Adaptive finite volume method based on a posteriori error estimators for solving two-phase flow in porous media

Carole Widmer

▶ **To cite this version:**

Carole Widmer. Adaptive finite volume method based on a posteriori error estimators for solving two-phase flow in porous media. Numerical Analysis [math.NA]. Université Pierre et Marie Curie (Paris 6), 2013. English. NNT : . tel-01900433

HAL Id: tel-01900433
https://theses.hal.science/tel-01900433

Submitted on 22 Oct 2018

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 PIERRE ET MARIE CURIE
École Doctorale des Sciences Mathématiques de Paris Centre

Adaptive finite volume method based on a posteriori error estimators for solving two-phase flow in porous media

THÈSE

pour obtenir le grade de
Docteur en Mathématiques Appliquées

réalisée à
IFP ENERGIES NOUVELLES

et soutenue par

Carole Widmer

le 21/11/2013,
au Laboratoire Jacques-Louis Lions,
devant le jury composé de

Frédéric Hecht	Université Pierre et Marie Curie	Président
Jocelyne Erhel	Inria Rennes	Rapporteur
Andrea Bonito	Texas A&M University	Rapporteur
Éric Flauraud	IFP Energies nouvelles	Examinateur
Vivette Girault	Université Pierre et Marie Curie	Directeur de thèse
Martin Vohralík	Université Pierre et Marie Curie	Co-directeur
Quang Huy Tran	IFP Energies nouvelles	Co-encadrant

Contents

Résumé court

Cette thèse présente, au chapitre 2, le modèle compositionel de Darcy et des méthodes de discrétisation par Volume Finis utilisées par l'IFP Énergies nouvelles (IFPEn). Ce problème couple des équations aux dérivées partielles, correspondant aux bilans de quantité de matière et d'énergie, avec des contraintes algébriques assurant la conservation du volume poreux, la partition de l'unité pour les fractions molaires et l'équilibre chimique pour chaque composant. Dans le but d'assurer la cohérence avec les applications IFPEn, nous nous sommes basés sur une formulation reposant sur les bilans de quantité de matière pour chaque composant. La difficulté principale de ce modèle est liée au fait que le jeu d'inconnues varie en chaque point du domaine. La discrétisation du problème s'effectue par des méthodes de volumes finis avec décentrage amont des flux et discrétisation implicite en temps. Le chapitre 3 traite un cas plus simple, celui d'un écoulement diphasique immiscible. La performance du calcul numérique est fortement dépendante des méthodes de discrétisation et des algorithmes de résolution des systèmes linéaires et non linéaires. On trouve alors dans cette partie la mise en œuvre de stratégies de résolution basées sur des indicateurs d'erreur a posteriori. Le but principal ici est d'optimiser les critères d'arrêt des solveurs linéaires et non linéaires, tout en conservant la qualité des résultats numériques, en particuler la précision du déplacement de l'interface entre les deux fluides et de la quantité de mouvement à travers le domaine. Le chapitre 4 de cette thèse est consacrée à la mise en œuvre d'un prototype de résolution du modèle de Darcy.

Short summay

In Chapter 2, this thesis presents Darcy's compositional model and some discrete Finite Volume methods used by IFPEn. This problem couples partial differential equations, stating the balance of mass, momentum, and energy, with algebraic constraints enforcing conservation of volume in the pores, partition of unity of molar fractions, and chemical equilibrium of each component. In order to respect the approach of IFPEn's applications, we base this formulation on the balance of mass and momentum for each component. The main difficulty of this model arises from the fact that the set of unknowns varies at each point of the domain. The problem is discretized by FV methods with flux upwinding in space and backward Euler implicit discretization in time. Chapter 3 is devoted to the simpler case of immiscible two-phase flow. The performance of the numerical computation depends strongly on the choice of discretizations and of algorithms for solving the nonlinear and linear systems. This part describes the implementation of resolution strategies based on a posteriori error indicators. Its main object is the optimization of stopping criteria of the nonlinear and linear solvers that preserve the quality of the numerical output, in particular the accuracy of the displacement of the interface between the two phases and the accuracy of the momentum in the domain. Chapter 4 is devoted to the elaboration of a prototype that solves the main features of Darcy's compositional model.

Résumé étendu

Cette thèse est consacrée à la présentation et discrétisation du problème de Darcy multiphasique, ainsi qu'à la mise en œuvre de stratégies de résolution du problème de Darcy diphasique basées sur des indicateurs d'erreur a posteriori. Les applications ciblées sont le procédé avancé de récupération d'huile *Steam Assisted Gravity Drainage* (SAGD) et la séquestration géologique du CO_2, puisque ces problèmes sont modélisés de la même manière.

L'amélioration des procédés de récupération des huiles lourdes, c'est-à-dire, à densité et viscosité très élevées, est aujourd'hui un enjeu crucial de l'extension des réserves existantes. La mobilité des huiles lourdes étant insuffisante pour l'exploitation, il est nécessaire de mettre en place des systèmes actifs d'extraction. Le plus souvent, ces systèmes se basent sur l'injection d'un fluide chaud qui améliore la mobilité de l'huile favorisant ainsi sa migration vers les puits. Plus précisément, dans le cas du procédé SAGD, on injecte de la vapeur d'eau. Le but des simulations faites à l'IFPEn est, dans ce cas, d'estimer la quantité d'huile récupérée et d'optimiser l'injection de la vapeur d'eau. Afin d'améliorer la qualité des résultats numériques, il est important de suivre avec précision le déplacement de l'interface entre le fluide chaud et l'huile. La performance est, en outre, fortement dépendante des méthodes d'intégration en temps et de résolution des systèmes linéaires et non linéaires.

La résolution du front de phase qui pénètre dans le sol est importante pour la fiabilité de la simulation. De plus, les problèmes liés au coût de calcul sont des enjeux majeurs des simulations, d'où la nécessité d'envisager des stratégies pour optimiser l'utilisation des ressources de calcul. Enfin, on demande de certifier la solution numérique pour la gestion des risques, ce

qui demande de disposer d'une borne supérieure précise de l'erreur.

Cette thèse est composée de trois parties. Dans une première partie, nous décrivons d'abord les équations du problème compositionnel de Darcy, et ensuite, nous lui appliquons des méthodes de discrétisation par Volumes Finis (VF). Dans une seconde partie, nous étudions le cas plus simple des écoulements diphasiques, leurs discrétisations et leurs algorithmes de calculs gérés par des indicateurs d'erreur a posteriori. Des calculs numériques illustrent les résultats théoriques. Enfin, la troisième partie est consacrée à l'implémentation d'un prototype de calcul complet pour le diphasique, et tenant compte de l'aspect générique du modèle compositionnel.

Le chapitre 2 de cette thèse contient la formulation d'un modèle unifié pour les applications industrielles citées ci-dessus. Afin de disposer d'un large spectre de cas, nous avons choisi de travailler dans le cas le plus général, c'est-à-dire, celui où on fixe uniquement le nombre de composants, tandis que les phases et leur composition dépendent des conditions thermodynamiques ainsi que des fractions molaires globales. Le problème correspondant mélange des équations aux dérivées partielles (EDPs), correspondant aux bilans de quantité de matière et d'énergie, avec des contraintes algébriques assurant la conservation du volume poreux, la partition de l'unité pour les fractions molaires et l'équilibre chimique pour chaque composant. Dans le but d'assurer la cohérence avec les applications IFP Énergies nouvelles déjà existantes, nous nous sommes basés sur une formulation reposant sur les bilans de quantité de matière pour chaque composant ; une approche alternative basée sur les bilans de masse est proposée dans Chen, Huan et Ma [33, Chapitre 9]. La difficulté principale de ce modèle est liée au fait que le jeu d'inconnues varie en chaque point du domaine. La discrétisation du problème s'effectue par des méthodes de volumes finis (VF) avec décentrage amont des flux par phase et intégration implicite en temps. Une caractéristique importante de ce modèle est qu'il est possible de réduire le nombre d'inconnues du système global en exploitant les contraintes algébriques locales de chaque maille.

Afin de garantir la consistance de la discrétisation VF sur des maillages topologiquement Cartésien et en présence de tenseurs de perméabilité pleins, nous utilisons des méthodes multi-points (MPVF) selon les idées de Agélas, Di Pietro et Masson [8]. Les méthodes multi-points ont été introduites indépendemment par Aavatsmark, Barkve, Bøe et Mannseth [2, 3, 4, 5] et Edwards et Rogers [44, 45] dans le contexte de la simulation de réservoirs pétroliers. L'idée de base consiste à remplacer le flux numérique à deux points par une version pouvant dépendre des valeurs de la solution discrète dans d'autres mailles que celles qui partagent la face. La

dépendance est typiquement obtenue par des constructions locales garantissant la consistance pour des fonctions affines par morceaux, et dont le flux diffusif est continu à travers les interfaces du maillage. Plus précisément, la discrétisation ici proposée est basée sur la méthode récemment introduite et analysée par Agélas, Di Pietro et Droniou [7].

Le chapitre 3 est consacré au cas plus simple d'écoulements diphasiques immiscible, donc à deux phases et deux composants. Bien que plus simple, les équations de ce modèles sont cependant fortement non linéaires. Dans une première partie, et sous des hypothèses convenables sur les données et les choix d'espaces des inconnues, nous construisons une formulation variationnelle de ce problème, dont nous démontrons l'équivalence. Dans une seconde partie, nous discrétisons les équations de la formulation variationnelle par une méthode de VF à deux points avec décentrage des flux en amont et nous décrivons les équations de l'algorithme de Newton pour linéariser le système. A chaque boucle de Newton, nous prenons en compte un algorithme utilisé pour résoudre le système linéarisé, mais le choix de l'algorithme est libre. La troisième partie contient les résultats d'analyse a posteriori proprement dits. Le but des estimations d'erreur a posteriori est de donner des bornes de l'erreur entre l'approximation numérique et la solution exacte inconnue qui puissent être calculées en pratique à partir de la solution approchée. Ceci permet, en particulier, d'estimer la grandeur de l'erreur et de la localiser. La localisation de l'erreur est l'ingrédient clé des stratégies d'adaptation de maillage et de pas de temps, qui visent à réduire le coût de calcul en adaptant localement la taille des mailles en espace ou en temps à l'échelle du problème. Egalement, les indicateurs d'erreur a posteriori peuvent être utilisés dans le but de contrôler l'erreur et d'optimiser l'efficacité des algorithmes de résolution.

Dans notre cas, nous avons affaire à un problème non linéaire instationnaire discrétisé par une boucle sur les pas de temps et à chaque pas de temps, par des itérations de linéarisation de Newton, et à chaque itération de Newton, par des itérations d'un solveur algébrique pour résoudre les systèmes linéarisés. Le but des estimations d'erreurs proposées dans le chapitre 3 est de

1. Estimer séparément les différentes composantes de l'erreur.

2. Distinguer entre les composantes de l'erreur *substantielles*, c'est-à-dire les erreurs qui vont toujours être présentes (par exemple, les erreurs de discrétisation en espace et en temps), et les erreurs *subsidiaires*, qui peuvent être rendues très petites (erreurs des

solveurs non linéaires et linéaire).

3. Arrêter les différents algorithmes itératifs au moment où les erreurs subsidiaires correspondantes n'affectent plus l'erreur totale de façon significative (*critères d'arrêt*).

La technique de dérivation d'estimateurs d'erreur a posteriori est ici mise au point sur un problème diphasique, mais la même approche s'applique dans le cas plus général du modèle compositionnel. L'idée de base consiste à introduire une norme de résidu que l'on peut borner par des indicateurs calculables tout au long de l'algorithme, methode développée par exemple par Verfürth [70], dans le contexte des modèles considérés par Cancès, Pop et Vohralík [22] et par Vohralík et Wheeler [73]. Ceci demande, en particulier, d'introduire des reconstructions opportunes des champs issus de la discrétisation VF. Les différentes composantes d'erreur sont ensuite identifiées en écrivant le problème discret sous une forme algorithmique qui met en évidence les différentes boucles évoquées ci-dessus. Une telle approche a été récemment développées par El Alaoui, Ern et Vohralík [46], Ern et Vohralík [47, 48], Vohralík [71, 72] et Jiránek, Strakoš et Vohralík [53]. Ce travail a d'ores et déjà été initié par Di Pietro, Vohralík et Widmer [41].

Les indicateurs de l'erreur servent aussi à ajuster les paramètres du calcul, par raffinement/déraffinement du maillage et du pas de temps, de telle sorte que les erreur substantielles soient distribuées de façon équilibrée (*équilibrage des composantes d'erreur*). L'adaptation en espace ou en temps est décrit dans le chapitre 4 n'est pas l'objet principal de ce travail de thèse.

Dans le chapitre 4, nous présentons les outils développés pour la mise en œuvre du simulateur de Darcy compositionnel et des estimateurs d'erreur du chapitre 3. La discussion porte sur trois aspects qui ont présenté une certaine difficulté. Dans un premier temps, on se concentre sur un point important : les outils pour la gestion des inconnues. Du fait d'avoir un jeu d'inconnues différent dans chaque maille, ces outils sont au cœur du simulateur, et les inconnues doivent répondre à des impératifs de souplesse et d'efficacité. Le dernier point ici traité est relatif à la configuration d'un jeu de données pour le modèle. Ce travail vise, à terme, des applications dans deux contextes différents, d'où la nécessité de disposer d'outils souples pour la description du modèle. On se concentre, dans un deuxième temps, sur la gestion des lois physiques qui bénéficie d'une mise en œuvre permettant, dans une certaine mesure, de disposer d'outils de différentiation automatique. Ceci permet à la fois de réduire le coût de

maintenance et de modification du code, ainsi que de gérer de manière simple la croissance de la complexité. On développe ensuite le travail d'implémentation fait pour exécuter les estimateurs d'erreur a posteriori et leur mise en œuvre au sein du code. Finalement, on explique les outils de gestion du raffinement de maillage. Afin de simplifier la mise en œuvre, on a choisi de se concentrer sur des maillages raffinés de manière non conforme à partir d'une grille topologiquement Cartésienne. Des domaines non rectangulaires peuvent être gérés car la mise en œuvre distingue topologie et métrique, cette dernière pouvant être non affine.

CHAPITRE 1

Introduction

1.1 Contexte et motivations

Grâce aux techniques actuelles, l'exploitation d'un réservoir de pétrole est de l'ordre de 35% en moyenne. Il existe des procédés qui tentent d'augmenter la récupération des hydrocarbures fortement visqueux, dits « huiles lourdes ». Ces huiles lourdes sont les plus difficiles à extraire. Le réservoir, afin d'être exploité, est perforé à divers endroits, des puits y sont alors installés. Ces procédés emploient trois techniques :

- la méthode thermique, qui utilise l'injection de vapeur d'eau pour chauffer le pétrole, ce qui le fluidifie et facilite sa production ;
- la méthode chimique, qui utilise des viscosifiants et/ou des tensio-actifs ;
- l'injection de CO2, qui apparaît comme une voie permettant à la fois d'augmenter le rendement du gisement et de stocker le dioxyde de carbone dans le cadre de la lutte contre le changement climatique.

Ces trois méthodes comportent deux types de puits horizontaux :

- Les puits injecteurs (en amont), dans lesquels on injecte en continu un fluide sélectionné pour ses caractéristiques chimiques et physiques. Ce fluide aura pour but de pousser les

hydrocarbures ;

- Les puits producteurs (en aval), dans lesquels les hydrocarbures (en particulier les huiles lourdes), que l'on a fluidifiées, sont pompées jusqu'en surface.

La figure 1.1 présente un de ces procédés de manière schématique.

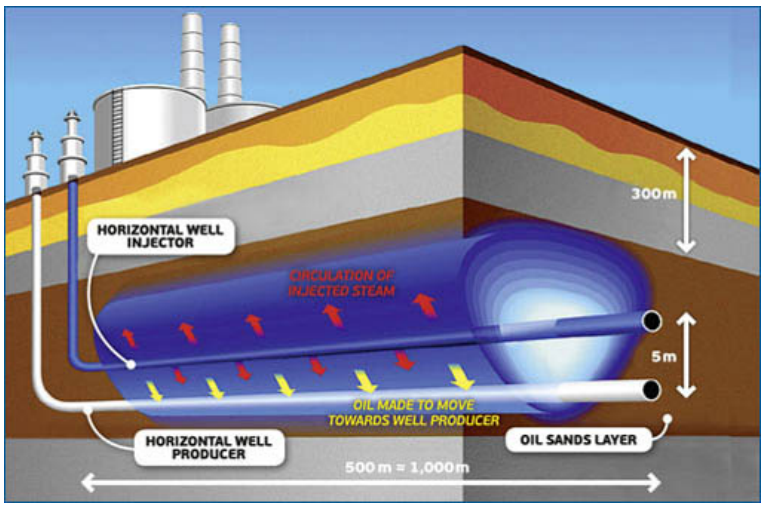

FIGURE 1.1 – Procédé de récupération assistée de pétrole ou Enhanced Oil Recovery (EOR).

Pour augmenter la récupération dans les gisements, il est essentiel d'améliorer la caractérisation des réservoirs. Une bonne connaissance des hétérogénéités sédimentaires et de leur distribution spatiale ainsi que des propriétés du réservoir permet d'optimiser le schéma d'implantation des puits producteurs et injecteurs, et la sélection des additifs chimiques à mettre en œuvre. Pour cela, des logiciels de simulation de réservoirs sont développés et la précision et la fiabilité de leurs résultats sont alors des facteurs clés pour augmenter la récupération. Cependant, les modélisations à l'échelle des réservoirs engendrent des coûts dûs à l'usage des super-calculateurs et, en particulier, aux temps (conséquent) de calcul.

Les industriels du pétrole cherchent donc à utiliser des logiciels de calcul à la fois fiables et optimisés.

1.2 État de l'art

La modélisation de réservoirs nécessite de prédire l'évolution, en milieu poreux, d'écoulements de plusieurs phases, chacune étant elle-même composée de plusieurs composants. On nomme ces écoulements : multi-phasiques et compositionnels (ou multi-composants). La conception de logiciels pour leur simulation requiert une discrétisation du domaine espace-temps et la mise en place de schémas numériques qui traitent des lois physiques de mécanique des fluides, de thermodynamique et qui gèrent les caractéristiques d'un milieu poreux. Le modèle de Darcy répond à ces attentes.

L'industrie pétrolière a recours à différentes méthodes de discrétisation en temps (complètement implicites ou implicites en temps et explicites en saturation). En espace, elle utilise des méthodes de volumes finis. Au début, les schémas étaient à deux points, mais maintenant, ce sont souvent des schémas multi-points. Ces méthodes tiennent leur noms de l'approche de discretisation des flux diffusifs qui se servent de deux, ou plusieurs pour les multi-points, valeurs discrètes aux centres des éléments voisins. Les méthodes multi-points ont été introduites dans les années 90 par Aavatsmark, Barkve, Bøe et Mannseth [4, 5], et par Edwards et Rogers [45]. Citons aussi le livre de Eymard, Gallouët et Herbin [50] qui présente un large éventail de ces méthodes. De plus, on peut citer les travaux sur grilles hexaédriques de Aavatsmark [1] et Aavatsmark, Eigestad, Klausen, Wheeler et Yotov [6]. Les travaux de thèse de Cao [23], mais aussi le livre de Chen, Huan et Ma [33], décrivent de manière approfondie l'état de l'art pour les formulations des écoulements polyphasiques et pour les méthodes de discrétisation de ces problèmes. Ces travaux tiennent compte aussi de l'impact des puits dans le domaine. Notons que les premiers travaux sur le sujet des puits ont été élaborés par Peaceman [61] en 1978.

A de très rares exceptions près, les méthodes numériques pour la résolution des systèmes approchés ne donnent évidemment pas la solution exacte, puisque celle-ci n'est pas discrète. Il est alors intéressant de se demander si la différence entre ces deux solutions (exactes et approchées) est quantifiable. Historiquement, les numériciens ont introduit des méthodes d'estimation d'erreur a priori. Ces méthodes permettent d'évaluer une borne de l'erreur avant la réalisation des calculs et cette borne de l'erreur fournit des grandeurs liées à la taille des mailles. Autrement dit, elles donnent un ordre de convergence assymptotique. Mais malheureusement, dans la plupart des cas, par exemple dès que le problème à résoudre est non linéaire, leurs constantes font intervenir des normes de la solution exacte, qui ne sont pas connues, et donc

en pratique incalculables.

Pour contrer cela, des méthodes d'estimation d'erreur a posteriori ont été développées ; ces méthodes bornent la différence entre la solution exacte et une solution approchée, grâce à des quantités relatives à la solution approchée. Elles sont donc évaluées une fois que la solution approchée est calculée. Ces méthodes d'analyse de l'erreur a posteriori ont été initiées, entre autres par Babuška et Rheinboldt [16], Ladevèze [55], Verfürth [69], Pousin et Rappaz [62], par Ainsworth et Babuška [9], Verfürth [70], Carstensen et Hu [24], Carstensen, Hu et Orlando [25] et Chaillou et Suri [27, 26].

Le développement de ces estimateurs d'erreur a posteriori permet donc d'évaluer la différence entre la solution approchée calculée et la solution exacte. Une application intéressante de l'analyse a posteriori de l'erreur est de permettre d'identifier différentes sources de l'erreur. Cette technique sert à mettre en place des méthodes de résolution adaptative. En premier lieu, on pense alors à des méthodes d'adaptation de maillage. En effet, on raffine les mailles là où l'erreur est la plus grande, et au contraire on déraffine les mailles là où l'erreur est faible. Ces idées sont développées, par exemple, par Braack et Ern dans [18] et dans les travaux de thèse de Mamaghani [56]. Dans un deuxième temps, en remarquant que tous les algorithmes de résolution d'un système discret ont recours à des méthodes itératives, ces estimateurs d'erreur vont permettre d'optimiser les critères d'arrêt utilisés. Ainsi, on peut espérer économiser un nombre conséquent d'itérations et a fortiori du temps de calcul. Ces méthodes ont été développées par exemple par Becker, Johnson et Rannacher [17], Arioli, Loghin et Wathen [14], Chaillou et Suri [27, 26] et Jiránek, Strakoš et Vohralík dans [53].

1.3 Objectif de la thèse

1.3.1 Chapitre 2 : Le modèle compositionnel de Darcy

Ce chapitre présente une formulation mathématique des écoulements multi-phasiques multi-compositionnels en milieu poreux. Les fluides considérés sont donc composés de plusieurs phases, chacune étant elle-même composée d'éléments chimiques appelés composants. Ces écoulements sont régits par le modèle de Darcy. L'un des enjeux de ces simulations réside dans l'influence du mouvement des phases qui permet à celles-ci d'apparaitre ou de disparaitre à tout instant et en tout point du domaine. On introduit donc la notion de contexte, qui permet d'associer à chaque point du domaine un indice, cet indice correspondant alors à l'ensemble

des phases présentent en ce point. Ainsi, la modélisation des phénomènes évoluent en chaque point du domaine en fonction de ce contexte. Le nombre d'équations et d'inconnues varient en fonction des phases présentes ou absentes. La gestion des inconnues en chaque point du domaine est difficile. Elle est traditionellement gérée par un algorithme Flash. Un exemple de cet algorithme est décrit dans l'Annexe A. Le modèle de Darcy est alors composé d'équations de conservation de la masse et de moment pour chaque composant, de conservation de volume pour chaque phase, de conservation de la quantité de matière dans chaque phase présente et d'équations d'équilibre des composants appartenant à plusieurs phases présentes. Ce modèle permet de modéliser les échanges thermiques. Dans ce cas, une équation supplémentaire est ajoutée : l'équation de conservation de l'énergie. Habituellement dans l'industrie petrolière, on utilise une formulation du type Coats [35], qui permet d'évaluer une pression de référence, les saturations de chaque phase, les fractions molaires pour chaque composant de chaque phase et le nombre de moles pour chaque composant absent du context. Le modèle continu est détaillé en section 2.1, on trouve la description des inconnues en 2.1.1, les propriétés du fluide et du milieux en 2.1.2, les équations en 2.1.3 et on y introduit la notion de contexte en 2.1.4. Afin de discrétiser ce modèle, on utilise des méthodes Volumes Finies (VF) multi-point, avec décentrage en amont des flux diffusifs. Ces méthodes de discrétisation sont décrites dans la section 2.2. On y détaille en particulier les même sujets que précédement mais dans des espaces discrets : la notion de contexte discret et les inconnues discrètes en 2.2.1, les équations discrètes en 2.2.2 et la mise en place du système à résoudre en 2.2.3. Le système ainsi assemblé est très complexe, non linéaire et de grande taille. La section 2.3 présente des stratégies mises en place pour résoudre ce problème approché.

1.3.2 Chapitre 3 : Estimateurs a posteriori pour des écoulements diphasiques

Le cas multiphasique étant très complexe, le chapitre 3 est consacré à un modèle restreint. En effet, on étudira dans ce chapitre, un écoulement diphasique immiscible (une phase aqueuse et une non-aqueuse), sans échange thermique. Le système d'équations et d'inconnues, bien que non linéaire, est alors considérablement simplifié. Les équations du modèle sont données à la section 3.1. Dans la section 3.2 on définit les espaces mathématiques nécessaires à une bonne formulation du problème. Ensuite dans les sous-sections 3.2.2 et 3.2.3 on construit deux formulations faibles du problème et on démontre leur équivalence avec le problème d'origine

moyennant des hypothèses de régularité sur les données et les espaces choisis pour les inconnues. La première formulation est plus naturelle, la seconde nous permettra d'établir notre analyse de l'erreur. Les sous-sections 3.3.1 et 3.3.2 décrivent les espaces discrets et les inconnues du problème qui serviront dans le reste du travail. Les équations discrètes obtenues grâce à des méthodes de discrétisation volume finie deux points avec décentrage en amont des flux diffusifs sont données en 3.3.3. Finalement, on présente dans la sous-section 3.3.4 des stratégies de résolutions du système discret. Dans le but de développer notre analyse de l'erreur a posteriori, il est nécessaire de post-traiter des quantités issues des simulations, voir section 3.4. Une fois ces reconstructions effectuées, on peut alors introduire une norme résiduelle de l'erreur. Cette quantité n'est pas calculable, car en général la solution exacte est inconnue. On introduit alors dans la sous-section 3.5 des estimateurs d'erreur calculables, qui majorent la norme résiduelle introduite plus haut. Historiquement, les estimations d'erreur a posteriori sont majoritairement utilisés pour améliorer l'adaptivité en temps ou en espace. Cependant, on a toujours besoin de solveur itératifs pour résoudre les problèmes issus de la discrétisation car ils sont de très grande taille et très souvent non linéaires. Actuellement, les méthodes permettant de définir les critères d'arrêt de ces solvers sont à choisir au mieux. La finalité de cette thèse est d'utiliser ces indicateurs a posteriori de l'erreur pour déterminer avec efficacités les critères d'arrêt. L'approche développée dans ce travail de thèse est basée sur l'utilisation de ces indicateurs a posteriori de l'erreur.

1.3.3 Chapitre 4 : Mise en œuvre informatique

Le travail fourni dans cette thèse a débuté par la mise en place d'un prototype de simulation, appelé *Cogito*, programmé en C++ orienté objet. Il a fallut créer un code capable de simuler des écoulements en milieu poreux, puis intégrer les outils nécessaire au développement de l'analyse a posteriori. Ce chapitre présente donc quelques contributions que j'ai mise en œuvre, leur structure et leur utilisation. Tout au long de ce chapitre des morceaux de codes sont présentés, notons que ce sont des morceaux simplifiés permettant seulement d'offrir une vision plus pratique des éléments implémentés. Les chapitres précédents ont relevé que l'une des difficultés pour la résolution du système discret de Darcy est l'influence de l'apparition ou disparition des phases sur ce système d'inconnues et d'équations. La section 4.2 présente comment le problème a été traité via un manager d'inconnues qui gère en fonction d'un fluide considéré les différents contextes en 4.2.1, qui génère les ensembles propres au modèle (e.g.

ensemble de composants dépendant d'un contexte, ensemble de phases contenant un composant présent dans le contexte, ...), ces ensembles sont traités en 4.2.2. Les sections 4.2.3 et 4.2.4 traitent quandt à elles de la génération des ensembles d'inconnues et de la selection des inconnues primaires et secondaires. La section 4.3 présente l'outil de gestion des propriétés physiques du fluide et du milieu, détaillées en section 2.1.2. Cet outil permet de considérer ces fonctions, comme des fonctions analytiques, les lois et leurs dérivées sont en effet calculées analytiquement. Grâce à tous ces développement présentés ci-dessus, il est alors possible d'intégrer les outils nécessaires à l'analyse d'erreur a posteriori ; cela est expliqué en section 4.4. On détaille l'implémentation des post-traitements nécessaires en 4.4.1 et en 4.4.2. Finalement, la section 4.4.3 a pour but de recenser les avantages que présente l'implémentation des estimateurs d'erreur a posteriori.

CHAPTER 2

The compositional Darcy model

In this chapter, we propose first to define a mathematical formulation of the compositional flow. Different types of formulations are possible, and we choose to follow the Coats formulation introduced by Coats in [35] and by Coats, Thomas, and Pierson in [36]. Its particularity is to consider the system in two types of equations, the conservative equations and the local closure equations. A complete state of the art can be found in the thesis of Cao [23]. We then discuss the numerical discretization of the model, following in particular the ideas of Chen, Huan, and Ma [33].

The compositional Darcy model states the basic principles of conservation of mass, momentum, and energy for a fluid flowing through a porous medium. We assume here that the fluid is composed of different *components* present in different *phases*. The term phase stands for a matter that has a homogeneous physical state, whereas a component is typically a chemical species. Here, we focus mostly on presenting the model in an abstract form. Particular situations of interest are presented as special cases. The material is organized as follows. In Section 2.1, we describe the continuous model, state the equations, and present the unknowns of the model. In Section 2.2, we present the discretization of the model based on a multi-point finite volume scheme with upwinding. The Section 2.3 presents the solution strategy. Finally, Section 2.4 shows the results of several numerical experiments.

2.1 The continuous setting

A mathematical model consists of a set of equations that describe a given physical phenomena. In petroleum industry we need to describe the flow of fluids in underground petroleum reservoirs, which are large geological domains containing hydrocarbons. They are porous media, but they are not vacuum bags filled with oil. Rather, they are comparable to a solid sponge: A rock full of pores filled with oil.

Let $\Omega \subset \mathbb{R}^d$, $d \geq 2$, be a bounded connected polygonal domain and let $t_{\mathrm{F}} > 0$. Here Ω represents the reservoir, while t_{F} is the simulation time. Our goal is to model the flow of the reservoir fluid through Ω in the time interval $(0, t_{\mathrm{F}})$. For this, we need to describe a fluid composed of $N_{\mathcal{P}}$ phases, phases being composed of $N_{\mathcal{C}}$ components ($N_{\mathcal{P}} < +\infty$ and $N_{\mathcal{C}} < +\infty$). We denote by $\mathcal{P} = \{p\}$ and $\mathcal{C} = \{c\}$ respectively the set of phases and components. In a general setting, a phase can contain only a subset of the components. For positive integers m and n, let $\mathbb{R}^{m,n}$ denote the set of real matrices with m lines and n columns. A synthetic description of the fluid system can be obtained using the *component-phase matrix*

$$\mathbf{M} = [m_{c,p}]_{c \in \mathcal{C},\, p \in \mathcal{P}} \in \mathbb{R}^{N_{\mathcal{C}}, N_{\mathcal{P}}}, \text{ where all entries belongs to } \{0, 1\},$$

such that, for all $c \in \mathcal{C}$ and all $p \in \mathcal{P}$,

$$m_{c,p} = \begin{cases} 1 & \text{if the component } c \text{ is contained in the phase } p, \\ 0 & \text{otherwise,} \end{cases} \tag{2.1.1}$$

$$\mathbf{M} = \begin{array}{c} \\ \\ \\ \mathcal{P}_c \\ \\ \\ \end{array} \begin{array}{c} \\ \\ \mathcal{C}_p \\ \\ \\ \\ \end{array} \begin{pmatrix} 1 & \dots & 0 & \dots & 1 \\ \vdots & \dots & \vdots & \dots & \vdots \\ 0 & \dots & 1 & \dots & 1 \\ \vdots & \dots & \vdots & \dots & \vdots \\ 1 & \dots & 1 & \dots & 0 \end{pmatrix}.$$

This matrix gives different sets:

- The set of components constituting the phase p:

$$\mathcal{C}_p = \{c \in \mathcal{C} \,|\, m_{c,p} = 1\}.$$

  The non-zero elements of column p represent the set of components composing the phase p.

- The set of phases containing the component c:

$$\mathcal{P}_c = \{p \in \mathcal{P} \,|\, m_{c,p} = 1\}.$$

The non-zero elements of line c represent the set of phases containing the component c.

Examples 2.7 and 2.8, in Section 2.1.5 below, illustrate the component-phase matrix two different particular situations. The description of the system is completed by a boolean *thermal index* I_θ:

- $I_\theta = 0$ in the isothermal case. This means that we do not consider the temperature in the model;
- $I_\theta = 1$ in the thermal case, in order to describe the temperature evolution in the domain. Therefore an additional partial differential equation (PDE) is present in the model.

The formulation used throughout this work relies on a system of equations which depends on the phases that are present in a space location at a given time. Indeed, if two phases flow side by side, one pushes the other and the fluid moves and evolves in time through the domain. Thus, if one phase pushes the other one, the second phase can disappear in one place and appear in another. The notion of *context* is introduced to describe this phenomenon.

Definition 2.1. *A context represents a set of phases that are present at a given time and position.*

Note that a context can only take a finite number of configurations. More specifically, since phases can either be present or not independently of each other, we have $N_\mathcal{K} = 2^{N_\mathcal{P}}$ possible contexts collected in the set $\mathcal{K} = \{k\}$.

Definition 2.2. *The context in which all phases are present is called the* reference context *and denoted by the symbol* k_{ref}.

2.1.1 Unknowns

Let $\Sigma := \Omega \times (0, t_\text{F})$ denote the space-time domain. The appearance or disappearance of phases in different points of time or space causes the main difficulty of the model. To account for this difficulty, we introduce the field $\mathfrak{K}$. Then at a given point of Σ, the field $\mathfrak{K}$:

$$\mathfrak{K} : \Sigma \to \mathcal{K} \tag{2.1.2}$$

gives the local context $k \in \mathcal{K}$; note that $\mathfrak{K}(\Sigma) \subset \mathcal{K}$. This mapping is itself an unknown of the problem and its determination requires a special computation. The flash algorithm described in Appendix A is devoted to the computation of the local context.

In what follows we describe a systematic way for stating the equations of mass, momentum and energy conservation.

Definition 2.3. *For a given context $k \in \mathcal{K}$, we let $\mathcal{P}_k$ denote the set of phases present in the context k. Similarly, $\mathcal{C}_k$ denotes the set of components present in the context k.*

Remark 2.4 (Indices of subsets)**.** To avoid the multiplicity of indices, we use in the same position the letter p, c, and k in the notation of the subsets of $\mathcal{P}$ and $\mathcal{C}$, but their meanings are clear and there can be no confusion.

Consider the following unknowns:

$$\mathcal{U}_k := \left\{ P, (\theta), \{S_p\}_{p\in\mathcal{P}_k}, \{C_{p,c}\}_{p\in\mathcal{P}_k, c\in\mathcal{C}_p}, \{n_c\}_{c\in\overline{\mathcal{C}_k}} \right\}, \tag{2.1.3}$$

where:

- P denotes the reference pressure;
- θ is the temperature, only present when $I_\theta = 1$;
- $\{S_p\}_{p\in\mathcal{P}_k}$ is the set of saturations for the phases p present in the context k;
- $\{C_{p,c}\}_{p\in\mathcal{P}_k, c\in\mathcal{C}_p}$ is the set of molar fractions for the components of each present phase;
- $\{n_c\}_{c\in\overline{\mathcal{C}_k}}$ contains the numbers of moles for the components absent from the context, which are collected in the set

$$\overline{\mathcal{C}_k} := \mathcal{C} \setminus \mathcal{C}_k.$$

These unknowns are considered separately and are described later.

For $p \in \mathcal{P}$, the *phase pressures* P_p are expressed in terms of the unknowns $\mathcal{U}_k$ as follows:

$$P_p(P, S_p) := P + P_{c_p}(S_p), \tag{2.1.4}$$

where $P_{c_p}(S_p)$ is the *capillary pressure* function whose dependency is made precise in Section 2.1.2.2.

The unknowns P, θ if $I_\theta = 1$, $\{S_p\}_{p\in\mathcal{P}_k}$, and $\{C_{p,c}\}_{p\in\mathcal{P}_k, c\in\mathcal{C}_p}$ are associated to non-trivial PDEs and they are collectively referred to as P(θ)CS unknowns. For the sake of conciseness, we introduce the following notation for the vectors of saturations and phase compositions: For all $k \in \mathcal{K}$,

$$\mathbf{S}^k := (S_p)_{p\in\mathcal{P}_k} \text{ and } \mathbf{C}_p^k := (C_{p,c})_{c\in\mathcal{C}_p} \text{ for all } p \in \mathcal{P}_k.$$

When $k = k_{\text{ref}}$, $\mathcal{U}_{k_{\text{ref}}}$ only contains P(θ)CS unknowns. In this case we drop the index and simply write

$$\mathcal{U} = \left\{P, (\theta), \{S_p\}_{p\in\mathcal{P}}, \{C_{p,c}\}_{p\in\mathcal{P}, c\in\mathcal{C}_p}\right\}.$$

Similarly, we let $\mathbf{S} := \mathbf{S}^{k_{\text{ref}}}$ and $\mathbf{C}_p := \mathbf{C}_p^{k_{\text{ref}}}$ for all $p \in \mathcal{P}$.

For each point X in Σ, the local context $\mathfrak{K}(X)$ is a subset of $\mathcal{K}$. We denote by a subscript $\mathfrak{K}$ all quantities related to the mapping $\mathfrak{K}$ defined in (2.1.2), such as

$$\mathcal{C}_{\mathfrak{K}} = \{c \in \mathcal{C}_k | k = \mathfrak{K}(X),\ X \in \Sigma\},$$

and

$$\mathcal{P}_{\mathfrak{K}} = \{p \in \mathcal{P}_k | k = \mathfrak{K}(X),\ X \in \Sigma\}.$$

Remark 2.5 (Empty context)**.** The empty context $\emptyset \in \mathcal{K}$ may be relevant in the thermal case $I_\theta = 1$. In this case the set of unknowns only contains the temperature, i.e., $\mathcal{U} = \{\theta\}$, and we solve the problem for the temperature field only.

We conclude this section on the unknowns by pinpointing that other choices are possible for the unknowns. These are not detailed herein; see, e.g., the work of Chen, Huan, and Ma [33, Chapter 9].

2.1.2 Fluid and rock properties

Before introducing the model, we need to present the different fluid and rock properties used to describe the present process. They are simply defined as fields over Σ and only the dependence on the unknowns is highlighted. For a property ℓ, let us denote by $[\ell]$ its dimension in the international system (IS) units. For example, if L is a length, its unit is the meter and we denote it by $[L] = m$. The temperature unknown θ, which is only present in the thermal cases, is written without parentheses in quantities that are intrinsic to the thermal cases and between parentheses in quantities that are common to both cases.

2.1.2.1 Medium properties

For the porous medium (the rock), we consider the following properties:

- The *porosity* ϕ of the porous medium is a dimensionless function satisfying $0 \leq \phi \leq 1$. It is the fraction of the volume of void over the total volume; It also gives the fraction of the volume available for the fluid. In this work, it is treated as a constant in space and in time.

- The *rock internal energy* $e_{\mathrm{r}}(P, \theta, \mathbf{C}_p)$ is a state function of the system. The units are $[e_{\mathrm{r}}] = \mathrm{kg} \cdot \mathrm{m}^2 \cdot \mathrm{s}^{-2}$.

- The *rock molar density* ζ_{r} represents the number of moles per unit volume and $[\zeta_{\mathrm{r}}] = \mathrm{mol} \cdot \mathrm{m}^{-3}$.

- The *thermal conductivity* λ is the property of a material describing its ability to conduct heat, and $[\lambda] = \mathrm{kg} \cdot \mathrm{m} \cdot \mathrm{s}^{-3} \cdot \mathrm{K}^{-1}$.

For all phases $p \in \mathcal{P}$ we introduce the following fields:

- The *absolute permeability tensor* $\mathbf{K}$ measures the permeability, or ability of the rock to transmit fluids. The usual unit is the Darcy (D) and the IS units give $[\mathbf{K}] = \mathrm{m}^2$.

- The *relative permeability* $k_{\mathrm{r}_p}(S_p)$ is a dimensionless number. It is the ratio of the effective permeability of the phase to the absolute permeability. The effective permeability represents the ability of a fluid to flow through a rock when another fluid is present in the pore space. The relative permeability k_{r_p} indicates the tendency of phase p to wet the porous medium.

2.1.2.2 The capillary pressure

For all phases $p \in \mathcal{P}$, the *capillary pressure* $P_{\mathrm{c}_p}(S_p)$ is specific to multiphase flows. It represents the difference in pressure between two fluids on either side of an interface. Indeed, the pressure in a wetting fluid is smaller than that in a non-wetting fluid. This difference in pressure, given by the capillary pressure, accounts for the curvature on the interface and the surface tension of the fluids at the interface between the two phases. Pressures are traditionally expressed in Pa. In IS units we have $[P_{\mathrm{c}_p}] = \mathrm{kg} \cdot \mathrm{m}^{-1} \cdot \mathrm{s}^{-2}$.

2.1.2.3 Thermodynamic properties

For all phases $p \in \mathcal{P}$ we introduce the following fields, where θ is in absolute degrees:

- The *molar density* $\zeta_p(P,(\theta),\mathbf{C}_p)$ represents the number of moles per unit volume and $[\zeta_p] = \text{mol} \cdot \text{m}^{-3}$.
- The *mass density* $\rho_p(P,(\theta),\mathbf{C}_p)$ represents the mass per unit volume and is expressed by $[\rho_p] = \text{kg} \cdot \text{m}^{-3}$.
- The *viscosity* $\mu_p(P,(\theta),\mathbf{C}_p)$ measures the resistance of a fluid to flow and $[\mu_p] = \text{kg} \cdot \text{m}^{-1} \cdot \text{s}^{-1}$.
- The *fugacity* $f_{c,p}(P,(\theta),\mathbf{C}_p)$ depends on a component c and on the set of components $\mathbf{C}_p$. It expresses a chemical equilibrium and represents the tendency of a component to escape. The fugacity is often expressed in Pa, the IS gives $[f_{c,p}] = \text{kg} \cdot \text{m}^{-1} \cdot \text{s}^{-2}$.
- The *enthalpy* $h_p(P,\theta,\mathbf{C}_p)$ is a thermodynamic potential; it is a measure of the total energy of a thermodynamic system and is expressed by $[h_p] = \text{kg} \cdot \text{m}^{-2} \cdot \text{s}^{-2} \cdot \text{mol}^{-1}$.
- The *phase internal energy* $e_p(P,\theta,\mathbf{C}_p)$ is a state function of the system. This function is only present in the thermal context. We express it by $[e_p] = \text{kg} \cdot \text{m}^{-2} \cdot \text{s}^{-2} \cdot \text{mol}^{-1}$.

2.1.3 Equations

We have seen that the set of unknowns depends on the context and the context depends on the position and on time. We recall that the field $\mathfrak{K}$ is itself an implicit function of the unknowns at a given location in Σ; hence it is a variable of the problem.

In order to have a physical model of the flow in the reservoir we must consider different types of equations expressing

- the mass conservation of the components,
- the momentum conservation of the components,
- the volume conservation,

- the conservation of the quantity of the matter in each phase,
- the equilibria of the components for those components that are present in more than one phase.

For all $c \in \mathcal{C}$, we let $q_c : \Sigma \to \mathbb{R}$ denote a source field. Let us recall

$$\forall c \in \mathcal{C}, \qquad \mathcal{P}_c := \{p \in \mathcal{P} \,|\, m_{c,p} = 1\}.$$

In the non-thermal case $I_\theta = 0$, we consider the following problem:

2.1.3.1 The partial differential equations

- Conservation of mass and momentum:

$$\partial_t n_c + \sum_{p \in \mathcal{P}_{\mathfrak{K}} \cap \mathcal{P}_c} \operatorname{div}\left(\frac{\zeta_p(P,(\theta),\mathbf{C}_p) k_{\mathrm{r}_p}(S_p)}{\mu_p(P,(\theta),\mathbf{C}_p)} C_{p,c} \overrightarrow{\mathbf{v}}_p(P,(\theta),S_p,\mathbf{C}_p)\right) = q_c, \qquad \forall c \in \mathcal{C}_{\mathfrak{K}}, \tag{2.1.5a}$$

$$\partial_t n_c = q_c, \qquad \forall c \in \overline{\mathcal{C}_{\mathfrak{K}}}. \tag{2.1.5b}$$

2.1.3.2 The closure equations

- The volume conservation:

$$\sum_{p \in \mathcal{P}_{\mathfrak{K}}} S_p = 1. \tag{2.1.6a}$$

- The conservation of the quantity of the matter in each phase:

$$\sum_{c \in \mathcal{C}_p} C_{p,c} = 1, \qquad \forall p \in \mathcal{P}_{\mathfrak{K}}. \tag{2.1.6b}$$

- The equilibria of the components present in different phases:

$$f_{c,p_1}(P,(\theta),\mathbf{C}_{p_1}) = f_{c,p_2}(P,(\theta),\mathbf{C}_{p_2}), \qquad \forall c \in \mathcal{C}_{\mathfrak{K}}, \forall p_1 \in \mathcal{P}_{\mathfrak{K}} \cap \mathcal{P}_c, p_2 \in \mathcal{P}_{\mathfrak{K}},\ p_1 \neq p_2. \tag{2.1.6c}$$

The definitions of n_c and $\overrightarrow{\mathbf{v}}_p$ are in order.

2.1.3.3 Mole number

Equations (2.1.5a)–(2.1.5b) are $N_{\mathcal{C}}$ PDEs linking the variation in time of the number of moles of a component $c \in \mathcal{C}$ at a given point in Σ with the flow of the phases containing the component c and the source term q_c. In particular, for all $k \in \mathcal{K}$, the number of moles n_c is an unknown for all $c \in \overline{\mathcal{C}_k}$ whereas, for all $c \in \mathcal{C}_k$ it can be inferred from the definition

$$n_c = \phi \sum_{p \in \mathcal{P}_k \cap \mathcal{P}_c} \zeta_p(P,(\theta),\mathbf{C}_p) S_p C_{p,c}. \tag{2.1.7}$$

2.1.3.4 Darcy velocity

In most formulations of Darcy's law, the phase mobility $\frac{k_{r_p}}{\mu_p}$ is included in the definition of the Darcy velocity. However, here we prefer to dissociate the mobility from the velocity and write it explicitly as a factor in the balance of mass and momentum. Therefore, for all $p \in \mathcal{P}$, we choose to define the average phase velocity by:

$$\overrightarrow{\mathbf{v}}_p(P,(\theta),S_p,\mathbf{C}_p) = -\mathbf{K}\left(\nabla P_p(P,S_p) - \rho_p(P,(\theta),\mathbf{C}_p)\mathbf{g}\right), \tag{2.1.8}$$

where $\mathbf{g}$ denotes the downward-oriented gravity acceleration and the phase pressure P_p is given by (2.1.4). In the physical situation considered here, the permeability tensor $\mathbf{K}$ is a diagonal matrix:

$$\mathbf{K} = \begin{bmatrix} \kappa_{11} & & \\ & \kappa_{22} & \\ & & \kappa_{33} \end{bmatrix}. \tag{2.1.9}$$

If $\kappa_{11} = \kappa_{22} = \kappa_{33}$, i.e. $\mathbf{K} = \kappa\,\mathbf{Id}$, the porous medium is called *isotropic*, otherwise, it is *anisotropic*. If $\mathbf{K}$ is constant over all Ω it is called *homogeneous*, otherwise, it is *heterogeneous*. To simplify the compositional model, we shall consider isotropic media. In contrast, we will study two-phase flows in anisotropic media, see Chapter 3, below.

