Message Passing Interface parallelization of a multi-block structured numerical solver. Application to the numerical simulation of various typical Electro-Hydro-Dynamic flows

Umesh Kumar Seth

THESE

Pour l'obtention du grade de

DOCTEUR DE L'UNIVERSITÉ DE POITIERS

UFR des sciences fondamentales et appliquées

Pôle poitevin de recherche pour l'ingénieur en mécanique, matériaux et énergétique - PPRIMME

(Diplôme National - Arrêté du 25 mai 2016)

École doctorale : Sciences et ingénierie des matériaux, mécanique, énergétique - SIMME

Secteur de recherche : Mécanique des milieux fluides

Présentée par

Umesh Kumar SETH

Message Passing Interface parallelization of a multi-block structured numerical solver.

Application to the numerical simulation of various typical Electro-hydro-dynamic flows.

Directeur de Thèse : Philippe TRAORÉ

Co-Directeurs : Eric MOREAU, Pédro VAZQUEZ

Soutenue le 29 mars 2019

Jury

Rapporteur	Sakir AMIROUDINE	Professor, University of Bordeaux, France
Rapporteur	Sylvain LAIZET	Senior Lecturer, Imperial College London, England
Membre	Azeddine KOURTA	Professor, University of Orléans, France
Membre	Philippe TRAORÉ	Associate Professor, University of Poitiers, France
Membre	Eric MOREAU	Professor, University of Poitiers, France
Membre	Pédro VAZQUEZ	Associate Professor, University of Seville, Spain

ACKNOWELEDGMENTS

I express my sincere gratitude towards my thesis directors Associate Prof. Philippe Traoré, Prof. Eric Moreau and Associate Prof. Pédro A. Vazquez for providing me this opportunity to work with them, and their valuable guidance, support and supervision throughout the duration of this work.

The contributions of my office colleagues Philippe Parnaudeau, Alexandre Poux and Francisco J. Duran-Olivencia have been crucial in accomplishing the overall work. Especially, their support with the daunting tasks of high-level scientific programming was invaluable. I also wish to thank Pierre-François and Francis for always extending their support at their earliest.

The cordial atmosphere of the office made it possible for me to sustain in a foreign culture, and the demanding researcher life. I am grateful to all the people who work in the H2 building who were always friendly, patient and encouraging regarding my French language competences. My friends Pierre, Thomas, Gwenael, Yann, Ugo, Etienne, Emmanuel, Sachin, Arthur, Clement, Ayyoub and others were always there with me when I needed them.

I extend my gratitude towards the French ministry of Education who provided the funding for this work.

Thanks to all the jury members for their interest in this research work.

I also thank my family members who have always been there with me in all ups and downs of life.

Abstract

Several intricately coupled applications of modern industries fall under the multi-disciplinary domain of Electrohydrodynamics (EHD), where the interactions among charged and neutral particles are studied in context of both fluid dynamics and electrostatics together. The charge particles in fluids are generated with various physical mechanisms, and they move under the influence of external electric field and the fluid velocity. Generally, with sufficient electric force magnitudes, momentum transfer occurs from the charged species to the neutral particles also. This coupled system is solved with the Maxwell equations, charge transport equations and Navier-Stokes equations simulated sequentially in a common time loop. The charge transport is solved considering convection, diffusion, source terms and other relevant mechanisms for species. Then, the bulk fluid motion is simulated considering the induced electric force as a source term in the Navier-Stokes equations, thus, coupling the electrostatic system with the fluid. In this thesis, we numerically investigated some EHD phenomena like unipolar injection, conduction phenomenon in weakly conducting liquids and flow control with dielectric barrier discharge (DBD) plasma actuators.

Solving such complex physical systems numerically requires high-end computing resources and parallel CFD solvers, as these large EHD models are mathematically stiff and highly time consuming due to the range of time and length scales involved. This thesis contributes towards advancing the capability of numerical simulations carried out within the EFD group at Institut Pprime by developing a high-performance parallel solver with advanced EHD models. Being the most popular and specific technology, developed for the distributed memory platforms, Message Passing Interface (MPI) was used to parallelize our multi-block structured EHD solver. In the first part the parallelization of our numerical EHD solver with advanced MPI protocols such as Cartesian topology and Inter-Communicators is undertaken. In particular a specific strategy has been designed and detailed to account for the multi-block structured grids feature of the code. The parallel code has been fully validated through several benchmarks, and scalability tests carried out on up to 1200 cores on our local cluster showed excellent parallel speed-ups with our approach. A trustworthy database containing all these validation tests carried out on multiple cores is provided to assist in future developments.

The second part of this thesis deals with the numerical simulations of several typical EHD flows. We have examined three-dimensional electroconvection induced by unipolar injection between two planar-parallel electrodes. Unsteady hexagonal cells were observed in our study. 3D flow phenomenon with electro-convective plumes was also studied in the blade-plane electrode configuration considering both autonomous and non-autonomous injection laws. Conduction mechanism based on the dissociation of neutral molecules of a weakly conductive liquid has been successfully simulated. Our results have been validated with some numerical computations undertaken with the commercial code Comsol. Physical implications of Robin boundary condition and Onsager effect on the charge species were highlighted in electro-conduction in a rectangular channel. Finally, flow control using Dielectric Barrier Discharge plasma actuator has been simulated using the Suzen-Huang model. Impacts of dielectric thickness, gap between the electrodes, frequency and waveform of applied voltage etc. were investigated in terms of their effect on the induced maximum ionic wind velocity and average

body force. Flow control simulations with backward facing step showed that a laminar flow separation could be drastically controlled by placing the actuator at the tip of the step with both electrodes perpendicular to each other.

Keywords: *MPI, Cartesian topology, Inter-communicators, Electroconvection, Unipolar injection, conduction phenomenon, Plasma discharge, Suzen-Huang Model*

Résumé long en français

Chapitre 1.

Introduction

Au XXIe siècle, les progrès dans tous les domaines de la science et du génie dépendent de plus en plus des progrès de l'informatique. La capacité des ordinateurs modernes d'effectuer un grand nombre de calculs mathématiques avec une rapidité inimaginable est au cœur de cette dépendance. La disponibilité du matériel et des logiciels pertinents est la clé de cette impressionnante capacité des ordinateurs. La loi de Moore a guidé l'industrie des semi-conducteurs au cours des cinquantes dernières années pour faire progresser et anticiper l'avenir du matériel informatique ; selon cette loi, le nombre de transistors sur circuits intégrés devait doubler chaque année depuis 1965, et après 1975, elle a été révisée pour doubler tous les deux ans. Cela a conduit à une croissance rapide dans les technologies de matériel informatique qui se poursuit encore. D'autre part, plusieurs modèles de programmation ont été proposés pour concevoir efficacement les programmes logiciels afin de bénéficier des capacités matérielles toujours croissantes.

Il est évident que pour vraiment profiter des progrès en matière de matériel et de logiciels, les deux technologies doivent progresser de façon cohérente l'une par rapport à l'autre. En fait, comme l'industrie du matériel informatique a pris de l'avance sur le domaine du logiciel, les innovations logicielles doivent suivre les architectures matérielles déjà disponibles. Les progiciels existants et les codes scientifiques existants doivent être modernisés avec les architectures de processeurs en constante évolution pour permettre aux utilisateurs de bénéficier de l'omniprésence des processeurs multicœurs. Le milieu de la recherche scientifique et technique est l'un des plus grands consommateurs de ces technologies de calcul haute performance (HPC) en pleine croissance. Et il y a des organisations qui montrent la voie à suivre à la communauté scientifique pour la tenir à jour avec ces technologies HPC qui évoluent rapidement. Par exemple, le Partnership for Advanced Computing in Europe (PRACE) est une organisation qui vise à faciliter les découvertes scientifiques à fort impact et la recherche en ingénierie dans toutes les disciplines en fournissant aux communautés européennes membres les ressources informatiques et de gestion des données les plus avancées disponibles en Europe.

De nombreux problèmes importants de la science et de l'ingénierie ne peuvent être résolus sans cette technologie moderne de calcul haute performance. Les prévisions climatiques ou météorologiques, les turbulences aux plus petites échelles, les études aérodynamiques du corps entier des véhicules aérospatiaux, etc. sont quelques problèmes où les échelles de longueur sont très importantes, d'autre part, la physique de la décharge du plasma, les collisions atomiques, etc. sont des problèmes où les échelles de temps sont à contrario extrêmement petites. Ces deux types de problèmes exigent des capacités modernes de calcul haute performance pour bien comprendre les phénomènes sous-jacents.

Le contexte de cette thèse est basé sur un solveur numérique développé au sein du groupe Electro-Fluido-Dynamique de l'Institut Pprime spécifiquement dédié à la simulation numérique des écoulements Electro-Hydro-dynamique. Ce code de calcul : Oracle3D, avec lequel nous souhaitons étudier l'interaction électrostatique et hydrodynamique est un solver des équations de Navier-Stokes incompressibles sur des maillages structurés multi-blocs. Il est basé sur la méthode des volume finis avec des schémas en temps et en espace du second ordre. Il utilise l'algorithme SIMPLE pour le couplage de la vitesse et de la pression. Les flux convectifs pour les quantités scalaires peuvent être traités avec des schémas TVD (Total Variation Diminishing) avec plusieurs limiteurs de flux disponibles. Le schéma à correction différée améliorée (IDC) est utilisé pour traiter les flux diffusifs pour les maillages fortement distordus. Ces caractéristiques sont expliquées en détail dans les chapitres suivants. Le code est principalement développé pour simuler divers problèmes électro-hydrodynamiques.

Il existe plusieurs situations réelles où certains types d'espèces ioniques interagissent avec les molécules neutres de fluide environnantes et il se produit un échange d'énergie, d'impulsion, de potentiel électrique, etc. entre les ions et les particules neutres, qui nécessite une étude détaillée. De telles interactions d'espèces ioniques avec le fluide neutre sont considérées sous la discipline de l'Electro-Hydro-Dynamique (EHD). L'objectif global de cette thèse consiste à moderniser le code Oracle3D pour s'attaquer à des simulations EHD complexes et de grande envergure sur des systèmes HPC avancés et à étudier numériquement les problèmes EHD dans des configurations tridimensionnelles.

Les applications de l'industrie moderne, qui sont intimement liées, regroupent les branches de l'hydrodynamique, de l'électrostatique, de l'électrochimie, etc. dans le cadre de la recherche pluridisciplinaire EHD. Le groupe Electro-Fluido-Dynamique (EFD) de l'Institut Pprime en France participe activement à l'élaboration de nouvelles technologies du domaine EHD et ouvre la voie à la résolution de nombreux problèmes industriels tels que le détachement d'écoulements aérodynamiques. Par son appartenance à l'Institut Pprime en collaboration avec l'Université de Poitiers le groupe EFD dispose d'installations expérimentales et HPC de pointe. Des efforts ont été faits, au cours de ce travail de thèse, pour faire progresser significativement la capacité de recherche numérique du groupe en concevant et mettant en œuvre la stratégie parallèle pour le code Oracle3D et en le validant rigoureusement. Un travail numérique détaillé et une méthodologie de calcul ont été fournis dans cette thèse pour les utilisateurs actuels et futurs de ce solveur EHD.

Ce travail ne se concentre pas en profondeur sur un seul problème physique ; cependant, il fournit une gamme d'applications EHD incluant l'injection unipolaire, l'électro-conduction, le contrôle des écoulements par décharges plasma. Des tests de validation rigoureux, dans plusieurs configurations ont été menés pour construire une large base de données pour le nouveau code. Certains problèmes classiques d'EHD ont été réexaminés et une bibliographie des travaux antérieurs pertinents a été fourni dans les chapitres et sections correspondants. Comme le code a été parallélisé à partir de l'ancienne version Fortran 77, les détails de l'implémentation MPI sont fournis dans cette thèse pour faciliter les avancées et développements futurs.

Le solveur EHD Oracle3D, est en cours de développement depuis plus d'une décennie maintenant, et des fonctionnalités avancées sont ajoutées régulièrement pour étendre sa portée à de nouveaux problèmes EHD plus complexes. Au cours de cette thèse, le solveur Oracle3D a

été mis à jour avec des fonctionnalités Fortran modernes et, plus important encore, il a été parallélisé avec l'interface MPI (Message Passing Interface) pour être exécuté sur des clusters CPU à mémoire distribuée. La parallélisation d'un code scientifique présente de nombreux défis en termes de programmation. Ainsi, la tâche de parallélisation du solveur EHD complet a été divisée en sous-tâches qui comprenaient la préparation de codes scalaires et parallèles individuels pour des modèles physiques simples tels que le solveur de Poisson, le solveur Navier-Stokes, un solveur de transport scalaire etc.

Pour simplifier l'étape de pré-processing, un outil a été développé afin de lire les données issues du mailleur BlockMesh et de les convertir dans le format approprié pour le solveur Oracle3D. Des fonctions MPI avancées telles que la topologie cartésienne, la topologie de groupes, les inter-communicateurs, etc. ont été implémentées dans le code pour préparer une stratégie évolutive de transmission de messages pour les calculs parallèles dans un contexte multi-bloc. La première partie de la thèse porte sur la méthodologie utilisée pour paralléliser les différents solveurs et leur validation avec les résultats existants. Voici un bref résumé des chapitres :

- Le chapitre 2 mentionne brièvement la méthode des volumes finis (MVF) comme étant l'approche numérique utilisée pour la discrétisation des équations utilisées dans Oracle3D. Au lieu d'une équation générale de transport, les équations spécifiques utilisées dans le code sont prises comme exemples de discrétisation. La nécessité et les modalités de mise en œuvre des schémas TVD sont détaillées pour les utilisateurs du code. De nouvelles conditions limites ont été introduites dans le code, qui sont expliquées à l'aide d'une approche de discrétisation pertinente.

- Le chapitre 3 détaille la méthodologie utilisée pour la parallélisation du code avec les fonctions MPI avancées. Un aperçu général vers des modèles de programmation parallèles est fourni dans la 1ère section. La section 2 traite des caractéristiques de la topologie cartésienne de MPI qui ont été initialement utilisées pour paralléliser les calculs dans des blocs individuels. La 3ème partie présente l'ensemble de la stratégie utilisée pour paralléliser les géométries basées sur des maillages multi-blocs, avec tous les détails de mise en œuvre pour les utilisateurs. Enfin, certains tests de scalabilité sont fournis avec des explications pour juger de l'efficacité parallèle du nouveau code.

- Le chapitre 4 présente tous les cas de test effectués pour valider les solveurs parallèles individuels : Solveur de Poisson, solveur de Navier-Stokes, solveur de transport scalaire. La plupart des nouvelles fonctionnalités ajoutées ont été validées avec un nombre différent de cœurs pour vérifier l'approche de transmission des messages dans les solveurs parallèles.

Dans la deuxième partie, nous fournissons les études EHD réalisées avec Oracle3D au cours de cette thèse.

- Le chapitre 5 traite de l'injection unipolaire EHD. Le problème de l'électro-convection est défini, et une brève revue de la littérature est donnée pour commencer. Quelques premières études 2D sont présentées pour valider le modèle d'injection unipolaire du code. La formation de motifs tridimensionnels de cellules convectives dans une configuration d'électrodes à plaques parallèles est étudiée en détail. Ensuite, les panaches d'injection 3D sont étudiés dans le cas d'une géométrie d'électrode lame-plan avec différentes lois d'injection.

- Le chapitre 6 présente plusieurs calculs dans le cadre du phénomène d'électro-conduction. Certains tests de validation ont été effectués pour comparer les résultats avec ceux obtenus par le code industriel Comsol. Un écoulement généré par le phénomène de conduction dans un canal 3D a été simulé pour la 1ère fois. Dans la deuxième section, nous donnons un aperçu de la configuration d'écoulement observée dans un cas de conduction avec la géométrie lame-plan.

- Le dernier chapitre traite de la simulation de décharge plasma. Nous avons utilisé, dans cette première approche, le modèle Suzen-Huang (SH) qui est décrit. L'impact de la longueur de Debye est brièvement exploré dans le contexte du modèle SH. Une étude paramétrique portant sur les paramètres géométriques et électriques caractérisant les actionneurs DBD est fournie. Une étude basée sur des données expérimentales de la force électrique utilisée comme terme source dans les équations de Navier-Stokes que nous résolvons est réalisée. Enfin, une brève étude avec un contrôle d'écoulement laminaire sur une marche arrière est fournie.

Chapitre 2.
Méthode des volumes finis dans le contexte d'Oracle3D

La méthode des volumes finis (FVM) est l'une des approches mathématiques les plus populaires parmi d'autres, qui sont utilisées pour résoudre les problèmes de mécanique des milieux continus en discrétisant les équations aux dérivées partielles correspondantes dans le temps et l'espace. La discrétisation spatiale d'un problème se réfère à la division du domaine spatial d'un problème en entités géométriques beaucoup plus petites comme : les cellules de calcul, les faces et les noeuds. Ensuite, le problème physique dans l'ensemble du domaine spatial est décrit de manière combinée par les relations algébriques définies sur ces cellules et nœuds de calcul individuels. Les équations algébriques pour les cellules de calcul individuelles sont obtenues en intégrant les équations aux dérivées partielles avec FVM sur chaque cellule discrète.

Lorsque le problème est de nature instationnaire, une discrétisation en temps est également nécessaire, ce qui s'effectue en subdivisant le temps de la simulation en sous-pas de temps beaucoup plus petits. L'évolution du problème physique avec des pas de temps plus petits fournit une solution instationnaire complète. Un organigramme complet du processus général de discrétisation, tel qu'il est habituellement suivi dans les analyses numériques est présenté à la figure 2.1. Dans ce chapitre, nous parlerons principalement des stratégies de FVM telles qu'utilisées dans notre solveur, Oracle3D. Nous avons essayé d'expliquer les différentes discrétisations nouvellement mises en œuvre et les conditions aux limites avec les problèmes réels rencontrés dans les modèles physiques qui sont disponibles dans le code. Il s'agit de faciliter la compréhension du code pour les futurs utilisateurs.

2.1 La méthode des volumes finis : un bref aperçu

Un grand nombre de méthodes et de schémas sont disponibles dans le cadre de l'approche des volumes finis, selon la nature du problème physique (diffusion, convection, etc.), l'ordre de précision requis, la structure du maillage, etc. La nature conservatrice inhérente de la méthode des volumes finis est sa caractéristique dominante qui la place en tête de toutes les autres techniques numériques lorsqu'il est question de dynamique des fluides numérique (CFD). Lorsqu'il s'agit de flux de quantités conservatrices sur les faces des cellules de calcul, il est imposé que le flux entrant dans un volume de contrôle soit identique au flux sortant du volume adjacent, ce qui rend la FVM strictement et fondamentalement conservatrice. En particulier, cette caractéristique est un avantage supplémentaire pour les problèmes de mécanique des fluides où nous devons satisfaire les lois de conservation de la masse, de la quantité de mouvement et de l'énergie à chaque pas de temps. Avec les progrès significatifs de la CFD, au cours des dernières décennies, la FVM a gagné beaucoup en popularité en étant capable de s'attaquer à toutes sortes de problèmes physiques complexes mais surtout de géométries complexes.

De la même façon que pour les autres approches de discrétisation telles qque la méthode des différences finies (FD) et la méthode des éléments finis (FEM), dans la méthode des volumes finis nous transformons aussi les équations aux dérivées partielles (EDP) en équations algébriques linéaires. Tous les phénomènes physiques qui nous concernent sont décrits par des EDP qui définissent distinctement la nature mathématique et physique du problème à l'étude. Par exemple, dans la CFD, les EDP les plus fréquemment rencontrées sont les équations de Navier-Stokes, qui sont définies par les lois de conservation de la masse et de la quantité de mouvement. Dans le processus de discrétisation des EDP, qui nécessite la transformation des intégrales de volume et de surface en équations algébriques discrètes, nous utilisons le théorème de la divergence (thorème de Green-Ostrogradski ou théorème de Gauss).

Le théorème de la divergence indique que le flux global d'un champ vectoriel ($\vec{u}$) au travers d'surface fermée (S) est égal au volume total de toutes les sources et puits sur la région confinée par cette surface, eq. (2.1). Ici, le volume total de toutes les sources et puits est défini par l'intégrale du volume de la divergence de ce champ vectoriel. Ainsi, avec ce théorème, nous convertissons habituellement les intégrales de volume en flux de surface, qui sont ensuite utilisés pour former les équations algébriques discrètes.

$$\int_V (\nabla . \vec{u})\ dV = \int_S \vec{u} . \vec{n}\ dS \tag{2.1}$$

Nous présentons ici, à titre d'exemple, l'équation de conservation d'une variable scalaire générale φ pour exprimer l'utilisation du théorème de la divergence en FVM. L'équation 2.2 montre les quatre termes présents dans une équation générale de conservation : le terme transitoire, le terme convectif, le terme de diffusion et le terme source. Ici, ρ est la masse volumique du fluide, $\vec{u}$ est le champ du vecteur vitesse, et Γ est le coefficient de diffusion de la variable φ. Nous gardons le traitement du terme transitoire pour plus tard et montrons ici la transformation de cette EDP (eq. 2.2) en flux de surface sur les volumes de contrôle. L'équation 2.3 représente la forme stationnaire de l'équation 2.2. Ces deux équations concernent l'ensemble du domaine.

$$\underbrace{\frac{\partial(\rho\varphi)}{\partial t}}_{\text{terme transitoire}} + \underbrace{\nabla . (\rho\varphi\vec{u})}_{\text{terme convectif}} = \underbrace{\nabla . (\Gamma^{\varphi}\nabla\varphi)}_{\text{terme diffusif}} + \underbrace{Q^{\varphi}}_{\text{terme source}} \tag{2.2}$$

$$\nabla . (\rho\varphi\vec{u}) = \nabla . (\Gamma^{\varphi}\nabla\varphi) + Q^{\varphi} \tag{2.3}$$

Nous intégrons l'équation 2.3 sur une cellule C ou volume de contrôle donné. L'équation 2.4 est la forme intégrale l'équation de conservation stationnaire sur un volume de contrôle. Nous utilisons maintenant l'équation 2.1 (théorème de la divergence) pour convertir les intégrales de volume des termes convectifs et diffusifs en intégrales de surface, comme le montre l'équation 2.5. Ici V_c est le volume de la cellule C et $\vec{S}$ S est le vecteur surface associé au volume de contrôle. L'équation 2.5 est habituellement appelée équation semi-discrétisée dans la FVM, car elle représente les contributions des cellules de volume fini individuelles[1].

$$\int_{V_c} \nabla . (\rho\varphi\vec{u})\, dV = \int_{V_c} \nabla . (\Gamma^{\varphi}\nabla\varphi)\, dV + \int_{V_c} Q^{\varphi}\ dV \tag{2.4}$$

$$\int_{\partial V_c} (\rho\varphi\vec{u}) . d\vec{S} = \int_{V_c} (\Gamma^{\varphi}\nabla\varphi) . d\vec{S} + \int_{V_c} Q^{\varphi}\ dV \tag{2.5}$$

Avec l'équation semi-discrétisée, nous devons obtenir les équations algébriques discrètes pour chaque cellule, qui seront la contribution des cellules individuelles en termes de diffusion, convection et source pour l'ensemble du problème. La diffusion, la convection et les sources/puits sont trois phénomènes de nature physique complètement différente. Ils sont traités séparément pour obtenir leur contribution, puis finalement combinés pour la solution globale. Nous avons discuté de l'approche mathématique pour obtenir les équations algébriques discrètes pour chaque terme dans ce chapitre.

Le schéma de discrétisation amont est le schéma le plus stable et il est inconditionnel ; cependant, il introduit un niveau élevé de diffusion numérique car il n'est précis qu'au 1er ordre[1,7]. Les schémas d'ordre supérieur tels que le schéma centré, le schéma QUICK sont plus précis, mais ils peuvent donner des oscillations intempestives (wiggles) lorsque le nombre de Peclet est élevé (>2). Ces oscillations numériques peuvent conduire à des valeurs physiquement irréalistes et rendre la solution instable. Pour remédier à cette caractéristique indésirable des schémas à haute résolution (HR) sont formulés. Les schémas HR sont formulés de manière à préserver la nature convective des schémas précédents tout en améliorant le critère d'une solution bornée.

Plusieurs schémas numériques ont été mis au point à ce jour et une revue de tous ces schémas est disponible dans[1,3]. Les schémas HR qui ont été développés dans notre cadre font partie de la classe des schémas convectifs de type TVD. Les schémas TVD sont spécialement développés pour contrer les oscillations parasites en ajoutant une diffusion artificielle ou une pondération vers la contribution en amont dans les équations discrétisées. L'implémentation des schémas TVD dans Oracle3D est détaillée dans ce chapitre.

La mise en œuvre correcte des conditions aux limites dans n'importe quelle résolution numérique est de la plus haute importance. Dans de nombreux problèmes physiques, les chercheurs ont souvent des points de vue différents sur les conditions aux limites numériques qui doivent représenter les conditions physiques. Parfois, les conditions aux limites numériques sont trop simplifiées ou approximatives afin d'éviter certaines difficultés et complexités numériques, tout en garantissant toutefois une certaine précision physique. Au cours de cette thèse, nous avons implémenté dans Oracel3D de nouvelles conditions aux limites qui étaient d'une importance physique énorme, et il a toujours été souligné que l'utilisateur devait faire très attention lorsqu'il s'agissait de discrétiser les équations aux limites.

Il est également très important que l'utilisateur comprenne à la fois la discrétisation générale de condition à la limite et sache comment elles sont réellement implémentées dans le code. Dans cette section, nous discutons de la discrétisation de certaines conditions limites (sections 2.6.1-2.6.6.3) et enfin nous fournissons une approche générale (section 2.6.4) pour implémenter tout type de conditions aux limites pour Oracle3D. On peut se référer à l'approche générale mentionnée à la section 2.6.4 avant de passer aux sections 2.6.1-2.6.3, pour un résumé rapide et une approche différente lors de l'implémentation des conditions limites.

Chapitre 3
Oracle3D parallèle avec MPI

Les programmes informatiques parallèles hautement évolutifs sont devenus des outils indispensables à l'avancement de la recherche numérique. Les chercheurs ont plus d'espoir que jamais auparavant pour s'attaquer à des problèmes d'ingénierie et scientifiques complexes et énormes en raison de la disponibilité des ressources informatiques nécessaires au cours des vingt dernières années. En ce qui concerne les ressources matérielles, les progrès semblent

largement supérieurs aux progrès dans le domaine des applications (logiciels). Il existe de nombreux codes hérités qui sont encore pertinents aujourd'hui, mais ils n'ont pas l'approche moderne pour utiliser efficacement les ressources matérielles disponibles. Parmi les problèmes de calcul énormes qui ne peuvent être résolus sans les technologies modernes de calcul haute performance (HPC), on peut citer : les prévisions météorologiques, l'analyse astrophysique, l'analyse tectonique des plaques, la modélisation de la turbulence, la physique des plasmas, etc. Il est inimaginable de résoudre ces problèmes sur un ordinateur scalaire mono-processeur et encore plus lorsque les calculs sont tri-dimensionnels. L'informatique parallèle est alors la seule façon de s'attaquer à la résolution de tels problèmes.

En général, le calcul parallèle consiste à résoudre simultanément des parties d'un problème sur des machines informatiques multi-cœurs. Un problème qui peut être divisé en plusieurs petites parties discrètes qui peuvent être résolues indépendamment, est un bon candidat pour le calcul parallèle. Les parties discrètes du problème sont résolues sur différents cœurs de calcul et une fois terminées, elles sont synchronisées pour fournir la solution à l'ensemble du problème. L'informatique parallèle offre plusieurs avantages aux utilisateurs : gain de temps et d'argent, résolution de problèmes complexes et volumineux, multitâches, etc. Les progrès du calcul haute performance ont fourni une autre façon d'appréhender la science grâce au calcul scientifique parallèlement aux branches plus classiques des sciences expérimentales et théoriques. Les informaticiens utilisent leurs méthodes de simulation lorsqu'elles sont plus avantageuses et plus réalisables que les approches classiques de la théorie et des expériences.

Les trois grands domaines du calcul parallèle sont le matériel, les algorithmes et les logiciels. Sur le plan matériel, l'ajout de plus en plus de cœurs et la mise en place d'un réseau d'intercommunication efficace entre les cœurs a accru la nature parallèle des machines informatiques. En termes algorithmiques, les scientifiques cherchent comment un problème peut être défini par des mécanismes physiques indépendants et comment il peut être résolu par un ensemble indépendant d'équations mathématiques. Cependant, un plus grand défi est posé par les logiciels inadéquats, qui ne sont pas en mesure de profiter pleinement des progrès réalisés en matière de matériel et d'algorithmiques. En termes de caractéristiques importantes, les codes modernes doivent être optimisés, portables et à l'épreuve du temps avec chaque évolution des technologies HPC. Comme le montre la Fig. 3.1, un code doit faire un usage optimal des propriétés matérielles telles que la conception du cache, les registres vectoriels, les noyaux multiples, etc. Il doit être développé avec les modèles standard de programmation parallèle tels que MPI, OpenMP, Offloading, etc.

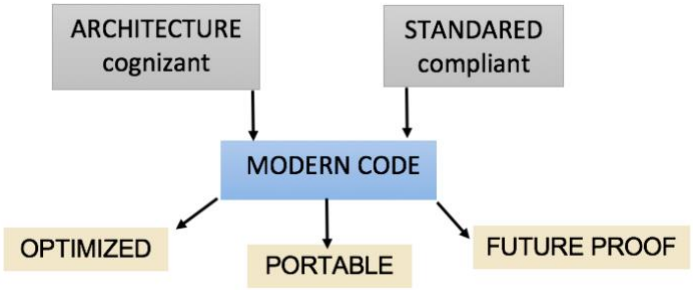


Fig. 3.1 Caractéristiques d'un code moderne

Ce chapitre traite de l'approche de parallélisation utilisée pour le code Electro-Hydro-Dynamique (EHD) :Oracle3D. Oracle3D est un solveur volumes finis pour maillages structurés par blocs. Il est parallélisé avec les protocoles de l'interface de transmission des messages (MPI) 3.1. Ce chapitre est principalement divisé en trois parties : 1) Modèles de programmation parallèle, 2) Parallélisation de la grille MPI et du maillage mono-bloc, et 3) Parallélisation sur maillages multi-blocs. Nous présentons brièvement quelques modèles de programmation parallèle et décrivons le modèle de transmission des messages dans la première partie. Dans la deuxième partie, nous décrivons en détail la méthodologie MPI utilisée pour paralléliser le code pour les grilles à blocs simples seulement.

Les fonctions de transmission de messages de la bibliothèque MPI qui sont pertinentes pour de telles approches sont fournies. Certains résultats de scalabilité sont fournis pour vérifier l'efficacité de l'approche. La troisième section traite de l'extension de l'approche pour les cas des maillages multi-blocs, où certaines caractéristiques plus avancées de MPI ont été utilisées. La mise en œuvre détaillée de cette stratégie basée sur la topologie cartésienne et les inter-communicateurs est proposée aux utilisateurs du code, et d'autres chercheurs qui travaillent avec des codes similaires et qui souhaitent paralléliser leur code en utilisant MPI pourraient également bénéficier de ce chapitre détaillé sur diverses fonctionnalités MPI.

Aperçu du présent chapitre :

1. Vue d'ensemble de la programmation parallèle
2. MPI et maillage mono-bloc
3. Extension MPI aux maillages multi-blocs

Dans la première section, nous avons discuté brièvement de quelques modèles de calcul parallèles pour fixer le contexte de l'approche que nous avons utilisée pour paralléliser notre code. Les modèles de calcul parallèles peuvent être classés de manière informelle en fonction de leur utilisation de la mémoire (partagée ou distribuée), du modèle de communication, des types d'opérations, etc.

La deuxième partie du chapitre traite des fonctionnalités de MPI en général, et telles qu'elles sont implémentées dans Oracle3D pour la mise en parallèle des grilles de blocs individuels en premier. Principalement, les caractéristiques topologiques cartésiennes du IPM sont discutées en détail, qui ont été utilisées pour optimiser l'efficacité des communications parallèles. Dans la troisième section, nous étendrons notre stratégie de topologie cartésienne aux maillages multi-blocs en utilisant des fonctions MPI plus avancées.

MPI a été développé pour combiner les meilleures caractéristiques de nombreux modèles de transmission de messages qui ont existé au fil des ans. Il s'agit d'une tentative d'organiser et d'améliorer les caractéristiques existantes des modèles de transmission de messages et de

préparer une norme qui reste portable à travers la gamme de matériel et de logiciels disponibles sur le marché. Comme définie par la norme, "MPI (Message-Passing Interface) est une spécification de bibliothèque de transmission de messages"[2,7]. MPI n'est pas un langage de programmation, c'est une bibliothèque de fonctions qui facilite le transfert de données pendant les communications parallèles. Ce protocole de communication est le modèle de transmission de messages le plus largement utilisé sur diverses architectures de mémoire distribuée à travers différents clusters de supercalculateurs. MPI est la première spécification qui permet d'écrire des bibliothèques parallèles réellement portables.

Maintenir la portabilité, l'efficacité et la fonctionnalité des programmes parallèles est l'objectif principal de MPI. Quelques caractéristiques avancées de MPI incluent la gestion dynamique des groupes de processus, des structures de processus orientées application, un grand nombre d'opérations collectives, etc. Les caractéristiques plus générales et fréquemment utilisées de MPI sont : les opérations point à point, les communicateurs, les opérations collectives, les groupes, etc. Nous avons discuté de ces caractéristiques dans ce chapitre.

Des fonctions MPI avancées de topologie cartésienne et d'inter-communicateurs ont été mises en œuvre pour préparer une stratégie d'échange de données hautement évolutive pour les maillages structurées à multi-blocs. Certains tests de scalabilité ont montré des scalabilités super-linéaires qui sont attribuées à des effets de cache favorables tout en augmentant le nombre de noyaux et grâce aux algorithmes avancés utilisés par les processeurs Intel modernes pour fonctionner dynamiquement à des fréquences supérieures à leurs fréquences de base. Pour le bénéfice des futurs utilisateurs de ce code, l'ensemble de la stratégie MPI a été décrite en détail dans la thèse, ce qui faciliterait également le développement futur du code.

Chapitre 4

Validation : Oracle3D parallèle

La stratégie MPI détaillée discutée dans le chapitre précédent a été implémentée dans Oracle3D au sein des différentes versions ou modules du solveur. Comme mentionné précédemment, le code a été ré-écrit avec une structure « module » du FORTRAN 90, pour le rendre plus organisé et orienté dans le cadre moderne Fortran. De la même manière, nous avons également préparé les versions individuelles pour les modules spécifiques comme : le solveur Navier-Stokes pur, le solveur de Poisson, le solveur de transport général, le solveur plasma basé Suzen-Huang et le modèle à 3 espèces, puis le solveur Oracle3D complet. Cet agencement du code en modules individuels plus petits a grandement facilité le transfert de la méthodologie MPI dans la version complète du code. De plus, chaque module plus petit nous a donné l'occasion de valider et de tester la performance de l'implémentation parallèle dans différents modèles mathématiques. Ce chapitre présente tous les tests de validation de la version finale des différents modules qui ont été parallélisé. Ainsi, ce chapitre fournit les toutes premières simulations avec le code

développé et permet de bâtir une certaine confiance pour les utilisateurs d'Orcale3D avec des cas de validation éprouvés.

La validation de tout nouveau développement dans le code est une étape cruciale avant les applications finales prévues. Ce développement a commencé avec la version de base Fortran 77 d'Oracle3D, qui a déjà été longuement validée et plusieurs études ont été publiées avec cette version[1-5]. Le développement a commencé avec la conversion de la version Fortran 77 du code vers la version Fortran 90. A chaque étape de ce processus de conversion, le nouveau développement a été vérifié par comparaison avec les résultats de la version précédente. Les étapes majeures de cette conversion incluent l'implémentation des fonctionnalités de Fortran 90 telles que : implicit none, l'allocation dynamique des tableaux, la vectorisation des boucles, l'utilisation des modules etc... Les versions finales Fortran 90 des différents solveurs ont été validées, cependant, nous ne présenterons ici que les résultats de validation avec les versions parallèles de ces solveurs.

4.1 Le solveur de Poisson parallèle

La méthodologie MPI, telle que détaillée précédemment, a d'abord été testée rigoureusement avec quelques échanges de données entières très simples aux deux types d'interfaces, dans plusieurs combinaisons de blocs et de sous-domaines. Après les essais préliminaires, la stratégie MPI a d'abord été mise en œuvre dans un solveur de Poisson. La célèbre équation de Poisson est une équation aux dérivées partielles de type elliptique. Nous utilisons ici la méthode des volumes finis pour discrétiser l'équation de Poisson dans l'espace. L'équation (4.1) est la forme générale de l'équation de Poisson :

$$\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}+\frac{\partial^2}{\partial z^2}\right)\varphi(x,y,z)=f(x,y,z) \tag{4.1}$$

$$\nabla^2\varphi(x,y,z)=f(x,y,z) \tag{4.2}$$

Le solveur parallèle de Poisson a également permis de vérifier la mise en œuvre du schéma de correction différée améliorée (IDC) avec les nouveaux développements du code. Le schéma IDC est une technique de discrétisation en volumes finis, spécialement développée dans notre groupe de l'Institut Pprime, pour la discrétisation des flux diffusifs sur des maillages complexes particulièrement distordus [1-2]. Des références publiées sont disponibles dans lesquelles le schéma IDC a été introduit et exploré sur différents maillages complexes. Les résultats disponibles dans les références ont été obtenus avec la version 2D de notre code Fortran 77. Ainsi, pour valider le nouveau solveur MPI Poisson, nous avons choisi la même équation de Poisson que celle utilisée par Traoré et al (2009) :

$$\Delta\varphi=2\pi^2\left(cos^2(\pi x)-sin^2(\pi x)\right)+2\pi^2\left(cos^2(\pi y)-sin^2(\pi y)\right) \tag{4.3}$$

$$\varphi = sin^2(\pi x) + sin^2(\pi y) \tag{4.4}$$

Des conditions aux limites périodiques pour les trois directions (X, Y et Z) ont été mises en œuvre dans le solveur de Poisson. Cette caractéristique est essentielle pour divers problèmes d'EHD où dans certains cas nous avons besoin de cette condition à la limite. Ici, une fonction spatialement périodique a été conçue pour tester l'implémentation périodique du solveur de Poisson.

4.2 Validation du solveur Navier-Stokes

Cette section est consacrée à la validation du solveur parallèle Navier-Stokes. Les caractéristiques MPI de la topologie et des inter-communicateurs ont été implémentées dans le solveur Navier-Stokes pur d'Oracle3D. C'est un solveur pour les écoulements incompressibles, dans lequel le couplage vitesse-pression est réalisé par l'algorithme SIMPLE standard de Patankar [11]. Le schéma de discrétisation centré est utilisé pour la discrétisation spatiale de tous les cas de validation. Le schéma de Gear est utilisé pour la discrétisation en temps. Les équations générales de continuité et de conservation de la quantité de mouvement pour les écoulements incompressibles résolues dans ce solveur prennent la forme :

$$\nabla.(\rho\vec{u}) = 0. \tag{4.5}$$

$$\frac{\partial(\rho\overrightarrow{u})}{\partial t} + \nabla.(\rho\vec{u}\,\vec{u}) = -\nabla p + \nabla.(\mu\,(\nabla\vec{u} + (\nabla\vec{u})^T) + \vec{f} \tag{4.6}$$

Le problème de référence standard de la cavité entraînée a été simulé pour valider ce solveur incompressible. Deux valeurs différentes (100 et 400) du nombre de Reynolds du débit ont été prises pour comparaison avec la solution de référence de Ghia et al [16]. Un autre problème standard et bien étudié pour la validation des solveurs Navier-Stokes est celui de l'écoulement derrière une marche descendante (Backward Facing Step). Plusieurs benchmarks et résultats expérimentaux sont disponibles avec différentes configurations du canal, selon la hauteur de la marche et la valeur du nombre de Reynolds de l'écoulement. Notre objectif est de valider notre implémentation MPI, en particulier les conditions limites d'entrée et de sortie à différents nombres de Reynolds, dans cette section.

Les conditions aux limites périodiques sont essentielles pour tout solveur Navier-Stokes. Nous fournissons ici les résultats obtenus sur deux cas test qui ont été réalisés pour valider l'implémentation de cette condition à limite périodique dans Oracle3D. Les écoulements classiques de Poiseuille et de Couette ont été utilisés pour la validation de l'implémentation des conditions aux limites périodiques dans notre code.

4.3 Parallélisation du solveur de transport

Dans le cadre d'Oracle3D, nous avons également développé un solveur de transport général. Ce solveur résout principalement une équation de transport convectif pour une variable scalaire (φ), comme la densité volumique de charge électrique ou le transports des ions dans l'écoulement. Le terme de diffusion des phénomènes de transport n'est pas inclus ici pour étudier exclusivement l'efficacité et la mise en œuvre des schémas convectifs. Les schémas

TVD (Total Variation Diminishing) sont mis en œuvre pour résoudre la discrétisation spatiale avec une précision supérieure au 2e ordre. Une attention particulière a été accordée à la mise en œuvre parallèle du schéma TVD aux nœuds d'interface et aux limites. La méthode des cellules fantômes est adoptée pour stocker les données des voisins aux interfaces, car dans les schémas TVD, les données de plus d'un nœud voisin dans chaque direction sont nécessaires en fonction de la direction du flux. De plus, les conditions aux limites périodiques ont été mises en œuvre et testé dans ce solveur de transport. L'équation générale de transport utilisée dans ce solveur est :

$$\frac{\partial(\rho\varphi)}{\partial t} + \nabla.(\rho\vec{u}\varphi) = 0\,. \tag{4.7}$$

Des simulations ont été effectuées pour valider la mise en œuvre parallèle du schéma TVD et des conditions aux limites périodiques pour le solveur de transport. Des calculs 2D et 3D ont été réalisés avec de multiples processus MPI. Dans le cas 2D, un domaine carré avec des limites périodiques dans les directions X et Z a été considéré. Le même calcul a été effectué cette fois dans un domaine 3D, où nous avons une périodicité dans les 3 directions. Le maillage pour ce cas était de 200 X 200 X 200 et 32 processus MPI ont été utilisés. La rotation d'un scalaire passif permet de tester l'efficacité des schémas convectifs en présentant plus de cisaillement dans l'écoulement. Également ce cas est intéressant pour tester l'efficacité de l'implémentation parallèle, car il présente un degré plus élevé de complexité dans la communication de données au niveau des interfaces du domaine MPI.

Chapitre 5

EHD Injection Unipolaire

Oracle3D est principalement un solveur électrohydrodynamique (EHD) qui a été utilisé pour réaliser plusieurs études dont l'injection unipolaire, la conduction EHD, l'électro-séparation, la convection électrothermique, les études de contrôle de flux utilisant des actionneurs plasma, etc. [1-11]. Tous les détails nécessaires et importants concernant le code Oracle3D ont été décrits dans les chapitres précédents, y compris les tests de validation et de performance. Ce chapitre présente les travaux numériques relatifs à l'électro-convection (EC) réalisés avec Oracle3D au cours de cette thèse. Oracle3D se compose d'un solveur de Poisson, d'un solveur de Navier-Stokes et d'un solveur de transport scalaire pour plusieurs espèces. Une combinaison de ces trois modèles physiques constitue le code complet d'Oracle3D.

