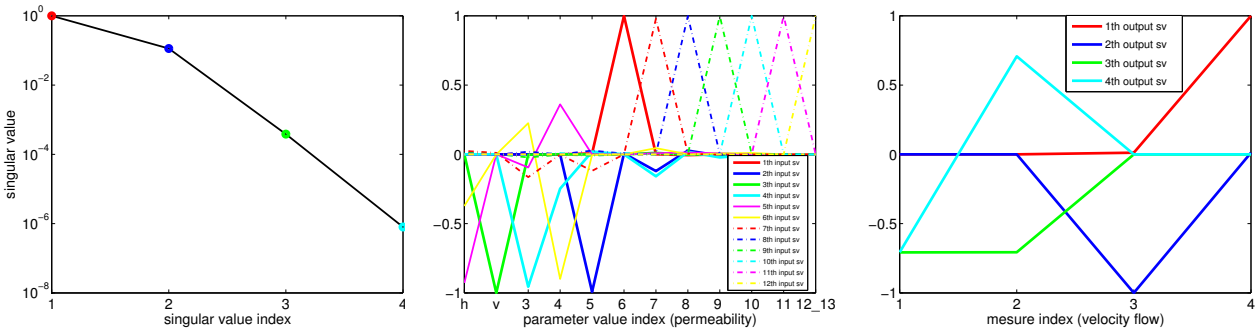
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-10}, k_v = 10^{-12}$	10^{-10}	10^{-7}	10^{-7}	10^{-6}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [9.32 10^{-3} m^3/s] (b) (c)

FIG. B.60 – Résultats d'analyse de sensibilité avec données.

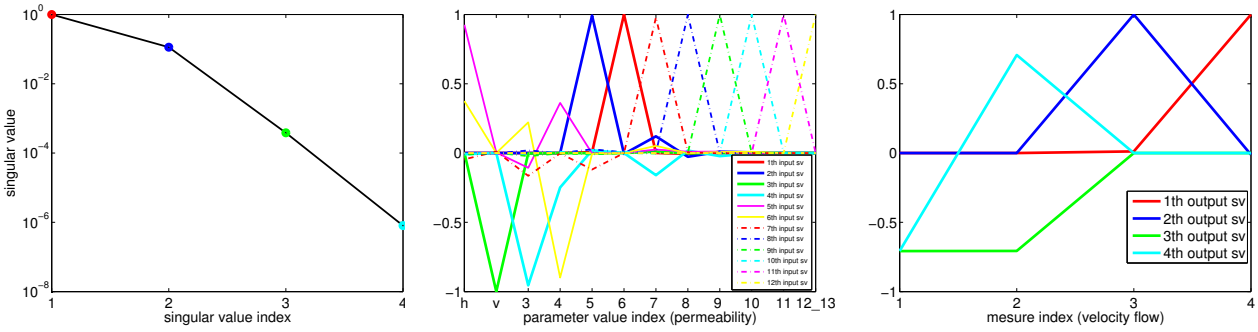
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-14}, k_v = 10^{-14}$	10^{-12}	10^{-11}	10^{-7}	10^{-6}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [9.30 10^{-3} m^3/s] (b) (c)

FIG. B.61 – Résultats d'analyse de sensibilité avec données.

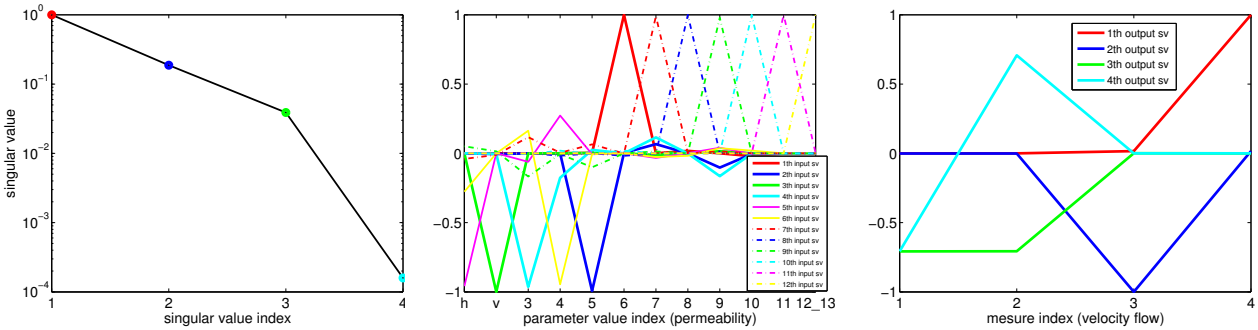
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-12}, k_v = 10^{-14}$	10^{-12}	10^{-11}	10^{-7}	10^{-6}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [9.30 10^{-3} m^3/s] (b) (c)

FIG. B.62 – Résultats d'analyse de sensibilité avec données.