Remark 2.6 (Absent components)**.** Observe that (2.1.5a)–(2.1.5b) could be alternatively written as

$$\partial_t n_c + \sum_{p \in \mathcal{P}_{\mathfrak{K}} \cap \mathcal{P}_c} \operatorname{div}\left(\frac{\zeta_p(P,(\theta),\mathbf{C}_p) k_{r_p}(S_p)}{\mu_p(P,(\theta),\mathbf{C}_p)} C_{p,c}\, \overrightarrow{\mathbf{v}}_p(P,(\theta),S_p,\mathbf{C}_p)\right) = q_c, \qquad \forall c \in \mathcal{C},$$

since $\mathcal{P}_{\mathfrak{K}} \cap \mathcal{P}_c = \emptyset$ for all $c \in \overline{\mathcal{C}_{\mathfrak{K}}}$. The formulation (2.1.5b) makes it clear that the only variation in the number of moles of components absent from the local context is due to source terms or a change in the local context.

2.1.3.5 Thermal case

In the thermal case $I_\theta = 1$, the system (2.1.5) is complemented by the energy conservation equation,

$$\partial_t e + \operatorname{div}\left(\sum_{p\in\mathcal{P}_{\mathfrak{K}}} \frac{\zeta_p(P,\theta,\mathbf{C}_p)k_{\mathrm{r}_p}(S_p)}{\mu_p(P,\theta,\mathbf{C}_p)} h_p(P,\theta,\mathbf{C}_p)\overrightarrow{\mathbf{v}}_p(P,\theta,S_p,\mathbf{C}_p) - \lambda\nabla\theta\right) = Q, \qquad (2.1.10)$$

where Q represents a thermal source term and the molar energy is given by

$$e = \phi\sum_{p\in\mathcal{P}_{\mathfrak{K}}} \zeta_p(P,\theta,\mathbf{C}_p)e_p(P,\theta,\mathbf{C}_p)S_p + (1-\phi)\zeta_{\mathrm{r}}e_{\mathrm{r}}(P,\theta,\mathbf{C}_p).$$

2.1.3.6 Boundary conditions

To close problem (2.1.5)–(2.1.8), we have to prescribe boundary conditions on $\partial\Omega\times(0,t_{\mathrm{F}})$. No-flow boundary conditions are prescribed for all phases, i.e. we choose:

$$\forall p\in\mathcal{P}, \qquad \overrightarrow{\mathbf{v}}_p\cdot\mathbf{n} = 0 \qquad \text{on } \partial\Omega\times(0,t_{\mathrm{F}}), \qquad (2.1.11)$$

where $\mathbf{n}$ is a unit exterior normal of Ω.

Moreover, for thermal problems, we assume fixed temperature boundary conditions such that

$$\theta = \overline{\theta} \qquad \text{on } \partial\Omega\times(0,t_{\mathrm{F}}), \qquad (2.1.12)$$

with $\overline{\theta}$ a given prescribed temperature.

2.1.3.7 Well model

Let us recall that the compositional formulation is established to model oil extraction processes used in the petroleum industry, such as the Steam Assisted Gravity Drainage (SAGD) process, and introduced in the 90th by Butler in [19], described in Section 1.1, see Example 2.8 below. This process contains a set of wells $\mathcal{W} = \{W\}$, some wells produce oil and belong to the class of *production* wells. In other wells, fluids (steam, water) are injected into the domain, and they belong to the class of *injection* wells.

It is assumed that the characterization of a given well is fixed, i.e., it cannot change during a simulation. Exchanges between a well and the domain are carried through several interior perforations of the well. The perforations allow the fluid to flow in or out of the domain

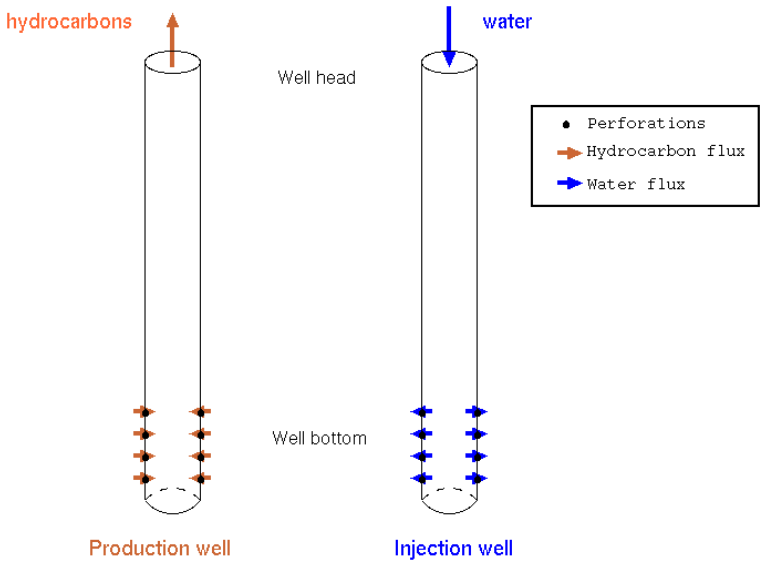

Figure 2.1: Schematic picture of production and injection wells

depending on whether the well is a production or an injection well, see Figure 2.1. Let us formulate fluxes flowing through the wells.

The effects of wells are expressed by adding the mass production or injection at the wells to the source terms $\{q_c\}_{c\in\mathcal{C}}$ and Q, in equations (2.1.5a)–(2.1.5b) and (2.1.10) respectively. The derivation of well flow equations relies on the basic assumption that the flow is radial in a neighborhood of the well. This leads to, for all $c \in \mathcal{C}_{\mathfrak{K}}$,

$$\partial_t n_c + \sum_{p\in\mathcal{P}_{\mathfrak{K}}\cap\mathcal{P}_c} \operatorname{div}\left(\frac{\zeta_p(P,(\theta),\mathbf{C}_p)k_{\mathrm{r}_p}(S_p)}{\mu_p(P,(\theta),\mathbf{C}_p)} C_{p,c}\,\vec{\mathbf{v}}_p(P,(\theta),S_p,\mathbf{C}_p)\right) = q_c + \sum_{W\in\mathcal{W}}\sum_{p\in\mathcal{P}_{k_W}\cap\mathcal{P}_c} \delta_W q^W_{p,c}, \tag{2.1.13}$$

where for a well $W \in \mathcal{W}$,

- δ_W is the characteristic function of W

$$\delta_W(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x}\in W, \\ 0 & \text{otherwise;} \end{cases} \tag{2.1.14}$$

- $q^W_{p,c}$ is the mass production or injection at this well;

- k_W is the local context of W;

- $\mathcal{P}_{k_W}$ is the set of phases in k_W.

We express $q_{p,c}^W$ as follows

$$q_{p,c}^W := \operatorname{div}\left(\frac{\zeta_p(P,(\theta),\mathbf{C}_p)k_{\mathrm{r}_p}(S_p)}{\mu_p(P,(\theta),\mathbf{C}_p)}C_{p,c}\overrightarrow{\mathbf{w}}_p^W(P,(\theta),S_p,\mathbf{C}_p)\right),$$

where the phase velocity $\overrightarrow{\mathbf{w}}_p^W$ in the well W is given by

$$\overrightarrow{\mathbf{w}}_p^W(P,(\theta),S_p,\mathbf{C}_p) := \tau^W\left(\nabla P_p(P,S) - \rho_p(P,(\theta),\mathbf{C}_p)\mathbf{g}\right).$$

The well index, also called *production index*, τ^W, is given by the Peaceman Formula, see [61]. Its discrete version is written in (2.2.26) below.

If $I_\theta = 1$, the source term of the energy conservation equation (2.1.10) is also complemented by a sum over the wells:

$$\begin{aligned}\partial_t e + \operatorname{div}\Big(\sum_{p\in\mathcal{P}_{\mathfrak{K}}}\frac{\zeta_p(P,\theta,\mathbf{C}_p)k_{\mathrm{r}_p}(S_p)}{\mu_p(P,\theta,\mathbf{C}_p)}h_p(P,\theta,\mathbf{C}_p)\overrightarrow{\mathbf{v}}_p(P,\theta,S_p,\mathbf{C}_p) - \lambda\nabla\theta\Big)&\\ = Q + \sum_{W\in\mathcal{W}}\delta_W Q_W,&\end{aligned} \tag{2.1.15}$$

where Q_W represents the energy produced by the wells; Details are presented in [33]. The above fluxes are discretized in Section 2.2.2.4.

2.1.3.8 Number of equations vs. number of unknowns

The mathematical analysis of the above problem is outside of the scope of this work. But at least, we can check that it is a square system of equations: For a given context $k \in \mathcal{K}$, the number of equations in (2.1.5)–(2.1.6) and in (2.1.10) matches the number of unknowns $\mathcal{U}_k$ (with $\mathcal{U}_k$ defined by (2.1.3)). Indeed, equations (2.1.5a)–(2.1.5b) and (2.1.10) are $N_\mathcal{C} + I_\theta$ PDEs; equations (2.1.6a) and (2.1.6b) add respectively 1 and $N_{\mathcal{P}_k}$ constraints; finally, the fugacity equilibria equations (2.1.6c) consist of

$$\sum_{c\in\mathcal{C}_k}(N_{\mathcal{P}_c} - 1) = \sum_{c\in\mathcal{C}_k}N_{\mathcal{P}_c} - N_{\mathcal{C}_k} = \sum_{p\in\mathcal{P}_k}N_{\mathcal{C}_p} - N_{\mathcal{C}_k}$$

equations.

As a consequence, for a given context $k \in \mathcal{K}$, the number of equations is

$$N_\mathcal{C} + I_\theta + 1 + N_{\mathcal{P}_k} + \sum_{p\in\mathcal{P}_k}N_{\mathcal{C}_p} - N_{\mathcal{C}_k} = 1 + I_\theta + N_{\mathcal{P}_k} + \sum_{p\in\mathcal{P}_k}N_{\mathcal{C}_p} + N_{\overline{\mathcal{C}_k}} = \operatorname{card}(\mathcal{U}_k).$$

2.1.4 The context

As we have already mentioned, the field $\mathfrak{K}$ is itself a function of the local unknowns. More specifically, we introduce the total molar fraction for all $c \in \mathcal{C}$:

$$Z_c = \frac{\sum_{p\in\mathcal{P}_\mathfrak{K}\cap\mathcal{P}_c} \zeta_p(P,(\theta),\mathbf{C}_p) S_p C_{p,c} + \sum_{c'\in\overline{\mathcal{C}_\mathfrak{K}}\cap\{c\}} n_{c'}}{\sum_{c'\in\mathcal{C}} \sum_{p\in\mathcal{P}_\mathfrak{K}\cap\mathcal{P}_{c'}} \zeta_p(P,(\theta),\mathbf{C}_p) S_p C_{p,c} + \sum_{c'\in\overline{\mathcal{C}_\mathfrak{K}}} n_{c'}}, \tag{2.1.16}$$

collected in the vector $\mathbf{Z} = \{Z_c\}_{c\in\mathcal{C}}$, with the condition $\sum_{c\in\mathcal{C}} Z_c = 1$. Then there holds:

$$\mathfrak{K} = \text{Flash}(P, \mathbf{Z}) \qquad \text{in } \Omega, \tag{2.1.17}$$

where Flash is an algorithm used in multiphase flow simulations to determine the local contexts; see Appendix A to find more details on the algorithm.

2.1.5 Examples

Example 2.7 (Immiscible isothermal two-phase flow)**.** A first example considered in this work is the isothermal ($I_\theta = 0$) immiscible two-phase flow corresponding to the following component-phase matrix:

$$\mathbf{M} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

We use the common notation $\mathcal{C} = \mathcal{P} = \{\text{w}, \text{o}\}$ (where w and o correspond to the wetting and non-wetting phase respectively, both mono-components). As a component is present only in one phase we have:

$$\begin{gathered} \mathcal{C}_\text{w} = \{\text{w}\},\ \mathcal{C}_\text{o} = \{\text{o}\}, \\ \{C_{p,c}\}_{p\in\mathcal{P},c\in\mathcal{C}} = \{C_{\text{w,w}}, C_{\text{o,o}}\} \text{ and } C_{\text{w,w}} = C_{\text{o,o}} = 1, \end{gathered} \tag{2.1.18}$$

so that $\{C_{p,c}\}_{p\in\mathcal{P},c\in\mathcal{C}}$ is known. Thus, the conservation of quantity of the matter equations (2.1.6b) disappear from the model. For the same reason, the fugacity equality (2.1.6c) is not useful here. The set of unknowns in the reference context reduces to:

$$\mathcal{U} = \{P, S_\text{o}, S_\text{w}\},$$

and the contexts are:

$$\mathcal{K} = \{\{\text{o}\}, \{\text{w}\}, \{\text{o}, \text{w}\}\}.$$

System (2.1.5)–(2.1.6) becomes:

$$\begin{aligned}
\partial_t(\phi\zeta_{\rm w}(P)S_{\rm w}) + \mathrm{div}\left(\frac{\zeta_{\rm w}(P)k_{\rm r_w}(S_{\rm w})}{\mu_{\rm w}(P)}\overrightarrow{\mathbf{v}}_{\rm w}(P,(\theta),S_{\rm w})\right) &= q_{\rm w} \qquad \text{in } \Sigma,\\
\partial_t(\phi\zeta_{\rm o}(P)S_{\rm o}) + \mathrm{div}\left(\frac{\zeta_{\rm o}(P)k_{\rm r_o}(S_{\rm o})}{\mu_{\rm o}(P)}\overrightarrow{\mathbf{v}}_{\rm o}(P,(\theta),S_{\rm o})\right) &= q_{\rm o} \qquad \text{in } \Sigma,\\
S_{\rm o} + S_{\rm w} &= 1 \qquad \text{in } \Sigma,
\end{aligned} \tag{2.1.19}$$

where, $\overrightarrow{\mathbf{v}}_{\rm o}$ and $\overrightarrow{\mathbf{v}}_{\rm w}$ are again given by (2.1.8). Boundary and initial conditions close the model.

A classical practical example of an isothermal, immiscible two-phase flow is provided by the five-spot case, described by, e.g. Trangenstein and Bell [67]. We have a square domain with an injection well in each corner and a production well in the center. We refer to Section 2.1.3.7 for a brief description of the well model. At initial time, the domain is full of oil ($S_{\rm o} = 1$) and we inject water through the injection wells. The oil is pushed out of the domain through the production well.

Example 2.8 (SAGD thermal flow)**.** A second example is provided by the Steam-Assisted Gravity Drainage (SAGD) model. The principle of the SAGD procedure is to inject water steam to heat the oil phase, thereby reducing its viscosity. The component-phase matrix associated to the problem is as follows:

$$\mathbf{M} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Here we have two components, say $\mathcal{C} = \{\mathrm{e}, \mathrm{h}\}$ corresponding to water and a heavy oil, and three phases, say $\mathcal{P} = \{\mathrm{w}, \mathrm{s}, \mathrm{o}\}$ corresponding to liquid water, steam, and oil, respectively. We have:

$$\begin{gathered}
\mathcal{C}_{\rm w} = \{\mathrm{e}\},\ \mathcal{C}_{\rm s} = \{\mathrm{e}\},\ \mathcal{C}_{\rm o} = \{\mathrm{h}\},\\
\{C_{p,c}\}_{p\in\mathcal{P},c\in\mathcal{C}} = \{C_{\rm w,e}, C_{\rm s,e}, C_{\rm o,h}\} \text{ and } C_{\rm w,e} = C_{\rm s,e} = C_{\rm o,h} = 1,
\end{gathered} \tag{2.1.20}$$

so that $\{C_{p,c}\}_{p\in\mathcal{P},c\in\mathcal{C}}$ is known. Then the set of unknowns in the reference context is

$$\mathcal{U} = \{P, (\theta), S_{\rm w}, S_{\rm s}, S_{\rm o}\},$$

and the contexts are:

$$\mathcal{K} = \{\emptyset, \{\mathrm{w}\}, \{\mathrm{s}\}, \{\mathrm{o}\}, \{\mathrm{w},\mathrm{s}\}, \{\mathrm{w},\mathrm{o}\}, \{\mathrm{s},\mathrm{o}\}, \{\mathrm{w},\mathrm{s},\mathrm{o}\}\}.$$

In this case, system (2.1.5)–(2.1.6)–(2.1.10) becomes (for simplicity, when there is no ambiguity, we omit the arguments in the functions)

$$
\begin{aligned}
\partial_t\left(\phi\zeta_{\mathrm{w}}(P,(\theta))S_{\mathrm{w}}+\ \phi\zeta_{\mathrm{s}}(P,(\theta))S_{\mathrm{s}}\right)+&\\
\operatorname{div}\Big(\ \frac{\zeta_{\mathrm{w}}(P,(\theta))k_{\mathrm{r_w}}(S_{\mathrm{w}})}{\mu_{\mathrm{w}}(P,(\theta))}\overrightarrow{\mathbf{v}}_{\mathrm{w}}(P,(\theta),S_{\mathrm{w}})+\frac{\zeta_{\mathrm{s}}(P,(\theta))k_{\mathrm{r_s}}(S_{\mathrm{s}})}{\mu_{\mathrm{s}}(P,(\theta))}\overrightarrow{\mathbf{v}}_{\mathrm{s}}(P,(\theta),S_{\mathrm{s}})\Big)&=q_{\mathrm{e}} \qquad \text{in } \Sigma,\\
\partial_t\left(\phi\zeta_{\mathrm{o}}(P,(\theta))S_{\mathrm{o}}\right)+\operatorname{div}\left(\frac{\zeta_{\mathrm{o}}(P,(\theta))k_{\mathrm{r_o}}(S_{\mathrm{o}})}{\mu_{\mathrm{o}}(P,(\theta))}\overrightarrow{\mathbf{v}}_{\mathrm{o}}(P,(\theta),S_{\mathrm{o}})\right)&=q_{\mathrm{h}} \qquad \text{in } \Sigma,\\
\partial_t e+\operatorname{div}\Bigg(\frac{\zeta_{\mathrm{w}}(P,(\theta))k_{\mathrm{r_w}}(S_{\mathrm{w}})}{\mu_{\mathrm{w}}(P,(\theta))}h_{\mathrm{w}}(P,(\theta))\overrightarrow{\mathbf{v}}_{\mathrm{w}}(P,(\theta),S_{\mathrm{w}})+&\\
\frac{\zeta_{\mathrm{s}}(P,(\theta))k_{\mathrm{r_s}}(S_{\mathrm{s}})}{\mu_{\mathrm{s}}(P,(\theta))}h_{\mathrm{s}}(P,\theta)\overrightarrow{\mathbf{v}}_{\mathrm{s}}(P,(\theta),S_{\mathrm{s}})+&\\
\frac{\zeta_{\mathrm{o}}(P,(\theta))k_{\mathrm{r_o}}(S_{\mathrm{o}})}{\mu_{\mathrm{o}}(P,(\theta))}h_{\mathrm{o}}(P,\theta)\overrightarrow{\mathbf{v}}_{\mathrm{o}}(P,(\theta),S_{\mathrm{o}})-\lambda\nabla\theta\Bigg)&=Q \qquad \text{in } \Sigma,\\
S_{\mathrm{w}}+S_{\mathrm{s}}+S_{\mathrm{o}}&=1 \qquad \text{in } \Sigma,\\
f_{\mathrm{e,w}}(P,(\theta))-f_{\mathrm{e,s}}(P,(\theta))&=0 \qquad \text{in } \Sigma.
\end{aligned}
$$

A complete description of a SAGD test case can be found, e.g., in the book of Aziz and Settari [15] or in the work of Mamaghani, Enchéry, and Chainais [57].

2.2 The discrete setting

Although the two-phase flow in the next chapter will be discretized in rectangular meshes, we present here a more general setting.

By definition a *mesh* $\mathcal{T}=\{T\}$ of the space domain $\Omega\in\mathbb{R}^d$, $d=2$ or 3, is a partition of Ω, into open nonempty disjoint triangles or quadrilaterals if $d=2$, and tetrahedra or hexahedra if $d=3$, such that

$$\overline{\Omega}=\bigcup_{T\in\mathcal{T}}\overline{T},$$

i.e., $\mathcal{T}$ covers Ω exactly. We assume that the mesh is conforming and we let $N_{\mathcal{T}}:=\operatorname{card}(\mathcal{T})$. The elements of $\mathcal{T}$ are sometimes referred to as cells since the latter term is more common in Finite Volume (FV) methods. For every element $T\in\mathcal{T}$, we denote by $|T|$ its d-dimensional measure and by h_T its diameter. The mesh size is defined by

$$h_{\mathcal{T}}:=\sup_{T\in\mathcal{T}}\ h_T.$$

Let $\mathcal{F} = \{\sigma\}$ be the set of hyperplanar mesh faces and, for all $T \in \mathcal{T}$, set

$$\mathcal{F}_T := \{\sigma \in \mathcal{F} \mid \sigma \subset \partial T\}. \tag{2.2.1}$$

The set of faces is additionally partitioned into $\mathcal{F} = \mathcal{F}^{\mathrm{i}} \cup \mathcal{F}^{\mathrm{b}}$ where $\mathcal{F}^{\mathrm{i}}$ collects all internal faces $\sigma \not\subset \partial\Omega$ while $\mathcal{F}^{\mathrm{b}}$ collects all boundary faces $\sigma \subset \partial\Omega$.

We denote by $\mathcal{T}_\sigma$ the set of cells that share the face σ

$$\mathcal{T}_\sigma = \{T \in \mathcal{T} \,|\, \sigma \subset \partial T\}. \tag{2.2.2}$$

We assume that an internal face $\sigma \in \mathcal{F}^{\mathrm{i}}$ is shared by exactly two cells $T_1, T_2 \in \mathcal{T}$ such that $\sigma = \partial T_1 \cap \partial T_2$. The ordering of these two cells is arbitrary but fixed. By convention, fluxes across σ as well as the normal vector $\mathbf{n}_\sigma$ are taken outward from T_1. The $(d-1)$-dimensional measure of a face $\sigma \in \mathcal{F}$ is denoted by $|\sigma|$. The barycenter of a face $\sigma \in \mathcal{F}$ is denoted by $\mathbf{x}_\sigma$. For all $T \in \mathcal{T}$ we identify a *cell-center*, i.e., the barycenter $\mathbf{x}_T \in T$. The component of $\mathbf{x}_T$ in the (oriented) opposite direction to gravity is denoted by z_T. For all $T \in \mathcal{T}$ and $\sigma \in \mathcal{F}_T$, we denote by $d_{T,\sigma}$ the Euclidean distance between the cell-center $\mathbf{x}_T$ and the face σ, and by $\mathbf{n}_{T,\sigma}$ the unit vector normal to σ outward to T.

For the time discretization, we use here a *fully implicit* time stepping scheme.
Let $\{t^n\}_{0\le n\le N_{\mathrm{F}}}$ be a strictly increasing sequence of discrete times such that $t^0 = 0$ and $t^{N_{\mathrm{F}}} = t_{\mathrm{F}}$. For $1 \le n \le N_{\mathrm{F}}$, we define the time interval $I_n :=]t^{n-1}, t^n]$ and the time step $\tau^n := t^n - t^{n-1}$.

For a function of time φ with sufficient regularity we set $\varphi^n := \varphi(t^n)$, $0 \le n \le N_{\mathrm{F}}$, and define the backward differencing operator:

$$\delta_t \varphi^n := \frac{1}{\tau^n}(\varphi^n - \varphi^{n-1}). \tag{2.2.3}$$

If, in addition, φ is affine inside each time interval I_n, $1 \le n \le N_{\mathrm{F}}$, there holds

$$\delta_t \varphi^n = \partial_t \varphi_{|I_n}.$$

Remark 2.9 (Adaptive mesh)**.** In the context of *Adaptive Mesh Refinement* (AMR), sequences of meshes $\{\mathcal{T}^n\}_{0\le n\le N_{\mathrm{F}}}$ are considered. On each time step, the mesh is updated according to a posteriori error estimates; Cells are conserved, coarsened, or refined. The following work could be done with such meshes but for a sake of simplicity, we only consider a fixed mesh $\mathcal{T}$.

2.2.1 Unknowns

In the spirit of FV methods, the unknowns appearing in the continuous problem are discretized using only one value per cell, which can be alternatively interpreted as an average over the cell or as the value at the cell-center. We introduce the piecewise constant discrete contexts $\{k_T^n\}_{T\in\mathcal{T}}$ such that, for all $1 \le n \le N_{\rm F}$ and all $T \in \mathcal{T}$, $k_T^n \in \mathcal{K}$ represents the context in the cell T, at time t^n. The set of phases present in the cell T is denoted by $\mathcal{P}_{k_T^n}$. It is updated during the computation at each time step. The discrete counterpart of the unknowns (2.1.3) is, for all $T \in \mathcal{T}$ and $0 \le n \le N_{\rm F}$,

$$\mathcal{U}_T^n := \left\{ P_T^n, (\theta_T^n), \{S_{p,T}^n\}_{p\in\mathcal{P}_{k_T^n}}, \{C_{p,c,T}^n\}_{p\in\mathcal{P}_{k_T^n}, c\in\mathcal{C}_p}, \{n_{c,T}^n\}_{c\in\overline{\mathcal{C}_{k_T^n}}} \right\}, \tag{2.2.4}$$

where we recall that, at time t^n, in the cell T

- P_T^n denotes the reference pressure;
- θ_T^n is the temperature, only present when $I_\theta = 1$;
- $\{S_{p,T}^n\}_{p\in\mathcal{P}_{k_T^n}}$ is the set of saturations for the phases present in the context k_T^n;
- $\{C_{p,c,T}^n\}_{p\in\mathcal{P}_{k_T^n}, c\in\mathcal{C}_p}$ is the set of molar fractions for the components of each phase present in the cell T;
- $\{n_c\}_{c\in\overline{\mathcal{C}_{k_T^n}}}$ contains the numbers of moles for the components absent from the context k_T^n, which are collected in the set $\overline{\mathcal{C}_{k_T^n}} := \mathcal{C} \setminus \mathcal{C}_{k_T^n}$.

Once the unknowns of the set $\mathcal{U}_T^n$ are ordered for all $T \in \mathcal{T}$, $0 \le n \le N_{\rm F}$, and this order is fixed, these unknowns are stored in a local vector $\mathbf{u}_T^n$, which concurs in forming the global vector $\mathbf{u}^n := (\mathbf{u}_T^n)_{T\in\mathcal{T}}$.

2.2.2 Discrete equations

We consider in this section a discretization of problem (2.1.5)–(2.1.6)–(2.1.10) based on Multi-Point FV (MPFV) scheme and phase-upwinding. Recall that throughout this work we restrict ourselves to fully implicit time discretizations. Thus we need to integrate the conservation equations over a control volume $T \in \mathcal{T}$ and over a time interval I_n. We have, for each

component $c \in \mathcal{C}$:

$$\int_{I_n}\int_T \left(\partial_t n_c + \sum_{p\in\mathcal{P}_{\mathfrak{K}}\cap\mathcal{P}_c} \operatorname{div}\left(\frac{\zeta_p(P,(\theta),\mathbf{C}_p)k_{\mathrm{r}_p}(S_p)}{\mu_p(P,(\theta),\mathbf{C}_p)} C_{p,c}\overrightarrow{\mathbf{v}}_p(P,(\theta),S_p,\mathbf{C}_p)\right)\right) d\mathbf{x}\,dt \\ = \int_{I_n}\int_T q_c\; d\mathbf{x}\,dt. \tag{2.2.5}$$

Next, using the Gauss theorem, we express the divergence integral over T, present in equation (2.2.5), in terms of fluxes on the boundary ∂T of T. As the summation on the phases does not depend on space or time, we obtain:

$$\int_T \left(n_c^n - n_c^{n-1}\right)\; d\mathbf{x} + \\ \sum_{p\in\mathcal{P}_{\mathfrak{K}}\cap\mathcal{P}_c} \left(\int_{I_n}\int_{\partial T} \left(\frac{\zeta_p(P,(\theta),\mathbf{C}_p)k_{\mathrm{r}_p}(S_p)}{\mu_p(P,(\theta),\mathbf{C}_p)} C_{p,c}\overrightarrow{\mathbf{v}}_p(P,(\theta),S_p,\mathbf{C}_p)\right)\cdot\mathbf{n}\,d\Gamma\,dt\right) \\ = \int_{I_n}\int_T q_c\; d\mathbf{x}\,dt. \tag{2.2.6}$$

Recall that $\mathcal{F}_T$ is the set of faces of T. Let us divide both sides of this equation by τ^n. The second term in (2.2.6) has the expression:

$$\frac{1}{\tau^n}\sum_{\sigma\in\mathcal{F}_T}\sum_{p\in\mathcal{P}_{\mathfrak{K}}\cap\mathcal{P}_c} \left(\int_{I_n}\int_\sigma \left(\frac{\zeta_p(P,(\theta),\mathbf{C}_p)k_{\mathrm{r}_p}(S_p)}{\mu_p(P,(\theta),\mathbf{C}_p)} C_{p,c}\overrightarrow{\mathbf{v}}_p(P,(\theta),S_p,\mathbf{C}_p)\right)\cdot\mathbf{n}\,d\Gamma\,dt\right). \tag{2.2.7}$$

Therefore,

$$\frac{1}{\tau^n}\int_T \left(n_c^n - n_c^{n-1}\right)\; d\mathbf{x} + \\ \frac{1}{\tau^n}\sum_{\sigma\in\mathcal{F}_T}\sum_{p\in\mathcal{P}_{\mathfrak{K}}\cap\mathcal{P}_c} \left(\int_{I_n}\int_\sigma \left(\frac{\zeta_p(P,(\theta),\mathbf{C}_p)k_{\mathrm{r}_p}(S_p)}{\mu_p(P,(\theta),\mathbf{C}_p)} C_{p,c}\overrightarrow{\mathbf{v}}_p(P,(\theta),S_p,\mathbf{C}_p)\right)\cdot\mathbf{n}\,d\Gamma\,dt\right) \\ = \frac{1}{\tau^n}\int_{I_n}\int_T q_c\; d\mathbf{x}\,dt. \tag{2.2.8}$$

Considering that the approximating unknowns are piecewise constant in each control volume, the first term in (2.2.8), has the straightforward approximation:

$$\frac{|\,T\,|}{\tau^n}\left(n_{c,T}^n - n_{c,T}^{n-1}\right). \tag{2.2.9}$$

More precisely, for a component c in the control volume T at time t^n, if $c \in \mathcal{C}_{k_T^n}$, we define the discrete *accumulation term* by discretizing the mole number equation (2.1.7):

$$n_{c,T}^n = \phi \sum_{p\in\mathcal{P}_{k_T^n}\cap\mathcal{P}_c} \zeta_p(P_T^n,(\theta)_T^n,\mathbf{C}_{p,T}^n) S_{p,T}^n C_{p,c,T}^n. \tag{2.2.10}$$

As stated in Section 2.1.1, when $c \in \overline{\mathcal{C}_{k_T}}$, i.e. when the component c is absent from the present context, then $n_{c,T}$ is an independent unknown.

The approximation of the term (2.2.7) is more delicate because of the fluxes $\overrightarrow{\mathbf{v}}_p \cdot \mathbf{n}|_\sigma$. This is the object of the next section. To improve readability, in what follows, the superscript n is systematically omitted from all time-dependent quantities.

2.2.2.1 Multi-point flux discretization

Here we concentrate on multi-point flux discretization. The two-point FV method for the two-phase flow will be described in the next chapter. We turn to the discretization of the fluxes $\overrightarrow{\mathbf{v}}_p \cdot \mathbf{n}|_\sigma$ over the faces σ of the mesh. A key ingredient of the method discussed in this section is the discretization of second-order elliptic terms on general meshes. It is a well-known fact that the classical two-point FV method fails to be consistent on meshes that do not satisfy the $\mathbf{K}$-orthogonality condition, for example for refined meshes, which are not conforming; see, e.g., Di Pietro [38, §5.1]. While several remedies have been suggested over the last years by Droniou and Eymard in [43], by Eymard, Gallouët and Herbin in [50], and by Di Pietro [37, 39, 40], we describe here for the compositional model the MPFV methods, which have been independently introduced in the late 90s by Aavatsmark, Barkve, Bøe, and Mannseth [4, 5] and Edwards and Rogers [45]. More specifically, the actual implementation relies on the MPFV method introduced and analyzed by Agélas, Di Pietro, and Droniou [7], following the idea of Agélas, Di Pietro, and Masson [8].

In finite volume methods, the fluxes are computed at the faces and can be written as a linear combination of the discrete unknowns presented before. Let us stress that we impose conservative fluxes on the faces, i.e., "What flows out corresponds to what flows in".

The following notation will be used in the sequel. The mean value of a function f in a region $\mathcal{O}$ is denoted by

$$\langle f \rangle_{\mathcal{O}} = \frac{1}{|\mathcal{O}|} \int_{\mathcal{O}} f(\mathbf{x}) d\mathbf{x}.$$

Fluxes discretization The fluxes that are relevant for the discretization of system (2.1.5) are those associated with the following auxiliary scalar problem:

$$\begin{aligned} -\operatorname{div}(\mathbf{K}\nabla\xi) &= f \qquad \text{in } \Omega, \\ \mathbf{K}\nabla\xi\cdot\mathbf{n} &= 0 \qquad \text{on } \partial\Omega, \\ \langle\xi\rangle_\Omega &= 0, \\ \langle f\rangle_\Omega &= 0. \end{aligned} \tag{2.2.11}$$

Observe that the homogeneous Neumann boundary condition is consistent with the no-flow boundary condition (2.1.11). The key idea of MPFV methods is to express fluxes as a linear combination of cell unknowns. The MPFV discretization of problem (2.2.11) reads

$$\text{Find } \boldsymbol{\xi}\in\mathbb{R}^{\mathcal{T}} \text{ such that } \sum_{\sigma\in\mathcal{F}_T\cap\mathcal{F}^{\mathrm{i}}}\Phi_{T,\sigma}(\boldsymbol{\xi}) = \langle f\rangle_T \text{ for all } T\in\mathcal{T},$$

where, for all $T\in\mathcal{T}$, $\mathcal{F}_T$ is defined by (2.2.1) and for all $\sigma\in\mathcal{F}_T\cap\mathcal{F}^i$, $\Phi_{T,\sigma}(\boldsymbol{\xi})$ denotes an approximation of the normal flow of $-\mathbf{K}\nabla\xi$ leaving T through σ expressed in terms of the values of $\boldsymbol{\xi}=(\xi_T)_{T\in\mathcal{T}}$. In particular, for all faces $\sigma\in\mathcal{F}$, we introduce a *flux stencil* $\mathcal{S}_\sigma^{\mathrm{mm}}\subset\mathcal{T}$, associated with the mass and momentum conservation system discretization (2.1.5). The stencil depends on the method. For example, for the two-point FV method, if $\sigma=\partial T_1\cap\partial T_2$ then the stencil $\mathcal{S}_\sigma^{\mathrm{mm}}=\{T_1,T_2\}$. Thus for all $T\in\mathcal{T}$, all $\sigma\in\mathcal{F}^{\mathrm{i}}\cap\mathcal{F}_T$, and all $\boldsymbol{\xi}\in\mathbb{R}^{\mathcal{T}}$,

$$\Phi_{T,\sigma}(\boldsymbol{\xi}) := \sum_{T'\in\mathcal{S}_\sigma^{\mathrm{mm}}}\tau_{T'}^{\sigma}\xi_{T'},$$

where for all $T\in\mathcal{S}_\sigma^{\mathrm{mm}}$, $\tau_T^\sigma\in\mathbb{R}$ are called *transmissibility coefficients* of the face σ. These coefficients satisfy

$$\sum_{T'\in\mathcal{S}_\sigma^{\mathrm{mm}}}\tau_{T'}^{\sigma}=0.$$

The following flux conservation property is verified:

$$\Phi_{T_1,\sigma}(\boldsymbol{\xi}) = -\Phi_{T_2,\sigma}(\boldsymbol{\xi}),$$

for all $\sigma\in\mathcal{F}^{\mathrm{i}}$, $\sigma=\partial T_1\cap\partial T_2$, and all $\boldsymbol{\xi}\in\mathbb{R}^{\mathcal{T}}$.

In what follows it is assumed that, for all $\sigma\in\mathcal{F}^{\mathrm{i}}$, $\mathcal{T}_\sigma$ is contained in $\mathcal{S}_\sigma^{\mathrm{mm}}$.

Remark 2.10 (Upwinding stencil)**.** In fluid dynamics computations, upwind numerical fluxes are often chosen for the discretization of the advective terms, see, e.g., the book of Eymard,

Gallouët, and Herbin [49]. Their advantage is that they simulate the direction of propagation of information in a fluid field and stabilize the scheme, but then the stencil used to compute the fluxes can vary in time and space.

Consider the case where the permeability tensor has the form

$$\mathbf{K} = \kappa \mathbf{Id}, \tag{2.2.12}$$

where κ is constant in time and is a positive piecewise constant function in space, uniformly bounded above and uniformly bounded away from 0.

We are now ready to define the phase fluxes for our model (2.1.5) in this case. Recall that, for the sake of brevity, we focus on the generic time step n and omit the superscript n from all the time-dependent quantities. For all $\sigma \in \mathcal{F}^{\mathrm{i}}$ and all $T \in \mathcal{T}_\sigma$, the diffusive flux of the phase $p \in \bigcup_{T\in\mathcal{T}_\sigma} \mathcal{P}_{k_T}$ is discretized by:

$$F_{p,T,\sigma}(\mathbf{u}) = F_{p,T,\sigma}\left(\{\mathbf{u}_{T'}\}_{T'\in\mathcal{S}_\sigma^{\mathrm{mm}}}\right) = \sum_{T'\in\mathcal{S}_\sigma^{\mathrm{mm}}} \tau_{T'}^\sigma \left(P_{p,T'} + \rho_{p,\sigma} g z_{T'}\right), \tag{2.2.13}$$

where g denotes the acceleration due to the standard gravity, defined as $9.80665\,\mathrm{m/s^2}$. Recall that $\mathcal{P}_{k_T}$ is the set of phases of the context k_T present in the cell T. The transmissibility coefficients τ_T^σ depend on the discretization grid, the permeability tensor $\mathbf{K}$ and on the chosen scheme. These coefficients can be preprocessed and stored, so that they can be used as an entry for solving the fluid dynamics system.

Let us compute $\tau_{T_1}^\sigma$, for a two-point FV scheme, for $\sigma = T_1 \cap T_2$, where T_1 and T_2 are two neighboring cells. We assume that in each cell the medium is homogeneous, that is, in each cell $\kappa := \kappa_T$ a constant. Let $\mathbf{x}_\sigma$ and $\mathbf{x}_T$ denote the barycenter of the face σ and the cell T respectively. Moreover let T_1 and T_2 be ordered in such a way that the normal vector $\mathbf{n}_\sigma$ points into T_2. Then the coefficient $\tau_{T_1}^\sigma$ is defined by:

$$\tau_{T_1}^\sigma = |\sigma| \frac{\frac{\kappa_{T_1}}{\|\mathbf{x}_\sigma - \mathbf{x}_{T_1}\|_2} \frac{\kappa_{T_2}}{\|\mathbf{x}_\sigma - \mathbf{x}_{T_2}\|_2}}{\frac{\kappa_{T_1}}{\|\mathbf{x}_\sigma - \mathbf{x}_{T_1}\|_2} + \frac{\kappa_{T_2}}{\|\mathbf{x}_\sigma - \mathbf{x}_{T_2}\|_2}} = \frac{|\sigma| \kappa_{T_1}\, \kappa_{T_2}}{\kappa_{T_1} \parallel \mathbf{x}_\sigma - \mathbf{x}_{T_2} \parallel_2 + \kappa_{T_2} \parallel \mathbf{x}_\sigma - \mathbf{x}_{T_1} \parallel_2} \tag{2.2.14}$$

and

$$\tau_{T_2}^\sigma = -\tau_{T_1}^\sigma, \tag{2.2.15}$$

where the negative sign of $\tau_{T_2}^\sigma$ enforces the conservativity of fluxes.

Remark 2.11. If the medium in addition is everywhere homogeneous and isotropic, i.e.

$$\forall T \in \mathcal{T},\ \kappa_T = \kappa,$$

then the coefficient τ_T^σ reduces to:

$$\tau_{T_1}^\sigma = |\sigma|\kappa \frac{\frac{1}{\|\mathbf{x}_\sigma - \mathbf{x}_{T_1}\|_2 \|\mathbf{x}_\sigma - \mathbf{x}_{T_2}\|_2}}{\frac{1}{\|\mathbf{x}_\sigma - \mathbf{x}_{T_1}\|_2} + \frac{1}{\|\mathbf{x}_\sigma - \mathbf{x}_{T_2}\|_2}} = \frac{|\sigma|\kappa}{\| \mathbf{x}_\sigma - \mathbf{x}_{T_2} \|_2 + \| \mathbf{x}_\sigma - \mathbf{x}_{T_1} \|_2} \tag{2.2.16}$$

and

$$\tau_{T_2}^\sigma = -\tau_{T_1}^\sigma. \tag{2.2.17}$$

The approximation of the mass density, $\rho_{p,\sigma}$, of the phase p at $\mathbf{x}_\sigma$ is given by

$$\rho_{p,\sigma} := \begin{cases} \rho_p(\mathbf{u}_{T_1}) & \text{if } p \in \mathcal{P}_{k_{T_1}} \setminus \mathcal{P}_{k_{T_2}}, \\ \rho_p(\mathbf{u}_{T_2}) & \text{if } p \in \mathcal{P}_{k_{T_2}} \setminus \mathcal{P}_{k_{T_1}}, \\ \left(\rho_p(\mathbf{u}_{T_1}) + \rho_p(\mathbf{u}_{T_2})\right)/2 & \text{otherwise.} \end{cases} \tag{2.2.18}$$

Other choices are possible for $\rho_{p,\sigma}$ but are not detailed herein.

Fourier fluxes discretization In the compositional model, if we want to study the thermal case, we have $I_\theta = 1$, and the Fourier heat fluxes relevant to the discretization of equation (2.1.10) are those associated with the following auxiliary scalar problem:

$$\begin{aligned} -\operatorname{div}(\lambda \nabla \xi) &= q \qquad \text{in } \Omega, \\ \xi &= \overline{\xi} \qquad \text{on } \partial\Omega. \end{aligned} \tag{2.2.19}$$

Observe that the Dirichlet boundary condition is consistent with the fixed temperature condition (2.1.12). We use a MPFV method to express the fluxes as a linear combination of cell unknowns, and the discretization of problem (2.2.19) reads

$$\text{Find } \boldsymbol{\xi} \in \mathbb{R}^{\mathcal{T}} \text{ such that } \sum_{\sigma \in \mathcal{F}_T} \Psi_{T,\sigma}(\boldsymbol{\xi}) = \langle q \rangle_T \text{ for all } T \in \mathcal{T},$$

where, for all $T \in \mathcal{T}$ and all $\sigma \in \mathcal{F}_T$, $\Psi_{T,\sigma}(\boldsymbol{\xi})$ denotes an approximation of the normal flow of $-\lambda \nabla \xi$ leaving T through σ expressed in terms of the values of $\boldsymbol{\xi} = (\xi_T)_{T \in \mathcal{T}}$. In particular, for all faces $\sigma \in \mathcal{F}$, we introduce the stencil $\mathcal{S}_\sigma^{\text{en}} \subset \mathcal{T}$ associated with the energy conservation equation (2.1.10) discretization. It depends on the method and is different from $\mathcal{S}_\sigma^{\text{mm}}$ if the

equation is discretized by another method. In what follows, it is assumed that, for all $\sigma \in \mathcal{F}^{\mathrm{i}}$, $\mathcal{T}_\sigma$ is contained in $\mathcal{S}_\sigma^{\mathrm{en}}$. Then, for all $T \in \mathcal{T}$, all $\sigma \in \mathcal{F}_T$, and all $\boldsymbol{\xi} \in \mathbb{R}^{\mathcal{T}}$,

$$\Psi_{T,\sigma}(\boldsymbol{\xi}) := \sum_{T' \in \mathcal{S}_\sigma^{\mathrm{en}}} \tau_{T'}^\sigma \xi_{T'},$$

where $\tau_T^\sigma \in \mathbb{R}$, for all $T \in \mathcal{S}_\sigma^{\mathrm{en}}$. Finally the following flux conservation property needs to be verified:

$$\Psi_{T_1,\sigma}(\boldsymbol{\xi}) = -\Psi_{T_2,\sigma}(\boldsymbol{\xi}),$$

for all $\sigma \in \mathcal{F}^{\mathrm{i}}$, $\sigma = \partial T_1 \cap \partial T_2$, and all $\boldsymbol{\xi} \in \mathbb{R}^{\mathcal{T}}$.

We can now define the Fourier fluxes, under the same assumptions as for the phase fluxes. Thus, for $\sigma \in \mathcal{F}^{\mathrm{i}}$, and all $T \in \mathcal{T}_\sigma$, the Fourier flux is given by:

$$G_{T,\sigma}(\mathbf{u}) = G_{T,\sigma}(\{\mathbf{u}_{T'}\}_{T' \in \mathcal{S}_\sigma^{\mathrm{en}}}) = \sum_{T' \in \mathcal{S}_\sigma^{\mathrm{en}}} \tau_{T'}^\sigma \theta_{T'}. \tag{2.2.20}$$

2.2.2.2 Upwinding the advective fluxes

As we saw in the previous part, we want to control the propagation of information in the flow and stabilize the scheme. To this end, for all phases $p \in \mathcal{P}$ and a given law $\ell \in \{\zeta_p, k_{\mathrm{r}_p}, \mu_p\}$, we introduce the notation $\ell_X := \ell(\mathbf{u}_X)$, where the control volume X is a cell T, a perforated cell W, or an upwind cell for the phase p defined as follows. The symbol $T_p^\uparrow$ denotes the *upwind cell* for the phase p defined for a face $\sigma \in \mathcal{F}^i$ by:

$$\forall \sigma = \partial T_1 \cap \partial T_2,\ \forall p \in \mathcal{P}_{T_1} \cup \mathcal{P}_{T_2}, \qquad T_p^\uparrow := \begin{cases} T_1 & \text{if } F_{p,T_1,\sigma}(\mathbf{u}) > 0, \\ T_2 & \text{otherwise.} \end{cases} \tag{2.2.21}$$

If the face is located on the boundary $\sigma \in \mathcal{F}^{\mathrm{b}}$, in view of the homogeneous Neumann condition, $F_{p,T,\sigma}(\mathbf{u}) = 0$ and we do not need to evaluate the transport term.
In Figure 2.2, the upwind cell for the phase p flowing through the face σ is T_1 because the phase p flows from T_1 to T_2. Thus to compute the new flux we need the information coming from T_1 and the laws are evaluated with the variable $\mathbf{u}_{T_1}$.