En termes de solveurs de transport, trois modules distincts sont maintenant disponibles pour résoudre individuellement les modèles d'injection unipolaire (une seule espèce de charge), de pompage par électroconduction (deux espèces de charge) et de décharge plasma (trois espèces de charge). Jusqu'à présent, les études avec Oracle3D portaient principalement sur les domaines bidimensionnels. Maintenant, alors que la version parallèle est préparée, nous revisitons quelques problèmes classiques de EHD en nous concentrant principalement sur leurs aspects 3D. Les grandes lignes de ce chapitre sont les suivantes : la première partie (5.1) de ce chapitre

traite du problème de l'EHD de l'injection unipolaire dans les liquides diélectriques entre plaques parallèles. Quelques tests initiaux avec quelques cas 2D sont fournis pour valider l'implémentation en code, puis l'électro-convection 3D est discutée en détail. La deuxième partie (5.2) traite des panaches EHD induits par l'injection d'ions en configuration lame-plan. Les résultats simulés sont comparés à ceux d'études similaires disponibles dans la littérature.

5.1 Injection unipolaire entre électrodes à plaques parallèles

Le phénomène de l'injection unipolaire dans les liquides diélectriques est bien documenté expérimentalement et numériquement. Plusieurs études numériques bidimensionnelles sont disponibles pour comparer les résultats qualitatifs et quantitatifs avec ce modèle [1,2,9,9,27,28]. Un bon nombre de publications ont été réalisées avec la version précédente d'Oracle3D (version de base Fortran 77) [1-11]. Ceci nous donne l'opportunité de valider le module d'injection unipolaire de notre code parallèle Oracle3D par rapport aux résultats du code scalaire. Quelques simulations 2D avec des résultats établis sont fournies après avoir expliqué l'électroconvection en injection unipolaire avec le modèle mathématique. Une analyse 3D détaillée du problème suivra par la suite.

5.1.1 Introduction

L'électroconvection entre électrodes planaires parallèles a été largement étudiée au cours des dernières décennies [12-23]. Un liquide diélectrique, confiné entre deux électrodes métalliques à plaques parallèles, ressent un impact significatif du champ électrique produit par les deux électrodes, lorsqu'il est alimenté avec une différence de potentiel électrique. Un champ électrique élevé entre ces électrodes provoque des réactions électrochimiques complexes à la surface des électrodes. Dans de telles situations, l'injection de particules de charge peut se produire à l'interface du liquide et de la surface de l'électrode, sur une ou les deux électrodes [2,25]. Lorsque l'injection d'ions se produit sur une seule des surfaces de l'électrode, on l'appelle cela l' injection unipolaire [1].

Les ions injectés migrent du fait du champ électrique. Un champ électrique assez fort développe une instabilité qui met également le liquide en mouvement, affectant la convection globale des ions injectés [6]. Un tel mouvement de liquide, avec une conductivité suffisamment faible, peut être comparé à un mouvement liquide dû à la différence de température entre les couches de liquide (thermo-convection) [13]. Les deux modes de convection dans les liquides, la thermo-convection et l'électro-convection sont souvent comparés en fonction de leurs similarités de modèles d'écoulement induit (rouleaux, hexagones, etc.) et des dissemblances dans les mécanismes sous-jacents [12, 37].

En injection unipolaire, plusieurs études bidimensionnelles ont été publiées par plusieurs groupes utilisant différentes méthodes numériques pour résoudre l'ensemble des équations EC [7,21,23,27]. Perez et ses collaborateurs (1989) ont étudié le rôle de la diffusion et de la répulsion de Coulomb avec les algorithmes de transport à flux corrigé (FCT) dans l'EC à amplitude finie. Ils ont souligné que la distribution des charges instationnaires dépend principalement des termes d'advection, et que la diffusion ne doit être incluse que si des solutions en régime permanent sont simulées [20]. Castellanos (1990) a étudié les instabilités induites par injection et a mis en évidence un écoulement chaotique dans l'injection unipolaire à des champs électriques élevés. Vazquez et ses collaborateurs (2008) ont effectué une analyse de stabilité et ont obtenu la structure à deux rouleaux par éléments finis - FCT et Particle-in-

cell (PIC). Traoré et ses collaborateurs (2013) ont étudié l'évolution de l'écoulement de la conductivité électrique d'une structure de cellule de convection à deux structures de cellule de convection , et enfin le régime chaotique au-dessus de T = 1500. Wu et ses collaborateurs (2013) se sont penchés sur la question de la stabilité dans les cavités délimitées par des parois avec différents rapports d'aspect, en utilisant la méthode du volume fini (FVM). Toutes ces études ont été réalisées dans des cavités bidimensionnelles.

Le problème de l'injection unipolaire a été étudié avec le solveur EHD complet. Des tests de validation 2D ont été effectués pour valider les résultats avec le module d'injection unipolaire par rapport aux données déjà disponibles dans la littérature. Ce problème de l'électroconvection (EC) est souvent étudié en considérant son analogie avec le problème de la convection thermique de Rayleigh-Benard (RBC). Dans les études expérimentales, des modèles de cellules de convection hexagonale sont observés dans l'EC entre des électrodes à plaques parallèles, comme c'est également le cas dans les GR. Nous avons réalisé une étude tridimensionnelle du modèle de cellules convectives pour reproduire numériquement les cellules hexagonales dans ce problème de l'EC. Nous avons observé la formation de cellules hexagonales à l'aide de notre solveur ; cependant, nous avons constaté un impact significatif des conditions aux limites de Neumann sur les cellules convectives en évolution qui interdisait la stabilisation de ces cellules hexagonales dans nos études.

Nous avons simulé différents cas en modifiant plusieurs paramètres tels que le pas de temps, la taille de la grille, les conditions de vitesse verticale initiale, etc. Mais il a été observé qu'une limite de Neumann à gradient zéro n'était pas appropriée pour stabiliser les cellules convectives. Différents cas ont été entrepris avec des tailles de grille variant de 2 millions à 25 millions de cellules, et avec 50 processus MPI à 400 processus MPI. L'efficacité parallèle du code, dans l'échange de données aux interfaces de processus, avec ce flux dépendant de l'instabilité, était conforme aux attentes.

5.2 Injection unipolaire entre la lame et les électrodes planes

Les phénomènes d'écoulement induits électrohydrodynamiquement dans les liquides diélectriques dans le cas d'une configuration d'électrodes à lames planes ont été étudiés, à la fois numériquement [48, 53-55] et expérimentalement [56-58]. Le flux EHD se produit dans les mécanismes d'injection et de conduction du transport de charge dans les liquides diélectriques. Dans cette section, nous traitons principalement du mécanisme d'injection, qui exige l'apparition d'un certain seuil de champ électrique et en dessous de ce seuil, la conduction du champ électrique domine dans les liquides diélectriques. Au-dessus de cette valeur critique du champ électrique, l'injection de charges se produit à l'interface lame-fluide. Dans les phénomènes d'injection, la lame fonctionne comme un émetteur de charges et l'électrode plane se comporte comme un collecteur. Les particules chargées injectées mettent également en mouvement le fluide environnant en transférant leur quantité de mouvement aux particules neutres du fluide. Le fluide se déplace ainsi comme un jet vers l'électrode plane. Cet écoulement de type jet est communément appelé « panache électrohydrodynamique » [54].

Ce type de jet a été étudié pour des applications telles que l'amélioration du transfert de chaleur, le mélange des fluides, le contrôle du débit, etc. Ainsi, tout comme les panaches thermiques, la description de ces panaches EHD est importante d'un point de vue industriel. Vazquez et ses collaborateurs (1995) ont entrepris une étude comparative des panaches thermiques et des panaches EHD, en analysant les panaches axisymétriques pour divers nombres de Prandtl.

Plusieurs études numériques de Vazquez et al. ont été réalisées avec des approches numériques par éléments finis pour décrire adéquatement les panaches de EHD et leurs caractéristiques [59-61]. Perez et ses collaborateurs (2009) ont analysé les panaches EHD dans une configuration plan-lame, avec la méthode des volumes finis à l'aide d'un schéma TVD (SMART), et ont examiné différents régimes d'écoulement et structures d'écoulement caractéristiques dans de tels écoulements de EHD.

La plupart des études précédentes étaient bidimensionnelles. Ici, nous avons étudié le phénomène d'injection de la lame en 3D dans un liquide diélectrique avec Oracle3D en parallèle. Un cas d'instabilité, avec Rel=5000, C=10 et M=10 a été simulé pour cette étude. Trois lois sur l'injection ont été prises en considération. Une simulation avec une loi d'injection autonome classique, dans laquelle la densité de charge à la surface de la lame est indépendante du champ électrique à la surface de la lame, a été réalisée comme cas de référence. Après avoir compris le phénomène, nous avons incorporé deux lois d'injection simulées par Traore et al (2013). Avec les lois d'injection, un couplage plus fort entre les variables comme la charge, le champ électrique, la vitesse, etc. est induit.

Chapitre 6

EHD Conduction

Dans ce chapitre, nous présentons le modèle d'électroconduction et quelques études de cas réalisées avec Oracle3D. Principalement, certains cas de validation avec analyse des caractéristiques d'écoulement dans une configuration de canal de conduction ont été comparés avec des solutions COMSOL et les résultats sont rapportés. Les résultats obtenus avec des conditions aux limites de Robin et de Neumann non homogènes nouvellement mises en œuvre sont présentés et leur signification physique est discutée. L'impact des formulations mathématiques : implicite et explicite en cas de discrétisation de la FVM du transport des espèces, et l'effet Onsager sur la conduction de la DHM sont brièvement soulignés. Dans la dernière section, le diagramme d'écoulement avec une d'électrode de type lame-plateau est discuté en 2D et en 3D.

6.1 Introduction

Les charges électriques présentes dans la conduction électrohydrodynamique (EHD) sont créées par dissociation et recombinaison d'un électrolyte faible dans un liquide non polaire ou légèrement polaire. Lorsqu'un champ électrique externe est appliqué, des couches avec une charge électrique nette apparaissent à côté de chaque électrode. Ce sont les couches d'hétéro-charges, avec une polarité opposée à celle des électrodes. Le mouvement des espèces chargées dans le liquide est dû à la densité de force électrique, $\vec{F}_e$, qui résulte de trois composantes physiques différents. La première et la plus importante est la force de Coulomb qui est le premier terme du côté droit de l'équation (1). Le deuxième terme est la force diélectrique qui n'est présente que lorsque le gradient de permittivité ($\nabla\varepsilon$) existe. Le troisième terme est connu sous le nom de force électrostrictive qui, étant le gradient d'un scalaire, peut être incorporé dans la pression [1,2].

$$\vec{F_e} = q\vec{E} - \frac{1}{2}E^2\nabla\varepsilon + \nabla\left[\rho\frac{E^2}{2}\left(\frac{\partial\varepsilon}{\partial\rho}\right)_T\right] \quad (1)$$

Ainsi, seule la force de Coulomb est à l'origine au mouvement EHD permanent dans de tels phénomènes de conduction. Dans la plupart des applications EHD, cette force de Coulomb met en mouvement le liquide qui est utilisé pour les applications prévues comme le pompage, les jets muraux, etc. La force nette de Coulomb n'est générée que s'il y a un déséquilibre dans les densités des porteurs de charge positifs et négatifs. Les configurations d'électrodes asymétriques jouent un rôle important dans la création de ce déséquilibre dans les densités des porteurs de charge, qui ont été explorées dans de nombreuses études [2-4]. Le mécanisme de conduction EHD fournit une approche non mécanique et à faible consommation d'énergie pour générer ou contrôler un flux de façon active, qui peut être utilisé pour des applications ciblées dans des conditions terrestres et en microgravité [6].

Nous expliquons la conduction EHD dans les liquides diélectriques à partir des couches hétérogènes d'espèces générées par la dissociation et la recombinaison des ions sous l'influence d'un champ électrique externe. En l'absence de champ électrique externe, les taux de dissociation (k_D) et de recombinaison (k_R) sont considérés constants. Nous avons étudié le modèle d'Onsager basé sur le concept d'amélioration du processus de dissociation par augmentation du champ électrique externe [7,9]. Le courant électrique est l'un des paramètres les plus importants lorsque l'on discute du phénomène d'électroconduction. Nous avons tracé le courant électrique plusieurs fois pour comparer et analyser les résultats obtenus avec nos simulations. Le courant électrique se manifeste principalement par le flux combiné des densités des espèces de charge à travers les surfaces des électrodes.

6.3 Études de validation et d'analyse des flux

Nous avons présenté les toutes premières études de cas qui ont été réalisées pour valider le modèle de conduction EHD tel qu'implémenté dans Oracle3D. Les effets des conditions aux limites telles que Neumann BC, Robin BC, Neumann non-homogène et Periodic BC sont étudiés, et les résultats ont été comparés aux résultats issus de COMSOL. L'écoulement par électroconduction a été simulé dans un canal rectangulaire à domaine non dimensionnel.

Résumé de cette section :

6.3.1 Cas de validation
- I. Substrat Neumann BC sans effet Onsager
- II. Substrat Robin BC avec effet Onsager
- III. Sans la C.-B. périodique sur les faces est et ouest

6.3.2 Impact de l'effet Onsager

6.3.3 Effet de la condition limite du merle sur le substrat

Nous avons comparé deux résultats de simulation préformés avec notre configuration de canal EHD avec et sans l'effet Onsager. Sur substrat, nous avons utilisé Robin BC pour l'espèce et Neumann non-homogène pour le potentiel électrique. Nos résultats avec effet Onsager montrent une augmentation du courant électrique sur l'électrode haute tension. Il est à noter que dans notre modèle de conduction, la nouvelle charge est générée par dissociation, et avec l'effet Onsager, nous augmentons le taux de dissociation des espèces neutres dans le liquide. Par conséquent, les flux de charges sur les électrodes augmentent, ce qui entraîne des valeurs de courant plus élevées.

6.4 Conduction dans la géométrie du plan de la lame

Les configurations d'électrodes asymétriques sont importantes pour générer une force de Coulomb nette dans les flux de DHM. Dans cette section, nous réexaminons le réglage de l'électrode du plan de la lame avec un écoulement dominé par la conduction. On pense généralement que l'apparition de phénomènes de conduction et d'injection dans les liquides diélectriques dépend de la tension électrique. La conduction est censée se manifester lorsque la tension électrique appliquée est inférieure à une tension de seuil nécessaire à l'injection [2]. Cependant, les expériences montrent que la conduction et l'injection peuvent coexister lorsqu'un fort champ électrique est présent dans un liquide. Les autres facteurs tels que les propriétés du liquide, la configuration des électrodes, la température de travail, la quantité d'impuretés dans le liquide, etc. ne peuvent pas être négligés lors de la décision sur la question de savoir quels phénomènes surviennent.

Il est essentiel de comprendre les caractéristiques de débit pour décrire complètement le phénomène de DHM qui les génère. Il a déjà été observé que dans les réglages dominants de conduction, le flux global est de l'électrode plane vers l'électrode lame[6], ce qui est opposé aux cas d'injection. Nous avons également simulé le cas classique (Traore et al. (2015)) avec notre code parallèle pour comprendre le comportement classique de la conduction dans ce contexte. Deux tourbillons contrarotatifs sont observés des deux côtés de l'électrode à lame avec des vecteurs de vitesse. La direction globale de l'écoulement devant la zone de la lame est de l'électrode plane à l'électrode de la lame, comme indiqué dans [6].

Nous avons effectué des tests numériques pour examiner les caractéristiques d'écoulement dans des configurations lame-plan avec un ensemble différent de paramètres, sous l'influence de la conduction. Pour les paramètres d'entrée, nous avons pris M=0,2, Co=0,1 avec le nombre électrique de Reynolds (R_{el}) égal à 2500. La simulation a été exécutée pour 5 unités de temps non dimensionnelles, et des vecteurs de vitesse ont été enregistrés après chaque 0,05 heure non dimensionnelle. Nous avons observé que l'écoulement initial se déplace d'un plan à l'autre (t=0,75) ; mais à t=5 nous observons que les vecteurs se déplacent d'un plan à l'autre. Ceci s'est avéré contradictoire avec ce qui a été observé dans notre cas de conduction classique et notre réf. [6]. Le comportement d'écoulement observé présentait deux types de régimes d'écoulement différents dans ce cas. Nous appelons le régime d'écoulement par conduction classique comme R1, où la direction de l'écoulement à l'état stationnaire est du plan vers la lame, comme indiqué dans [6]. Le comportement d'écoulement observé à l'état d'équilibre avec ces réglages de cas est appelé régime R2, où la direction de l'écoulement est de la lame vers l'électrode plane. Cependant, le flux est resté 2D dans ce cas, même avec une grille 3D.

Nous avons étudié un autre cas avec un nombre de Reynolds électrique plus élevé (Rel = 10000), les autres paramètres étant M=0,1, Co=0,1. Dans ce cas, nous avons réduit de moitié la valeur de M. Nous avons d'abord analysé les vecteurs de vitesse obtenus avec les simulations 2D. Nous avons observé la même configuration que celle décrite ci-dessus avec M=0,2, les vecteurs de l'écoulement allant d'abord de la plaque vers la lame, mais après un certain temps, l'écoulement s'inverse et va de la lame vers la plaque. Comme la valeur Rel est quatre fois plus élevée dans ce cas, nous avons également étudié le cas 3D pour les mêmes réglages.

Nous avons tracé les iso-surfaces de la z-vorticité pour analyser la symétrie des tourbillons, s'il y en a. Dans les premières étapes temporelles, nous avons observé que les iso-surfaces de la corrélation z étaient symétriques le long de l'axe z. Cependant, avec l'évolution du flux, nous

avons commencé à observer la nature turbulente du flux, et la symétrie du flux le long de la direction z a été reprise par des caractéristiques 3D réelles. Les iso-surfaces de corrélation de la torsion z à t=1,5 suggèrent que l'écoulement est évidemment tridimensionnel.

Il montre que le diagramme d'écoulement en conduction avec des configurations d'électrodes asymétriques dépend fortement des propriétés du liquide et non seulement de la tension appliquée. Des analyses plus détaillées avec différentes propriétés de liquide et une fraction d'impuretés variables doivent être étudiées afin d'améliorer encore la compréhension de ces écoulements de DHM. Nous soulignons également que dans de tels écoulements électroconducteurs, il existe des combinaisons de paramètres non dimensionnels (M, C0, Rel) pour lesquels l'écoulement restera en régime R1, ce qui est considéré comme un comportement de conduction typique. Cependant, nous pouvons avoir quelques combinaisons de propriétés du fluide et de paramètres d'entrée qui conduiront à un tel ensemble de paramètres (M, C0, Rel) pour lesquels nous observerons l'inversion de flux vers le régime R2. Nous avons mentionné plusieurs facteurs qui affectent ce comportement et d'autres études sont nécessaires pour une compréhension plus large de ces phénomènes.

Chapitre 7.

Modèle Suzen-Huang pour actionneurs DBD

Le contrôle de l'écoulement d'air a de nombreuses applications pratiques. Particulièrement dans les industries liées à l'aérodynamique, le contrôle ou la modification de l'état d'un écoulement reste une préoccupation majeure pour les ingénieurs. Au cours des 15 dernières années, les « actionneurs plasma » ont fait l'objet de nombreuses recherches où ces actionneurs ont été étudiés comme des candidats de choix pour des applications de contrôle d'écoulement.

Un actionneur plasma comporte généralement deux électrodes séparées par un matériau diélectrique. Une électrode est généralement exposée à l'air et l'autre est mise à la terre, Fig. 7.1.1.1. Cette configuration est connue sous le nom d'actionneur plasma à décharge à barrière diélectrique (SDBD) [1,2]. Lorsqu'il est alimenté par une tension électrique, un champ électrique est généré entre les deux électrodes qui, s'il est suffisamment fort, peut produire une décharge électrique dans l'air ambiant. Au sein de la décharge, des ions sont produits par des mécanismes tels que l'ionisation, la recombinaison, le détachement, la photoionisation, etc. qui se produisent à des échelles de temps picosecondes [2,9]. La force électrique induite fait dériver ces charges sous l'influence du champ électrique en fonction de leur polarité. Pendant le mouvement, une partie de la quantité de mouvement de ces ions est transférée aux molécules neutres par collisions entre elles et, après un certain temps, tout le fluide entourant l'électrode HV est mis en mouvement. Ce phénomène d'écoulement est appelé vent ionique ou vent électrique.

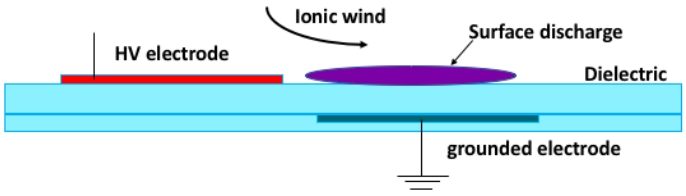


Fig. 7.1.1.1 Schéma type d'un actionneur plasma à décharge à barrière diélectrique.

Généralement, la force électrique produit un jet le long de la surface diélectrique juste au-dessus de l'électrode mise à la terre en raison du phénomène de décharge, figure 7.1.1. En raison de ce jet, l'air au-dessus du bord droit de l'électrode haute tension est aspiré le long de la paroi diélectrique, avec une vitesse de quelques mètres par seconde. Ce jet peut être utilisé pour modifier l'écoulement global le long de la paroi. Principalement, dans les applications de contrôle de décollement, cet actionneur est placé près du point de séparation. Plusieurs études expérimentales [2,6] et numériques[4,5,8,10] ont été publiées pour explorer les caractéristiques des actionneurs DBD. Moreau (2007) et Benard et ses collaborateurs (2014) ont fourni un état de l'art détaillé des études expérimentales effectuées au cours des 15 dernières années, tout comme Corke et ses collaborateurs (2010).

Ces actionneurs présentent deux directions d'étude : (1) l'étude de la physique des plasmas impliqués dans le phénomène, (2) l'exploration de l'intérêt de ces actionneurs plasmas dans des applications industrielles. Plusieurs physiciens ont étudié numériquement le phénomène du vent ionique avec une, deux ou plusieurs espèces chimiques dans le plasma [14,15,30]. Des études numériques récentes ont utilisé trois espèces (électrons, ions positifs et négatifs) et ont montré le mécanisme en résolvant les équations de transport de ces espèces [31-34]. Les modèles numériques qui impliquent le calcul de la distribution de la densité de charge dans le temps sont appelés modèles autosuffisants. Plusieurs modèles numériques ont été proposés en fonction de la nature des phénomènes électriques [1,14,22]. Dans ce chapitre, nous traitons principalement du modèle de Suzen-Huang [5].

Seth et ses collaborateurs (2018) passent en revue la plupart des études initiales réalisées avec le modèle Suzen-Huang (SH). Ces recherches étaient basées sur les différentes mises à jour [13,20,22] du modèle et sur diverses applications qui ont montré des résultats prometteurs [7,12,24]. Le modèle SH est fondamentalement un modèle d'ingénierie simple qui n'intègre pas toute la physique des plasmas impliqués dans les actionneurs DBD. Cependant, il se rapproche de la densité de charge basée sur les résultats expérimentaux et calcule la force de Coulomb induite par l'actionneur DBD. Cette force de Coulomb est utilisée comme terme source dans les équations de Navier-Stokes pour simuler les conditions d'écoulement. Plus récemment, Mahfoze et ses collaborateurs (2017) ont utilisé deux variantes du modèle SH pour étudier la réduction de la traînée de frottement dans un canal. La simplicité de mise en œuvre et les faibles coûts de calcul de ce modèle ont permis à la communauté d'explorer la pertinence de ce modèle dans de nombreuses applications. Ici, le modèle SH a également été implémenté dans Oracle3D et nous avons réalisé une étude paramétrique et quelques cas de contrôle d'écoulement seront présentés dans ce chapitre.

La dérivation mathématique du modèle SH a été expliquée dans ce chapitre. On voit qu'une des faiblesse du modèle SH que nous avons besoin de connaître la distribution de densité de charge.

Et, comme cette distribution de densité de charge peut changer avec différentes configurations d'actionneurs, nous devons calibrer ce modèle avec différentes configurations d'actionneurs. La densité de charge maximale (ρ_c^{max}) et la longueur de Debye (λ_D) sont les deux paramètres scalaires du modèle SH qui dépendent des conditions expérimentales. La distribution numérique globale de la densité de charge dans le modèle SH dépend de ces deux paramètres.

Le modèle Suzen-Huang (SH) a été implémenté dans Oracle3D pour explorer ses performances dans la simulation du flux avec des configurations d'actionneurs plasma DBD. Une étude paramétrique a été publiée sur la base des paramètres géométriques qui caractérisent le fonctionnement des actionneurs DBD. L'impact de l'épaisseur du diélectrique, de l'écart entre les électrodes, de la fréquence et de la forme d'onde de la tension, etc. a été décrit en fonction de leur effet sur la vitesse maximale induite et la force EHD moyenne. Une brève étude de contrôle d'écoulement a été mise en place, en utilisant la valeur de la force EHD obtenue expérimentalement comme terme source numérique dans les équations de Navier-Stokes, de façon à reproduire numériquement les vitesses expérimentales. La capacité des actionneurs DBD à manipuler un écoulement a été démontrée avec une configuration de marche descendante, à faible nombre de Reynolds. Il a été démontré qu'une séparation laminaire pouvait être considérablement contrôlée en plaçant l'actionneur à l'extrémité de la marche avec les deux électrodes perpendiculaires l'une à l'autre. Dans l'ensemble, nous avons préparé le code parallèle pour ces simulations d'actionneurs DBD, de sorte que des études 3D complètes puissent maintenant être réalisées avec différentes configurations de DBD et des nombres de Reynolds plus important.

ACRONYMS

2D/3D	Two/Three Dimensional
AVX	Advanced Vector Extension
BC	Boundary Condition
BFS	Backward Facing Step
CDS	Central Differencing Scheme
CFD	Computational Fluid Dynamics
CPU	Central Processing Unit
CV	Control Volume
DBD	Dielectric Barrier Discharge
DNS	Direct Numerical Simulation
EC	Electro-Convection
EHD	Electro-hydro-dynamics
FVM	Finite Volume Method
HPC	High Performance Computing
HV	High Voltage
IDC	Improved Deferred Correction
LES	Large Eddy Simulation
MPI	Message Passing Interface
NS	Navier-Stokes
OpenMP	Open Multi-Processing
PDEs	Partial Differential Equations
PIV	Particle Image Velocimetry
QUICK	Quadratic Upstream Interpolation for Convective Kinematics
RAM	Random Access Memory
RBC	Rayleigh-Benard Convection
SDBD	Single Dielectric Barrier Discharge
SDC	Standard Deferred Correction
SH	Suzen-Huang
SIMPLE	Semi-Implicit Method for Pressure Linked Equations
SMART	Sharp and Monotonic Algorithm for Realistic Transport
TVD	Total Variation Diminishing
UD	Upwind Difference

NOMENCLATURE

Symbol	*Discription*	*Unit*
ε_0	Permittivity of vacuum/air	[F/m]
$\vec{E}$	Electric field vector	[V/m]
$\vec{F}$	Force field vector	[N/m^3]
$\vec{J}$	Electric Current density	[A/m^2]
k_B	Boltzmann Constant	[J/K]
k_D	Dissociation constant	
k_R	Recombination constant	
N_p	Positive charge density	[C/m^3]
N_n	Negative charge density	[C/m^3]
N_{eq}	Charge density at equilibrium	[C/m^3]
$\vec{u}$	Velocity vector	[m/s]
λ_D	Debye Length	[m]
ρ_0	Density	[kg/m^3]
ε	Permittivity	[F/m]
D	Diffusion coefficient	[m^2/s]
I	Electric Current	[A]
V	Electric voltage	[V]
e	Elementary electronic charge	[C]
K	ionic mobility coefficient	[m^2/V.s]
q	Ionic charge density	[C/m^3]
μ	Dynamic viscosity	[kg/m.s]
ν	Kinetic viscosity	[m^2/s]
$\vec{n}$	unit normal vector	

NON-DIMENSIONAL PARAMETERS

P_e	Peclet Number
R	Reynolds Number
R_{el}	Electric Reynolds Number
T	Electric Rayleigh Number
C	Injection Strength
M	Mobility Parameter
α	Ionic diffusion number
Dc	Non-dimensional Coulomb force parameter in SH model
ε_r	Relative permittivity

Table of Contents

Chapter 1.
Introduction

In 21st century, the advancements in every domain of science and engineering have become closely dependent on the progress in computer science. The ability of modern computers to do huge number of mathematical calculations in unimaginably quick time is at the core of this dependence. Availability of hardware and relevant software is the key behind such impressive ability of the computers. Moore's law has guided the semiconductor industry for last 5 decades to advance and anticipate the future of computer hardware; according to which the number of transistors on integrated circuits were to double every year from 1965, and after 1975 it was revised to double every two years. This led to a rapid growth in the computing hardware technologies which still continues. On the other hand, several programming models were proposed to effectively design the software programs to benefit from the ever-increasing hardware capabilities.

It is evident that to really profit from the progress in hardware and software, both technologies must advance coherently with each other. In fact, as the hardware industry has advanced ahead of the software domain, the software innovations have to follow according to the already available hardware architectures. The existing software packages and the legacy scientific codes have to be modernized with the ever-changing processor architectures to allow the users to benefit from the ubiquitous multicore processors. Scientific and engineering research community is one of the biggest consumers of these growing high-performance computing (HPC) technologies. And, there are organizations which are leading the way forward for the scientific community to keep it updated with these fast-paced HPC technologies. For example, the Partnership for Advanced Computing in Europe (PRACE) is an organization to facilitate high-impact scientific discoveries and engineering research across all disciplines by providing most advanced computing and data management resources available across Europe to the member European communities.

Many important problems of science and engineering cannot be undertaken without the modern HPC systems. The climate or weather prediction, turbulence at the smallest scales, full body aerodynamic studies of aerospace vehicles etc. are some problems where the length scales matter a lot, on the other hand, the physics of plasma discharge, atomic collisions etc. are the problems where the time scales are also immensely important. Both types of problems require the modern HPC capabilities to make proper understanding of the underlying phenomena.

The context of this thesis is based on an in-house electrohydrodynamic solver, Oracle3D, with which we wish to study the interaction of electrostatics with hydrodynamics. Oracle3D is a Finite Volume based multi-block structured grid, incompressible Navier-Stokes solver for steady and unsteady flow conditions. It uses SIMPLE algorithm for pressure velocity coupling and 2nd order Gear Scheme for temporal discretization. It approximates the convective fluxes with Total Variation Diminishing (TVD) scheme with several available flux limiters. Improved Deferred Correction (IDC) scheme is used to treat the diffusive fluxes. These features are

explained with details in following chapters. The code is mainly developed to simulate various electro-hydrodynamic problems.

There are several real-life situations where some kinds of ionic species interact with the surrounding neutral molecules of fluid and there occurs an exchange of energy, momentum, electric potential etc. between the ions and neutral particles, which demands detailed study. Such interactions of ionic species with the neutral fluid is considered under the discipline of Electrohydrodynamics (EHD). The overall aim of this thesis consists of modernizing the code Oracle3D to tackle large and complex EHD simulations on advanced HPC systems and numerically studying the EHD problems in three dimensional configurations.

Intricately coupled applications of modern industries bring together the branches of hydrodynamics, electrostatics, electrochemistry etc. under the multi-disciplinary EHD research. The Electro-Fluido-Dynamics (EFD) group at the Institut Pprime in France is actively participating in building the new technologies comprising the EHD domain and leading the way in solving many industrial problems such as the aerodynamic flow separation. The group has state-of-the art experimental and HPC facilities at the Institut Pprime in collaboration with the University of Poitiers. Efforts have been made, during this PhD work, to significantly advance the numerical research capability of the group by designing and implementing the parallel strategy for the code Oracle3D and rigorously validating it. Detailed numerical work and computational methodology has been provided in this thesis for the current and forthcoming users of this EHD solver.

This work does not focus in-depth on a single physical problem; however, it provides a range of EHD applications including unipolar injection, electro-conduction, flow separation control with plasma discharge etc. with initial validation tests in several domain configurations to build a broad database for the new code. Some classical EHD problems were revisited and the background summary with bibliography of the relevant previous works has been provided within corresponding chapters and sections. As the code was parallelized beginning from the legacy Fortran 77 version, MPI implementation details are provide in this thesis to facilitate further advancements and developments in future.

Scope of this thesis

The in-house EHD solver, Oracle3D, has been under development for more than a decade now, and advanced features are being added regularly to extend its scope to newer and more complex EHD problems. During the course of this thesis, the solver ‘Oracle3D’ was upgraded with modern Fortran features and more importantly it was parallelized with Message Passing Interface (MPI) to be run on distributed memory CPU clusters. Parallelizing a scientific code presents a lot of challenges in terms of programming the intricate core level details, and then adapting the whole scalar code to the parallel methodology. Thus, the task of parallelizing the complete EHD solver was divided in sub-tasks which included preparing individual scalar and parallel codes for simple physical models such as the Poisson’s solver, the Navier-Stokes solver, a scalar transport solver etc.

To simplify pre-processing stage of grid preparation, a Fortran code was designed to read block-grid data from ‘BlockMesh’ and convert it into the suitable format for multi-block Oracle3D grids. Advanced MPI features of Cartesian topologies, groups, inter-communicators etc. were

implemented in the code to prepare a scalable message passing strategy for the parallel computations. The first part of the thesis deals with the methodology used to parallelize the solver codes and their validation with existing results. A short summary of the chapters is as follows:

- Chapter 2 briefly mentions the Finite Volume method (FVM) as the numerical approach used for discretization of governing equations in Oracle3D. Instead of general transport equation, specific equations used in the code are taken as discretization examples wherever possible. The need and the implementation details of 'TVD' schemes are detailed for the users of the code. Some new boundary conditions were implemented in the code which are explained with relevant discretization approach.

- Chapter 3 details the methodology used for parallelization of the code with advanced MPI features. A general outline towards parallel programming models is provided in 1st section. Section 2nd deals with the Cartesian topology features of MPI which were initially used to parallelize single block grids. The 3rd section puts forth the whole strategy used to parallelize the multi-block grid geometries, with complete implementation details for users. Finally, some scalability tests are provided with explanations to judge the parallel efficiency of the new code.

- Chapter 4 provides all the test cases performed to validate the individual parallel solvers: Poisson solver, Navier-Stokes solver, advective transport solver. Most of the new features added were validated with different number of cores to verify the message passing approach in the parallel solvers.

In the second part, we provide the EHD studies performed with Oracle3D during the course of this thesis.

- Chapter 5 deals with EHD unipolar injection. The electro-convection problem is defined, and a brief literature review is given to start with. Some initial 2D studies are presented to validate the unipolar injection model of the code. Three-dimensional convective cells' pattern formation in parallel plate electrode configuration is investigated in detail. Then, 3D injection plumes are investigated with blade-plane electrode geometry under different injection laws.

- Chapter 6 accounts for several computations in the framework of electro-conduction phenomenon. Some validation tests were performed to compare with Comsol, and results are provided. A 3D conduction channel was simulated for the 1st time. In second section, we provide some insight into the flow pattern observed with our conduction case settings in blade-plane electrode geometry.

- The last chapter deals with plasma discharge. We have used, in this first approach the Suzen-Huang (SH) model which is described with the derivation of model and its parameters. Impact of the Debye length is briefly explored in context of SH model. A parametric study dealing with the geometrical and electrical parameters characterizing the DBD actuators is provided. A study with experimental force used as a source term

in Navier-Stokes equations is highlighted. Lastly, a brief study with a laminar flow control over a backward facing step is provided.

Chapter 2.
Finite Volume Method in context of Oracle3D

Finite Volume method (FVM) is one of the most popular mathematical approaches among others, which are used to solve the problems of continuum mechanics by discretizing the corresponding partial differential equations in time and space. Spatial discretization of a problem refers to dividing the spatial domain of a problem into much smaller geometrical entities like computational cells, faces and nodes. Then, the physical problem in the whole spatial domain is combinedly described by the algebraic relations defined on these individual computational cells and nodes. Algebraic equations for the individual computational cells are obtained by integrating the partial differential equations with FVM over each discrete cell.

When the problem is of unsteady nature, discretization in time is also required which is carried out by dividing the overall problem time into much smaller time steps. The evolution of physical problem with smaller time steps altogether provides the complete unsteady solution. A complete flow chart for the general discretization process as usually followed in numerical analyses in given in Fig. 2.1. In, this chapter, we will mainly talk about the FVM strategies as used in our solver, Oracle3D. We have tried to explain the various newly implemented discretizations and the boundary conditions with the actual problems which are encountered in the physical models that are available in the code. This is to facilitate the understanding of the code for the future users.

2.1 Finite Volume Method: a brief overview

A large number of methods and schemes are available within the framework of Finite Volume approach, depending on the nature of physical problem (diffusion, convection etc.), required order of accuracy, nature of grid etc. Inherent conservative nature of Finite Volume method is its prominent feature which puts it ahead all other numerical techniques when computational fluid dynamics (CFD) is talked about. When dealing with fluxes of conservative quantities over the faces of computational cells, it is stated that the flux entering a control volume is identical to the flux leaving the adjacent volume, making the FVM strictly and inherently conservative. Especially, this feature is an added advantage for the fluid mechanics problems where we have to satisfy the conservation laws of mass, momentum and energy in every single problem at each time step. With the significant advancements in CFD, in last few decades, FVM has gained a lot of popularity by being able to tackle all kinds of complex physical problems.

As the other numerical approaches, like Finite Difference method (FD) and Finite Element method (FEM), in Finite volume also we transform the partial differential equations (PDEs) into linear algebraic equations. All the physical phenomena are described by some kind of PDEs, which distinctly define the mathematical and physical nature of the problem under consideration. For example, in CFD the most frequently encountered PDEs are the Navier-Stokes equations, which are defined by the conservation laws of mass and momentum. Towards the discretization process of PDEs, which requires the transformation of volume and surface

integrals into discrete algebraic equations, we undeniably come across the divergence theorem (Gauss' theorem) in FVM.

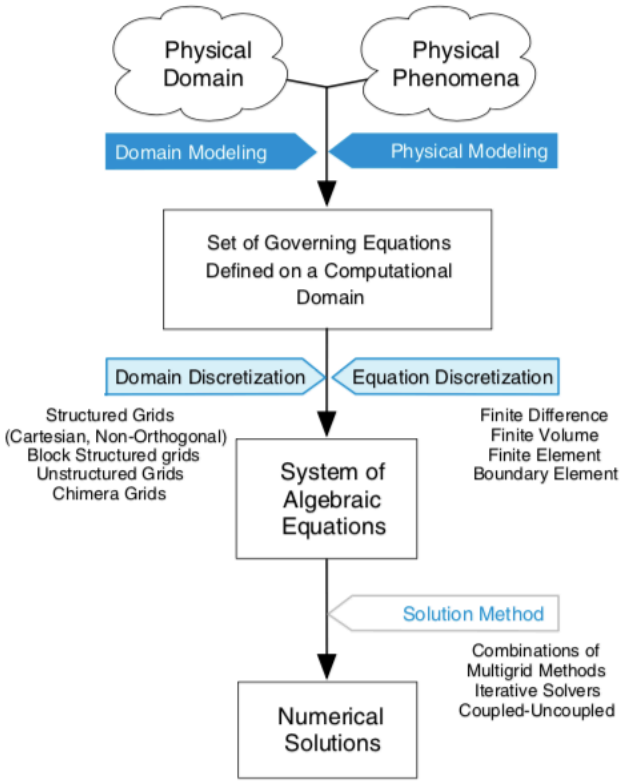


Fig. 2.1 A general overview of discretization process (courtesy Moukalled et al. 2016)

The divergence theorem states that the overall flux of a vector field ($\vec{u}$) through any closed surface (S) is equal to the total volume of all the sources and sinks over the region confined by that surface, eq. (2.1). Here the total volume of all sources and sinks is defined by the volume integral of the divergence of that vector field. Thus, with this theorem we usually convert the volume integrals into surface fluxes, which are then used to form the discrete algebraic equations.

$$\int_V (\nabla . \vec{u})\ dV = \int_S \vec{u} . \vec{n}\ dS \tag{2.1}$$

We present here, as an example, the conservation equation for a general scalar variable φ to express the utilization of divergence theorem in FVM. Eq. 2.2 shows the four terms present in a general conservation equation: transient term, convective term, diffusion term and the source term. Here, ρ is the density of fluid, $\vec{u}$ is the velocity vector field, and Γ is the diffusion coefficient of the variable φ. We keep the treatment of the transient term for later and show here the transformation of this PDE (eq. 2.2) into surface fluxes of control volumes. Eq. 2.3 represents the steady state form of eq. 2.2. These two equations are for the whole problem domain.

$$\underbrace{\frac{\partial(\rho\varphi)}{\partial t}}_{\text{Transient term}} + \underbrace{\nabla.(\rho\varphi\vec{u})}_{\text{convective term}} = \underbrace{\nabla.(\Gamma^{\varphi}\nabla\varphi)}_{\text{diffusive term}} + \underbrace{Q^{\varphi}}_{\text{source term}} \tag{2.2}$$

$$\nabla.(\rho\varphi\vec{u}) = \nabla.(\Gamma^{\varphi}\nabla\varphi) + Q^{\varphi} \tag{2.3}$$

We now take the discretized finite volume cells of the domain and integrate eq. 2.3 over a cell C. Eq. 2.4 is the volume integral form of the steady conservation equation, over a control volume cell. Now, we use the eq. 2.1 (divergence theorem) to convert the volume integrals of convective and diffusive terms into surface integrals, as depicted by eq. 2.5. Here V_c is the volume of the cell C and $\vec{S}$ is the surface vector of the cell surface. Equation 2.5 is usually termed as the semi-discretized equation in FVM, as it represents the contributions by individual finite volume cells [1].

$$\int_{V_c} \nabla.(\rho\varphi\vec{u})\, dV = \int_{V_c} \nabla.(\Gamma^{\varphi}\nabla\varphi)\, dV + \int_{V_c} Q^{\varphi}\, dV \tag{2.4}$$

$$\int_{\partial V_c} (\rho\varphi\vec{u}).d\vec{S} = \int_{V_c} (\Gamma^{\varphi}\nabla\varphi).d\vec{S} + \int_{V_c} Q^{\varphi}\, dV \tag{2.5}$$

With the semi-discretized equation, we have to obtain the discrete algebraic equations for each cell, which will be the contribution of individual cells in diffusion, convection and source terms for the whole problem. Diffusion, convection and the effect sources/sinks are three phenomena which are completely of different physical nature, so while explaining they are generally dealt separately to obtain their contribution and then finally combined for the overall solution. We will discuss the mathematical approach to get the discrete algebraic equations for each term in following sections.