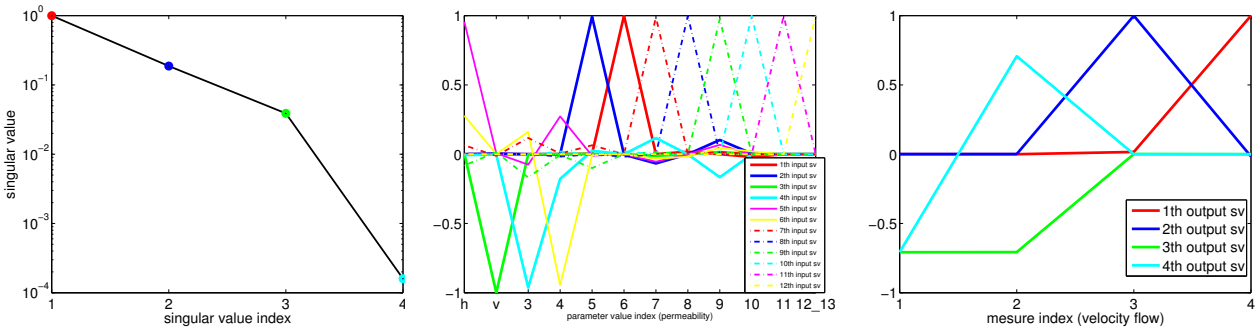
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-14}, k_v = 10^{-14}$	10^{-12}	10^{-11}	10^{-9}	10^{-8}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [8.93 10^{-5} m^3/s] (b) (c)

FIG. B.63 – Résultats d'analyse de sensibilité avec données.

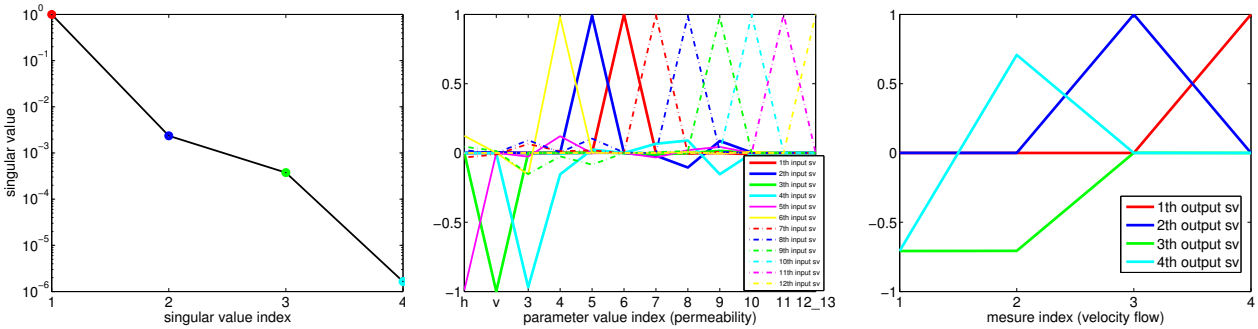
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-12}, k_v = 10^{-14}$	10^{-12}	10^{-11}	10^{-9}	10^{-8}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [8.93 10^{-5} m^3/s] (b) (c)

FIG. B.64 – Résultats d'analyse de sensibilité avec données.

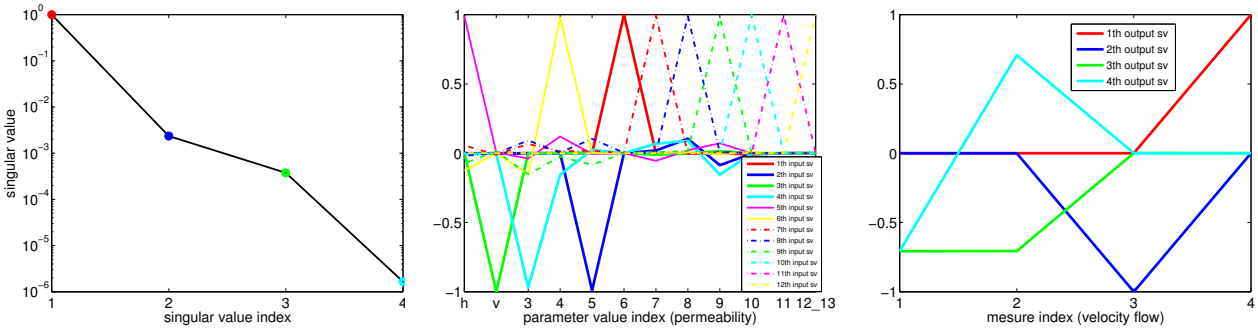
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-14}, k_v = 10^{-14}$	10^{-12}	10^{-11}	10^{-9}	10^{-6}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [9.27 10^{-3} m^3/s] (b) (c)

FIG. B.65 – Résultats d'analyse de sensibilité avec données.

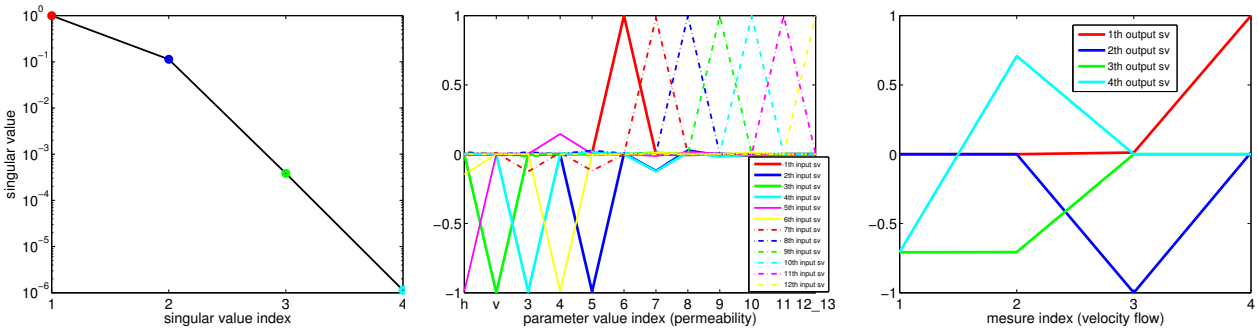
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-12}, k_v = 10^{-14}$	10^{-12}	10^{-11}	10^{-9}	10^{-6}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [9.27 10^{-3} m^3/s] (b) (c)

FIG. B.66 – Résultats d'analyse de sensibilité avec données.