2.2.2.3 The boundary conditions

Summing up, no-flow boundary conditions are enforced for all phases (see the paragraph on boundary conditions in Section 2.1.3.6), i.e. fluxes flowing across the boundaries are equal to

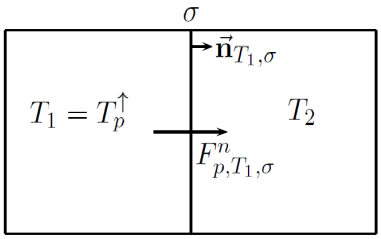

Figure 2.2: Example of an internal upwind cell configuration.

0. Then for $p \in \mathcal{P}$, $T \in \mathcal{T}$ such that ∂T contains a face σ^{b} on the boundary:

$$F_{p,T,\sigma^{\mathrm{b}}}(\mathbf{u}_T) = 0. \tag{2.2.22}$$

For thermal problems, the diffusive Fourier fluxes are evaluated using the two-point FV scheme. Thus, for $\sigma \in \mathcal{F}^{\mathrm{b}}$, we have $\mathcal{T}_\sigma = T^{\mathrm{b}}$, with T^{b} the boundary cell such that $\sigma \in \partial T^{\mathrm{b}}$, and

$$G^{\mathrm{b}}_{T,\sigma}(\mathbf{u}) = G^{\mathrm{b}}_{T,\sigma}(\mathbf{u}_{T^{\mathrm{b}}}) = \tau^{\sigma}_{T^{\mathrm{b}}}(\theta_{T^{\mathrm{b}}} - \overline{\theta}). \tag{2.2.23}$$

2.2.2.4 Wells

Now, let us sketch the discretization of the contribution of wells, introduced in Section 2.1.3.7. Recall that the wells are collected in the set $\mathcal{W}$, composed of injection and production wells. Injection wells are collected in the set $\mathcal{W}^{\mathrm{i}}$, production wells are collected in the set $\mathcal{W}^{\mathrm{p}}$. Note that

$$\mathcal{W}^{\mathrm{i}} \cup \mathcal{W}^{\mathrm{p}} = \mathcal{W} \quad \text{and} \quad \mathcal{W}^{\mathrm{i}} \cap \mathcal{W}^{\mathrm{p}} = \emptyset.$$

Recall that the injected fluid will push the hydrocarbons, whereas the production wells permit the fluid to flow out of the reservoir.

For all $W \in \mathcal{W}$, we denote by $\mathcal{T}_W \subset \mathcal{T}$ the set of cells which contain the perforations of W. For all $W \in \mathcal{W}$ and all perforated cells $T \in \mathcal{T}_W$, we denote by $z_{W,T}$ the height of the perforation opposite to the direction along which gravity acts. For $0 \leq n \leq N_{\mathrm{F}}$, for every injection well $W \in \mathcal{W}^{\mathrm{i}}$, and for all perforations $T \in \mathcal{T}_W$, we additionally introduce the set $\mathcal{P}^n_{W,T}$ of phases present in the injected fluid flowing throughout this perforation. Then

$$\mathcal{U}^n_{W,T} := \left\{P^n_{W,T}, (\theta^n_{W,T}), \{S^n_{p,W,T}\}_{p\in\mathcal{P}_{W,T}}, \{C^n_{p,c,W,T}\}_{p\in\mathcal{P}_{W,T},c\in\mathcal{C}_p}\right\},$$

denotes the set of variables corresponding to the physical state of the injected fluid, they are supposed known.

For $0 \leq n \leq N_{\mathrm{F}}$, for every production well $W \in \mathcal{W}^{\mathrm{p}}$, and for all perforations $T \in \mathcal{T}_W$. We similarly introduce the set of unknowns:

$$\mathcal{U}_{W,T}^n := \left\{ P_{W,T}^n, (\theta_{W,T}^n), \{S_{p,W,T}^n\}_{p \in \mathcal{P}_{W,T}}, \{C_{p,c,W,T}^n\}_{p \in \mathcal{P}_{W,T}, c \in \mathcal{C}_p} \right\}.$$

As before, ordering the elements of $\mathcal{U}_{W,T}^n$, $0 \leq n \leq N_{\mathrm{F}}$, according to the choice of Section 2.2.1 yields the vector $\mathbf{u}_{W,T}^n$.

We are now ready to define phase fluxes across the perforations of wells. The discretization of a well contribution in the formulation (2.1.13), using a two-point method, is obtained by introducing fluxes flowing through the faces of each perforated cell. Thus, for all injection wells $W \in \mathcal{W}^{\mathrm{i}}$, all perforated cells $T \in \mathcal{T}_W$, and all phases $p \in \mathcal{P}_{W,T}$, the flux of the phase p entering from the perforation in T at time t^n is approximated by:

$$F_{p,W,T}(\mathbf{u}_T^n) = \tau_T^W \left[P_{p,T}^n - P_{W,T}^n + \rho_p(\mathbf{u}_{W,T}^n) g(z_T - z_{W,T}) \right], \tag{2.2.24}$$

where τ_T^W is the so-called *production index* of the well. The production index is a data determined by the characteristics of the well. Observe that this flux only depends on the unknowns associated with the perforated cell T.

Similarly, for all production wells $W \in \mathcal{W}^{\mathrm{p}}$, all perforated cells $T \in \mathcal{T}_W$, and all phases $p \in \mathcal{P}_{k_T}$, the flux of the phase p leaving the system from the perforation in T is approximated by:

$$F_{p,W,T}(\mathbf{u}_T^n) = \tau_T^W \left(P_{p,T}^n - P_{W,T}^n + \rho_p(\mathbf{u}_T^n) g(z_T - z_{W,T}) \right), \tag{2.2.25}$$

where τ_T^W is the so-called *production index* of the well given by the data.

In our model, τ_T^W is given by the Peaceman formula:

$$\tau_T^W := \frac{2\pi\kappa h_z}{ln(0.14(h_x^2 + h_y^2)) + s_k}, \tag{2.2.26}$$

where h_x (respectively h_y) denotes the cell width in the x (respectively y) direction, and h_z is the depth of the cell T. Recall that κ represents the value of the permeability tensor. The *skin factor*, s_k, is a dimensionless number that accounts for the effects resulting from the formation damage caused by drilling.

Moreover, we can define the Fourier fluxes for all wells $W \in \mathcal{W}^{\mathrm{i}}$ and all perforated cells $T \in \mathcal{T}_W$ by:

$$G_{W,T}(\mathbf{u}_T^n) = \tau_T^W (\theta_T^n - \theta_{W,T}^n). \tag{2.2.27}$$

Finally, there is no need to define the Fourier fluxes across the perforation of production wells, because they are assumed to be zero.

2.2.2.5 Closure equations

At each time step, the algebraic closure equations (2.1.6a)–(2.1.6c) are discretized in each cell $T \in \mathcal{T}$. Thus, for $k := k_T^n$, we have:

$$\sum_{p \in \mathcal{P}_k} S_{p,T}^n - 1 = 0, \tag{2.2.28a}$$

$$\sum_{c \in C_p} C_{p,c,T}^n - 1 = 0, \quad \forall p \in \mathcal{P}_k, \tag{2.2.28b}$$

$$f_{c,p_1}(P_T^n, (\theta_T^n), \mathbf{C}_{p,T}^n) - f_{c,p_2}(P_T^n, (\theta_T^n), \mathbf{C}_{p,T}^n) = 0, \quad \forall c \in \mathcal{C}_k,\ \forall p_1,\ p_2 \in \mathcal{P}_k \cap \mathcal{P}_c,\ p_1 \neq p_2. \tag{2.2.28c}$$

2.2.3 Assembling the discrete equations

In this paragraph, we describe some tools for computing the discrete model and then we present the strategy to solve the system. We discuss in this section the computation at each generic time step from t^{n-1} to t^n, $1 \leq n \leq N_{\mathrm{F}}$.

For a real $x \in \mathbb{R}$, we introduce the following notation for the positive and negative part of x:

$$x^{\oplus} := \frac{1}{2}\left(|x| + x\right), \qquad x^{\ominus} := \frac{1}{2}\left(|x| - x\right).$$

Observe that both $x^{\oplus}$ and $x^{\ominus}$ are non-negative.

2.2.3.1 Vector form of the discrete system

Recall that $\mathbf{u}^n := (\mathbf{u}_T^n)_{T \in \mathcal{T}}$ is the global vector of unknowns. We have:

$$\mathbf{u}_T^n := \begin{bmatrix} P_T^n \\ (\theta_T^n) \\ \{S_{p,T}^n\}_{p \in \mathcal{P}_{k_T^n}} \\ \{C_{p,c,T}^n\}_{p \in \mathcal{P}_{k_T^n},\, c \in \mathcal{C}_p} \\ \{n_{c,T}^n\}_{c \in \overline{\mathcal{C}_{k_T^n}}} \end{bmatrix}. \tag{2.2.29}$$

Then, at each time step n and for each cell $T \in \mathcal{T}$, we impose

- The N_c conservation of mass and momentum equations:

$$\mathbf{R}_{c,T}^{\mathrm{mm}}(\mathbf{u}^n) = \mathbf{0}, \qquad \forall c \in \mathcal{C}. \tag{2.2.30a}$$

- The energy conservation equations:

$$\mathbf{R}_{T}^{\mathrm{en}}(\mathbf{u}^n) = \mathbf{0}, \qquad \text{if } I_\theta = 1. \tag{2.2.30b}$$

- The closure equations:

$$\mathbf{D}_T(\mathbf{u}_T^n) = \mathbf{0}. \tag{2.2.30c}$$

This discrete equations are specified below. To account for the presence of possible wells, we introduce a Kronecker symbol for any pair of cells T and T':

$$\delta_{T,T'} = \begin{cases} 1 & \text{if } T = T', \\ 0 & \text{otherwise.} \end{cases}$$

The dependence on n is omitted in some places because it leads to very heavy notation. For example, the upwind cell $T_p^{\uparrow}$ also depends on n, and thus, $p \in \mathcal{P}_{k_{T_p^{\uparrow}}}$ depends on n.

To simplify the notation, we introduce the phase mobility, defined as follows, for all phases $p \in \mathcal{P}$,

$$\nu_p(\mathbf{u}) := \frac{\zeta_p(P,(\theta),\mathbf{C}_p)k_{\mathrm{r}_p}(S_p)}{\mu_p(P,(\theta),\mathbf{C}_p)}.$$

2.2.3.2 Mass and momentum equations discretization

For all $0 \le n \le N_{\mathrm{F}}$, all $T \in \mathcal{T}$, and for all $c \in \mathcal{C}$, we define the quantity $R_{c,T}^{\mathrm{mm}}(\mathbf{u}^n) := (R_{c,T}^{\mathrm{mm}})^n(\mathbf{u}^n)$ associated with the PDEs (2.1.5a)–(2.1.5b) as follows:

$$\begin{aligned} R_{c,T}^{\mathrm{mm}}(\mathbf{u}^n) :=& |T|\delta_t n_{c,T} \\ &+ \sum_{\sigma\in\mathcal{F}^{\mathrm{i}}\cap\mathcal{F}_T} \sum_{p\in\mathcal{P}_{k_{T_p^\uparrow}}\cap\mathcal{P}_c} \nu_p(\mathbf{u}_{T_p^\uparrow}^n) C_{c,p,T_p^\uparrow}^n F_{p,T,\sigma}(\mathbf{u}^n) \\ &+ \sum_{W\in\mathcal{W}^{\mathrm{p}}} \sum_{T'\in\mathcal{T}_W} \sum_{p\in\mathcal{P}_{k_W}\cap\mathcal{P}_c} \delta_{T,T'}\ \nu_p(\mathbf{u}_T^n) C_{c,p,T}^n F_{p,W,T}^{\oplus}(\mathbf{u}_T^n) \\ &- \sum_{W\in\mathcal{W}^{\mathrm{i}}} \sum_{T'\in\mathcal{T}_W} \sum_{p\in\mathcal{P}_{k_W}\cap\mathcal{P}_c} \delta_{T,T'}\ \nu_p(\mathbf{u}_W^n) C_{c,p,W}^n F_{p,W,T}^{\ominus}(\mathbf{u}_T^n) \\ &- q_{c,T}^n, \end{aligned} \tag{2.2.31}$$

where

1. For all components $c \in \mathcal{C}$ and all cells $T \in \mathcal{T}$, the discretization of the source term is
$$q_{c,T}^n = \frac{1}{\tau^n}\int_{I_n}\int_T q_c\ d\mathbf{x}\,dt; \tag{2.2.32}$$
2. For all $T \in \mathcal{T}$ and all $\sigma \in \partial T \setminus \partial\Omega$, $F_{p,T,\sigma}(\mathbf{u})$ is given by (2.2.13);
3. For all $W \in \mathcal{W}^{\mathrm{i}}$ and all $T \in \mathcal{T}_W$, $F_{p,W,T}$ is given by (2.2.24);
4. For all $W \in \mathcal{W}^{\mathrm{p}}$ and all $T \in \mathcal{T}_W$, $F_{p,W,T}$ is given by (2.2.25);
5. For all $p \in \mathcal{P}$, $T_p^\uparrow$ is the upwind cell.

Note that in a production well, the information comes from the cells that are exterior to the well in the neighborhood of the perforated cell. In contrast, in an injection well, the information comes from well data.

Observe that the residual $R_{c,T}^{\mathrm{mm}}$ depends, in general, on the unknowns associated with cells other than T owing, in particular, to the diffusive Darcy fluxes across internal faces; cf. Section 2.2.2.1. This dependence is precisely, for all $T \in \mathcal{T}$ and all $c \in \mathcal{C}$,

$$R_{c,T}^{\mathrm{mm}}(\mathbf{u}) = R_{c,T}^{\mathrm{mm}}(\{\mathbf{u}\}_{T\in\mathcal{S}_T}),$$

where we have set

$$\mathcal{S}_T^{\mathrm{mm}} := \bigcup_{\sigma\in\mathcal{F}_T\cap\mathcal{F}^{\mathrm{i}}} \mathcal{S}_\sigma^{\mathrm{mm}}.$$

Observe also that the no-flow condition across boundary faces has been accounted for in the definition of the residual by implicitly setting $F_{p,T,\sigma} = 0$ for all $\sigma \in \mathcal{F}^{\rm b}$ with $\sigma \subset \partial T$ and all $p \in \mathcal{P}$. We introduce the following notation for the vector of residuals:

$$\mathbf{R}^{\rm mm}(\mathbf{u}) := \big((R^{\rm mm}_{c,T}(\mathbf{u}))_{c\in\mathcal{C}}\big)_{T\in\mathcal{T}}.$$

As we have already mentioned, the field $\{k_T^n\}_{T\in\mathcal{T}}$ is itself a function of the local unknowns and considering (2.1.17), we assume that for all $T \in \mathcal{T}$, the Flash algorithm (described in Appendix A) gives:

$$k_T = \mathrm{Flash}(P_T, \mathbf{Z}_T), \qquad \text{with } \mathbf{Z}_T = \{Z_{T,c}\}_{c\in\mathcal{C}}, \tag{2.2.33}$$

where $Z_{T,c}$ represents the discrete counterpart of the molar fraction for the component c defined by (2.1.16).

2.2.3.3 Energy equation discretization

In the thermal case $I_\theta = 1$; for all $0 \le n \le N_{\rm F}$ and $T \in \mathcal{T}$, we define the residual $R^{\rm en}_T(\mathbf{u}) = (R^{\rm en}_T)^n(\mathbf{u}^n)$ associated with the energy conservation equation (2.1.10) by the following formula:

$$\begin{aligned}
R^{\rm en}_T(\mathbf{u}^n) :=& |T|\delta_t e_T^n \\
&+ \sum_{\sigma\in\mathcal{F}^{\rm i}\cap\mathcal{F}_T} G_{T,\sigma}(\mathbf{u}^n) \\
&+ \sum_{\sigma\in\mathcal{F}^{\rm i}\cap\mathcal{F}_T} \sum_{p\in\mathcal{P}_{k_{T_p^\uparrow}}} \nu_p(\mathbf{u}^n_{p,T_p^\uparrow}) h_p(\mathbf{u}^n_{p,T_p^\uparrow}) F_{p,T,\sigma}(\mathbf{u}^n) \\
&+ \sum_{W\in\mathcal{W}^{\rm p}} \sum_{T'\in\mathcal{T}_W} \delta_{T,T'} \sum_{p\in\mathcal{P}_{k_W}\cap\mathcal{P}_c} \nu_p(\mathbf{u}^n_{p,T}) h_p(\mathbf{u}^n_{p,T}) F^{\oplus}_{p,W,T}(\mathbf{u}^n_T) \\
&- \sum_{W\in\mathcal{W}^{\rm i}} \sum_{T'\in\mathcal{T}_W} \delta_{T,T'} \sum_{p\in\mathcal{P}_{k_W}\cap\mathcal{P}_c} \nu_p(\mathbf{u}^n_{p,W}) h_p(\mathbf{u}^n_{p,W}) F^{\ominus}_{p,W,T}(\mathbf{u}^n_T) \\
&+ \sum_{\sigma\in\mathcal{F}^{\rm b}\cap\mathcal{F}_T} G^{\rm b}_{T,\sigma}(\mathbf{u}^n) + \sum_{W\in\mathcal{W}^{\rm i}} \sum_{T'\in\mathcal{T}_W} \delta_{T,T'} G_{W,T}(\mathbf{u}^n_T) - Q^n_T,
\end{aligned} \tag{2.2.34}$$

where

1. For all $T \in \mathcal{T}$ and all $\sigma \in \partial T \setminus \partial\Omega$, $F_{p,T,\sigma}(\mathbf{u})$ is given by (2.2.13);
2. For all $W \in \mathcal{W}^{\rm i}$ and all $T \in \mathcal{T}_W$, $F_{p,W,T}$ is given by (2.2.24);
3. For all $W \in \mathcal{W}^{\rm p}$ and all $T \in \mathcal{T}_W$, $F_{p,W,T}$ is given by (2.2.25);

4. For all $T \in \mathcal{T}$ and all $\sigma \in \partial T \setminus \partial\Omega$, $G_{T,\sigma}(\mathbf{u})$ is given by (2.2.20);

5. For all $T \in \mathcal{T}$ and all $\sigma \in \partial T \cap \partial\Omega$, $G^{\rm b}_{T,\sigma}(\mathbf{u})$ is given by (2.2.23);

6. For all $W \in \mathcal{W}^{\rm i}$ and all $T \in \mathcal{T}_W$, $G_{W,T}$ is given by (2.2.27);

7. For all phases $p \in \mathcal{P}$ and a given law $\ell \in \{\zeta_p, k_{{\rm r}_p}, \mu_p, h_p\}$, we have introduced the notation $\ell_X := \ell(\mathbf{u}_X)$ for $X \in \{T, T_p^{\uparrow}, W\}$, where $T_p^{\uparrow}$ is defined as in (2.2.21);

8. For all cells $T \in \mathcal{T}$, we set
$$Q_T^n = \frac{1}{\tau^n}\int_{I_n}\int_T Q \; d\mathbf{x}\, dt.$$

As previously, we introduce the following notation for the vector of residuals:

$$\mathbf{R}^{\rm en}(\mathbf{u}) := (R_T^{\rm en}(\mathbf{u}))_{T\in\mathcal{T}}\,.$$

Below, we will also use

$$\mathcal{S}_T^{\rm en} := \bigcup_{\sigma\in\mathcal{F}_T\cap\mathcal{F}^{\rm i}} \mathcal{S}_\sigma^{\rm en},$$

and we have

$$\mathcal{S}_T := \bigcup_{\sigma\in\mathcal{F}_T\cap\mathcal{F}^{\rm i}} \{\mathcal{S}_\sigma^{\rm mm} \cup \mathcal{S}_\sigma^{\rm en}\}.$$

For all $T \in \mathcal{T}$ and letting $k := k_T$, the algebraic closure equations (2.1.6a)–(2.1.6c) are discretized in Section 2.2.2.5 and give the system of equations (2.2.28). Let us recall that we denote these closure equations in a synthetic form by the equations $\mathbf{D}_T(\mathbf{u}_T) = \mathbf{0}$, see (2.2.30c).

2.2.3.4 Global discrete problem

Finally, the discrete problem to be solved at every time step $1 \le n \le N_{\rm F}$ can be synthetically expressed as

$$\mathbf{R}^{\rm mm}(\mathbf{u}^n) = \mathbf{0} \in \mathbb{R}^{N_\mathcal{C}\times N_\mathcal{T}}, \tag{2.2.35a}$$

$$\mathbf{R}^{\rm en}(\mathbf{u}^n) = \mathbf{0} \in \mathbb{R}^{N_\mathcal{T}}, \qquad \text{if } I_\theta = 1, \tag{2.2.35b}$$

$$\mathbf{D}_T(\mathbf{u}_T^n) = \mathbf{0} \in \mathbb{R}^{N_{{\rm alg},T}}, \qquad \forall T \in \mathcal{T}, \tag{2.2.35c}$$

where $N_{{\rm alg},T}$ is the number of algebraic closure equations for the cell $T \in \mathcal{T}$ given by (setting $k := k_T^n$),

$$N_{{\rm alg},T} := 1 + N_{\mathcal{P}_k} + \sum_{p\in\mathcal{P}_k} N_{\mathcal{C}_p} - N_{\mathcal{C}_k}.$$

Problem (2.2.35) is a nonlinear system of algebraic equations that can be solved using standard techniques as described in the next section. An important remark is that, for all $T \in \mathcal{T}$, the closure equations express constraints involving only the unknowns $\mathbf{u}_T$. This allows, in particular, to eliminate some of the unknowns by solving local systems. In contrast, the equation linked to the PDEs, depend on the unknowns $\{\mathbf{u}_{T'}^n\}_{T' \in \mathcal{S}_T}$ defined in other cells.

Remark 2.12 (Time discretization)**.** Of course, the implicit time stepping that we have used here is not the only possible strategy. For instance, we can use the IMPES method: Implicit pressure, explicit saturations and compositions.

2.3 Solution strategy

The system (2.2.35) is nonlinear and a standard procedure consists in linearizing it by Newton's method. However, (2.2.35) is a large system and it is important to reduce its size. This can be achieved by pre-eliminating some of the unknowns. The idea is to use the closure equations (2.2.28) in order to:

- Select primary and secondary unknowns;
- By means of Newton's algorithm, compute the secondary unknowns in terms of the primary unknowns, thereby reducing the set of unknowns.

This procedure also improves the condition number of the global system.

Beforehand, we must determine the local context. Let $1 \leq n \leq N_{\mathrm{F}}$. During the computation, for all cells $T \in \mathcal{T}$, we evaluate the local context k_T^n according to (2.2.33). To do that we have to evaluate all the present phases, appearing or disappearing. Once the discrete contexts $\{k_T^n\}_{T \in \mathcal{T}}$ are known, we define the set of unknowns for all cells $T \in \mathcal{T}$ as in (2.2.4).

2.3.1 Reduction of the number of unknowns

Now we describe the reduction strategy. For a fixed n, $1 \leq n \leq N_{\mathrm{F}}$, let $\mathbf{u}^{n,0}$ be given (typically $\mathbf{u}^{n,0} = \mathbf{u}^{n-1}$) and let $\mathbf{u}^{n,i}$ denote the approximation of $\mathbf{u}^n$ computed at the i-th step of Newton's method. We introduce the increment:

$$\delta \mathbf{u}_T^{n,i} = \mathbf{u}_T^{n,i} - \mathbf{u}_T^{n,i-1},$$

for all cells $T \in \mathcal{T}$. The separation into primary and secondary unknowns is inspired by Newton's formula applied to (2.2.35c):

$$\left[\frac{\partial \mathbf{D}_T^n}{\partial \mathbf{u}_T^n}\left(\mathbf{u}_T^{n,i-1}\right)\right] \delta \mathbf{u}_T^{n,i} = -\mathbf{D}_T^n(\mathbf{u}_T^{n,i-1}). \tag{2.3.1}$$

To simplify, let $N_{\mathbf{u},T}$ denote the dimension of $\mathbf{u}_T^n$, and recall that $\mathbf{D}_T^n$ has $N_{\mathrm{alg},T}$ equations, with $N_{\mathrm{alg},T} < N_{\mathbf{u},T}$. Therefore, $\frac{\partial \mathbf{D}_T^n}{\partial \mathbf{u}_T^n}\left(\mathbf{u}_T^{n,i-1}\right)$ is an $N_{\mathrm{alg},T} \times N_{\mathbf{u},T}$ matrix. Assume for the moment that the set of unknowns $\mathcal{U}_T^{n,i}$ are separated into $N_{\mathbf{u}^{\mathfrak{P}},T}$ primary unknowns $\mathcal{U}_T^{\mathfrak{P},n,i}$ and $N_{\mathbf{u}^{\mathfrak{S}},T}$ secondary unknowns $\mathcal{U}_T^{\mathfrak{S},n,i}$ such that:

$$\mathcal{U}_T^{n,i} = \mathcal{U}_T^{\mathfrak{P},n,i} \cup \mathcal{U}_T^{\mathfrak{S},n,i}, \qquad \mathcal{U}_T^{\mathfrak{P},n,i} \cap \mathcal{U}_T^{\mathfrak{S},n,i} = \emptyset,$$

and

$$N_{\mathbf{u}^{\mathfrak{P}},T} + N_{\mathbf{u}^{\mathfrak{S}},T} = N_{\mathbf{u},T}.$$

We order the unknowns in each set and form the vectors $\mathbf{u}_T^{\mathfrak{P},n,i}$ and $\mathbf{u}_T^{\mathfrak{S},n,i}$.

Then we have the equality:

$$\left[\frac{\partial \mathbf{D}_T^n}{\partial \mathbf{u}_T^n}\left(\mathbf{u}_T^{n,i-1}\right)\right] \delta \mathbf{u}_T^{n,i} = \left[\frac{\partial \mathbf{D}_T^n}{\partial \mathbf{u}_T^{\mathfrak{P},n}}\left(\mathbf{u}_T^{n,i-1}\right)\right] \delta \mathbf{u}_T^{\mathfrak{P},n,i} + \left[\frac{\partial \mathbf{D}_T^n}{\partial \mathbf{u}_T^{\mathfrak{S},n}}\left(\mathbf{u}_T^{n,i-1}\right)\right] \delta \mathbf{u}_T^{\mathfrak{S},n,i}, \tag{2.3.2}$$

where:

- $\frac{\partial \mathbf{D}_T^n}{\partial \mathbf{u}_T^{\mathfrak{P},n}}\left(\mathbf{u}_T^{n,i-1}\right)$ is an $N_{\mathrm{alg},T} \times N_{\mathbf{u}^{\mathfrak{P}},T}$ matrix;
- $\frac{\partial \mathbf{D}_T^n}{\partial \mathbf{u}_T^{\mathfrak{S},n}}\left(\mathbf{u}_T^{n,i-1}\right)$ is an $N_{\mathrm{alg},T} \times N_{\mathbf{u}^{\mathfrak{S}},T}$ matrix.

When substituted into (2.3.1), this equality gives:

$$\left[\frac{\partial \mathbf{D}_T^n}{\partial \mathbf{u}_T^{\mathfrak{S},n}}\left(\mathbf{u}_T^{n,i-1}\right)\right] \delta \mathbf{u}_T^{\mathfrak{S},n,i} = -\left[\frac{\partial \mathbf{D}_T^n}{\partial \mathbf{u}_T^{\mathfrak{P},n}}\left(\mathbf{u}_T^{n,i-1}\right)\right] \delta \mathbf{u}_T^{\mathfrak{P},n,i} - \mathbf{D}_T^n(\mathbf{u}_T^{n,i-1}). \tag{2.3.3}$$

There remains to select the secondary unknowns $\mathbf{u}_T^{\mathfrak{S},n,i}$. The choice depends on the particular problem under consideration, and is not unique, but it is always dictated by the following considerations:

1. We have:

$$N_{\mathrm{alg},T} = N_{\mathbf{u}^{\mathfrak{S}},T}. \tag{2.3.4}$$

2. The square matrix $N_{\mathbf{u}^{\mathfrak{S}},T} \times N_{\mathbf{u}^{\mathfrak{S}},T}$ (i.e. $N_{\mathrm{alg},T} \times N_{\mathrm{alg},T}$) extracted from $\frac{\partial \mathbf{D}_T^n}{\partial \mathbf{u}_T^{\mathfrak{S},n}}\left(\mathbf{u}_T^{n,i-1}\right)$, acting on $\delta\mathbf{u}_T^{\mathfrak{S},n,i}$, must be invertible. We denote this matrix by $\widetilde{\frac{\partial \mathbf{D}_T^n}{\partial \mathbf{u}_T^{\mathfrak{S},n}}}\left(\mathbf{u}_T^{n,i-1}\right)$.

Taking this into account, we write:

$$\delta\mathbf{u}_T^{\mathfrak{S},n,i} = -\left(\widetilde{\frac{\partial \mathbf{D}_T^n}{\partial \mathbf{u}_T^{\mathfrak{S},n}}}\left(\mathbf{u}_T^{n,i-1}\right)\right)^{-1}\left[\mathbf{D}_T^n\left(\mathbf{u}_T^{n,i-1}\right) + \frac{\partial \mathbf{D}_T^n}{\partial \mathbf{u}_T^{\mathfrak{P},n}}\left(\mathbf{u}_T^{n,i-1}\right)\delta\mathbf{u}_T^{\mathfrak{P},n,i}\right], \tag{2.3.5}$$

and we substitute (2.3.5) into system (2.2.35) to eliminate all the secondary unknowns. Consequently we have a reduced system to solve because we can express equations in terms of primary unknowns. The system we have to solve becomes:

$$\begin{aligned} \mathbf{R}^{\mathrm{mm}}(\mathbf{u}^{\mathfrak{P}}) &= \mathbf{0} \in \mathbb{R}^{N_{\mathcal{C}} \times N_{\mathcal{T}}}, \\ \mathbf{R}^{\mathrm{en}}(\mathbf{u}^{\mathfrak{P}}) &= \mathbf{0} \in \mathbb{R}^{N_{\mathcal{T}}}, \qquad \text{if } I_\theta = 1. \end{aligned} \tag{2.3.6}$$

The system (2.3.6) is always nonlinear and we apply the Newton method to solve it. For all cells $T \in \mathcal{T}$, let us denote $\mathbf{R}^{\mathrm{mm}}$ or $\mathbf{R}^{\mathrm{en}}$ by $\mathbf{R}$. The incremental method consists in solving the following equation:

$$\sum_{T' \in \mathcal{S}_T} \frac{\partial \mathbf{R}_T^n}{\partial \mathbf{u}_{T'}^n}\left(\mathbf{u}^{n,i-1}\right)\delta\mathbf{u}_{T'}^{n,i} = -\mathbf{R}_T^n(\mathbf{u}^{n,i-1}). \tag{2.3.7}$$

Note that there is no cell index to $\mathbf{u}^{n,i-1}$ because $\mathbf{R}_T^n$ is not local to a single cell. By applying the decomposition into primary and secondary unknowns, we write:

$$\frac{\partial \mathbf{R}_T^n}{\partial \mathbf{u}_{T'}^n}\left(\mathbf{u}^{n,i-1}\right)\delta\mathbf{u}_{T'}^{n,i} = \frac{\partial \mathbf{R}_T^n}{\partial \mathbf{u}_{T'}^{\mathfrak{P},n}}\left(\mathbf{u}^{n,i-1}\right)\delta\mathbf{u}_{T'}^{\mathfrak{P},n,i} + \frac{\partial \mathbf{R}_T^n}{\partial \mathbf{u}_{T'}^{\mathfrak{S},n}}\left(\mathbf{u}^{n,i-1}\right)\delta\mathbf{u}_{T'}^{\mathfrak{S},n,i}. \tag{2.3.8}$$

The expression (2.3.5) for $\delta\mathbf{u}_{T'}^{\mathfrak{S},n,i}$ in (2.3.8) gives:

$$\begin{aligned} \frac{\partial \mathbf{R}_T^n}{\partial \mathbf{u}_{T'}^n}\left(\mathbf{u}^{n,i-1}\right)\delta\mathbf{u}_{T'}^{n,i} = \frac{\partial \mathbf{R}_T^n}{\partial \mathbf{u}_{T'}^{\mathfrak{P},n}}\left(\mathbf{u}^{n,i-1}\right)\delta\mathbf{u}_{T'}^{\mathfrak{P},n,i} \\ -\frac{\partial \mathbf{R}_T^n}{\partial \mathbf{u}_{T'}^{\mathfrak{S},n}}\left(\mathbf{u}^{n,i-1}\right)\left(\widetilde{\frac{\partial \mathbf{D}_{T'}^n}{\partial \mathbf{u}_{T'}^{\mathfrak{S},n}}}\left(\mathbf{u}_{T'}^{n,i-1}\right)\right)^{-1}\left[\mathbf{D}_{T'}^n\left(\mathbf{u}_{T'}^{n,i-1}\right)\right. \\ \left. + \frac{\partial \mathbf{D}_{T'}^n}{\partial \mathbf{u}_{T'}^{\mathfrak{P},n}}\left(\mathbf{u}_{T'}^{n,i-1}\right)\delta\mathbf{u}_{T'}^{\mathfrak{P},n,i}\right]. \end{aligned} \tag{2.3.9}$$

Thus, (2.3.7) becomes:

$$\begin{aligned}\sum_{T'\in\mathcal{S}_T}\Bigg(&\frac{\partial \mathbf{R}_T^n}{\partial \mathbf{u}_{T'}^{\mathfrak{P},n}}\left(\mathbf{u}^{n,i-1}\right)\delta\mathbf{u}_{T'}^{\mathfrak{P},n,i}\\ &-\frac{\partial \mathbf{R}_T^n}{\partial \mathbf{u}_{T'}^{\mathfrak{S},n}}\left(\mathbf{u}^{n,i-1}\right)\left(\widetilde{\frac{\partial \mathbf{D}_{T'}^n}{\partial \mathbf{u}_{T'}^{\mathfrak{S},n}}\left(\mathbf{u}_{T'}^{n,i-1}\right)}\right)^{-1}\left[\frac{\partial \mathbf{D}_{T'}^n}{\partial \mathbf{u}_{T'}^{\mathfrak{P},n}}\left(\mathbf{u}_{T'}^{n,i-1}\right)\delta\mathbf{u}_{T'}^{\mathfrak{P},n,i}+\mathbf{D}_{T'}^n\left(\mathbf{u}_{T'}^{n,i-1}\right)\right]\Bigg)\\ &=-\mathbf{R}_T^n(\mathbf{u}^{n,i-1}).\end{aligned} \tag{2.3.10}$$

In this expression, the term with factor $\mathbf{D}_{T'}^n\left(\mathbf{u}_{T'}^{n,i-1}\right)$ only involves known quantities and therefore can be passed to the right-hand side. The equation becomes:

$$\begin{aligned}\sum_{T'\in\mathcal{S}_T}\Bigg(&\frac{\partial \mathbf{R}_T^n}{\partial \mathbf{u}_{T'}^{\mathfrak{P},n}}\left(\mathbf{u}^{n,i-1}\right)\delta\mathbf{u}_{T'}^{\mathfrak{P},n,i}\\ &-\frac{\partial \mathbf{R}_T^n}{\partial \mathbf{u}_{T'}^{\mathfrak{S},n}}\left(\mathbf{u}^{n,i-1}\right)\left(\widetilde{\frac{\partial \mathbf{D}_{T'}^n}{\partial \mathbf{u}_{T'}^{\mathfrak{S},n}}\left(\mathbf{u}_{T'}^{n,i-1}\right)}\right)^{-1}\frac{\partial \mathbf{D}_{T'}^n}{\partial \mathbf{u}_{T'}^{\mathfrak{P},n}}\left(\mathbf{u}_{T'}^{n,i-1}\right)\delta\mathbf{u}_{T'}^{\mathfrak{P},n,i}\Bigg)\\ &=-\mathbf{R}_T^n(\mathbf{u}^{n,i-1})+\sum_{T'\in\mathcal{S}_T}\frac{\partial \mathbf{R}_T^n}{\partial \mathbf{u}_{T'}^{\mathfrak{S},n}}\left(\mathbf{u}^{n,i-1}\right)\left(\widetilde{\frac{\partial \mathbf{D}_{T'}^n}{\partial \mathbf{u}_{T'}^{\mathfrak{S},n}}\left(\mathbf{u}_{T'}^{n,i-1}\right)}\right)^{-1}\mathbf{D}_{T'}^n\left(\mathbf{u}_{T'}^{n,i-1}\right).\end{aligned} \tag{2.3.11}$$

Summarizing, we proceed along the following steps:

- Assume $\mathbf{u}^{\mathfrak{P},n,0}$ is given;
- At the beginning of Newton's iteration: For each cell T, precompute and store the matrix $\mathbf{A}_T$ and the vector $\mathbf{B}_T$ defined by:
  - $\mathbf{A}_T := \left(\widetilde{\frac{\partial \mathbf{D}_T^n}{\partial \mathbf{u}_T^{\mathfrak{S},n}}\left(\mathbf{u}_T^{n,i-1}\right)}\right)^{-1}\frac{\partial \mathbf{D}_T^n}{\partial \mathbf{u}_T^{\mathfrak{P},n}}\left(\mathbf{u}_T^{n,i-1}\right)\in\mathbb{R}^{N_{\mathbf{u}^{\mathfrak{S}},T},N_{\mathbf{u}^{\mathfrak{P}},T}}$;
  - $\mathbf{B}_T := \left(\widetilde{\frac{\partial \mathbf{D}_T^n}{\partial \mathbf{u}_T^{\mathfrak{S},n}}\left(\mathbf{u}_T^{n,i-1}\right)}\right)^{-1}\mathbf{D}_T^n\left(\mathbf{u}_T^{n,i-1}\right)\in\mathbb{R}^{N_{\mathbf{u}^{\mathfrak{S}},T}}$;
- For $k\geq 1$, compute a new algebraic solution $\mathbf{u}^{\mathfrak{P},n,i}$ from the previous known quantities $\mathbf{u}^{n,i-1}$ by solving a system of algebraic equations whose lines, for each cell T, are:

$$\begin{aligned}\sum_{T'\in\mathcal{S}_T}\left(\frac{\partial \mathbf{R}_T^n}{\partial \mathbf{u}_{T'}^{\mathfrak{P},n}}\left(\mathbf{u}^{n,i-1}\right)-\frac{\partial \mathbf{R}_T^n}{\partial \mathbf{u}_{T'}^{\mathfrak{S},n}}\left(\mathbf{u}^{n,i-1}\right)\mathbf{A}_{T'}\right)\delta\mathbf{u}_{T'}^{\mathfrak{P},n,i}\\ =-\mathbf{R}_T^n\left(\mathbf{u}^{n,i-1}\right)+\sum_{T'\in\mathcal{S}_T}\frac{\partial \mathbf{R}_T^n}{\partial \mathbf{u}_{T'}^{\mathfrak{S},n}}\left(\mathbf{u}^{n,i-1}\right)\mathbf{B}_{T'}.\end{aligned} \tag{2.3.12}$$

2.3.2 Greedy algorithm

In the general case, we need to determine at each time step and in each cell the sets of primary and secondary unknowns. To compute these sets, we propose here an algebraic procedure for separating the unknowns by means of a Greedy algorithm. The method is based on the fact that the closure equations are local in each cell.

For $1 \leq n \leq N_{\mathrm{F}}$ and $T \in \mathcal{T}$, let $\mathbf{u}_T^n$ be the vector of unknowns.

i) First, we impose that the pressure P is always a primary unknown.

ii) In the thermal case, we also impose that the temperature θ is always a primary unknown.

iii) Next, we select a secondary saturation; then the other saturations become primary unknowns. We use the volume conservation equation (2.2.28a) to relate this secondary saturation to those primary saturations.

iv) For constructing the Jacobian matrix $\frac{\partial \mathbf{D}_T^n}{\partial \mathbf{u}_T^{\mathfrak{S},n}}$, see (2.3.3), we extract, from the full matrix $\frac{\partial \mathbf{D}_T^n}{\partial \mathbf{u}_T^n}$, a set of $N_{\mathrm{alg},T}$ linearly independent columns that we orthonormalize according to the Greedy algorithm, briefly described in Algorithm 2.1.

Below, $\mathbf{V}_i$ denote the columns of $\frac{\partial \mathbf{D}_T^n}{\partial \mathbf{u}_T^n}$ before applying the algorithm, whereas $\mathbf{U}_i$ denote the columns of $\frac{\partial \mathbf{D}_T^n}{\partial \mathbf{u}_T^{\mathfrak{S},n}}$ after applying the algorithm.

2.4 Numerical experiments

2.4.1 Immiscible isothermal two-phase flow case

We consider here, as in Example 2.7, the immiscible isothermal two-phase flow five-spot case, and we use the cell-centered finite volume scheme presented in Section 2.2. The domain is a $200\mathrm{km} \times 200\mathrm{km}$ square and the mesh is conforming and uniform 20×20 (i.e. 400 cells). In this numerical experiment, we do not use space or time adaptation. We consider a homogeneous isotropic media, where the permeability constant is equal to $\kappa_T = 1.0 \cdot 10^{-13}$, for all $T \in \mathcal{T}$ and the permeability tensor is diagonal. The time step τ is equal to $2.16 \cdot 10^6$s and the final time is $4.32 \cdot 10^6$s. We prescribe the following initial conditions on the saturation. At the beginning of the simulation, the reservoir is full of oil, therefore $S_{\mathrm{o},T}^0 = 1$ for all $T \in \mathcal{T}$ and hence $S_{\mathrm{w},T}^0 = 0$ for all $T \in \mathcal{T}$. Since we propose to inject water through the injection wells,

Algorithm 2.1 Coarse view of a Greedy algorithm

Set $i := 1$

for $j = 1, \cdots, N_{\mathbf{u},T}$ **do**

$d(j) := \|\mathbf{V}_j\|$;

end for

$l(1) :=$ smallest J, such that $d(J) \leq d(j), \forall 1 \leq j \leq N_{\mathbf{u},T}$; //Choose the maximum norm

$\mathbf{U}_1 := \frac{\mathbf{V}_{l(1)}}{||\mathbf{V}_{l(1)}||}$; // Normalization

for $i = 2, \cdots, N_{\mathbf{u},T}$ **do**

Set $\mathbf{W} := \mathbf{V}_{i-1}$; // Permutation

$\mathbf{V}_{i-1} := \mathbf{V}_{l(i-1)}, \mathbf{V}_{l(i-1)} := \mathbf{W}$;

for $j = i, \cdots, N_{\mathbf{u},T}$ **do**

$d(j) := ||\mathbf{V}_j|| - \left(\sum_{l=1}^{i-1} (\mathbf{U}_l, \mathbf{V}_j)^2\right)^{1/2}$

end for

$l(i) :=$ smallest, J such that $d(J) \leq d(j), \forall i \leq j \leq N_{\mathbf{u},T}$;

$\mathbf{U}_i = \mathbf{V}_{l(i)} - \sum_{l=1}^{i-1} \left(\mathbf{U}_l, \mathbf{V}_{l(i)}\right) \mathbf{U}_l$;

$\mathbf{U}_i = \frac{\mathbf{U}_i}{||\mathbf{U}_i||}$; // Normalization

end for

we also impose that $S^0_{\mathrm{w},W,T} = 1$. Moreover, we have to enforce boundary conditions as in Equation (2.1.11):

$$\forall p \in \mathcal{P}, \qquad \overrightarrow{\mathbf{v}}_p \cdot \mathbf{n} = 0 \qquad \text{on } \partial\Omega.$$

The porosity in the domain is constant and equal to $\phi = 0.1$. To respect physical data, we have to define the laws governing the test case, described in Section 2.1.2. First, the viscosity of the oil $\mu_{\mathrm{o}}(P) = 5 \cdot 10^{-4}$ and the molar density $\zeta_{\mathrm{o}}(P) = 1$. Then for the wetting phase, the viscosity is equal to $\mu_{\mathrm{w}}(P) = 1 \cdot 10^{-4}$ and the molar density $\zeta_{\mathrm{w}}(P) = 1$. Finally, for both phases, the relative permeability $k_{\mathrm{r}_p} = S_p^2$. To simplify the resolution of the model, we consider no capillary pressure.

Following the idea presented in Section 2.3, the selection of primary and secondary unknowns is immediate and requires no computation. For each time step n, Newton iteration k, and cell T, we set:

$$\mathbf{u}_T^{\mathfrak{S},n,i} = \{S^{n,i}_{\mathrm{w},T}\} \text{ and then } \mathbf{u}_T^{\mathfrak{P},n,i} = \{P_T^{n,i}, S^{n,i}_{\mathrm{o},T}\}. \tag{2.4.1}$$

Note that this splitting is independent of T and n. The closure equations reduce to:

$$D_T^n(\mathbf{u}_T^{n,i}) = S^{n,i}_{\mathrm{o},T} + S^{n,i}_{\mathrm{w},T} - 1. \tag{2.4.2}$$

We suppress the index T and the superscript n, because in this case, the mapping $D_T^n(\mathbf{u}_T^{n,i})$ is independent of T and n. Thus

$$\frac{\partial D}{\partial \mathbf{u}_T^{\mathfrak{S}}}(\mathbf{u}_T^{n,i}) = 1 \text{ and } \frac{\partial D}{\partial \mathbf{u}_T^{\mathfrak{P}}}(\mathbf{u}_T^{n,i}) = [0\ 1], \tag{2.4.3}$$

and $\mathbf{A}_T \in \mathcal{M}_{1,2}$ and $B_T \in \mathbb{R}$ are given by:

$$\mathbf{A}_T = [0\ 1] \text{ and } B_T = S^{n,i}_{\mathrm{o},T} + S^{n,i}_{\mathrm{w},T} - 1. \tag{2.4.4}$$

In Figures 2.3–2.6, we present numerical results obtained by using the method described above. The figures represent the water saturation and the global pressure in each cell of the mesh $\mathcal{T}$ at different time steps n. The results were validated by an existing code at IFPEn.

2.4.2 Immiscible isothermal two-phase flow case in a heterogeneous isotropic media

As in the previous numerical experiment 2.4.1, we consider the same settings for the physical laws, the initial and boundary conditions and the locations of wells. We also use the same

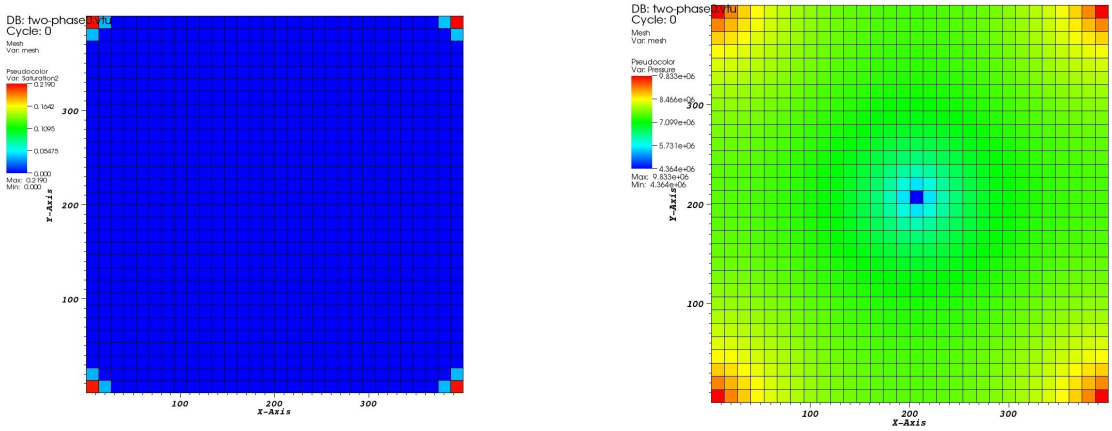

Figure 2.3: Water saturation (left) and global pressure (right) at the initial time.