2.2 Discretization of Diffusion term

We take an equation with diffusion term, as an example, to understand this discretization process. This equation (Eq. 2.6) is the charge density equation from the Suzen-Huang model [2], which is a part of the overall framework of this thesis. This equation was newly implemented in the Oracle3D code, so it was felt important to provide the discretization of this equation here as an example problem. In equation 2.6, the left-hand side (LHS) term solves our requirement of a diffusion term.

$$\nabla.(\varepsilon_r\nabla\,\rho_c) = {}^{\rho_c}/_{\lambda_D^2} \tag{2.6}$$

Here ρ_c refers to the charge density, λ_D is the Debye length and ε_r is the relative permittivity of the medium. Rearranging this equation to get all terms with ρ_c on one side as:

$${}^{\rho_c}/_{\lambda_D^2} - \nabla.(\varepsilon_r\nabla\,\rho_c) = 0 \tag{2.7}$$

Let us integrate eq. (2.7) as per the FVM approach to discretize it, which gives us

$$\int_{V_c} \left(\rho_c / \lambda_D^2\right) dV - \int_{V_c} \nabla.(\varepsilon_r \nabla \rho_c)\ dV\ = 0 \tag{2.8}$$

Using the divergence theorem for the diffusion term:

$$\int_{V_c} \left(\rho_c / \lambda_D^2\right) dV - \int_{S} (\varepsilon_r \nabla \rho_c).\vec{n}\ dS\ = 0 \tag{2.9}$$

Let us take $\rho_c = \varphi$, following our code's convention. Integrating the first term of equation (2.9) separately, for control volume with centre node P

$$\int_{V_c} \left(\varphi / \lambda_D^2\right) dV = \left(\frac{\Delta V}{\lambda_D^2}\right) \varphi_P \tag{2.10}$$

Where ΔV is the volume of the cell with center node P. Now considering the second term of eq. (2.9) for cell with centre node P, which is a surface integral over this cell. We can replace the surface integral with a summation over the control volume faces. This transformation is the first approximation introduced in our FVM approach. Here *k* represents all the faces of the control volume in all six directions (east, west, north, south, bottom and top). A 2D arrangement of cells with neighbour nodes in respective directions, represented with capital letters, are shown in Fig. 2.2. Orthogonal cells are shown in Fig. 2.2 just to simplify the understanding of nomenclature used here.

$$\int_{S} (\varepsilon_r \nabla \varphi).\vec{n}\ dS\ = \sum_{k=newsbt} (\varepsilon_r \nabla \varphi)_k .\vec{n}_k S_k = \sum_{k=newsbt} \varepsilon_{rk} S_k (\nabla \varphi)_k .\vec{n}_k \tag{2.11}$$

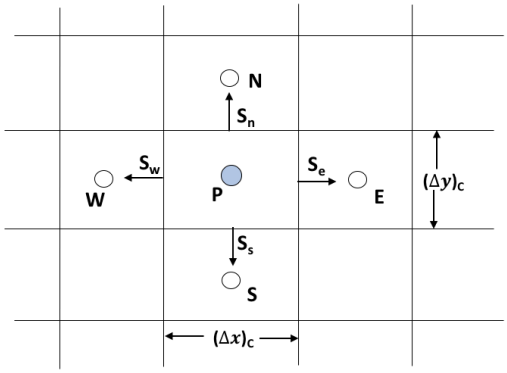


Fig. 2.2 Control volumes arrangement in a 2D grid

2.2.1 The Standard Deferred Correction (SDC)

Here S_k is the area of the respective face of the control volume, and $\vec{n}_k$ is the unit normal vector for k face. Now in the above expression we need to compute $(\nabla \varphi)_k .\vec{n}_k$ with a suitable approximation. As we often deal with complex geometries where non-orthogonal grids are usually encountered, we demonstrate here the standard deferred correction (SDC) method which is used to deal with the non-orthogonal control volumes, along with the diffusion term discretization [3,8]. We follow the normal decomposition approach (Fig. 2.3) for our discretization in which $\vec{n}_k$ can be represented as follows:

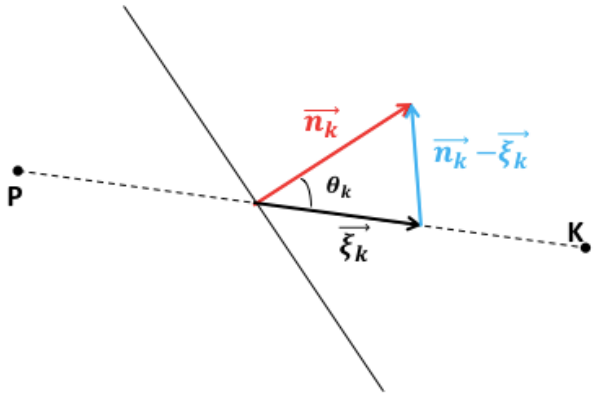


Fig. 2.3 Decomposition of surface normal vector ($\vec{n}_k$) in SDC

$$\vec{n}_k = \vec{\xi}_k + (\vec{n}_k - \vec{\xi}_k)$$

Here $\vec{\xi}_k$ is the unit vector joining the central nodes of the two adjacent control volumes, using this we can write

$$(\nabla \varphi)_k . \vec{n}_k = (\nabla \varphi)_k . \vec{\xi}_k + (\nabla \varphi)_k . (\vec{n}_k - \vec{\xi}_k) \tag{2.12}$$

SDC utilizes this normal decomposition and gives an iterative procedure to solve the diffusive term in equation (2.11):

$$\sum_{k=newsbt} \varepsilon_{rk} S_k (\nabla \varphi)_k . \vec{n}_k = \sum_{k=newsbt} \varepsilon_{rk} S_k (\nabla \varphi)_k^{\,m} . \vec{\xi}_k + \sum_{k=newsbt} \varepsilon_{rk} S_k (\nabla \varphi)_k^{\,m-1} . (\vec{n}_k - \vec{\xi}_k) \tag{2.13}$$

Where *m* refers to the solution of $(\nabla \varphi)_k$ during current iteration and m-1 is the solution from the previous iteration. The second term on the RHS in equation (2.13) is solved explicitly from the terms of previous iterations and they form the deferred correction part of the solution. The first term on RHS can be approximated with second order central scheme for the first derivate which is written as eq. (2.14):

$$\varepsilon_{rk} S_k (\nabla \varphi)_k^{\,m} . \vec{\xi}_k = \frac{\varepsilon_{rk} S_k}{d(P,K)} (\varphi_K^m - \varphi_P^m) \tag{2.14}$$

Here d(P, K) is the distance from Point P to point K. For, simplicity we consider a 2D grid, as shown in Fig. 2.2, and the extension to 3D cases will remain similar. So, using equation (2.10) and (2.13) in SH model equation (2.9) we get eq. (2.15). We keep the deferred correction term on the RHS of eq. (2.15) as it is computed explicitly from the known values of the previous iterations (m-1).

$$\left(\frac{\Delta V}{\lambda_D^2}\right)\varphi_P^m - \sum_{k=news}\frac{\varepsilon_{rk}S_k}{d(P,K)}\left(\varphi_K^m - \varphi_P^m\right) = \left[\sum_{k=news}\varepsilon_{rk}S_k\,(\nabla\varphi)_k^{\,m-1}.(\vec{n}_k - \vec{\xi}_k)\right] \quad (2.15)$$

We solve the eq. (2.15) without the RHS term now, just to show the following approach more clearly. The RHS term of eq. (2.15), which is the deferred correction contribution, is used as a source term in Oracle3D in the algebraic equations and is explained at the end of this section. Expanding the 2nd term of LHS by writing the flux contribution from all 4 faces of the cell with node P, (Fig. 2.2), we get:

$$\left(\frac{\Delta V}{\lambda_D^2}\right)\varphi_P^m - \left[\frac{\varepsilon_{re}S_e}{d(P,E)}\left(\varphi_E^m - \varphi_P^m\right) - \frac{\varepsilon_{rw}S_w}{d(P,W)}\left(\varphi_P^m - \varphi_W^m\right) + \frac{\varepsilon_{rn}S_n}{d(P,N)}\left(\varphi_N^m - \varphi_P^m\right) - \frac{\varepsilon_{rs}S_s}{d(P,S)}\left(\varphi_P^m - \varphi_S^m\right)\right] = 0 \quad (2.16)$$

In eq. (2.16) we only have the terms at current time step (m), so we drop the superscript 'm' and rearrange the terms for node P and neighbouring nodes respectively as:

$$\left[\frac{\Delta V}{\lambda_D^2} + \frac{\varepsilon_{re}S_e}{d(P,E)} + \frac{\varepsilon_{rw}S_w}{d(P,W)} + \frac{\varepsilon_{rn}S_n}{d(P,N)} + \frac{\varepsilon_{rs}S_s}{d(P,S)}\right]\varphi_P - \left(\frac{\varepsilon_{re}S_e}{d(P,E)}\right)\varphi_E - \left(\frac{\varepsilon_{rw}S_w}{d(P,W)}\right)\varphi_W - \left(\frac{\varepsilon_{rn}S_n}{d(P,N)}\right)\varphi_N - \left(\frac{\varepsilon_{rs}S_s}{d(P,S)}\right)\varphi_S = 0 \quad (2.17)$$

$$A_P\varphi_P + A_E\,\varphi_E + A_W\varphi_W + A_N\varphi_N + A_S\varphi_S = 0 \quad (2.18)$$

Eq. (2.18) is the final algebraic equation for cell P and here the A coefficients are called the discretization coefficients, from corresponding neighbouring control volumes, of the equation. Their values are:

$$A_E = -\frac{\varepsilon_{re}S_e}{d(P,E)}, \qquad A_W = -\frac{\varepsilon_{rw}S_w}{d(P,W)}$$

$$A_N = -\frac{\varepsilon_{rn}S_n}{d(P,N)}, \qquad A_S = -\frac{\varepsilon_{rn}S_n}{d(P,S)}$$

A_P is the coefficient of the control volume under consideration itself, and it is given by eq. (2.19).

$$A_P = -A_E - A_W - A_N - A_S + S_P \quad (2.19)$$

Here $S_P = \frac{\Delta V}{\lambda_D^2}$, it comes from the RHS term of the base eq. (2.6). In Oracle3D, we treat such contributions, which come from other than the neighbouring nodes, in the S_P term. We do not have any source term in the equation (2.6) so here SU = 0, which is the variable used in Oracle3D for the contribution from the source/sink terms.

For a three-dimensional case we will have the terms for top and bottom node contributions also, and the final discretized algebraic equation for node P will become:

$$A_P\varphi_P + A_E\,\varphi_E + A_W\varphi_W + A_N\varphi_N + A_S\varphi_S + A_T\varphi_T + A_B\varphi_B = 0 \qquad (2.20)$$

For diffusion coefficient (ε_r) we use the linear interpolation, where λ_e is the interpolation factor based on the ratios of distances between the cell centroids:

$$\varepsilon_{re} = \varepsilon_P(1-\lambda_e) + \varepsilon_E\lambda_e$$

$$\lambda_e = \frac{d_{Pe}}{d_{Pe} + d_{Ee}}$$

Here *e* stands for the central node on the interface between the two cells, and $d_{Pe},\ d_{Ee}$ refer to the corresponding distances between the cell centroids and the node *e*. The deferred correction terms, which were dropped in eq. (2.16), are approximated and used as source terms in algebraic equations. We write here the RHS term of eq. (2.15) and expand it in two terms as follows:

$$\sum_{k=newsbt} \varepsilon_{rk} S_k \left(\nabla\varphi\right)_k^{m-1} .\left(\vec{n}_k - \vec{\xi}_k\right) = \left[\sum_{k=newsbt} \varepsilon_{rk} S_k \left(\nabla\varphi\right)_k^{m-1} .\,\vec{n}_k - \sum_{k=newsbt} \varepsilon_{rk} S_k \left(\nabla\varphi\right)_k^{m-1} .\,\vec{\xi}_k\right] \qquad (2.21)$$

These two terms on the RHS of eq. (2.21) are stored in variables SUEH and SUEL in Oracle3D. Contributions in these terms from all the cell faces can be obtained similarly as shown above for the LHS of eq. (2.15). Some code relevant details corresponding these terms are provided in appendix I with further explanations. The final algebraic equation corresponding to eq. (2.6) for a control volume cell as used in Orcale3D can be written as:

$$A_P\varphi_P + \sum_{k=newsbt} A_k\varphi_k = SUEH - SUEL \qquad (2.22)$$

2.2.2 The Improved Deferred Correction (IDC)

Based on the idea of SDC scheme, Traore et al. (2009) introduced an improvement in the approximation of diffusive flux for non-orthogonal grids which was tested with extremely skewed grids and proved to be robust with better efficiency than the SDC scheme [8,9]. It was also reported that the convergence properties and the order of accuracy of the discretization were not degraded with this improved deferred correction (IDC) scheme even in extremely skewed grids. The decomposition of surface normal vector in IDC is shown in Fig. 2.4. The main idea is to express the surface normal ($\vec{n}_k$) in term of a unit vector parallel to the surface S_k. The surface normal ($\vec{n}_k$) in this case can be written as:

$$\vec{n}_k = \lambda_k\overrightarrow{\xi_k} + \beta_k\overrightarrow{t_k}$$

Where $\lambda_k = \frac{1}{\cos\theta_k}$; and $\beta_k = \tan\theta_k$. This updated value of $\vec{n}_k$ should be used in eq. (2.11) to obtain the new approximation of diffusive flux with IDC scheme. The derivation with IDC will follow in similar manner as explained above for the SDC scheme. This IDC scheme was implemented in parallel Oracle3D code and the same tests cases, as provided in Traore et al. (2009), were performed to validate the Poisson solver of the code. The detailed validation cases having the comparison with reference [8] are provided in chapter 4 of this thesis. For further details on the IDC scheme interested readers should refer [8,9].

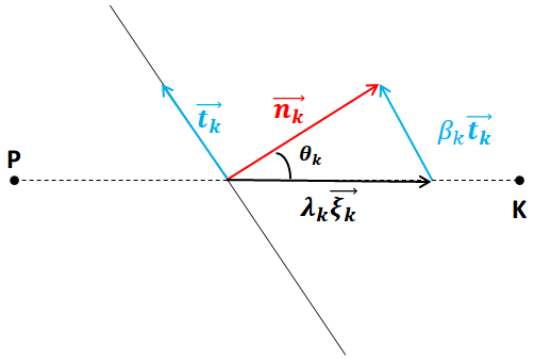


Fig. 2.4 Decomposition of surface normal vector in IDC scheme (Traore et al. (2009))

2.3 Discretization of convective term

We take the integral form of convective term from eq. (2.5) and write the integral of surface as the sum of fluxes from the control volume faces in all directions. We introduce the mass flux ($\dot{m}$) variable here which is $\dot{m}_k = \rho_k \vec{u}_k \vec{n}_k S_k$. This represents the mass flux which pass through the surface S_k.

$$\int_{\partial V_c} (\rho\varphi\vec{u}).d\vec{S} = \int_{\partial V_c} (\rho\varphi\vec{u}).\vec{n}\, dS = \sum_{k=newsbt} (\rho\varphi\vec{u})_k . \vec{n}\, S_k = \sum_{k=newsbt} \dot{m}_k\, \varphi_k \qquad (2.23)$$

Now, we need to find the approximations for velocity field ($\vec{u}_k$) and the scalar variable (φ_k) through the control volume faces. A simple linear interpolation for the velocity field is considered in our solver which takes the form:

$$\vec{u}_k = \alpha\, \vec{u}_P + (1-\alpha)\, \vec{u}_K$$

Where the interpolation factor is defined as

$$\alpha = \frac{d(K,k)}{d(K,P)}$$

So, the final expression for the mass flux becomes:

$$\dot{m}_k = \rho_k \vec{u}_k \vec{n}_k S_k = \rho_k (\alpha\, \vec{u}_P + (1-\alpha)\vec{u}_K)\vec{n}_k S_k \qquad (2.24)$$

There are several schemes available in literature to approximate the value of (φ_k), on the centre of faces which are common between two control volumes. Central Differencing Scheme (CDS) is the simplest and used quite often to approximate the convective terms. With CDS (φ_k) takes the form:

$$\varphi_k \approx \alpha\, \varphi_P + (1-\alpha)\ \varphi_K$$

And the total convective flux with CDS scheme ($Flux_k^{C,CDS}$) from face k becomes:

$$Flux_k^{C,CDS} = \dot{m}_k\, \varphi_k = \dot{m}_k\, (\alpha\, \varphi_P + (1-\alpha)\, \varphi_K) \qquad (2.25)$$

When used in a transport problem with dominant diffusive nature, the CDS approximation of convective flux works well and gives physical results. But, as the convective nature of the problem becomes more dominating, than diffusion, then with CDS scheme unphysical results are very likely. The linear profile with CDS scheme considers equal weight on both upwind and downwind nodes. However, the convective phenomena are strictly flow direction dependent, and the contribution from the upwind nodes will be dominant in strong convective flows. By analysing the discretization further in uniform grids, it is observed that if the cell Peclet number ($P_e = \rho\, u\, \Delta x / \Gamma^{\varphi}$) is larger than 2 then unphysical results are obtained with the CDS [1,3]. Peclet number is a dimensionless number which is usually defined as the ratio of rates of advection and diffusion of a physical quantity.

Thus, in place of CDS the Upwind Difference (UD) scheme is better suited for the convective fluxes, which approximates φ for the east face as:

$$\varphi_e = \begin{cases} \varphi_P & if\ \dot{m}_e \geq 0 \\ \varphi_E & if\ \dot{m}_e < 0 \end{cases}$$

The Upwind scheme mimics the underlying physics of advective flows better, as the face values of the fluxes are made dependent on the upwind node values, which makes the flux more tuned with the flow direction than the CDS approach. The Upwind scheme is only 1st order accurate but it gives bounded solutions even for higher values of Peclet numbers. The CDS being 2nd order accurate gives better results than Upwind, for flows with $P_e < 2$, but the solutions are unphysical for higher values of P_e numbers. Thus, numerical analysis of convective flows wanders between the issues of accuracy and stability with these two schemes and leads the way forward for the need of more robust and accurate schemes.

2.4.1 TVD schemes

Upwind differencing scheme is the most stable and unconditionally bounded scheme; however, it introduces a high level of false diffusion (numerical diffusion) because it is only 1st order accurate [1,7]. Higher order schemes like Central differencing, hybrid and QUICK schemes are

more accurate but they can give spurious oscillations (wiggles) in the form of undershoots and overshoots when the Peclet number is high. These wiggles can lead to physically unrealistic values and make the solution unstable. To address this undesirable feature of numerical oscillations in these schemes, High Resolution (HR) schemes are formulated. HR schemes are formulated to preserve the convective biased nature of previous schemes with improving the boundedness criterion.

Several HR schemes have been developed till date and a good review of all these schemes is available in [1,3]. The HR schemes that were developed in the Total Variation Diminishing (TVD) framework make the TVD class of convective schemes. TVD schemes are especially developed to counter the spurious oscillations by adding artificial diffusion or weighting towards upstream contribution in discretized equations. Mathematically the Total Variation is defined as:

$$TV = \sum_{i} |\varphi_{i+1} - \varphi_i| \tag{2.26}$$

Here i is the index of the control volume node in the spatially discretized domain. A scheme is said to satisfy the TVD nature if the total variation (TV) in the solution does not increase with time, which is written as:

$$TV\ (\varphi^{t+\Delta t}) \leq TV\ (\varphi^{t}) \tag{2.27}$$

Pioneering work on TVD schemes by Harten (1983) and Sweby (1984) led to the TVD class of HR convective schemes [5,6]. Harten proved in his work on HR schemes that a monotone scheme is TVD, and a TVD scheme is monocity preserving [6]. If the value of a local minimum does not decrease and the value of a local maximum does not increase in the solution domain, then the scheme is said to be monotonicity preserving [3]. In other words, monotonicity preserving schemes will not produce overshoots and undershoots in the solution domain. Several formulations were developed by different researchers which toady come under the TVD class of schemes. The implementation of TVD schemes in Orcale3D follows upwind biased formulation [7]. TVD schemes with different flux limiter options are available for the convective part of the transported variables.

2.4.2 Implementation of TVD scheme in Oracle3D

This section explains the general TVD formulation in the specific way it is implemented in Oracle3D. Some details are intended for the users/developers of the code alone. We will talk about some variables which are used in code as they are described here. In 3D problems, each control volume has 6 contributions of fluxes from each of the 6 faces (east, west, north, south, top and bottom). However, for any two adjacent cells the flux through the common face is same in magnitude but opposite in direction (for one cell the flux goes out from the common face and the same flux enters into the adjacent cell). This leads us to reduce some computational cost, by calculating the common flux from a face only for one cell and using the same flux with a negative sign for the adjacent cell for the other direction.

Thus, we scan the control volume faces only in 3 directions (east, north and top) for computing the fluxes and coefficients in Oracle3D. The contribution from the other sides (west, south and bottom) is taken from the adjacent cell values as explained above. For example, the subroutines CELQ is called 3 times in Transport subroutine for the three directions – east, north and top.

The contributions of the common faces for the adjacent cells are also calculated at the end of this routine.

SU(INP) = SU(INP) + (SUEH-SUEL)
SU(INE) = SU(INE) - (SUEH-SUEL)

Here INP is the index of a cell and INE is the index of the adjacent cell in the positive direction. SU(INP) is the variable for source term contributions for a cell, and SU(INE) will make the west, south and bottom source terms for the respective adjacent cells. The TVD schemes have their contribution in the source term of convective part, in the discretization used in Oracle3D. The TVD contributions to source terms (SU) are treated similarly as the deferred corrections for the diffusion term, so the related terms are placed on the right-hand side of the equations.

To understand the basic aspect of TVD schemes, let us consider a one-dimensional flow in positive x direction (u > 0). We take ϕ as a general convecting scalar variable here. In Fig. 2.5, with the Upwind scheme we will get the east face value of ϕ as

$$\phi_e = \phi_P$$

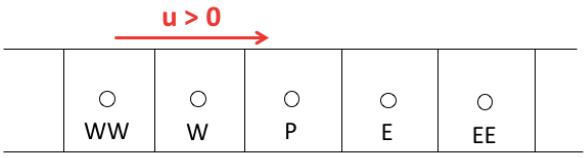


Fig. 2.5 1D control volumes arrangement with positive flow velocity

Now, as explained above the Upwind scheme is very stable but only 1st order accurate. Let us add an additional flux contribution in ϕ_e to make it higher order accurate by incorporating two upwind node values. We can add a linear upwind biased profile such as:

$$\phi_e = \phi_P + \frac{1}{2}(\phi_P - \phi_W)$$

This additional term makes the updated formulation a second order accurate scheme, which is called as a linear upwind differencing (LUD) scheme [7]. Similarly, other schemes like CDS, QUICK etc. can be updated by incorporating upwind biased terms [7]. Based on this idea, a general notation for an upwind biased ϕ_e , within a convective discretization scheme can be given as

$$\phi_e = \phi_P + \frac{1}{2}\Psi(r)(\phi_E - \phi_P) \qquad (2.28)$$

Here, we approximate ϕ_e with the value on the previous upwind node (ϕ_P) and an additional term. With $\Psi(r)$ being an appropriate limiter function of r to provide the required features of TVD in the scheme, and, r is the ratio of upwind-side gradient to downwind-side gradient of corresponding transported quantity, which is given by

$$r = \left(\frac{\phi_P - \phi_W}{\phi_E - \phi_P}\right)$$

With the introduction of eq. (2.28), the task of developing a TVD scheme is reduced to simply finding a limiter function ($\Psi(r)$), for an existing scheme to make it TVD or monotone [7].

Several flux limiters are developed with this idea and further details are available in [1, 7]. We focus in the following part the implementation method of TVD schemes for our solver, especially for the boundary nodes. Consider Fig. 2.6 to understand a more general (u > 0 and u < 0) notation of convective flux contribution in the overall flux (for 1D) by the TVD schemes as written by eq. (2.29):

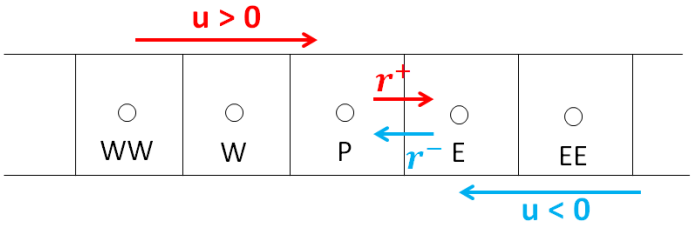


Fig. 2.6 1D domain showing both flow directions with corresponding r notations for the east face of node P

$$S_u^{TVD} = \frac{1}{2}F_e[(1-\alpha_e)\Psi(r_e^-) - \alpha_e.\Psi(r_e^+)].(\phi_E - \phi_P) + \frac{1}{2}F_w[\alpha_w.\Psi(r_w^+) - (1-\alpha_w).\Psi(r_w^-)](\phi_P - \phi_W) \quad (2.29)$$

The value of α_e depends on flow direction in the cell, α_e= 0 for u < 0 and α_e= 1 for u > 0. The mass fluxes ($\dot{m}_k$) through cell faces are denoted by F_k. Notations r_e^+ and r_e^- correspond to the variable r when the flow is in positive or negative x direction respectively. Here the source term contribution by TVD convective fluxes, for node P, from both the sides (east and west faces) is given in eq. (2.29). However, as explained above, in Oracle3D only the east side source term contributions are exclusively computed. The west side for this case will be taken as the negative of east side of adjacent cell. So, the equation we consider in Oracle3D becomes only:

$$S_u^{TVD} = \frac{1}{2}F_e[(1-\alpha_e)\Psi(r_e^-) - \alpha_e.\Psi(r_e^+)](\phi_E - \phi_P) \quad (2.30)$$

According to the direction of the flow we will have either the r_e^- or the r_e^+ part of eq. (2.30), as shown in Fig. 2.6. Moreover, for the north and top directions also we use the same notations of the variables in the code, just the corresponding directions are changed with the loop limits (I, J, K).

$$r_e^+ = \left(\frac{\phi_P - \phi_W}{\phi_E - \phi_P}\right), \quad r_e^- = \left(\frac{\phi_{EE} - \phi_E}{\phi_E - \phi_P}\right)$$

Now, with eq. (2.30) we compute the TVD source terms for the cells, for east, north and top directions in full 3D problem, which is done in CELQ subroutine in code. The contributions from the west, south and bottom faces are obtained by the adjacent cell flux values, as explained above. The loops for scanning the required faces in all 3 directions start with I=J=K=2 and ends with NI-2, NJ-2 and NK-2 respectively. These loops compute the TVD flux contribution for all the cells (for all 6 faces) except the contributions from the boundary side faces of the boundary cells. The contribution from the boundary side faces, are computed in the MODQ subroutine.

Here we should note that for the positive flow direction (u > 0) we work with r_e^+ factor and the value of ϕ_W for the boundary cells in west, south and bottom directions is not known to us. In Fig. 2.7 P is the index of boundary cell centroid and W is the node lying on the west boundary face itself. In all such cases the value of ϕ at the boundary face itself (FI(INE), as set in SETBC subroutine) is considered as ϕ_W in Oracle3D.

Similarly, when the flow is in negative direction (u < 0), we use r_e^-, then the value of ϕ_{EE} is not known to us for the last cells of the scanning loop in east (NI-2), north(NJ-2) and top(NK-2) directions. In this case we take the boundary value of ϕ to be the ϕ_{EE} for these cells in east, north and top directions (Fig. 2.8).

Flux from the Boundary faces

- **CASE 1.1** - For u > 0 at the west boundary cell node 'P' (I=2), Fig. 2.7, we need the contribution of fluxes from the 2 faces (east and west) of the cell. For this boundary cell we have the contribution of east face already from the previous section (as explained above) which is computed with eq. (2.30). Now we only need the contribution from the west side face. Being at the boundary, this west side face of the west direction boundary cell does not contribute to the TVD source terms. The treatment with eq. (2.30) is not required here, as we do not need to approximate the value of ϕ on this west face, we have the exact ϕ as the imposed boundary condition. Their contribution is taken directly from the value imposed on the boundary ($\phi_{WEST\_boundary}$).

  West face contribution to the source term = $F_w\ \phi_{WEST\_boundary}$

  East face contribution to source term = $\frac{1}{2}F_e[-\Psi(r_e^+)](\phi_E - \phi_P)$

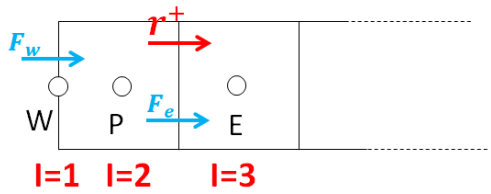


Fig. 2.7 Convective flux contribution to the west direction boundary node (P), u>0

- **CASE 1.2** - For u > 0 at the east boundary cell 'E' (NI-1), Fig. 2.8, there will be no TVD contribution from the east boundary face. It will be computed directly from the $\phi_{EAST\_boundary}$ imposed on that boundary face. The west face contribution comes from eq. (2.30) from the calculations of adjacent cell's (NI-2) east face.

  East face contribution to source term = $F_e\ \phi_{EAST\_boundary}$

  West face contribution to source term = $\frac{1}{2}F_w[\Psi(r_e^+)](\phi_E - \phi_P)$

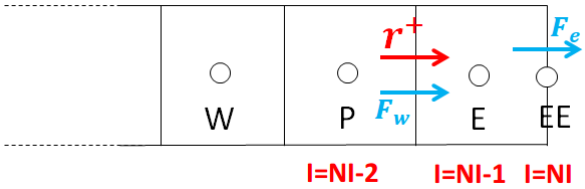


Fig. 2.8 Convective flux contribution to the east direction boundary node (E), u>0

- **CASE 2.1** - For u < 0 at the east boundary cell 'E'(NI-1), Fig. 2.9, the west face contribution comes from the adjacent cell's (NI-2) east contribution which is computed with eq. (2.30) in previous section. And, the east face contribution does not have any

TVD source contribution because this comes directly from the boundary value $[F_e\ \phi_{EAST\_boundary}]$.

East face contribution to source term = $F_e\ \phi_{EAST\_boundary}$

West face contribution to source term = $\frac{1}{2}F_w[-\Psi(r_e^-)](\phi_E - \phi_P)$

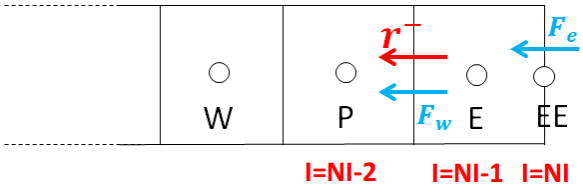


Fig. 2.9 Convective flux contribution to the east direction boundary node (E), u <0

- **CASE 2.2** - For u < 0 at the west boundary cell (I=2):

  West face contribution to source term = $F_w\ \phi_{WEST\_boundary}$

  East face contribution to source term = $\frac{1}{2}F_e[\Psi(r_e^-)](\phi_E - \phi_P)$

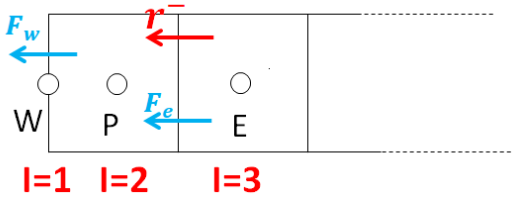


Fig. 2.10 Convective flux contribution to the west direction boundary node (P), u<0

Finally, the main points to note (especially for the users of the code) are:

1) there is no TVD contribution in the source terms from the boundary faces (boundary side faces of the boundary control volumes), e.g. the west faces of the west boundary cells and east faces of the east boundary cells are not treated in TVD manner.
2) all the rest of the faces of all the control volumes (cells) are treated in same TVD manner as explained with eq. (2.30).
3) Source term contributions and coefficients of discretization equations, from the common faces between two adjacent cells, are computed once only for the positive directions (east, north and top), e.g. in CELQ, CELUVW etc. subroutines. The contributions from the west, south and bottom faces come from the adjacent cell's flux contribution from that common face, with only a change of direction (negative sign).

2.4 Discretization of transient term

We write the integral form, for FVM, of the transport equation for a scalar φ, as previously shown in eq. (2.2)

$$\int_{V_c} \frac{\partial(\rho\varphi)}{\partial t} dV + \int_{V_c} \mathcal{L}(\varphi)\ dV = 0 \tag{2.31}$$

The second term in eq. (2.31) contains all the other terms (diffusive, convective, source etc.). We need to discretize now the transient term with FVM. We need an approximation for the derivative of $(\rho\varphi)$ with time, which is the done with Gear scheme for Oracle3D [3,4]. It corresponds to 2nd order accuracy for the first derivate and written as shown in eq. (2.32). Here n+1 depicts the value at current time step. Gear scheme is an implicit three-level scheme, which uses the variable values from two previous time steps.

$$\int_{V_c} \frac{\partial(\rho\varphi)}{\partial t} dV \approx \rho\left(\frac{3\varphi_P^{n+1} - 4\,\varphi_P^{n} + \varphi_P^{n-1}}{2\Delta t}\right)\Delta V \tag{2.32}$$

2.5 Combined formulation of Convection-Diffusion discretization

Discretization of both diffusion and convection terms are briefly explained in the previous sections. We revisit the combined equation of a steady convection-diffusion without any sources to provide a combined formulation for the final algebraic equation for individual control volumes. We recall eq. (2.5) without the source term and consider a uniform 1D grid as shown in Fig. 2.11.

$$\int_{\partial V_c} (\rho\varphi\vec{u}).d\vec{S} = \int_{V_c} (\Gamma^{\varphi}\nabla\varphi).d\vec{S} \tag{2.33}$$

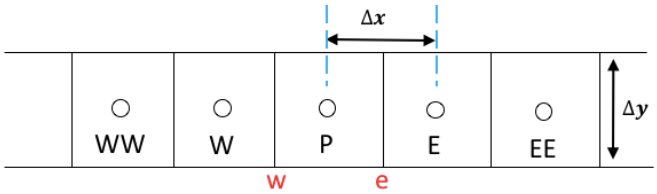


Fig. 2.11 A 1D grid arrangement

This can be simplified in term of summation of fluxes from all the faces of a control volume (here in 1D case just west and east).

$$\sum_{k=ew} \left(\dot{m}\varphi - \Gamma^{\varphi} S \frac{d\varphi}{dx} \right)_k = 0 \tag{2.34}$$

Let us expand the terms in eq. (2.34) for the control volume with center node P, we note that the surface vectors on the opposite sides (east and west) of the control volume have opposite signs. In Oracle3D, the effect of surface vector sign for the convective fluxes are already considered in mass flux ($\dot{m}_k = \rho_k \vec{u}_k \vec{n}_k S_k$), but we need to consider the sign for the diffusion term. We consider the CDS scheme to approximate φ and 2nd order approximation for $\frac{d\varphi}{dx}$. We can write:

$$\frac{1}{2}\dot{m}_e (\varphi_P + \varphi_E) + \frac{1}{2}\dot{m}_w (\varphi_P + \varphi_W) - \left(\Gamma_e^{\varphi} S_e \frac{\varphi_E - \varphi_P}{\Delta x} - \Gamma_w^{\varphi} S_w \frac{\varphi_P - \varphi_W}{\Delta x} \right) = 0$$

$$\frac{1}{2}\dot{m}_e \varphi_E + \frac{1}{2}\dot{m}_w \varphi_W + \left(\frac{1}{2}\dot{m}_e + \frac{1}{2}\dot{m}_w \right) \varphi_P - D_e \varphi_E - D_w \varphi_W + (D_e + D_w)\varphi_P = 0$$

Where $D_k = \frac{\Gamma_k^{\varphi} S_k}{(\Delta x)_k}$, is termed as the diffusion coefficient for the algebraic equation from face k. We collect together the contributions from individual cells.

$$\left(\frac{\dot{m}_e}{2} - D_e \right) \varphi_E + \left(\frac{\dot{m}_w}{2} - D_w \right) \varphi_W + \left(\frac{\dot{m}_e}{2} + \frac{\dot{m}_w}{2} + D_e + D_w \right) \varphi_P = 0. \tag{2.35}$$

With the continuity equation, we can write

$$\dot{m}_e + \dot{m}_w = 0$$

Let us take A_P is the coefficient of φ_P in eq. (2.35) which can be rewritten by subtracting the continuity equation from it, as:

$$A_P = \frac{\dot{m}_e}{2} + \frac{\dot{m}_w}{2} + D_e + D_w - (\dot{m}_e + \dot{m}_w)$$

$$A_P = -\frac{\dot{m}_e}{2} - \frac{\dot{m}_w}{2} + D_e + D_w$$

Rewriting eq. (2.35) in the form of algebraic equation with discretization coefficients, we get

$$A_P \varphi_P + A_E \varphi_E + A_W \varphi_W = 0 \tag{2.36}$$

Here the A_k coefficients are called the discretization coefficients, from corresponding neighbouring control volumes, of the equation. Their values are:

$$A_E = \frac{\dot{m}_e}{2} - D_e, \qquad A_W = \frac{\dot{m}_w}{2} - D_w$$

$$A_P = -(A_E + A_W)$$

For the full 3D discretization, the algebraic equation for cell P will be written as eq. (2.37), and the discretization coefficient are obtained as shown above with the east and west contributions. This equation is valid for all the control volumes of the domain; however, the boundary side fluxes are treated differently which is described in next section with some examples.

$$A_N = \frac{\dot{m}_n}{2} - D_n\,, \qquad A_S = \frac{\dot{m}_s}{2} - D_s$$

$$A_B = \frac{\dot{m}_b}{2} - D_b\,, \qquad A_T = \frac{\dot{m}_t}{2} - D_t$$

$$A_P = -(A_E + A_W + A_N + A_S + A_B + A_T)$$

$$A_P\varphi_P + A_E\,\varphi_E + A_W\varphi_W + A_N\varphi_N + A_S\varphi_S + A_T\varphi_T + A_B\varphi_B = 0 \qquad (2.37)$$

2.6 Discretization of Boundary Conditions

Correct implementation of boundary conditions in any numerical solver is of utmost importance. In many physical problems, there are often some differing views of researchers regarding which numerical boundary conditions will represent the physical conditions more accurately. Sometimes, the numerical boundary conditions are over simplified or approximated to avoid certain numerical difficulties and complexities; keeping the physical accuracy at stake. During the course of this thesis, we implemented some new boundary conditions in Oracel3D which were of huge physical importance, and it was always noticed that the user should pay close attention while discretizing the boundary equations.

It is also very important that the user understand both the general discretization of the boundary equation and simultaneously know how they are really implemented in the code. In this section, we will discuss the discretization of some boundary conditions (sections 2.6.1-2.6.3) and finally we will provide a general approach (section 2.6.4) to work out all the boundary conditions for Oracle3D. The general approach as mentioned in section 2.6.4 can also be referred before going to sections 2.6.1-2.6.3, for a quick summary and having a different approach for implementing the boundary conditions.

2.6.1 Robin Boundary condition

Let us consider a 2D grid arrangement at a south direction boundary, Fig. 2.12. In this figure, the blue line is the boundary in the south direction, and the node S which is shown in red is the node on the boundary side face. The directions of surface normal vectors are shown at all four faces of the control volume with centre node P.

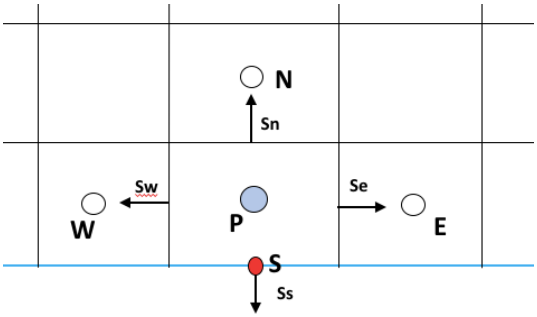


Fig. 2.12 A 1D grid arrangement with the south side boundary nodes

Setting the boundary values for all the variables is very important which can be provided by several formulations depending on the physics involved. One of the simplest forms of boundary conditions is termed as Dirichlet boundary condition, in which, a fixed value of the variable is given on the boundary node S. In zero flux Neumann boundary condition zero gradient of the variable is provided at the boundary node S. Robin boundary conditions are a kind of mixed boundary conditions between Dirichlet and Neumann boundary conditions. They are generally provided as some partial differential equations which represent the physical phenomena at the boundary nodes. These equations are then discretized by FVM to get the flux contribution from the boundary faces. Here, we take an example Robin boundary condition for electric charge density variable (φ) which was encountered in the electro-conduction problems as implemented in Oracle3D. The Robin BC is given as follows:

$$\left(K\vec{E}\varphi - \Gamma \nabla\varphi \right).\vec{n} = 0 \tag{2.38}$$

Here K is the mobility of ions, $\vec{E}$ is the electric field vector, Γ is the diffusion coefficient and $\vec{n}$ is the unit normal vector to the boundary surface. In this case, the surface vector for the boundary face points in the negative y direction as shown in Fig. 2.12. Now, this equation is valid only for the boundary node S. We discretize this equation with FVM as:

$$\int_V \left(K\vec{E}\varphi - \Gamma \nabla\varphi \right).\vec{n}\, dV = 0$$

$$\left(KE_y\varphi - \Gamma \nabla\varphi \right)_b = 0$$

$$(KE_y\varphi)_b - \Gamma_b \left(\frac{\partial\varphi}{\partial y}\right)_b = 0$$

Here ‘b’ implies that these expressions are to be evaluated at the boundary. The gradient is approximated with downwind scheme, where d(P, S) is the distance between node P and S.

$$(KE_y)_b\varphi_S - \Gamma_b \frac{\varphi_P - \varphi_S}{d(P,S)} = 0$$

$$\left(\left(KE_y\right)_b + \frac{\Gamma_b}{d(P,S)}\right)\varphi_S - \frac{\Gamma_b}{d(P,S)}\,\varphi_P = 0$$

$$\varphi_S = \frac{D}{C+D}\,\varphi_P \tag{2.39}$$

$$\varphi_S = \beta\,\varphi_P \tag{2.40}$$

Where $\beta = \frac{D}{C+D}$, $D = \frac{\Gamma_b}{d(P,S)}$ and C = $\left(KE_y\right)_b$. Equation (2.39) gives the value of variable φ at the boundary node S, in terms of the central node P. It is very important to understand how finally this value of φ_S is utilized, with other information, to implement the overall boundary condition (eq. (2.38)) in the code. We recall the algebraic equation for cell P, eq. (2.37), and write it for a 2D grid as:

$$A_P\varphi_P + A_E\,\varphi_E + A_W\varphi_W + A_N\varphi_N + A_S\varphi_S = 0$$

When this equation is written for the control volumes which are not at the boundary then A_P is given by:

$$A_P = -(\,A_E + A_W + A_N + A_S\,)$$

However, when we are dealing with boundary cells, the flux contribution from the boundary face is different from other faces. The fluxes from the boundaries are not obtained with CDS and central schemes approximations, as they are computed from the given boundary conditions directly. For example, the south boundary face coefficient A_s will be given by $A_S = \dot{m}_s - D_s$. The A_S coefficient for boundary nodes is defined as 'ADC' in Oracle3D. Thus, in general derivation for the boundary cells the A_P coefficient becomes:

$$A_P = -(\,A_E + A_W + A_N + ADC\,) \tag{2.41}$$

However, In Oracle3D, the flux contribution from the boundary nodes are computed separately within S_P term which is later added in A_P term to complete the coefficient values. So, in Oracle3D the A_P coefficient, for the boundary cell, before adding the boundary face contribution is written only as

$$A_P = -(\,A_E + A_W + A_N)$$

We rewrite the algebraic equation using the φ_S for this case from eq. (2.40), we get

$$A_P\varphi_P + A_E\,\varphi_E + A_W\varphi_W + A_N\varphi_N + A_S\,\beta\,\varphi_P = 0$$

$$(A_P + A_S\,\beta)\,\varphi_P + A_E\,\varphi_E + A_W\varphi_W + A_N\varphi_N = 0 \tag{2.42}$$

From eq. (2.42), it is clear that the coefficient of φ_P has an extra contribution other than A_P, as given by $A_S\,\beta$, which comes from the boundary node S. And, also the value of A_P does not have '$-A_S$' included in it as per the general notation of A_P term. So, these two contributions are provided within the S_P term in Oracle3D. In this specific case the S_P term will be:

$$S_P = -A_S + A_S\,\beta = A_S\,(\beta - 1) \tag{2.43}$$

2.6.2 Non-Homogeneous Neumann boundary condition

Electro-conduction model as implemented in Oracle3D also provides a non-homogeneous Neumann boundary condition for electric potential on dielectric substrate. The boundary condition is given as:

$$\nabla\varphi.\vec{n} = \sigma \tag{2.44}$$

We discretize eq. (2.44) with FVM, as explained above.

$$\int_V \nabla\varphi.\vec{n}\ dV = \int_V \sigma\ dV$$

We consider this BC at the south face, as shown in Fig. 2.12, thus the unit normal vector points in – y direction. Here σ is known as the surface charge density variable, accumulated over the substrate. We integrate the equation and solve further.

$$-\left(\frac{\partial\varphi}{\partial y}\right)_b V_p = \sigma\, V_p$$

Taking downwind scheme to approximate the gradient of φ on boundary node as:

$$-\frac{\varphi_P - \varphi_S}{d(P,S)} = \sigma$$

$$\varphi_S = \varphi_P + \sigma\, d(P,S) \tag{2.45}$$

Eq. (2.45) gives the electric potential value on the boundary node S when eq. (2.44) is set as the boundary condition for the potential variable. The algebraic equation for a boundary cell for this BC is written as eq. (2.46). The electric potential is governed by a Poisson equation, and in this case the S_P variable for the node P is obtained as zero.

$$(A_P + A_S)\varphi_P + A_E\,\varphi_E + A_W\varphi_W + A_N\varphi_N = -\ A_S\,\sigma\, d(P,S) \tag{2.46}$$

2.6.3 Dirichlet and zero flux boundary conditions

As briefed above, a Dirichlet boundary condition refers to a fixed (specified) value of φ on the boundary, given as

$$\varphi_S = \varphi_{specified} \tag{2.47}$$

The algebraic equation for a boundary cell for this BC is written as eq. (2.48). The S_P value for this case as computed with above mentioned procedure comes out to be $S_P = -A_S$.

$$A_P\,\varphi_P + A_E\,\varphi_E + A_W\varphi_W + A_N\varphi_N + A_S\,\varphi_{specified} = 0 \tag{2.48}$$

A zero flux Neumann boundary condition is set by setting the gradient of the variable zero on the boundary node, such as

$$\nabla\varphi.\vec{n} = 0$$

With FVM this boundary condition gives

$$\varphi_S = \varphi_P \tag{2.49}$$

The algebraic equation for this BC is eq. (2.50), and the corresponding S_P is zero.

$$(A_P + A_S)\varphi_P + A_E \varphi_E + A_W \varphi_W + A_N \varphi_N = 0 \tag{2.50}$$

2.6.4 A generalized approach to discretize boundary conditions

Let us take a generalised procedure to summarise the implementation of above mentioned boundary conditions in Oracle3D. We take a general boundary condition given as

$$a\,\varphi_b + b\left(\frac{\partial \varphi}{\partial n}\right)_b = c \tag{2.51}$$

We discretized this eq. (2.51) with FVM taking 1st order scheme for gradient approximation

$$a\,\varphi_b + b\,\frac{\varphi_P - \varphi_S}{d(P,S)} = c \tag{2.52}$$

In eq. (2.52), we have not considered the direction of the surface vector, for now, let's say that we keep the sign of corresponding direction of surface vector in distance variable d(P, S). We rearrange eq. (2.52) to get the boundary value of φ

$$\varphi_S = \frac{cd - b\varphi_P}{ad - b}$$

$$\varphi_S = \left(\frac{cd}{ad - b}\right) - \left(\frac{b}{ad - b}\right)\varphi_P \tag{2.53}$$

Let us say that in our problem the final algebraic equation for a control volume at boundary is given by:

$$A_P \varphi_P + A_E \varphi_E + A_W \varphi_W + A_N \varphi_N + A_S \varphi_S = SU \tag{2.54}$$

Here we have SU as the contribution from source terms, as used in Oracle3D. If the boundary value of φ is also given in the form of φ_P as:

$$\varphi_S = A + B\,\varphi_P \tag{2.55}$$

Then we use eq. (2.55) in (2.54) to obtain

$$(A_P + A_S B)\varphi_P + A_E \varphi_E + A_W \varphi_W + A_N \varphi_N = SU - A_S A \tag{2.56}$$

From eq. (2.56) it is observed that

$$S_P = A_S B$$

$$S_U = -A_S A$$

As, it was explained above, with eqs. (2.41 – 2.43) that in Oracle3D the algebraic equations are not used as exactly given by eq. (2.56). We recall eq. (2.43), which gives the value for S_P when implementing boundary conditions in Oracle3D.

$$S_P = -A_S + A_S\, B \tag{2.57}$$

Let us discuss the above-mentioned boundary conditions with this generalized approach to obtain quickly the S_P, S_U and φ_S values for Oracle3D.