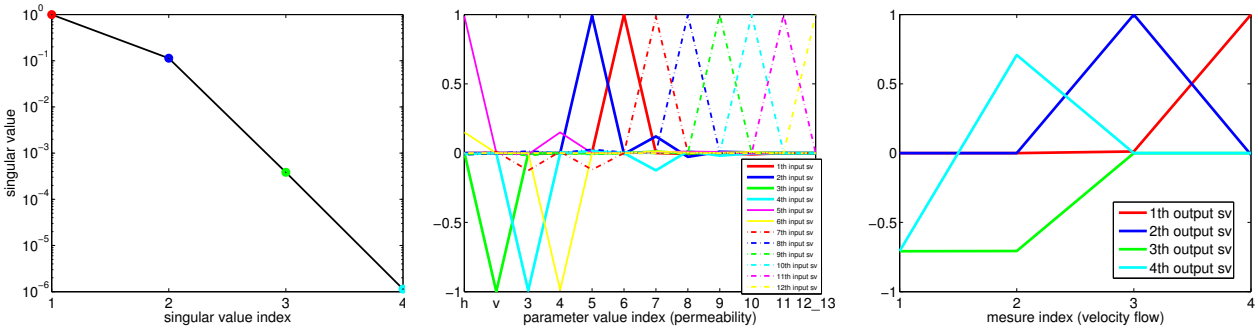
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-14}, k_v = 10^{-14}$	10^{-12}	10^{-9}	10^{-7}	10^{-6}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [9.30 10^{-3} m^3/s] (b) (c)

FIG. B.67 – Résultats d'analyse de sensibilité avec données.

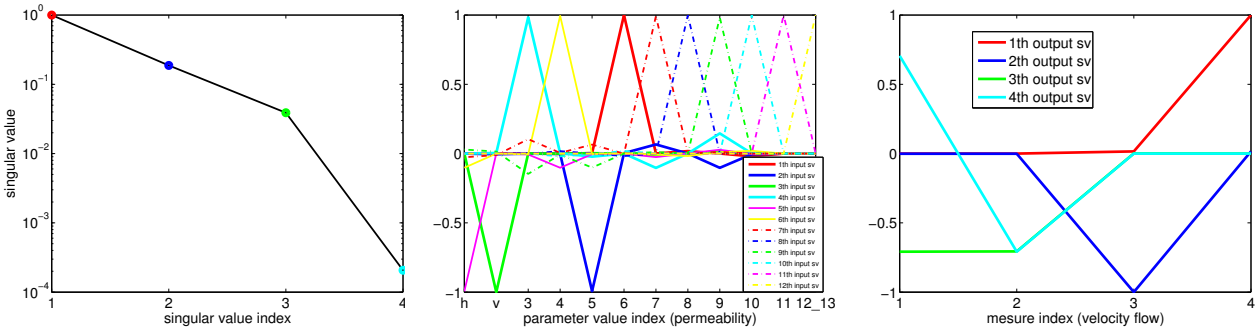
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-12}, k_v = 10^{-14}$	10^{-12}	10^{-9}	10^{-7}	10^{-6}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [9.30 10^{-3} m^3/s] (b) (c)

FIG. B.68 – Résultats d'analyse de sensibilité avec données.

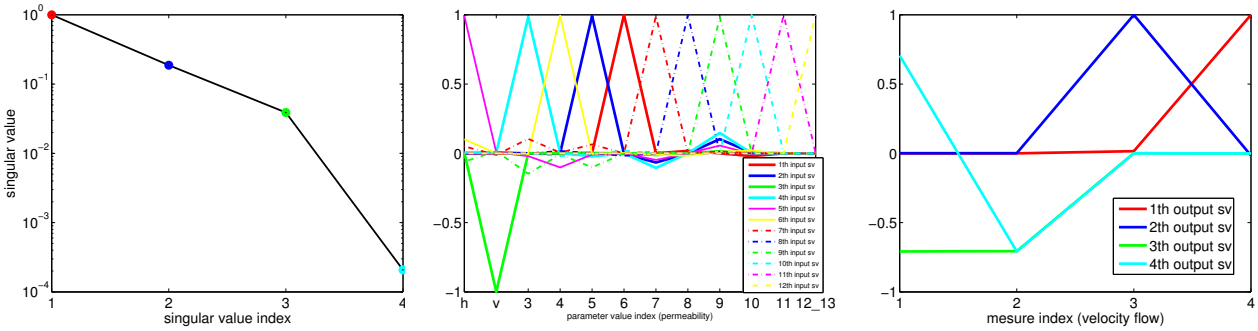
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-14}, k_v = 10^{-14}$	10^{-12}	10^{-9}	10^{-9}	10^{-8}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [8.93 10^{-5} m^3/s] (b) (c)

FIG. B.69 – Résultats d'analyse de sensibilité avec données.