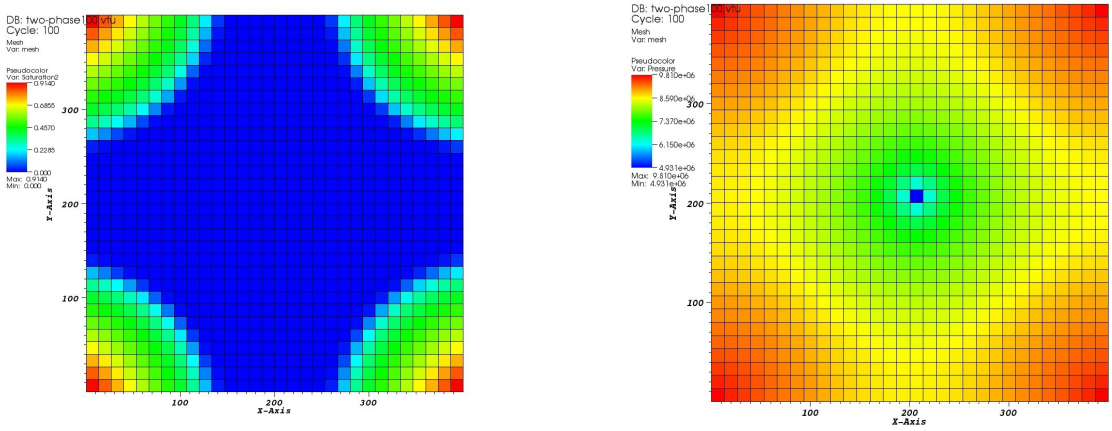

Figure 2.4: Water saturation (left) and global pressure (right) at time $2.16 \cdot 10^6$s.

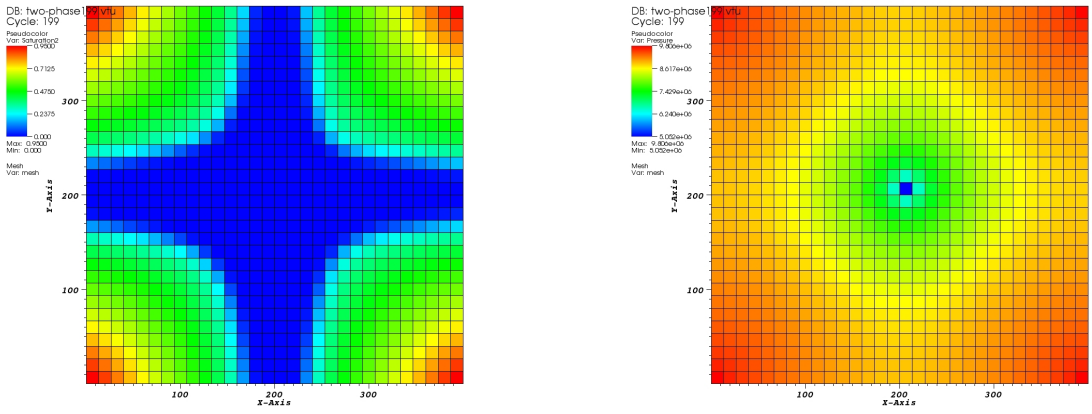

Figure 2.5: Water saturation (left) and global pressure (right) at time $4.32 \cdot 10^6$s.

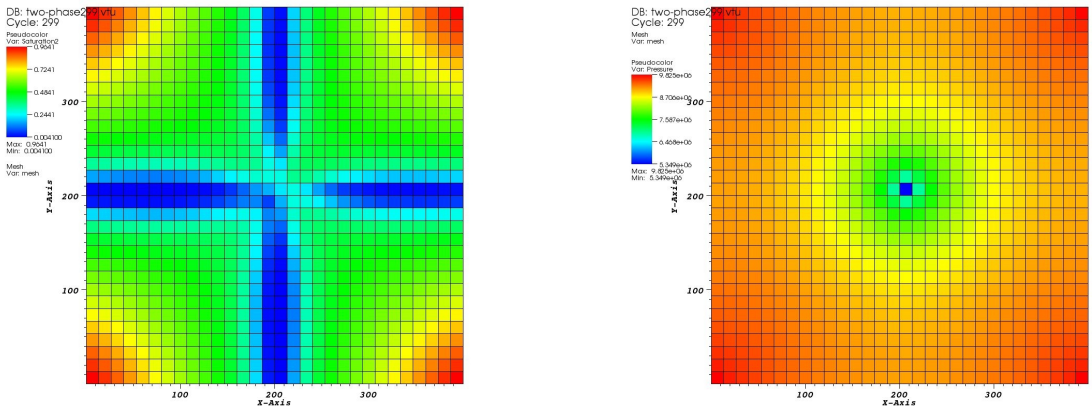

Figure 2.6: Water saturation (left) and global pressure (right) at time $6.48 \cdot 10^6$s.

discretization and the same numerical resolution of the discrete model. But now, the permeability is randomly distributed in the domain. More precisely, it is defined cell by cell as follows:

$$\mathbf{K}_T = v_T * 1.0 \cdot 10^{-13}, \qquad \text{for all } T \in \mathcal{T},$$

where $v_T \neq 0$ and $1.0 \cdot 10^{-2} \leq v_T \leq 1.0 \cdot 10^2$. Figure 2.7 depicts the permeability distribution

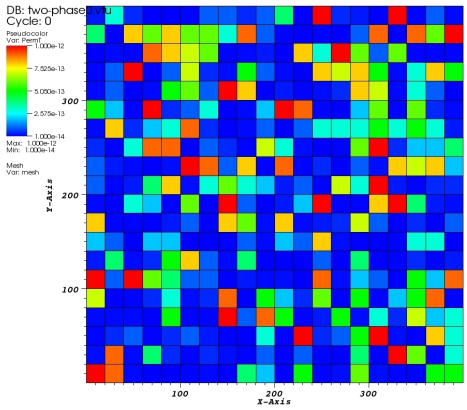

Figure 2.7: Random permeability in the domain.

in the domain. In Figures 2.8–2.11, we present numerical results obtained by using the method described above. The figures represent the water saturation and the global pressure in the mesh $\mathcal{T}$ at different time steps n.

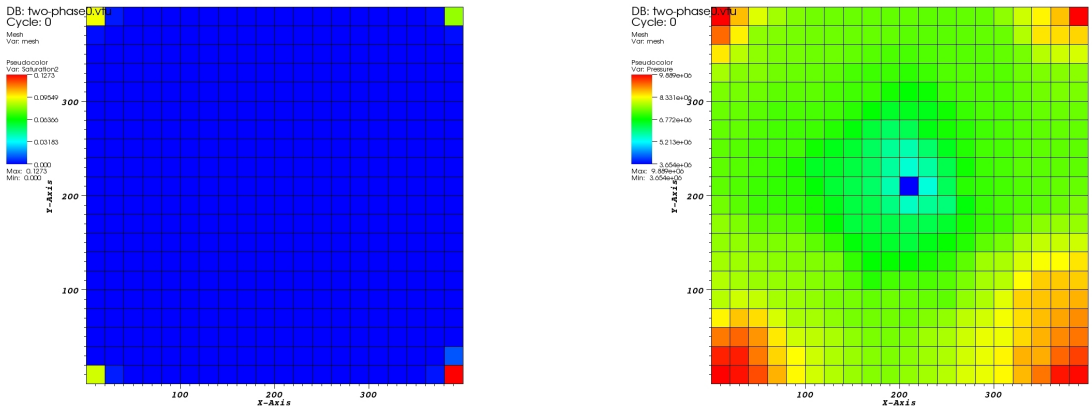

Figure 2.8: Water saturation (left) and global pressure (right) at the initial time.

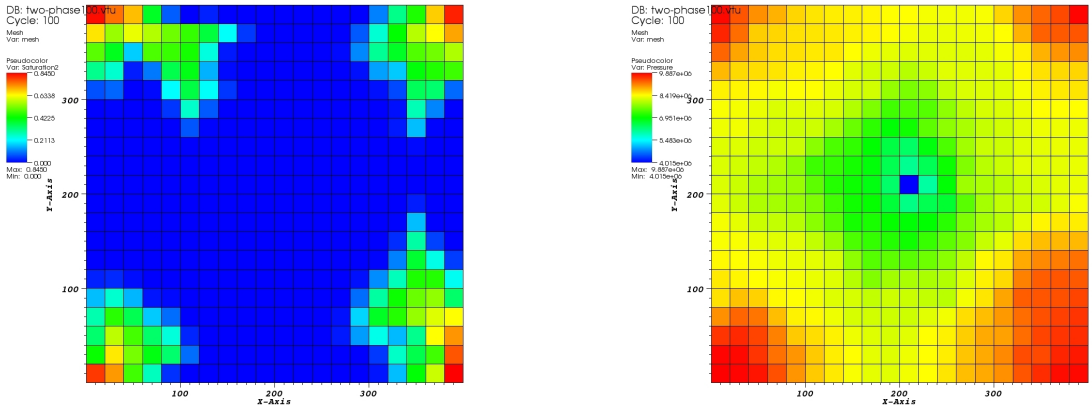

Figure 2.9: Water saturation (left) and global pressure (right) at time $2.16 \cdot 10^6$s.

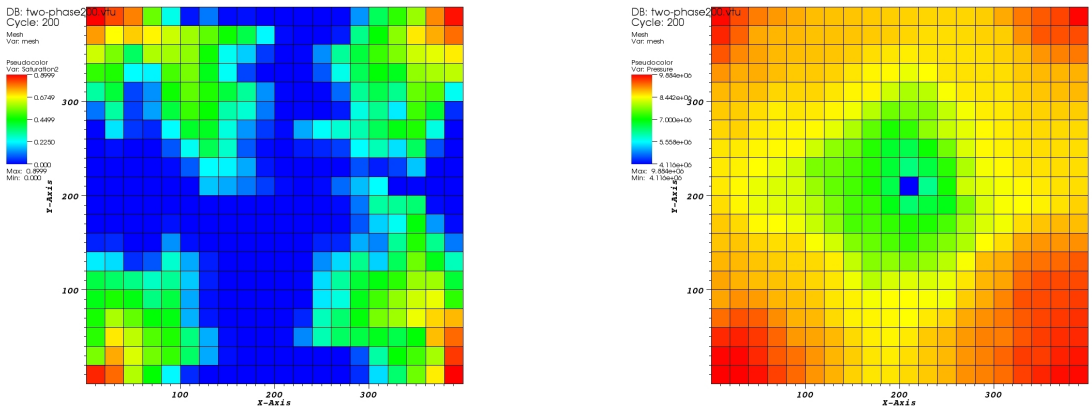

Figure 2.10: Water saturation (left) and global pressure (right) at time $4.32 \cdot 10^6$s.

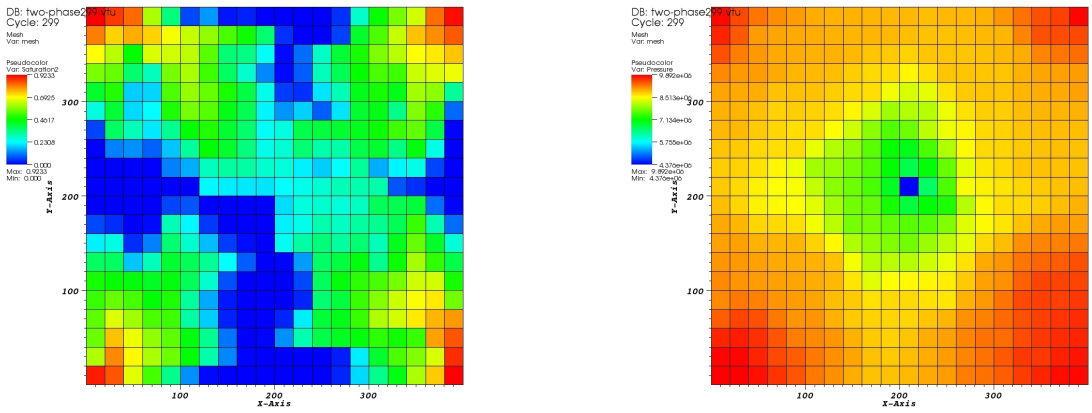

Figure 2.11: Water saturation (left) and global pressure (right) at time $6.48 \cdot 10^6$s.

CHAPTER 3

Variational formulation, discretization of a posteriori estimates for two-phase flows

This chapter is devoted to the simpler case of a two-phase Darcy flow: with two phases and two components, see Example 2.7. Albeit simpler than a general compositional model, it is still highly nonlinear, and its analysis, both theoretical and numerical is outside of the scope of this thesis. Here we propose to set the problem in a suitable equivalent variational formulation, and discretize it with a Two-Point Finite Volume scheme. The ideas are the same for the MPFV. Then we derive a posteriori error estimates for this scheme. The proposed estimators yield a fully computable upper bound for the norm of the residual error. The indicators allow to estimate separately and compare the Newton linearization and algebraic errors and the time and space discretization errors. This enables, in particular, to design a discretization algorithm so that all sources of error are suitably balanced. Namely, the linear and nonlinear solvers can be stopped as soon as the algebraic and linearization errors drop to a level at which they do not affect the overall error. This can lead to significant computational savings, since performing an excessive number of unnecessary iterations can be avoided. Similarly, the errors in space and in time can be balanced with the time steps and the mesh. The mesh adaptivity is presented in Chapter 4, but is not treated theoretically or in the context of a posteriori error analysis in the present work.

3.1 Introduction

3.1.1 Two-phase model

As in Section 2.1, $\Omega \subset \mathbb{R}^d$, where the dimension $d \geq 2$, denotes a bounded connected polygonal domain, with unit exterior normal $\mathbf{n}_\Omega$, and $t_\mathrm{F} > 0$ represents the final time of the modelization. Let $\Sigma := \Omega \times (0, t_\mathrm{F})$ denote the space-time domain. The two-phase flow involves two phases: The non-wetting phase o and the wetting phase w, so that

$$\mathcal{P} = \{\mathrm{o}, \mathrm{w}\}.$$

Each phase has one component and the phases are non-miscible with respect to each other. Therefore, the set of components corresponds here to the set of phases. Usually, the non-wetting phase represents the oil and the wetting phase the water. Thus, we can suppress some unknowns (molar fractions, contexts) and we do not need to express the components in the formulation: We can identify them with their phases, as it was detailed in Example 2.7.

We also neglect the gravity term and we only consider the isothermal case. Therefore, the unknowns are

$$\mathcal{U} = \{P, S_\mathrm{o}, S_\mathrm{w}\}.$$

To simplify the mathematical analysis we assume that the molar density ζ_p is constant. This allows to divide both sides of the governing equations by ζ_p; for simplicity we do not change the notation of the source term. We denote by

$$\nu_p(P, S_p) = \frac{k_{\mathrm{r}_p}(S_p)}{\mu_p(P)}$$

the phase mobility. The equations of the model can then be written as follows; cf. Example 2.7:

$$\begin{cases} \partial_t(\phi S_\mathrm{o}) + \nabla \cdot \big(\nu_\mathrm{o}(P, S_\mathrm{o}) \overrightarrow{\mathbf{v}}_\mathrm{o}(P, S_\mathrm{o})\big) = q_\mathrm{o} & \text{in } \Sigma, \\ \partial_t(\phi S_\mathrm{w}) + \nabla \cdot \big(\nu_\mathrm{w}(P, S_\mathrm{w}) \overrightarrow{\mathbf{v}}_\mathrm{w}(P, S_\mathrm{w})\big) = q_\mathrm{w} & \text{in } \Sigma, \\ S_\mathrm{o} + S_\mathrm{w} = 1 & \text{in } \Sigma, \end{cases} \tag{3.1.1}$$

where for $p \in \mathcal{P}$,

- The phase pressures satisfy (2.1.4), i.e.

$$P_p(P, S_p) = P + P_{\mathrm{c}_p}(S_p),$$

where P is the reference pressure

- The velocities satisfy Darcy's law (2.1.8)

$$\overrightarrow{\mathbf{v}}_p(P, S_p) = -\mathbf{K}(\nabla[P + P_{c_p}(S_p)]). \tag{3.1.2}$$

Remark 3.1 (Reference pressure)**.** In the practical situations considered here, for one of the phases (say p), the capillary pressure is set to zero, so that the reference pressure is equal to the phase pressure

$$P_p(P, S_p) = P.$$

The choice of the phase p is arbitrary, in the numerical experiments below, we consider the non-wetting phase.

Problem (3.1.1) is complemented by the initial conditions:

$$S_{\mathrm{o}}(\cdot, 0) = S_{\mathrm{o}}^0 \quad \text{in } \Omega, \tag{3.1.3}$$

and by no-flow homogeneous Neumann boundary conditions:

$$\overrightarrow{\mathbf{v}}_p(P, S_p) \cdot \mathbf{n}_\Omega = 0, \quad \text{in } \partial\Omega \times (0, t_{\mathrm{F}}), \text{for } \ p \in \mathcal{P}. \tag{3.1.4}$$

3.1.2 Regularity assumptions

In order to present some elements of mathematical analysis, we make the following assumptions:

i) The permeability tensor $\mathbf{K}$ is diagonal, piecewise constant, uniformly bounded and elliptic, see (2.1.9): There exists constants $c_{\mathbf{K}} > 0$ and $C_{\mathbf{K}} > 0$, such that

$$\forall \mathbf{x} \in \bar{\Omega}, 0 < c_{\mathbf{K}} \leq \kappa_{ii}(\mathbf{x}) \leq C_{\mathbf{K}}, \quad 1 \leq i \leq d, \tag{3.1.5}$$

where κ_{ii} are the diagonal terms of $\mathbf{K}$.

ii) For $p \in \{\mathrm{o}, \mathrm{w}\}$, the relative permeability k_{r_p} is uniformly bounded above, and the sum of the relative permeabilities is bounded away from zero on $\mathbb{R}$: There exists a constant $c_{k_{\mathrm{r}}} > 0$, such that

$$\forall x \in \mathbb{R}, c_{k_{\mathrm{r}}} \leq k_{\mathrm{r}}(x) + k_{\mathrm{r}}(x). \tag{3.1.6}$$

iii) For $p \in \{\mathrm{o}, \mathrm{w}\}$, the viscosity μ_p is uniformly bounded above and away from zero on $\mathbb{R}$: There exists constants $c_{\mu_p} > 0$ and $C_{\mu_p} > 0$, such that

$$\forall x \in \mathbb{R}, 0 < c_{\mu_p} \leq \mu_p(x) \leq C_{\mu_p}. \tag{3.1.7}$$

iv) The porosity $\phi > 0$ is constant in space and in time.

Note that the assumptions (*ii*) and (*iii*) imply that

v) For each $p \in \{\mathrm{o}, \mathrm{w}\}$, the phase mobility ν_p is uniformly bounded above, and the sum of the mobilities is bounded away from zero on $\mathbb{R}$:

$$\forall (x_1, x_2) \in \mathbb{R}^2, \max_{p \in \{\mathrm{o},\mathrm{w}\}} \left(\frac{c_{k_\mathrm{r}}}{c_\mu} \right) \leq \nu_\mathrm{o}(x_1, x_2) + \nu_\mathrm{w}(x_1, x_2). \tag{3.1.8}$$

3.1.3 Outline and some bibliographical notes

We discretize system (3.1.1)–(3.1.4) by cell-centered FV methods in space and the backward Euler scheme in time, see Chapter 2. The book of Eymard, Gallouët, and Herbin [49] details a lot of FV methods presented in this chapter.

The two-phase problem is historical in petroleum research and many relevant results are already known and established. Results such as existence of a solution, uniqueness and welled-posedness are established by, e.g., Kröner and Luckhaus in [54], by Chavent and Jaffré in [28], by Antontsev, Kazhikhov, and Monakhov in [12], by Arbogast in [13], by Chen in [29, 30, 31], by Cancès, Gallouët, and Porretta in [21], and by Amaziane, Jurak, and Keko in [10]. Discretization methods were studied by Douglas, Ewing, and Wheeler [42], by Russel and Wheeler [63], by Michel [58], and by Cancès [20]. Ideas for mesh adaptation were also developed by Saad and Zhang in [64] or by Chen and Ewing in [32]. The linearization method and linear solver techniques are discussed, e.g., by Vassilevski in [68] and by Wallis, Kendall, Little, and Nolen in [74].

There is a very extensive literature on a posteriori error estimates, starting with the work of Pousin and Rappaz [62], Verfürth [70], Chaillou and Suri [27, 26], Carstensen and Hu [24], and Carstensen, Hu, and Orlando [25]. Following the ideas of Jiránek, Strakoš, and Vohralík, [53], El Alaoui, Ern, and Vohralík [46], Ern and Vohralík [47, 48], Vohralík [71, 72], and Vohralík and Wheeler [73], we separate the estimates into contributions representing the *space discretization error*, *time discretization error*, *linearization error*, and *algebraic error*. One advantage of this approach is that it enables to save computing time, because at each time step, the linearization algorithm and the iterative algebraic solver can be stopped as soon as the corresponding errors no longer affect significantly the total error, and space and time errors can be balanced. This process permits to implement an adaptive algorithm for solving the problem. We develop here, in particular, the ideas of Di Pietro, Vohralík, and Widmer, [41].

The rest of this chapter is organized as follows. In Section 3.2, we present a weak formulation of the two-phase flow model. Then, we develop in Section 3.3, the discretization of the model and the solving strategy. In Section 3.4 and 3.5, we introduce the post-processing steps and some elements of the error analysis we make. We introduce different error indicators and we bound the residual error norm by these indicators. Numerical experiments are the content of Section 3.6.

3.2 The continuous setting

3.2.1 Function spaces

Let us first define some function spaces in a domain $\Omega \subset \mathbb{R}^d$, $d \geq 2$. We denote by $\mathcal{D}(\Omega)$ the space of $\mathcal{C}^\infty$ functions with compact support in Ω. Its dual space $\mathcal{D}'(\Omega)$ is the space of distributions in Ω. All derivatives below are taken in the sense of distributions, see [65].

We use the classical Sobolev space

$$H^1(\Omega) := \{\theta \in L^2(\Omega) \,|\, \nabla\theta \in [L^2(\Omega)]^d\},$$

which is a Hilbert space for the graph norm:

$$\|\theta\|_{H^1(\Omega)} := \left(\|\theta\|^2_{L^2(\Omega)} + \|\nabla\theta\|^2_{L^2(\Omega)}\right)^{1/2}.$$

Here, $\|\cdot\|_{L^2(\Omega)}$ denotes the norm of $L^2(\Omega)$:

$$\|\theta\|_{L^2(\Omega)} := \left(\int_\Omega |\theta(\mathbf{x})|^2 d\mathbf{x}\mathrm{d}\mathbf{x}\right)^{1/2}.$$

We recall that the functions of $H^1(\Omega)$ have a well-defined trace on the boundary $\partial\Omega$ of Ω, as well as on Lipschitz-continuous curves $\mathcal{S}$ contained in Ω. Loosely speaking, the functions of $H^1(\Omega)$ are continuous in the sense of traces.

The trace space of $H^1(\Omega)$ on the boundary $\partial\Omega$ is $H^{1/2}(\partial\Omega)$, which is a Hilbert space equipped with the norm:

$$\|f\|_{H^{1/2}(\partial\Omega)} = \left(\|f\|^2_{L^2(\partial\Omega)} + \int_{\partial\Omega}\int_{\partial\Omega} \frac{|f(\mathbf{x}) - f(\mathbf{y})|^2}{|\mathbf{x}-\mathbf{y}|^d}\mathrm{d}\mathbf{x}\mathrm{d}\mathbf{y}\right)^{1/2}. \tag{3.2.1}$$

This norm is equivalent to

$$\inf_{\theta\in H^1(\Omega);\theta|_{\partial\Omega}=f} \|\theta\|_{H^1(\Omega)}.$$

In a domain $\Omega \subset \mathbb{R}^d$, the quantity defined by (3.2.1) is a particular case of

$$\|f\|_{H^{1/2}(\Omega)} = \left(\|f\|^2_{L^2(\Omega)} + \int_\Omega \int_\Omega \frac{|f(\mathbf{x}) - f(\mathbf{y})|^2}{|\mathbf{x}-\mathbf{y}|^{d+1}} \mathrm{d}\mathbf{x}\mathrm{d}\mathbf{y} \right)^{1/2}$$

and

$$H^{1/2}(\Omega) = \{ f \in L^2(\Omega) \,|\, \|f\|_{H^{1/2}(\Omega)} < +\infty \}.$$

The factor d in (3.2.1) instead of $d+1$ arises from the fact that in $\mathbb{R}^d$, $\partial\Omega$ is of dimension $d-1$. The definition of $H^{1/2}(\partial\Omega)$ can be extended to any part, say Γ, of $\partial\Omega$ (or to any Lipschitz curve in Ω). Strictly speaking, the functions of $H^{1/2}(\Gamma)$ have no trace on $\partial\Gamma$, the boundary of Γ. Nevertheless, a weak form of zero trace on $\partial\Gamma$ can be prescribed by defining the space

$$H^{1/2}_{00}(\Gamma) = \left\{ f \in H^{1/2}(\Gamma) \,|\, \int_\Gamma \frac{|f(\mathbf{x})|^2}{d_{\partial\Gamma}(\mathbf{x})} \mathrm{d}\mathbf{x} < \infty \right\},$$

normed by

$$\|f\|_{H^{1/2}_{00}(\Gamma)} = \left(\|f\|^2_{H^{1/2}(\Gamma)} + \int_\Gamma \frac{|f(\mathbf{x})|^2}{d_{\partial\Gamma}(\mathbf{x})} \mathrm{d}\mathbf{x} \right)^{1/2},$$

where $d_{\partial\Gamma}(\mathbf{x})$ is the distance of $\mathbf{x}$ to $\partial\Gamma$. When Γ coincides with $\partial\Omega$, then

$$H^{1/2}(\Gamma) = H^{1/2}_{00}(\Gamma),$$

but if $|\partial\Omega \setminus \Gamma| > 0$, then $H^{1/2}_{00}(\Gamma)$ is a proper subspace of $H^{1/2}(\Gamma)$. The space $H^{1/2}_{00}(\Gamma)$ can also be defined as the trace space of functions $\theta \in H^1(\Omega)$ that vanish on $\partial\Omega \setminus \Gamma$.

The distinction between $H^{1/2}_{00}(\Gamma)$ and $H^{1/2}(\Gamma)$ is made clear by the fact that when the functions of $H^{1/2}_{00}(\Gamma)$ are extended by zero on $\partial\Omega$, the extended function belongs to $H^{1/2}(\partial\Omega)$, whereas this is not true in general for functions of $H^{1/2}(\Gamma)$.

For functions of $H^1(\Omega)$ with zero trace, we use $H^1_0(\Omega)$:

$$H^1_0(\Omega) = \{ \theta \in H^1(\Omega) \,|\, \theta|_{\partial\Omega} = 0 \}.$$

By virtue of the Poincaré inequality, valid in a bounded, Lipschitz domain: There exists a constant C such that

$$\forall \theta \in H^1_0(\Omega), \|\theta\|_{L^2(\Omega)} \leq C \|\nabla\theta\|_{L^2(\Omega)}, \tag{3.2.2}$$

we use the semi-norm

$$|\theta|_{H^1(\Omega)} = \|\nabla\theta\|_{L^2(\Omega)}$$

as an equivalent norm in $H^1_0(\Omega)$. Similarly, by virtue of the generalized Poincaré inequality, valid in a bounded, connected, Lipschitz domain: There exists another constant C such that

$$\forall \theta \in H^1(\Omega)/\mathbb{R}, \|\theta\|_{L^2(\Omega)} \leq C\|\nabla\theta\|_{L^2(\Omega)}, \tag{3.2.3}$$

we use the semi-norm

$$|\theta|_{H^1(\Omega)} = \|\nabla\theta\|_{L^2(\Omega)}$$

as an equivalent norm in $H^1(\Omega)/\mathbb{R}$. It is sometimes convenient to represent the classes of $H^1(\Omega)/\mathbb{R}$ by choosing the representative with zero mean-value, i.e. to replace $H^1(\Omega)/\mathbb{R}$ by $H^1(\Omega) \cap L^2_0(\Omega)$, where

$$L^2_0(\Omega) = \{f \in L^2(\Omega) \,|\, \langle f \rangle_\Omega = 0\}.$$

We shall also use the dual space $H^{-1}(\Omega)$ of $H^1_0(\Omega)$. It has the following interesting characterization: $\ell \in H^{-1}(\Omega)$ if and only if there exist functions $f_i \in L^2(\Omega)$, $0 \leq i \leq d$, such that

$$\ell = f_0 + \sum_{i=1}^{d} \frac{\partial f_i}{\partial x_i}. \tag{3.2.4}$$

We use $H^1(\Omega)$ to represent scalar quantities. For some vector-valued functions, such as velocities, it will be convenient to use the space $H(\mathrm{div}, \Omega)$:

$$H(\mathrm{div}, \Omega) := \{\boldsymbol{\theta} \in [L^2(\Omega)]^d \,|\, \nabla \cdot \boldsymbol{\theta} \in L^2(\Omega)\},$$

a Hilbert space for the graph norm:

$$\|\boldsymbol{\theta}\|_{H(\mathrm{div},\Omega)} := \left(\|\boldsymbol{\theta}\|^2_{L^2(\Omega)} + \|\nabla \cdot \boldsymbol{\theta}\|^2_{L^2(\Omega)} \right)^{1/2}.$$

It can be shown (cf. Girault and Raviart [51]) that the smooth functions are dense in $H(\mathrm{div}, \Omega)$, that the normal trace $\mathbf{u} \cdot \mathbf{n}$ is well-defined from $H(\mathrm{div}, \Omega)$ onto $H^{-1/2}(\partial\Omega)$ (where $H^{-1/2}(\partial\Omega)$ is the dual space of $H^{1/2}(\partial\Omega)$), and the following Green formula holds:

$$\forall \mathbf{u} \in H(\mathrm{div}, \Omega),\ \forall \theta \in H^1(\Omega), \quad \langle \mathbf{u} \cdot \mathbf{n}, \theta \rangle_{\partial\Omega} = \int_\Omega (\nabla \cdot \mathbf{u}) \theta \, d\mathbf{x} + \int_\Omega \mathbf{u} \cdot \nabla\theta \, d\mathbf{x}, \tag{3.2.5}$$

where $\langle \cdot, \cdot \rangle_{\partial\Omega}$ denotes the duality pairing between $H^{-1/2}(\partial\Omega)$ and $H^{1/2}(\partial\Omega)$. This permits to define the subspace

$$\mathbf{H}_0(\mathrm{div}, \Omega) = \{\boldsymbol{\theta} \in H(\mathrm{div}, \Omega) | \boldsymbol{\theta} \cdot \mathbf{n} = 0 \text{ on } \partial\Omega\}.$$

For functions depending on time and on space, it is convenient to separate the time from the space variable by considering functions of time with values in a Banach space, say H. More precisely, if (a,b) is an interval of time and $\|\cdot\|_H$ is the norm of H,

$$L^2(a,b;H) = \left\{ f \text{ measurable in } (a,b) \,|\, f(t) \in H, \text{ a.e. in } (a,b), \int_a^b \|f(t)\|_H^2 \mathrm{d}t < +\infty \right\},$$

equipped with the norm

$$\|f\|_{L^2(a,b;H)} = \left(\int_a^b \|f(t)\|_H^2 \mathrm{d}t \right)^{1/2}.$$

It is a Banach space if H is a Banach space and a Hilbert space if H is a Hilbert space.

Similarly, we define

$$\mathcal{C}^0([a,b];H) = \left\{ f \in \mathcal{C}^0([a,b]) \,|\, f(t) \in H, \ \forall t \in [a,b] \right\},$$

normed by

$$\|f\|_{\mathcal{C}^0([a,b];H)} = \sup_{t\in[a,b]} \|f(t)\|_H,$$

and

$$H^1(a,b;H) = \left\{ f \in L^2(a,b;H) \,|\, \frac{\partial f}{\partial t} \in L^2(a,b;H) \right\},$$

normed by

$$\|f\|_{H^1(a,b;H)} = \left(\|f\|^2_{L^2(a,b;H)} + \left\| \frac{\partial f}{\partial t} \right\|^2_{L^2(a,b;H)} \right)^{1/2}.$$

It can be shown that, up to a set of zero measure,

$$H^1(a,b;H) \subset \mathcal{C}^0([a,b];H), \tag{3.2.6}$$

with continuous embedding. This is a consequence of a more general result established for instance by Temam [66] or Girault and Raviart [51] [Theorem 1.1, Chapter V]. Therefore the functions of $H^1(a,b;H)$ have well-defined point values in time and we can define

$$H^1_0(a,b;H) = \{ f \in H^1(a,b;H) \,|\, f(a) = f(b) = 0 \}.$$

More generally, $\mathcal{D}(a,b;H)$ denotes the space of $\mathcal{C}^\infty$ functions of time with compact support in (a,b), and with values in H. As far as negative spaces in time are concerned, we shall use the space $H^{-1}(a,b;H)$ of all distributions of the form

$$f_0 + \frac{\partial}{\partial t} f_1, \text{ where } f_0 \text{ and } f_1 \text{ are functions of } L^2(a,b;H).$$

This definition is motivated by (3.2.4).

In this work we shall use the spaces $L^2(0,t_{\rm F};L^2(\Omega))$, $L^2(0,t_{\rm F};H^1(\Omega))$, $L^2(0,t_{\rm F};H^1_0(\Omega))$, $H^1(0,t_{\rm F};L^2(\Omega))$, $H^1(\Omega\times(0,t_{\rm F}))$. It is easy to check that

$$L^2(\Omega\times(0,t_{\rm F})) = L^2(0,t_{\rm F};L^2(\Omega)),$$
$$H^1(\Omega\times(0,t_{\rm F})) = \left\{ f\in L^2(0,t_{\rm F};H^1(\Omega)) \,|\, \frac{\partial f}{\partial t}\in L^2(0,t_{\rm F};L^2(\Omega))\right\}.$$

Owing to (3.2.6), we have the continuous embedding, up to a set of zero measure,

$$H^1(\Omega\times(0,t_{\rm F})) \subset H^1(0,t_{\rm F};L^2(\Omega)) \subset \mathcal{C}^0([0,t_{\rm F}];L^2(\Omega)). \tag{3.2.7}$$

3.2.2 A basic weak variational formulation

Let us set problem (3.1.1)–(3.1.4) into a weak variational form. We assume that all variables involved are distributions. For the phase $p\in\{\mathrm{o},\mathrm{w}\}$, consider the equation

$$\phi\partial_t(S_p) + \nabla\cdot\big(\nu_p(P,S_p)\overrightarrow{\mathbf{v}}_p(P,S_p)\big) = q_p. \tag{3.2.8}$$

The idea is to write its left-hand side in full divergence form. For this it is convenient to work in the space-time cylinder of $\mathbb{R}^{d+1}$

$$\Sigma = \Omega\times(0,t_{\rm F}),$$

and define the vector $\mathbf{U}_p\in\mathbb{R}^{d+1}$ by

$$\begin{aligned} U_{p,i} &= \nu_p(P,S_p)u_p^i(P,S_p), \quad \text{for } 1\le i\le d,\\ U_{p,d+1} &= \phi S_p, \end{aligned} \tag{3.2.9}$$

where u_p^i is the i-th component of $\overrightarrow{\mathbf{v}}_p$. Then, the divergence of $\mathbf{U}_p$ in Σ is:

$$\begin{aligned} \nabla_\Sigma\cdot\mathbf{U}_p &= \frac{\partial U_{p,1}}{\partial x_1} + \cdots + \frac{\partial U_{p,d}}{\partial x_d} + \frac{\partial U_{p,d+1}}{\partial t}\\ &= \nabla\cdot(\nu_p(P,S_p)\overrightarrow{\mathbf{v}}_p(P,S_p)) + \frac{\partial}{\partial t}(\phi S_p), \end{aligned}$$

and the first two lines of (3.1.1) read

$$\nabla_\Sigma\cdot\mathbf{U}_p = q_p, \text{ for } p\in\{\mathrm{o},\mathrm{w}\}. \tag{3.2.10}$$

Let us search for the unknowns P and S_p such that:

$$\nu_p(P,S_p)\overrightarrow{\mathbf{v}}_p(P,S_p)\in[L^2(\Sigma)]^d \quad \text{and} \quad \phi S_p\in L^2(\Sigma). \tag{3.2.11}$$

It follows from assumption (iv) that (3.2.11) is equivalent to

$$\nu_p(P,S_p)\overrightarrow{\mathbf{v}}_p(P,S_p) \in [L^2(\Sigma)]^d \quad \text{and} \quad S_p \in L^2(\Sigma), \tag{3.2.12}$$

whence $\mathbf{U}_p \in L^2(\Sigma)^{d+1}$.

Now we consider Darcy's law (3.1.2). As P and $P_{\mathrm{c}_p}(S_p)$ are distributions and $\mathbf{K}$ is not smooth, the right-hand side of (3.1.2) is not defined. However, by assumption (i), $\mathbf{K}$ is invertible and therefore we interpret (3.1.2) by

$$\nabla[P + P_{\mathrm{c}_p}(S_p)] = -\mathbf{K}^{-1}\overrightarrow{\mathbf{v}}_p(P,S_p). \tag{3.2.13}$$

Since we look for $\overrightarrow{\mathbf{v}}_p(P,S_p)$ in $[L^2(\Sigma)]^d$ and $\mathbf{K}^{-1}$ belongs to $L^\infty(\Sigma)$, (3.2.13) implies that $\nabla(P + P_{\mathrm{c}_p}(S_p)) \in L^2(\Sigma)^d$. According to Nečas [59] or to Amrouche and Girault in [11], this implies that the distribution $P + P_{\mathrm{c}_p}(S_p)$ belongs to $L^2(0,t_{\mathrm{F}};H^1(\Omega))$. This gives a meaning to the Darcy law (3.1.2) and it stems for the above considerations that the natural space for $P + P_{\mathrm{c}_p}(S_p)$ is $L^2(0,t_{\mathrm{F}};H^1(\Omega))$, $p \in \{\mathrm{o},\mathrm{w}\}$.

Now let us assume that

$$q_p \in L^2(\Sigma). \tag{3.2.14}$$

Then (3.2.10) implies that

$$\nabla_\Sigma \cdot \mathbf{U}_p \in L^2(\Sigma), \tag{3.2.15}$$

and with (3.2.11) this gives $\mathbf{U}_p \in H(\mathrm{div}_\Sigma,\Sigma)$, where div_Σ refers to the full divergence operator $\nabla_\Sigma\cdot$ in Σ. Therefore $\mathbf{U}_p \cdot \mathbf{n}_\Sigma \in H^{-1/2}(\partial\Sigma)$, where $\mathbf{n}_\Sigma$ denotes the exterior normal to $\partial\Sigma$, and the Green formula (3.2.5) gives

$$\forall v_p \in H^1(\Sigma), \quad \int_\Sigma (\nabla_\Sigma \cdot \mathbf{U}_p) v_p \,\mathrm{d}\mathbf{x}\,\mathrm{d}t + \int_\Sigma \mathbf{U}_p \cdot \nabla_\Sigma v_p \,\mathrm{d}\mathbf{x}\,\mathrm{d}t = \langle \mathbf{U}_p \cdot \mathbf{n}_\Sigma, v_p\rangle_{\partial\Sigma}, \tag{3.2.16}$$

where $\nabla_\Sigma v_p$ denotes the full gradient of v_p in Σ. Let us expand the terms of (3.2.16). First, by (3.2.10) and (3.2.14),

$$\int_\Sigma (\nabla_\Sigma \cdot \mathbf{U}_p) v_p \,\mathrm{d}\mathbf{x}\,\mathrm{d}t = \int_0^{t_{\mathrm{F}}} \int_\Omega q_p \, v_p \,\mathrm{d}\mathbf{x}\,\mathrm{d}t. \tag{3.2.17}$$

Next, by definition of $\mathbf{U}_p$, we have

$$\int_\Sigma \mathbf{U}_p \cdot \nabla_\Sigma v_p \,\mathrm{d}\mathbf{x}\,\mathrm{d}t = \int_0^{t_{\mathrm{F}}} \int_\Omega \nu_p(P,S_p)\overrightarrow{\mathbf{v}}_p(P,S_p) \cdot \nabla_{\mathbf{x}} v_p \mathrm{d}\mathbf{x}\,\mathrm{d}t + \int_0^{t_{\mathrm{F}}} \int_\Omega \phi S_p \frac{\partial v_p}{\partial t} \mathrm{d}\mathbf{x}\,\mathrm{d}t, \tag{3.2.18}$$

where $\nabla_{\mathbf{x}}$ denotes the gradient with respect to $\mathbf{x}$. Finally to treat the boundary term, we must clarify the normal vector $\mathbf{n}_\Sigma$ and the product $\mathbf{U}_p \cdot \mathbf{n}_\Sigma$:

$$\begin{aligned}
&\text{On } \Omega \times \{t = 0\},\ \mathbf{n}_\Sigma = (\vec{0}, -1) \ \text{ and } \mathbf{U}_p \cdot \mathbf{n}_\Sigma = -\phi S_p(\cdot, 0) = -\phi S_p^0,\\
&\text{on } \Omega \times \{t = t_{\mathrm{F}}\},\ \mathbf{n}_\Sigma = (\vec{0}, 1),\ \text{and } \mathbf{U}_p \cdot \mathbf{n}_\Sigma = \phi S_p(\cdot, t_{\mathrm{F}}),\\
&\text{on } \partial\Omega \times (0, t_{\mathrm{F}}),\ \mathbf{n}_\Sigma = (\mathbf{n}_\Omega, 0) \text{ and } \mathbf{U}_p \cdot \mathbf{n}_\Sigma = \nu_p(P, S_p)\overrightarrow{\mathbf{v}}_p(P, S_p) \cdot \mathbf{n}_\Omega = 0,
\end{aligned} \tag{3.2.19}$$

using (3.1.3) and (3.1.4). The only unknown quantity in the right-hand side of (3.2.19) is $S_p(\cdot, t_{\mathrm{F}})$. In order to eliminate it, we prescribe on the test function v_p the additional condition:

$$v_p(\cdot, t_{\mathrm{F}}) = 0, \text{ a.e. in } \Omega.$$

This condition makes sense, in view of (3.2.7). Thus, assuming that the initial data has the regularity

$$S_p^0 \in L^2(\Omega), \tag{3.2.20}$$

the boundary term reduces to

$$\langle \mathbf{U}_p \cdot \mathbf{n}_\Sigma, v_p \rangle_{\partial\Sigma} = -\int_\Omega \phi S_p^0 v_p(\mathbf{x}, 0) \mathrm{d}\mathbf{x}. \tag{3.2.21}$$

Hence, substituting (3.1.2) into (3.1.1), we deduce the weak variational formulation: Find $P, S_{\mathrm{o}}, S_{\mathrm{w}}$ satisfying

$$S_p \in L^2(\Sigma), \quad P + P_{\mathrm{c}_p}(S_p) \in L^2(0, t_{\mathrm{F}}; H^1(\Omega)), \quad p \in \{\mathrm{o}, \mathrm{w}\}, \tag{3.2.22}$$

such that for all $v_p \in H^1(\Sigma)$and $v_p(\cdot, t_{\mathrm{F}}) = 0$,

$$\begin{aligned}
\int_0^{t_{\mathrm{F}}} \int_\Omega \nu_p(P, S_p) \mathbf{K} \nabla_{\mathbf{x}} [P + P_{\mathrm{c}_p}(S_p)] \cdot \nabla_{\mathbf{x}} v_p \mathrm{d}\mathbf{x}\, \mathrm{d}t - \int_0^{t_{\mathrm{F}}} \int_\Omega \phi S_p \frac{\partial v_p}{\partial t} \mathrm{d}\mathbf{x}\, \mathrm{d}t \\
= \int_0^{t_{\mathrm{F}}} \int_\Omega q_p\, v_p\, \mathrm{d}\mathbf{x}\, \mathrm{d}t + \int_\Omega \phi S_p^0 v_p(\cdot, 0)\, \mathrm{d}\mathbf{x}, \qquad \text{for } p = \mathrm{o}, \mathrm{w},
\end{aligned} \tag{3.2.23a}$$

$$S_{\mathrm{o}} + S_{\mathrm{w}} = 1. \tag{3.2.23b}$$

This formulation is weakly equivalent to the original problem (3.1.1)–(3.1.4) in the following sense.

Proposition 3.2 (Equivalence)**.** *Let q_p belong to $L^2(\Sigma)$ and S_p^0 belong to $L^2(\Omega)$. Then, every solution P, S_{o}, and S_{w} of problem* (3.1.1)–(3.1.4)*, with $S_p \in L^2(\Sigma)$, $P + P_{\mathrm{c}_p}(S_p) \in$*

$L^2(0,t_{\rm F};H^1(\Omega))$, *for* $p\in\{{\rm o},{\rm w}\}$ *satisfies* (3.2.23). *Conversely every solution* P, $S_{\rm o}$, *and* $S_{\rm w}$ *of* (3.2.23), *with* $S_p\in L^2(\Sigma)$, $P+P_{{\rm c}_p}(S_p)\in L^2(0,t_{\rm F};H^1(\Omega))$, *for* $p\in\{{\rm o},{\rm w}\}$ *solves* (3.1.1)–(3.1.4), *by setting*

$$\overrightarrow{\mathbf{v}}_p(P,S_p)=-\mathbf{K}(\nabla[P+P_{{\rm c}_p}(S_p)]),\quad \textit{for } p={\rm o},{\rm w}. \tag{3.2.24}$$

Proof. If the system (3.1.1)–(3.1.4) admits a solution satisfying (3.2.22), then (3.2.23b) is given by the problem and the above argument gives immediately (3.2.23a).