1. **Dirichlet BC :**

$$\varphi_S = \varphi_{specified}$$

After comparing with eq. (2.55); A = $\varphi_{specified}$, B = 0

$$S_P = -A_S$$

$$S_U = -A_S\, \varphi_{specified}$$

2. **Neumann BC (zero gradient):**

$$\frac{\partial \varphi}{\partial n} = 0$$

After comparing with eq. (2.51)

a = 0, b =1, c= 0

$$\varphi_S = \varphi_P$$

comparing with eq. (2.55): A = 0, B = 1

$$S_P = -A_S + A_S = 0$$

$$S_U = -A_S A = 0$$

3. **Non-Homogeneous Neumann BC:**

$$-\left(\frac{\partial \varphi}{\partial y}\right)_b = \sigma$$

After comparing with eq. (2.51)

a = 0, b = -1, c= σ

$$\varphi_S = \sigma\, d + \varphi_P$$

comparing with eq. (2.55): A = $\sigma\, d$, B = 1

$$S_P = -A_S + A_S B = 0$$

$$S_U = -A_S\, \sigma\, d$$

4. **Robin BC:**

$$\left(K\vec{E}\varphi - \Gamma \nabla\varphi \right).\vec{n} = 0$$

$$(KE_y\varphi)_b - \Gamma_b \left(\frac{\partial\varphi}{\partial y}\right)_b = 0$$

After comparing with eq. (2.51)

$$a = (KE_y)_b \text{ , } b = -\Gamma_b, \; c = 0$$

$$\varphi_S = \frac{\Gamma_b}{(KE_y d)_b + \Gamma_b}\,\varphi_P$$

comparing with eq. (2.55): $A = 0$, $B = \frac{\Gamma_b}{(KE_y d)_b + \Gamma_b}$

$$S_P = -A_S + A_S B = A_S\,(B - 1)$$

$$S_U = -A_S\,A = 0$$

In this section, we have described a general approach based on a generalized equation for boundary conditions to discretize boundary condition equations with FVM. With this general approach we obtained the corresponding S_P, S_U and φ_S variable for the respective BC. The values obtained with this generalized approach and the discretizing approach for individual boundary conditions reach the same solutions.

2.6.5 Periodic Boundary condition

Periodic boundary conditions are used to deal with different type of symmetries (geometrical, physical etc.) in the problem domain [7]. Usually, for setting the periodic boundaries the values of the variables exiting the outlet-periodic plane are equal to the variables entering the inlet-periodic plane.

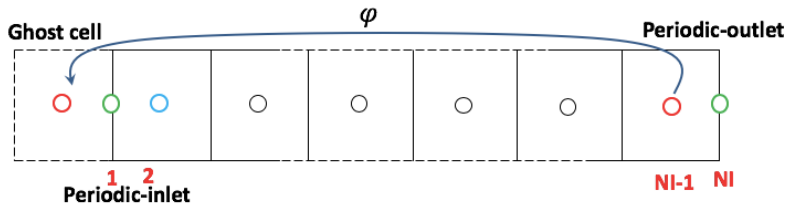


Fig. 2.13 Location of cells at inlet and outlet periodic boundaries

In Fig. 2.13, a 1D grid arrangement shows the location of control volume cells at periodic-inlet and periodic-outlet boundary sides. The green coloured node is situated on the face of the boundaries and blue and red are the cell centre nodes at inlet and outlet locations respectively. In our implementation of periodic boundaries, we have equated the node values on the boundary face nodes. To explain the steps, we first show transferring of the value of node at NI-1 (outlet location) to a ghost cell just ahead the inlet cell (I =2), as shown Fig. 2.13. With the values of inlet node (blue) and the outlet node (red) we compute the values at the inlet face center node (green), with CDS, eq. (2.58). After computing this inlet face node value (I = 1), this value is

also transferred to the outlet face node (NI). Thus, by making the boundary values (green) same we fulfil the conditions for the periodic boundaries.

$$\varphi(1) = \frac{\varphi(2) + \varphi(NI-1)}{2} \tag{2.58}$$

$$\varphi(NI) = \varphi(1) \tag{2.59}$$

After the φ values are computed with eqs. 2.58 and 2.59, the periodic BC on both sides are same as having Dirichlet BC, and thus, the coefficients are computed in same manner as done for Dirichlet BC.

$$S_P = -A_S$$

$$S_U = -A_S \, \varphi_{specified}$$

Bibliography

[1] F. Moukallel, L. Managani, M. Darwish, "The Finite Volume Method in Computational Fluid Dynamics," Springer International Publishing, Switzerland (2016)

[2] Y. B. Suzen, P. G. Huang, J. D. Jacob and D. E. Ashpis, "Numerical simulations of plasma-based flow control applications," AIAA Fluid Dynamics Conference, Toronto, Ontario Canada, 35, (2005)

[3] J. H. Ferziger, M. Peric, "Computational Methods for Fluid Dynamics", 3rd edition, Springer (2002)

[4] P. Traore, "Simulation numerique en mecanique des fluides et en dynamique des milieux granulaires. Modelisation des phenomenes Electro-Hydro-Dynamiques. L'Habilitation A Diriger des Recherches, Universite de Poitiers.

[5] A. Harten, "High Resolution Schemes for Hyperbolic Conservation Laws," J. Computational Phys., 49, 357-393 (1983)

[6] P. K. Sweby, "High Resolution Schemes using Flux Limiters for Hyperbolic Conservation Laws," SIAM Journal on Numerical Analysis, 21, 5, 995-1011 (1984)

[7] H. K. Versteeg, W. Malalasekara,"An Introduction to Computational Fluid Dynamics: The Finite Volume Method," 2nd edition, Pearson Education Limited, (2007)

[8] P. Traore, Y. M. Ahipo, C. Louste, "A robust and efficient finite volume scheme for the discretization of diffusive flux on extremely skewed meshed in complex geometries," Journal of Computational Physics, 228, 5148-5159 (2009)

[9] Y. M. Ahipo, P. Traore, "A robust iterative scheme for finite volume discretization of diffusive flux on highly skewed meshes," Journal of Computational and Applied Mathematics, 231, 478-491 (2009)

Chapter 3
Parallel Oracle3D with MPI

Highly scalable, parallel computer programs have become indispensable tools for the advancement of numerical research. Researchers are more hopeful than ever before, in tackling complex and huge engineering and scientific problems due to the availability of required computational resources in past twenty years. In terms of the hardware resources the progress seems well ahead of the progress in application (software) domain. There are many legacy codes which are still relevant in today, but they lack the modern approach to efficiently use the available hardware resources. Some of the computationally huge problems which cannot be solved without the modern high-performance computing (HPC) technologies include: weather forecasting, astrophysical analysis, plate tectonics analysis, turbulence modeling, plasma physics etc. It is unimaginable to work out these problems on single core computers, parallel computing is the only way forward for such problems.

In general, parallel computing refers to solving parts of a problem simultaneously on multi-core computing machines. A problem which can be broken into multiple smaller and discrete parts which can be solved independently, makes a good candidate for parallel computing. The discrete parts of the problem are solved on different computing cores and after finishing they are synchronized to provide the solution of the whole problem. Parallel computing offers several benefits to users: saving time and money, solving complex and large problems, multi-tasking etc. Advancements in HPC have provided another way – '*computational science'*, of doing science along with the classical branches of experimental and theoretical sciences. Computational scientists make use of their simulation methods when they are more advantageous and feasible over the classical approaches of theory and experiments.

Three broad areas of parallel computing are hardware, algorithms and software. In hardware, adding more and more cores and providing efficient inter-communication network among cores has increased the parallel nature of computing machines. In algorithmic terms, scientists seek how a problem can be defined by independent physical mechanisms, and, how it can be solved with independent set of mathematical equations. However, a bigger challenge is posed by the inadequate software, which are not fully able to profit by the progress made in hardware and algorithms. In terms of important characteristics, the modern codes should be optimized, portable and future-proof with every evolving HPC technologies. As shown in Fig. 3.1, a code should make optimal use of the hardware properties such as the cache design, vector registers, multiple cores etc. It should be developed with the standard parallel programming models such as MPI, OpenMP, Offloading etc.

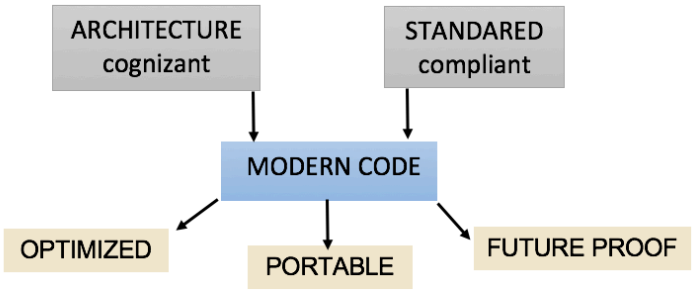


Fig. 3.1 Characteristics of a modern code

This chapter deals with the parallelization approach used for the in-house electrohydrodynamic (EHD) code- 'Oracle3D'. Oracle3D is a multi-block structured grid, finite volume solver. It is parallelized with message passing interface (MPI) 3.1 protocols. This chapter is mainly divided in three parts: 1) Parallel programming models, 2) MPI and single block grid parallelization, and 3) multi-block grid parallelization. We briefly introduce some parallel programming models and describe message passing model in first part. In the second part we describe in detail the MPI methodology used to parallelize the code for the single block grids only.

The message passing features of MPI library which are relevant to such approaches are provided. Some scalability results are provided to verify the efficiency of the approach. Third section deals with the extension of the approach for the multi-block grid cases, where some more advanced features of MPI have been used. The detailed implementation of this strategy based on the Cartesian topology and Inter communicators is put forth for the users of the code, and other researcher who are working with similar codes and wish to parallelize their code using MPI could also benefit with this detailed chapter on various MPI features.

Outline of this Chapter:

1. Overview of Parallel Programming
2. MPI and Single Block grid
3. MPI extension to Multi-block grids

3.1 Parallel Programming Models

We will discuss briefly about some parallel computational models to draw a background for the approach we used to parallelize our code. Parallel computational models can be informally classified on the basis of their memory utilization (shared or distributed), communication pattern, types of operations etc. Some of the parallel computational models are:

- **Data parallelism**

The notion of data parallelism model can be simply explained with vectorization technique of modern processors. It is based on the SIMD (single instruction, multiple data) framework, in

which an operation is performed on a set of array elements simultaneously by the special vector resistors (e.g. Intel AVX). It was one of the first ideas from where the whole idea of parallel computing started. There can be many instances in program where the processors operate on multiple data elements at the same time. The sole idea is to parallelize the data.

- **Shared memory**

Multiple computing cores sharing a common memory among them is a kind of *control parallelism* where data independence is not present directly, and the parallelism is explicitly performed by programmers. Most of the modern multicore processors are shared memory machines. Intel Xeon Phi coprocessors have up to 64 cores embedded together utilizing a common memory. These kinds of multicore processors are assembled together to build large supercomputing clusters of distributed memory (Fig. 3.1.1 (b)).

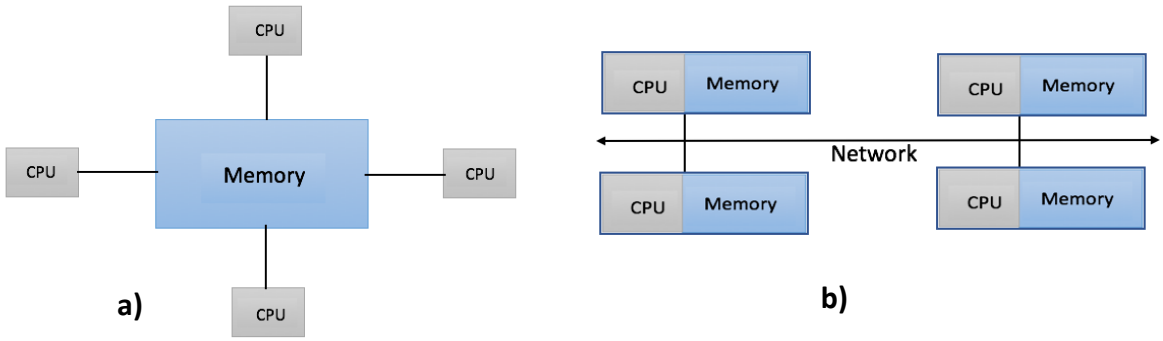


Fig. 3.1.1 a) Shared, and b) Distributed memory architecture/model

- **Message passing**

Message passing is a software concept, in which different processes transfer their data with other processes. Here, a process is an instance of a computer program which is being executed. In terms of hardware and software relation, a process is usually associated with an individual hardware core. Thus, in message passing parallel models processes work with their own local memory which is attached with their core, and they can communicate with other processes to access their data. The data of other processes is accessed by executing send and receive operations, which are performed by both the involved processes. Oracle3D is parallelized with message passing interface (MPI) which is a message passing model. This model supports the hardware parallelism where large number of processors (hardware) are used in distributed memory schemes, and data is transferred among distantly lying processors by communications done through the processes (software).

- **Remote memory operations**

On Cray T3E machine *put* and *get* data operations model was used, in which the remote memory of a processor is accessed by other processors only by one-way operation. The participation of both processors is not required. Some implementations use the 'active message' operations where a subroutine is executed in other processors memory. This active-message model provides the *remote memory copying* feature with just one-sided operations. This kind of parallel model is said to be lying half way between the shared memory and

message passing models. TMC CM-5 was the first machine to commercially popularize this model [1].

- **Threads**

OpenMP is the most well-known models of parallel computing which works with the threads model. It is a high-level programming approach where multiple threads are created dynamically during the execution of a program. Compiler directives are used to tell the code to create threads when they are needed. It is also a shared memory model, in which all the threads use the common memory with some kind of locking system while accessing common data simultaneously. POSIX Standard is another widely available thread model [1]. Modern processors technologies like Intel's Hyper-Threading can also be considered under threading parallel models where a physical core is used by multiple processing threads, making it work as multiple logical cores. Hyperthreading is also termed as simultaneous multithreading [1].

- **Hybrid models**

All of the world's largest computing systems work with a combination of above described parallel models. The supercomputing machines are hybrid at the level of hardware itself. They have several nodes of multiple processors connected by network cables, making it the distributed memory hardware at bigger level. And, at the same time the individual processors in each computing node have their memories shared among their member cores, Fig. 3.1.2. To exploit optimally such hybrid hardware systems, we must use hybrid programming models also.

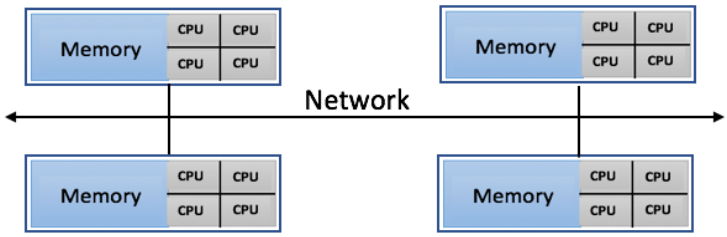


Fig. 3.1.2 Hybrid nature of hardware memory architectures

One example of hybrid programming model is to use MPI and OpenMP together on large machines. MPI uses the message passing model to communicate data at the distributed memory level (node level), and concurrently OpenMP is implemented to make use of shared memory in processors by providing multiple threads [3]. Additionally, when these models are doing their respective operations the data parallelism is in place in vector resistors, thus making the whole parallel computing really of hybrid nature. Fig. 3.1.3 illustrates a truly hybrid paradigm of software and hardware, where we have latest Intel Xeon Phi coprocessors networked with other host CPUs. In this figure MPI and OpenMP are used with the *'Offloading model'*, by which the executing program is dynamically offloaded to Xeon Phi coprocessors.

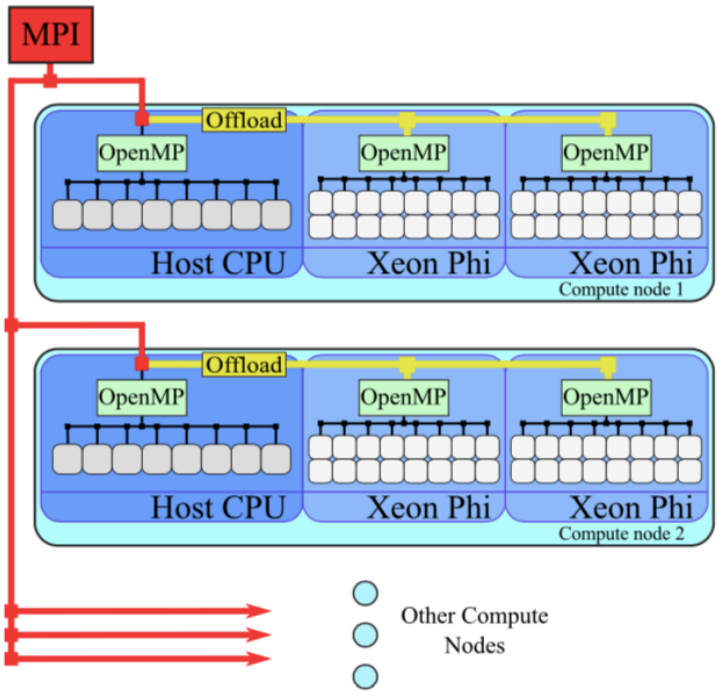


Fig. 3.1.3 Hybrid parallel programming on hybrid Intel hardware [3].

3.1.1 Message-Passing Model

This parallel programming model is based on message-passing among the hardware cores in multi-core architectures. In this model, the executing processes have local memories only and they send and receive data (messages) among other processes as required. The operation to transfer certain data from a process to another process is carried out by both the processes. One process sends data and another process receives the data. This method of two-way communication from two memory addresses defines the mechanism of message-passing parallel models.

Some Advantages of the Message-Passing model:

a) **Universality**: The message passing model is used at most of the places, be it the world's largest supercomputing systems or the work station networks, utilizing the available hardware capabilities.

b) **Expressivity**: Message passing is a complete model to express parallel algorithms. Message passing can be used for both shared and distributed memory architectures.

c) **Ease of debugging**: Several high-capability debuggers exist for message passing models. As these models control memory references more explicitly in comparison with other models, locating error of memory reads and write are easier than other models.

d) **Performance**: In distributed memory architectures, as the core count is increased by adding more processors to nodes or more cores to processors, the memory and cache is also increased with the number of cores. In these systems, memory bound applications can exhibit super-linear speedups. And, they are best exploited with message passing models only. Because of the performance gains the message passing models will remain permanent part of the parallel programming frameworks for long time in future.

3.2 MPI and Single block grid parallelization

This part of chapter deals with the features of MPI in general, and as they are implemented in Oracle3D for parallelizing the single block grids first. Mainly, the Cartesian topology features of MPI are discussed in detail, which were used to optimize the efficiency of parallel communications.

Problem description and objective

We briefly describe our problem before going into the MPI solutions in following sections. Fig. 3.2.1 presents an example single block grid with total control volumes (grid cells shown with thin black lines) equal to '20×12', distributed in x and y directions respectively. Assuming this problem has huge computational requirements and we wish to solve it in parallel with multiple CPUs. We decide to solve this problem with 12 computing cores (hardware processors) and thus we need to divide the whole grid domain in 12 sub-domains, each of which will be assigned to one of the 12 individual cores to do the relevant computations for their sub-domain grid nodes. Fig. 3.2.1 shows the outer domain boundary in dark black lines and the red lines are the sub-domain interfaces. After this domain decomposition each sub-domain has 5×4 grid cells as visible in Fig. 3.2.1.

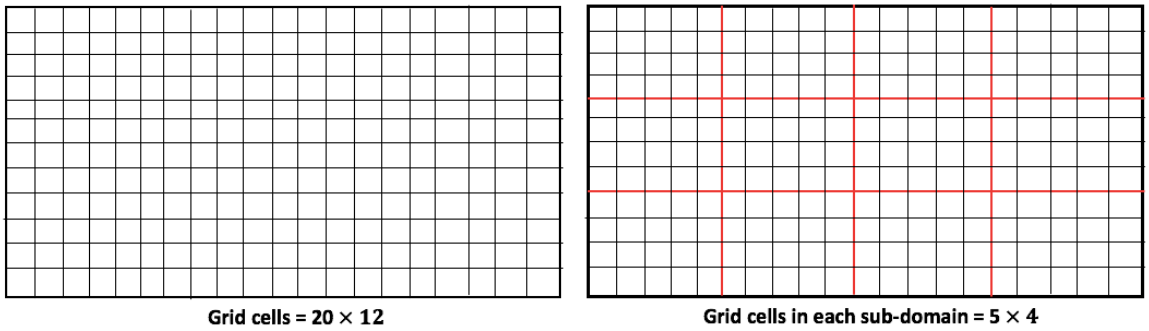


Fig.3.2.1 Example single block grid before and after MPI domain decomposition

In Finite Volume discretization methods, to compute the value of a variable on a grid cell node we require the variable values from one or more neighbor nodes also, depending upon the discretization schemes used. We need these neighbor nodes' data specially to prepare the algebraic equations for all the variables at all the mesh nodes. Usually, the internal nodes of the sub-domains have access to the data of their neighbor nodes as for each sub-domain all internal nodes belong to the same core. However, the nodes which belong to the cells at the sub-domain interfaces (red lines) do not have access to their neighbor nodes' data which lie on the other side of the interface. The neighbor nodes belong to another cores and their memory is not shared with other cores in MPI. To get the data from the neighbor nodes we need to make parallel communications between two adjoining cores. An MPI process (a program instance) is created to work on each individual core in our MPI strategies with which we will set the whole parallel communication environment. So, the main aim of this chapter is to describe how the whole MPI

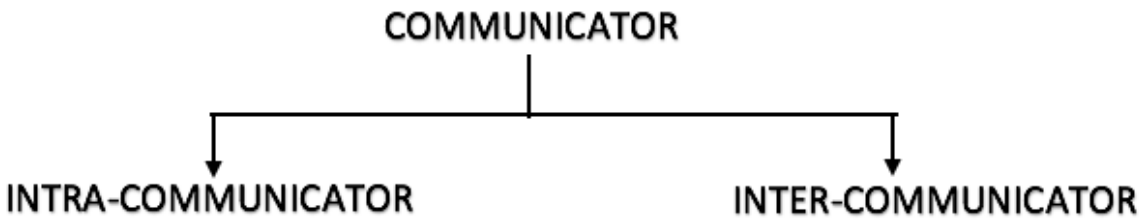
environment is set by the code in both single and multi-block grids. And, we discuss these strategies in detail in following sections of this chapter.

3.2.1 Message-Passing Interface (MPI)

MPI is developed to combine together the best features of many message-passing models that existed over the years. It is an attempt to organize and improve the existing features of message passing models and preparing a standard which remains portable across the range of hardware and software, available in market. As defined by the standard, "MPI (Message-Passing Interface) is a message-passing library specification" [2,7]. MPI is not a programming language, it is a library of functions which facilitate data transfer during the parallel communications. This communication protocol is the most widely used message passing model on various distributed memory architectures across various supercomputing clusters. MPI is the first specification which makes it possible to write truly portable parallel libraries.

Maintaining the portability, efficiency and functionality of parallel programs is the primary goal of MPI. Some advanced features of MPI include dynamic management of process groups, application-oriented process structures, large set of collective operations etc. More general and frequently utilized features of MPI are: point to point operations, communicators, collective operations, groups etc. We will discuss these features in following sections.

COMMUNICATOR. A communicator defines the message-passing or communication context among the MPI processes for all the communication operations in the MPI framework. It specifies the scope for all the MPI features like groups, topologies etc. With the help of a communicator a distinct communication universe is set by the MPI library for each distinct message-passing context. MPI_COMM_WORLD is the default pre-defined communicator provided by MPI, which makes available all the processes for different communication operations which are accessible after MPI is initialized. Following two types of communicators are defined in MPI



In MPI terminology, a communicator is a universe (collection) of processes which are independent of the other universes of processes (other communicators) in the context of the overall program. The processes of a communicator do not have direct communication links with

the processes of other communicators. These individual communicators are called as intra-communicators. An intra communicator is a collection of processes which make a same communicator context and can make message passing within this context only. As shown in Fig. 3.2.2, we have two intra-communicators, A and B, depicted with blue colored boundaries and containing their individual processes (P1, P2 etc.). The processes of either of these intra-communicators cannot make message passing with the processes of the other intra-communicator. To make communication between processes of different communicators MPI provides the concept of 'Inter-communication' [1,2].

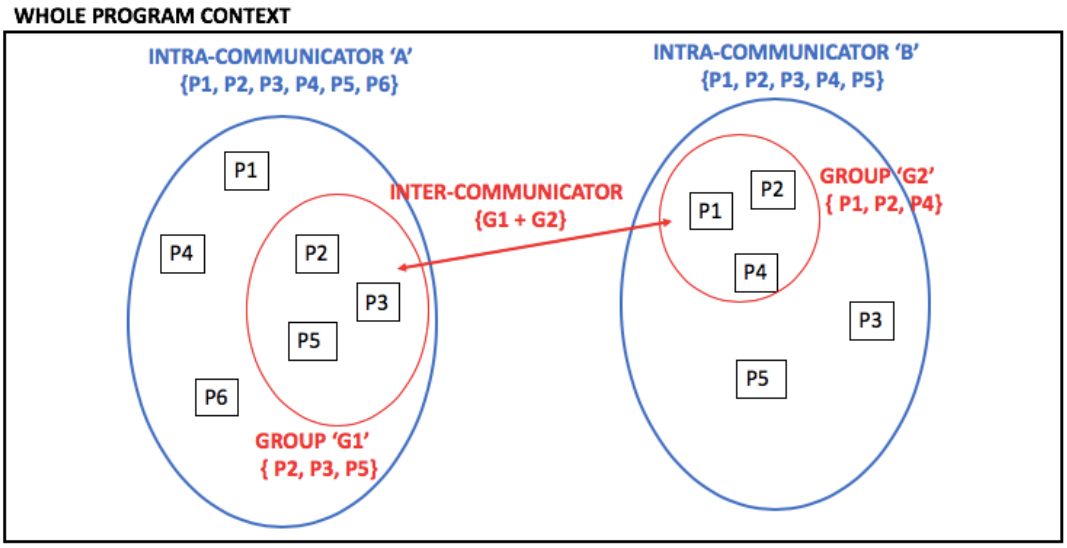


Fig.3.2.2 Sketch for Intra and inter-communicators

INTER-COMMUNICATOR. Parallel communications which involve the member processes of a same group are performed with intra-communicators. In intra-communicators, the send and receive contexts are identical. A group in intra-communicator is any group of processes which belong to this intra-communicator. In multi-disciplinary and modular applications, several groups of processes are required to work in separate communication contexts. In some situation, the processes of different groups need to communicate with each other. In these cases, the communications require the ranks of target processes and their group identifiers. In MPI, these types of communications are termed as inter-communications. In simple words, inter-communications are the message passing between processes in different groups which are disjoint. And, these groups belong to different MPI communicators.

An inter-communicator is created with two intra-communicators. For each inter-communicator there are two groups of processes. One group is termed as local group and the other group is termed as remote group for each communication operation. The group whose processes initiate some inter-communication operation becomes the local group for that operation, and the other group becomes the remote group for that operation. For example, for a send operation the group which has the sending processes (source) is the local group, and for the receive operation the group which has the receiver processes is the local group. The group of target processes is always the remote group. In a send operation the receiver's group is the remote group; and in

receive operation the sender's group is the remote group. MPI guarantees not to have conflict between operations of inter-communicators and other communicators.

Both groups belong to one of the parent intra-communicators which are used to create the inter-communicator. In Fig. 3.2.2, we see that intra-communicator 'A' has a group of processes (P2, P3 and P5) and the intra-communicator 'B' has a group of process (P1, P2 and P4). To make some data transfer between the processes of these two groups we need to make an inter-communicator which will provide a communication context for the data transfer between these processes which belong to different intra-communicators. The detailed description for creating such communicators is provided in following sections.

PROCESSOR TOPOLOGIES. In literal meaning, a topology is an arrangement of constituents of a group of things which are under study. A topology provides the information about the linkages and inter-connections between various members of the group or network. In parallel computing framework of MPI, two types of topologies are discussed: 1) hardware cores' topology and 2) the process topology [2]. First one reflects the arrangement or structure of the underlying hardware cores (processors) in the super-computing machines or clusters. The users of these machines have no control over the hardware topology, as it is decided by the manufacturer and the user uses it as it is available.

The second one is also termed as the 'virtual topology' or 'application topology'. Virtual topology as discussed here is the pattern of linkages of processes with each other in individual applications. It is clearly application dependent and the user controls it as the problem requirements suggest. Here, a *process* refers to a computer program which is actively executing certain instructions which it is asked to. It is a purely software concept which is different from a processor which is a hardware unit. A *processor* may consist of one or several computing cores which have the central processing units embedded in them.

For example, one of the super-computing clusters at Institut Pprime – 'THOR', on which most of the computations of this work have been carried out, has Intel Xeon E5-2680 V2 processors. Each of these processors has 10 physical computing cores (CPUs). Intel defines a core as: "*a hardware term that describes the number of independent central processing units in a single computing component (die or chip)" [4]*. This Intel Xeon processor has Intel Hyper-Threading (Intel HT Technology) which delivers two processing threads per physical core, making 20 virtual CPUs on 10 cores. Thread is defined by Intel as: "*A Thread, or thread of execution, is a software term for the basic ordered sequence of instructions that can be passed through or processed by a single CPU core*" *[4]*.

As described in introduction, a communicator sets up an appropriate message passing scope for all the communication operations as desired with the available number of cores. Within a communicator with N processes, all the processes are ranked from 0 to N-1. This sequential ranking arrangement of processes does not adequately reflect the problem specific logical communication pattern of processes. The communication pattern of the processes is always

problem dependent and is defined by the underlying geometry and numerical algorithms used for the problem. The problem geometries are often two or three dimensional in scientific computations according to the phenomena under study. These 2D or 3D grid problems usually lead to 2D or 3D topologies of communicating processes.

This virtual topology of processes is machine-independent and used to map the communication pattern on the underlying hardware topology. At the run time, the virtual topology is exploited by the system in assigning these software processes onto the physical processors (cores). This mapping of processes onto the cores usually provides performance gains in terms of computing hours on super-computing machines. If the user has no way to prepare a virtual topology of the processes on the hardware cores, then a random mapping results. A random mapping may lead to difficulties and contentions in the interconnecting processor network on some machines. Publications are available which report about the performance gains from a good process-to-processor mapping [1,3,7]. In addition to the possible performance benefits, virtual topologies also provide for a convenient process naming structure which significantly improve the readability of programs.

Graphs can be used to represent the communication pattern topologies of any type. In graphs, nodes can represent the processes and the edges of graph show the inter-connections between two processes. A virtual topology with graphs can be used for all applications, however, many applications have regular communication patterns and they can be more easily defined with other efficient methods. Many parallel applications make use of rings, 2D or 3D grids, tori etc. for their process topologies. In regular 2D and 3D geometries, the process topologies are easy to define with number of dimensions and total number of processes in each coordinate direction. These standard rings, tori and 2D-3D grids are easier than implementing graph topologies. Cartesian topology is one such standard process topologies which is explained explicitly in the literature and in Oracle3D Cartesian topologies have been used for significant performance benefits.

CARTESIAN TOPOLOGY. MPI provides support for three types of topologies: 1) Cartesian, 2) graph and 3) distributed graph. There are separate MPI calls to prepare each of these topologies as required by the problem. In a Cartesian topology, the process coordinates begin their numbering at 0. Cartesian topologies use row-major numbering for the processes. For example, a 2×2 grid of processes would be assigned to four-member processes as shown in Fig. 3.2.3. It shows the coordinates of each process in the 2D Cartesian topology and red numbers are the ranks assigned to the respective processes in this topology.

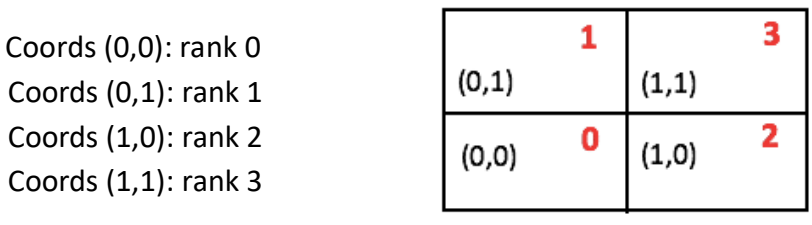


Fig.3.2.3 Sample process ranks and respective coordinates in a 2×2, Cartesian grid.

A topology is an extra and optional feature which is attached to a communicator. To set up a topology we need a communicator with a number of processes which we want to be the part of that topology. For a Cartesian topology creation MPI has the standard 'MPI_CART_CREATE' operation.

MPI_CART_CREATE(COMM, nDIMS, DIMS, PERIODS, REORDER, CART_COMM, IERR)
Creates a Cartesian communicator

COMM: input communicator handle
nDIMS: number of dimensions of Cartesian grid (integer)
DIMS: number of processes in each direction (integer array)
PERIODS: for periodic faces (logical array)
REORDER: managing the order of rankings of processes
CART_COMM: new communicator with Cartesian topology
IERR: error indicator

The first argument for the MPI_CART_CREATE operation is an input, intra-communicator handle, which will be converted into a Cartesian topology communicator. A Cartesian topology cannot be created with an inter-communicator. Second input argument is nDIMS which is the number of dimensions of required Cartesian topology. It has to be an integer which will be 2 for a 2D topology, and 3 for a 3D topology. DIMS is an integer array whose size is equal to nDIMS. Each element of DIMS(1:3) represents the total number of processes assigned in X,Y and Z direction respectively. The PERIODS argument is a logical array with true or false for each of the three directions. It is true for a direction where the grid has periodic boundary condition, else it is false. Reordering of the process ranks in new communicator is carried out with the argument REORDER. It may be required for good embedding of the process topology onto the hardware topology. It reorders the ranks in new communicator if certain performance gain could be obtained with reordering. If REORDER = false, then the ranks of processes in new communicator is kept same as in input communicator. The output argument is the new communicator handle 'CART_COMM' which is attached with the Cartesian topology, which is provided by MPI as a result of this call.

MPI_DIMS_CREATE(NPROCS, nDIMS, DIMS, IERR)
To obtain number of processes in each direction of the Cartesian communicator

MPI also provides some convenience functions for the creation of Cartesian topology. 'MPI_DIMS_CREATE' assists user in allotting a balanced distribution of processes in each coordinate direction. This function takes total number of available cores (NPROCS) as an input argument along with nDIMS. As an output, this function gives the variable DIMS in return. The user can also specify some constraints with variable DIMS. If DIMS is initialized with 0 then the MPI automatically provides a good distribution in respective directions. If the user wants to fix some specified number of processes in some direction, then he has to initialize DIMS with that number for that particular direction. For example, initializing DIMS(2) = 5 will fix the number of processes in y direction to 5. A divisibility algorithm is used by MPI to set DIMS for each direction as close to the other direction as possible. DIMS, when set by the call, will be in non-increasing order for the three directions.

Table 3.2.1 provides some examples with this function. In first case, no preferred DIMS are given by user to MPI library so 16 processes are distributed as 4 × 4 in two directions. In case 2, 13 process are divided in 13 × 1 as no other combination is possible. Case 3 sets DIMS in x direction to 3 as input, so 15 processes are divided in 3 × 5 × 1 for a 3-dimensional topology. In case 4, DIMS in y is fixed as 3 with input and total processes are 7. As 7 cannot be factorized in factors of 3, there will be an error in creating the output DIMS with this call.

Table 3.2.1 Some examples for 'MPI_DIMS_CREATE' operation.

DIMS **(input)**	**Function call**	DIMS **(output)**
(0,0)	MPI_DIMS_CREATE(16,2, DIMS)	(4,4)
(0,0)	MPI_DIMS_CREATE(13,2, DIMS)	(13,1)
(3,0,0)	MPI_DIMS_CREATE(15,3, DIMS)	(3,5,1)
(0,3,0)	MPI_DIMS_CREATE(7,3, DIMS)	Error

MPI_COMM_RANK(CART_COMM, CART_RANK, IERR)
Assign ranks to all processes of a communicator

Ranks of the processes should always be attached to them after creation of a new communicator. There is a standard function to assign a rank variable to each process in a communicator. MPI_COMM_RANK takes the communicator handle as input argument in which we want to assign ranks to a variable. And the output is the new ranks variable for all the processes of that communicator. Here, we input the newly created Cartesian communicator 'CART_COMM' as the input and 'CART_RANK' will be the output of this function here. So, after this function is returned; each process in the CART_COMM will attach its rank in the CART_COMM to the variable CART_RANK.

MPI_CART_COORDS(CART_COMM, CART_RANK, nDIMS, COORDS, IERR)
Assign coordinates to each process in the Cartesian communicator

In the beginning of this section, the concept of coordinates of the processes was highlighted in Fig. 3.2.3. MPI has another important function for providing the coordinates of processes in a Cartesian topology. The coordinates of the processes are also attached with the Cartesian communicator. The Cartesian communicator handle is the first input argument of the function to get process coordinates. The coordinates are provided for each process of the communicator; thus, each process has to make a call to this function with the second argument being the Cartesian rank of that process. Number of total dimension of the Cartesian topology is the third input argument for MPI_CART_COORDS function. The coordinates of each process in the Cartesian topology are returned in fourth argument. The COORDS variable is an integer array with a size of number of dimensions of the topology.

MPI_CART_SHIFT(COMM, direction, disp, source, destination, IERR)
Obtain ranks of the neighbor processes, for each process, in all the direction

MPI_CART_SHIFT is one of the most important functions of the Cartesian topology. It provides the neighbor process ranks to the calling process in all 6 dimensions of the topology. Thus, each process after calling this function knows its neighbors. It is most useful in the cases where data shift operations are performed with the neighbors in all directions. In one data shift operation, a process sends some data to a neighboring process and also receives some data from some neighbor via the corresponding interface nodes. The required user inputs for MPI_CART_SHIFT are the communicator handle, direction in which neighbor ranks are needed and the step size after which the ranks are needed. The source and destination arguments represent the output ranks of the source process and destination process of the caller process in the direction specified by the second argument. In cases where the caller process is at the boundary of the topology, and so there is no neighbor on the boundary side, then value MPI_PROC_NULL is returned for the respective source or neighbor rank. MPI_PROC_NULL tells that the rank of source or destination in that direction is out of range of the topology.

There are several MPI send and receive operations according to the requirements of the situations. Here, we have used 'MPI_SENDRECV' operation for all the data exchange via the interfaces. It is a blocking send-receive operation. With MPI_SENDRECV the processes make combined send and receive operations in one single call. The send and receive can be to the same process or a different one. If there is a requirement of data shift operation across a chain of processes, then MPI_SENDRECV can be the most suitable operation. In individual blocking send and blocking receive operations user has to take care of the cyclic dependencies of the calls in data shift operations. So, the user has to correctly order the send and receive calls to avoid a deadlock situation while using individual send and receive. With MPI_SENDRECV the MPI communication subsystem takes care of these troubling issues. A small description of the basic MPI_SEND and MPI_RECV operations is provided in Appendix II, and more details can be found in [1,2].

In the MPI_SENDRECV, the calling process needs the destination rank for sending part and the origin rank for the receiving part of the operation. In the Cartesian topology environments MPI_SENDRECV is especially very useful with MPI_CART_SHIFT operation. With MPI_CART_SHIFT each process knows is neighbors and it can quickly make use of the MPI_SENDRECV to make the data shift operations. The standard MPI_SENDRECV operation as used in Oracle3D (SWAP_3D subroutine) is as follows:

```
! Message sending and receiving in X direction
MPI_SENDRECV(SEND_WEST, inpW, MPI_RK, NBR_WEST, TAG,
                    RECV_WEST, inpW, MPI_RK, NBR_WEST, TAG,
                     CART_COMM, STATUS, IERR)
```

Making a blocking send and receive with a single function call

The first five arguments are for the send operation and the next five arguments are for the receive operation. Starting from SEND_WEST the arguments respectively are: send buffer, send buffer size, send data type, destination rank and tag for the send operation. Same sequence

is followed in the receive part of the call. The last three arguments are the communicator handle, status object and the error indicator variable. This send-receive operation can also receive a message sent by a regular send operation; the sent message by this send-receive can also be received by a regular receive. Both the send and receive can have different sizes and data types. And, they both must be disjoint; meaning a same buffer cannot simultaneously send and receive data. There is only one communicator handle so both the send and receive operation must be performed within same communicator.