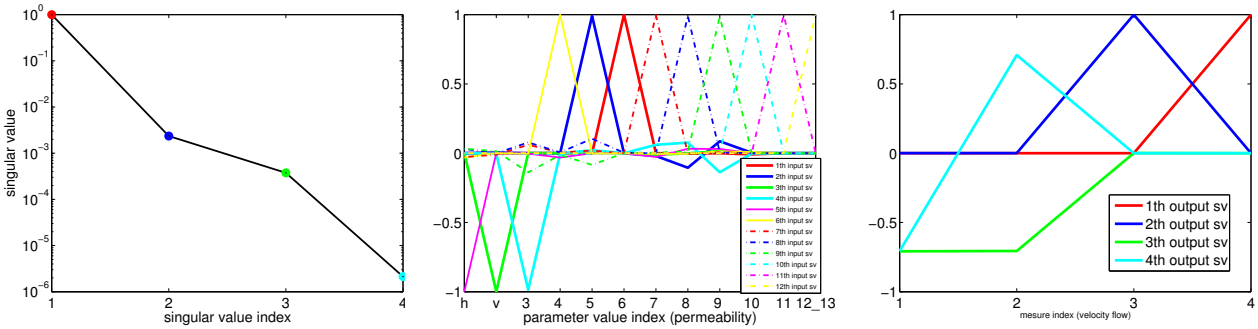
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-12}, k_v = 10^{-14}$	10^{-12}	10^{-9}	10^{-9}	10^{-8}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [8.93 10^{-5} m^3/s] (b) (c)

FIG. B.70 – Résultats d'analyse de sensibilité avec données.

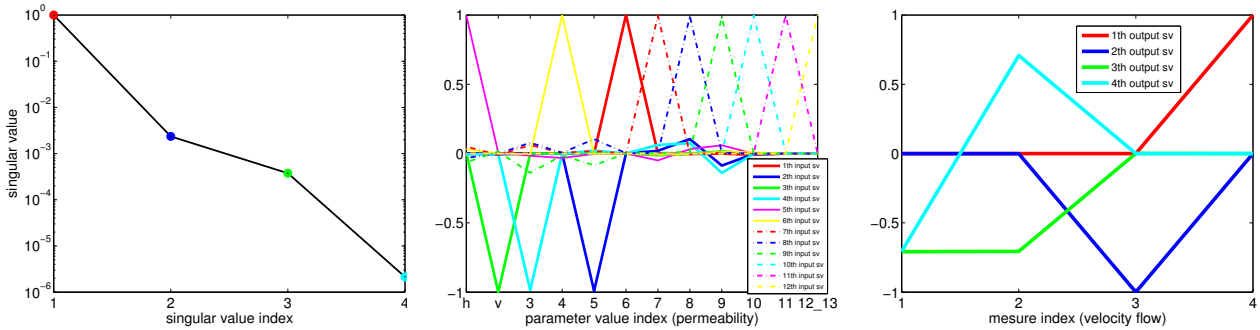
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-14}, k_v = 10^{-14}$	10^{-12}	10^{-9}	10^{-9}	10^{-6}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [9.27 10^{-3} m^3/s] (b) (c)

FIG. B.71 – Résultats d'analyse de sensibilité avec données.

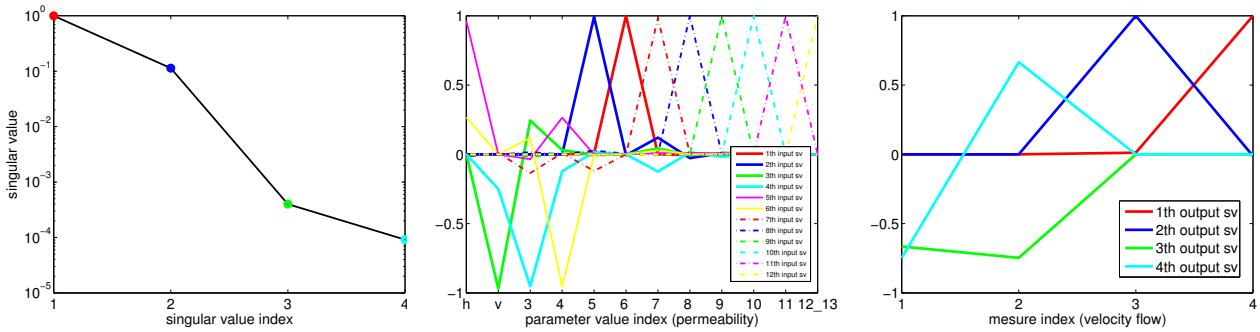
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-12}, k_v = 10^{-14}$	10^{-12}	10^{-9}	10^{-9}	10^{-6}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [9.27 10^{-3} m^3/s] (b) (c)

FIG. B.72 – Résultats d'analyse de sensibilité avec données.