Conversely, let P, $S_{\rm o}$, and $S_{\rm w}$ be a solution of (3.2.23), with the regularity (3.2.22). By setting

$$\overrightarrow{\mathbf{v}}_p(P,S_p)=-\mathbf{K}(\nabla[P+P_{{\rm c}_p}(S_p)]),\quad \text{for } p={\rm o},{\rm w},$$

the assumption (iv) and property (v) (that stems from assumptions (ii) and (iii)), imply that $\nu_p(P,S_p)\overrightarrow{\mathbf{v}}_p(P,S_p)\in[L^2(\Sigma)]^d$, and $\phi S_p\in L^2(\Sigma)$. If we choose $v_p\in\mathcal{D}(\Sigma)$, the equality (3.2.23a) with the notation (3.2.9) reduces in the sense of distributions to

$$-\langle\mathbf{U}_p,\nabla_\Sigma v_p\rangle_\Sigma=\int_0^{t_{\rm F}}\int_\Omega q_p v_p\,{\rm d}\mathbf{x}\,{\rm d}t\quad\Leftrightarrow\quad\langle\nabla_\Sigma\cdot\mathbf{U}_p,v_p\rangle_\Sigma=\langle q_p,v_p\rangle_\Sigma,$$

where $\langle\cdot,\cdot\rangle_\Sigma$ denotes the duality pairing between $\mathcal{D}'(\Sigma)$ and $\mathcal{D}(\Sigma)$. This is equivalent to

$$\nabla_\Sigma\cdot\mathbf{U}_p=q_p,$$

and this equality holds a.e. in Σ since $q_p\in L^2(\Sigma)$.Therefore, the first two equalities of (3.1.1) are satisfied, and the regularity of q_p implies that $\nabla_\Sigma\cdot\mathbf{U}_p\in L^2(\Sigma)$; thus

$$\mathbf{U}_p\in H({\rm div}_\Sigma,\Sigma).$$

Then, Green's formula (3.2.5) gives for all $v_p\in H^1(\Sigma)$ with $v_p(\cdot,t_{\rm F})=0$ in Ω:

$$-\int_0^{t_{\rm F}}\int_\Omega\mathbf{U}_p\nabla_\Sigma v_p\,{\rm d}\mathbf{x}\,{\rm d}t+\langle\mathbf{U}_p\cdot\mathbf{n}_\Sigma,v_p\rangle_{\partial\Sigma}=\int_0^{t_{\rm F}}\int_\Omega q_p v_p\,{\rm d}\mathbf{x}\,{\rm d}t.$$

To recover the initial condition, let us choose $v_p\in H^1(0,t_{\rm F};\mathcal{D}(\Omega))$ with $v_p(\cdot,t_{\rm F})=0$ in Ω. On the one hand,

$$v_p|_{\Omega\times\{t=0\}}=v_p(\cdot,0)$$

belongs to $\mathcal{D}(\Omega)$ and on the other hand, as $\mathbf{U}_p\cdot\mathbf{n}_\Sigma$ is in $H^{-1/2}(\partial\Sigma)$, its restriction to $\Omega\times\{t=0\}$ is a distribution on Ω. With this choice of test function v_p, the boundary term reduces to

$$\langle\mathbf{U}_p\cdot\mathbf{n}_\Sigma,v_p\rangle_{\partial\Sigma}=-{}_{\mathcal{D}'(\Omega)}\langle\phi S_p(\cdot,0),v_p(\cdot,0)\rangle_{\mathcal{D}(\Omega)}.$$

Comparing with (3.2.23), this yields

$$\forall\ \psi \in \mathcal{D}(\Omega), \quad \phi\langle S_p(\cdot,0),\psi\rangle = \phi\int_\Omega S_p^0\psi\,\mathrm{d}\mathbf{x},$$

i.e. in the sense of the distributions

$$S_p(\cdot,0) = S_p^0,$$

and since the right-hand side belongs to $L^2(\Omega)$, this equality holds a.e. in Ω and $\phi S_p(\cdot,0)$ belongs to $L^2(\Omega)$, whence (3.1.3).

Similarly, to recover the boundary condition on $\partial\Omega$, we choose $v_p \in H_0^1(0,t_\mathrm{F};H^1(\Omega))$ and observe that the restriction of $\mathbf{U}_p\cdot\mathbf{n}_\Sigma$ to $\partial\Omega\times(0,t_\mathrm{F})$ belongs to $H^{-1}(0,t_\mathrm{F};H^{-1/2}(\partial\Omega))$. With this choice of v_p, the boundary term reduces to

$$\langle \nu_p(P,S_p)\overrightarrow{\nabla}_p(P,S_p)\cdot\mathbf{n}_\Omega, v_p\rangle,$$

where $\langle\cdot,\cdot\rangle$ denotes the duality pairing between the spaces $H^{-1}(0,t_\mathrm{F};H^{-1/2}(\partial\Omega))$ and $H_0^1(0,t_\mathrm{F};H^{1/2}(\partial\Omega))$. Comparing with (3.2.23), we obtain in the sense of distributions, $\nu_p(P,S_p)\overrightarrow{\nabla}_p(P,S_p)\cdot\mathbf{n}_\Omega = 0$, whence (3.1.4). □

3.2.3 A second variational formulation

The previous variational formulation is adequate when the solution is rough, but is not convenient for proposing a numerical discretization, because it involves the derivative in time of the test function.

Assume that the data are such that (3.2.14) and (3.2.20) hold. When the solution of Problem (3.1.1) is smoother, more precisely, if in addition to (3.2.11) it is such that

$$\frac{\partial}{\partial t}(S_p) \in L^2(\Sigma),$$

i.e., $\phi S_p \in H^1(0,t_\mathrm{F};L^2(\Omega))$, then by integrating by parts the term involving $\frac{\partial v_p}{\partial t}$ and using the initial condition (3.1.3), the variational formulation (3.2.23) yields

$$\int_0^{t_\mathrm{F}}\int_\Omega \partial_t(\phi S_p)\,v_p\,\mathrm{d}\mathbf{x}\mathrm{d}t + \int_0^{t_\mathrm{F}}\int_\Omega \nu_p(P,S_p)\mathbf{K}\nabla_\mathbf{x}[P+P_{\mathrm{c}_p}(S_p)]\cdot\nabla_\mathbf{x}v_p\,\mathrm{d}\mathbf{x}\mathrm{d}t = \int_0^{t_\mathrm{F}}\int_\Omega q_p\,v_p\,\mathrm{d}\mathbf{x}\mathrm{d}t, \tag{3.2.25}$$

for all $v_p \in H^1(\Sigma)$, with $v_p(\cdot,t_\mathrm{F}) = 0$. In particular, this is also true for all $v_p \in H^1(0,t_\mathrm{F};H^1(\Omega))$ with $v_p(\cdot,t_\mathrm{F}) = 0$. But $H_0^1(0,t_\mathrm{F};H^1(\Omega))$ is dense in $L^2(0,t_\mathrm{F};H^1(\Omega))$ because $H_0^1(a,b)$ is dense

in $L^2(a,b)$. Thus, as all three terms in (3.2.25) are scalar products of $L^2(0,t_{\rm F})$, (3.2.25) also holds for all v_p in $L^2(0,t_{\rm F};H^1(\Omega))$. Therefore the dependence of v_p on t and the integration with respect to t in (3.2.25) are unnecessary and (3.2.25) reduces to (for simplicity we revert to the notation of gradient without index $\mathbf{x}$):

Find $P, S_{\rm o}, S_{\rm w}$, such that $P + P_{{\rm c}_p}(S_p) \in L^2(0,t_{\rm F};H^1(\Omega))$, for $p \in \{{\rm o},{\rm w}\}$, and $S_p \in H^1(0,t_{\rm F};L^2(\Omega))$ such that

$$\begin{aligned}\forall v_p \in H^1(\Omega),\ \int_\Omega \partial_t(\phi S_p)\, v_p \,{\rm d}\mathbf{x} + \int_\Omega \nu_p(P,S_p)\mathbf{K}\nabla[P + P_{{\rm c}_p}(S_p)]\cdot\nabla v_p \,{\rm d}\mathbf{x} \\ = \int_\Omega q_p\, v_p \,{\rm d}\mathbf{x},\ \text{a.e. in } (0,t_{\rm F}),\ \text{for } p\in\{{\rm o},{\rm w}\},\end{aligned} \tag{3.2.26}$$

with the initial condition (3.1.3)

$$S_p(\cdot,0) = S_p^0, \quad p \in \{{\rm o},{\rm w}\},$$

and the pore volume conservation condition (3.2.23b)

$$S_{\rm o} + S_{\rm w} = 1.$$

It is easy to prove that this problem is equivalent to (3.1.1)–(3.1.4).

Proposition 3.3 (Equivalence)**.** *Let q_p belong to $L^2(\Sigma)$ and S_p^0 to $L^2(\Omega)$. Every solution P, $S_{\rm o}$, and $S_{\rm w}$ of problem* (3.1.1)–(3.1.4) *with*

$$S_p \in H^1(0,t_{\rm F};L^2(\Omega)) \text{ and } P + P_{{\rm c}_p}(S_p) \in L^2(0,t_{\rm F};H^1(\Omega)), \quad p\in\{{\rm o},{\rm w}\}, \tag{3.2.27}$$

solves (3.2.26), (3.1.3), *and* (3.2.23b).

Conversely, each solution of (3.2.26), (3.1.3), *and* (3.2.23b) *satisfying* (3.2.27) *solves* (3.1.1)–(3.1.4), *setting* (3.1.2).

Remark 3.4 (Pressure regularity)**.** As mentioned in Remark 3.1, in practical situations, $P_{{\rm c}_{p_1}}(S_{p_1}) = 0$ for one of the phases p_1. This implies that the reference pressure P belongs to $L^2(0,t_{\rm F};H^1(\Omega))$ and next that $P_{{\rm c}_{p_2}}(S_{p_2})$ belongs to $L^2(0,t_{\rm F};H^1(\Omega))$ for the other phase p_2. If this is not done, then the above mathematical assumptions are not sufficient to conclude that $P \in L^2(0,t_{\rm F};H^1(\Omega))$. We then need additional hypothesis on the capillary pressure $P_{{\rm c}_{p_2}}(S_{p_2})$.

3.3 The discrete setting for two-phase flow

In this section, we briefly recall the discretization methods adapted to the present two-phase flow model. As in Section 2.2.2, we discretize problem (3.1.1)–(3.1.4) with a two-point FV method and phase-by-phase upwind. We introduce some notation used in this section and we present mathematical tools useful for estimating the error made during the computation. Furthermore, we give some ideas for constructing these tools during a computation.

3.3.1 Discrete spaces

Here we describe the discrete spaces in dimensions $d = 2$ or 3. Because we use a TPFV method, we are restricted to rectangular cells. Thus, we introduce a mesh $\mathcal{T}$ of Ω made of rectangular cells, when $d = 2$ or rectangular parallelepipeds, when $d = 3$. Let T denote a generic cell, with diameter h_T bounded by the mesh size $h_{\mathcal{T}}$. We assume that the mesh is shape regular (see Ciarlet [34]), more precisely there exists a constant C_1 independent of h, such that

$$\forall T \in \mathcal{T}, \frac{h_T}{\rho_T} \leq C_1,$$

where ρ_T is the diameter of the largest ball contained in T. For simplicity, the mesh is supposed to be fixed during the computation, not evolving in time. Moreover we suppose that the mesh follows the discontinuities of $\mathbf{K}$ (recall that $\mathbf{K}$ is piecewise constant), so that $\mathbf{K}$ is constant in each cell.

We define the broken Sobolev space on $\mathcal{T}$

$$H^1(\mathcal{T}) := \{u \in L^2(\Omega) \,|\, \forall T \in \mathcal{T}, u_{|_T} \in H^1(T)\}.$$

On each rectangular parallelepiped T, we define the discrete space

$$\mathbf{RTN}(T) := [\mathbb{P}_0(T)]^d + \mathbb{M}_0(T)\mathbf{x},$$

where $\mathbb{M}_0(T)$ is a diagonal matrix, constant in T:

$$\mathbb{M}_0(T) = \begin{bmatrix} c_1 & & \\ & \ddots & \\ & & c_d \end{bmatrix},$$

whith $c_1, \cdots, c_d$ in $\mathbb{R}$. The normal components of functions in these spaces are constant on each face and consequently have one degree of freedom per face. It is well-known that by

choosing these degrees of freedom on all faces of the mesh, we can represent discrete functions in $\mathbf{H}(\mathrm{div},\Omega)$. If, in addition, these degrees of freedom vanish on all boundary faces of the mesh, then the discrete functions belong to $\mathbf{H}_0(\mathrm{div},\Omega)$. These two properties motivate the definition of the Raviart–Thomas–Nédélec spaces:

$$\mathbf{RTN}(\mathcal{T}) := \{\mathbf{v}_h \in \mathbf{H}(\mathrm{div},\Omega) \,|\, \mathbf{v}_{h|T} \in \mathbf{RTN}(T),\, \forall T \in \mathcal{T}\},$$

and

$$\mathbf{RTN}_0(\mathcal{T}) := \mathbf{RTN}(\mathcal{T}) \cap \mathbf{H}_0(\mathrm{div},\Omega).$$

Remark 3.5 (Triangular mesh)**.** On simplices the discrete space are defined by

$$\mathbf{RTN}(T) := [\mathbb{P}_0(T)]^d + \mathbb{P}_0(T)\mathbf{x}.$$

3.3.2 Unknowns

From now on, we restrict the discussion to $d = 2$. As specified in Section 2.2.1, the discrete unknowns take one value per cell. Let $0 \le n \le N_{\mathrm{F}}$ be the index of the discrete times t^n and T the cells of the mesh $\mathcal{T}$. For each n and each T, $k_T^n \in \mathcal{K}$ is the context in the cell T, at time t^n, and $\mathcal{P}_{k_T^n}$ is the set of phases present in T at time t^n. For the two-phase flow considered here,

$$\mathcal{K} = \{\{\mathrm{o}\},\{\mathrm{w}\},\{\mathrm{o},\mathrm{w}\}\},$$

and

$$\mathcal{P} = \{\mathrm{o},\mathrm{w}\}.$$

For all $T \in \mathcal{T}$ and $0 \le n \le N_{\mathrm{F}}$, we have seen in Section 2.4.1 that the set of unknowns (2.2.4) reduces to:

$$\mathcal{U}_T^n := \left\{P_T^n, \{S_{p,T}^n\}_{p\in\mathcal{P}_{k_T^n}}\right\}, \tag{3.3.1}$$

where we recall that at time t^n, in the cell T:

- P_T^n denotes the reference pressure;
- $\{S_{p,T}^n\}_{p\in\mathcal{P}_{k_T^n}}$ is the set of saturations for the phases present in the context k_T^n.

Once a fixed ordering is chosen, for all $0 \le n \le N_{\mathrm{F}}$ and $T \in \mathcal{T}$, the unknowns in the set $\mathcal{U}_T^n$ can be ordered in a local vector $\mathbf{u}_T^n$, which when assembled forms the global vector $\mathbf{u}^n := (\mathbf{u}_T^n)_{T\in\mathcal{T}}$.

For $p \in \mathcal{P}$, let P_h^n, respectively $S_{p,h}^n$, be the piecewise constant functions such that

$$\forall T \in \mathcal{T}, \quad P_h^n\,|_T = P_T^n, \qquad \text{respectively} \qquad S_{p,h}^n\,|_T = S_{p,T}^n.$$

3.3.3 Equations

Throughout this work we restrict ourselves to fully implicit time discretizations. As stated in Section 2.2.2, we consider a discretization of problem (3.1.1)–(3.1.4) based on two-point FV fluxes and phase-upwind. Phase-upwind is developed in Section 2.2.2.2. We describe below the two-point FV method for the flux discretization.

The discrete conservation equations are obtained by first integrating the first two lines of (3.1.1) on a control volume $T \in \mathcal{T}$ and on a time interval I_n. For each phase $p \in \{\mathrm{o}, \mathrm{w}\}$, this gives:

$$\int_T \left(\phi\partial_t(S_p) + \nabla\cdot\left(\nu_p(P,S_p)\overrightarrow{\mathbf{v}}_p(P,S_p)\right)\right) d\mathbf{x} = \int_T q_p \; d\mathbf{x}. \tag{3.3.2}$$

Using the Gauss theorem and the same notation as in the discretization of the compositional model, after division by τ^n, the second term in (3.3.2) has the expression:

$$\frac{1}{\tau^n}\sum_{\sigma\in\mathcal{F}_T}\left(\int_{I_n}\int_\sigma \left(\nu_p(P,S_p)\overrightarrow{\mathbf{v}}_p(P,S_p)\right)\cdot\mathbf{n}\,\mathrm{d}\Gamma\,dt\right), \tag{3.3.3}$$

where for $p \in \mathcal{P}$, the velocities satisfy Darcy's law

$$\overrightarrow{\mathbf{v}}_p(P,S_p) = -\mathbf{K}(\nabla[P + P_{\mathrm{c}_p}(S_p)]).$$

Considering that the discrete saturations are constant in each cell, the first term is then discretized by:

$$\phi\frac{|\,T\,|}{\tau^n}\left(S_{p,T}^n - S_{p,T}^{n-1}\right). \tag{3.3.4}$$

To discretize the flux $\overrightarrow{\mathbf{v}}_p\cdot\mathbf{n}$, we proceed as in Section 2.2.2. For all $\sigma \in \mathcal{F}^{\mathrm{i}}$ and all $T \in \mathcal{T}_\sigma$ (the set of elements sharing the face σ) we approximate the diffusive flux of any phase $p \in \bigcup_{T\in\mathcal{T}_\sigma}\mathcal{P}_{k_T}$ by:

$$F_{p,T,\sigma}^n(\mathbf{u}^n) = F_{p,T,\sigma}^n\left(\{\mathbf{u}_{T'}^n\}_{T'\in\mathcal{S}_\sigma^{\mathrm{mm}}}\right) = \sum_{T'\in\mathcal{S}_\sigma^{\mathrm{mm}}}\tau_{T'}^\sigma P_{p,T'}^n, \tag{3.3.5}$$

where τ_T^σ is the transmissibility coefficient. Its definition is an extension of (2.2.14) that takes into account that $\mathbf{K}$ is not proportional to the identity. More precisely, let $\sigma = T_1 \cap T_2$,

where T_1 and T_2 are two neighboring cells, ordered as T_1 and T_2, with the normal vector to σ oriented from T_1 to T_2. Let $\mathbf{x}_\sigma$ and $\mathbf{x}_T$ denote the barycenter of the face σ and of the cell T, respectively, and let $\mathbf{t}_\sigma$ denote a unit tangent vector to σ. Then the coefficient τ_T^σ is defined by replacing κ_T by the scalar product $(\mathbf{K}_T\mathbf{t}_\sigma, \mathbf{t}_\sigma)$ in (2.2.14):

$$\tau_T^\sigma = |\sigma| \frac{\frac{(\mathbf{K}_T\mathbf{t}_\sigma,\mathbf{t}_\sigma)}{\|\mathbf{x}_\sigma-\mathbf{x}_T\|_2}\frac{(\mathbf{K}_{T'}\mathbf{t}_\sigma,\mathbf{t}_\sigma)}{\|\mathbf{x}_\sigma-\mathbf{x}_{T'}\|_2}}{\frac{(\mathbf{K}_T\mathbf{t}_\sigma,\mathbf{t}_\sigma)}{\|\mathbf{x}_\sigma-\mathbf{x}_T\|_2}+\frac{(\mathbf{K}_{T'}\mathbf{t}_\sigma,\mathbf{t}_\sigma)}{\|\mathbf{x}_\sigma-\mathbf{x}_{T'}\|_2}} \text{ and } \tau_{T'}^\sigma = -\tau_T^\sigma, \tag{3.3.6}$$

where we recall that the negative sign of $\tau_{T'}^\sigma$ enforces the conservativity of fluxes.

Furthermore, due to the no-flow boundary condition, for $p \in \mathcal{P}$ and $T \in \mathcal{T}$ such that $\mathcal{F}_T$ contains a face σ^b on the boundary, we impose:

$$F_{p,T,\sigma^b}^n(\mathbf{u}_T^n) = 0. \tag{3.3.7}$$

Finally, let us present the fluxes on the (injection or production) wells. As gravity is neglected, the definition of the fluxes is the same regardless of production or injection. For all wells $W \in \mathcal{W}$, all perforated cells $T \in \mathcal{T}_W$, and all phases $p \in \mathcal{P}_{k_T}$, the flux of the phase p between the system and the perforation in T is approximated by:

$$F_{p,W,T}^n(\mathbf{u}_T^n) = \tau_T^W \left(P_{p,T}^n - P_{W,T}^n\right), \tag{3.3.8}$$

where τ_T^W is the so-called *production index* of the well and is known from the data.

In order to discretize the factor ν_p appearing in (3.3.3), we use the values at the previous time t^{n-1} of the pressure and saturation, P^{n-1} and S_p^{n-1}, taken in the upwind cell for the phase p, denoted by $T_p^\uparrow$. The upwind cell, defined in (2.2.21), is determined by the flux $F_{p,T,\sigma}^{n-1}$ that has been computed at the previous time step. The discretization of the source term is:

$$q_{p,T}^n = \frac{1}{\tau^n}\int_{I_n}\int_T q_p \,\mathrm{d}\mathbf{x}\,\mathrm{d}t. \tag{3.3.9}$$

Collecting these discretizations, the discrete version of (3.3.2) reads:

$$\phi\,\frac{|\,T\,|}{\tau^n}\left(S_{p,T}^n - S_{p,T}^{n-1}\right) + \sum_{\sigma\in\mathcal{F}_T}\left(\nu_p(P_{T_p^\uparrow}^{n-1}, S_{p,T_p^\uparrow}^{n-1})F_{p,T,\sigma}^n\right) = q_{p,T}^n. \tag{3.3.10}$$

Remark 3.6 (An initial pressure)**.** When $n = 1$, Formula (3.3.10) requires an initial pressure P^0 that is not part of the data, since only the saturation is given (and required) at initial time. This numerical difficulty that occurs frequently is usually by-passed by defining an artificial initial pressure using the hydrostatic pressure formula. For an incompressible fluid,

a reasonably good estimate can be made by assuming a constant density ρ throughout the liquid (the same assumption cannot be made for a gaseous fluid). Also, since the height of the fluid column is often reasonably small compared to the radius of the Earth, one can neglect the variation of the gravity g. Under these circumstances, the initial hydrostatic pressure P^0 is a constant given by the simple formula

$$P^0 = \rho g z, \tag{3.3.11}$$

where in two dimensions z is the depth of the position of the domain. This is the formula used to complement (3.3.10).

As mentioned previously, in a two-phase flow, the separation into primary and secondary unknowns is trivial. Indeed the secondary unknown is the wetting phase saturation, $S^n_{\mathrm{w},T}$. It is eliminated from (3.3.10) by means of the local volume conservation equation:

$$S^n_{\mathrm{w},T} = 1 - S^n_{\mathrm{o},T}.$$

The reduced set of unknowns is

$$\tilde{\mathbf{u}}^n := \{\mathbf{P}^n, \mathbf{S}^n_{\mathrm{o}}\},$$

where $\mathbf{P}^n = \{P^n_T\}_{T\in\mathcal{T}}$ and $\mathbf{S}^n_{\mathrm{o}} = \{S^n_{\mathrm{o},T}\}_{T\in\mathcal{T}}$. This simplifies the set of equations (3.3.10) and we can write the following discrete model for the equation (3.1.1)–(3.1.4): for all $1 \le n \le N_{\mathrm{F}}$, all $T \in \mathcal{T}$, and all $p \in \{\mathrm{o}, \mathrm{w}\}$:

$$\begin{cases} R^n_{\mathrm{o},T}(\tilde{\mathbf{u}}^n) := \phi \dfrac{|\,T\,|}{\tau^n}\,(S^n_{\mathrm{o},T} - S^{n-1}_{\mathrm{o},T}) + \displaystyle\sum_{\sigma\in\mathcal{F}_T} \nu_{\mathrm{o}}(P^{n-1}_{T_p^\uparrow}, S^{n-1}_{\mathrm{o},T_p^\uparrow}) F^n_{\mathrm{o},T,\sigma} - q^n_{\mathrm{o},T} = 0, \\ R^n_{\mathrm{w},T}(\tilde{\mathbf{u}}^n) := \phi \dfrac{|\,T\,|}{\tau^n}\,(S^{n-1}_{\mathrm{o},T} - S^n_{\mathrm{o},T}) + \displaystyle\sum_{\sigma\in\mathcal{F}_T} \nu_{\mathrm{w}}(P^{n-1}_{T_p^\uparrow}, 1 - S^{n-1}_{\mathrm{o},T_p^\uparrow}) F^n_{\mathrm{w},T,\sigma} - q^n_{\mathrm{w},T} = 0, \end{cases} \tag{3.3.12}$$

where the discretization of the initial condition is given by

$$S^0_{\mathrm{o},T} := \frac{(S^0_{\mathrm{o}}, 1)_T}{|T|}, \tag{3.3.13}$$

and

$$P^0_T := \frac{(P^0, 1)_T}{|T|}, \tag{3.3.14}$$

where P^0 is given by (3.3.11). Henceforth, for a function ξ we express $\xi(S_{\mathrm{w}})$ by $\widehat{\xi}(S_{\mathrm{o}}) := \xi(1 - S_{\mathrm{o}})$. As a consequence,

$$\widehat{k}_{\mathrm{rw}}(S_{\mathrm{o}}) = k_{\mathrm{rw}}(1 - S_{\mathrm{o}}),\ \widehat{P}_{\mathrm{cw}}(S_{\mathrm{o}}) = P_{\mathrm{cw}}(1 - S_{\mathrm{o}}),\ \text{and}\ \widehat{\overrightarrow{\mathbf{v}}}_{\mathrm{w}} = \overrightarrow{\mathbf{v}}_{\mathrm{w}}(P, 1 - S_{\mathrm{o}}).$$

Then (3.3.12) becomes:

$$\begin{cases} R^n_{\mathrm{o},T}(\tilde{\mathbf{u}}^n) := \phi\dfrac{|\,T\,|}{\tau^n}\,(S^n_{\mathrm{o},T} - S^{n-1}_{\mathrm{o},T}) + \displaystyle\sum_{\sigma\in\mathcal{F}_T} \nu_{\mathrm{o}}(P^{n-1}_{T_p^{\uparrow}}, S^{n-1}_{\mathrm{o},T_p^{\uparrow}})F^n_{\mathrm{o},T,\sigma} - q^n_{\mathrm{o},T} = 0, \\ R^n_{\mathrm{w},T}(\tilde{\mathbf{u}}^n) := \phi\dfrac{|\,T\,|}{\tau^n}\,(S^{n-1}_{\mathrm{o},T} - S^n_{\mathrm{o},T}) + \displaystyle\sum_{\sigma\in\mathcal{F}_T} \widehat{\nu}_{\mathrm{w}}(P^{n-1}_{T_p^{\uparrow}}, S^{n-1}_{\mathrm{o},T_p^{\uparrow}})F^n_{\mathrm{w},T,\sigma} - q^n_{\mathrm{w},T} = 0. \end{cases} \tag{3.3.15}$$

The problem is thus to find the unknowns $\tilde{\mathbf{u}}^n$ such that:

$$\begin{cases} \mathbf{R}^n_{\mathrm{o}}(\tilde{\mathbf{u}}^n) = \mathbf{0}, \\ \mathbf{R}^n_{\mathrm{w}}(\tilde{\mathbf{u}}^n) = \mathbf{0}, \end{cases} \tag{3.3.16}$$

where for each $p \in \{\mathrm{o},\mathrm{w}\}$, the symbol $\mathbf{R}^n_p$ stands for the vector equations for all cells $T \in \mathcal{T}$. Summing up, on each mesh element, we have to solve two nonlinear systems in two unknowns.

3.3.4 Solving strategy

Problem (3.3.16) is a system of nonlinear algebraic equations that we linearize at each time step by Newton's algorithm; The linear algebraic system at each Newton's step is solved by an algebraic solver. In the numerical experiments, we use a GMRES solver, but the method described here applies to an arbitrary iterative algebraic solver. The algorithm consists of two loops at each time step: An outer Newton's loop and an inner algebraic loop that is executed at each Newton's iteration. At a fixed time t^n, $1 \le n \le N_{\mathrm{F}}$, let $\tilde{\mathbf{u}}^{n,0}$ be given (typically, $\tilde{\mathbf{u}}^{n,0} := \tilde{\mathbf{u}}^{n-1}$).

For all cells $T \in \mathcal{T}$ and all phases $p \in \{\mathrm{o},\mathrm{w}\}$, the linear system of the i-th Newton iteration is:

$$\sum_{T'\in\mathcal{T}} \frac{\partial R^n_{p,T}}{\partial \tilde{\mathbf{u}}^n_{T'}}(\tilde{\mathbf{u}}^{n,i-1})\tilde{\mathbf{u}}^{n,i}_{T'} = -R^n_{p,T}(\tilde{\mathbf{u}}^{n,i-1}) + \sum_{T'\in\mathcal{T}} \frac{\partial R^n_{p,T}}{\partial \tilde{\mathbf{u}}^n_{T'}}(\tilde{\mathbf{u}}^{n,i-1})\tilde{\mathbf{u}}^{n,i-1}_{T'}, \tag{3.3.17}$$

where $\tilde{\mathbf{u}}^{n,i}_T = \{P^{n,i}_T, S^{n,i}_{\mathrm{o},T}\}$ denotes the approximate solution in T. The system (3.3.17) is solved approximately by means of an iterative algebraic solver. Its residual vector components at a given step $j \ge 1$, for all $T \in \mathcal{T}$ and $p \in \{\mathrm{o},\mathrm{w}\}$, are defined by:

$$Res^{n,i,j}_{p,T} := \sum_{T'\in\mathcal{T}} \frac{\partial R^n_{p,T}}{\partial \tilde{\mathbf{u}}^n_{T'}}(\tilde{\mathbf{u}}^{n,i-1})\left(\tilde{\mathbf{u}}^{n,i,j}_{T'} - \tilde{\mathbf{u}}^{n,i-1}_{T'}\right) + R^n_{p,T}(\tilde{\mathbf{u}}^{n,i-1}), \tag{3.3.18}$$

where $\tilde{\mathbf{u}}^{n,i,j}_T = \{P^{n,i,j}_T, S^{n,i,j}_{\mathrm{o},T}\}$ denotes the approximate solution at time t^n, i-th Newton iteration, and j-th algebraic solver iteration, in the cell T..

The following Algorithm 3.1 presents the solving strategy for $1 \leq n \leq N_{\mathrm{F}}$. The time step parameter γ_t gives some simple automatic adaptivity in time. When the time step becomes too large, it may happen that the Newton algorithm fails to satisfy the stopping criterion after a preassigned maximum number of iterations. This is what we call "time step crash". In this case, we halve the time step and repeat the current computation.

In the numerical experiments, the matrix of system (3.3.17) is preconditioned by ILU0 (incomplete LU factorization, with zero level fill–In).

Remark 3.7 (Usual stopping criteria)**.** The stopping criteria are usually defined at the beginning of the algorithm, they compare the residuals with a well-chosen small quantity (e.g. $\|\mathbf{Res}_p^{n,i,j}\| < 10^{-12}$ for the algebraic solver).

Remark 3.8 (Initialization)**.** At each first iteration of the nonlinear and linear solvers, we traditionally prescribe $\mathbf{S}_{\mathrm{o}}^{n,0} := \mathbf{S}_{\mathrm{o}}^{n-1}$ and $\mathbf{P}^{n,0} := \mathbf{P}^{n-1}$, and $\mathbf{S}_{\mathrm{o}}^{n,i,0} := \mathbf{S}_{\mathrm{o}}^{n,i-1}$ and $\mathbf{P}^{n,i,0} := \mathbf{P}^{n,i-1}$.

Algorithm 3.1 Basic solution algorithm without adaptation

Create the mesh $\mathcal{T}$, an initial time step τ^0, set $t^0 = 0$ and $n = 0$;

Choose the initial saturation $\mathbf{S}_{\mathrm{o}}^0$ by (3.3.13);

Compute the artificial pressure $\mathbf{P}^0$ according to (3.3.14).

Time Loop

while $t^n \leq t_{\mathrm{F}}$ **do**

Set $n := n + 1$;

if Time step crash **then**

Set $n := n - 1$, $\tau^n := \frac{\tau^n}{2}$, $t^n := t^{n-1} + \tau^n$

else

Set $\tau^n := \gamma_t \tau^{n-1}$, $t^n := t^{n-1} + \tau^n$;

end if

Set $i := 0$; Choose initial saturations $\mathbf{S}_{\mathrm{o}}^{n,0}$ and pressures $\mathbf{P}^{n,0}$.

Newton's Loop

while Linearization criterion is not reached **do**

$i := i + 1$;

Evaluate the laws and their derivatives;

Evaluate the fluxes and update the upwind cells;

Assemble the Jacobian matrix, set up the **linear system** (3.3.17);

Set $j := 0$; Choose initial saturation $\mathbf{S}_{\mathrm{o}}^{n,i,0}$ and pressure $\mathbf{P}^{n,i,0}$;

Algebraic Loop

while Algebraic criterion is not reached. **do**

$j := j + 1$;

Precondition the matrix system;

Knowing $\mathbf{S}_{\mathrm{o}}^{n,i,j-1}$ and $\mathbf{P}^{n,i,j-1}$, compute $\mathbf{S}_{\mathrm{o}}^{n,i,j}$ and $\mathbf{P}^{n,i,j}$ by a step of the **algebraic solver**;

Compute the algebraic residual vectors $\mathbf{Res}_p^{n,i,j}$ defined by (3.3.18).

end while

Set $\mathbf{S}_{\mathrm{o}}^{n,i} := \mathbf{S}_{\mathrm{o}}^{n,i,j}$ and $\mathbf{P}^{n,i} := \mathbf{P}^{n,i,j}$.

end while

Set $\mathbf{S}_{\mathrm{o}}^{n} := \mathbf{S}_{\mathrm{o}}^{n,i}$ and $\mathbf{P}^{n} := \mathbf{P}^{n,i}$.

end while

3.4 Post-processings and elements of error analysis

Developing a mathematical theory for adaptive computation with finite volumes in two or three dimensions is not easy because the usual adaptation analysis relies on variational formulations satisfied by the exact solution, and the approximate solution is supposed to be reasonably smooth to insert it therein. But the solution $\tilde{\mathbf{u}}^n$ of the system (3.3.12), or $\tilde{\mathbf{u}}^{n,i,j}$ solving (3.3.18), is a set of step functions that are constant in each cell and time interval, and are not usefully substituted into the variational formulation. The purpose of this section is to introduce convenient reconstructions that transform the computed cell-wise values into suitable more regular functions that will be used in the definitions of the error indicators.

3.4.1 Pressure and capillary pressure post-treatments

3.4.1.1 Flux post-prossessing

Since the Finite Volume method yields approximations of the normal components of the velocity on element faces, it is convenient, for theoretical reasons, to transform them into $\mathbf{H}(\mathrm{div},\Omega)$ functions, more precisely, to interpolate them in $\mathbf{RTN}(\mathcal{T})$. Furthermore, it is convenient to split them as the sum of the contribution of the reference pressure and the contribution of the capillary pressure. Thus at each time t^n, linearization step i, and algebraic solver iteration j, for all $T \in \mathcal{T}$, all $p \in \mathcal{P}$ and all $\sigma \in \mathcal{F}_T$, we define the fluxes:

$$\overline{F}_{T,\sigma}^{n,i,j} = \sum_{T' \in \mathcal{T}_\sigma} \tau_T^\sigma P_{T'}^{n,i,j}, \tag{3.4.1}$$

and

$$\overline{\overline{F}}_{p,T,\sigma}^{n,i,j} = \sum_{T' \in \mathcal{T}_\sigma} \tau_T^\sigma P_{\mathrm{c}_p}(S_{p,T'}^{n,i,j}), \tag{3.4.2}$$

with $\mathcal{T}_\sigma$ the stencil for the face σ. But in fact, $\overline{\overline{F}}_{p,T,\sigma}^{n,i,j}$ is only non-zero for the wetting phase. This gives

$$F_{p,T,\sigma}^{n,i,j} = \overline{F}_{T,\sigma}^{n,i,j} + \overline{\overline{F}}_{p,T,\sigma}^{n,i,j}. \tag{3.4.3}$$

Then we define the approximate phase velocity $\mathbf{v}_{p,h}^{n,i,j} \in \mathbf{RTN}(\mathcal{T})$ corresponding to the total flux by

$$\langle \mathbf{v}_{p,h}^{n,i,j} \cdot \mathbf{n}_\sigma, 1 \rangle_\sigma = F_{p,T,\sigma}^{n,i,j}, \tag{3.4.4}$$

the approximate velocity corresponding to $\overline{\mathbf{v}}_h^{n,i,j} \in \mathbf{RTN}(\mathcal{T})$, corresponding to the reference pressure:

$$\langle \overline{\mathbf{v}}_h^{n,i,j} \cdot \mathbf{n}_\sigma, 1 \rangle_\sigma = \overline{F}_{T,\sigma}^{n,i,j}, \tag{3.4.5}$$

and the approximate velocity $\overline{\overline{\mathbf{v}}}_{\mathrm{w},h}^{n,i,j} \in \mathbf{RTN}(\mathcal{T})$ corresponding to the non-zero capillary pressure:

$$\langle \overline{\overline{\mathbf{v}}}_{\mathrm{w},h}^{n,i,j} \cdot \mathbf{n}_\sigma, 1 \rangle_\sigma = \overline{\overline{F}}_{\mathrm{w},T,\sigma}^{n,i,j}. \tag{3.4.6}$$

As the functions of $\mathbf{RTN}(\mathcal{T})$ have one degree of freedom per face, these formulas define uniquely the degrees of freedom, for instance (recall that $\mathbf{x}_\sigma$ is the barycenter of the face σ)

$$\overline{\mathbf{v}}_h^{n,i,j} \cdot \mathbf{n}_\sigma(\mathbf{x}_\sigma) = \frac{1}{|\sigma|} \overline{F}_{T,\sigma}^{n,i,j}. \tag{3.4.7}$$

Therefore, the functions $\overline{\mathbf{v}}_h^{n,i,j}$ and $\overline{\overline{\mathbf{v}}}_{\mathrm{w},h}^{n,i,j}$ are uniquely defined in $\mathbf{RTN}(\mathcal{T})$ by (3.4.4) and by (3.4.6). Moreover, $\mathbf{v}_{\mathrm{o},h}^{n,i,j} = \overline{\mathbf{v}}_h^{n,i,j}$ and $\mathbf{v}_{\mathrm{w},h}^{n,i,j} = \overline{\mathbf{v}}_h^{n,i,j} + \overline{\overline{\mathbf{v}}}_{\mathrm{w},h}^{n,i,j}$.

3.4.1.2 Pressure post-processing

In order to give a meaning to the gradient operator appearing in the phase velocity formula (3.1.2) in the discrete setting, we need to regularize the approximate reference pressures $\{P_T^{n,i,j}\}_{T\in\mathcal{T}}$ and capillary pressures $\{P_{\mathrm{c_w},T}^{n,i,j}\}_{T\in\mathcal{T}}$ (note that $P_{\mathrm{c_w},T}^{n,i,j} := P_{\mathrm{c_w}}(S_{\mathrm{w},T}^{n,i,j})$). As in the work done in [71, 53, 47, 73], these quantities are post-processed element wise yielding piecewise quadratic functions of $H^1(\mathcal{T})$: $\tilde{P}_h^{n,i,j}$ and $\tilde{P}_{\mathrm{c_w},h}^{n,i,j}$, at each time t^n, $1 \leq n \leq N_p$, Newton's iteration i, and linear solver iteration j.

Consider the reference pressure. We construct $\tilde{P}_h^{n,i,j}$, a piecewise quadratic function, such that

$$\tilde{P}_h^{n,i,j}(x,y) = a_T^{n,i,j}\, x^2 + b_T^{n,i,j}\, y^2 + c_T^{n,i,j}\, x + d_T^{n,i,j}\, y + e_T^{n,i,j},$$

with $a_T^{n,i,j}$, $b_T^{n,i,j}$, $c_T^{n,i,j}$, $d_T^{n,i,j}$, and $e_T^{n,i,j}$ real coefficients, defined by the following formula:

$$\begin{cases} \nabla \tilde{P}_h^{n,i,j} = -\mathbf{K}^{-1} \overline{\mathbf{v}}_h^{n,i,j}, \\ \dfrac{(\tilde{P}_h^{n,i,j}, 1)_T}{|\,T\,|} = P_T^{n,i,j}. \end{cases} \tag{3.4.8}$$

A schematic visualization is given in Figure 3.1.

The capillary pressure is post-processed in a similar fashion. We construct $\tilde{P}_{\mathrm{c_w},h}^{n,i,j}$, a piecewise quadratic function, such that

$$\tilde{P}_{\mathrm{c_w},h}^{n,i,j}(x,y) = a_T^{n,i,j}\, x^2 + b_T^{n,i,j}\, y^2 + c_T^{n,i,j}\, x + d_T^{n,i,j}\, y + e_T^{n,i,j},$$

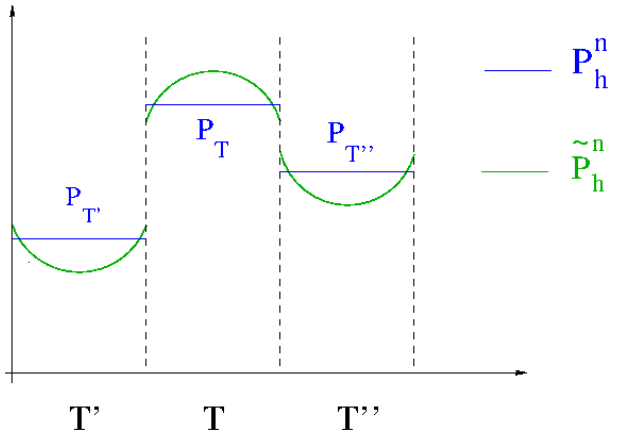

Figure 3.1: Schematic visualization of pressure reconstruction: calculated pressure (blue) and quadratic interpolated pressure (green).

with $a_T^{n,i,j}$, $b_T^{n,i,j}$, $c_T^{n,i,j}$, $d_T^{n,i,j}$ and $e_T^{n,i,j}$ real coefficients, chosen by the following formula:

$$\begin{cases} \nabla \tilde{P}_{\mathrm{c_w},h}^{n,i,j} = -\mathbf{K}^{-1} \overline{\overline{\mathbf{v}}}_{\mathrm{w},h}^{n,i,j}, \\ \dfrac{(\tilde{P}_{\mathrm{c_w},h}^{n,i,j}, 1)_T}{|\,T\,|} = P_{\mathrm{c_w}}(S_{\mathrm{w},T}^{n,i,j}). \end{cases} \tag{3.4.9}$$

In both cases, (3.4.8) and (3.4.9) are linear systems of five equations in five unknowns that are easily and explicitly solved. For our use below, we define the phase pressure post-processings as:

$$\tilde{P}_{\mathrm{o},h}^{n,i,j} = \tilde{P}_h^{n,i,j} \quad \text{and} \quad \tilde{P}_{\mathrm{w},h}^{n,i,j} = \tilde{P}_h^{n,i,j} + \tilde{P}_{\mathrm{c_w},h}^{n,i,j}.$$

Finally, at time t^0, the initial constant reference pressure P^0 needs no post-processing.

3.4.1.3 Post-processing in time

In order to give a good meaning to the time derivative, we also introduce a post-processing in time. This is straightforward. Let v_h^n belong to $L^2(\mathcal{T})$, $0 \leq n \leq N_F$. We associate with v_h^n the function $v_{h\tau}$ defined by

$$v_{h\tau}|_{I_n} = v_h^{n-1} + \frac{t - t^{n-1}}{\tau^n}(v_h^n - v_h^{n-1}), \quad t \in I_n, \tag{3.4.10}$$

on each interval I_n. The resulting function $v_{h\tau}$ is globally continuous in time, piecewise affine (hence it belongs to $H^1(0, t_{\mathrm{F}})$ with respect to time), and satisfies

$$v_{h\tau}(t^n) = v_h^n, \quad 0 \leq n \leq N_F.$$

This post-treatment is used for the two phase pressures, both are post-processed in time, giving function $\tilde{P}^{i,j}_{\mathrm{o},h\tau}$ and $\tilde{P}^{i,j}_{\mathrm{w},h\tau}$, defined by (3.4.10). Since the capillary pressure is a given function of saturations, we introduce the following notation:

$$\mathbf{v}_p(P_p) := \mathbf{v}_p(P, S_p),$$

given in each cell T by:

$$\mathbf{v}_p(\tilde{P}^{i,j}_{p,h\tau}) = -\mathbf{K}\nabla(\tilde{P}^{i,j}_{p,h\tau}), \qquad \text{for } p \in \{\mathrm{o}, \mathrm{w}\}.$$

The same piecewise affine-in-time post-processing functions $S^{i,j}_{p,h\tau}$ are also used for the phase saturations.

3.4.2 A total residual norm

For each n, $1 \le n \le N_{\mathrm{F}}$, let

$$X_n := L^2(I_n; H^1(\Omega)),$$

equipped with the norm:

$$\|\theta\|^2_{X_n} := \int_{I_n} \left(\sum_{T\in\mathcal{T}} \pi^2 h_T^{-2} \|\theta(t)\|^2_{L^2(T)} + \|\nabla\theta(t)\|^2_{L^2(T)} \right) \mathrm{d}t.$$

Similarly, let

$$X := L^2(0, t_{\mathrm{F}}; H^1(\Omega)),$$

equipped with the norm:

$$\|\theta\|^2_X := \int_0^{t_{\mathrm{F}}} \left(\sum_{T\in\mathcal{T}} \pi^2 h_T^{-2} \|\theta(t)\|^2_{L^2(T)} + \|\nabla\theta(t)\|^2_{L^2(T)} \right) \mathrm{d}t = \sum_{n=1}^{N_{\mathrm{F}}} \|\theta\|^2_{X_n}.$$

Let LHS denote the left-hand side of (3.2.26), summed over the phases, with test function θ in X, integrated over $(0, t_{\mathrm{F}})$:

$$LHS = \sum_{p\in\{\mathrm{o},\mathrm{w}\}} \int_0^{t_{\mathrm{F}}} \left(\phi \int_\Omega \partial_t(S_p)\, \theta \,\mathrm{d}\mathbf{x} + \int_\Omega \nu_p(P, S_p) \mathbf{K}\nabla[P + P_{\mathrm{c}_p}(S_p)] \cdot \nabla\theta \,\mathrm{d}\mathbf{x} \right) \mathrm{d}t. \quad (3.4.11)$$

Similarly, let LHS_h be the expression obtained by replacing in LHS S_p, P, and P_{c_p} by the post-processed functions $S_{p,h\tau}$, $\tilde{P}_{h\tau}$, and $\tilde{P}_{\mathrm{c}_p,h\tau}$, with integrals localized in each T to take into account the discontinuity of the functions:

$$LHS_h = \sum_{p\in\{\mathrm{o},\mathrm{w}\}} \int_0^{t_{\mathrm{F}}} \left(\phi \int_\Omega \partial_t(S_{p,h\tau})\, \theta \,\mathrm{d}\mathbf{x} + \sum_{T\in\mathcal{T}} \int_T \nu_p(\tilde{P}_{h\tau}, S_{p,h\tau}) \mathbf{K}\nabla[\tilde{P}_{h\tau} + \tilde{P}_{\mathrm{c}_p,h\tau}] \cdot \nabla\theta \,\mathrm{d}\mathbf{x} \right) \mathrm{d}t. \quad (3.4.12)$$

The difference $LHS - LHS_h$ is the residual. Specific terms must be added to it to take into account the discontinuity of the post-processed pressures. This suggests the following formulation for the error in the non-wetting saturation and reference pressure that expresses the residual and the nonconformity of the pressures:

$$\begin{aligned}|||(S_{\rm o} - S_{{\rm o},h\tau}, P - \tilde{P}_{h\tau})||| := &\left\{ \sum_{p\in\{{\rm o},{\rm w}\}} \left[\sup_{\theta\in X,\, \|\theta\|_X=1} \int_0^{t_{\rm F}} \left\{ \int_\Omega \Big(\phi(\partial_t(S_p) - \partial_t(S_{p,h\tau}))\theta \right.\right.\right. \\ &\left.\left.\left. - \Big(\nu_p(P,S_p)\overrightarrow{\mathbf{v}}_p(P,S_p) - \nu_p(\tilde{P}_{h\tau},S_{p,h\tau})\mathbf{v}_p(\tilde{P}_{p,h\tau})\Big)\cdot\nabla\theta \Big) {\rm d}\mathbf{x} \right\} {\rm d}t \right]^2 \right\}^{\frac{1}{2}} \\ &+ \left\{ \sum_{p\in\{{\rm o},{\rm w}\}} \inf_{\delta_p\in X} \int_0^{t_{\rm F}} \Big\| \nu_p(\tilde{P}_{h\tau},S_{p,h\tau})\mathbf{v}_p(\tilde{P}_{p,h\tau}) - \nu_p(\tilde{P}_{h\tau},S_{p,h\tau})\mathbf{v}_p(\delta_p) \Big\|^2_{L^2(\Omega)} {\rm d}t \right\}^{\frac{1}{2}}. \end{aligned} \tag{3.4.13}$$

Note that if $S_{p,h\tau}$ coincides with S_p, for $p \in \{{\rm o},{\rm w}\}$, and $\tilde{P}_{h\tau}$ with P, then

$$|||(S_p - S_{p,h\tau}, P - \tilde{P}_{h\tau})||| = 0.$$

We can also define for each n, $1 \le n \le N_{\rm F}$, the norm of the local residual in time

$$\begin{aligned}|||(S_{\rm o} - S_{{\rm o},h\tau}, P - \tilde{P}_{h\tau})|||_{I_n} := &\left\{ \sum_{p\in\{{\rm o},{\rm w}\}} \left[\sup_{\theta\in X_n,\, \|\theta\|_{X_n}=1} \int_{I_n} \left\{ \int_\Omega \Big(\phi\big(\partial_t(S_p) - \partial_t(S_{p,h\tau})\big)\theta \right.\right.\right. \\ &\left.\left.\left. - \Big(\nu_p(P,S_p)\overrightarrow{\mathbf{v}}_p(P,S_p) - \nu_p(\tilde{P}_{h\tau},S_{p,h\tau})\mathbf{v}_p(\tilde{P}_{p,h\tau})\Big)\cdot\nabla\theta \Big) {\rm d}\mathbf{x} \right\} {\rm d}t \right]^2 \right\}^{\frac{1}{2}} \\ &+ \left\{ \sum_{p\in\{{\rm o},{\rm w}\}} \inf_{\delta_p\in X_n} \int_{I_n} \Big\| \nu_p(\tilde{P}_{h\tau},S_{p,h\tau})\mathbf{v}_p(\tilde{P}_{p,h\tau}) - \nu_p(\tilde{P}_{h\tau},S_{p,h\tau})\mathbf{v}_p(\delta_p) \Big\|^2_{L^2(\Omega)} {\rm d}t \right\}^{\frac{1}{2}}. \end{aligned} \tag{3.4.14}$$

3.5 A posteriori error estimates

We have presented in Section 3.3.4 the way we choose to solve the problem. Our resolution has a linearization loop (using a Newton method), and each linearization step in the loop requires an algebraic solver. The non-adaptive method stipulates that the Newton and the iterative solver converge until a chosen stopping criterion is reached. We propose an a posteriori error analysis that provides an efficient choice for this stopping criterion. We will see that this a posteriori error analysis permits to identify different sources of the error. This requires

beforehand some velocity reconstructions in Section 3.5.1 and some pressure reconstruction in Section 3.5.2. We then show how we obtain the error estimators in Section 3.5.3. Then, using these estimators and evaluating the different error components, see Sections 3.5.4 and 3.5.5, we can use them to adapt the resolution method (the stopping criteria or the time and space discretizations). The resulting algorithm is given in Section 3.5.6.