In this example call, the calling process sends and receives data from its neighbor in west direction (NBR_WEST). SEND_WEST and RECV_WEST are the variables which hold the data to send and data which will be received by the calling process from NBR_WEST. Fig. 3.2.4 shows an example 2D cartesian topology with 12 processes, ranked from 0 to 11. The neighbors for rank 7 are shown in all 4 directions in this figure (NBR_WEST=4, NBR_EAST=10, NBR_NORTH=8, NBR_SOUTH=6). As explained above the MPI_CART_SHIFT function when called by process ranked 7, the output will be the ranks of these four neighbor processes.

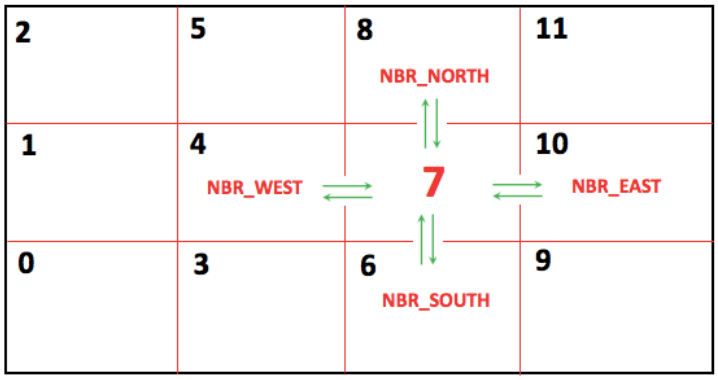


Fig.3.2.4 Example 2D cartesian topology with process ranks

In Fig. 3.2.4, the outer black line represents the whole problem domain and we divide this domain in as many numbers of sub-domains as we have MPI processes. Here we have divided this domain in 12 sub-domains (4 × 3) which have red lines as their interfaces with neighbor sub-domains. At each sub-domain interface, we have grid nodes (on both sides) for which we need to send and receive some data to the neighbor on the other side of interface, which is performed with MPI_SENDRECV. Each sub-domain will be assigned to an MPI process according to the created cartesian topology, and this MPI process will handle all the computational work related with its sub-domain. More details on domain decomposition are provided in section 3.3 while discussing the multi-block grid problems.

3.2.2 Implementation of Cartesian topology in Oracle3D

The Cartesian topology features of MPI are implemented in Oracle3D as they are described in previous section. The major overall tasks to parallelize single block grids include:

I. Grid management,
II. Initializing MPI environment
III. Setting Cartesian topology
IV. Finding the neighbors in all direction, and
V. Making the MPI communications with MPI_SENDRECV.

All of these stages are explained in much more detail in next section (section 3.3.1) when we describe the overall, broad strategy to parallelize the multi-block grids with examples. Understanding the concept and use of Cartesian topology features is sufficient for this part of the report. This whole methodology of MPI was implemented in a Poisson solver, as it provides a simple problem to assess the working of MPI. A similar strategy for parallelizing a Poisson solver is well explained by W. Gropp et al. (1999) in their book titled 'Using MPI' [1]. This book along with the MPI 3.1 standard manual should be referred for more advanced features on Cartesian topology as provided by MPI. After the Poisson solver was parallelized, we tested the performance of the code with a sample problem. Several detailed validation cases are provided in next chapter, here, we restrict ourselves to analyze the initial performance of the code with the newly implemented MPI features.

Two values of time taken for the program execution are noted: 1) total time taken for the complete execution of code, 2) time taken during the actual computations alone (Computational time). Computational time for this steady case is mainly devoted to the linear system solver, which is the most time-consuming part. Except this computational time, the other tasks in code (total time) include mainly reading the grid, partitioning and distributing the sub-domain to all processes etc., these tasks are done by only one process, so they don't come into the parallel part of the code. There are also the subroutines which compute some geometrical parameters of the grid like volume of cells, interpolation factors etc., these tasks are done after the grid management, so they are done by individual processes on their own sub-domain, however, they are not included in computational time here.

Parallel Efficiency

A single block orthogonal grid with 200 cells in each direction (X, Y and Z), was taken as the test problem here. We solve the Poisson equation: $\nabla^2\varphi(x,y,z) = 6$, for this test. Here we show the performance of adapted MPI methodology with scalability and speedup plots mainly. Very strict tolerance ($1.E^{-25}$) for linear system solver was set, so that it remains unachievable in given maximum number of iterations (4 million in this case), because it is essential to have exactly the same computational load for all the cases with different number of MPI cores. Seven sets of cores (10, 20, 40, 100, 200, 400 and 800) were taken to analyze the efficiency of the code with increasing computational power.

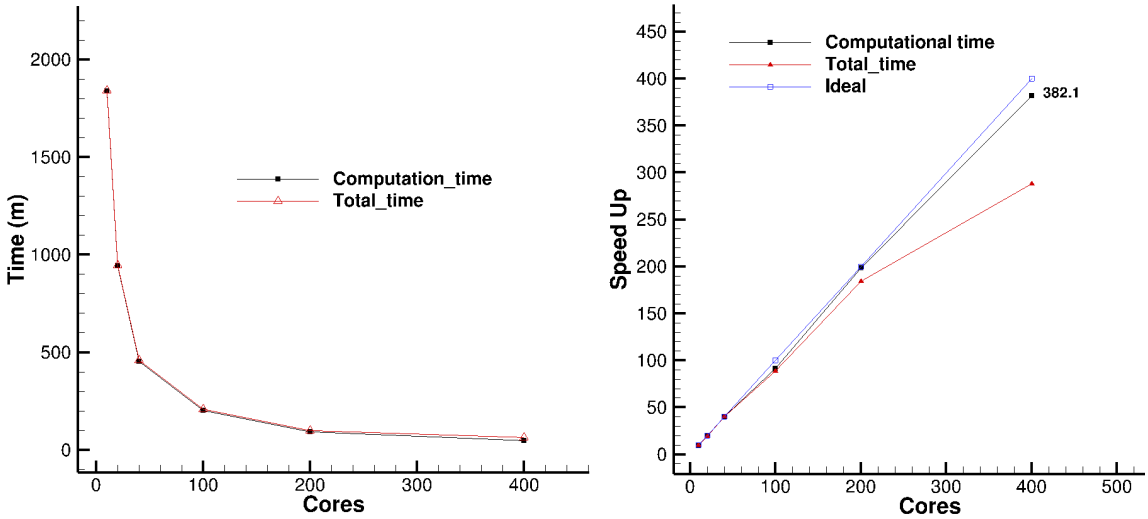


Fig. 3.2.5 Performance plots with single block MPI implementation; a) scalability, b) speedup

Excellent performance is achieved with our implementation for single block Poisson case. Fig. 3.2.5 (a) illustrates the standard scalability plot with decreasing time as we increase the number of MPI cores. Speed-up plot shows almost matching performance with computational time in comparison with the ideal values till 400 MPI cores. Here it should be noted that the base value for speedup comparison was taken as the time taken by 10 cores, because with the current problem size it would have taken roughly 15 days on a single core. Taking the baseline speedup to 8,10,16 or 128 cores is a usual practice where scalability is performed with huge number of cores on large problems [8,9,12]. However, a smaller problem was simulated to verify the speedup from 1 core to 16 cores, and desired speedup was obtained in that problem, Fig. 3.2.6.

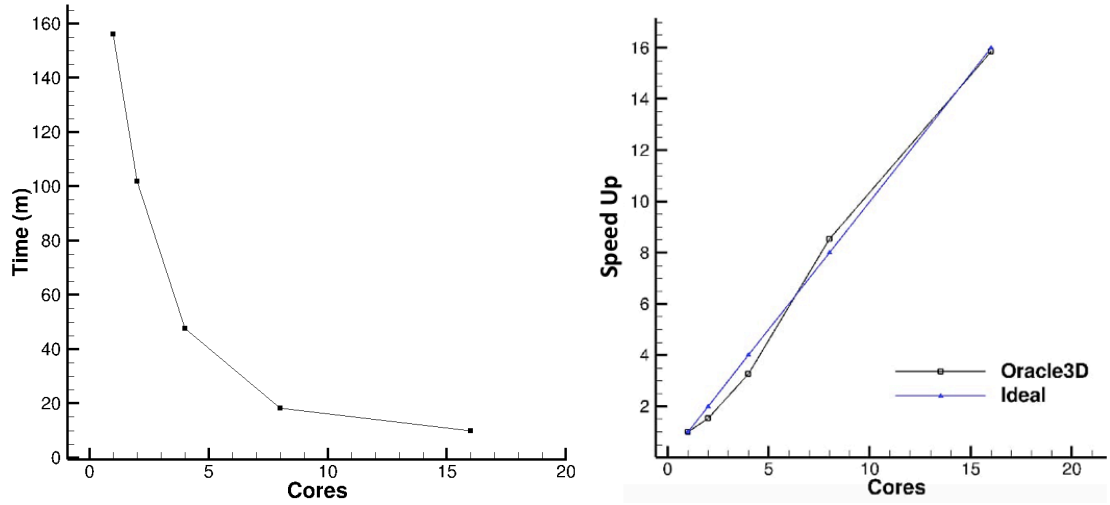


Fig.3.2.6 Performance plots with 1 to 16 cores ; a) scalability, b) speedup

Interestingly, the total time is increasing with increasing number of cores, this is due to the sequential part of the code, mainly, the grid management task which is always performed by a single master process. So, as the number of cores increases there are more number of partitioned to be done, and the partitioned data is to be sent to more number of member processes. Thus, increasing the unidirectional MPI_SEND calls to send the sub-domain data to the member

processes by the master process. We computed the difference of total time and computational time for all the cases and found that this difference in time, which represent the part of sequential code, was almost showing linear behavior after 200 cores, Fig. 3.2.7 (a).

We also observe that for 800 cores the desired speedup is not obtained with this problem size, Fig. 3.2.7 (b). Table 3.2.2 slightly assists in understanding this problem. We see that the computational time for 800 cores is only ~39 minutes. We can say that this problem size was not sufficiently large for achieving a desired speedup with 800 cores. In such fixed size problems data per processor is decreasing as the number of processors are increased, leading to a higher communication-to-computation ratios which prevents achieving a desired speed up [10]. In this case the communication overhead was not small in comparison with the computational time. We again note that the total number of iterations in this case was 4 million and at each iteration every core makes the communication with neighbors. We will revisit such problem in multi-block case also and there we will increase the computational load of the problem and check the performance with increased work load.

Table. 3.2.2. Time values in minutes for three sets of cores

Cores	Total_time (T1)	Computational_time (T2)	T1-T2
200	100	92.71	7.29
400	64	48.10	15.9
800	71	38.94	32.06

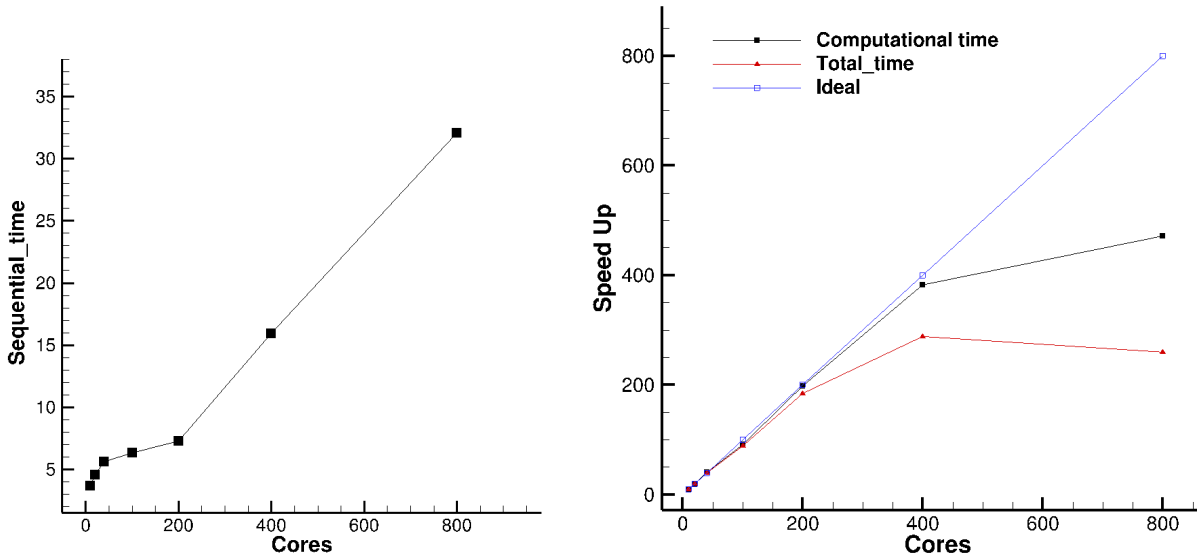


Fig.3.2.7 a) effect of sequential part of code, b) Speedup results showing the effect of communication overhead with 800 cores.

3.3 MPI extension to Multi-Block grids

Numerical simulations are performed on either structured or unstructured grids. Structured grids have some notable advantages over the unstructured grids, such as: lower memory usage, better cache utilization, vectorization, simpler coding, faster convergence etc. [14]. Our code 'Oracel3D' works with multi-block structured grids to make use of these advantages of structured grids. Generally, complex engineering geometries are very difficult to mesh with one block structured mesh. Multi-block structured grids are created in such cases, in which a geometry is meshed with several individual structured mesh blocks as shown in Fig. 3.3.1. In the previous section we discussed how Cartesian topology features of MPI provide us with an advantageous methodology to prepare scalable MPI codes in single block grids. In this section, we will extend our Cartesian topology strategy to multi-block grids using some more advanced MPI features.

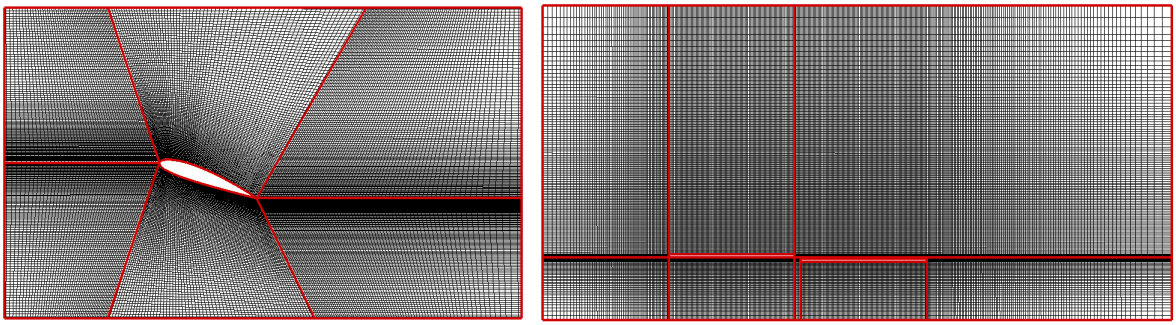


Fig.3.3.1 Example multi-block structured grids for a NACA and DBD actuator problem

In single block grid cases we created a Cartesian topology of the MPI processes for the grid block and MPI communications among the partitioned sub-domains of the grid block were carried out within the context of the created Cartesian communicator. With single block grids, we obtained a very good scalability with the features of Cartesian communicators. These results from the previous section led us to state that Cartesian topology features provide excellent MPI communication speedup. Keeping this in mind, we proposed that in multi-block grids we make individual Cartesian communicators for each grid block to achieve a very good scalability within individual blocks. And, as we achieve very good scalability within a block, consequently we will achieve a desired scalability in overall grid also, provided we manage the interface communications effectively as well.

In single block grids we have to make MPI communications at sub-domain interfaces for FVM discretization. In multi-block grids, two neighbor grid blocks have common interface between them, and the nodes which belong to the cells at these block interfaces also need to access the data of the cell nodes on the other side of this block interface. As proposed above, multi-block grids will require individual Cartesian communicators for individual grid blocks. At an interface the neighboring nodes belong to the MPI process on the other side of the interface. And, in multi-block grids, this neighbor process belongs to a different Cartesian communicator altogether. And there is no direct link of communication between the processes of different communicators unless they have an inter-communicator as a bridge between them.

From the definition itself, a communicator is a universe of processes which has no direct knowledge of the other universes. Thus, if we create individual Cartesian communicators for each grid block, then we must have inter-communicators to have data exchange at the interfaces of these individual Cartesian communicators. By this point we have explained the need to have multiple Cartesian communicators and inter-communicators. Let us discuss, in more detail, the concept of inter-communicators.

3.3.1 Setting the MPI environment for multi-block grids

This section provides a brief problem statement and a summary of the rest of the chapter. Oracle3D works with multi-block structured grids, the first task is to read the whole grid in accordance with its block structure. Then we need to decompose the grid of each block in as many numbers of sub-domains as we have MPI processes for that grid block. The grid data (coordinates, boundary nodes etc.) of the decomposed sub-domains are then distributed among the respective MPI processes with standard MPI send and receive operations. To set up the whole parallel communication pattern map we have to go through several steps to create the complete MPI environment with various groups and communicators, which we discuss one by one in this section. First, let us explain the main objective with an example case and some sketches.

Fig. 3.3.2 represents the problem input grid and the created inter-communicators at the interfaces. Here, we consider a geometry which has 4 grid blocks, as shown numbered from 1 to 4 in green color. The green colored lines are the interfaces between the two neighboring blocks and the outer black lines are the boundaries. This is our base grid before preparing the MPI communication environment for the problem. The right part of the sketch represents specifically the output inter-communicators which are created at the four block interfaces. Each block's inner partition is done with 9 cores each which will make one Cartesian communicator per grid block and at the interfaces we would need the inter-communicators to exchange data between two blocks. The inter-communicator between block 1 and 2 is named as INTER_COMM12 and, similarly, the other inter-communicators are also named as shown in Fig. 3.3.2. Thus, it is our main objective in this section to set the whole MPI mapping of cores for the communication in multi-block geometries.

Major steps towards creating inter-communicators:

I. Grid management
II. Cartesian Topology
III. Interface Groups
IV. Interface Intra-communicators
V. Interface Inter-communicators
VI. The NUM_CFI array
VII. Data exchange among MPI processes

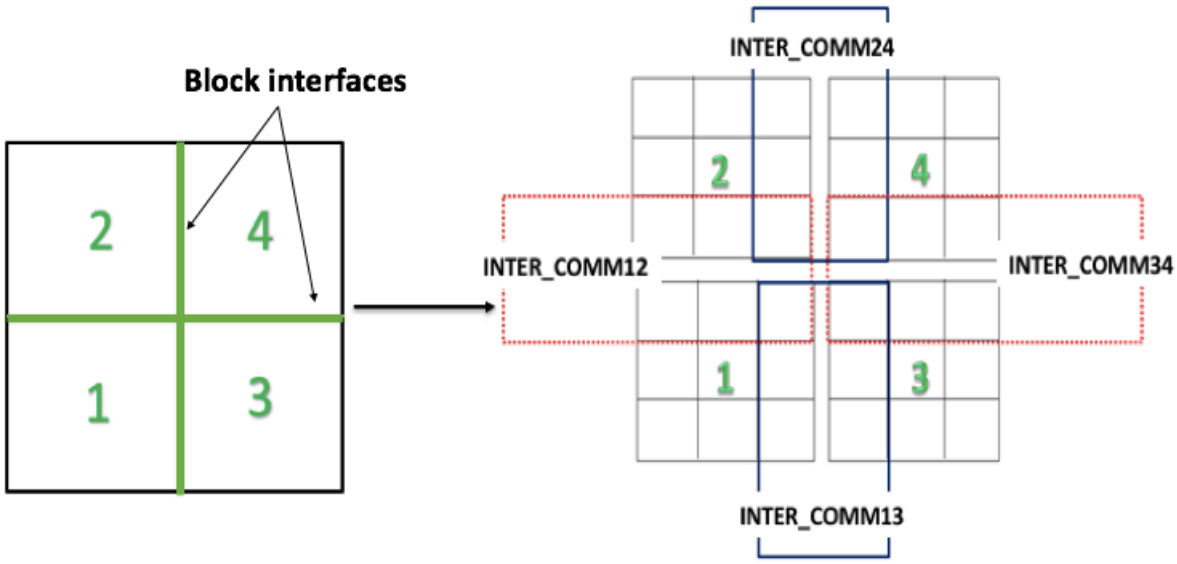


Fig.3.3.2 The input 4 block geometry and the final output showing the inter-communicators at the interfaces of the grid blocks.

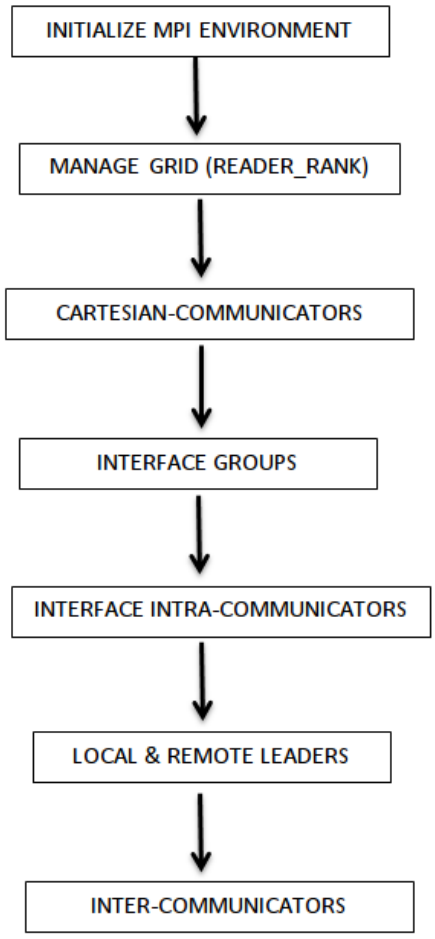


Fig.3.3.3 Flowchart with the sequence of operations to create inter-communicators

Fig. 3.3.3 provides us a summary flowchart for preparing the inter-communicators from MPI_COMM_WORLD. All the major steps towards the creation of final interface inter-communicators are shown in this flowchart and these steps are discussed in detail with corresponding MPI operations in following sections.

I. Grid Management

Main Objectives:

a) allocation of MPI processes to each grid block
b) reading the multi-block grid for parallel implementation
c) partitioning each block into MPI sub-domains
d) distribution of sub-domain data to respective MPI processes

As soon as user provides the number of available MPI processes (NPROCS) for the problem and the number of geometrical blocks in grid (NBLOCKT), we have to distribute the NPROCS among the grid blocks. For this, we first decide how many processes should be provided to each grid block according the sizes (total number of control volumes) of all the blocks. We make calculations to have a proper load balancing among all the MPI processes and decide number of processes for each block. After this we start with first grid block and first MPI processes (RANK=0) and proceed with allotting processes to each successive block with successive RANKS. For example, if we have 4 grid blocks and 60 MPI processes and the number of processes each block after good load balancing is 12,18,15 and 15. Then the MPI processes are allotted to blocks as shown in Tab.3.3.1.

Tab.3.3.1 Example MPI process allocation to grid blocks

MY_GEOM_BLK	COLOR	Total MPI processes	MPI processes (RANKs)	READER_RANK
1	1	12	0 - 11	0
2	2	18	12 - 29	12
3	3	15	30 - 44	30
4	4	15	45 - 59	45

The MPI processes distribution is done in CORE_INFO subroutine. In this subroutine we also assign some important variables to all the processes, which are local to the specific processes. Each process assigns the variable ‘MY_GEOM_BLK’ the ID of the block to which this process has been allotted. For instance, in above example all the processes from RANK 12 to 29 will assign MY_GEOM_BLK to 2; and these cores will be called the member processes of that block. Variable ‘COLOR’ is again assigned the ID of the block to which the processes belongs. The variable ‘COLOR’ is specifically used with the MPI operation of communicator split as used in standard references [1,2], and is explained in next section. And, variable ‘MY_GEOM_BLK’ is used at all other places in code where the processes have to provide their block IDs. A variable named ‘READER_RANK’ is also assigned for each process, which is

the rank of the processes which will read the grid data of its block, decompose the grid data and distribute the grid data to the respective member processes. We have assigned the first process' rank of each grid block as the 'READER_RANK' of that block. In the tab.3.3.1 example, ranks 0, 12, 30 and 45 are respectively the 'READER_RANKs' of blocks 1,2,3 and 4.

This 'READER_RANK' strategy to read and partition the grid blocks by different processes is especially efficient in cases of multi-block grids of huge sizes. During the scalability tests we have observed that in a 4-block grid of 8 million control volumes in total, the test with 512 cores took nearly 45 minutes just to read, partition and distribute the sub-domain data among all the processes. In that case there were 4 reader cores which managed the 4 grid block data. Thus, if we keep the grid management job to just one master process and it alone does all the partition and distribution, then certainly it will take much more time than the reader process method. The reader process strategy also avoids certain possible deadlock situations while distributing the grid data to member processes of a block. In this strategy, the sending and receiving of data is happening only in one direction; the reader processes of the blocks only send the grid data and the member processes only receive the data. This would not have been possible if we would have taken the RANKS 0 to 3 to manage the grid of 4 blocks; it could lead to possible deadlock situations in our code.

In the grid file (*.grd) we mainly have the coordinates or vertices, control volume numbers in each direction and the boundary condition data related to the grid blocks. All the data related to a particular grid block is written in same section of the grid file and after data of one block finishes the data of next block starts. During preparation of the grid file the sequence of grid blocks from 1 to NBLOCK is strictly followed. Now, with the 'READER_RANK' available only the reader processes read and store the information of their respective blocks, this task of reading the grid file is done in subroutine READGRID in Oracle3D.

Then, MANAGE_MPI_GRID subroutine calls two different subroutines to decompose the coordinates and boundary condition data of the grid blocks. In these two subroutines: PARTITION_COORDINATES and PARTITION_BOUNDARY_POINTS, the reader processes make the partition of their grid blocks into the number of processes allotted for their blocks. After the partition the decomposed sub-domain data are sent to the respective member processes of that block with standard MPI_SEND and MPI_RECV operations. And, the reader processes also keep their own sub-domain data to themselves.

```
MPI_SEND(X, NXYZA, MPI_RK, DEST, TAG, CART_COMM, ierr)
MPI_RECV(X, NXYZA, MPI_RK, SOURCE, TAG, CART_COMM, STATUS, ierr)
```

At this stage, all of these processes (RANK 0 to 59) belong to the default MPI communicator which is the MPI_COMM_WORLD. Our experience with single geometrical block case tells us that with Cartesian topology features of MPI we can get very good scalability and speed up with increasing number of MPI processes. The experience and results with Cartesian communicators lead us to think about implementation of this Cartesian topology feature in the multi-block cases also. However, a single Cartesian communicator, as applied to single block

cases, will not work in case of multi-block grids. But the Cartesian topology can be used with individual grid blocks, which will certainly give us very good parallel efficiency within individual blocks. Considering these points, we decided to divide our default communicator (MPI_COMM_WORLD) into as many number of Cartesian communicators as there are grid blocks.

II. Cartesian Topology

Main Objectives:

a) Splitting the default communicator into required intra-communicators

b) creating one Cartesian communicator for each block

c) Assigning local Cartesian ranks, coordinates to all processes within their respective Cartesian communicators

d) Finding neighbor processes of each process on all directions

The creation of Cartesian topology for individual blocks is managed in subroutine CART_TOPO_MUTIBLOCK. We have our default world communicator for the input here. A Cartesian communicator can be created with the standard MPI operation – 'MPI_CART_CREATE'. This operation needs an input communicator handle as its first argument, which will be converted into a Cartesian communicator. In the single block cases we needed only one Cartesian communicator, so the default world communicator was completely converted into a Cartesian communicator. But, in multi-block cases we need as many numbers of input communicators as the number of Cartesian communicators required, which is equal to the number of blocks we have. So, first we have to split the world communicator into the required number of intra-communicators. And, following which we will convert those newly created intra-communicators into the Cartesian communicators.

The task of splitting the world communicator is carried out with the standard MPI operation - 'MPI_COMM_SPLIT'. This operation has five arguments, out of which first three are input arguments and last two are the output arguments. The first argument is the handle of the communicator which we want to split, here it is our world communicator (MPI_COMM_WORLD). The second is the 'COLOR' argument. This 'COLOR' argument is the identification of the groups to which the processes belong. In our context the identification of the groups corresponds to the grid block ID, meaning the group of processes which are assigned to a same grid block.

So, processes assigned to same grid block will all have same value of the 'COLOR' argument. And, as a result the processes which have same value of 'COLOR' will all combinedly make a new intra-communicator among themselves which will have only these processes as its members. The third argument is the rank of the process which is reading/making this call of splitting. Every process makes a call to MPI_COMM_SPLIT with its RANK and COLOR values which lets each process associate with the output communicator and the other member processes of the that communicator.

MPI_COMM_SPLIT(MPI_COMM_WORLD, COLOR, RANK, BLOCK_COMM, IERR)
Split a communicator in required number of intra-communicators

MPI_COMM_WORLD: input communicator handle
COLOR: identification of the group
RANK: rank of the calling core
BLOCK_COMM: output communicator handle
IERR: error indicator

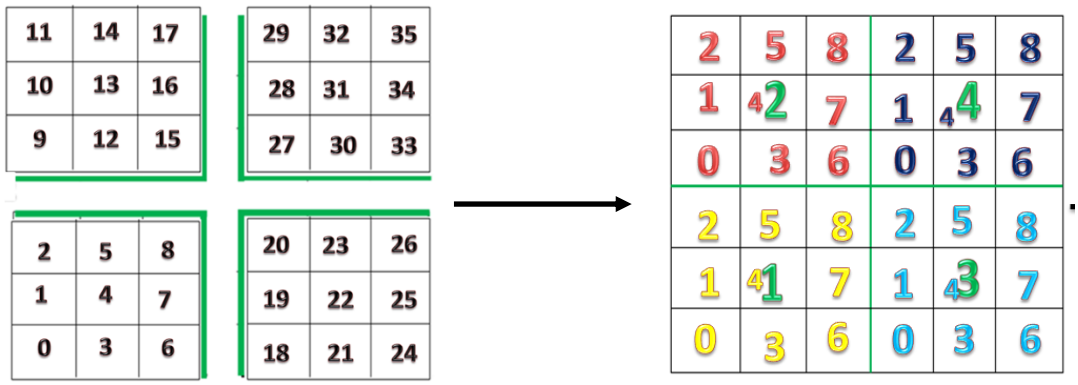


Fig.3.3.3 Global ranks within MPI_COMM_WORLD and local ranks in each BLOCK_COMMs

As a result of this 'MPI_COMM_SPLIT' operation, there will be as many new communicators as there are different 'COLOR' values (or blocks). The new communicator handle, which is 'BLOCK_COMM' in our code, is the same for all the processes because every process recognizes only the 'BLOCK_COMM' communicator which it created itself. A process makes the communicator with its 'COLOR' value and it has no knowledge of the communicator which was created with a different 'COLOR' value by some other process. In other words, a 'BLOCK_COMM' is known to only those processes which has the 'COLOR' value which was used to create this 'BLOCK_COMM'.

In the example of Fig. 3.3.2 we considered that each block will have 9 processes. So, the 'COLOR' values of the processes ranked 0 to 8 will be 1; ranks 9 to 17 will have COLOR as 2 and similarly for rest of the processes. For example, ranks 18 to 26 all will have color value 3 and they will make a BLOCK_COMM for their group by the split operation. Fig. 3.3.3 shows this example with 4 blocks having their ranks shown in different colors corresponding to individual BLOCK_COMMs. Here we also note that whenever a new communicator is created the local ranks of the processes which belong to this new communicator will start from zero again, within this new communicator.

***Note**: Although we could have used the variable 'MY_GEOM_BLK' in place of variable 'COLOR' because both the variables are exactly the same. But we kept the standard approach as used in many references which use the variable 'COLOR' as the identification of the groups. Moreover, it will be easier for new users to relate this operation with 'COLOR' variable to standard references and understand quickly.*

MPI_CART_CREATE(BLOCK_COMM, MAX_DIMS, DIMS, PERIODS, REORDER, CART_COMM, IERR)
Creates a Cartesian communicator 'CART_COMM' from existing intra-communicator 'BLOCK_COMM'

After splitting of MPI_COMM_WORLD we have obtained one communicator for each block : 'BLOCK_COMM'. We will convert these 'BLOCK_COMMs' into Cartesian communicators for the respective blocks. We use the MPI operation – MPI_CART_CREATE, as explained in the single block case and obtain the required Cartesian communicators for each block. Fig. 3.3.4, illustrates the steps of creating the Cartesian communicators from the existing world communicator. We use other features of Cartesian topology to get the information on the newly created Cartesian communicators. These features were well explained previously, they are:

MPI_COMM_RANK(CART_COMM, CART_RANK, IERR)
Returns local Cartesian ranks of all processes in variable 'CART_RANK' within respective CART_COMM

MPI_CART_COORDS(CART_COMM, CART_RANK, MAX_DIMS, COORDS, IERR)
Returns coordinates of processes in the topology of respective CART_COMMs

MPI_CART_SHIFT(CART_COMM, 0, 1, NBR_WEST, NBR_EAST, IERR)
Returns the ranks of neighbor processes in respective directions

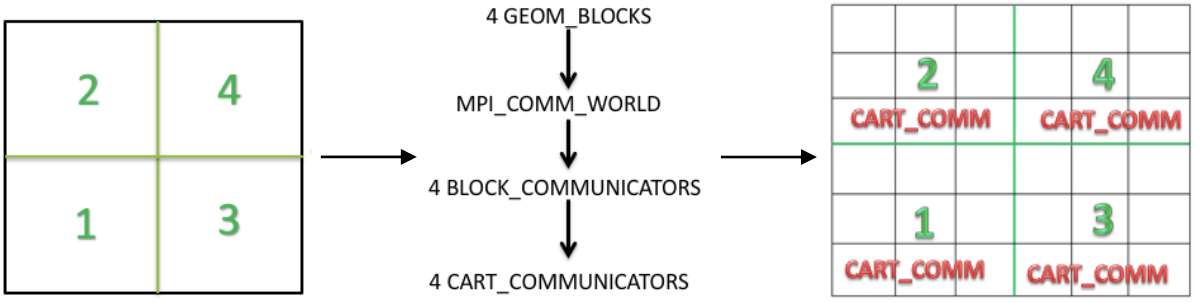


Fig. 3.3.4 Sequence of creating Cartesian communicators from 4 grid blocks

We identify the local ranks within each Cartesian communicator with variable 'CART_RANK', and the global ranks in world communicator were identified with variable 'RANK'.

Tab.3.3.2 Global and local ranks within communicators

BLOCK ID	RANK (MPI_COMM_WORLD)	CART_RANK (CART_COMM)
1	0 – 8	0 – 8
2	9 – 17	0 – 8
3	18 – 26	0 – 8
4	27 – 35	0 – 8

With the Cartesian communicators we will manage the computations and message passing within the individual blocks. But, we have to make data exchanges between the blocks also to compute the variables on the nodes which belong to the interface cells of the block grids. To make these inter-block data exchanges we would need to make communications among the processes of different Cartesian communicators, as each block is assigned to one Cartesian communicator. And, there is no direct communication link among the processes of different communicators. For this, we proceed towards creating the inter communicators.

Now, not all of the processes of a Cartesian communicator will take part in data exchange with neighbor Cartesian communicators. Only those processes which are at the interfaces of two neighbor blocks will have to communicate with each other for data exchange between them. Also, a Cartesian communicator may have more than one direction in which it would have neighbor Cartesian communicators. Thus, a communicator needs to have several local groups of processes at different interfaces which will communicate with the processes of neighbor Cartesian communicators at respective interfaces. These local groups of processes will be used to create the intra-communicators at the corresponding interfaces, and subsequently two intra-communicators lying on either side of an interface will be used to create an inter-communicator at that block interface.

III. Interface Groups

Main Objectives:

a) Finding the MPI processes at the block interfaces

b) Creating MPI groups of processes which belong to the same ISIDE of a block

As a next step towards creating the inter-block communications, we make groups of processes within each Cartesian communicator at the directions in which it has to exchange data. We follow the MPI standards and first create group of all the processes of a Cartesian communicator with operation: MPI_COMM_GROUP. This operation takes as input the communicator handle in which we want to create the group of all the processes. And it returns a group handle which consists of all the processes of the input communicator. In our case 'CART_COMM' is the input communicator handle and 'CART_GROUP' is the return group handle.

MPI_COMM_GROUP(CART_COMM, CART_GROUP, IERR)

Creates an MPI group 'CART_GROUP' from Communicator 'CART_COMM'

Here, it should be noted that in MPI new groups can't be created from scratch. Groups can be constructed by manipulating already existing groups [2]. MPI provides the base group which is associated with the global communicator MPI_COMM_WORLD, and this group can be accessed with the function MPI_COMM_GROUP. Thus, we need to create the parent group of all the processes in each of our CART_COMMs, and then the sub-groups at the interfaces will be created from the parent group with the processes at respective interfaces.

With the newly created ‘CART_GROUP’ within each Cartesian communicator, we can now create several sub-groups of processes which belong to the block interfaces in respective directions. For this, we have to first find which processes within a Cartesian communicator belong to which interface. Interfaces are identified with the direction to which they belong. We are working with 3D block structured grids and each of our grid blocks have six faces. We refer these six faces with six directions respectively: west, east, south, north, back, and front. All the block interfaces will belong to one of these six directions and will be identified with variable ISIDE. Where ISIDE being 1, 2, 3, 4, 5 and 6 corresponding to the above mentioned six directions.

The important aspect to note here is that after the creation of Cartesian communicators, the CART_RANKs will be used for all the operations within a Cartesian communicator. So, when we need to find the processes which are at the interfaces we have to find their local Cartesian communicator ranks (CART_RANK). This job is done in subroutine ‘INTERFACE_GROUPS’. This subroutine uses the fact that all the ranks in a Cartesian communicator are arranged in a pre-defined direction of increment. The ranks start with zero, the inner most loop to assign the ranks runs in Z direction, then the middle loop runs in Y direction and the outer most loop goes in X direction. Fig. 3.3.5 Shows the arrangement of local ranks in an example 3D Cartesian topology.

As we now understand how the local processes in a 3D Cartesian topology are arranged, we can easily find out the processes which belong to the 6 faces of the block grids. We will not need to find the processes at all the six faces rather, we know from the grid file which faces have interfaces with neighbor grid blocks. So, we find the ISIDE in which a Cartesian communicator has a block interface, and then we run the loop according to the face direction at which we are looking for the ranks. The subroutine ‘INTERFACE_GROUP’ prepares the variable ‘LOCAL_RANKS_INF’ which contains those processes (CART_RANKs) of the Cartesian communicator which are at the interfaces. The first index of this array is the ISIDE, and the second index is the Cartesian ranks of the processes at that ISIDE interface.

Note: ‘Group’ is again an MPI concept like communicators. A group defines a collection of processes which have ordered ranks in the communication pattern. Groups define the scope of process names (ranks) in collective and point-to-point communication operations in the MPI environment. In other words, a group is attached to and used within a communicator to describe the constituent processes in that communication universe. Although, groups are defined and manipulated separately from communicators but only communicators are used in all the communication operations.

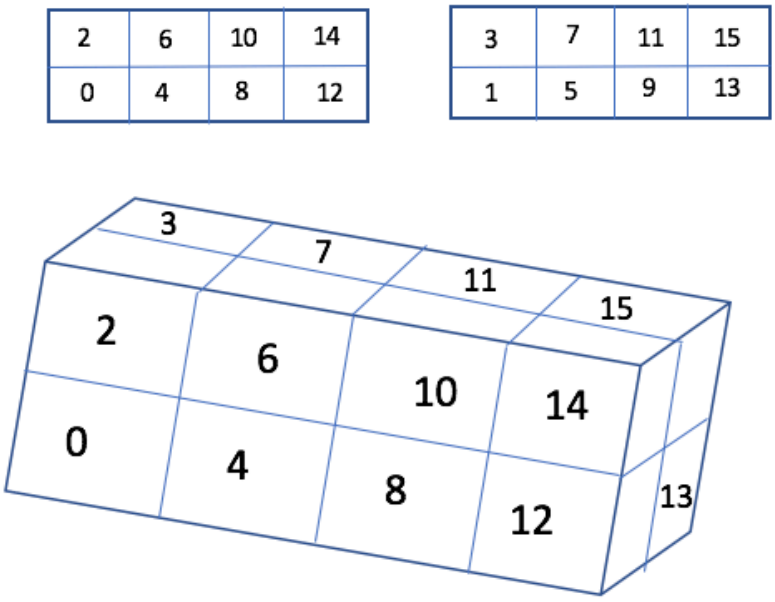


Fig. 3.3.5 Arrangement of ranks of cores in a 3D Cartesian topology

These processes at the individual interfaces will now make local groups at interfaces. These local interface ranks' groups are denoted with handle 'INF_GROUP'. This way each Cartesian communicator will have one interface group on the face side which has block interface. And these 'INF_GROUP' processes will be used to create respective intra – communicator among themselves for the interface they belong. The standard sub-group creation operation used here is:

MPI_GROUP_INCL(CART_GROUP, N_CORES, GROUP_RANKS(1:N_CORES),
INF_GROUP(NBL,ISIDE_), IERR)

Creates an MPI sub group from an existing group of processes

CART_GROUP: group of all cores of the Cartesian communicator
N_CORES: number of cores which needs to be included in the new group
GROUP_RANKS: ranks (CART_RANK) of the cores which needs to be included in the new group
INF_GROUP: the new group handle
IERR: error indicator

MPI_GROUP_INCL operation creates a sub-group from an existing group. As mentioned earlier a group is always attached to a communicator and the processes which are part of a group have their ranks ordered as in parent communicator. For sub-group creation we need the total number of processes which we want to include in the new group, this is the second argument in the group include operation. The third argument is an integer array which has the ranks of the processes which are to be included in new group. First argument is the input parent group handle, and, the fourth argument is the output sub-group handle. Fig. 3.3.6, on left, shows the local Cartesian ranks (CART_RANK) within each Cartesian communicator, and on the right

are just the ranks which are at the interfaces and which are selected to create the interface groups. Thus, each Cartesian communicator has different local ranks at the different interfaces, and these interface processes will alone take part in the formation of inter-communicators.