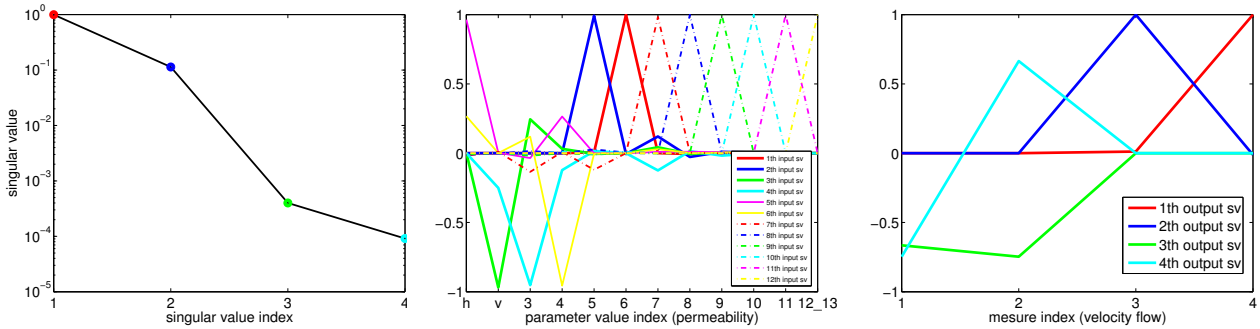
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-14}, k_v = 10^{-14}$	10^{-10}	10^{-9}	10^{-7}	10^{-6}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [9.30 10^{-3} m^3/s] (b) (c)

FIG. B.73 – Résultats d'analyse de sensibilité avec données.

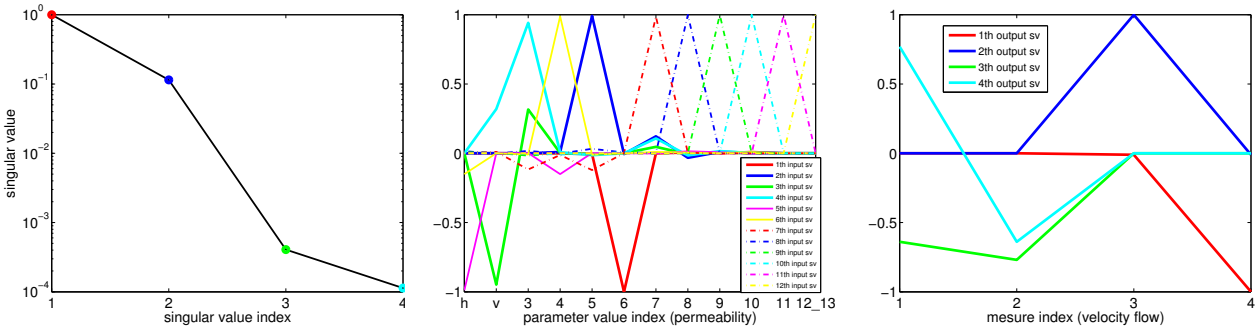
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-12}, k_v = 10^{-14}$	10^{-10}	10^{-9}	10^{-7}	10^{-6}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [9.30 10^{-3} m^3/s] (b) (c)

FIG. B.74 – Résultats d'analyse de sensibilité avec données.

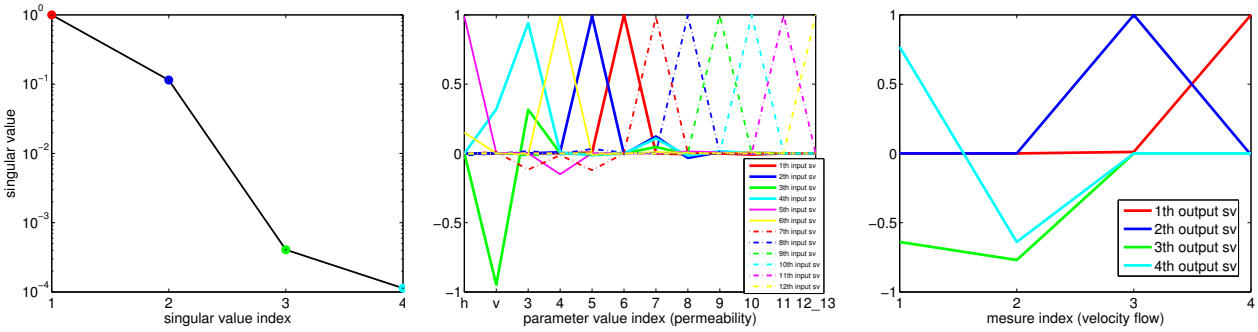
zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-14}, k_v = 10^{-14}$	10^{-10}	10^{-7}	10^{-7}	10^{-6}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [9.30 10^{-3} m^3/s] (b) (c)

FIG. B.75 – Résultats d'analyse de sensibilité avec données.

zone n^o	1, 2, 14	3	4	5	6	7	8	9	10	11	12-13
k_α or K^α(m.s^{-1})	$k_h = 10^{-12}, k_v = 10^{-14}$	10^{-10}	10^{-7}	10^{-7}	10^{-6}	10^{-9}	2.10^{-7}	10^{-12}	10^{-11}	3.10^{-4}	3.10^{-5}



(a) [9.30 10^{-3} m^3/s] (b) (c)

FIG. B.76 – Résultats d'analyse de sensibilité avec données.

Bibliographie

[1] P. Al Khoury, G. Chavent, F. Clément, and P. Hervé. Inversion of spectroscopic data, application on co_2 radiation of flame combustion. *Inverse Problems in Science and Engineering*, 13(3) :219–240, June 2005.