3.5.1 Velocity reconstructions

To estimate the terms in the right-hand side of (3.4.13), it is convenient that the product of the mobility and the discrete velocity be also locally conservative. Indeed, this property is not necessarily satisfied at the discrete level. Following [46, 47, 73], we introduce for $T \in \mathcal{T}$, $1 \leq n \leq N_{\rm F}$, the i-th linearization step, j-th algebraic solver iteration, and $p \in \mathcal{P}$ the flux reconstructions $\boldsymbol{\chi}_{p,h}^{n,i,j} \in \mathbf{RTN}_0(\mathcal{T})$.

We suppose that they satisfy the following local conservation property:

$$(q_p^n - \partial_t(\phi S_{p,h\tau}^{i,j}) - \nabla\cdot\boldsymbol{\chi}_{p,h}^{n,i,j}, 1)_T = 0. \tag{3.5.1}$$

3.5.2 Pressure reconstruction

We have reconstructed in Section 3.4.1 piecewise quadratic phase pressure functions $\tilde{P}_{p,h,T}^{n,i,j}$, for $p \in \{\mathrm{o}, \mathrm{w}\}$, which are quadratic on each control volume T. Now we would like to evaluate the error due to the nonconformity of these discrete phase pressures. For this purpose, we construct by interpolation two phase pressure functions $\delta_{p,h}^{n,i,j}$ that are continuous in space. Since our triangulation is made of rectangles, the interpolated functions are constructed as continuous and bilinear (note that this would also hold on quadrilaterals). They are obtained by means of a regularization at the nodes of the form

$$\mathcal{I}(\tilde{P}_{p,h}^{n,i,j})(D) = \frac{1}{\text{Card } \mathcal{T}_{\rm D}} \sum_{T\in\mathcal{T}_D} \tilde{P}_{p,h}^{n,i,j}|_T(D), \tag{3.5.2}$$

where D denotes nodes of the triangulation and $\mathcal{T}_D$ denotes the stencil of D, i.e., the set of elements of $\mathcal{T}$ that share D; see Figure 3.2. The nodal interpolation (3.5.2) is a classical averaging formula. Once the nodal degrees of freedom are defined by (3.5.2), the interpolant $\mathcal{I}(\tilde{P}_h^{n,i,j})$ is the unique bilinear function in each cell T that takes the values $\mathcal{I}(\tilde{P}_h^{n,i,j})(D)$ at the four verticies D of T. It has the form

$$\forall (x,y) \in T,\ \mathcal{I}(x,y) = a_T^{n,i,j}\, xy + b_T^{n,i,j}\, x + c_T^{n,i,j}\, y + d_T^{n,i,j}, \tag{3.5.3}$$

with $a_T^{n,i,j}$, $b_T^{n,i,j}$, $c_T^{n,i,j}$, and $d_T^{n,i,j}$ real coefficients on each cell T.

We set

$$\delta_{p,h}^{n,i,j} := \mathcal{I}(\tilde{P}_{p,h}^{n,i,j}), \tag{3.5.4}$$

and denote by $\delta_{p,h\tau}^{i,j}$ its piecewise affine-in-time interpolant defined in (3.4.10). Note that by construction $\delta_{p,h\tau}^{i,j}$ belongs to X, more precisely, it belongs to $W^{1,\infty}(0,t_{\mathrm{F}};H^1(\Omega))$.

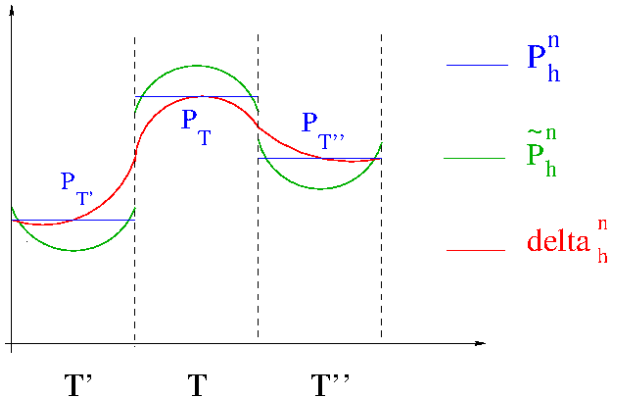

Figure 3.2: Schematic visualization of pressure reconstruction: calculated pressure $P_h^{n,i,j}$ (blue), quadratic interpolated pressure $\tilde{P}_h^{n,i,j}$ (green), and bilinear phase pressure $\delta_{p,h}^{n,i,j}$ (red).

3.5.3 A basic a posteriori error estimate

Consider $1 \le n \le N_{\mathrm{F}}$, the i-th linearization step, the j-th linear solver step, $T \in \mathcal{T}$, and $p \in \mathcal{P}$. We define the *residual estimators* $\eta_{\mathrm{R},T,p}^{n,i,j}$ as:

$$\eta_{\mathrm{R},T,p}^{n,i,j} := \frac{h_T}{\pi} \| q_p^n - \partial_t(\phi S_{p,h\tau}^{i,j}) - \nabla \cdot \boldsymbol{\chi}_{p,h}^{n,i,j} \|_T, \tag{3.5.5}$$

where the constant π comes from the Poincaré inequality. Let us recall this inequality for convex element T, see the work of Payne and Weinberger in [60]:

$$\|\varphi - \varphi_T\|_T \le \frac{h_T}{\pi} \|\nabla \varphi\|_T, \ \forall \varphi \in H^1(T), \tag{3.5.6}$$

where φ_T is the mean value of the function φ on the element T.

Let us define the *flux estimators* $\eta_{\mathrm{F},T,p}^{n,i,j}$, evaluating the error due to the possible $\mathbf{H}(\mathrm{div},\Omega)$-nonconformity of the fluxes:

$$\forall t \in I_n, \ \eta_{\mathrm{F},T,p}^{n,i,j}(t) := \left\| \boldsymbol{\chi}_{p,h}^{n,i,j} - \nu_p(\tilde{P}_{h\tau}^{i,j}, S_{p,h\tau}^{i,j}) \mathbf{v}_p(\tilde{P}_{p,h\tau}^{i,j})(t) \right\|_T. \tag{3.5.7}$$

Finally, for the phase p, the *nonconformity estimators* $\eta_{\mathrm{NC},T,p}^{n,i,j}$, relative to the error caused by the $H^1(\Omega)$-nonconformity of the phase pressures, are defined as:

$$\forall t \in I_n,\ \eta_{\mathrm{NC},T,p}^{n,i,j}(t) := \left\| \nu_p(\tilde{P}_{h\tau}^{i,j}, S_{p,h\tau}^{i,j}) \mathbf{v}_p(\tilde{P}_{p,h\tau}^{i,j})(t) - \nu_p(\tilde{P}_{h\tau}^{i,j}, S_{p,h\tau}^{i,j}) \mathbf{v}_p(\delta_{p,h\tau})(t) \right\|_T. \tag{3.5.8}$$

With the above material,we are ready to prove.

Theorem 3.9 (Residual error estimate at time step n, linearization step i, and algebraic solver step j). *Let $\mathcal{P} = \{\mathrm{o}, \mathrm{w}\}$, $1 \le n \le N_{\mathrm{F}}$, and I_n the time interval. Let P and S_{o} be the exact reference pressure and non-wetting saturation given by* (3.2.26) *and set $S_{\mathrm{w}} = 1 - S_{\mathrm{o}}$. Consider the i-th linearization step and the j-th algebraic solver step. Then*

$$\begin{aligned} |||(S_{\mathrm{o}} - S_{\mathrm{o},h\tau}^{n,i,j}, P - \tilde{P}_{h\tau}^{n,i,j})|||_{I_n} \le & \left\{ \sum_{p \in \{\mathrm{o},\mathrm{w}\}} \int_{I_n} \sum_{T \in \mathcal{T}} \left(\eta_{\mathrm{R},T,p}^{n,i,j} + \eta_{\mathrm{F},T,p}^{n,i,j}(t) \right)^2 \mathrm{d}t \right\}^{\frac{1}{2}} \\ & + \left\{ \sum_{p \in \{\mathrm{o},\mathrm{w}\}} \int_{I_n} \sum_{T \in \mathcal{T}} \left(\eta_{\mathrm{NC},T,p}^{n,i,j}(t) \right)^2 \mathrm{d}t \right\}^{\frac{1}{2}}. \end{aligned} \tag{3.5.9}$$

Proof. The proof is straightforward considering the definition of the residual norm (3.4.14). Let $1 \le n \le N_{\mathrm{F}}$, $1 \le i$, and $1 \le j$ be given. The second term in (3.5.9) clearly stems from the second term in the right hand-side of (3.4.14). We thus only have to prove that the first term is an upper bound on the first term in the right hand-side of (3.4.14).

Let $\theta \in X_n$, $\|\theta\|_{X_n} = 1$, and $p \in \mathcal{P}$. Set $\mathbf{w}_p := \nu_p(P, S_{\mathrm{o}}) \mathbf{v}_p(P, S_{\mathrm{o}})$ and $\mathbf{w}_{p,h\tau}^{i,j} := \nu(\tilde{P}_{h\tau}^{i,j}, S_{\mathrm{o},h\tau}^{i,j}) \mathbf{v}_p(\tilde{P}_{p,h\tau}^{i,j})$. Then using the characterization of the weak solution (3.2.26),

$$\begin{aligned} & \int_{I_n} \left\{ \int_\Omega \Big(\phi(\partial_t S_p - \partial_t S_{p,h\tau}^{i,j})\theta - \big(\mathbf{w}_p - \mathbf{w}_{p,h\tau}^{i,j}\big) \cdot \nabla\theta \Big) \mathrm{d}\mathbf{x} \right\} \mathrm{d}t \\ = & \int_{I_n} \left\{ \int_\Omega \Big((q_p - \partial_t(\phi S_{p,h\tau}^{i,j}))\theta + \mathbf{w}_{p,h\tau}^{i,j} \cdot \nabla\theta \Big) \mathrm{d}\mathbf{x} \right\} \mathrm{d}t. \end{aligned}$$

Adding and subtracting

$$\int_\Omega \boldsymbol{\chi}_{p,h\tau}^{i,j} \cdot \nabla\theta \mathrm{d}\mathbf{x},$$

where $\boldsymbol{\chi}_{p,h\tau}^{i,j}$ is piecewise constant in time, given on each time interval I_n by $\boldsymbol{\chi}_{p,h}^{n,i,j}$, using the Green theorem, the local conservation property (3.5.1), the Poincaré inequality, and the

Cauchy–Schwarz inequality

$$\begin{aligned}
&\int_\Omega q_p\theta \mathrm{d}\mathbf{x} - \int_\Omega \partial_t(\phi S_{p,h\tau}^{i,j})\theta \mathrm{d}\mathbf{x} + \int_\Omega \mathbf{w}_{p,h\tau}^{i,j}\cdot\nabla\theta \mathrm{d}\mathbf{x}\\
&= \int_\Omega \Big((q_p - \partial_t(\phi S_{p,h\tau}^{i,j}) - \nabla\cdot\boldsymbol{\chi}_{p,h}^{n,i,j})\theta\Big)\mathrm{d}\mathbf{x} + \int_\Omega \Big(\mathbf{w}_{p,h\tau}^{i,j} - \boldsymbol{\chi}_{p,h}^{n,i,j}\cdot\nabla\theta\Big)\mathrm{d}\mathbf{x}\\
&= \int_\Omega \Big((q_p - \partial_t(\phi S_{p,h\tau}^{i,j}) - \nabla\cdot\boldsymbol{\chi}_{p,h}^{n,i,j})(\theta - \Pi_0\theta)\Big)\mathrm{d}\mathbf{x} + \int_\Omega \Big((\mathbf{w}_{p,h\tau}^{i,j} - \boldsymbol{\chi}_{p,h}^{n,i,j})\cdot\nabla\theta\Big)\mathrm{d}\mathbf{x}\\
&\le \sum_{T\in\mathcal{T}} (\eta_{\mathrm{R},T,p}^{n,i,j} + \eta_{\mathrm{F},T,p}^{n,i,j}(t))\|\nabla\theta\|_T,
\end{aligned}$$

where Π_0 denotes the L^2-orthogonal projection onto piecewise constants on $\mathcal{T}$.

Finally, the assertion follows by the Cauchy–Schwarz inequality and the fact that the suppremum in (3.4.13) is with $\|\theta\|_{X_n} = 1$. □

Remark 3.10. The terms in (3.5.8) represent the nonconformity of the two phase pressures. If we have continuous approximate phase pressures, these terms disappear. In a cell-centered FV method, these pressures are not continuous, so we need to evaluate and to manage these nonconformities.

3.5.4 Identification of different components of the error

We have, in the previous section, introduced the a posteriori error estimators. Our aim is now to distinguish the origins of the errors:

- Space errors;
- Temporal errors;
- Linearization errors;
- Algebraic error.

In Section 3.3.3, we define the nonlinear system (3.3.15) and we solve it in Section 3.3.4 using an iterative solver for the Newton algorithm. Let $1 \le n \le N_\mathrm{F}$, $T \in \mathcal{T}$, and $p \in \mathcal{P}$. Then, we fix a linearization step i and an iterative algebraic solver step j, the flux reconstructions $\boldsymbol{\chi}_{p,h}^{n,i,j} \in \mathbf{RTN}(\mathcal{T})$, already introduced in Section 3.5.1, satisfy the conservativity property (3.5.1)

$$(q_p^n - \partial_t(\phi S_{p,h\tau}^{i,j}) - \nabla\cdot\boldsymbol{\chi}_{p,h}^{n,i,j}, 1)_T = 0, \quad \forall T \in \mathcal{T}. \tag{3.5.10}$$

Let us now define a flux associated to the discretization $\overline{\boldsymbol{\chi}}_{p,h}^{n,i,j} \in \mathbf{RTN}(\mathcal{T})$, a linearized solver flux $\mathbf{l}_{p,h}^{n,i,j} \in \mathbf{RTN}(\mathcal{T})$, and an algebraic solver flux $\mathbf{r}_{p,h}^{n,i,j} \in \mathbf{RTN}(\mathcal{T})$ such that

$$\boldsymbol{\chi}_{p,h}^{n,i,j} = \overline{\boldsymbol{\chi}}_{p,h}^{n,i,j} + \mathbf{l}_{p,h}^{n,i,j} + \mathbf{r}_{p,h}^{n,i,j}.$$

We define them as follows. For $T_1 \in \mathcal{T}$ and $T_2 \in \mathcal{T}_T$, with a face $\sigma = T_1 \cap T_2 \subset \partial T_1$, the degree of freedom of $\overline{\boldsymbol{\chi}}_{p,h}^{n,i,j}$ is defined by:

$$\langle \overline{\boldsymbol{\chi}}_{p,h}^{n,i,j} \cdot \mathbf{n}_\sigma, 1 \rangle_\sigma := \nu_p(P_{T_p^\uparrow}^{n-1}, S_{\mathrm{o},T_\mathrm{o}^\uparrow}^{n-1}) F_{p,T_1,\sigma}^{n,i,j}. \tag{3.5.11}$$

Introduce the following notation:

$$B_{p,T,\sigma}^{n,i,j} := \nu_p(P_{T_p^\uparrow}^{n,i-1}, S_{p,T_p^\uparrow}^{n,i-1}) F_{p,T,\sigma}^{n,i,j}. \tag{3.5.12}$$

The linear system (3.3.18) is then equivalent to the following sum of diagonal terms and face fluxes:

$$\begin{aligned} \phi \frac{|T|}{\tau^n} \left(S_{p,T}^{n,i,j} - S_{p,T}^{n,i-1} \right) + \sum_{\sigma \in \mathcal{F}_T} \sum_{T' \in \mathcal{S}_\sigma} \frac{\partial B_{p,T,\sigma}^{n,i,j}}{\partial \tilde{\mathbf{u}}_{T'}} \left(\tilde{\mathbf{u}}^{n,i,j} - \tilde{\mathbf{u}}^{n,i-1} \right) \\ + R_{p,T}^n(\tilde{\mathbf{u}}^{n,i-1}) = Res_{p,T}^{n,i,j}, \end{aligned} \tag{3.5.13}$$

where $Res_{p,T}^{n,i,j}$ is the algebraic residual.
We then set

$$\langle (\overline{\boldsymbol{\chi}}_{p,h}^{n,i,j} + \mathbf{l}_{p,h}^{n,i,j}) \cdot \mathbf{n}_\sigma, 1 \rangle_\sigma := \sum_{T' \in \mathcal{S}_\sigma} \left(\frac{\partial B_{p,T,\sigma}^{n,i,j}}{\partial \tilde{\mathbf{u}}_{T'}} (\tilde{\mathbf{u}}^{n,i,j} - \tilde{\mathbf{u}}^{n,i-1}) \right) + \nu_p(P_{T_p^\uparrow}^{n,i-1}, S_{p,T_p^\uparrow}^{n,i-1}) F_{p,T',\sigma}^{n,i-1} \tag{3.5.14}$$

and

$$(\nabla \cdot \mathbf{r}_{p,h}^{n,i,j}, 1)_T = -Res_{p,T}^{n,i,j}. \tag{3.5.15}$$

Note that $\overline{\boldsymbol{\chi}}_{p,h}^{n,i,j}$ and $\mathbf{l}_{p,h}^{n,i,j}$ are fully specified; $\mathbf{r}_{p,h}^{n,i,j}$, satisfying (3.5.15) can be constructed as in Section 7.3 of [53]. This gives:

$$(q_p^n - \partial_t(\phi S_{p,h\tau}^{i,j}) - \nabla \cdot (\overline{\boldsymbol{\chi}}_{p,h}^{n,i,j} + \mathbf{l}_{p,h}^{n,i,j}), 1)_T = (\nabla \cdot \mathbf{r}_{p,h}^{n,i,j}, 1)_T, \qquad \text{for } p \in \mathcal{P} \tag{3.5.16}$$

which is the requested local conservation property (3.5.1).

We are now ready to distinguish the different parts of the error using (3.5.5)–(3.5.8). Indeed we want to have:

$$\eta_{\mathrm{R},T,p}^{n,i,j} + \eta_{\mathrm{F},T,p}^{n,i,j}(t) + \eta_{\mathrm{NC},T,p}^{n,i,j}(t) \leq \eta_{\mathrm{sp},T,p}^{n,i,j}(t) + \eta_{\mathrm{tm},T,p}^{n,i,j}(t) + \eta_{\mathrm{lin},T,p}^{n,i,j} + \eta_{\mathrm{alg},T,p}^{n,i,j},$$

where the error estimators are separated in two parts and are defined as follows:

1. The *substantial errors*, which have an important impact on the final precision of the discrete solution. We have:

   (a) The *spatial estimator* $\eta^{n,i,j}_{\mathrm{sp},T,p}$:

   $$\eta^{n,i,j}_{\mathrm{sp},T,p}(t):=\left\|\nu_p(\tilde{P}^{n,i,j}_h,S^{n,i,j}_{p,h})\mathbf{v}_p(\tilde{P}^{n,i,j}_{p,h})-\overline{\boldsymbol{\chi}}^{n,i,j}_{p,h}\right\|_T+\eta^{n,i,j}_{\mathrm{R},T,p}+\eta^{n,i,j}_{\mathrm{NC},T,p}(t), \quad (3.5.17)$$

   which quantifies the error due to the space discretization; that is to say the coarser is the mesh, the larger is the error;

   (b) The *temporal estimator* $\eta^{n,i,j}_{\mathrm{tm},T,p}$:

   $$\eta^{n,i,j}_{\mathrm{tm},T,p}(t):=\parallel \nu_p(\tilde{P}^{i,j}_{h\tau},S^{i,j}_{p,h\tau})\mathbf{v}_p(\tilde{P}^{i,j}_{p,h\tau})(t)-\nu_p(\tilde{P}^{n,i,j}_h,S^{n,i,j}_{p,h})\mathbf{v}_p(\tilde{P}^{n,i,j}_{p,h})\parallel_T, \quad (3.5.18)$$

   which is linked to the time discretization; if the error is too large we have to reduce the time step.

2. The *subsidiary errors*, which can be made arbitrarily small when the iterative linearization and algebraic solvers converge. The associated estimators are:

   (a) The *linearization estimator* $\eta^{n,i,j}_{\mathrm{lin},T,p}$:

   $$\eta^{n,i,j}_{\mathrm{lin},T,p}:=\left\|\mathbf{l}^{n,i,j}_{p,h}\right\|_T, \quad (3.5.19)$$

   whose values depend on the number of linearization steps; The more we make iterations, the smaller is the error;

   (b) The *algebraic estimator* $\eta^{n,i,j}_{\mathrm{alg},T,p}$:

   $$\eta^{n,i,j}_{\mathrm{alg},T,p}:=\|\mathbf{r}^{n,i,j}_{p,h}\|_T, \quad (3.5.20)$$

   associated to the error caused by the algebraic solver, similarly to the linearization error estimator; The more we make iterations, the smaller is the error.

With these estimators, we can now evaluate the error distributions in time and in space, and the error dependent of the linearization and algebraic resolution.

3.5.5 A posteriori error estimate distinguishing the error components

In order to use the a posteriori error estimators and to design an adaptive algorithm, let us define the global version of the estimators (3.5.17)–(3.5.20) as follows:

- The global spatial error estimator:

$$\eta_{\mathrm{sp}}^{n,i,j} := \left\{ 2 \sum_{p\in\{\mathrm{o,w}\}} \int_{I_n} \sum_{T\in\mathcal{T}^n} (\eta_{\mathrm{sp},T,p}^{n,i,j}(t))^2 \,\mathrm{d}t \right\}^{\frac{1}{2}}; \tag{3.5.21}$$

- The global temporal error estimator:

$$\eta_{\mathrm{tm}}^{n,i,j} := \left\{ \sum_{p\in\{\mathrm{o,w}\}} \int_{I_n} \sum_{T\in\mathcal{T}^n} (\eta_{\mathrm{tm},T,p}^{n,i,j}(t))^2 \,\mathrm{d}t \right\}^{\frac{1}{2}}; \tag{3.5.22}$$

- The global linearization error estimator:

$$\eta_{\mathrm{lin}}^{n,i,j} := \left\{ \sum_{p\in\{\mathrm{o,w}\}} \int_{I_n} \sum_{T\in\mathcal{T}^n} (\eta_{\mathrm{lin},T,p}^{n,i,j}(t))^2 \,\mathrm{d}t \right\}^{\frac{1}{2}}; \tag{3.5.23}$$

- The global algebraic solver error estimator:

$$\eta_{\mathrm{alg}}^{n,i,j} := \left\{ \sum_{p\in\{\mathrm{o,w}\}} \int_{I_n} \sum_{T\in\mathcal{T}^n} (\eta_{\mathrm{alg},T,p}^{n,i,j})^2 \,\mathrm{d}t \right\}^{\frac{1}{2}}. \tag{3.5.24}$$

Finally we can introduce the following corollary using Theorem 3.9, the triangle inequality, and the Cauchy–Schwarz inequality:

Corollary 3.11 (An a posteriori error estimate distinguishing the space, temporal, linearization, and algebraic errors)**.** *Let the assumptions of Theorem 3.9 be verified. Then*

$$|||(S_{\mathrm{o}} - S_{\mathrm{o},h\tau}^{i,j}, P - \tilde{P}_{h\tau}^{i,j})|||_{I_n} \leq \eta_{\mathrm{sp}}^{n,i,j} + \eta_{\mathrm{tm}}^{n,i,j} + \eta_{\mathrm{lin}}^{n,i,j} + \eta_{\mathrm{alg}}^{n,i,j}.$$

To have an idea of each part of error, we should evaluate these estimators at each time step. Then, they permit us to adapt

1. Our time or space step;

2. The stopping criteria for the linear solver and for the algebraic solver.

3.5.6 Adaptive algorithm

To solve the nonlinear system (3.3.16), let us introduce the following adaptive algorithm using the tools we have detailed previously, issued from the non-adaptive Algorithm 3.1.

Remark 3.12 (Space and time adaptation)**.** Let us precise that space or time adaptivity are not described in the following algorithm, but the idea is to refine (respectively coarsen) cells when $\eta_{\mathrm{sp},T,p}^{n,i,j}$ is large (respectively small), or to increase (respectively decrease) time step, using γ_t, when $\eta_{\mathrm{tm}}^{n,i,j}$ is large (respectively small).

Algorithm 3.2 Adaptive algorithm

Create the mesh $\mathcal{T}$, an initial time step τ^0 , set $t^0 = 0$ and $n = 1$.

Choose initial saturation $\mathbf{S}_{\mathrm{o}}^0$ and artificial pressure $\mathbf{P}^0$ according to (3.3.13) and (3.3.14).

while $\sum_{i=1}^{n} \tau_i \leq t_{\mathrm{F}}$ **do**

Set $t^n := t^{n-1} + \tau^n$, set $\tau^n := \tau^{n-1}$; Set $i = 0$.

Choose initial saturations $\mathbf{S}_{\mathrm{o}}^{n,0}$ and pressures $\mathbf{P}^{n,0}$ according to (3.1.3).

(Typically, $\mathbf{S}_{\mathrm{o}}^{n,0} = \mathbf{S}_{\mathrm{o}}^{n-1}$ and $\mathbf{P}^{n,0} = P^{n-1}$).

while

$$\eta_{\mathrm{lin}}^{n,i,j} > \gamma_{\mathrm{lin}}(\eta_{\mathrm{sp}}^{n,i,j} + \eta_{\mathrm{tm}}^{n,i,j}) \tag{3.5.25}$$

do

$i := i + 1$; Set $j = 0$.

Set up the **linear system** (3.3.17).

Choose initial saturation $\mathbf{S}_{\mathrm{o}}^{n,i,0} = \mathbf{S}_{\mathrm{o}}^{n,i-1}$ and pressure $\mathbf{P}^{n,i,0} = \mathbf{P}^{n,i-1}$.

while

$$\eta_{\mathrm{alg}}^{n,i,j} > \gamma_{\mathrm{alg}}(\eta_{\mathrm{sp}}^{n,i,j} + \eta_{\mathrm{tm}}^{n,i,j} + \eta_{\mathrm{lin}}^{n,i,j}) \tag{3.5.26}$$

do

$j := j + 1$;

Perform a step of a chosen **iterative algebraic method** for the solution of (3.3.17), starting from $\mathbf{S}_{\mathrm{o}}^{n,i,j-1}$ and $\mathbf{P}^{n,i,j-1}$; This gives $\mathbf{S}_{\mathrm{o}}^{n,i,j}$ and $\mathbf{P}^{n,i,j}$.

\\ **A posteriori error estimators part**

See Algorithm 3.3

end while

Set $\mathbf{S}_{\mathrm{o}}^{n,i} := \mathbf{S}_{\mathrm{o}}^{n,i,j}$ and $\mathbf{P}^{n,i} := \mathbf{P}^{n,i,j}$.

end while

Set $\mathbf{S}_{\mathrm{o}}^{n} := \mathbf{S}_{\mathrm{o}}^{n,i}$ and $\mathbf{P}^{n} := \mathbf{P}^{n,i}$.

end while

Algorithm 3.3 A posteriori part

\\ **A posteriori error estimators part**
Post-process locally the pressures $\mathbf{P}^{n,i,j}$ and the capillary pressures $\{P_{\mathrm{c}_p}(S_T^{n,i,j})\}_{T\in\mathcal{T}}$.
Construct the fluxes $\overline{\boldsymbol{\chi}}_{p,h}^{n,i,j} \in \mathbf{RTN}(\mathcal{T})$, $p \in \mathcal{P}$, according to 3.5.11;
Construct the fluxes $\mathbf{l}_{p,h}^{n,i,j} \in \mathbf{RTN}(\mathcal{T})$, $p \in \mathcal{P}$, according to 3.5.14;
From the algebraic residual vectors $\mathbf{Res}_p^{n,i,j}$ construct the fluxes $\mathbf{r}_{p,h}^{n,i,j} \in \mathbf{RTN}(\mathcal{T})$, $p \in \mathcal{P}$, according to (3.5.15);
Evaluate all the indicators (3.5.17)–(3.5.20);
Define their global versions by their Hilbertian sums described in (3.5.21)–(3.5.24);
\\ **End a posteriori error estimators part**

Remark 3.13 (The criteria and the constants γ)**.** Let us give some remarks on the constants present in the stopping criteria formulas:

- In the stopping criterion for the algebraic solver, $0 < \gamma_{\mathrm{alg}} \leq 1$ is a user-given weight, typically of order 0.001;
- The same observation is made for $0 < \gamma_{\mathrm{lin}} \leq 1$.

Let us now give the meaning of the criteria:

- Criterion (3.5.25) expresses that there is no need to continue with the linearization iterations if the overall error is dominated by the other components.
- Criterion (3.5.26) expresses that there is no need to continue with the algebraic solver iterations if the overall error is dominated by the other components.

Remark 3.14 (A posteriori error analysis implementation)**.** If we observe the structure of the adaptive Algorithm 3.2, we see that an interest of the a posteriori error analysis is that it can be implemented in an "independent block". We will see this implementation point of view in Chapter 4.

3.6 Numerical results

In this section, we test the computational performances of the adaptive algorithm introduced in this chapter. We take the same five spot case as in Section 2.4 to study the efficiency of the method, in a square domain 200km $\times$ 200km. We compare the numerical results produced by two algorithms:

- A non-adaptive algorithm described in Algorithm 3.1;
- The adaptive Algorithm 3.2.

The mesh is uniform 20×20, independent of time. The initial time step is equal to $\tau = 2.16 \cdot 10^4$s and the final time is $4.32 \cdot 10^6$s.

The Section 3.6.1 presents the same two-phase flow five-spot case as in Section 2.4.1: we compare the results and we present the efficiency of the adaptive algorithm in terms of solver iterations. In Section 3.6.2, as in Section 2.4.2, we consider the same settingsbut the media is heterogeneous. The heterogeneity is given by the permeability tensor which is randomly distributed in the domain.

In order to compute the adaptive Algorithm 3.2 we need to define the parameters in the stopping criteria (3.5.25) and (3.5.26); These parameters are fixed at the beginning of the simulation. On the one hand, the parameter γ_{alg} is equal to 10^{-3}. On the other hand, the parameter γ_{lin} is equal to 10^{-3}.

Both cases give results which reveal that if we perform the algorithm using the a posteriori analysis, the overall error is not affected significantly. This means that even if we save on linearization or algebraic iterations, we can be confident in the accuracy of the results we calculate.

3.6.1 Homogeneous porous media

The settings used for this test case are exactly the same as in Section 2.4.1. We also use the same mesh settings, same time discretization setting, and same physical settings. Let us consider a homogeneous isotropic medium, where the permeability constant is equal to $\kappa_T = 1.0 \cdot 10^{-13}$, for all $T \in \mathcal{T}$.

3.6.1.1 Identification of the different sources of the error

First, we present the quantities we evaluate and the values of the unknowns (pressure, phase saturations), the values of the a posteriori error estimators. Figures 3.3 and 3.4 represent the water saturation, and respectively the global pressure, computed at different simulation times with the non-adaptive method and with the adaptive one. There are no observable differences between both results.In particular, Figure 3.5 shows two curves of the pressure on a diagonal line of the domain obtained with the non-adaptive and the adaptive algorithm. These curves are also similar and such results permit to be confident in the use of a posteriori error estimators to adapt the stopping criteria.

In Figures 3.6 and 3.7, we present some a posteriori error estimators and their evolutions through the simulation of the adaptive method. We observe easily that they follow the water-oil front. We can also use those error estimates for spatial or temporal adaptation, as done in the work of [56]. But we recall this is not the purpose of the present work.

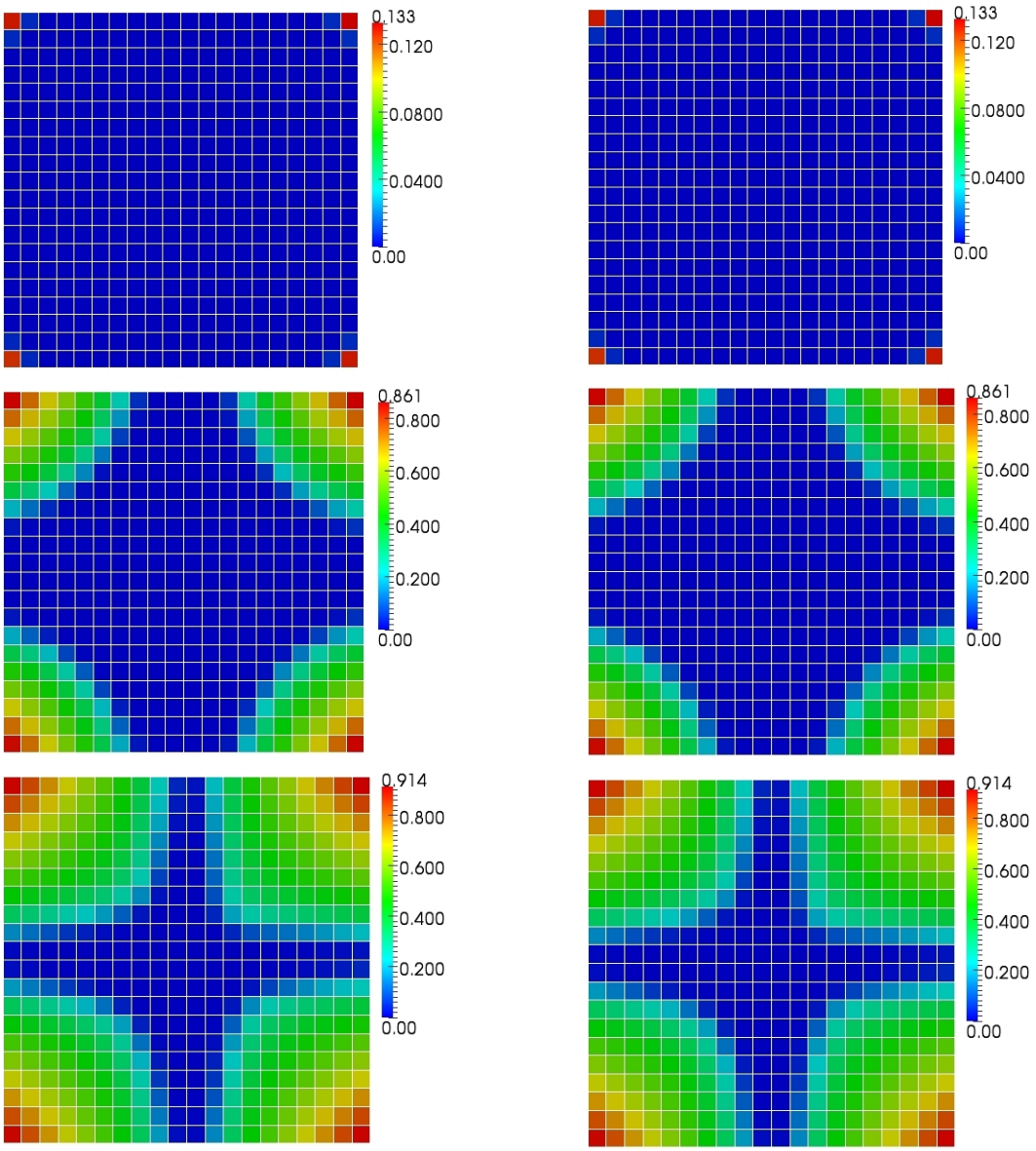

Figure 3.3: Wetting phase saturation obtained by the non-adaptive algorithm (left) and adaptive algorithm (right) at time $2.16 \cdot 10^4$s (top), at time $2.16 \cdot 10^6$s (middle) and at time $4.32 \cdot 10^6$s (bottom).

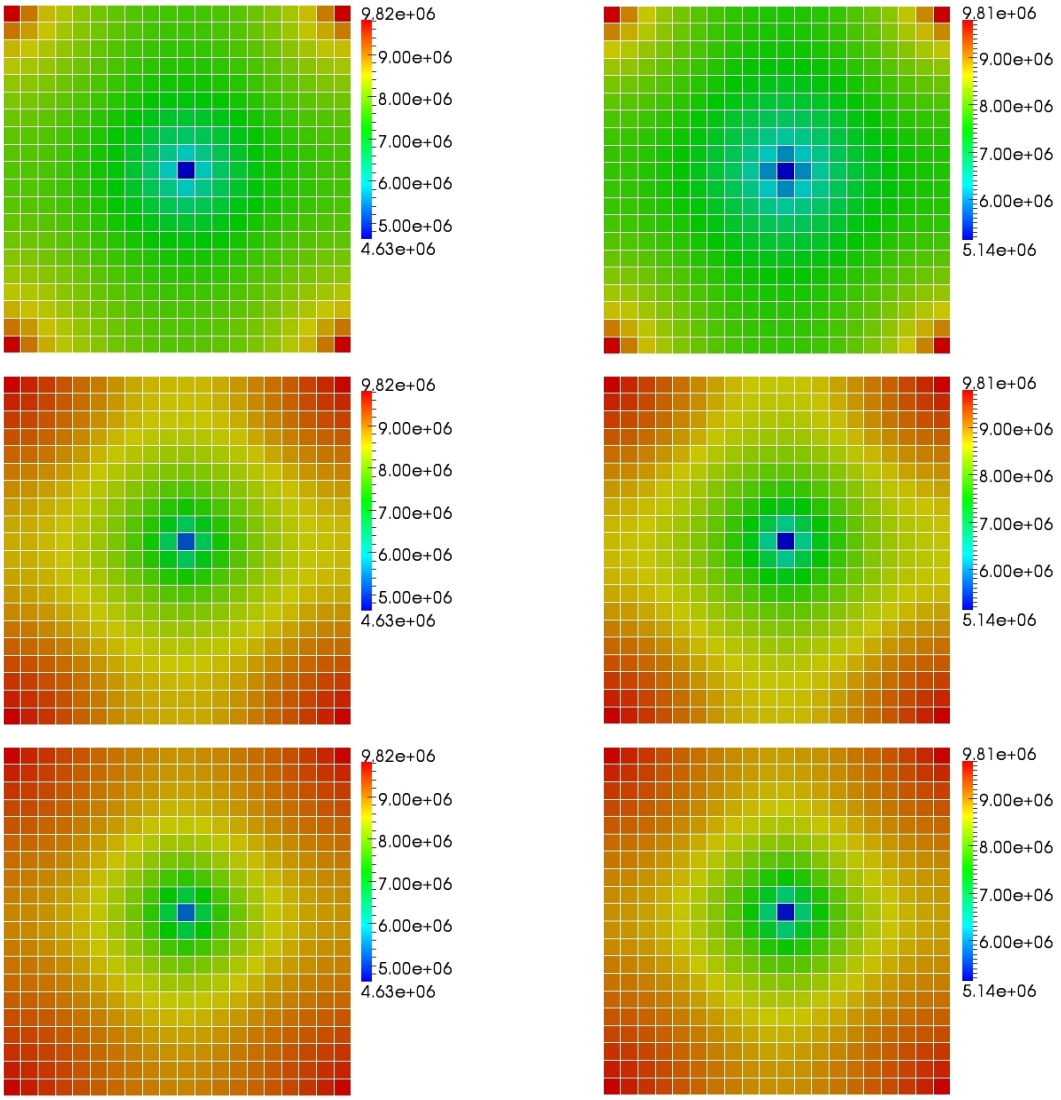

Figure 3.4: Reference pressure obtained by the non-adaptive algorithm (left) and the adaptive algorithm (right) at time $2.16 \cdot 10^4$s (top), at time $2.16 \cdot 10^6$s (middle) and at time $4.32 \cdot 10^6$s (bottom).

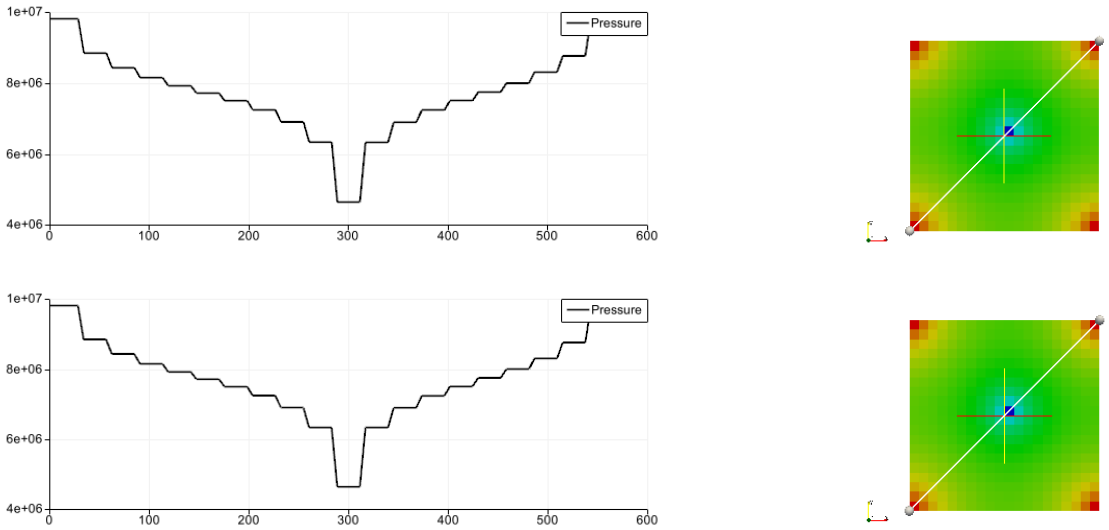

Figure 3.5: Reference pressure curves (left) obtained by the non-adaptive algorithm (top) and the adaptive algorithm (bottom) at time $4.32 \cdot 10^4$s (left), on a line in the domain (right).

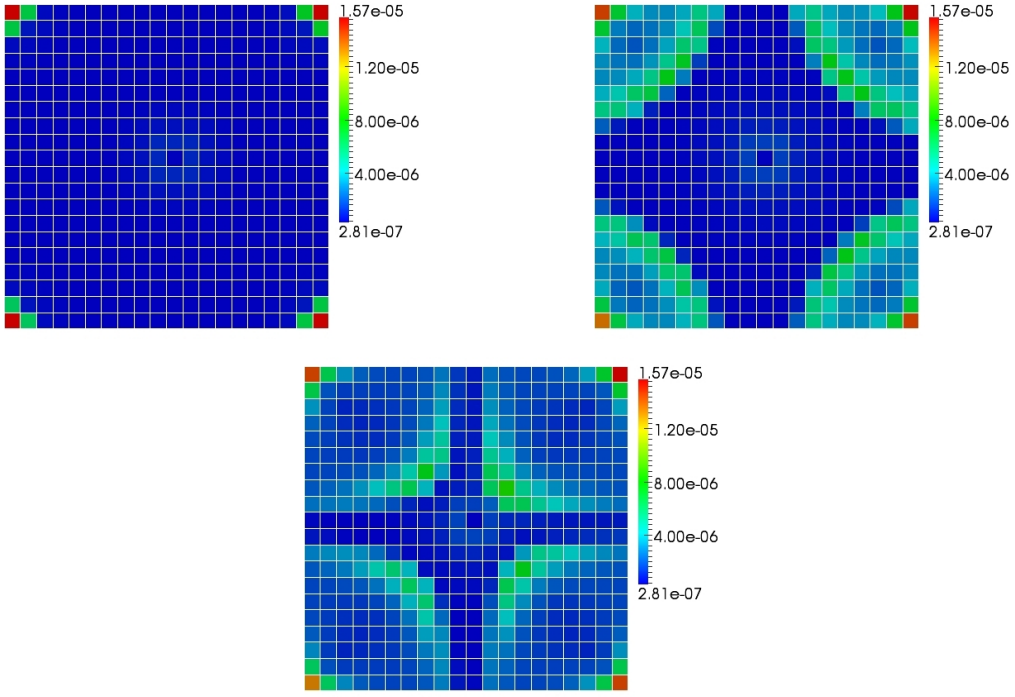

Figure 3.6: Spatial a posteriori error estimator $\eta^{n,i,j}_{\mathrm{sp},T,p}(t)$ at time $2.16 \cdot 10^4$s (top), at time $2.16 \cdot 10^6$s (middle) and at time $4.32 \cdot 10^6$s (bottom) .