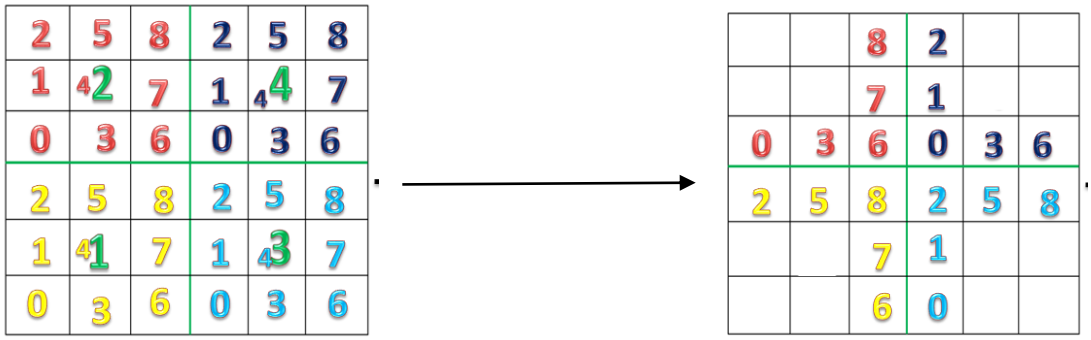


Fig.3.3.6 Selection of those Cartesian communicator ranks which are at the interfaces

IV. Interface Intra-communicator

Main Objectives:

a) Creating interface intra-communicators from interface MPI groups

b) Assigning the local and remote leaders for each ISIDE

In subroutine INTERFACE_GROUP, we prepare another important variable named - 'LOCAL_LEADER', which is required to make inter-communicators. It is an argument in the inter-communicator creation operation – 'MPI_INTERCOMM_CREATE'. This MPI operation requires two intra-communicators, with each having one of its processes as the local leader. The local leader of the remote group is the remote leader for any MPI communication. The rank of this local leader process, along with the remote leader rank, is used to make the communication link between the two intra-communicators. As in MPI, a communicator has an independent communication context, and it can't communicate with other communicators on its own. We decided that the highest ranks of the interface groups will be the local leader rank in respective interface intra-communicator. We can take any process as the local leader and there is no constraint on that. While preparing the interface groups, we made the local leaders such that each process knows the rank of its local leader process.

```
MPI_COMM_CREATE_GROUP(CART_COMM, INF_GROUP(NBL,ISIDE_),  &
                      TAG_IG(NBL, ISIDE_), INF_COMM(NBL,ISIDE_),IERR )
```

Create an intra-communicator from an existing group of processes within a communicator

CART_COMM: communicator handle to which the interface group belongs
INF_GROUP: the group handle for the processes at the interface
TAG_IG: tag used to differentiate multiple calls by same processes
INF_COMM: new communicator handle which is created with interface processes
IERR: error indicator

Subroutine ‘INTERFACE_INTRA_COMMS’ creates intra-communicators from the interface groups created previously. These groups of processes at interfaces are now directly used to create interface intra-communicators with the standard MPI operation - ‘MPI_COMM_CREATE_GROUP’. The first input argument is the parent communicator from which a new intra-communicator is to be created; in our case it is the Cartesian communicator. The second argument provides the handle of the sub-group of the processes which will belong to the new communicator. ‘INF_GROUP’ is the required sub-group of interface processes.

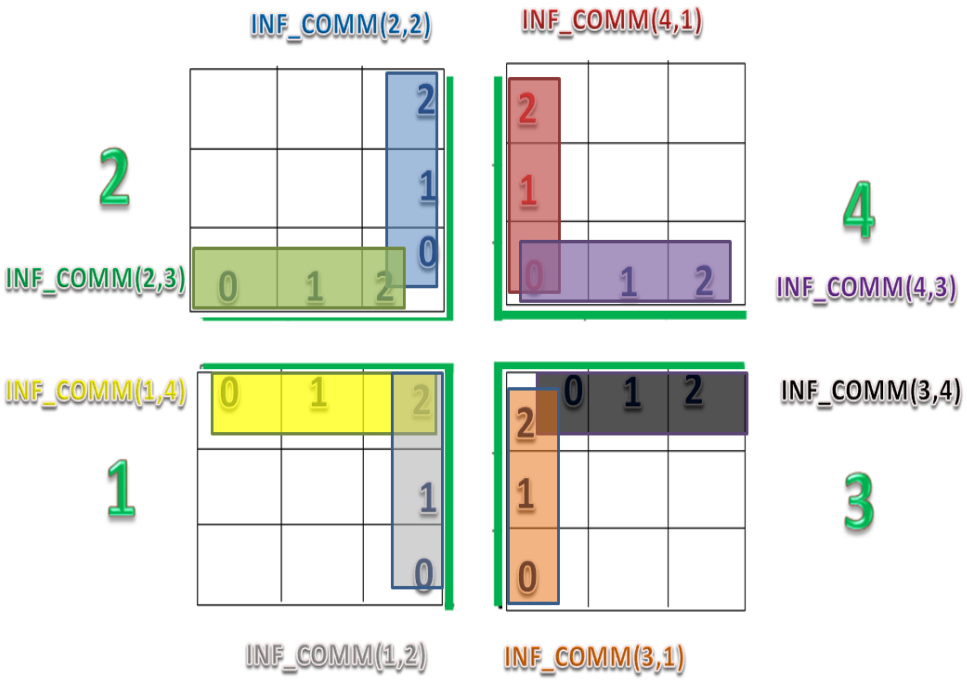


Fig.3.3.7 Different interface intra-communicators at the various interfaces of the grid blocks.

Tags for the creation of these new communicators have to be prepared properly, because there are some processes which will belong to more than one new communicator. And, to create distinct communicators they need distinct tags. The output argument in our case is the INF_COMM handle, which is the newly created intra-communicator. We used a 2D array for this new communicator handle here. The first index of the array is the identification of the block and the second index represents the direction to which this interface communicator belongs to. Fig. 3.3.7 shows all the different intra-communicators at respective interfaces. These are new communicators and therefore their member processes will again have local ranks starting from zero, as illustrated in Fig. 3.3.7.

Subroutine ‘INTERFACE_INTRA_COMMS’ also prepares the variable ‘REMOTE_LEADER’ for each interface intra-communicator. As summarized in the introduction of this chapter, each inter communicator has two groups of processes: local group and remote group. These two groups of processes are mandatorily disjoint from each other, meaning that these two groups cannot have common processes between them. An overlap of local and remote groups that are used to make an inter communicator is prohibited [2]. Such an

overlap of the processes is erroneous and could lead to a deadlock situation. Both the groups belong to respective intra-communicators which are located at either side of a block interface.

V. Interface Inter-communicator

Main Objectives:

a) Preparing the 'TAG' argument for all the interface processes for creating inter-communicators

b) Preparing the 'INTER_COMM' array for inter-communicator handle

c) creating the inter-communicators

Within an inter-communicator context, the 'LOCAL_LEADER' of a group is the 'REMOTE_LEADER' of the corresponding remote group. However, it is very important to note that the 'LOCAL_LEADER' variable is the rank of the process in the intra-communicator (which is a local rank); but the 'REMOTE_LEADER' is the rank of the process in a peer communicator (global communicator in our case) to which the leaders of both the groups belong to. In other words, there must exist a peer communicator to which the local leader and its corresponding remote leader belong, and they must know each other's rank in this peer communicator. Fig. 3.3.8 shows the local ranks of interface processes and their ranks in the global world communicator which is the peer communicator for us.

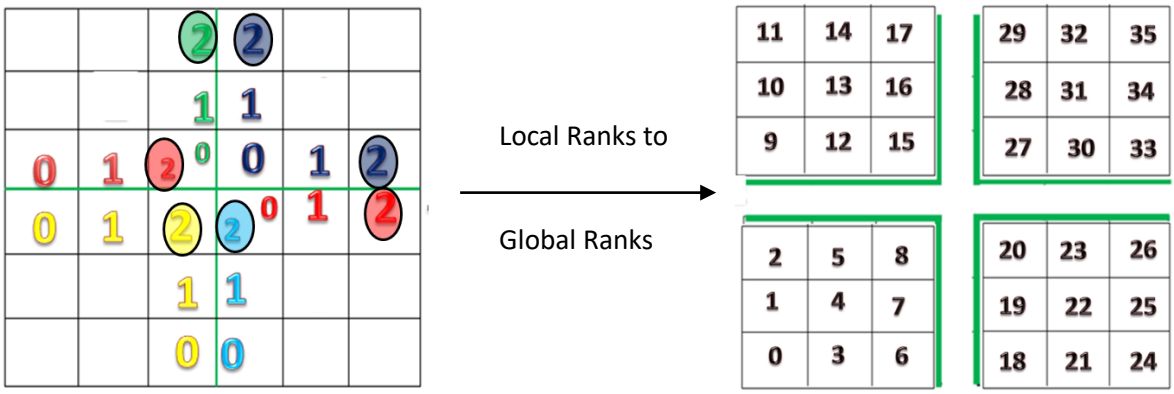


Fig.3.3.8 Local interface intra-communicator ranks with local leader ranks circled, and the respective global world communicator ranks of all the processes

In standard practices, the default MPI_COMM_WORLD or its duplicate communicator is usually used as the peer communicator. This peer communicator provides a communication context in which both the leader processes can communicate with each other. It is also mandatory that all the other member processes of these interface intra-communicators know the ranks of their local leader and the remote leader. It is required because all the interface processes have to call the inter-communicator creation function to create their respective inter-communicators, and the leader ranks are input arguments for that MPI function.

Now, we have mentioned that the local leader ranks were assigned during the interface group creation and all the interface processes know their local leader's rank. However, the remote

leader rank is not yet provided to processes. The remote leaders are members of the remote group, which is a group of MPI processes in a different intra-communicator (precisely the communicator on the other side of block interface). And, as mentioned earlier there is no direct communication link between two intra-communicators, and, this is the sole reason why we need to create inter-communicators. However, here the remote leader rank must be transferred between the processes of two different communicators even to create the inter-communicator between them. This remote leader rank is the rank of remote leader process in the peer communicator, not its local rank in the intra-communicator.

These two tasks are managed in subroutine INTERFACE_INTRA_COMMS. First, we try to find the remote leader rank in peer communicator. We have mentioned that with MPI_COMM_RANK function a process can assign its rank in a particular communicator to a variable. This function was used to assign the CART_RANKs and INF_RANKs of processes in their Cartesian and interface intra-communicators. We also note that local leader ranks were stored in LOCAL_LEADER(NBL, ISIDE) array, whose first index is the geometrical block ID and the second index corresponds to the direction of the interface. Similarly, we allocate a two-dimensional array for the remote leader rank – REMOTE_LEADER(GEOM_BLOCKS, 6).

We run a loop on all geometrical blocks and their interface directions to get the array values for LOCAL_LEADER(NBL, ISIDE). At each ISIDE we separate the local leader process and make it call the function MPI_COMM_RANK to get its rank in the peer communicator. The peer communicator rank of local leader is stored in array REMOTE_LEADER with first index referring to neighbor block ID (NBR) and second index referring to ISIDE value of neighbor block (NBR_SIDE) for the respective direction. Corresponding code lines are provided in Fig. 3.3.9. Note that only the relevant code lines are provided here to explain this task.

```
872  DO NBL=1, GEOM_BLOCKS
873    DO SIDE_=1, INF_REGIONS(NBL)
874     ! local_leader is the inf_rank of leader of inf_comm
875         IF(INF_RANK(NBL,ISIDE_)==LOCAL_LEADER(NBL,ISIDE_)) THEN
876           CALL MPI_COMM_RANK(MPI_COMM_WORLD, MY_WORLD_RANK, IERR)
877           REMOTE_LEADER(NBR,NBR_SIDE) = MY_WORLD_RANK
878           OPEN(UNIT=20, FILE='REMOTE_LEADER.dat',STATUS='OLD',POSITION= "APPEND",ACTION='WRITE')
879           WRITE(20,*) MY_WORLD_RANK
880           CLOSE(20)
881         END IF
882    END DO
883  END DO
```

Fig.3.3.9 Code lines for finding remote leader rank in peer communicator

Along with storing the required remote leader rank in a suitable array position, this rank is also written in a text file 'REMOTE_LEADER.dat'. Keep in mind that this work is done by the local leader alone. As we work in distributed memory framework, the job done by individual process is known only to itself unless it is transferred to others. So, until now the values filled in REMOTE_LEADER array are only known to the local leaders who filled this array in their

own memory. We have to transfer this remote leader array to all the interface processes. We have the remote leader ranks written in the text file, which is available to all processes. This file was written for the sequence from block 1 to GEOM_BLOCKS and within block index we followed the sequence of respective ISIDEs. Now, all the processes should open and read this file in the same sequence for NBL and ISIDE. Each process will store the values of this file in the variable REMOTE_LEADER with the relevant indices for block and direction values as shown with code lines in Fig. 3.3.10.

```
902     OPEN(20,FILE="REMOTE_LEADER.dat",STATUS='OLD',ACTION='READ')
903       DO NBL=1 , GEOM_BLOCKS
904         DO SIDE_=1, INF_REGIONS(NBL)
905
906               READ(20,*) REMOTE_LEADER(NBR,NBR_SIDE)
907         END DO
908       END DO
909   CLOSE(20)
```

Fig.3.3.10 Storing the remote leader value done by all processes at interface

We used the MPI operation – 'MPI_INTERCOMM_CREATE' to create inter communicators at the block interfaces. The operation call details are as follows:

MPI_INTERCOMM_CREATE(INF_COMM(NBL, ISIDE_), LOCAL_LEADER(NBL, ISIDE_), &
PEER_COMM, REMOTE_LEADER(NBL,ISIDE_), TAG, INTER_COMM(ID), IERR)
Creates an inter-communicator with two existing intra-communicators

INF_COMM: the parent intra communicator handle which is used to create inter communicator
LOCAL_LEADER: local leader rank in intra communicator
PEER_COMM: common communicator handle to which both leaders belong
REMOTE_LEADER: remote leader rank in peer_comm
TAG: tag to differentiate calls for different inter communicators
INTER_COMM(ID): output inter communicator handle
IERR: error identifier

Again, all the processes which are at the block interfaces will make the above call to create their corresponding inter-communicators. These processes have to provide the five arguments as the input for the inter-communicator creation function. The 'PEER_COMM' is same for all the processes, however, the other four arguments depend on the block ID and the ISIDE values. By this, we make sure that all the processes which are at a particular interface provide the same values for these input arguments for the combination of NBL and ISIDE. The values for 1st, 2nd and 4th arguments have already been computed above for all the processes, now we prepare the 'TAG' argument.

In inter-communicator creation, it is very important that the tag argument is well set. Here, all the processes of the local and remote groups will make the above call to create their respective

inter-communicators. The processes of one particular local group and the corresponding remote group must all give exactly the same value for the tag argument to create their inter-communicator. Only the processes of the local group and the respective remote group which provide the same tag value will create an exclusive inter-communicator among themselves.

In Oracle3D, we need to create the inter-communicators dynamically depending upon the present blocks in the grid, and thus this tag should also be generated dynamically according to the base grid. It is also important to prepare this tag carefully because, firstly, there are always some processes which belong to two or more interface intra-communicators. And secondly, while creating inter-communicators for a particular ISIDE these processes must have a particular tag for that inter-communicator. Now, we know that there is always only one local leader and a corresponding remote leader for all the processes of an interface. The corresponding ranks of these two leader processes are known to all the processes of local and remote groups. And these ranks are specific for a particular interface alone. Thus, it was decided that the multiplication of the ranks of local and remote leader will be used as the tag for the corresponding interface. This will always be specific to one particular interface and this will also be the same for all the local and remote processes of that inter-communicator. Thus, we chose to have tag as follows:

$$TAG = REMOTE\_LEADER(NBL, ISIDE\_) * REMOTE\_LEADER(NBR,NBR\_SIDE)$$

```
943      L = 0
944     DO NBL = 1, GEOM_BLOCKS
945       DO SIDE_=1, INF_REGIONS(NBL)
946           ISIDE_    = MY_ISIDE(NBL,SIDE_)
947           NBR       = NEIGHBOUR(NBL,ISIDE_,1)
948         IF(.NOT.HAVE_INTERCOMM(NBL,NBR)) THEN
949
950               L = L + 1
951               INTERCOMM_ID(NBL,NBR) =  L
952               INTERCOMM_ID(NBR,NBL) =  L
953               HAVE_INTERCOMM(NBL,NBR) = .TRUE.
954               HAVE_INTERCOMM(NBR,NBL) = .TRUE.
955         END IF
956     END DO
957   END DO
```

Fig.3.3.11 Creating the Inter-COMM array

Following the similar requirement as the 'TAG' argument, all the member processes of both the groups of a particular interface must also provide the same handle for the new inter-communicator, because they combinedly make this new inter-communicator exclusively among themselves. It was achieved by utilizing an integer array (INTER_COMM) for inter-communicator handle. This INTER_COMM array was prepared by all interface processes before making the call to create their inter-communicator. Within a loop on blocks and ISIDES, all the processes, on the two sides of a particular interface, were assigned a same integer identifier for the common inter-communicator. And, these identifiers were used to create the array INTER_COMM to keep it same for the processes of only a particular interface. The

creation of this inter communicator array should easily be understood by following the loop in subroutine INTER_COMMS as shown in Fig. 3.3.11.

Another important aspect was to keep same number of processes on both sides of an interface. In our multi-block structured grids, the number of control volumes on both sides of a particular block interface are always the same and each control volume has a common cell interface with the neighbor control volume of the neighbor block. Thus, on the interface the grids on both the sides are exactly the same. The idea of this grid level symmetry at block interfaces was used while decomposing the blocks into sub-domains also. We decided that at each block interface the number of sub-domains (subsequently the MPI processes) on both sides will be the same. And, a sub-domain will have an exclusive interface only with a single sub-domain on the other side of interface as shown in Fig. 3.3.12 (a).

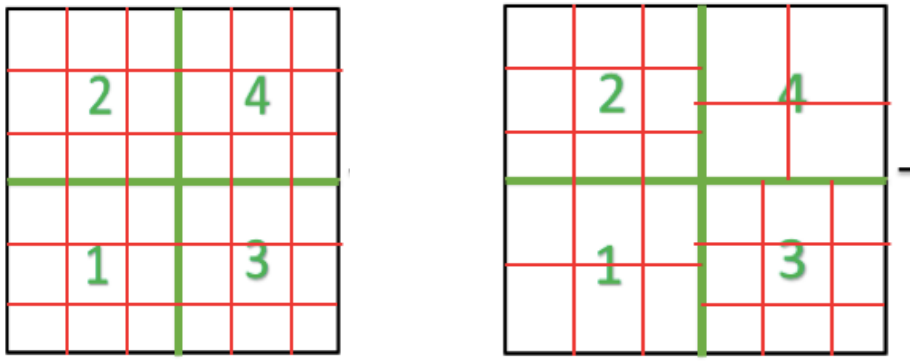


Fig. 3.3.12 a) Symmetric and b) asymmetric decomposition of grid blocks at block interfaces

This symmetry of sub-domains at interface facilitates one to one data exchange between two neighbor processes on each side of the interface. An asymmetrical arrangement of processes, Fig. 3.3.12 (b), at the interfaces would lead to significantly more time to send and receive data at the interfaces. In an asymmetrical arrangement, a process may need to send and receive data from more than one processes on the opposite side. Managing which control volumes' data to be sent to which processes on the opposite side, and, then which data to be received from which processes requires scanning and copying of big arrays on both sides of an interface. This is an unnecessarily time-consuming task.

Moreover, the data at the interfaces have to be exchanged at each time step of computation and also for each variable making this communication very expensive. Keeping asymmetrical arrangement of processes on interfaces would have given us flexibility in terms of having any number of processes in each Cartesian communicator but it costs us huge amount of time during very large simulations. Therefore, the code was prepared to work with such symmetrical configurations of decomposition at interfaces. Fig. 3.3.13 shows the created inter-communicators at the interfaces of Cartesian communicators, with the local interface ranks of the individual communicators. For facilitating the explanation, the inter-communicator between block 1 and 2 is named as INTER_COMM12 and, similarly, the other inter-communicators are also named as shown in Fig. 3.3.13.

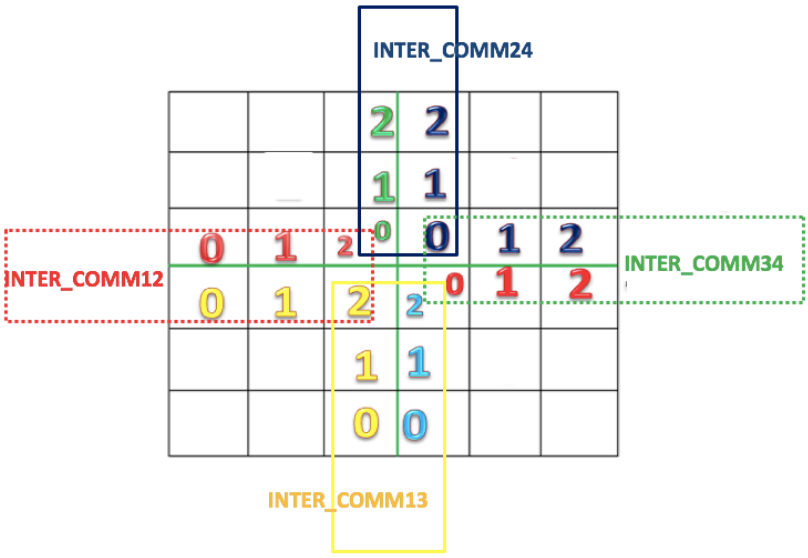


Fig. 3.3.13 Inter-communicators with their member processes on both sides of interface

VI. The NUM_CFI array

Main Objectives:

a) Creating a separate array for managing the interface nodes' data

b) Utilizing the COORDS feature of Cartesian topology to find the interfaces

By creating the required Cartesian and inter-communicators, we have set the parallel environment within which the processes can communicate with each other for message passing. We have two types of interfaces where we have to make the data exchanges: 1) internal sub-domain interfaces within the Cartesian communicators, and 2) block interfaces between two Cartesian communicators. According the location of a process in Cartesian topology, a process may have one or both types of interfaces. In Oracle3D, we have named an array 'NUM_CFI' to keep all the grid related information of nodes which lie at the interface cell centers (INP) and interface cell face centers (IND). Let us consider an example to understand the information contained by the NUM_CFI array. This array is prepared in the code in subroutine called NUM_CFI_SUB_BLOCKS. Fig. 3.3.14 represents a block interface (blue line) common between two blocks which extend in west and east direction of this interface as shown.

Here we are considering only one sub-domain on each side of the blue interface and both sub-domains have four control volumes (grid cells) at this interface. The west side sub-domain has its interface cells' central nodes (INPs) denoted with black dots, and the east side sub-domain has the interface cells' central nodes (INPs) denoted with red dots. In parallel computations each sub-domain is handled by an individual MPI process, so the indices of nodes shown in Fig. 3.3.14 are local to individual sub-domains. The blue dots lying on the block interface are the face central nodes (INDs) of these interface control volumes. Both the sub-domains have their own local numbering for these interface side face central nodes. Here we have taken a random numbering to explain the NUM_CFI array. This NUM_CFI is a 2D array with the 1st dimension having only one element which is the number of that node in the list of interface cell

central nodes. Here we have four interface cells, so this number goes from 1 to 4 for both sub-domains. The 2nd dimension has three elements: INPs, ISIDE and INDs of the corresponding cell central node. Fig. 3.3.14 describes the example shown with the corresponding NUM_CFI array values. This array helps us while making the data exchange at the interfaces.

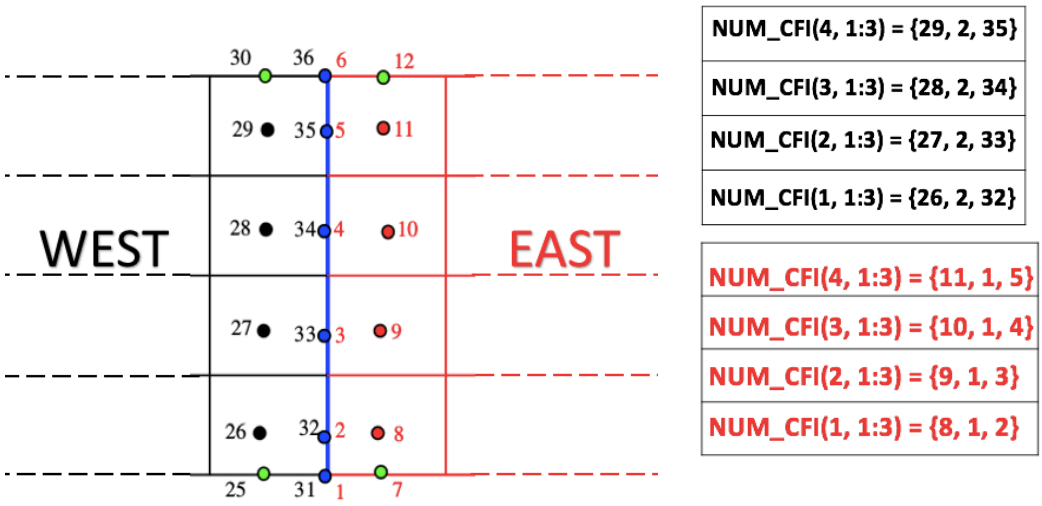


Fig.3.3.14 Example for NUM_CFI array values

We have three types of grid nodes in Oracle3D: the inner nodes, boundary nodes and the interface nodes. And, these different types of nodes are treated differently in the code, according to the underlying mathematics. In this section we discuss briefly the handling of interface nodes.

As mentioned above, each of our blocks have six outer faces with respective directions. In the same way, after the partitioning of grid, all the sub-domains also have six outer faces. Every MPI process takes care of all the nodes of its own sub-domain. Each process has to prepare the interface array NUM_CFI with all the interface nodes it has in all relevant direction. In subroutine NUM_CFI_SUB_BLOCKS each process prepares this NUM_CFI array in two parts. First, the internal interfaces are managed and then the block interfaces. 'COORDS' property of Cartesian topology is used to verify if a process has an internal interface in certain direction or not. After a Cartesian communicator is created with MPI_CART_CREATE, MPI has another operation to assign coordinates to all processes of a Cartesian communicator.

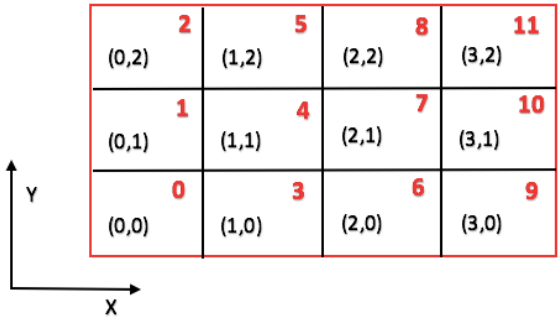


Fig.3.3.15 Example 2D Cartesian topology with COORDS and CART_RANKs

With the operation of MPI_CART_COORDS, each process knows its coordinate in the Cartesian topology. A sample 12- process, 2-D Cartesian topology is shown in Fig.3.3.15. The outer red lines of the block represent the boundaries of the grid block and the internal black lines are the internal interfaces between the sub-domains of the grid, which are created after partitioning of the domain. The domain had one block before partitioning and there are now twelve sub-domains which are assigned to 12 processes of the Cartesian communicator. The red numbers on the top right corners of each sub-domain are the local ranks (CART_RANK) of the processes and in brackets the coordinates of each process are shown, according to its location in the Cartesian topology.

COORDS(1) is X index, and COORDS(2) represents Y index of topology coordinates. Here, the DIMS variable values are 4 and 3 respectively in x and y direction; which means this Cartesian topology has 4 processes in x and 3 processes in y direction. With COORDS and DIMS information a process can easily know if it has an internal interface on a certain direction (ISIDE) or not. For example, COORDS(1) will be equal to 0 for all the processes on the west boundary of the block. Thus, to have an internal sub-domain interface on the west face a process must have COORDS(1) > 0. Because COORDS(1) = 0 represents the west boundary of the block, and there are no neighbor sub-domains on the boundaries. Similarly, COORDS(2) = DIMS(2) – 1 for all the processes on the north boundary of the block. Thus, to have an internal interface on the north face a process must have COORDS(2) < DIMS(2)-1. Rest of the combination for a 3D grid block are shown below:

```
IF(COORDS(1)>0)                ! west interface
IF(COORDS(1)<DIMS(1)-1)        ! east interface
IF(COORDS(2)>0)                ! south interface
IF(COORDS(2)<DIMS(2)-1)        ! north interface
IF(COORDS(3)>0)                ! back interface
IF(COORDS(3)<DIMS(3)-1)        ! front interface
```

With this information the NUM_CFI array for the internal interfaces is prepared. The next step is to see if a process also has a block interface. During, the preparation of Inter-communicators each process was assigned a logical variable ‘INTERFACE_PROC’; for the processes which were at the block interfaces this variable was assigned ‘TRUE’ and rest were given ‘FALSE’. Also, ‘INF_RANK’ which is the rank of a process in its interface intra-communicator was assigned to each process which was at a block interface. With these two variables and grid information regarding the block interfaces and respective ISIDES, we managed to prepare the NUM_CFI array for the block interfaces also.

VII. Data exchange among MPI processes

Main Objectives:

a) making the MPI communication

Throughout the code, this NUM_CFI array is used to manage the information regarding the interface nodes. There are separate subroutines which manage the computations with different variables on the interfaces. This NUM_CFI array is prepared once, just after the creation of the inter-communicators, by each process for its different interfaces, and then used afterwards in the code. There are several instances in the code where we need to make data exchange among the processes at block interfaces. This communication is managed with subroutine 'SWAP_3D'. This subroutine has two sections for data exchange at internal and block interfaces. At the locations where data exchange is required a call to subroutine SWAP_3D is made by all the processes individually. Each process goes through this routine and first manage the internal interfaces and then block interfaces if it has any.

SWAP_3D routine utilizes MPI_SENDRECV for data exchange. The method to use MPI_SENDRECV is same as it was explained for single block cases. In multi-block cases, we have multiple Cartesian communicators for internal communications, and multiple inter-communicators for data exchange at the interfaces. In place of the COMM argument, the process has to use respective communicator handle. The standard MPI_SENDRECV operation as used in SWAP_3D subroutine is as follows:

```
! Message sending and receiving in X direction
MPI_SENDRECV(SEND_WEST, inpW, MPI_RK, NBR_WEST, TAG,
                    RECV_WEST, inpW, MPI_RK, NBR_WEST, TAG,
                     COMM, STATUS, IERR)
```

Making a blocking send and receive with a single function call

```
1233
1234   CALL MPI_SENDRECV(SEND_WEST,SIZE(SEND_WEST),MPI_RK,INF_RANK(NBL,ISIDE_),TAG, &
1235                     RECV_WEST,SIZE(SEND_WEST),MPI_RK,INF_RANK(NBL,ISIDE_),TAG, &
1236                     INTER_COMM(I),STATUS,IERR)
```

Fig.3.3.16 Sample MPI_SENDRECV function used for inter-communicator data exchange

Some additional INTER_COMMUNICATOR functions which could be utilized in Oracle3D on requirement are following:

MPI_COMM_TEST_INTER(COMM, FLAG, IERR)
Tests whether a communicator is inter-communicator or not

MPI_COMM_REMOTE_SIZE(COMM, SIZE, IERR)
Provides the number of processes in the remote group of the inter-communicator

MPI_COMM_REMOTE_GROUP(COMM, GROUP, IERR)
Provides a handle for the remote group of an inter-communicator

MPI_INTERCOMM_MERGE(INTER_COMM, HIGH, NEW_INTRA_COMM, IERR)
Creates an intra-communicator by merging the local and remote groups of inter-communicator

MPI_COMM_SET_NAME(COMM, NAME, IERR)
Associates a name string with a communicator

MPI_COMM_GET_NAME(COMM, NAME, RESULTLEN, IERR)
Returns the name and length of name string , associated with a communicator

3.3.2 Scalability Results with Oracle3D

The whole MPI environment was programmed in Oracle3D, first for a Poisson solver. After the code is ready, two important questions have to be answered carefully: 1) parallel performance and 2) validation. We have prepared a separate chapter (chapter 4) for detailed analysis on the validation of code with the MPI strategy, with different types of solvers on varied physical problems. Here, we confine this part with some preliminary performance tests only. Testing the codes with a multi-block grid problem is the most important thing here, which is the extension from the previously dealt single block grid section.

Fig. 3.3.17 (a) illustrates the 4 blocks grid which was considered for this scalability test. The individual control volumes in each direction for each block (e.g. B1= Block 1) is mentioned in figure. The total number of control volumes for the case was 8 million. Six different cases for different number of MPI processes (16, 32, 64, 128, 320 and 512) were set to analyze the time taken with increasing number of processes. It is very important for comparison that all six cases have exactly the same computational load. So, a very strict tolerance (1.E-25) was set for the linear system solver, so that it is never reached, and the supplied maximum number of iterations are always reached in all cases. The maximum number of iterations were set such that even with 512 processes the communication overhead remains sufficiently lower than the computation time.

As mentioned in single block case, following two values of time taken are noted here also: 1) total time, and 2) computational time. However, the grid management is not completely serial in case of multi-block grids, as we use reader processes for managing the individual grid blocks. It is sufficient to mention here that a Laplace problem, $\nabla^2\varphi(x,y,z) = 0$, was solved, for an SDBD actuator setting. This other details regarding this problem are given in dedicated chapter 7, here we only test the parallel efficiency of the MPI strategy used in the code. Other Poisson problems with different sources and necessary boundary conditions are also described in the next chapter, where provide the validation of the code with these problems. Here we consider φ as the electric potential variable which is essential for our EHD problems. An example 2D solution of our problem is illustrated in Fig.3.3.17 (b).

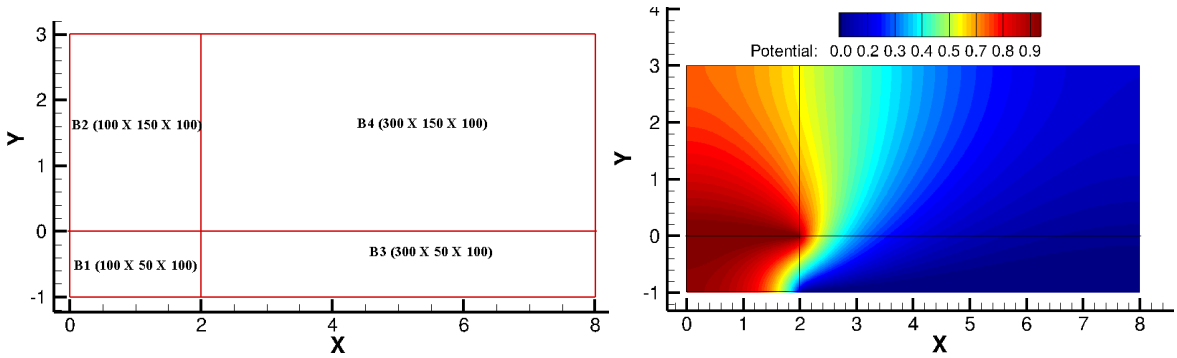


Fig.3.3.17 a) Grid blocks and cell counts (left), b) Solution of Laplace equation on the grid (right)

For the first test, maximum number of iterations were set to $2.1E^{+6}$. The results for this case are shown in Fig.3.3.18. We find that both the total time and computational time decrease with increasing number of MPI processes. The computational time with 16 processes was 1217.47 min which reduced to roughly 22 times (53.5 min) with 320 processes, Fig.3.3.18 (left). The speedup results are a little surprising as we see a super-linear speedup with an increase in the MPI processes count. It is possible to have such super linear speed ups in parallel computations, as reported by many researchers [1,10,11]. First thing we note here that our base solution to compare the speedup is the time take by 16 processes, we showed a speed-up plot for 1 to 16 processes and it gave us desired speedup, Fig. 3.2.4. We also mentioned several studies where 8,10,16,32 or 128 cores were taken as baseline solution assuming these cases give perfect scalability [10-13]. Mavriplis et al. (2005) reported achieving a super-linear speed up of 2395 with 2008 CPUs on single grid, and a speed up of 2250 on 2008 CPUs with four level multi-grid in their case configurations. They explained the super-linear speed-ups with the favorable cache effects in their cases [11].

In parallel computing, when we keep the overall grid size same and keep on increasing the number of MPI processes, the grid size per process decreases. We note here that for all our cases one MPI process was assigned to one physical core. In distributed memory architectures, like ours, each new core has with its own memory (RAM) and cache. Thus, by increasing the number of cores we not only increase the physical computing units (cores) but also the cache and RAM available for the MPI subdomains. In such situations, there may come a time when the grid size (mathematical problem size) per core reduces so much that it can fit into the cache itself and it decreases the latency due to memory I/O. So, the ideal speedup suggests a scenario where we increase only the computing cores and keep problem size fixed, which is not the case in modern HPC systems.

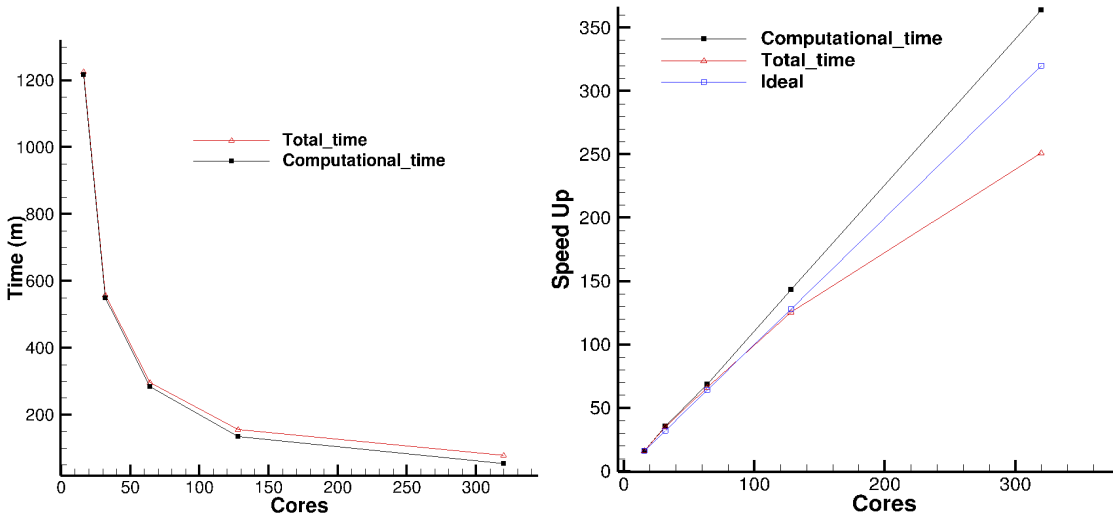


Fig.3.3.18 Perforamce plots with multi-block grid case on Poisson problem

There are also the effects of some of modern processor technologies which are embedded in the processors. In our case, we used the *Intel Xeon E5-2680 V2* processors, which have with following technologies by Intel: 1)*Turbo Boost technology 2.0* , 2) *Hyper-Threading*, 3) *Enhanced Inel SpeedStep* etc. Intel Turbo Boost technology automatically accelerates the processor performance for peak loads if they are running below the power, current and temperature specification limits. It does it by allowing the processors to run above the rated operating frequency below the specification limits. Xeon E5-2680 V2 processor has base frequency of 2.8 GHz and the maximum turbo frequency is 3.6 GHz. The maximum turbo frequency is the highest possible frequency achievable when the working conditions are suitable [4].

Several factors are dynamically considered by the processor algorithms to check whether to enter into the turbo boost mode or not. These factors include workload, processors temperature, numer of active cores per node, software, other supporting hardware, overall system configurations etc. Intel has multiple parallel algorithms to manage the working parameters of processors and the above mentioned factors do not present an exhaustive list [4]. Thus in summary, this technology gives the user a burst of speed by taking advantage of favourable factors when it is needed, and in the other cases an increase energe efficieny is maintained.

With Hyper-Threading technology each physical core of our processor can work as two logical cores, but this depends on user to provide a multi-threaded job to the processor. It was not done in our code. Enhanced SpeedStep technology works on the demand based switching of applied voltage and frequency. It works as a power management technology which keeps the applied volatge and clock speeds to minimum necessary level until a boost in efficieny is not required. It also works in tandem with other processor technologies like turbo boost technology to provide anhanced efficieny when required. These all factors with favorable cache effects explain the super-linear speed ups. For more details on these processor technologies the reader is suggested to refer the corresponding Intel manuals for the mentioned processors.

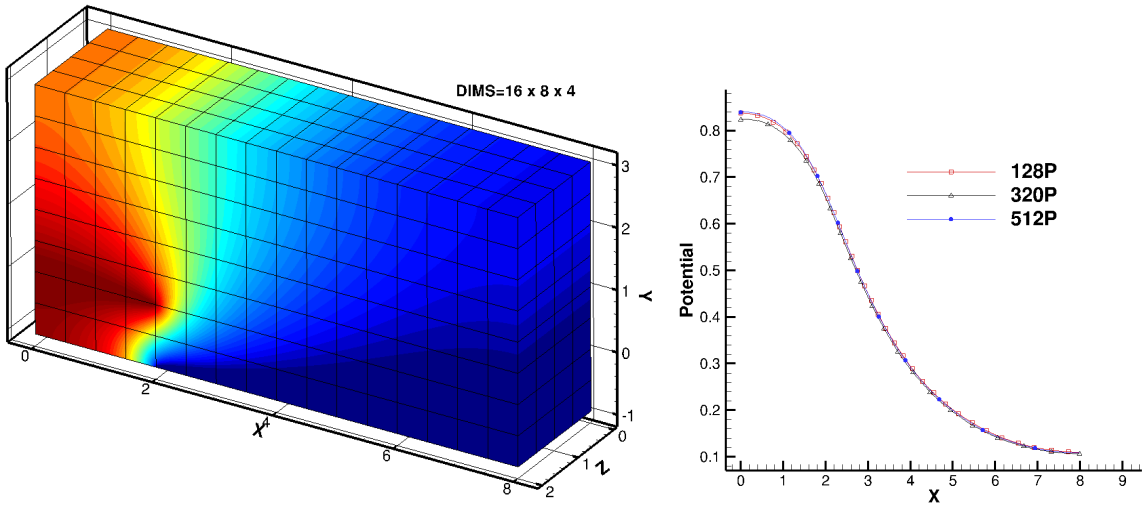


Fig.3.3.19 a) Iso-metric view with 512 cores, b) comparison of multi-cores solution along a section in XY palne at Y=1.

The first guess on maximum number of iterations was found to be not sufficient for 512 processes to overcome the communication head. For this case, the solution time with 512 processes was noted to be much higher than the time taken by 320 cores. So, we decided to make a run with 5 times the number of maximum iterations with only 512 processes. We decided to extrapolate the values of time with the other sets of processes (16,32 etc.) for the same number of maximum iteration as were taken for 512 processes, such extrapolation of time values for higher number of iteration is justifiable as the computations performed per iteration are exaclty the same and the base number of iteration to extrapolate the values is 2.1 million which is enough to get an average.

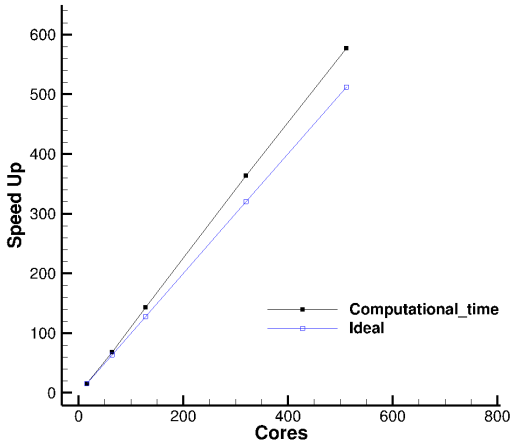


Fig. 3.3.20 Estimated appoximate speedup up to 512 cores

Fig. 3.3.19 (a) illustrate the decomposed domain with 512 processes having the solution of our Laplace equation, in total we have 16 processes along X, 8 processes along Y and 4 processes along Z direction, making the simulation to be completely parallel in 3D. A sample validation along the section Y=1, in XY plane, shown in Fig. 3.3.19 (b) assures that the soultion remains

same with number of processes. The approximated speedup obtained with 512 processes for higher work load is provided in Fig.3.3.20, showing a perfect scalability on our HPC cluster.