[2] D. N. Arnold and F. Brezzi. Mixed and nonconforming finite element methods : Inplementation, postprocessing and error estimates. *Mathematical Modeling and Numerical Analysis*, 19(1) :7–32, 1985.

[3] P. Aubert, N. Di Césaré, and O. Pironneau. Automatic differentiation in C++ using expression templates and application to a flow control problem. *Comput Visual Sci*, 3 :197–208, 2001.

[4] P. Bastian. *Numerical Computation of Multiphase Flows in Porous Media*, chapter Modeling Immiscible Fluis Flow in Porous Media. Technischen Fakultät der Christian-Albrechts-Universität Kiel, Habilitationsschrift, 1999.

[5] J. Bear and A. Verruijt. *Modeling Groundwater Flow and Pollution*. D.Reidel Publishing Company, 1987.

[6] C. H. Bischof, H. M. Bücker, and A. Rasch. Sensitivity analysis of turbulence models using automatic differentiation. *SIAM J. on Scient. Comp.*, 26 :510–522, 2004.

[7] Björck. *Numerical methods for least squares problems*. SIAM, 1996.

[8] J. F. Bonnans, J. C. Gilbert, C. Lemaréchal, and C. A. Sagastizábal. *Numerical Optimization Theoretical and Practical Aspects*, chapter 3,11–16. Springer, 1997.

[9] H. Brezis. *Analyse fonctionnelle*. Dunod, 1999.

[10] F. Brezzi and M. Fortin. *Mixed and Hybrid Finite Element Methods*. Springer, 1991.

[11] D. G. Cacuci and M. Ionescu-Bujor. A comparative review of sensitivity and uncertainty analysis of large-scale systems—ii : Statistical methods. *Nuclear Science and Engineering*, 147 :204–217, 2004.

[12] D. G. Cacuci, M. Ionescu-Bujor, and I. M. Navon. *Sensitivity and Uncertainty Analysis*, volume II — Applications to Large-Scalr Systems. Chapman & Hall/CRC, 2005.

[13] Dan G. Cacuci. *Sensitivity and Uncertainty Analysis*, volume I — Theory. Chapman & Hall/CRC, 2003.

[14] G. Chavent and F. Clément. Détermination de profils de température pendant la détonation d'un explosif liquide, le nitrométhane. Technical Report 4641, INRIA, 2002.

[15] G. Chavent and J. Jaffre. *Mathematical models and finite elements*. Elsevier Science Publishers B.V., 1986.

[16] G. Chavent and J. Roberts. A unified physical presentation of mixed, mixed-hybrid finite elements and standard finite difference approximations for the determination of velocities in waterflow problems. *Advances in Water Ressources*, 14(6) :329–348, December 1991.

[17] Z. Chen, G. Huan, and B. Li. An improved impes method for two-phase flow in porous media. *Transport in Porous Media*, 54, 2004.

[18] M. T. Chu and R. E Funderlic. The centroid decomposition : Relationships between discrete variational decomposition and SVDs. *SIAM J. Matrix Anal. Appl.*, 23 :1025–1044, 2002.

[19] F. Clément, N. Khvoenkova, A. Cartalade, and Ph. Montarnal. Analyse de sensibilité et estimation de paramètres de transport pour une équation de diffusion, approche par état adjoint. Technical Report 5132, INRIA, 2004.

[20] L. C. Cowsar, J. Mandel, and M. F. Wheeler. Balancing domain decomposition for mixed finite elements. *Math. of Comp.*, 64 :989–1015, 1995.

[21] H. Darcy. Détermination des lois d'écoulement de l'eau à travers le sable. In *Les Fontaines Publiques de la Ville de Dijon*, chapter Appendice - Note D. V. Dalmont, Paris, 1856.

[22] P. Deuflhard. *Newton Methods for Nonlinear Problems*. Springer, 2004.

[23] K. V. M. Fernando and H. Nicholson. Karhunen-Loève expansion with reference to singular-value decomposition and separation of variables. *Proc IEE-D*, 127 :204–206, 1980.

[24] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design patterns elements of reusable object-oriented software*. Addison-Weisley, 1995.

[25] G.Chavent. The global pressure, a new concept for the modelization of compressible two-phase flows in porous media. In *Flow and Transport in Porous Media*, pages 191–198. Euromech 143 and Delft, A. A. BALKEMA, 1981.

[26] J. J. Gerbrands. On the relationships between SVD, KLT and PCA. *Pattern Recognition*, 14 :375–381, 1981.

[27] E. Giffaut, G. Pepin, C. Sallaberry, S. Schumacher, and J. Wendling. Projet HAVL. mise en œuvre probabiliste d'un cas-test de calculs de sûreté. hypothèses et données. Technical Report C NT ACSS 02-110, ANDRA, 2002.

[28] G. Golub and Ch.F. van Loan. *Matrix computations*. John Hopkins University Press, 1996.

[29] A. Griewank. *Evaluating Derevatives : Principles and Techniques of Algorithmic Differentiation*. Frontiers in Applied Mathematics. SIAM, Philadelphia, 2000.