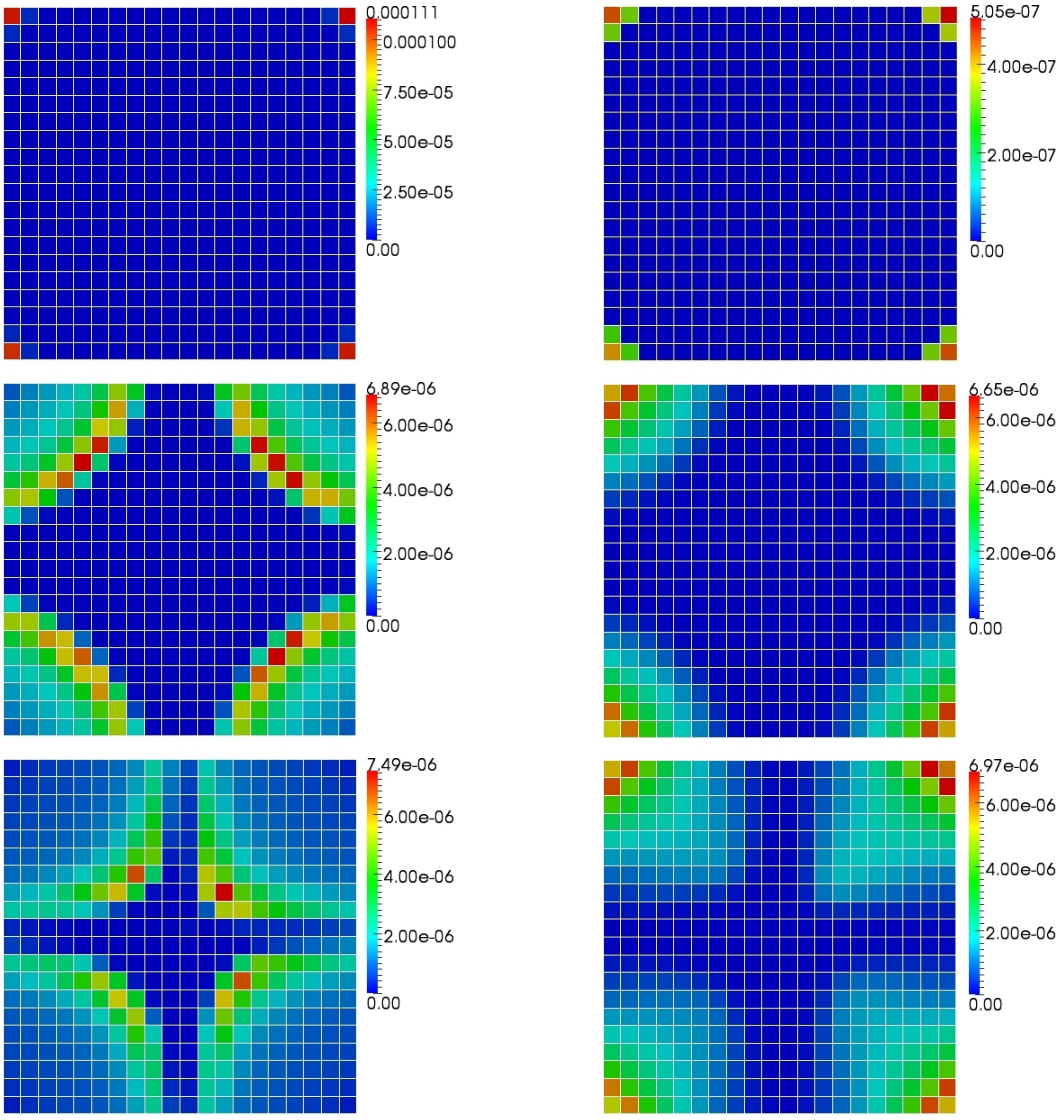

Figure 3.7: Residual a posteriori error estimator $\eta_{\mathrm{R},T,p}^{n,i,j}(t)$ (left), diffusive flux a posteriori error estimator $\eta_{\mathrm{F},T,p}^{n,i,j}(t)$ (right) at time $2.16 \cdot 10^4$s (top), at time $2.16 \cdot 10^6$s (middle) and at time $4.32 \cdot 10^6$s (bottom) .

3.6.1.2 Computational performances

We present in this section graphs that show the computational savings of the adaptive method, in the sense that it performs less iterations without loss of accuracy.

We observe in Figure 3.8 that the estimator linked to the linearization evolves as a function of the number of Newton's iterations and the algebraic error estimator evolves as function of the number of algebraic iterations. We then observe that the quantity corresponding to the algebraic error is the smallest one, compared to the other errors. Since it is the smallest one, this quantity does not affect much the global error. Moreover, it is remarkable that the other errors do not diminish during the algebraic iterations. This suggests to stop the solver when the algebraic error has the same order as the linearization error. Figure 3.9 presents the impact of the adaptive stopping criteria on the errors and on the number of Newton's and algebraic iterations. We immediatly observe that we reduce the number of algebraic solver iterations if we use the adaptive algorithm. All these results confirm the theoretical a posteriori error analysis, it confirms the interest of adaptive stopping criteria.

Let us also remark that for a numerical experiment which requires more linearization steps per time step, the iterations savings would be much greater.

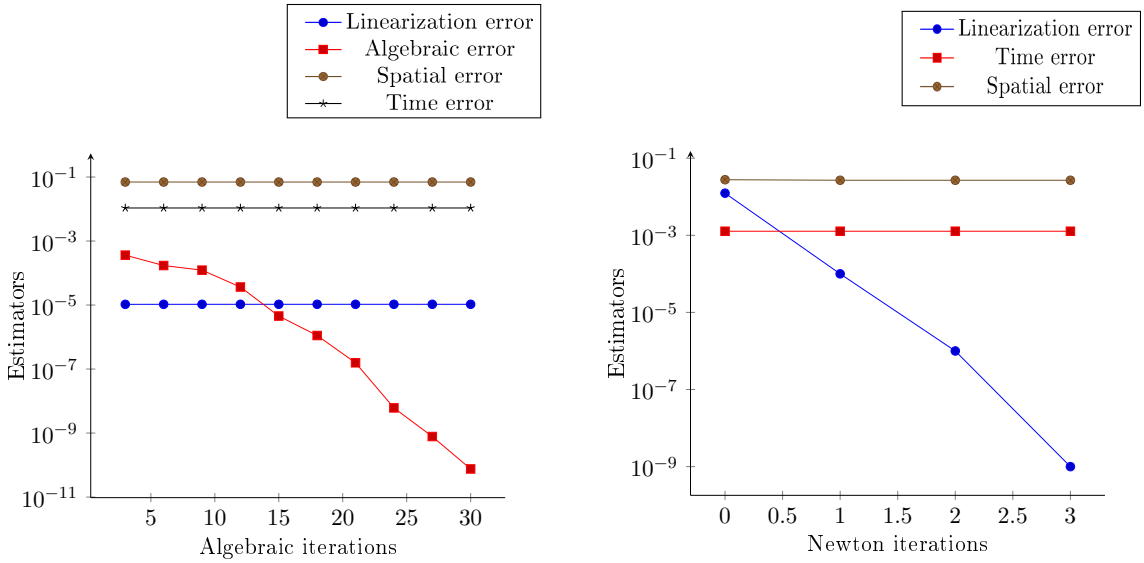

Figure 3.8: Evolution of the error estimators $\eta_{\text{sp}}^{n,i,j}$, $\eta_{\text{tm}}^{n,i,j}$, $\eta_{\text{lin}}^{n,i,j}$ and $\eta_{\text{alg}}^{n,i,j}$ evaluated as function of the algebraic iterations on the first Newton's iteration (left) and as function of Newton's iterations (right) at time $4.32 \cdot 10^4$s.

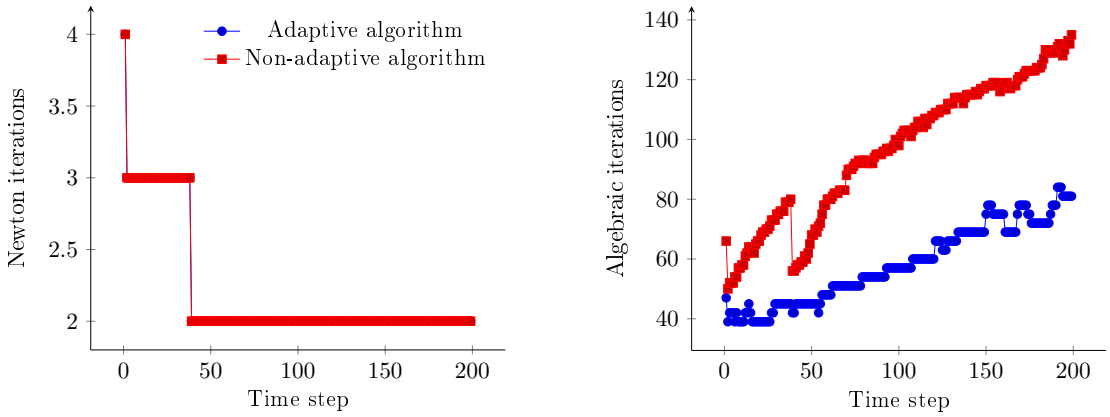

Figure 3.9: Number of Newton's iterations (left) and accumulated number of algebraic iterations (right) per time step for the non-adaptive algorithm (blue curve) and for the adaptive algorithm (red curve).

3.6.2 Heterogeneous porous media

In this section, the results presented for the adaptive algorithm are obtained using the same settings (initialization, discretization, physical laws) as in previous section; The differences is given by the medium. Let us consider a heterogeneous medium, where the permeability constants are initialized randomly by

$$\mathbf{K}_T = v_T * 1.0 \cdot 10^{-13}, \quad \text{for all } T \in \mathcal{T},$$

where $v_T \neq 0$ and $1.0 \cdot 10^{-2} \leq v_T \leq 1.0 \cdot 10^{2}$, as in Section 2.4.2. The permeability generated for this numerical experiment is shown in Figure 3.10.

Let us precise that due to the random generation of the permeability tensor, we do not compare the precision of the results between the non-adaptive and the adaptive resolution algorithm. But we observe that the solutions for the pressure and the saturations are similar, of the same order. Figure 3.11 shows the evolution of the water saturation during the simulation and Figure 3.12 shows the reference pressure. We can easily observe the water–oil front and Figure 3.13 shows that the spatial error estimator also detects this front.

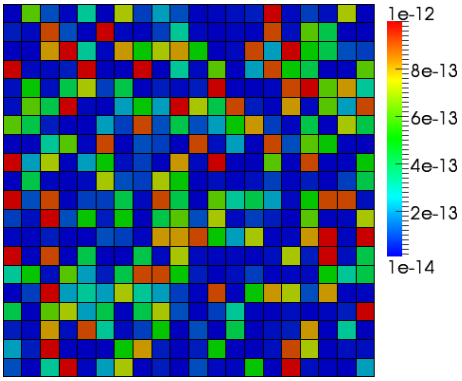

Figure 3.10: Random permeability in the domain.

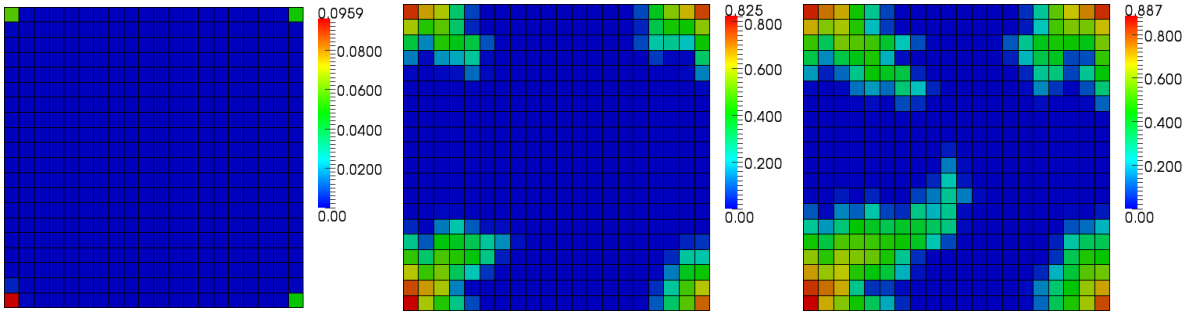

Figure 3.11: Wetting phase saturation at time $2.16 \cdot 10^4$s (left), at time $2.16 \cdot 10^6$s (middle) and at time $4.32 \cdot 10^6$s (right).

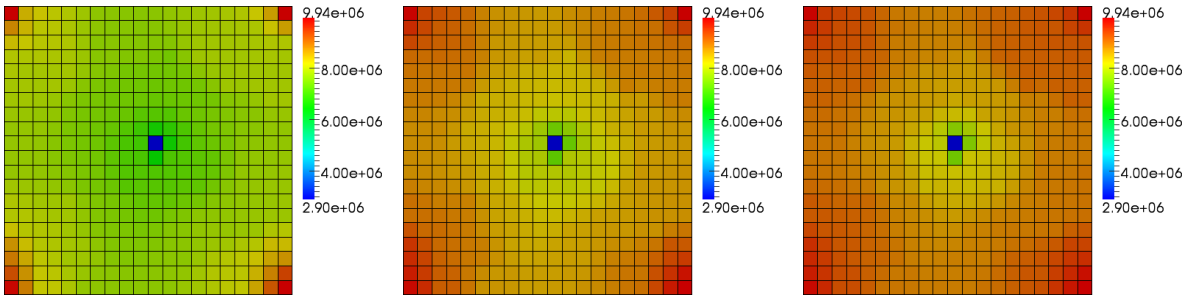

Figure 3.12: Reference pressure at time $2.16 \cdot 10^4$s (left), at time $2.16 \cdot 10^6$s (middle) and at time $4.32 \cdot 10^6$s (right).

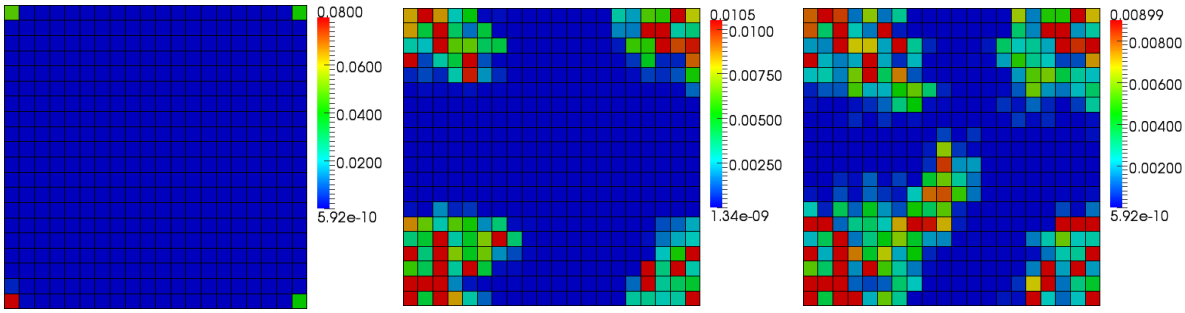

Figure 3.13: Spatial estimator at time $2.16 \cdot 10^4$s (left), at time $2.16 \cdot 10^6$s (middle) and at time $4.32 \cdot 10^6$s (right).

3.6.2.1 Computational performances

Let us present in the following figures the comparisons, in terms of the number of iterations, between two simulations for heterogeneous media.

Therefore, we compare the efficiency of the algorithm, more precisely, the number of algebraic and Newton's iterations done during a computation.

Figure 3.14 presents the total sum number of Newton's and algebraic iterations at each time step. First, we easily observe that due to small number of Newton's iterations executed to solve the discrete problem, we do not save a lot in this direction. But for other cases, needing more linearization steps, it would be also interesting. However, the picture on the right, showing the total number of algebraic solver iteration per time step, shows that a real win is made. The Figure 3.16 shows the number of algebraic and Newton's iterations executed during one time step. We focus on the first and the second time steps.

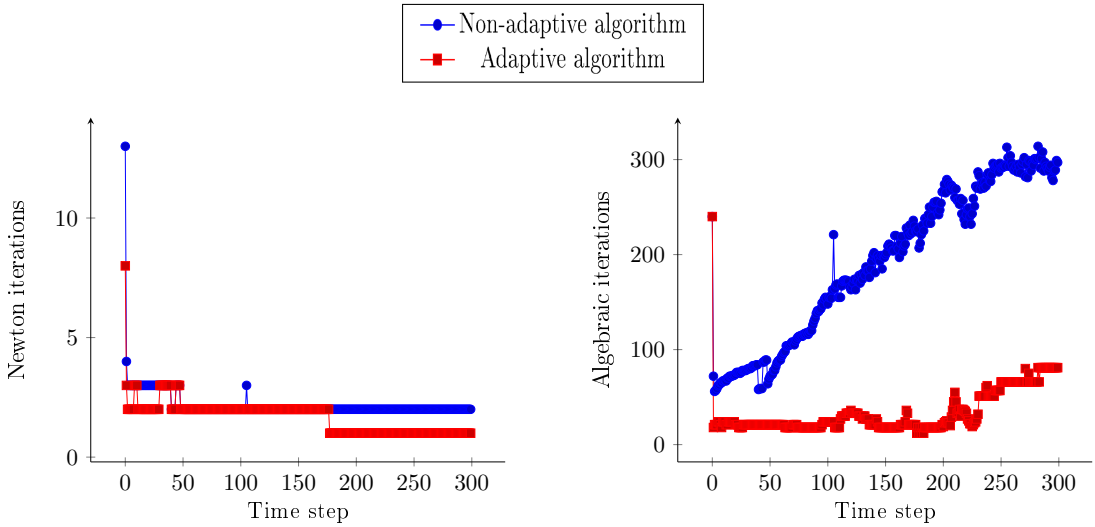

Figure 3.14: Number of Newton's iterations (left) and accumulated number of algebraic iterations (right) per time step for the non-adaptive algorithm (red curve) and for the adaptive algorithm (blue curve).

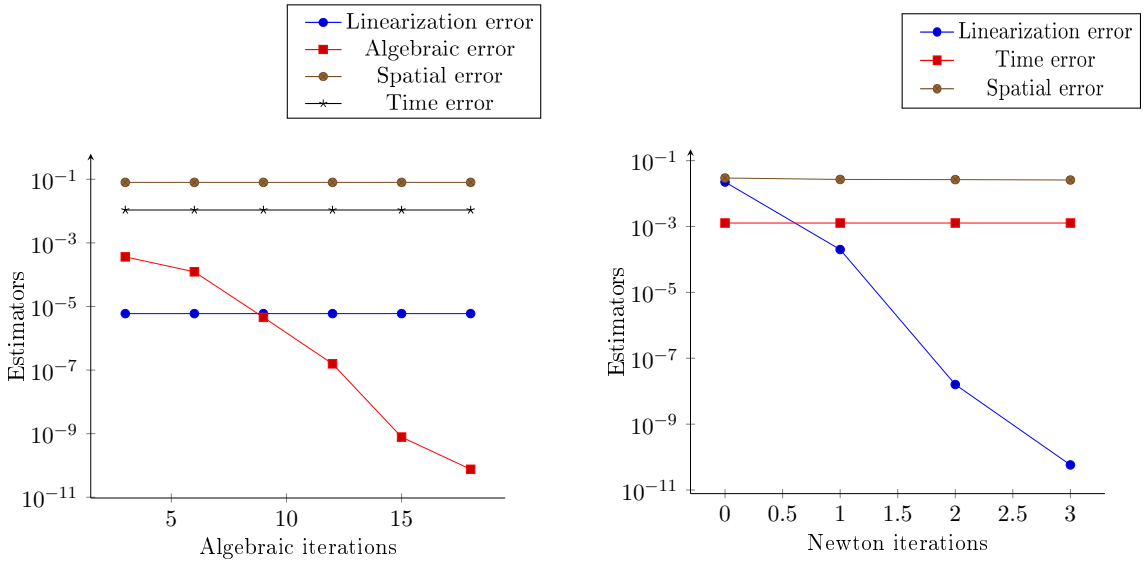

Figure 3.15: Evolution of the error estimators $\eta_{\text{sp}}^{n,i,j}$, $\eta_{\text{tm}}^{n,i,j}$, $\eta_{\text{lin}}^{n,i,j}$, and $\eta_{\text{alg}}^{n,i,j}$ evaluated as function of algebraic iterations on the first Newton's iteration (left) and as function of Newton's iterations (right) at time $4.32 \cdot 10^4$s.

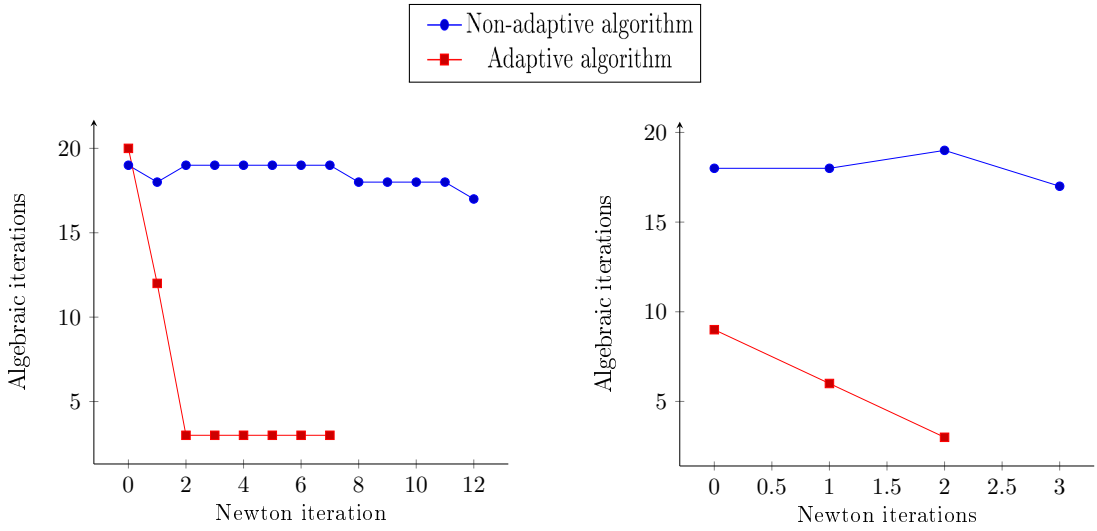

Figure 3.16: Number of algebraic iterations per Newton's iterations at time $4.32 \cdot 10^4$s (left) and at time $6.48 \cdot 10^4$s (right).

CHAPTER 4

Implementation

The ultimate goal of this work is to fully benefit from an a posteriori analysis, in order to adapt the solution algorithm and the mesh. We present in this Chapter how we choose to perform the computational resolution of the SAGD problem. We provide an overview of the facilities that have been implemented for handling the stopping criteria and adaptively refined meshes. The program is a 2D prototype, with a quadrilateral mesh, and is written in C++. We give here a simplified version of the main instructions. The complete code is the property of IFPEn.

4.1 Introduction

An important feature of the formulation of the multiphase compositional flow proposed in Chapter 2 is that we possibly have a different set of unknowns in each cell and at each time step. We present in Section 4.2 the management of unknowns during the evolution of the computation. Here the focus is on the tools that allow to deal with this situation efficiently. To do this, we develope the implementation view of the contexts, see Section 4.2.1.

Furthermore an inherent difficulty in this complex model is the description of the underlying physical system. In Section 4.3, we discuss how physical laws have been implemented to ensure efficiency of the code. In particular, we create tools to evaluate analytical functions or

operations on those functions (derivation, addition, ...).

One of our mean goals is to reduce the waste of time during simulations while optimizing the algorithm used in solving the problem. The idea developed in Chapter 3 is to stop the iterations of the linear and nonlinear solvers as soon as the algebraic and linearization errors drop to a level at which they do not affect significantly the overall error. We present how to implement the a posteriori error analysis in a resolution code in Section 4.4.

Finally, in Section 4.5, we present the particular tools implemented in processing the mesh and time adaptation. Here the key idea is to use nonconforming quadrilateral meshes derived from a topologically cartesian coarse grid. Mesh topology and metric are dealt with separately, allowing to represent non-rectangular domains.

4.1.1 General variables of the implementation

We introduce in this Section the main important variables used in the implementation; they are declared in Listing 4.1. They will be used throughout this chapter.

Listing 4.1: Declaration of general unknwowns

```
1  BooleanMatrix M;                                    // Component-phase matrix
2  UnknownsManager UM(M);                              // Unknowns manager
3  LawType zeta, kr, mu, pc;                           // Laws
4  // Mesh
5  MetricType Mh(data.mesh.hx, data.mesh.hy);
6  MeshType Th(data.mesh.nx, data.mesh.ny, data.mesh.l, Mh);
7  // Absolute permeability tensor
8  CellDiagonalTensorVariable perm;
9  // Global unknowns
10 RealMatrix X(Th.numActiveCells()+data.injectionWell.size(),
11                UM.numberOfUnknowns());
12 IntegerVector K(Th.numActiveCells());              // Context per cells
13 // Molar accumulations
14 RealMatrix N=ZeroRealMatrix(Th.numActiveCells(),N_C+UM.isThermal());
15 // Primary-secondary unknowns selector
16 UnknownsSelectorImmiscibleTwoPhase US(UM,K,X);
17 // Wells
18 std::list<InjectionWell> injection_wells;
19 std::list<ProductionWell> production_wells;
```

```
20    // Jacobian matrix
21    MatrixType J((N_C+UM.isThermal())*Th.numActiveCells(), 5*N_C);
22    RealVector x(N_C*Th.numActiveCells());        // Solution vector
23    RealVector R(N_C*Th.numActiveCells());        // residual for PDEs
24    // Model initialization
25    ModelType model(Th,laws,injection_wells,production_wells,UM,US,
26                    K,X,face_flux_lc);
27    // Residual estimators per cells and phases
28    RealMatrix ResidualEstimatorMat(Th.numActiveCells(),N_P);
29    // Quadratic pressure
30    QuadraticMap<MetricType> QuadraticPressure(Th);
31    // Quadratic capillary pressure
32    QuadraticMap<MetricType> QuadraticCapillaryPressure(Th);
33    BilinearMap<MetricType> bilinearForm(Th);      // Bilinear pressure map
34    RTNSpace<MetricType> theta(Th);                // Flux in RTN space
35    RTNSpace<MetricType> theta_bar(Th);            // Flux in RTN space
```

4.1.2 General functions of the implementation

Listing 4.2 presents some functions implemented for computing the multiphase flow, or the a posteriori error indicators in two phase flow computation. The salient features of these functions will be described subsequently in this chapter.

Listing 4.2: General functions

```
1     // Initialize and manage the unknowns
2     UM.prepare(UnknownsManagerProperty::EAll);
3     UM.compute();
4     // Compute mesh connectivity
5     Th.update(MeshTopologyTraits::ACTIVE_NODES |
6               MeshTopologyTraits::ACTIVE_FACES |
7               MeshTopologyTraits::NODE_2_ACTIVE_FACES);
8     // Initialize wells
9     initializeWells<TwoPhase,RHO>(Th,UM,data,injection_wells,
10                                  production_wells,X);
11    // Loop in time
12        // Loop on the mesh
```

```
13            ...
14            // Solve the problem
15            model.computeMolarAccumulations(N);
16            model.residual(N,N_nwt_old,R,flux,source);
17            model.Jacobian(J,RHS,flux);
18            ...
19        // End Loop on the mesh
20    // End loop in Time
```

Listing 4.2 doest not display all functions; we have only selected a few. Let us describe them briefly. Lines 2-3 prepare and compute the unknowns manager, more details are presented in Section 4.2. Then in Line 5, we initialize the mesh and the connectivity of its elements. The wells are treated in Line 9, they are initialized using the mesh, the unknowns and the data. Finally, during the resolution, at each step, the molar accumulation is computed (Line 15), the residual is evaluated (Line 16), the Jacobian matrix and the right-hand side term are assembled (Line 17). The laws management is developed in a separate section, see Section 4.3.

4.2 Unknowns management

We recall that the main difficulty of the Darcy model is the evolution (in time and in space) of the phases in the domain. We have seen that $\mathcal{C}$ and $\mathcal{P}$ denote, respectively, the sets of components and phases present in the fluid. In practice, the presence or absence of a phase p in a region causes differences in the sets of unknowns. As a phase p moves, it can appear or disappear from any region. The aim of the UnknownsManager (*UM*) is to provide representations of a minimal set of data uniquely defining the problem: the global unknowns, the contexts, the local unknowns or the components depending on the context, ...

To initialize the *UM*, we consider as unique entry point the boolean component-phase matrix $\mathbf{M} = [m_{c,p}]_{c\in\mathcal{C},\, p\in\mathcal{P}} \in \mathbb{R}^{N_\mathcal{C}, N_\mathcal{P}}$, where all entries belongs to $\{0,1\}$. The entries are defined by (2.1.1), see an example of initialization in Listing 4.1, Lines 2-3.

4.2.1 Contexts

During the implementation, a context is associated to a unique identifier integer. The family of contexts $\mathcal{K}$ is deduced from the component-phase matrix $\mathbf{M}$ in a pre-processing stage. All

combinations of phases are stored in this set of sets. Furthermore, the context k_{ref}, in which all phases are present, is called the *reference context*. The reference context provides the largest possible set of unknowns. To handle purely thermal models in the same framework, we also define the empty context $\emptyset_{\mathcal{K}} \in \mathcal{K}$ in the spirit of Remark 2.5. The context is initialized as in Listing 4.1, Line 12. From a computational point of view, we use the Standard Library (std), and the type of the context is given by std :: vector<Integer>. Note that we store one context per cell.

In practice, the set of contexts is computed during the updating stage, by means of a Flash algorithm:

```
Flash::apply(UM, Z, K(iT), X_T);
```

where the parameters are:

- The unknowns manager object, UM,
- The molar fraction, Z,
- The context K(iT) in the cell iT,
- The values of the unknowns X_T.

The Flash algorithm is mentioned in Section 2.1.1 and is described briefly in Appendix A.

Summarizing, we define an index in each cell, representing its context. It is associated to the set of present phases and produces the set of local unknowns. The contexts are updated at each time step, this yields unknowns that are local in space and in time.

4.2.2 Indexed families of subsets

The Jacobian matrix for Newton's method is constructed in Section 2.2.3. Its residual vector is the right-hand side of equation (2.3.1). Both Jacobian matrix and residual vectors are assembled by summation over sets of components, phases, or contexts. The assembling process also requires indexed families of subsets of components $\mathcal{C}$, phases $\mathcal{P}$ and contexts $\mathcal{K}$. In this paragraph, we address this issue in a systematic way.

Let $\mathcal{I} := \{\mathcal{C}, \mathcal{P}, \mathcal{K}\}$ collect the three sets $\mathcal{C}, \mathcal{P}$, and $\mathcal{K}$. We require that the subsets of any element I of $\mathcal{I}$ satisfy the following properties:

(i) Its elements are ordered. The order is arbitrary but it is fixed throughout the computation;

(ii) The standard set operations between subsets of the same set I or between subsets of different sets I, I' and I'' should be available and efficient.

In view of the latter point, by analogy with the characteristic function of a set, a convenient representation of a subset $\mathbb{S} \subset I$ is obtained by means of the vector $\mathbf{v}^{\mathbb{S}} = (v_i^{\mathbb{S}})_{i \in I} \in (0,1)^I$ such that

$$v_i^{\mathbb{S}} = \begin{cases} 1 & \text{if } i \in \mathbb{S}, \\ 0 & \text{otherwise.} \end{cases} \tag{4.2.1}$$

Then, the set operations between any two subsets can be implemented using the bitwise logical operators. Starting from the representation (4.2.1), fulfilling requirement (i) simply amounts to the construction of an iterator over the non-zero elements of the set. This can be achieved, e.g., using the library `boost::multi_index`.

The *UM* allows to access families of subsets of a set $I_2 \in \mathcal{I}$ indexed by the elements of $I_1 \in \mathcal{I}$, that is to say,

$$\mathcal{S}_{I_1 I_2} := \{\mathbb{S}_i \subset I_2\}_{i \in I_1} \qquad I_1,\, I_2 \in \mathcal{I}. \tag{4.2.2}$$

The set I_1 defines the *support type*, whereas I_2 provides the *element type*. Relevant examples of families of the form (4.2.2) include

- $\{\mathcal{C}_p\}_{p \in \mathcal{P}}$, the set of components present in a given phase ($I_1 = \mathcal{P}$, $I_2 = \mathcal{C}$);
- $\{\mathcal{P}_c\}_{c \in \mathcal{C}}$, the set of phases in which a given component is present ($I_1 = \mathcal{C}$, $I_2 = \mathcal{P}$);
- $\{\mathcal{C}_k\}_{k \in \mathcal{K}}$, the set of components present in a given context ($I_1 = \mathcal{K}$, $I_2 = \mathcal{C}$);
- $\{\mathcal{P}_k\}_{k \in \mathcal{K}}$, the set of phases present in a given context ($I_1 = \mathcal{K}$, $I_2 = \mathcal{P}$). This family of subsets formally coincides with the definition of the contexts. In the implementation, however, it may be useful to distinguish between the identifier of the context (in our case, the symbol k) and its set representation;
- $\{\overline{\mathcal{C}_k}\}_{k \in \mathcal{K}} := \{\mathcal{C} \setminus \mathcal{C}_k\}_{k \in \mathcal{K}}$, i.e., the set of components absent from a given context ($I_1 = \mathcal{K}$, $I_2 = \mathcal{C}$).

All of the above families of subsets can be computed and stored in a pre-processing stage. From the implementation standpoint, a family of the form (4.2.2) can be represented by a matrix $\mathbf{S}_{I_1 I_2} = [m_{i_1 i_2}] \in \mathbb{R}^{I_1, I_2}$, where all entries belongs to $\{0, 1\}$ such that

$$m_{i_1 i_2} = \begin{cases} 1 & \text{if } i_2 \in \mathbb{S}_{i_1}, \\ 0 & \text{otherwise.} \end{cases} \tag{4.2.3}$$

The assembly of the local Jacobian matrix $\frac{\partial \mathbf{D}_T}{\partial \mathbf{u}_T}$ is a good illustration of the use we make of the families of indexed sets introduced in this section. We briefly describe it in Algorithm 4.1.

Algorithm 4.1 Assembly of the local Jacobian $\frac{\partial \mathbf{D}_T}{\partial \mathbf{u}_T}$ in (2.3.1)

1: **for** $p \in \mathcal{P}_k$ **do** {Phases present in the context k}
2: Assemble the derivatives of the first line of eq. (2.2.28)
3: **for** $c \in \mathcal{C}_p$ **do** {Components present in the phase p}
4: Assemble the derivatives of the second line of eq. (2.2.28)
5: **end for**
6: **end for**
7: **for** $c \in \mathcal{C}_k$ **do**
8: $\mathcal{P}_{c,k} \leftarrow \mathcal{P}_c \cap \mathcal{P}_k$ {Phases present in the context k_T in which c is present}
9: **for** $p_1, p_2 \in \mathcal{P}_{c,k}$, $p_1 \neq p_2$ **do**
10: Assemble the derivatives of the third line of eq. (2.2.28)
11: **end for**
12: **end for**

In lines 1, 3, and 7 of Algorithm 4.1 we have used the indexed sets $\{\mathcal{P}_k\}_{k \in \mathcal{K}}$, $\{\mathcal{C}_p\}_{p \in \mathcal{P}}$, and $\{\mathcal{C}_k\}_{k \in \mathcal{K}}$, whereas the intersection of subsets is employed in line 8 to compute $\mathcal{P}_{c,k}$. The binary sets $\{(p_1, p_2) \in \mathcal{P}_k \times \mathcal{P}_k,\ p_1 \neq p_2\}$ can be efficiently computed using the representation (4.2.1).

4.2.3 Unknowns

A second important purpose of the *UM* is the selection of the unknowns of the model in a given context $k \in \mathcal{K}$. We stress that the aim of the *UM* is to provide a complete description of the problem based on the minimal data set synthetically represented by the matrix $\mathbf{M}$. As a consequence, unknowns partitioning related to the primary/secondary or explicit/implicit classification is out of the scope of the *UM*.

We assume an arbitrary but fixed ordering for the set of unknowns: pressure, temperature, saturations and molar fraction unknowns. Let us recall the form of the unknowns presented in Section 2.1.1 for a context $k \in \mathcal{K}$:

$$\mathbf{u}_k := \left(P, (\theta), (S_p)_{p\in\mathcal{P}_k}, (C_{p,c})_{p\in\mathcal{P}_k,\, c\in\mathcal{C}_p}\right).$$

The vector of unknowns in the reference context, $\mathbf{u}_{k_{\text{ref}}}$, is maximal, and can be used to establish the global numbering of the unknowns. For brevity of notation, when $k = k_{\text{ref}}$, we drop the index k and let $\mathbf{u} = \mathbf{u}_{k_{\text{ref}}}$. Moreover, we denote by $N_{\mathbf{u},k}$ the number of unknowns in the context $k \in \mathcal{K}$ (the index k is again omitted whenever $k = k_{\text{ref}}$). From the point of view of implementation, an unknown in a given context $k \neq k_{\text{ref}}$ can be referenced via its unique identifier in the reference context k_{ref}. To complete the set of unknowns of the compositional model, we introduce the vector of moles of the components that are only present in phases, which are absent from a given context, i.e.

$$\forall k \in \mathcal{K} \setminus \{k_{\text{ref}}\}, \qquad \mathbf{n}_k := (n_c)_{c\in\overline{\mathcal{C}_k}}.$$

Remark 4.1. There is no need to define $\mathbf{n}_{k_{\text{ref}}}$, because all phases are present in the reference context k_{ref}.

For the pressure P (Unknowns::Pressure), unknowns can be accessed by type, as follows:

- localUnknown<Unknowns::Pressure>(k) returns the index of the local unknown pressure in the context k;
- globalUnknown<Unknowns::Pressure>() returns the global index of the pressure.

In all cases, -1 is returned whenever an inappropriate request is made. The following means to access the unknowns are provided:

- For all $k \in \mathcal{K}$, $k \neq k_{\text{ref}}$, we let $\mathfrak{D}_{\Phi_k} := (0, 1, 2, \ldots, N_{\mathbf{u},k}-1)$, and $\mathfrak{C} := (0, 1, 2, \ldots, N_{\mathbf{u}}-1)$. We introduce the mapping

  $$\Phi_k : \mathfrak{D}_{\Phi_k} \to \mathfrak{C}$$

  that associates an index $j \in \mathfrak{D}_{\Phi_k}$ of the vector of unknowns $\mathbf{u}_k$ to the corresponding index $\Phi_k(j) \in \mathfrak{C}$ of the vector $\mathbf{u}$.

- The map Φ_k is a one-to-one mapping from $\mathfrak{D}_{\Phi_k}$ onto its image $\mathfrak{C}'_{\Phi_k} := \Phi_k(\mathfrak{D}_{\Phi_k})$, and therefore its inverse map Φ_k^{-1} is available. Moreover, in practice, whenever the inverse map is evaluated at an element $j \notin \mathfrak{C}'_{\Phi_k}$, an error code is returned.

- In addition to the mapping between the local numbering in the context $k \in \mathcal{K}$, $k \neq k_{\text{ref}}$, unknowns can be identified by type.

- Vector representation is available for the saturation and molar fraction unknowns, i.e.

$$\forall k \in \mathcal{K}, \qquad \mathbf{S}_k = (S_p)_{p \in \mathcal{P}_k} \text{ and } \mathbf{C}_k = (C_{p,c})_{p \in \mathcal{P}_k, c \in \mathcal{C}_p}.$$

- Given any subset $\mathbb{S} \subset \mathcal{U}$, it is possible to iterate over the unknowns of a given type (pressure, temperature, saturation or molar fraction) present in $\mathbb{S}$.

In Listing 4.3, we present an example of use of the *UM*.

Listing 4.3: UnknownsManager

```
1  // Initialize the unknowns manager with the phase-component matrix
2  UnknownsManager UM(M);
3
4  // Retrieve phases by components
5  const ISet * Pc =
6    UM.subSet(UnknownsManagerProperty::EPhasesByComponent);
7  // Retrieve phases by context
8  const ISet * Pk =
9    UM.subSet(UnknownsManagerProperty::EPhasesByContext);
10
11 // Compute the intersection P_c ∩ P_k for given c ∈ C and k ∈ K and store it in P_{c,k}
12 SetOperations::Result Pck;
13 SetOperations::Intersection(Pc->asMatrixRow(c),
14                             Pk->asMatrixRow(k),
15                             Pck);
16
17 // Compute the union P_{k1} ∪ P_{k2} for given k1, k2 ∈ K
18 SetOperations::Result Pk1k2;
19 SetOperations::Union(Pk->asMatrixRow(k1),
20                      Pk->asMatrixRow(k2),
```

```
21                              Pk1k2);
22
23      // Infer context index from a list of present phases
24      unsigned P[2] = {0, 1};
25      UM.context(P, P+1);
26
27      // return the ranges of unknowns for a given context k ∈ K
28      Range P_range = UM.range<Unknowns::Pressure>(k);
29      Range θ_range = UM.range<Unknowns::Temperature>(k);
30      Range S_range = UM.range<Unknowns::Saturation>(k);
31      Range C_range = UM.range<Unknowns::Composition>(k);
```

4.2.4 Selection of primary and secondary unknowns

The number of unknowns in the global system can be reduced using a local elimination procedure, based on the Jacobian matrix $\frac{\partial \mathbf{D}_T}{\partial \mathbf{u}_T}$ of the closure relations (2.2.35c) evaluated locally; see Section 2.3. We consider here a given time step n, $0 \leq n \leq N_{\mathrm{F}}$, and a Newton iteration k, and drop both indices to simplify the notation.

The first step consists in establishing, for all $T \in \mathcal{T}$, a suitable partition $\{\mathcal{U}_T^{\mathfrak{P}}, \mathcal{U}_T^{\mathfrak{S}}\}$ for the set of unknowns $\mathcal{U}_T$. The cardinalities of the sets $\mathcal{U}_T^{\mathfrak{P}}$ and $\mathcal{U}_T^{\mathfrak{S}}$ only depend on the context k_T, but the contents of these sets may depend on other parameters. Recall that the unknowns collected in $\mathcal{U}_T^{\mathfrak{P}}$ are termed primary unknowns, whereas those contained in $\mathcal{U}_T^{\mathfrak{S}}$ are termed secondary unknowns. Introducing an ordering in the sets $\mathcal{U}_T^{\mathfrak{P}}$ and $\mathcal{U}_T^{\mathfrak{S}}$, we obtain the vectors of primary and secondary unknowns, respectively denoted by $\mathbf{u}_T^{\mathfrak{P}}$ and $\mathbf{u}_T^{\mathfrak{S}}$. Furthermore, we recall that the selection of the primary and secondary unknowns is not generic and is closely linked to the test case.

In order to separate the unknowns, we create a class named UnknownsSelector (*US*). This class has a template argument, that permits to consider the different test case types (two or three phase flow, immiscible or miscible fluid). Each case requires a separation method.

In practice, we compute the selections in a pre-processing stage, considering all contexts and all criteria (see Section 2.3). The inputs for the *US* class are an object of type *UM* and the vector of contexts. From the standpoint of implementation, $\mathcal{U}_T^{\mathfrak{P}}$ and $\mathcal{U}_T^{\mathfrak{S}}$ are std::list of global unknowns index. Moreover, $\mathbf{u}_T^{\mathfrak{P}}$ and $\mathbf{u}_T^{\mathfrak{S}}$ are std::vector.

We also construct in each cell, the two matrices A_T and B_T described in Section 2.3, linked to the unknowns separation process. Recall that they permit to express the approximate conservation equations exclusively in terms of the primary unknowns. Some functions are briefly presented in Listing 4.4.

Listing 4.4: UnknownsSelector

```
1   // Primary or secondary unknowns selector
2   UnknownsSelectorImmiscibleTwoPhase US(UM,K,X);
3   US.compute();
4   // iT a cell number
5   // lists of primary and secondary unknowns
6   const IntegerList & Up_T=US.primary(iT);
7   const IntegerList & Us_T=US.secondary(iT);
8   // A_T and B_T are pre-computed and used in the Jacobian construction.
9   const RealMatrix & AT=US.LHSDXsOverDXp(iT);
10  const RealVector & BT=US.RHSDXsOverDXp(iT);
11  // Check if the pressure ip is a primary unknown
12  Integer ip=UM.globalUnknown(Unknowns::Pressure());
13  if(US.variableAsPrimaryUnknown(iT,ip)>=0)
14    std::cout << "Pressure is a primary unknown " << std::endl;
15  }
```

4.3 Physical laws management

The multi-phase flow model requires operations on laws describing the physics (the media, the fluids), see Section 2.1.2. We also need to build functions (molar density, capillary pressure, ...) and their derivatives. Each function and its derivatives are evaluated at each time step, in each cell. Since performing these operations represents an important work, the aim of our implementation is to compute all functions only once per time step, in each cell. Then we store the results.

We first need to define the concept of *function* and of *differentiable function*. Both concepts involve some different types: valueType, variableType, gradientType. Finally, each concept needs a routine, called eval, that computes the value of the function.

We also create a template interface to evaluate a function and its differentiable functions,

with all their arguments. This interface uses also template arguments; therefore it allows to consider different types of functions. Listing 4.5 gives some examples.

Listing 4.5: Differentiable function evaluator

```
template<typename F, typename DF, typename X>
struct IDifferentiableFunctionEvaluator{
  typedef F valueType;
  typedef DF gradientType;
  typedef X variableType;

  virtual void eval(valueType& f,
                    gradientType& df,
                    variableType& x) const =0;
}
```

One of the causes of the model difficulties is that it involves multiple laws that, moreover, depend on different sets of arguments (e.g. $\zeta_p(P, \theta, \mathbf{C}_p)$ and $P_{c_p}(S_p)$). Therefore, it is difficult to write a general program that differentiates these functions. Nevertheless, this can be achieved by the creation of a template interface that manages the laws using a *Boost* tool. This allows to treat the laws and their derivative functions considering their types as template argument for the interface.

For example, for $T \in \mathcal{T}$, let ℓ be a law, $\mathbf{u}_{k_T}$ the set of variables for the context k_T in $T \in \mathcal{T}$, and $u_T \in \mathbf{u}_{k_T}$. We easily access the values:

$$\ell(u_T) = \ell_{u_T} \qquad \text{and} \qquad \frac{\partial \ell}{\partial u}(\mathbf{u}_{k_T}) = \ell'_{\mathbf{u}_{k_T}}.$$

We also create a three-dimensional table using boost::multy_array<Real,3>. The first dimension is devoted to the laws, the second is devoted to their evaluations (function values and the values of its first derivatives) and the third dimension is devoted to the cells (for each cell, we have the set of laws, their evaluations and the evaluation of their first derivatives). This tabular dimension is: $N_{laws} \times (1 + N_{variables}) \times N_{cells}$.

When we initialize a set of laws (e.g. $\{\zeta, P_c, \nu\}$), we evaluate all laws and their partial derivatives for each cell with the function eval. This three-dimensional table containing the laws and their evaluations is stored.

To access the evaluated results, we can identify the laws by their names (string type) or by

their own index (integer type), see Listing 4.6. From a computational point of view, searching the laws by their indices is substantially less time-consuming than searching by their names.