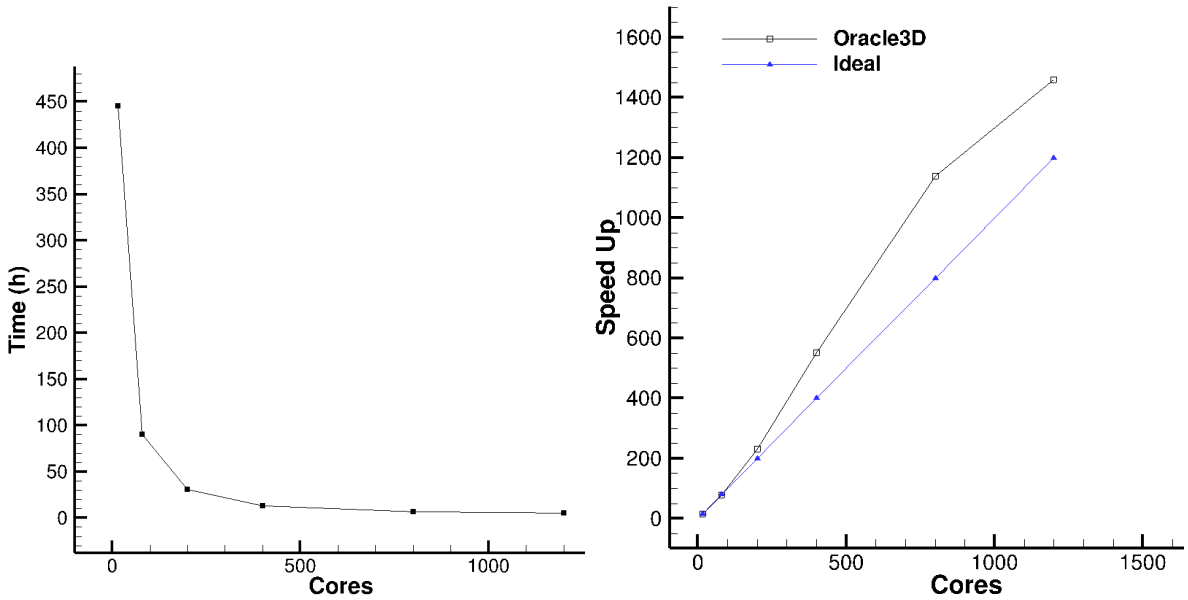


Fig.3.3.21 Perforamce plots: a) scalability , b) speed up with Navier-Stokes solver

We also performed a scalability test with a Navier-Stokes problem. We solved the classical lid-driven cavity problem with a four symmetrical grid blocks geometry. The whole grid has 8 million control volumes which were equally distributed in four geometrical blocks (2 million cells each block). The number of iterations were set such that the problem was sufficient for up to 1200 cores. Results show that in this case also the code achieves super-linear speed up when the number of processes go beyond 500, Fig. 3.3.21. This problem was more symmetric then the Laplace problem mentioned above as the 4 grid blocks were of same size, so the workload on the 4 grid reader processes was also balanced equally. Same computational load was assured by keeping equal grid nodes for each MPI process and exactly the same number of iterations, by setting an unattainable tolerance. The causes of super-linear speed up should be explained as detailed above.

It should also be noted that for all of these scalability test problems the load balancing among all the processes was managed equally, so each process had same amount of computational work to perform, which is important to achieve high scalability. This assures that there was no waiting for any MPI process at practically any time during the linear system solver computations. With our MPI strategy of cartesian communicators and inter-communicators we made sure that each process knows its 6 neighbours on 6 directions easily, and the communication link with all the neighbours was set in the beginning. At each time step, every process calls the communication subroutine and this does the two way communication (send and receive) without any delay. After achieveing the desired parallel performance we will carry out the validation of various solvers in next chapter.

Bibliography

[1] W. Gropp, E. Lusk, A. Skjellum, "Using MPI: Portable Parallel Programming with the Message-Passing Interface," 2nd edition, The MIT Press, (1999)

[2] MPI: A Message-Passing Interface Standard, Version 3.1, (2015)

[3] A. Vladimirov, R. Asai, V. Karpusenko, "Parallel Programming and Optimization with Intel Xeon Phi coprocessors," Colfax International, 2nd Edition, (2015)

[4] www.intel.com

[5] www.idris.fr/formations/mpi

[6] https://comptuting.llnl.gov/tutorials/mpi

[7] https://www.mpi-forum.org

[8] M. J. Berger, M. J. Aftosmis, D. D. Marshall, S. M. Murman, "Performance of a new CFD flow solver using a Hybrid Programming Paradigm," Journal of Parallel and Distributed Computing 65, 414-423 (2005)

[9] W. D. Gropp, D. K. Kaushik, D. E. Keyes, B. F. Smith, "Analyzing the Parallel Scalability of an implicit unstructured mesh CFD code," 7th International conference High Performance Computing- HiPC, 395-404, (2000)

[10] M. Aftosmis, M. Berger, R. Biswas, M. J. Djomehri, R. Hood, H. Jin, C. Kiris, "A detailed performance characterization of Columbia using Aeronautics Benchmarks and applications," 44th AIAA Aerospace Sciences Meeting and Exhibit, (2006)

[11] D. J. Mavriplis, M. Aftosmis, M. Berger, "High Resolution Aerospace applications using the NASA Columbia Supercomputer," Proceedings of the ACM/IEEE Conference on Supercomputing, (2005) [DOI: 10.1109/SC.2005.32]

[12] H. Jin, R. F. Van der Wijngaart, "Performance Characteristics of the Multi-zone NAS Parallel Benchmarks," Journal of Parallel and Distributed Computing 66, 674-685 (2006)

[13] F. C. Wong, R. P. Martin, R. H. Arpaci-Dusseau, D. E. Culler, "Architectural Requirements and Scalability of the NAS Parallel Benchmarks," Proceedings of ACM/IEEE Conference on Supercomputing, (1999) [DOI: 10.1109/SC.1999.10044]

[14] F. Moukallel, L. Managani, M. Darwish, "The Finite Volume Method in Computational Fluid Dynamics," Springer International Publishing, Switzerland (2016)

Chapter 4

Validation: Parallel Oracle3D

The detailed MPI strategy discussed in previous chapter was implemented in Orcale3D within the various individual solver versions of the code. As mentioned previously, the code was transformed in the 'module' structure of FORTRAN 90, to make it more organized and oriented in the modern Fortran framework. In the same manner, we have also prepared the individual versions for specific solvers like: Pure Navier-Stokes solver, Poisson solver, general transport solver, 3 species plasma solver and then the complete Oracle3D. This arrangement of code in smaller individual versions helped immensely in transferring the MPI methodology into the full version of code. And also, each smaller version presented us with the opportunity to validate and test the performance of the parallel implementation in different mathematical models. This chapter presents all the validation tests of the final version of various solvers which were parallelized. Thus, this chapter provides the very initial simulations with the developed code and build the trust with verification cases for the users of Orcale3D.

Validation of any new development in the code is a crucial step before the intended final applications. This development started with the base line Fortran 77 version of Oracle3D, which has been previously validated, and several studies have been published with it [1-5]. The development started with the conversion of Fortran 77 version of the code to the Fortran 90 version. At each step of this conversion process the new development was verified by comparing with the results of previous version. The major stages of this conversion include implementing the Fortran 90 features like: implicit none, dynamic allocation of arrays, vectorization of loops, modules and several other. The final Fortran 90 versions of the different solvers were validated, however, here we will present only the validation results with the parallel versions of the solvers.

4.1 The Parallel Poisson Solver

The MPI methodology, as detailed earlier, was first tested rigorously with some very simple integer data exchanges at the two kinds of interfaces, in several combinations of blocks and sub-domains. After the preliminary testing, the MPI strategy was first implemented in a Poisson solver. The well-known Poisson's equation is a partial differential equation of elliptic type, which does not have an unsteady term. Here we use the Finite volume methods to discretize the Poisson equation in space. Equation (4.1) is the general form of Poisson's equation:

$$\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}+\frac{\partial^2}{\partial z^2}\right)\varphi(x,y,z)=f(x,y,z) \tag{4.1}$$

$$\nabla^2\varphi(x,y,z)=f(x,y,z) \tag{4.2}$$

Here φ is a scalar variable and f is the source term for the Poisson's equation. The Laplacian operator ∇^2 is also denoted by Δ, making the equation $\Delta\varphi = f$. A Poisson's equation solver is a good candidate to test the working of our parallel strategy. There are several resources available for the sample parallel Poisson solvers, to start with [6-9]. For us, it also important to note that Oracle3D is mainly an electrohydrodynamic solver, where we always deal with electrical charges. And, in Maxwell's equations we solve the electric potential due to the distribution of charge with a Poisson's equation. Consequently, we decided first to prepare the parallel version of a Poisson solver in the framework of Oracle3D. And then, only after validating this parallel Poisson solver the parallel strategy would be implemented in other solvers.

At first, the parallel Poisson solver was developed for single block grids using the Cartesian topology features. This single block version resulted in very good scalable code on parallel cluster 'THOR' at Institut Pprime. The scalability and speed up of the code were measured with different number of processes, and desired performance was obtained, as reported in Chapter III. The strategy of parallelizing with Cartesian communicator proved very efficient in our single block grids cases. This led us to decide that even in multi-block geometries the data exchange inside the individual grid blocks should be done with Cartesian topology features. And, for the data exchange at the grid block interfaces the Inter-communicators were used. After completing the development and some initial testing, we have to strictly validate the code and check the parallel performance. The Poisson's equation being an independent partial differential equation has the analytical solution readily available to validate the simulated results.

4.1.1 Poisson solver validation on distorted grids

The parallel Poisson solver also presented an opportunity to verify the implementation of Improved Deferred Correction (IDC) scheme with the new developments in code. IDC scheme is a finite volume discretization technique, especially developed, in our group at Institut Pprime, for discretization of diffusive fluxes on highly skewed grids [1-2]. Published references are available in which the IDC scheme was introduced and explored with various skewed grids. The results available in references were obtained with 2D version of our Fortran 77 code. So, to validate the newly developed MPI Poisson solver, we chose the same Poisson's equation as used by *Traore et al. (2009):*

$$\Delta\varphi = 2\pi^2\left(cos^2(\pi x) - sin^2(\pi x)\right) + 2\pi^2\left(cos^2(\pi y) - sin^2(\pi y)\right) \tag{4.3}$$

$$\varphi = sin^2(\pi x) + sin^2(\pi y) \tag{4.4}$$

We chose the same three grids as found in *Traore et al. (2009),* I) orthogonal grid , II) highly skewed anisotropically distorted grid, and III) randomly distorted grid. Grids II and III provide very stiff configurations to test IDC scheme and also to test the domain decomposition method in parallel code. The grids as shown in Fig. 4.1.1 were created, which have 51 control volumes each in both x and y directions. So, the square domain [1 × 1] has 2601 control volumes in total. The analytical solution of the Poisson equation, eq. 4.3, on these grids when all boundaries of domain are set by eq. 4.4, will be given by the same eq. 4.4. The analytical solution is the surface as depicted in Fig. 4.1.2, where the z axis shows the magnitude of φ.

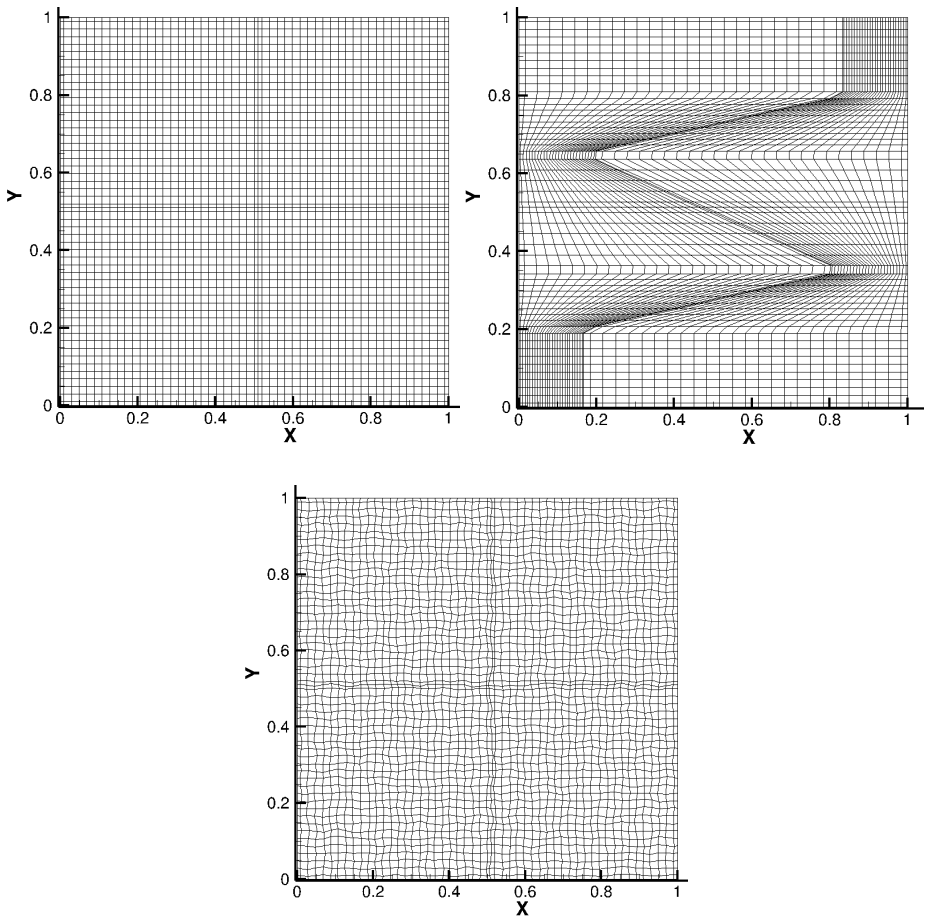


Fig. 4.1.1 Orthogonal, anisotropically skewed and randomly distorted grids, partitioned with 4 sub-blocks.

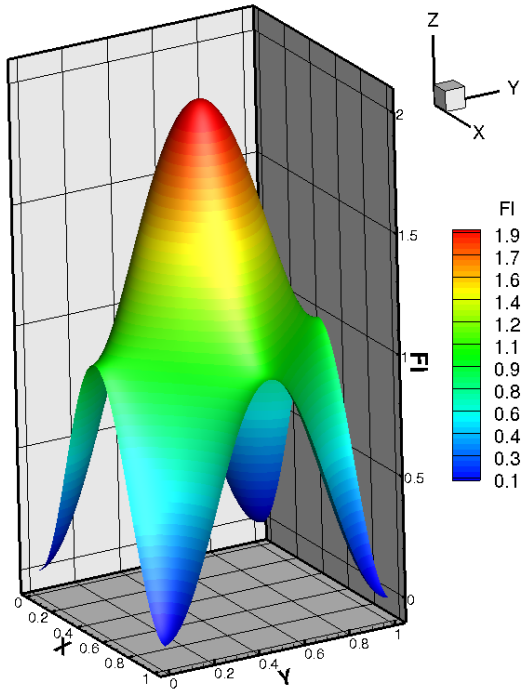


Fig. 4.1.2 Analytical solution surface ($\varphi = sin^2(\pi x) + sin^2(\pi y)$)

Absolute error was computed, with respect to the analytical solution, to check the validity of the parallel code, and the results were compared with the results obtained by the scalar Fortran 77 code [2]. The results in Fig. 4.1.2 should be compared with the results in Fig. 9 of Traore et al. (2009). It was observed that the magnitude of the absolute error was a very good match with the previous results. Figure 4.4 show the analytical solution in 2D and also the solution obtained with the parallel code. The parallel code solution for Fig. 4.1.4 was performed with 33 MPI processes, to check the domain decomposition capability of code in such odd number cells grids (51 × 51), with random number of partitions (11 × 3). An extensive study of the IDC scheme with many other grids, with the scalar version of the code, is available in [10].

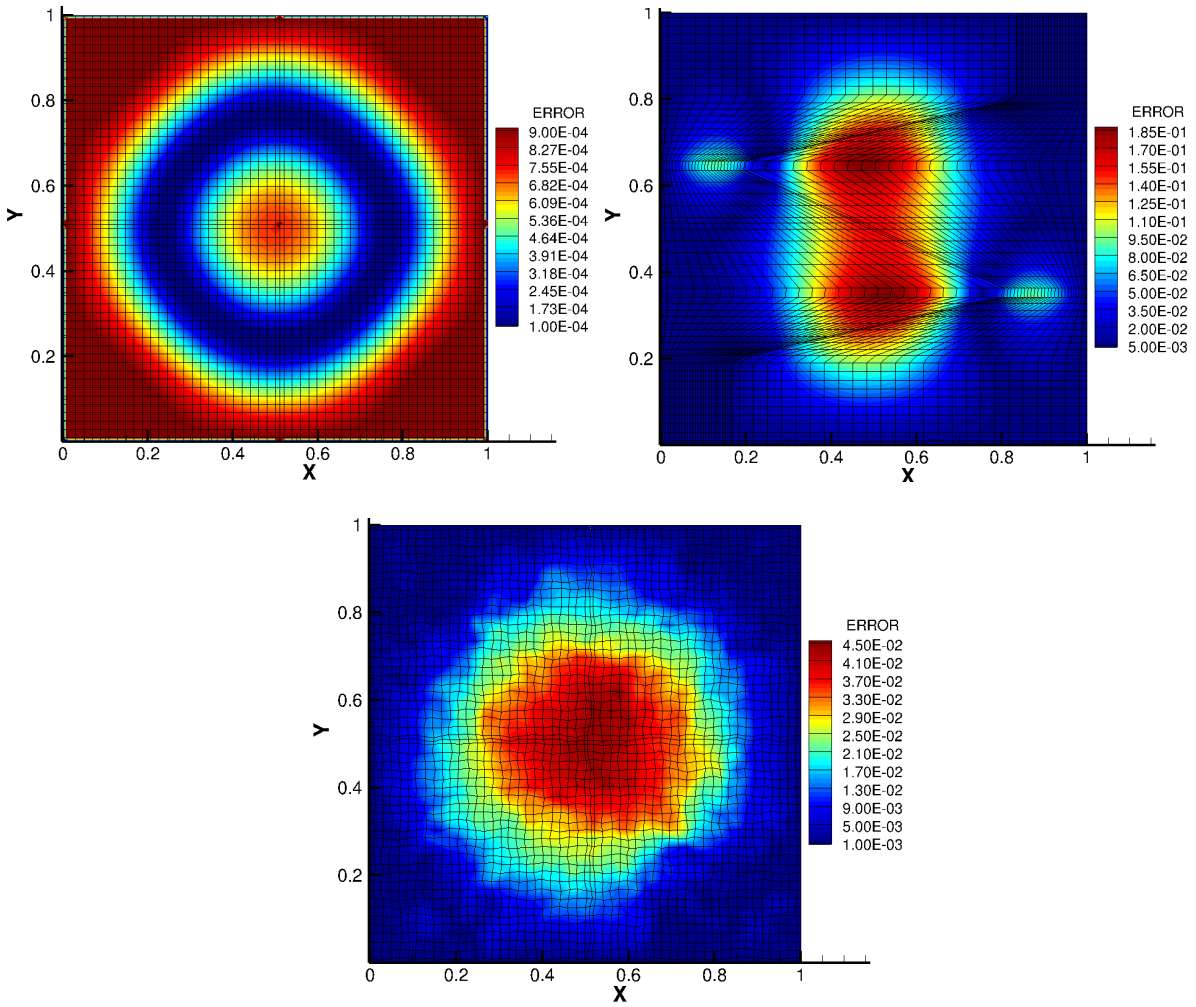


Fig. 4.1.3 Iso-contours of absolute error made with respect to the analytical solution

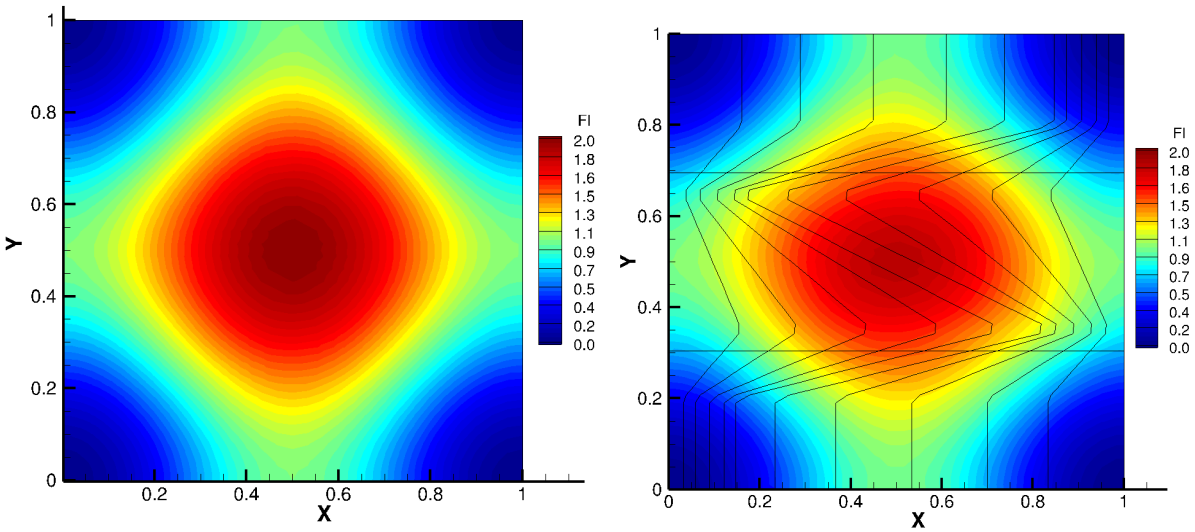


Fig. 4.1.4 The 2D analytical solution on the xy plane, and, solution with grid II performed with 33 sub-domains.

Absolute error with this random domain decomposition of grid was also plotted, Fig. 4.1.5, and the magnitude of error was found to be the same as with the previous solution of scalar code. The black lines inside the square domains in Fig. 4.1.4 and 4.1.5 are the boundaries of the partitioned MPI sub-domains. This exercise proves the accuracy of the parallel methodology used in our code, stating that with random number of MPI processes the solution remains same. It should also be noted here that these grids were fairly coarse, having just 51 control volumes in each direction, and with the refined grids the solution will definitely improve further. In the next test for validation of Poisson solver we have used a 3D grid with 8 million control volumes in total.

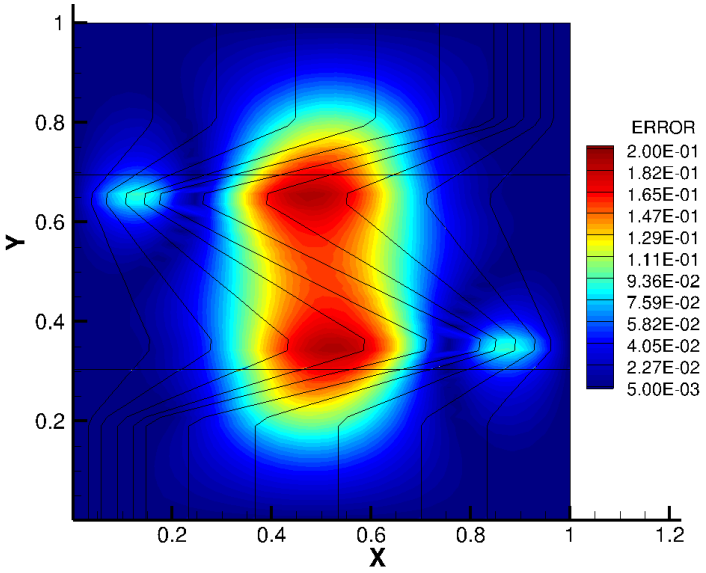


Fig. 4.1.5 Absolute error with a random domain decomposition of grid (11×3*)*

4.1.2 Validation of 3D decomposition

This section deals with some more quantitative validation tests for the parallel Poisson solver in 3D. We have taken grids with multiple number of geometrical blocks to verify the working

of data exchange by the inter-communicators, which are specifically responsible for the MPI communications at the block interfaces. For the first test case we have taken 4 orthogonal grids: I) 1 block, II) 2 blocks, III) 4 symmetrical blocks, and, IV) 4 asymmetrical blocks. Fig. 4.1.6 shows blocks II to IV, where the red lines are the geometrical block boundaries and the thin black lines inside the blocks are the boundaries of the MPI sub-domains after partitioning. Each of the blocks have 8 million control volumes in total, and the distribution of the control volumes is consistent with size of blocks. Complete details on mesh sizes is given in Tab. 4.1.1 below. Fig. 4.1.7 shows an example mesh for the 4 symmetrical blocks case.

Table 4.1.1 Details of mesh for the 4 test cases

Grid	**Geometry**	**Mesh details**
I	1 block	(200 × 200 ×200)
II	2 blocks	Each block: (100 × 200 ×200)
III	4 symmetrical blocks	Each block: (100 × 100 ×200)
IV	4 asymmetrical blocks	B1: (50 × 150 ×200) B2: (50 × 50 ×200) B3: (150 × 150 ×200) B4: (150 × 50 ×200)

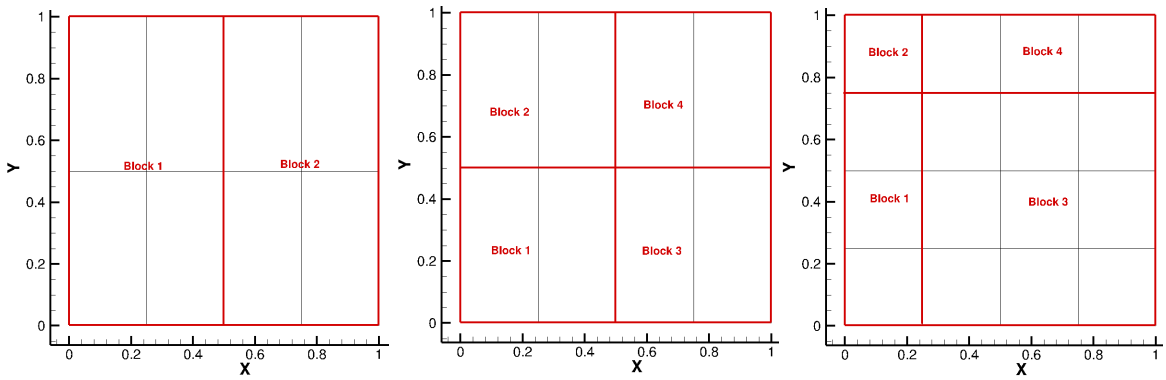


Fig. 4.1.6 The multi-block grids: a) 2 blocks, b) 4 symmetrical blocks, c) 4 asymmetrical blocks

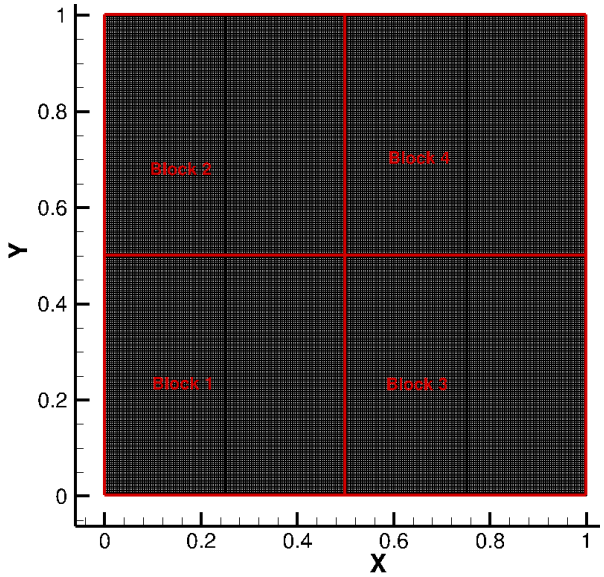


Fig.4.1.7 Orthogonal grid for the 4 symmetrical blocks case, each block: (100 × 100 ×200)

The following Poisson problem was considered for this case:

$$\nabla^2\varphi = 6. \tag{4.5}$$

The boundary condition for this case on all the boundary faces was given by eq. (4.6):

$$\varphi(x,y,z) = x^2 + y^2 + z^2 \tag{4.6}$$

The analytical solution of the eq. 4.5 is given by eq. 4.6. We solved this 3D problem on the 4 test grids mentioned in tab. 4.1. Three plots were drawn along three different sections, and, the solution was compared with the obtained analytical solution. Fig. 4.1.8 show the results when the domain decomposition was performed only in x and y directions. Blocks I and II were solved with 8 MPI processes, and blocks III and IV were partitioned with 16 MPI processes. First plot in Fig. 4.1.8 was taken along line X=0.6 in plane Z=0.6. Second plot corresponds to the solution along line Y=0.4 in plane X=0.6. Solutions were obtained with 8 and 16 procs to check if the solution depends on the number of MPI processes used, and also, because in the 4 blocks asymmetrical case the minimum number of MPI processes for a good load balancing were 16. As mentioned the control volumes in each grid were 8 million, so the grids were taken very fine to avoid the discrepancy due to grid coarseness. The tolerance for the linear system solver was also kept very strict (10^{-9}) to be sure of a good convergence of the linear system residuals.

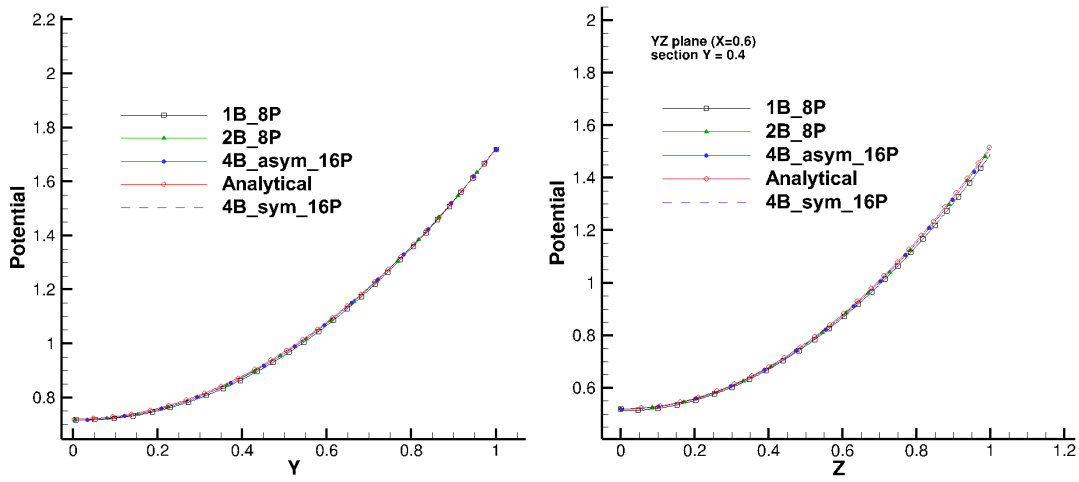


Fig. 4.1.8 Comparison on two planes with 2D decomposition: a) XY plane (Z=0.6), b) YZ plane (X=0.6)

Results show a very good quantitative match among all the MPI simulations and the analytical solution. During the simulations, it was observed that the solution gets better and better with the decreasing tolerance of the linear system solver and by refining the grid sizes. The outcomes of these tests give confidence with the parallel methodology used in our code, both quantitatively and qualitatively. One case was simulated with decomposition in z direction also. The grids III and IV were simulated with 32 MPI processes, having two sub-domains in z direction, and 4 sub-domains each in x and y. Results are shown in Fig. 4.1.9 with the analytical solution. This case also was a perfect match with the analytical solution. A 2D iso-contour for this case is also provided in plane Y=0.85, Fig. 4.1.9 (b), the thin black lines show that there were 2 MPI sub-domains in z direction. To better understand the MPI partitioning, Fig. 4.1.10 depicts two isometric views of the full 3D domain decomposed with 32 MPI Processes, along with the solution corresponding to Poisson equations: $\nabla^2\varphi = 6$.

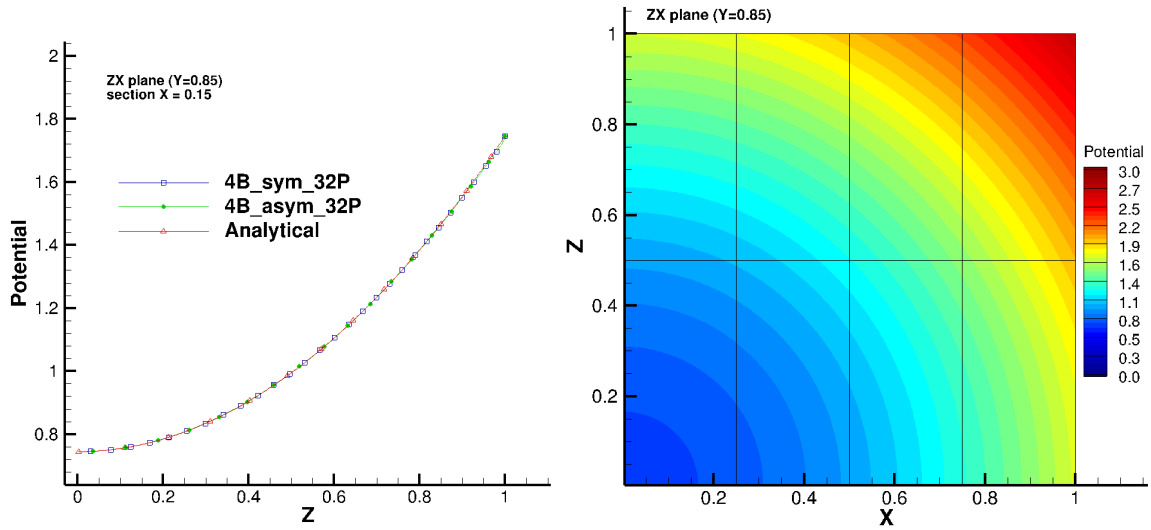


Fig. 4.1.9 a) Solution with 3D decomposition in plane Y=0.85. b) iso-contour in plane Y=0.85

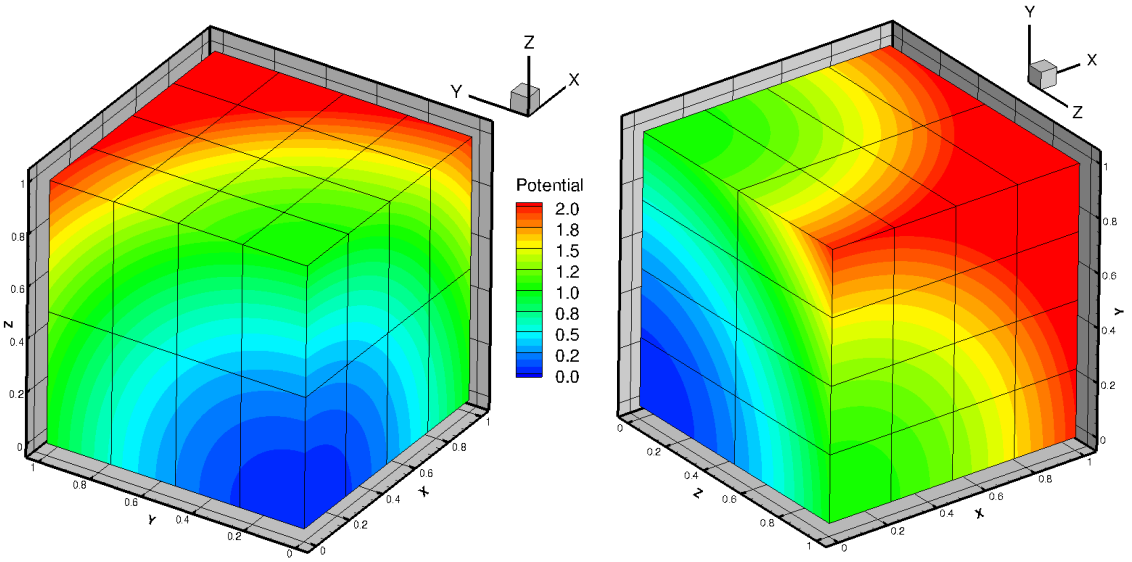


Fig. 4.1.10 3D isometric view of the solution with 32 MPI processes (4 × 4 × 2)

A test case with eq. 4.5 was also performed with a 3D Kershaw grid. This was a single block skewed grid as shown in Fig. 4.1.11 (a); all the 3 directions have 51 control volumes each, giving an odd number distribution of cells. This grid was simulated with 18 MPI processes. The MPI provided domain decomposition is shown in Fig. 4.1.11 (b), where the domain is divided in 4 × 3 × 2 sub-domains, respectively in x, y and z directions. The solution for the Poisson equation was kept in transparent mode in this figure, to better visualize the complex domain decomposition and the respective sub-domain shapes after the MPI partitioning.

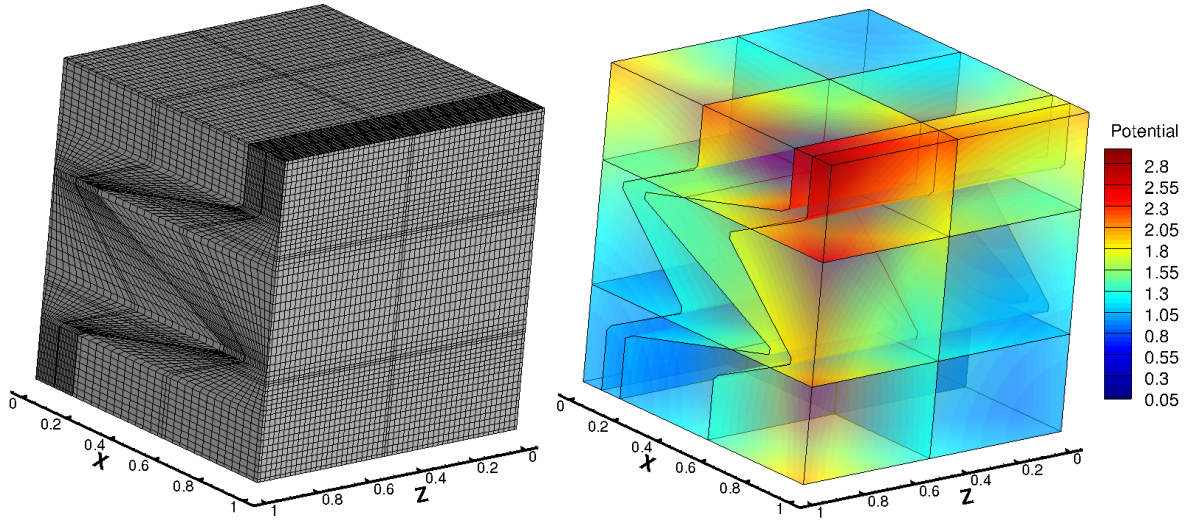


Fig. 4.1.11 3D skewed grid with decomposition in x, y, z direction 24 Procs (4 × 3 × 2)

For the quantitative analysis of solution in this stiff case, we took a section along the highly skewed grids cells. Fig. 4.1.12 shows the red line along which the computed solution is compared with the analytical one; this red line lies in the plane Z = 0.8. Fig. 4.1.12 (b) shows the comparison of two solutions. For almost half length of the curve, the two solutions match very well, and the mismatch in rest of the curve is also not too drastic. In fact, as noted in previous sections that the solution improves with the grid refinement and here we have very coarse grid for this case. It is also reported in the study by *Traore et al. (2009)* that the IDC scheme is theoretically a second order accurate scheme, and thus refining the grid will reduce the simulation errors. They have reported solutions for different refined skewed grids where these trends were observed [1-2,11]. Thus, in such skewed grids with coarsely distributed grids cells it was worth noting that a desired solution was obtained without any divergence issues, even with multiple MPI processes.

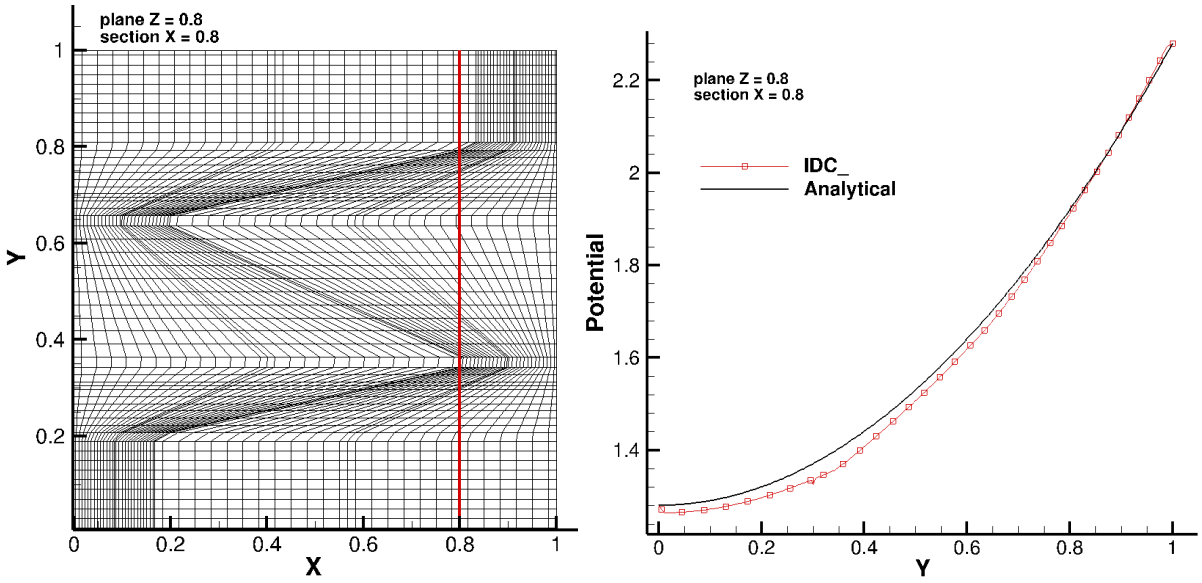


Fig. 4.1.12 Location of section X=0.8 and the results with IDC and analytical solution.

One more simulation was performed to test the MPI Poisson solver with higher number of MPI processes. For this test, we chose the four symmetrical blocks grid (grid III, Tab.4.1). Three

simulations were performed: 1) 80 processes, 2) 200 processes with MPI provided DIMS, 3) 200 processes with manually setting DIMS values for three directions. The results were compared on a section X=0.15 in plane Y=0.85, with analytical solution. As observed previously, the solution was an exact match for all the three cases with the analytical one, Fig. 4.1.13. Thus, it is assured here that the solution is completely independent of the number of MPI processes used. It proves that the message passing strategy as implemented in the code works perfectly.

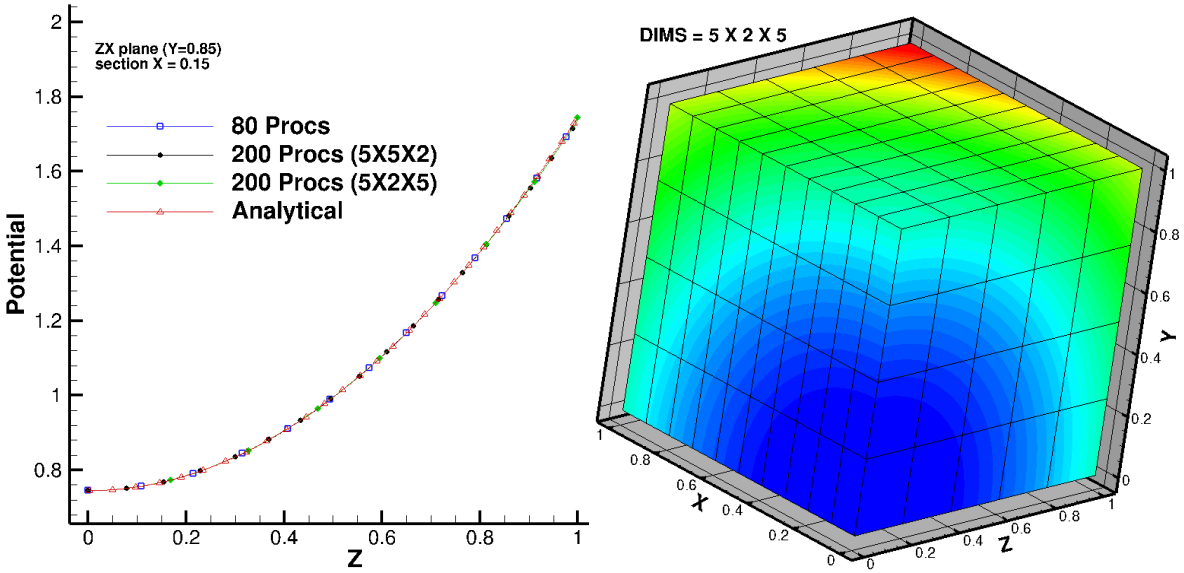


Fig. 4.1.13 a) Solution with higher number of MPI processes*, b) DIMS set by user in Z direction*

In this grid of four symmetrical blocks, each block is assigned equal number of processes as the grid sizes are same. Moreover, we have more than one option to assign number of processes in X, Y and Z directions, according to the possible factors of 50. Initially, for 2nd case we left the choice to MPI library to distribute the processes in three directions. We had 50 processes for each block and MPI assigned (5,5,2) processes in respectively X, Y and Z direction. It was noted that the grid size of each block was: 100 X 100 X 200, which makes it denser in Z direction. But, MPI factorizes the number of processes with a balanced and decreasing value in Z direction approach.