[30] A. Griewank, A. Kowarz, J. Utke, O. Vogel, and A. Walther. ADOL-C : A package for the automatic differentiation of algorithms written in C/C++. documentation for

ADOL-C, version 1.10.0, http ://www.math.tu-dresden.de/˜adol-c/adolc110.ps, July 2005.

[31] A. Griewank and A. Walther. Algorithm 799 : ADOL-C : An implementation of checkpointing for the reverse and adjoint mode of computational differentiation. *ACM Transactions on Mathematical Software*, 26(1) :19–45, March 2000.

[32] J. C. Helton and F. J. Davis. Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems. Technical report, Sandia National Laboratories, November 2002.

[33] Jon C. Helton and Freddie J. Davis. *Sensitivity Analysis*, chapter 6. Wiley series in probability and statistics. Wiley, 2000.

[34] R. A. Horn and C. R. Johnson. *Topics in matrix analysis*. Cambridge University Press, 1991.

[35] http ://caml.inria.fr/.

[36] http ://valgrind.org/.

[37] http ://www.autodiff.org.

[38] http ://www.boost.org/.

[39] http ://www.cplusplus.com/doc/tutorial/.

[40] http ://www.cppreference.com/.

[41] http ://www.haskell.org.

[42] http ://www.lifev.org.

[43] http ://www.math.tu-dresden.de/˜adol c/.

[44] http ://www.math.tu dresden.de/wir/project/revolve.

[45] P. Knabner I. S. Pop, F. Radu. Mixed finite elements for the richards equation : Linearization procedure. *J. Comput. Appl. Math.*, (168) :365–373, 2004.

[46] M. Ionescu-Bujor and D. G. Cacuci. A comparative review of sensitivity and uncertainty analysis of large-scale systems—i : Deterministic methods. *Nuclear Science and Engineering*, 147 :189–203, 2004.

[47] J. C. Helton, J. D. Johnson, C. J. Sallabery, and C. B. Storlie. Survey of sampling-based methods for uncertainty and sensitivity analysis. Technical report, Sandia National Laboratories, June 2006.

[48] J. C. Helton and C. J. Sallabery. Illustration of sampling-based approaches to the calculation of expected dose in performance assessments for the proposed high level radioactive waste repository at yucca mountain, nevada. Technical report, Sandia National Laboratories, April 2007.

[49] J. Jaffré, J. E. Roberts, and X. Wang. Generalized cell-centered finite volume methods for two-phase flow in a porous medium with two rock types. 2001.

[50] C. T. Kelley. *Iterative Methods for Linear and Nonlinear Equations*, chapter 8-Global Convergence. siam, 1995.

[51] Andrew Koenig and Barbara E. Moo. *Accelerated C++ Practical Programming by Example*. Addison-Wesley, 2000.

[52] Yu. Kusnetzov and S. Repin. New mixed finite element on polygonal and polyhedral meshes. *Russ. J. Numer. Anal. Math. Modelling*, 18 :261–278, 2003.

[53] A. Marsh La Venue and J. F. Pickens. Application of a coupled adjoint sensitivity and kriging approach to calibrate a groundwater flow model. *Water Resources Research*, 28(1543–1569), June 1992.

[54] E. Laporte and P. Le Tallec. *Numerical methods in sensitivity analysis and shape optimization*. Birkäuser, 2003.

[55] P. Le Tallec. Domain decomposition methods in computational mechanics. *Computational Mechanics Advances*, 1(2), February 1994.

[56] R. J. LeVeque. *Numerical methods for conservation laws*. Birkhaüser, 1990.

[57] Y. C. Liang, H. P. Lee, S. P. Lim, W. Z. Lin, K. H. Lee, and C. G. Wu. Proper orthogonal decomposition and its applications—Part I : Theory. *J. Sound Vibration*, 252 :527–544, 2002.

[58] J. Mandel. Balancing domain decomposition. *Comm. Appl. Numer. Methods*, 9 :233–241, 1993.

[59] V. Martin. *Simulation multidomaine d'un écoulement autour d'un site de stockage de déchets*. PhD thesis, Université Paris IX Dauphine, 2004.

[60] S. Meyers. *Effective C++*. Addison-Weisley, 1998.

[61] H.J. Morel-Seytoux, P.D. Meyerand R.J. Lenhard M. Nachabe, J. Touma, and M.T. van Genuchten. Parameter equivalence for the brooks-corey and van genuchten soil characteristics : Preserving the effective capillary drive. *Water Resources Research*, 32(5) :1251–1258, 1996.

[62] J.-C. Nedelec. Mixed finite elements in $\mathbb{R}^3$. *Numer. Math.*, (35) :315–341, 1980.

[63] J. D. Paduano and D. R. Downing. Sensitivity analysis of digital flight control systems using singular-value concepts. *J. Guidance Control Dynam.*, 12 :297–303, 1989.

[64] E. Polak. *Optimization Algorithms and Consistent Approximations*. Springer, 1991.

[65] L. B. Rall. *Automatic Differentiation*. Springer, 1981.

[66] P.-A. Raviart and J.-M. Thomas. A mixed finite element method for second order elliptic problems. In I. Galligani and E. Magenes, editors, *Mathematical Aspects of Finite Element Methods*, number 606 in Lecture Notes in Mathematics, pages 292–315. Springer, Berlin.