Listing 4.6: LawSet initialization and use for a phase p

```
// initialization of the law set for the phase p, X is the vector of unknowns
LawSet<Real> laws_p;
laws_p << std::make_pair("molarDensity", zeta_p)
       << std::make_pair("capillaryPressure", pc_p)
       << std::make_pair("mobilityKr",
                         calculator(zeta_p*kr_p/{mu_p}));
laws_p.eval<MetricType,MetricType::rho,
            UnknownsSelectorImmiscibleTwoPhase>(Th,US,X);
// iT a cell number
Real PcT=laws_p.f("capillaryPressure", iT);
Integer molarDensity_index=laws[iP].index("molarDensity");
Real DMolarDensityTDX=laws_p.dfDx(molarDensity_index,iT);
```

In practice, to construct the Jacobian matrix, we consider the laws by phases. For this, we create a vector of set of laws using *std::vector<LawSet<Real> >*. Before assembling the Jacobian matrix, we evaluate the functions and their derivatives. Then, we store all quantities.

Listing 4.7: LawSet Vector initialization and use for 2 phase

```
// initialization of the vector with 2 lawSet
ModelType::LawSetVector laws(2);
laws[0]=laws_0;
laws[1]=laws_1;
//evaluation of the laws and their derivatives for the phase ip={0,1}
for(Integer ip=0;ip!=2;ip++)
    laws[ip].eval<MetricType,
                  MetricType::rho,
                  UnknownsSelectorImmiscibleTwoPhase>(Th,US,X);
//values of the mobility evaluated in the cell T
Real ZetaT=laws[0].f("mobility", iT);
Integer mobility_index=laws[iP].index("mobility");
ZetaT=laws[1].f(mobility_index, iT);
```

4.4 A posteriori estimators for adaptive algorithm

The aim of a posteriori error estimators is to identify the origin of the error in the simulation. The error indicators represent the distribution of errors due to the discretization (in space or in time) or due to the algorithm solving the problem (linearization algorithms or algebraic resolution of linear systems). We use an a posteriori error analysis to optimize the computation. Our theoretical and numerical results are presented in Chapter 3. In this section, we describe how we implement the estimators.

To store all estimators, at each (time, linear or algebraic) iteration, in each cell, we use a boost::vector<Real>. Furthermore, to implement the a posteriori error estimators, for example global estimators defined in Section 3.5.5, we define in Section 4.4.1, a method to approximate the integration. Furthermore, to build the a posteriori error estimators, we need tools to reconstruct different quantities. We also need different maps which handle the different items (affine functions, quadratic and bilinear forms or the RTN Space). In Section 4.4.2 we describe how we deal with these issues. Finally, in Section 4.4.3, we explain the facilities implemented for the user.

4.4.1 Quadrature method

We need to discretize integrals in order to evaluate the local and the global estimators. To this end, we use the *Gauss quadrature method*. In practice, a std::map is created, which permits to choose the desired accuracy. For this, the number n of quadrature nodes is given by the user. Then, the corresponding quadrature coefficients are computed. These are called the weights w_i, with $0 \leq i \leq n$. In 1D, the Gauss quadrature has the form:

$$I = \int_a^b \eta(x)\, dx \sim \sum_{i=1}^{n} w_i\, \eta(x_i), \tag{4.4.1}$$

where the nodes x_i are distinct, and strictly contained in $]a, b[$. This method is exact for polynomials of degree $2n - 1$.

4.4.2 Reconstructions

The reconstructions, defined in Section 3.4, are operations that transform point values into functions. They can be viewed as a regularization or an interpolation. They are necessary

because the indicators are defined on functions (mostly in RTN spaces) and not on point values.

Affine functions We need piecewise affine functions in time for example to interpolate the pressure (recall that $P_{h\tau}(t^n) = P_h^n$). To this end, we define two eval functions, one for affine functions and the other one for their derivatives.

Piecewise constant and constant functions As in Section 3.5.3, in order to approximate the pressure, we use piecewise quadratic pressure in each cell and globally continuous bilinear functions in each cells. We create two maps to achieve this reconstruction:

- The QuadraticMap<**typename** MetricType>(**const** MeshType & Mesh);
- The BilinearMap<**typename** MetricType>(**const** MeshType & Mesh);

both with the mesh as sole argument. We then compute in each cell the coefficients a_T^n, b_T^n, c_T^n, d_T^n, and e_T^n of the quadratic form with the approximate values of the pressure P_T^n, the fluxes $F_{p,T,\sigma}^n$, and the permeability tensor κ_T (see Section 3.4.1.2 for more details on the construction method). Listing 4.8 presents how we compute the value of the function and its derivatives at a point.

The same work is done for the continuous pressure interpolation, presented in Section 3.5.2: We implement a bilinear map, and we compute for each cell its coefficients a_T^n, b_T^n, c_T^n, and d_T^n that determine the polynomial function and its derivatives at every point in the domain. These coefficients are obtained by nodal averages. Summing up, we build a class that computes and stores these nodal averages, the NodeAveragingInterpolation<MetricType> class. The arguments are the mesh and the discontinuous quadratic pressure, computed previously. Listing 4.8 shows this object in Line 8-9, then in Line 10, we give an abridged version of a continuous bilinear pressure reconstruction. Lines 16-17 presents examples of the eval functions.

Flux reconstruction In order to reconstruct fluxes, we need to create continuous velocities, as described in Section 3.5.1. For this we implement a map that manipulates Raviart Thomas Nédélec spaces. It evaluates in particular, for all cells, the values of the coefficients a_T^n, b_T^n, c_T^n, and d_T^n, such that the functions have the form:

$$\boldsymbol{\chi}_{p,h}^n|_T = \begin{pmatrix} a_T^n \\ b_T^n \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} c_T^n \\ d_T^n \end{pmatrix}.$$

These continuous velocities are construct in each cell using the values of the fluxes on the faces. The Listing 4.8 shows in Lines 12-13, how to initialize a RTNSpace<MetricType>(**const** MeshType & mesh), and in Line 18, how the function is evaluated at a point.

Listing 4.8: Reconstruction functions

```
1  // Quadratic pressure and bilinear pressure creation
2  QuadraticMap<MetricType> QuadraticPressure(Th);
3  BilinearMap<MetricType> bilinearForm(Th);
4  // Pressure global index
5  Integer iPressure=UM.globalUnknown(Unknowns::Pressure());
6  QuadraticPressure.compute(X(iPressure),flux_F,perm);
7  // Pressure interpolation
8  NodeAveragingInterpolation<MetricType>
9       NodePressureInterpolation(Th, QuadraticPressure;
10 bilinearForm.compute(NodePressureInterpolation);
11 // Flux construction in RTN space
12 RTNSpace<MetricType> theta(Th);
13 theta.compute(flux_F);
14
15 // P is a point in the 2D cell T
16 Real Pressure=QuadraticPressure.eval(P,T);
17 Point DbilinearDX=bilinearForm.gradient(P,T);
18 Point u=theta.eval(P,T);
```

4.4.3 Implementation facilities of the method

The a posteriori error estimators have been implemented as a user-friendly module that is presented as an "independent bloc" in the resolution algorithm. In this block we:

- Create all reconstructions used to evaluate the estimators.
- Store all quantities evaluated in the basic algorithm (fluxes, unknowns, source terms).
- Perform all operations required by the a posteriori error analysis.
- Compute the a posteriori error estimators.

The estimators are ultimately used to stop the iterations.

In Algorithm 4.2, we present a coarse view of the a posteriori error analysis and all steps added to the basic algorithm. It shows that the a posteriori error evaluation is inserted as an overlay, which does not affect the remaining part of the method. More details are presented in Section 3.5.6.

Algorithm 4.2 Overview of the adaptive algorithm

Initialization of the quantities, the stopping criteria
\\ *Time loop*
while Evaluation time is smaller than the final time **do**
  Update the settings
  \\ *Newton linearization loop*
  while Newton linearization stopping criterium not reached **do**
    Initialize the quantities, set up the system
    \\ *Algebraic iterations*
    while Algebraic stopping criterium not reached **do**
      Set up the iterative solver
      \\ ———————————————————————————
      \\ *A posteriori error estimators part*
      Reconstruct the functions
      Evaluate the local indicators and their global version
      Update the algebraic and linear stopping criteria
      \\ *End of the a posteriori error estimator part*
      \\ ———————————————————————————
    end while
    Update the unknowns
  end while
  Update the unknowns for the current time step
end while

4.5 Mesh adaptation

The simulations of the SAGD process are extremely sensitive to the grid size. Indeed, this process creates flow interfaces, between oil and water for example. One of the issues of the simulations is to follow those interfaces, named fronts. Those fronts are thin in comparison with the reservoir size (of the order of a meter for the front versus a kilometer for the reservoir). Thus, one of the difficulties of this numerical simulation is to deal with this constraint. The common strategy is to perform a dynamic mesh adaptation, which evolves in function of the front. A posteriori error estimators can help locating the cells that need to be refined or coarsened, see for example [57].To this end, we create some tools to locally refine or coarsen cells. Then we need to update all the data at each change.

The initial mesh is a conforming rectangular mesh. The mesh is composed of cells, nodes and faces. To initialize a mesh, we use the class *Mesh*, from which the class *MeshTopology* (*MT*) inherits. We also need to input the number of coarse cells in all directions for the first level (level 0), the number of refinement levels we want, and the type of metric we choose. Finally, we can define an integer parameter *RHO* which defines the division factor for the refinement order. For example, if *RHO* is an integer, to refine a cell, we divide it into *RHO* cells in each direction.

MT then creates ordered indices for cells, nodes, and faces. Finally, at initialization, all elements of the mesh at level zero become active using the boolean value *true*, whereas all other elements are initialized with an inactive statut thanks to the boolean values *false*.

To summarize, all elements have general mesh properties as:

- The *global index* which permits to distinguish or to compare two elements which are not necessarily at the same stage.

- The *refinement level*, the different levels correspond to the different stages of refined mesh. At the beginning, the mesh is composed of coarse cells, whose levels are 0. Then, each refinement step corresponds to a higher level.

- The *state index* is a boolean value expressing if they are active or not. Due to operations applied on the mesh, all cells can be active or not. This means that if a "parent" cell is refined, into several "children" cells, the "parent" cell becomes inactive, whereas the "children" cells become active. On the contrary, if a group of "children" cells are coarsened

into a "parent" cell, all the "children" cells become inactive while the "parent" cell becomes active.

Listing 4.9: Mesh initialization

```
1 integer RHO; // factor of cell division for the refinement step
2 Integer N, M; // number of cells in x-direction, and y-direction
3 ConstantMetric<RHO> Mh(1./N, 1./M); // unit square domain
4 Mesh<ConstantMetric<Rho> > Th(N, M, L, Mh); // initial mesh
```

We consider a mesh with two coarse cells in the x-direction and two coarse cells in the y-direction. We define RHO $= 2$. See in Figure 4.1, the mesh at initial level, only composed by coarse cells (level 0, colored in yellow). In figure 4.2, the mesh has been refined and is composed of three different levels of cells (level 0; level 1, in orange; level 2, in red). Figures 4.1–4.3, present the different active cells and their different levels.

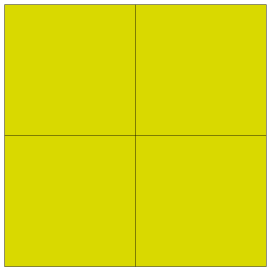

Figure 4.1: Trivial example of a coarse mesh.

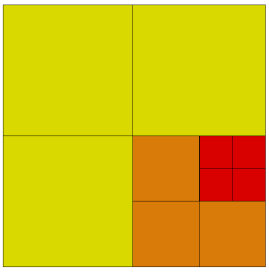

Figure 4.2: Trivial example of a refined mesh.

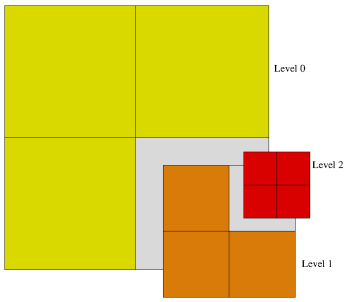

Figure 4.3: The different level of the refined mesh.

The multi-phase flow model imposes to manage sets and subsets of cells or faces, for example for the loop over the faces in the Jacobian construction, e.g. (2.2.31) or (2.2.34). Thus, the *MT* class permits to obtain some mesh characteristics, e.g. the number of (active) cells, nodes or faces, or the boundary faces. We also have access to list of elements which are linked to each other, for example, the set of neighboring cells for a given cell, or the set of nodes belonging to a face, or the set of faces touching a node. Finally the code returns iterator couples to browse the different lists of elements we get; It is useful to build loops. We also use *Boost* loop: BOOST_FOREACH(element, elementList){...}, and we easily loop on each type of list, see Listing 4.10 to have examples.

Listing 4.10: Subset access using *MT*

```
// Let T be the mesh
// Loop over the active cells
BOOST_FOREACH(const Cell & T, Th.getActiveCellList()) {
    Real mT=Th.getMeasure(T);
    const Cell T_active=Th.findCellActiveParent(T);
    BOOST_FOREACH(const Face & F, Th.getCellFaceList(T)){
        Integer iF=Th.getFaceIndex(F);
   }
}
```

With *MT*, a mesh can be refined or coarsened, depending on the criterion we choose. We also use the parameter *RHO* to define how a cell can be refined or coarsened. Furthermore, we impose that two neighboring cells, cannot have more than two level difference. So using a specific criterion, each cell can be refined in $RHO \times RHO$ "children" cells or considering *RHO* neighboring cells in all directions, they can be coarsened into a "parent" cell, see Figure 4.4–4.5 to have a schematic representation.

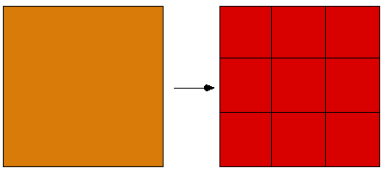

Figure 4.4: Refining process with $RHO = 3$.

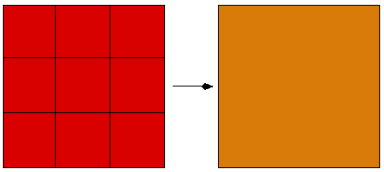

Figure 4.5: coarsening process with $RHO = 3$.

Example 4.2 (Refined mesh)**.** We consider a coarse mesh with 4 cells in the x-direction and 4 cells in the y-direction. We have RHO = 2. Listing 4.11 presents briefly a function to test the refinement tools. Then Figure 4.6 show the mesh at initial level and Figure 4.7 the refined mesh at level 2 with three different levels of refined cells.

Listing 4.11: Refined functionalities

```
  // v is an eval function
  MyFunction v;
  //v_h is the vector of results for all cells
  CellRealVariable vh;
```

```
5   // Cell set initialization
6   MeshTopologyTraits::CellSet refine, coarsen;
7   // tag the cells to refine and to coarsen
8   Th.tagCells(refine, coarsen);
9   // update the mesh
10  Th.update(MeshTopologyTraits::ACTIVE_FACES |
11  MeshTopologyTraits::NODE_2_NONCONFORMING_FACES);
12  // evaluation of the function v on all cells
13  // results are stored in the vector v_h
14  BarycentricInterpolator_T::eval(Th, vh, v);
```

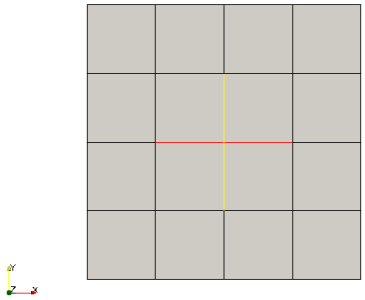

Figure 4.6: Coarse mesh (level= 0).

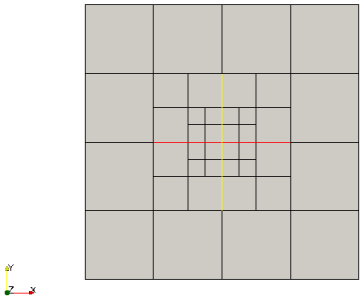

Figure 4.7: Refined mesh (level= 2)

Finally the tools created to manage the different mesh elements, such as cells, faces, or nodes, are easily available and optimize as much as possible the storage of all the relevant data.

CHAPITRE 5

Conclusion et perspectives

5.1 Conclusion du manuscrit

Dans cette thèse, nous avons d'une part présenté l'ensemble des équations d'un modèle compositionnel de Darcy, où nous avons présenté une méthode de réduction des nombres d'inconnues. Nous avons aussi :

- proposé un schéma discret par des volumes finis en espace et un schéma de Euler implicite retrograde en temps,
- proposé un algorithme de linéarisation de Newton,
- présenté les principaux aspects de leur implémentation.

D'autre part, nous avons étudié de plus près le cas simplifié d'un modèle de Darcy diphasique immiscible et isotherme. Nous avons choisi les espaces des inconnues, mis le problème sous forme variationnelle et démontré l'équivalence avec le problème d'origine, sous des hypothèses convenables. Nous l'avons discrétisé en espace sur une grille rectangulaire par un schéma de volumes finis à deux points avec décentrage en amont des flux diffusifs.

Nous avons ensuite défini une norme faible du résidu, établi des indicateurs d'erreur et majoré cette norme faible par ces indicateurs. Enfin, nous avons utilisé ces indicateurs pour

contrôler le nombre d'itérations de l'algorithme de Newton et du solveur linéaire. Ces algorithmes et des outils de gestions du raffinement de maillage ont été implémentés dans un prototype et nous avons validé l'étude théorique par des essais numériques.

5.2 Perspectives futures

Etant donné la difficulté mathématiques du problème étudié et l'intérêt que présentent l'usage des estimateurs d'erreur a posteriori pour optimiser les choix des critères d'arrêt des solveurs non linéaire et linéaire, il serait intéréssant de

- considérer des schémas multipoints avec des grilles déformées (quadrilatères ou hexaèdres),
- simplifier la formulation des indicateur d'erreur, pour éviter les reconstructions dans les bases de Raviart-Thomas-Nédélec,
- procéder à des comparaisons entre différents estimateurs de l'erreur.

APPENDIX A

Thermodynamic Flash

A.1 Introduction

Let us recall that in the Darcy model, the context is a number that represents the set of phases present at a given place and a given time. The context is denoted by $k \in \mathcal{K}$ and is discretized for the computation. Furthermore, as the context evolves during the computation, it is treated as an unknown. We have to determine it at the beginning and then we compute at each time step n, in each cell T, the context k_T^n. The thermodynamic Flash algorithm gives, for a given pressure, temperature, and fluid composition, the phases that are present and their respective compositions. The algorithm is briefly described in this section, more details can be found in the work of Cao [23] and the thesis Guichard [52].

First we recall the expression of the total molar fraction Z_c defined by (2.1.16) for all $c \in \mathcal{C}$

$$Z_c = \frac{\sum_{p \in \mathcal{P}_{\mathfrak{K}} \cap \mathcal{P}_c} \zeta_p(P,(\theta),\mathbf{C}_p) S_p C_{p,c} + \sum_{c' \in \overline{\mathcal{C}_{\mathfrak{K}}} \cap \{c\}} n_{c'}}{\sum_{c' \in \mathcal{C}} \sum_{p \in \mathcal{P}_{\mathfrak{K}} \cap \mathcal{P}_{c'}} \zeta_p(P,(\theta),\mathbf{C}_p) S_p C_{p,c} + \sum_{c' \in \overline{\mathcal{C}_{\mathfrak{K}}}} n_{c'}}, \tag{A.1.1}$$

collected in the vector $\mathbf{Z} = \{Z_c\}_{c \in \mathcal{C}}$ and such that $\sum_{c \in \mathcal{C}} Z_c = 1$.

The entries of the algorithm are,

- the pressure P,
- the temperature (for thermal cases) θ,

- the total molar fraction Z_c.

The purpose of the algorithm is to find $\mathcal{K}$ the context, $(\mathbf{C}_p)_{p\in\mathcal{P}}$ the compositions of the component c in the phase p, and $(\vartheta_p)_{p\in\mathcal{P}}$ the molar fractions of the phase p in the fluid:

$$\text{Flash}(P,(\theta),\mathbf{Z}) \Rightarrow \mathcal{K},(\mathbf{C}_p)_{p\in\mathcal{P}},(\vartheta_p)_{p\in p}. \tag{A.1.2}$$

Let us restrict the description to isothermal cases and disregard the temperature unknown. The definition of a Flash algorithm depends on the properties of the fluid under consideration: phases, components, miscibility. In petroleum industry, setting up a Flash algorithm is a substantial part of the programming load. Here we present a Flash algorithm for an immiscible two-phase flow.

A.2 Two-phase flash algorithm

In two-phase flows, $k_{\text{ref}} = \{p_1, p_2\}$, the two phases composing the fluid, and the number of components $N_{\mathcal{C}} = 2$. We suppose Z_c known by formula (A.1.1), such that $0 \leq Z_c \leq 1$ and $\sum_{c\in\mathcal{C}} Z_c = 1$.

The algorithm requires equilibrium constants of each component $c \in \mathcal{C}$ denoted by K_c. These non negative constants depend on the reference pressure P and on the characteristics of the model. They are part of the data, obtained from field experiments, provided by reservoir engineers.

We search for ϑ_1 and ϑ_2, and the molar compositions $\mathbf{C}_{p_1} = (C_{p_1,c})_{c\in\mathcal{C}}$ and $\mathbf{C}_{p_2} = (C_{p_2,c})_{c\in\mathcal{C}}$, by solving the following system of equations:

$$\begin{cases} C_{p_2,c} - K_c C_{p_1,c} = 0,\ c \in \mathcal{C}, & \text{thermodynamic equilibrium,} \\ \displaystyle\sum_{c\in\mathcal{C}} C_{p_1,c} = 1, & \text{concentration balance of } p_1, \\ \displaystyle\sum_{c\in\mathcal{C}} C_{p_2,c} = 1, & \text{concentration balance of } p_2, \\ Z_c = \vartheta_1 C_{p_1,c} + \vartheta_2 C_{p_2,c}, & \text{material equilibrium,} \end{cases} \tag{A.2.1}$$

where $C_{p_1,c} \geq 0$ and $C_{p_2,c} \geq 0$, for $c \in \mathcal{C}$. Since $N_{\mathcal{C}} = 2$, this system consists of 6 independent equations in 6 unknowns.

Remark A.1. We observe that $\vartheta_1 + \vartheta_2 = 1$. Indeed,

$$\begin{aligned} 1 =& \sum_{c\in\mathcal{C}} Z_c = \sum_{c\in\mathcal{C}} \vartheta_1 C_{p_1,c} + \sum_{c\in\mathcal{C}} \vartheta_2 C_{p_2,c} \\ =& \vartheta_1 \sum_{c\in\mathcal{C}} C_{p_1,c} + \vartheta_2 \sum_{c\in\mathcal{C}} C_{p_2,c} \\ =& \vartheta_1 + \vartheta_2. \end{aligned} \tag{A.2.2}$$

Thus we write $\vartheta = \vartheta_1$ to simplify the notation:

$$Z_c = \vartheta C_{p_1,c} + (1-\vartheta) C_{p_2,c}. \tag{A.2.3}$$

Remark A.2. When the component c is missing from the phase p_2, we set $K_c = 0$. Then $K_c C_{p_1,c} = 0$ and the first line of (A.2.1) gives

$$C_{p_2,c} = 0.$$

On the other hand, when the component c is missing from the phase p_1, we divide the first line of (A.2.1) by K_c and we set $K_c = +\infty$. Then

$$\frac{1}{K_c} C_{p_2,c} = 0,$$

and the first line of (A.2.1) gives

$$C_{p_1,c} = 0.$$

From (A.2.3) and from the first line of (A.2.1), we deduce for each component $c \in \mathcal{C}$

$$C_{p_1,c} = \frac{Z_c}{\vartheta + (1-\vartheta)K_c} \tag{A.2.4}$$

and

$$C_{p_2,c} = \frac{K_c Z_c}{\vartheta + (1-\vartheta)K_c}. \tag{A.2.5}$$

Therfore, the second and third lines of (A.2.1) imply

$$\sum_{c\in\mathcal{C}} \frac{Z_c}{\vartheta + (1-\vartheta)K_c} = 1 = \sum_{c\in\mathcal{C}} \frac{K_c Z_c}{\vartheta + (1-\vartheta)K_c}.$$

Hence

$$\sum_{c\in\mathcal{C}} \frac{Z_c(K_c - 1)}{\vartheta + (1-\vartheta)K_c} = 0.$$

This motivates the definition of the Rachford-Rice function:

$$f^{RR}(\vartheta) = \sum_{c\in\mathcal{C}} \frac{Z_c(K_c-1)}{\vartheta + (1-\vartheta)K_c}. \tag{A.2.6}$$

We want to solve for the roots ϑ of

$$f^{RR}(\vartheta) = 0, \tag{A.2.7}$$

provided that the product $Z_c(K_c-1)$ does not vanish for all $c \in \mathcal{C}$. To this end, we introduce the set

$$\tilde{\mathcal{C}} = \{c \in \mathcal{C} \,|\, K_c \neq 1 \text{ and } Z_c \neq 0\}.$$

If $\tilde{\mathcal{C}} = \emptyset$ then for all $c \in \mathcal{C}$, the product $Z_c(K_c-1)$ vanishes and (A.2.7) does not determine ϑ since it is satisfied for all ϑ. On the other hand, if $\tilde{\mathcal{C}} \neq \emptyset$ then we have the following proposition.

Proposition A.3 (Flash resolution)**.** *If $\tilde{\mathcal{C}} \neq \emptyset$, then solving the system* (A.2.1) *is equivalent to finding the roots ϑ of $f^{RR}(\vartheta)$.*

Proof. We have seen that (A.2.1) leads to (A.2.7). Conversely, let ϑ be a root of $f^{RR}(\vartheta)$. Then, (A.2.4) and (A.2.5) define $\mathbf{C}_{p_1}$ and $\mathbf{C}_{p_2}$, and the first and the fourth lines of (A.2.1) are satisfied. Moreover, the relation $\sum_{c\in\mathcal{C}} Z_c = 1$ implies that

$$\vartheta \sum_{c\in\mathcal{C}} C_{p_1,c} + (1-\vartheta)\sum_{c\in\mathcal{C}} K_c C_{p_1,c} = 1. \tag{A.2.8}$$

On the other hand, by substituting (A.2.6) into (A.2.7), and using the expressions (A.2.4) and (A.2.5) for $\mathbf{C}_{p_1}$ and $\mathbf{C}_{p_2}$, we deduce that

$$\sum_{c\in\mathcal{C}} C_{p_1,c} = \sum_{c\in\mathcal{C}} C_{p_2,c}. \tag{A.2.9}$$

Finally, the second and third lines of (A.2.1) follow from (A.2.8) and (A.2.9). □

If $Z_c(K_c-1)$ does not vanish for any c, the Rachford-Rice function (A.2.6) has $N_{\mathcal{C}}$ poles, and each pole has the expression:

$$\vartheta_c = 1 - \frac{1}{1-K_c}, \text{ for } c \in \tilde{\mathcal{C}}.$$

We then use the extended Flash method, which looks for $\vartheta \in \mathbb{R}$ (not only on $[0,1]$). We define:

$$\begin{cases} K_{\max} = \sup_{c\in\tilde{\mathcal{C}}} K_c \\ K_{\min} = \inf_{c\in\tilde{\mathcal{C}}} K_c. \end{cases} \tag{A.2.10}$$

and the poles of ϑ belongs to $[\vartheta_0, \vartheta_1]$ where

$$\begin{cases} \vartheta_0 = 1 - \frac{1}{1-K_{\max}} \\ \vartheta_1 = 1 - \frac{1}{1-K_{\min}}. \end{cases} \tag{A.2.11}$$

We also determine ϑ, such that $f^{RR}(\vartheta) = 0$. Using this method, the continuity of the different quantities are conserved, even if the context changes, and the thermodynamic conservation equality is respected:

$$C_{p_1,c} - K_c C_{p_2,c} = 0, \text{ for all } c \in \mathcal{C}. \tag{A.2.12}$$

Finally, the results are treated depending on three configurations:

1. If $\vartheta_0 \leq 0$ (i.e. $K_{\max} \leq 1$) then the extend flash gives no positive solutions $C_{p_1,c}$ and $C_{p_2,c}$ unless when $\vartheta \to -\infty$. The Flash gives a single phase state p_2.

$$\begin{cases} \vartheta & = -\infty, \\ C_{p_2,c} & = Z_c, \text{ for } c \in \mathcal{C}. \end{cases} \tag{A.2.13}$$

2. If $\vartheta_1 \geq 1$ (i.e. $K_{\min} \geq 1$) then the extend flash gives no positive solutions $C_{p_1,c}$ and $C_{p_2,c}$ unless when $\vartheta \to +\infty$. The Flash gives a single phase state p_1.

$$\begin{cases} \vartheta & = +\infty, \\ C_{p_1,c} & = Z_c, \text{ for } c \in \mathcal{C}. \end{cases} \tag{A.2.14}$$

3. For the other cases, the Rachford-Rice function admits an unique solution $\vartheta \in [0,1]$, and gives $C_{p_1,c}$ and $C_{p_2,c}$, both positive numbers. We have

$$\begin{cases} \vartheta & \in [0,1], \text{ s.t.} f^{RR}(\vartheta) = 0, \\ C_{p_1,c} & = \frac{Z_c}{\vartheta + (1-\vartheta)K_c}, \text{ for } c \in \mathcal{C}, \\ C_{p_2,c} & = K_c C_{p_1,c}, \text{ for } c \in \mathcal{C}. \end{cases} \tag{A.2.15}$$

Table of notation

Symbol	Definition	Section	Page
t_{F}	The simulation time	2.1	26
$N_{\mathcal{P}}$	Number of phases	2.1	26
$N_{\mathcal{C}}$	Number of components	2.1	26
$\mathcal{P}$	Set of phases p	2.1	26
$\mathcal{C}$	Set of components c	2.1	26
$\mathbf{M}$	Boolean component-phase matrix	2.1	26
I_θ	Thermal index	2.1	27
$\mathcal{K}$	Set of contexts k	2.1	27
k_{ref}	Reference context	2.1	27
$\mathcal{U}_k$	The unknowns for the model with a context k	2.1	28
P	The reference pressure	2.1	28
θ	The temperature, present only in thermal context	2.1	28
S_p	The saturation for the phase $p \in \mathcal{P}$	2.1	28
$C_{p,c}$	The molar fraction for the component $c \in \mathcal{C}$ of the phase $p \in \mathcal{P}$	2.1	28
$n^n_{c,T}$	The number of moles for the component $c \in \mathcal{C}$	2.1	28
$\overline{\mathcal{C}_k}$	The set of absent components for the context k	2.1	28
P_p	The phase pressure	2.1	28

Symbol	Definition	Section	Page
P_{c_p}	The capillary pressure for the phase $p \in \mathcal{P}$	2.1	28
ϕ	The porosity	2.1	30
ζ_{r}	The rock molar density	2.1	30
e_{r}	The rock internal energy	2.1	30
$\mathbf{K}$	The absolute permeability	2.1	30
k_{r_p}	The relative permeability for the phase $p \in \mathcal{P}$	2.1	30
ζ_p	The molar density for the phase $p \in \mathcal{P}$	2.1	31
ρ_p	The mass density for the phase $p \in \mathcal{P}$	2.1	31
μ_p	The viscosity for the phase $p \in \mathcal{P}$	2.1	31
f_c	The fugacity of a component c	2.1	31
h_p	The enthalpy for the phase $p \in \mathcal{P}$	2.1	31
e_p	The internal energy for the phase $p \in \mathcal{P}$	2.1	31
$\mathcal{T}$	The computational mesh	2.2	39
T	A cell in the mesh $\mathcal{T}$	2.2	39
$\mathcal{F}$	The set of mesh faces σ	2.2	40
$\mathcal{F}^{\mathrm{i}}$	The set of internal faces	2.2	40
$\mathcal{F}^{\mathrm{b}}$	The set of boundary faces	2.2	40
N_{F}	The number of time steps	2.2	40
k_T	The context for the cell T	2.2	41
$\mathcal{P}_{k_T^n}$	The set of phases present in the cell T	2.2	41
P_T^n	The discrete reference pressure in the cell T at time t^n	2.2	41
θ_T^n	The discrete temperature in the cell T at time t^n	2.2	41
$S_{p,T}^n$	The discrete saturation for the phase $p \in \mathcal{P}_{k_T^n}$ in the cell T at the time t^n	2.2	41
$C_{p,c,T}^n$	The discrete molar fraction for $c \in \mathcal{C}$ of $p \in \mathcal{P}_T$ T at time t^n	2.2	41
n_c	The discrete number of moles for the component $c \in \mathcal{C}$	2.2	41
$\overline{\mathcal{C}_{k_T^n}}$	The set of absent components for the context k_T^n	2.2	41
$\mathbf{u}_T^n$	Vector of P, θ, S, and C unknowns in the cell T at time step n	2.2	41

Symbol	Definition	Section	Page
τ_T^W	The production index for well W	2.2	49
ν_p	The phase mobility term	2.2	51
$N_{\mathbf{u},T}$	The dimension of the vector of unknowns $\mathbf{u}_T^n$ in the cell	2.3	56
$\eta_{\mathrm{R},T,p}^{n,i,j}$	The residual estimator for the phase p, in the cell T, at the time step n, i-th nonlinear step, j-th linear step.	3.5	95
$\eta_{\mathrm{F},T,p}^{n,i,j}$	The flux estimator for the phase p, in the cell T, at the time step n, i-th nonlinear step, j-th linear step.	3.5	95
$\eta_{\mathrm{NC},T,p}^{n,i,j}$	The nonconformity estimator for the phase p, in the cell T, at the time step n, i-th nonlinear step, j-th linear step.	3.5	96
$(\vartheta_p)_{p\in\mathcal{P}}$	the molar fractions of the phase p in the fluid	A.2	144

List of Figures

List of Algorithms

List of Listings

Bibliography

[1] I. Aavatsmark, *An introduction to multipoint flux approximations for quadrilateral grids*, Comput. Geosci., 6 (2002), pp. 404–432.

[2] I. Aavatsmark, T. Barkve, O. Bøe, and T. Mannseth, *Discretization on non-orthogonal, curvilinear grids for multi-phase flow*, in Proc. of the 4th European Conf. on the Mathematics of Oil Recovery, vol. D, Røros, Norway, 1994.

[3] ——, *Discretization on non-orthogonal, quadrilateral grids for inhomogeneous, anisotropic media*, J. Comput. Phys., 127 (1996), pp. 2–14.

[4] ——, *Discretization on unstructured grids for inhomogeneous, anisotropic media. I. Derivation of the methods*, SIAM J. Sci. Comput., 19 (1998), pp. 1700–1716 (electronic).

[5] I. Aavatsmark, T. Barkve, Ø. Bøe, and T. Mannseth, *Discretization on unstructured grids for inhomogeneous, anisotropic media. II. Discussion and numerical results*, SIAM J. Sci. Comput., 19 (1998), pp. 1717–1736 (electronic).

[6] I. Aavatsmark, G. T. Eigestad, R. A. Klausen, M. F. Wheeler, and I. Yotov, *Convergence of a symmetric MPFA method on quadrilateral grids*, Comput. Geosci., 11 (2007), pp. 333–345.

[7] L. Agélas, D. A. Di Pietro, and J. Droniou, *The G method for heterogeneous anisotropic diffusion on general meshes*, M2AN Math. Model. Numer. Anal., 44 (2010), pp. 597–625.

[8] L. Agélas, D. A. Di Pietro, and R. Masson, *A symmetric and coercive finite volume scheme for multiphase porous media flow problems with applications in the oil industry*, Finite volumes for complex applications V, (2008), pp. 35–51.

[9] M. Ainsworth and I. Babuška, *Reliable and robust a posteriori error estimating for singularly perturbed reaction-diffusion problems*, SIAM J. Numer. Anal., 36 (1999), pp. 331–353.

[10] B. Amaziane, M. Jurak, and A. Žgaljić Keko, *An existence result for a coupled system modeling a fully equivalent global pressure formulation for immiscible compressible two-phase flow in porous media*, J. Differential Equations, 250 (2011), pp. 1685–1718.

[11] C. Amrouche and V. Girault, *Decomposition of vector spaces and applications to the Stokes problem in arbitrary dimensions*, Czech. Math. Journal, 44 (1994), pp. 109–140.

[12] S. N. Antontsev, A. V. Kazhikhov, and V. N. Monakhov, *Boundary value problems in mechanics of nonhomogeneous fluids*, vol. 22 of Studies in Mathematics and its Applications, North-Holland Publishing Co., Amsterdam, 1990. Translated from the Russian.

[13] T. Arbogast, *The existence of weak solutions to single porosity and simple dual-porosity models of two-phase incompressible flow*, Nonlinear Anal., 19 (1992), pp. 1009–1031.

[14] M. Arioli, D. Loghin, and A. J. Wathen, *Stopping criteria for iterations in finite element methods*, Numer. Math., 99 (2005), pp. 381–410.

[15] K. Aziz and A. Settari, *Petroleum reservoir simulation*, Applied Science Publishers, London, UK, 1979.

[16] I. Babuška and W. C. Rheinboldt, *Error estimates for adaptive finite element computations*, SIAM J. Numer. Anal., 15 (1978), pp. 736–754.

[17] R. Becker, C. Johnson, and R. Rannacher, *Adaptive error control for multigrid finite element methods*, Computing, 55 (1995), pp. 271–288.

[18] M. Braack and A. Ern, *A posteriori control of modeling errors and discretization errors*, Multiscale Model. Simul., 1 (2003), pp. 221–238 (electronic).

[19] R. M. Butler, *Thermal recovery of oil and bitumen*, Prendice–Hall, (1997).

[20] C. Cancès, *Finite volume scheme for two-phase flows in heterogeneous porous media involving capillary pressure discontinuities*, M2AN Math. Model. Numer. Anal., 43 (2009), pp. 973–1001.

[21] C. Cancès, T. Gallouët, and A. Porretta, *Two-phase flows involving capillary barriers in heterogeneous porous media*, Interfaces Free Bound., 11 (2009), pp. 239–258.

[22] C. Cancès, I. S. Pop, and M. Vohralík, *An a posteriori error estimate for vertex-centered finite volume discretizations of immiscible incompressible two-phase flow*, Math. Comp., (2013). Accepted for publication.

[23] H. Cao, *Developement of techniques for general purpose simulators*, Ph.D. thesis, Standford University, 2002.

[24] C. Carstensen and J. Hu, *A unifying theory of a posteriori error control for nonconforming finite element methods*, Numer. Math., 107 (2007), pp. 473–502.

[25] C. Carstensen, J. Hu, and A. Orlando, *Framework for the a posteriori error analysis of nonconforming finite elements*, SIAM J. Numer. Anal., 45 (2007), pp. 68–82 (electronic).

[26] A. Chaillou and M. Suri, *A posteriori estimation of the linearization error for strongly monotone nonlinear operators*, J. Comput. Appl. Math., 205 (2007), pp. 72–87.

[27] A. L. Chaillou and M. Suri, *Computable error estimators for the approximation of nonlinear problems by linearized models*, Comput. Methods Appl. Mech. Engrg., 196 (2006), pp. 210–224.

[28] G. Chavent and J. Jaffré, *Mathematical Models and Finite Elements for Reservoir Simulation*, vol. 17 of Studies in Mathematics and its Applications, North-Holland, Amsterdam, 1986.

[29] Z. Chen, *Degenerate two-phase incompressible flow. I. Existence, uniqueness and regularity of a weak solution*, J. Differential Equations, 171 (2001), pp. 203–232.

[30] ——, *Degenerate two-phase incompressible flow. II. Regularity, stability and stabilization*, J. Differential Equations, 186 (2002), pp. 345–376.

[31] ——, *Numerical analysis for two-phase flow in porous media*, Comput. Methods Appl. Math., 3 (2003), pp. 59–75 (electronic). Dedicated to Raytcho Lazarov.

[32] Z. Chen and R. E. Ewing, *Degenerate two-phase incompressible flow. IV. Local refinement and domain decomposition*, J. Sci. Comput., 18 (2003), pp. 329–360.

[33] Z. Chen, G. Huan, and Y. Ma, *Computational methods for multiphase flows in porous media*, Computational Science & Engineering, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2006.

[34] P. G. Ciarlet, *The Finite Element Method for Elliptic Problems*, vol. 4 of Studies in Mathematics and its Applications, North-Holland, Amsterdam, 1978.

[35] K. H. Coats, *Implicit compositional simulation of single-porosity and dual-porosity reservoirs*, SPE Symposium on Reservoir Simulation, (1989). paper SPE 18427-MS.

[36] K. H. Coats, L. Thomas, and R. Pierson, *Implicit compositional simulation of single-porosity and dual-porosity reservoirs*, SPE Symposium on Reservoir Simulation, (1995).

[37] D. A. Di Pietro, *Cell centered Galerkin methods*, C. R. Math. Acad. Sci. Paris, 348 (2010), pp. 31–34.

[38] D. A. Di Pietro, *Méthodes non conformes pour des équations aux dérivées partielles avec diffusion.* Habilitation thesis. Available at `http://tel.archives-ouvertes.fr/tel-00550230/fr/`, 2010.

[39] D. A. Di Pietro, *A compact cell-centered Galerkin method with subgrid stabilization*, C. R. Math. Acad. Sci. Paris, 349 (2011), pp. 93–98.

[40] D. A. Di Pietro and A. Ern, *Mathematical aspects of discontinuous Galerkin methods*, vol. 69 of Mathématiques & Applications (Berlin) [Mathematics & Applications], Springer, Heidelberg, 2012.

[41] D. A. Di Pietro, M. Vohralík, and C. Widmer, *An a posteriori error estimator for a finite volume discretization of the two-phase flow*, Finite volumes for complex applications. VI. Problems & perspectives, 4 (2011), pp. 341–349.

[42] J. J. Douglas, R. E. Ewing, and M. F. Wheeler, *The approximation of the pressure by a mixed method in the simulation of miscible displacement*, ESAIM, 17 (1983), pp. 17–33.

[43] J. Droniou and R. Eymard, *A mixed finite volume scheme for anisotropic diffusion problems on any grid*, Numer. Math., 105 (2006), pp. 35–71.

[44] M. G. Edwards and C. F. Rogers, *A flux continuous scheme for the full tensor pressure equation*, in The 4th European Conference on the Mathematics of Oil Recovery, vol. D, Røros, Norway, 1994.

[45] ——, *Finite volume discretization with imposed flux continuity for the general tensor pressure equation*, Comput. Geosci., 2 (1998), pp. 259–290.

[46] L. El Alaoui, A. Ern, and M. Vohralík, *Guaranteed and robust a posteriori error estimates and balancing discretization and linearization errors for monotone nonlinear problems*, Comput. Methods Appl. Mech. Engrg., 200 (2011), pp. 2782–2795.

[47] A. Ern and M. Vohralík, *A posteriori error estimation based on potential and flux reconstruction for the heat equation*, SIAM J. Numer. Anal., 48 (2010), pp. 198–223.

[48] ——, *Adaptive inexact Newton methods with a posteriori stopping criteria for nonlinear diffusion PDEs*, SIAM J. Sci. Comput., (2013). DOI 10.1137/120896918.

[49] R. Eymard, T. Gallouët, and R. Herbin, *Finite volume methods*, in Handbook of Numerical Analysis, Vol. VII, North-Holland, Amsterdam, 2000, pp. 713–1020.

[50] ——, *Discretization of heterogeneous and anisotropic diffusion problems on general non-conforming meshes SUSHI: a scheme using stabilization and hybrid interfaces*, IMA J. Numer. Anal., 30 (2010), pp. 1009–1043.

[51] V. Girault and P.-A. Raviart, *Finite element methods for Navier-Stokes equations*, vol. 5 of Springer Series in Computational Mathematics, Springer-Verlag, Berlin, 1986. Theory and algorithms.

[52] C. Guichard, *Schémas volumes finis sur maillages généraux en milieux hétérogènes anisotropes pour les écoulements polyphasiques en milieux poreux*, Ph.D. thesis, Université Paris-Est, 2011.

[53] P. Jiránek, Z. Strakoš, and M. Vohralík, *A posteriori error estimates including algebraic error and stopping criteria for iterative solvers*, SIAM J. Sci. Comput., 32 (2010), pp. 1567–1590.

[54] D. Kröner and S. Luckhaus, *Flow of oil and water in a porous medium*, J. Differential Equations, 55 (1984), pp. 276–288.

[55] P. Ladevèze, *Comparaison de modèles de milieux continus*, Ph.D. thesis, Université Pierre et Marie Curie (Paris 6), 1975.

[56] M. Mamaghani, *Suivi de fronts par des méthodes de raffinement de maillage adaptatif et application à la simulation du procédé de récupération Steam Assisted Gravity Drainage*, Ph.D. thesis, Université Blaise Pascal, 2010.

[57] M. Mamaghani, G. Enchéry, and C. Chainais-Hillairet, *Development of a refinement criterion for adaptive mesh refinement in steam-assisted gravity drainage simulation*, Computational Geosciences, 15 (2011), pp. 17–34.

[58] A. Michel, *A finite volume scheme for two-phase immiscible flow in porous media*, SIAM J. Numer. Anal., 41 (2003), pp. 1301–1317 (electronic).

[59] J. Nečas, *Equations aux dérivées partielles*, Presses de l'Université de Montréal, Montréal, 1966.

[60] L. E. Payne and H. F. Weinberger, *An optimal Poincaré inequality for convex domains*, Arch. Rational Mech. Anal., 5 (1960), pp. 286–292 (1960).

[61] D. W. Peaceman, *Interpretation of well-block pressures in numerical reservoir simulation*, SPE Journal, 18 (1978), pp. 183–194.

[62] J. Pousin and J. Rappaz, *Consistency, stability, a priori and a posteriori errors for petrov-galerkin methods applied to nonlinear problems*, Numer. Math., 69 (1994), pp. 213–231.

[63] T. F. Russell and M. F. Wheeler, *Finite element and finite difference methods for continuous flows in porous media*, in The Mathematics of Reservoir Simulation, SIAM, Philadelphia, 1983, pp. 35–106.

[64] M. Saad and H. Zhang, *Front tracking for two-phase flow in reservoir simulation by adaptive mesh*, Numer. Methods Partial Differential Equations, 13 (1997), pp. 673–697.

[65] L. Schwartz, *Méthodes Mathématiques pour les Sciences Physiques*, Enseignement des sciences, Hermann, Paris, 1965.

[66] R. Temam, *On the Theory and Numerical Analysis of the Navier–Stokes Equations*, North-Holland, Amsterdam, 1977.

[67] J. A. Trangenstein and J. B. Bell, *Mathematical structure of the black-oil model for petroleum reservoir simulation*, SIAM Journal on Applied Mathematics, 49 (1989), pp. 749–783.

[68] P. S. Vassilevski, *Multilevel Block Factorization Preconditioners*, Springer, New York, 2008.

[69] R. Verfürth, *A posteriori error estimators for the Stokes equations*, Numer. Math., 55 (1989), pp. 309–325.

[70] ——, *Robust a posteriori error estimates for nonstationary convection-diffusion equations*, SIAM J. Numer. Anal., 43 (2005), pp. 1783–1802 (electronic).

[71] M. Vohralík, *Residual flux-based a posteriori error estimates for finite volume and related locally conservative methods*, Numer. Math., 111 (2008), pp. 121–158.

[72] ——, *Estimations d'erreur a posteriori, critères d'arrêt et implémentations peu couteuses.* Habilitation thesis, 2010.

[73] M. Vohralík and M. F. Wheeler, *A posteriori error estimates, stopping criteria, and adaptivity for two-phase flows*, Comput. Geosci., (2013).

[74] J. R. Wallis, R. P. Kendall, T. E. Little, and J. S. Nolen, *Constrained residual acceleration of conjugate residual methods*, SPE Symposium on Reservoir Simulation, (1985). paper SPE 13536-MS.