[67] J. E. Roberts and J.-M. Thomas. Mixed and hybrid methods. In P. G. Ciarlet and J. L. Lions, editors, *Handbook of Numerical Analysis*, volume II. Elsevier, 1991.

[68] S. M. Rump. Estimation of the sensitivity of linear and nonlinear algebraic problems. *Lin. Algebra Appl.*, 153 :1–34, 1991.

[69] C. Sallaberry. Projet HAVL. mise en œuvre probabiliste d'un cas-test de calculs de sûreté. résultats. Technical Report C NT ACSS 03-042, ANDRA, 2003.

[70] A. Saltelli, S. Tarantola, F. Campolongo, and M. Ratto. *Sensitivity Analysis in Practice*. Wiley, 2004. A Guide to Assessing Scientific Models.

[71] D. Savage, editor. *The Scientific and Regulatory Basis for the Geological Disposal of Radioactive Waste*. Wiley, 1995.

[72] A. Sboui. *Quelques méthodes numériques robustes pour l'écoulement et le transport en milieu poreux*. PhD thesis, Université Paris IX Dauphine, 2007.

[73] R. W. Skaggs. *Hydrologic Modeling of small Watershell*, chapter 4-Infiltration. Number 5. American Society of Agricultural Engineers, 1982.

[74] G. Stanley and L.D. Stewart. *Design sensitivity analysis : computational issues of sensitivity equation methods*. SIAM, 2002.

[75] G. W. Stewart. On the early history of the singular value decomposition. *SIAM Review*, 35 :551–566, 1993.

[76] J.W. Thomas. *Numerical Partial Differential Equations*. Springer, 1995.

[77] M. Th. van Genuchten. A closed-form for predicting the hydraulic conductivity of unsaturated soils. *Soil Sci. Soc. Am. Journal*, 44 :892–898, 1980.

[78] P. Weis and X. Leroy. *Le Langage Caml*. Dunod, 2e edition, 1999.

Vu : le Président

Vu : les suffragants

Vu et permis d'imprimer : le Vice-Président du Conseil Scientifique Chargé de la Recherche de l'Université Paris Dauphine

Résumé : Les questions de sûreté et d'incertitudes sont au centre des études de faisabilité pour un site de stockage souterrain de déchets nucléaires, en particulier l'évaluation des incertitudes sur les indicateurs de sûreté qui sont dues aux incertitudes sur les propriétés du sous-sol et des contaminants. L'approche globale par les méthodes probabilistes de type Monte Carlo fournit de bons résultats, mais elle demande un grand nombre de simulations. La méthode déterministe étudiée ici est complémentaire. Reposant sur la décomposition en valeurs singulières de la dérivée du modèle, elle ne donne qu'une information locale, mais elle est beaucoup moins coûteuse en temps de calcul.
Le modèle d'écoulement suit la loi de Darcy et le transport des radionucléides autour du site de stockage est modélisé par une équation de diffusion-convection linéaire. Différentiation à la main et différentiation automatique sont comparées sur ces modèles en mode direct et en mode adjoint. Une étude comparée des deux approches probabiliste et déterministe pour l'analyse de la sensibilité des flux de contaminants aux exutoires par rapport aux variations des paramètres d'entrée est menée sur des données réalistes fournies par l'ANDRA.
Des outils génériques d'analyse de sensibilité et de couplage de code sont développés en langage Caml. Ils permettent à l'utilisateur de ces plates-formes génériques de ne fournir que la partie spécifique de l'application dans le langage de son choix.
Une étude sur les écoulements diphasiques eau/air partiellement saturés en hydrogéologie porte sur les limitations des approximations de Richards et de la formulation en pression globale issue du domaine pétrolier.

Mots clés : Analyse de sensibilité, approche déterministe, décomposition en valeurs singulières, écoulement de Darcy, équations de convection-diffusion, éléments finis mixte hybrides, différentiation automatique, programmation fonctionnelle, plate-formes génériques, couplage de codes, stockage de déchets nucléaires

Abstract : The questions of safety and uncertainty are central to feasibility studies for an underground nuclear waste storage site, in particular the evaluation of uncertainties about safety indicators which are due to uncertainties concerning properties of the subsoil or of the contaminants. The global approach through probabilistic Monte Carlo methods gives good results, but it requires a large number of simulations. The deterministic method investigated here is complementary. Based on the Singular Value Decomposition of the derivative of the model, it gives only local information, but it is much less demanding in computing time.
The flow model follows Darcy's law and the transport of radionuclides around the storage site follows a linear convection-diffusion equation. Manual and automatic differentiation are compared for these models using direct and adjoint modes. A comparative study of both probabilistic and deterministic approaches for the sensitivity analysis of fluxes of contaminants through outlet channels with respect to variations of input parameters is carried out with realistic data provided by ANDRA.
Generic tools for sensitivity analysis and code coupling are developed in the Caml language. The user of these generic platforms has only to provide the specific part of the application in any language of his choice.
We also present a study about two-phase air/water partially saturated flows in hydrogeology concerning the limitations of the Richards approximation and of the global pressure formulation used in petroleum engineering.

Keywords : Sensitivity analysis, deterministic approach, singular value decomposition, Darcy flow, convection-diffusion equations, mixed hybrid finite elements, automatic differentiation, fonctional programing, generic platforms, codes coupling, nuclear waste storage