MPI cannot recognize that the actual grid size is denser in a particular direction. Thus, it is advisable that user keep attention for such situations where the grid may be denser in Z direction and needs more processes than x and y directions. So, in this study with 200 processes we manually set the DIMS value for Z as 5 and the remaining factors (5 and 2) could be given to either direction as the number of cells are same in X and Y. Fig. 4.1.13 (b) shows the partitioned grid with DIMS value (5,2,5). However, whatever the distribution of the processes the solution will not change. But, in some situations number of send and receive operations may be reduced by properly assigning the DIMS values considering the number of grid cells in each direction.

4.1.3 Validation of Periodic Boundary

Periodic boundary conditions for all three directions (X, Y and Z) were implemented in the Poisson solver. This feature is essential for various EHD problems, where most of the variables of the problem are assigned periodic boundary conditions for certain pairs of boundary faces. A general explanation and specific implementation details for periodic boundary conditions are

provided in chapter 2. Here, a spatially periodic function was designed to test the periodic implementation of Poisson solver, which is given by

$$\Delta\varphi = -12\pi^2 \left[cos\left(2\pi\left(x-\frac{1}{2}\right)\right) * cos\left(2\pi\left(y-\frac{1}{2}\right)\right) * cos\left(2\pi\left(z-\frac{1}{2}\right)\right)\right] \qquad (4.7)$$

This function is periodic in X, Y and Z directions. Periodic implementation on the three pairs of boundaries: west-east (x direction), south-north (y direction) and top-bottom (z direction) respectively, will imply that the value of the variable φ on each node of a periodic boundary face will be exactly the same on the corresponding node of the paired periodic face of that boundary face. The solution of eq. (4.7) with periodic boundaries on a cubic domain will be as given in Fig. 4.1.14.

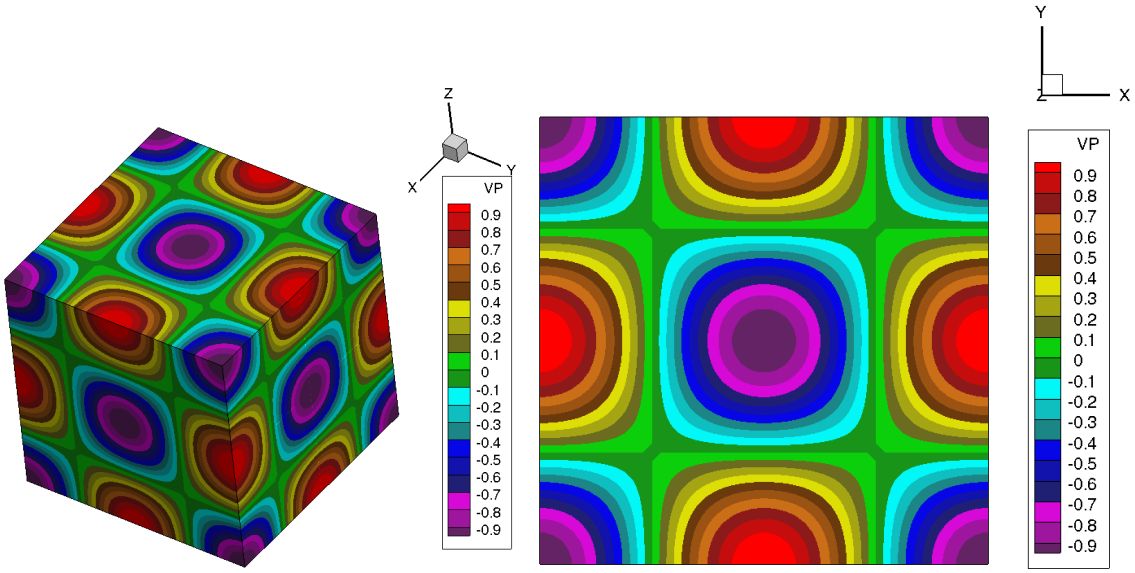


Fig. 4.1.14 a) 3D iso-metric, and b) 2D planar (XY plane) view of potential contours with periodic BC

4.2 Validation of Navier-Stokes solver

This section is devoted towards the validation of parallel Navier-Stokes solver. The MPI features of topology and inter-communicators were implemented in the pure Navier-Stokes solver of Oracle3D. It is an incompressible flow solver, in which the pressure-velocity coupling is managed with the standard SIMPLE algorithm of Patankar et al. [11]. Central differencing scheme is used for spatial discretization for all the validation cases. Gear scheme manages the discretization in time. The general equations for continuity and momentum conservation, for incompressible flows, as solved in this solver are:

$$\nabla.(\vec{u}) = 0. \qquad (4.8)$$

$$\rho\left(\frac{\partial\vec{u}}{\partial t} + \nabla.(\vec{u}\,\vec{u})\right) = -\nabla p + \nabla.(\mu\,(\nabla\vec{u} + (\nabla\vec{u})^T) + \vec{f} \qquad (4.9)$$

where ρ is the fluid density, μ is the dynamic viscosity of the fluid, $\vec{u}$ is the velocity vector and p is the pressure. External forces are considered within the source term $\vec{f}$.

4.2.1 Lid driven Cavity

The standard benchmark problem of Lid driven cavity was considered to validate this incompressible solver. Two different values (100 and 400) of flow Reynolds number were taken for comparison with benchmark solution of Ghia et al. [16]. The problem is defined as shown in Fig. 4.2.1. The top wall of square (1 X 1) cavity is fixed as a moving wall with a constant U velocity of value 1, and the other three walls are provided with no slip boundary conditions (U=0.0, V=0.0). It should be noted that non-dimensional computations are performed. Two dimensional versions of same four grid cases (Fig. 4.1.6) were considered, so we had only 1 control volume in Z direction.

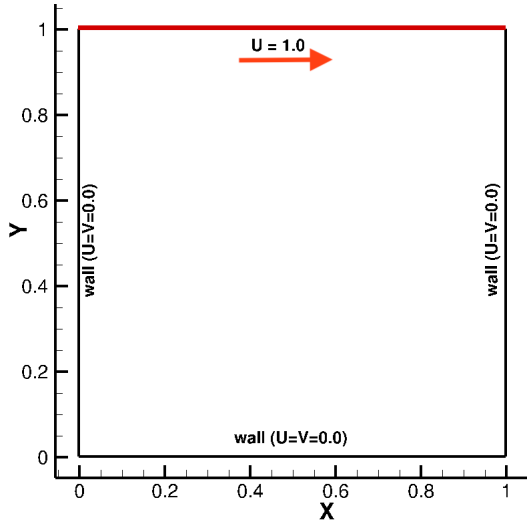


Fig. 4.2.1 The problem definition of Lid driven cavity test case

First, the results with all the cases with different block and different number of processes were compared with each other to check whether all the cases give same results or not. Fig. 4.2.2 illustrates this comparison and shows that the results are an exact match within all the cases. The sections taken for both the cases pass from the cavity center. Section for U velocity comparison is X=0.5, and section for V velocity comparison is Y=0.5. Thus, proving that the parallelization of Navier-Stokes solver is also carried out well and the solution is independent of number of blocks and number of processes. The obtained two dimensional contours of velocity components are provided in Fig. 4.2.3, it is visible from this figure that the data exchange at the sub-domain interfaces is very smooth and the MPI communications are correctly performed. The black lines in Fig. 4.2.3. represent the sub-domain interfaces, and each sub-domain is managed by a different MPI process.

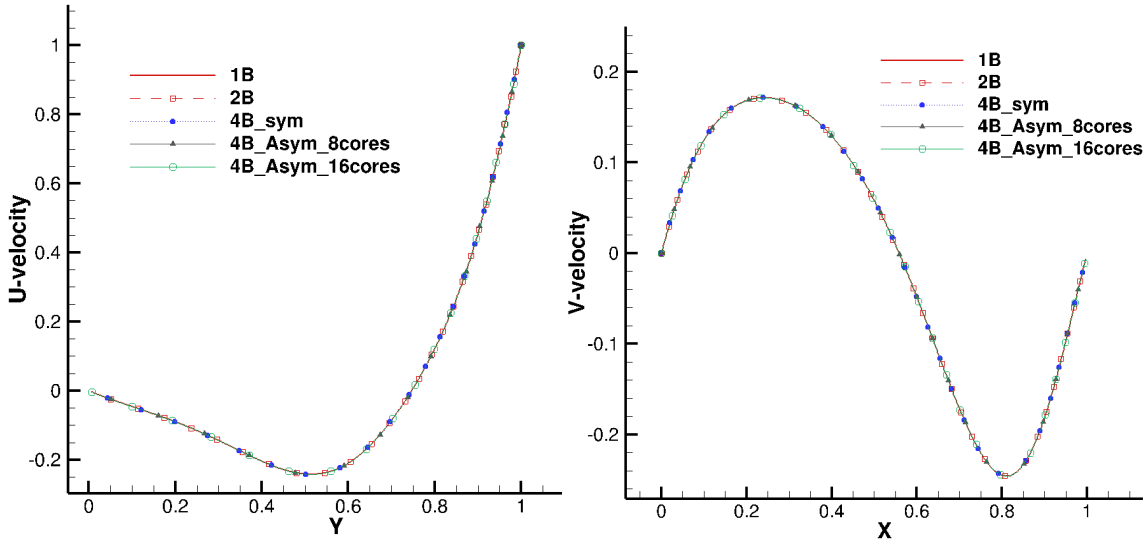


Fig. 4.2.2 U and V velocity components for different blocks and processes

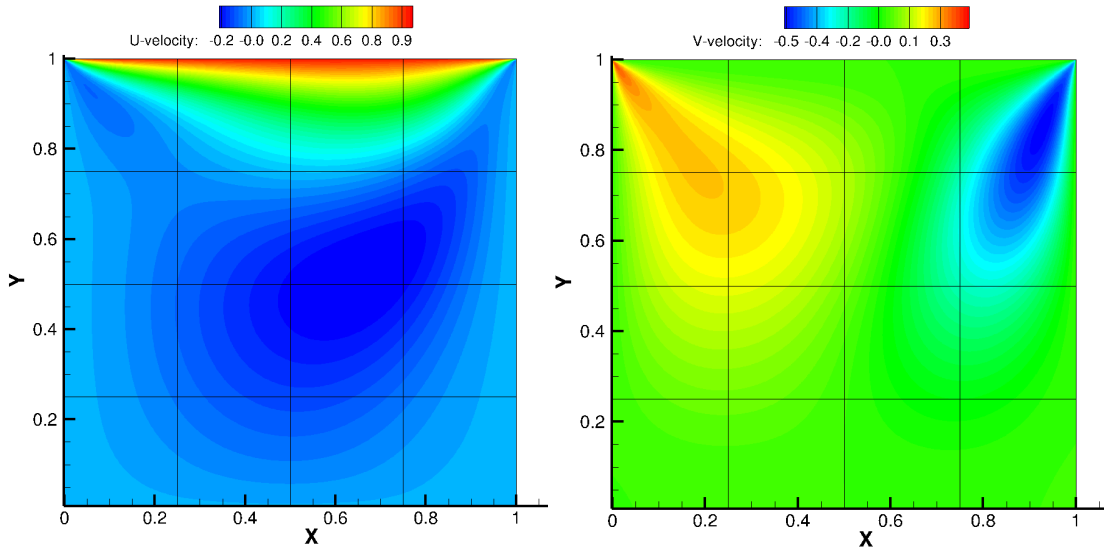


Fig. 4.2.3 Two dimensional contours of U and V velocities for 16 processes *case.*

As the results with various cases matched with each other, the validation results are only shown with the 16 processes asymmetrical block case in Fig.4.2.4. The values for velocity components were extracted from the sections as mentioned above. The comparison illustrates a perfect match with the benchmark results of Ghia et al. for the case of Reynolds number 100. The benchmark results are available in Ghia et al. (1982) [16]. A streamline figure was also produced with the 16 processes asymmetrical block case. A big vortex at the cavity center is observed together with two smaller vortices at the cavity bottom corners. The vortex at the bottom right corner is slightly bigger than the left corner vortex as reported by other researchers. This Fig. 4.2.5 should be compared with the Fig. 3 of reference Ghia et al. (1982) [16].

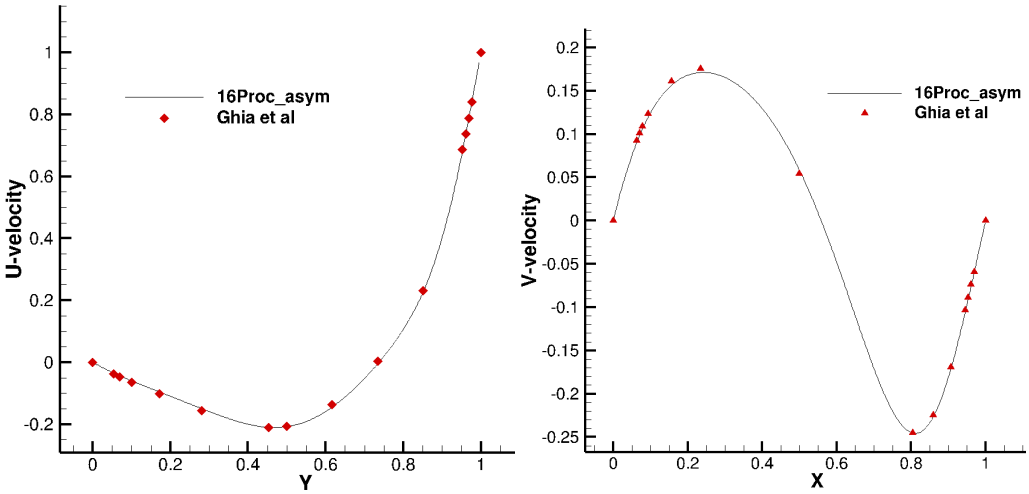


Fig. 4.2.4 Comparison of 16 processes *case with benchmark results by Ghia et al (R=100).*

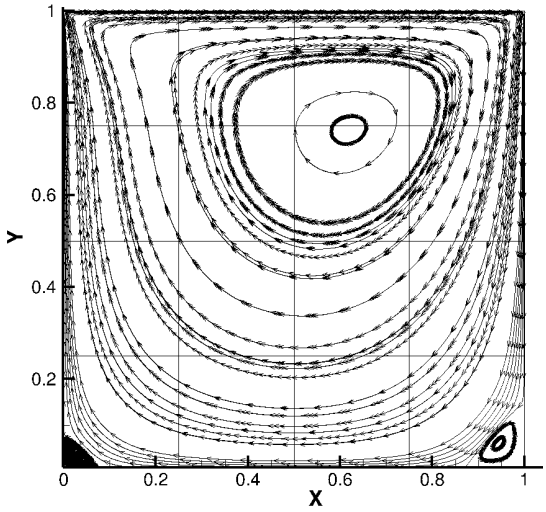


Fig. 4.2.5 Flow streamlines with R=100 in 16 processes *MPI case.*

The second validation case was carried out with R=400. This case was performed with higher values of MPI processes. The three cases consist of 400, 200 and 16 processes each. The results of these simulation were again compared against the benchmark provided by Ghia et al. Same sections passing through the cavity center were taken for velocity data. As reported for the R=100 case, the results with R=400 also give an exact match with the benchmark. Fig. 4.2.6 illustrates the comparison of U and V velocity values for R=400 case with Ghia's results.

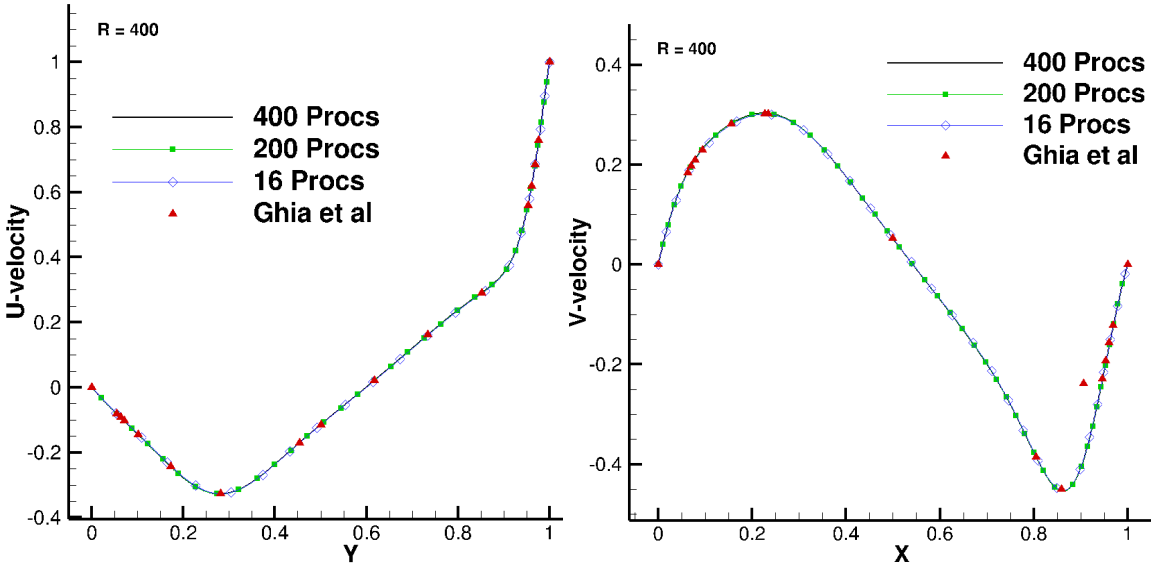


Fig. 4.2.6 Comparisons results for R=400 cases: U-velocity (left), V-velocity (right)

Streamlines plot was also compared with benchmark plot at R=400. The center of the bigger vortex is slightly shifted towards the bottom left corner, in comparison with R=100 case. Streamlines plot shown in Fig. 4.2.7 is taken with 400 MPI processes case, the light red lines behind the streamlines represent the sub-domain interfaces. This figure should be compared with the streamlines fig. 3 in Ghia et al (1982). We worked with 400 and 200 processes in this case so a very fine mesh (1000 X 1000) was taken to have enough computational load for the processes and to obtain grid independent solution. The sub-domains are partitioned with equal sizes, we see smaller sub-domains near the top wall than the bottom wall, because the grid was much refined at the top wall.

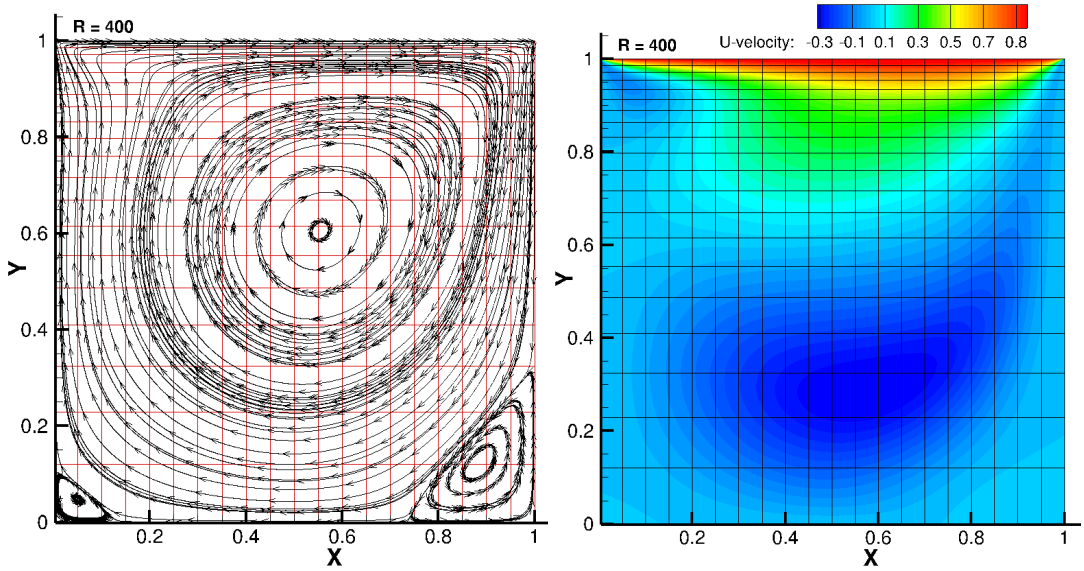


Fig. 4.2.7 Streamlines and 2D U velocity contour for R=400 case with 400 MPI processes.

4.2.2 Backward facing step case

Backward facing step (BFS) flow is another standard and well-studied problem for validation of Navier-Stokes solvers. Several benchmarks, both experimental and numerical, are available with different configurations of the channel, the step height and flow Reynolds number. We aim to validate our MPI implementation, especially the inlet and outflow boundary conditions, with a steady incompressible BFS flow at different Reynolds numbers, in this section.

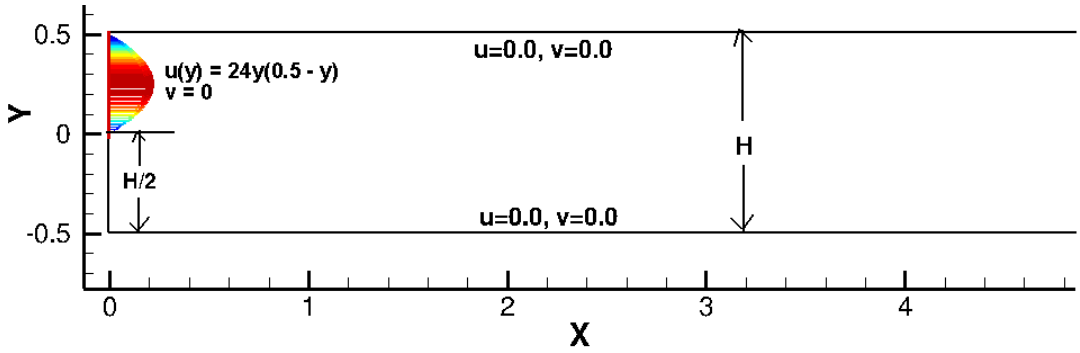


Fig. 4.2.8 The BFS configuration used for our simulations

The problem configuration is illustrated in Fig. 4.2.8. We chose to use the configuration used by Gartling (1990), as benchmark quantitative results are available with this setting which will be compared with our solution. The velocity inlet is located exactly above the channel step and a fully developed parabolic velocity profile, $u(y) = 24y(0.5 - y)$ for $0 \leq y \leq 0.5$, is used as the inlet flow. This inlet velocity profile was used by Gartling (1990), and it gives a maximum inlet velocity of $u_{max} = 1.5$ and an inlet average velocity of u_{avg} =1.0. This helps us remove the inlet channel portion which is used if the inlet velocity is not fully developed (e.g. u =1.0) for a channel flow. The step height and channel inlet height are 0.5, making the overall channel width in the downstream portion to 1.0. We use the non-dimensional computations to facilitate comparison with other similar settings [13,15]. All the walls are set to no slip boundaries for the velocities as shown, and an outflow boundary condition is set at the channel outlet such that the upstream flow, mainly the recirculating regions, is not affected by the outflow boundary conditions.

The overall channel length is 30 for this case where we compare with the quantitative results of Gartling (1990). This is equal to 60 times the channel height and is sufficiently long to have a developed channel flow, and, provide accurate solutions in the sensitive near step regions of the domain. Flow was simulated with Reynolds number 800, until a converged steady-state solution is obtained. Two well defined stable recirculating regions are observed as visible in Fig. 4.2.9. The recirculation region just behind the step extends until x ~ 6.0, 12 times the step height, as reported in previous studies [12-15]. The upper wall vortex starts forming at x ~ 4.8 and extends until x ~ 10.6. The approximate recirculation lengths for both of these regions are reported in various studies [12-15], as mentioned in Tab. 4.2.1.

Important point to mention as reported by Armaly et al. (1983) is that above Reynolds 400, three dimensional effects become significant and the comparison with 2D studies fail to provide accurate match. These approximate results are reported in Gartling (1990), and it was also noted that in other studies no tabular results were given so it was difficult to get the accurate quantities. Graphical results were optically scanned to get these results. However, our results match quite satisfactorily with the other 2D studies, Tab. 4.2.1.

Tab. 4.2.1 Approximate lengths of upper and lower wall recirculation regions.

Study	$Length_{lower-wall}$	$Length_{upper-wall}$
Gartling (1990)	6.1	5.6
Armaly et al.	7.2	4.1
Kim et al.	6.0	5.75
Sohn et al.	5.8	4.7
Oracle3D	5.91	5.8

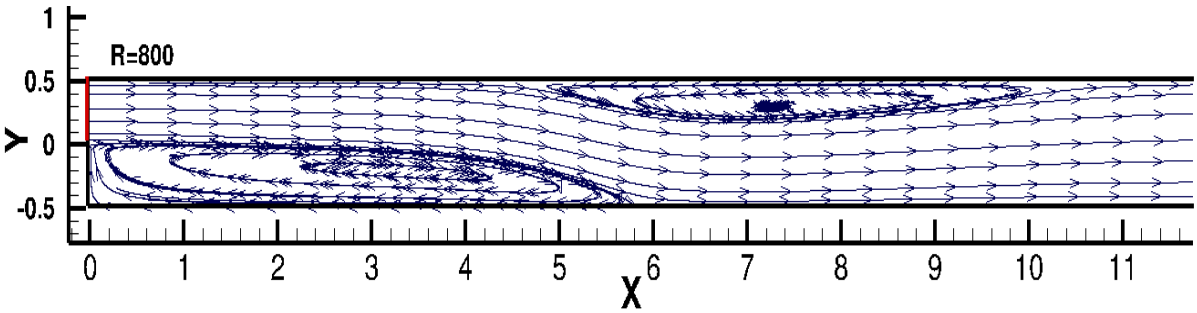


Fig. 4.2.9 Flow streamlines obtained with Gartling's configuration at R=800, with Oracle3D.

For quantitative comparison, U velocity results for two downstream sections at X=7.0 and X=15 were extracted to compare with benchmark results of Gartling (1990). Fig.4.2.10 shows that results with our parallel Navier-Stokes solver match excellently with the benchmark results provided in [12]. It is evident that the downstream flow at x=15.0 has developed a parabolic velocity profile and far away from the step the flow behaves as a normal channel flow. The solution was found to converge well and there was no impact of the outflow boundary on the flow upstream. Fig. 4.2.11 shows velocity vectors on different x section along the Y direction. In these figures, complete domain length for X = 30 is not shown to better visualize the results.

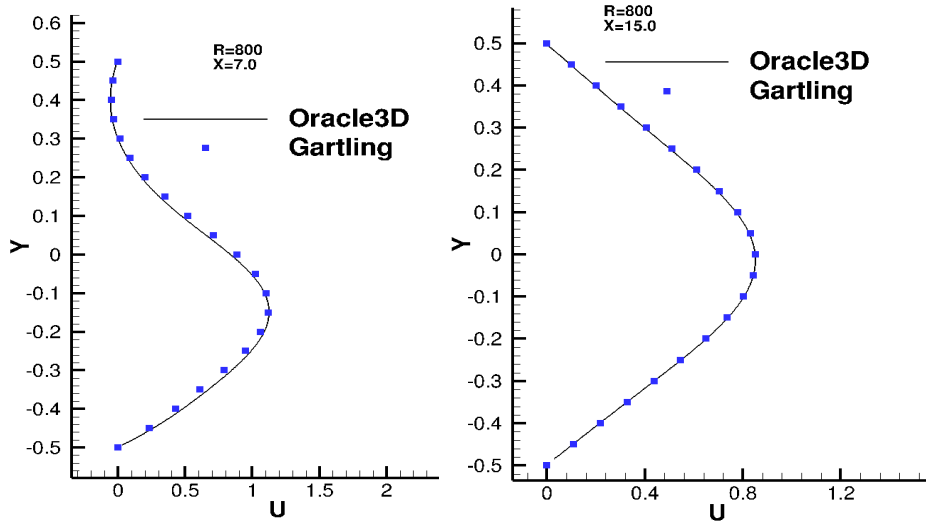


Fig. 4.2.10 U velocity comparison plots: a) X=7.0, b) X=15.0

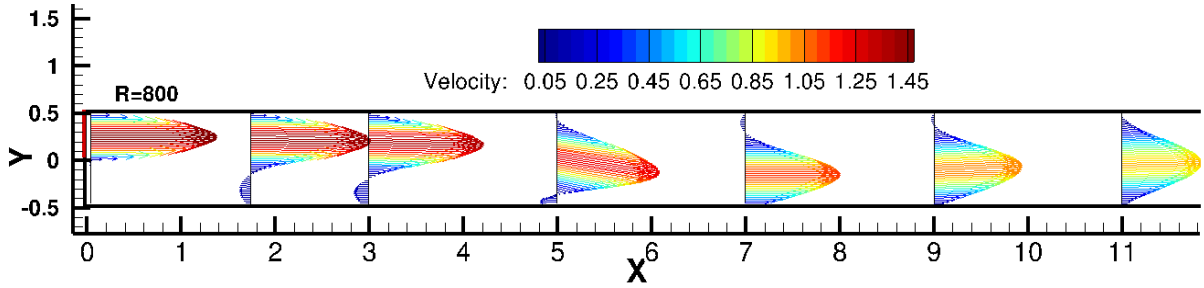


Fig. 4.2.11 Velocity vectors on different sections, R=800

It is also important to note here that this simulation was performed with 80 MPI processes. The grid size was 4000 X 200 control volumes, which was refined at the bottom and top walls. Fig. 4.2.12 shows the velocity contour along with thin black lines which represent the MPI sub-domain interfaces. Thus, each smaller domain as defined with these black lines in Fig.4.2.12 was managed by a different MPI process.

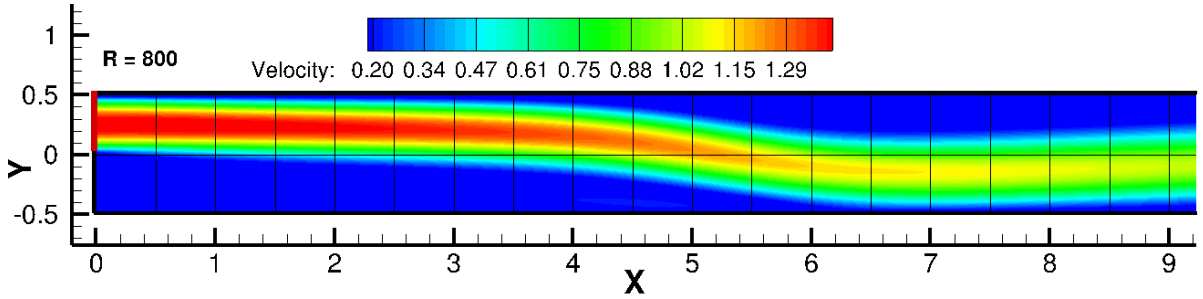


Fig. 4.2.12 Velocity contours within individual MPI sub-domains.

Same configuration of BFS (Fig. 4.2.8) was also analyzed for other Reynolds numbers: 100, 200, 389. In these cases, however, the domain length was reduced to x = 10.0 only, as there are no phenomena of interest occurring further downstream. These problems were solved with 180 MPI processes, and the grid size was 4000 X 900 control volumes. The decomposition of domain in MPI sub-domain is presented in Fig. 4.2.13. The distribution of processes is 40 X 9, in X and Y respectively. The grid is finer near the step and gets coarser and coarser after x = 5.0, thus the distribution of processes is finer near the step, as each process has equal size of computational cells to work on. Figure 4.2.14 illustrate the full domain with velocity contours provided within each MPI sub-domain, for Reynolds 389.

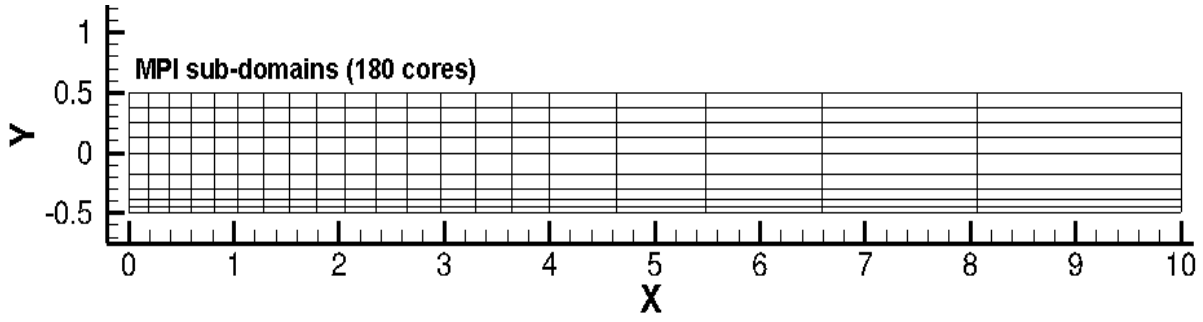


Fig. 4.2.13 BFS configuration with domain decomposition in 180 processes *case.*

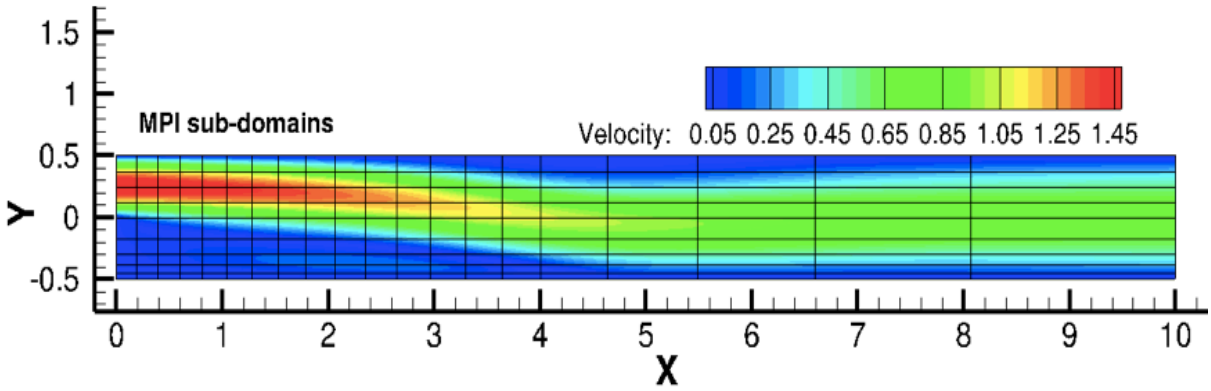
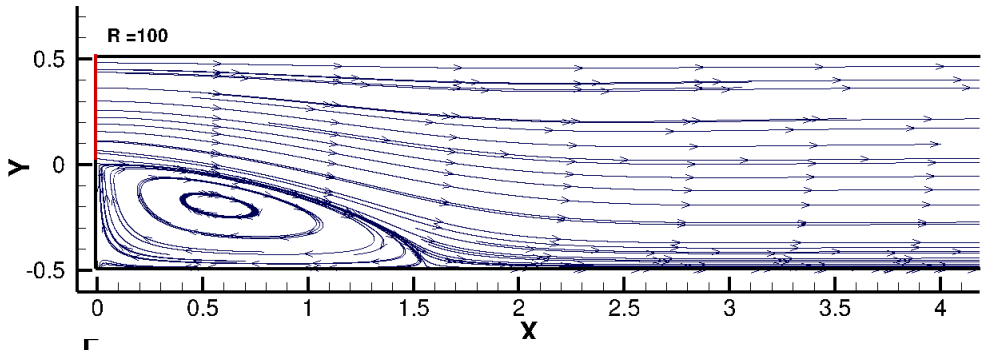


Fig. 4.2.14 Velocity contour in individual sub-domain for R=389

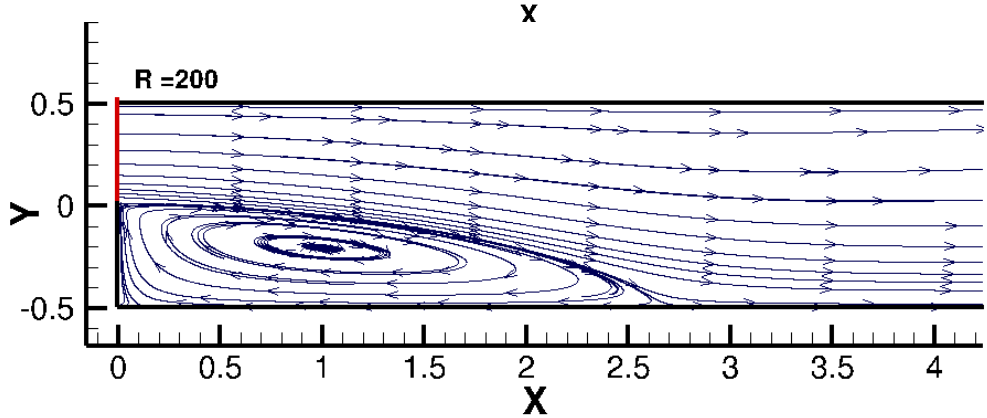
The reattachment lengths of the separated flow from the step corner is well reported, and it is often used to compare the results with different methods. Here, we consider the reattachment lengths with above mentioned three Reynolds numbers. Fig. 4.2.15 illustrates the flow streamlines for steady state flow with Reynolds numbers – 100, 200 and 389. This is laminar regime and the size of the recirculation zone increases with increasing Reynolds number, as evident from Fig. 4.2.15. The location of reattachment as obtained from the numerical results of Armaly et al. (1983), and corresponding values obtained with our simulations are provided in tab. 4.2.2. Non-dimensional values of reattachment length are used here which is the division of reattachment length with the step height: Xr/h. Here, step height (h) is 0.5.

*Table 4.2.2. Comparison of numerically obtained values of reattachment lengths (*Xr/h*)*

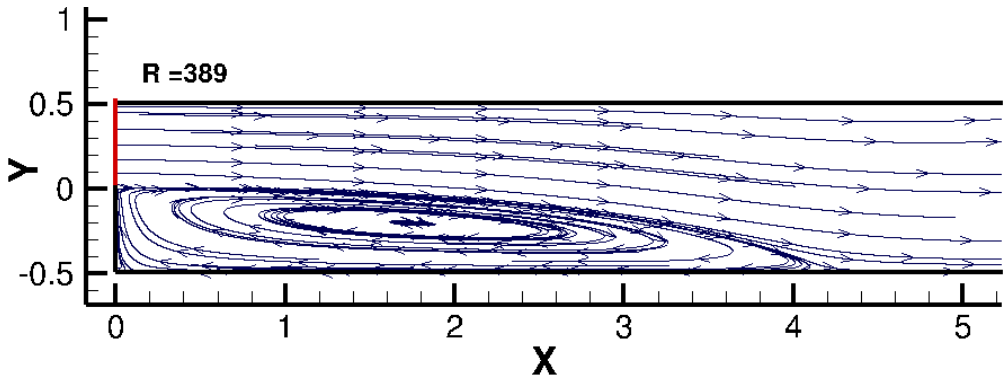
	Reattachment length		
	R=100	R=200	R=389
Armaly et al. (1983)	3.18	5.0	7.9
Oracle3D	3.1	5.2	8.2



(a)



(b)



(c)

Fig. 4.2.15 Flow streamlines: a) R=100, b) R=200, c) R=389

4.2.3 Validation of Periodic Boundary

Periodic boundaries are essential part of any Navier-Stokes solver. We provide here the solution figures of two test cases which were performed to validate the implementation of periodic boundary condition's implementation in Oracle3D. Periodic boundary conditions were newly implemented in the MPI version of the code, so it was felt important to provide the test cases for the benefit of prospective users of Oracle3D. Classical Poiseuille and Couette flows were used for the validation of the periodic boundary implementation in our code. Details of these flows can be obtained from any standard Fluid Mechanics books [17, 18].

Fig. 4.2.16 shows Poiseuille flow with Periodic boundary conditions, which depicts that the flow comes out at the east face in a parabolic profile, and the same flow enters at the west face of the channel. In Couette flow, we provide a velocity U=1 at the top face of channel as the boundary condition. Due to the difference in velocities of adjacent layers of fluid a pressure gradient is developed in the channel which makes the bottom portion of the fluid moving in negative x direction. And, as we have periodic boundaries on east and west faces, we see that the flow which comes out the west face goes in at the east face. These two simple test prove that the periodic boundaries are well implemented in the parallel solver.

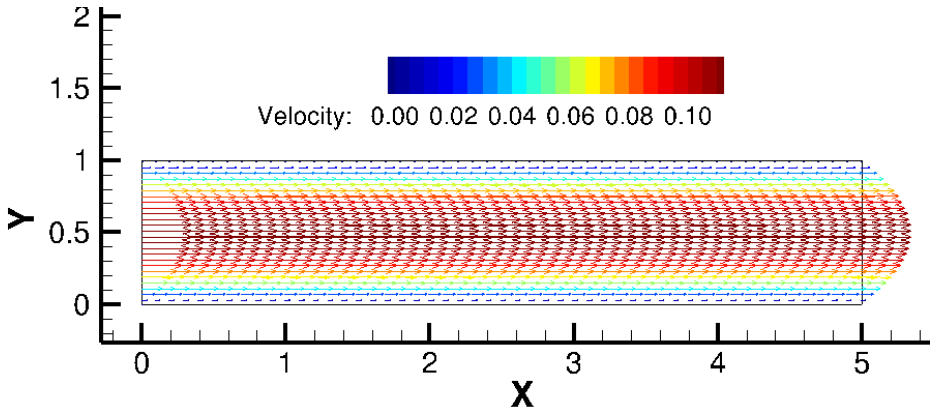


Fig. 4.2.16 Poiseuille Flow with periodic boundaries along X direction

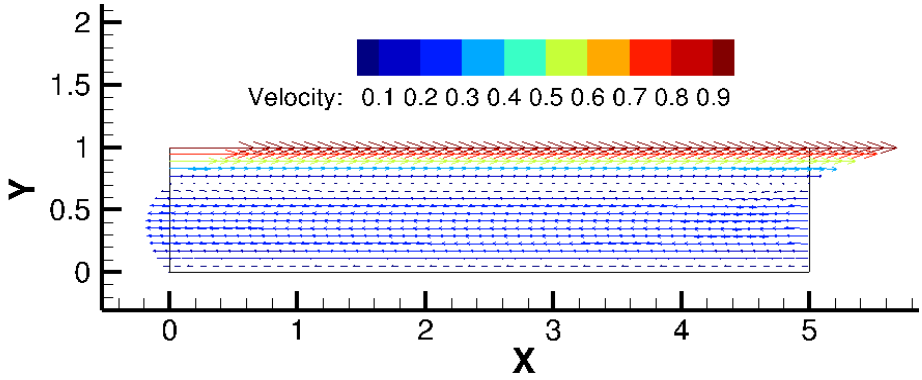


Fig. 4.2.17 Couette Flow with periodic boundaries along X direction, with an adverse pressure gradient

4.3 Parallelization of Transport Solver

Within the Oracle3D framework, we developed a general transport solver also. This solver mainly solves a convective transport equation for a scalar variable (φ), like electric charge density. The diffusion term of the transport phenomena is not included here to exclusively study the efficiency and implementation of the convective schemes. Total Variation Diminishing (TVD) schemes are implemented to resolve the spatial discretization with higher than 2nd order accuracy. Special attention was paid for the parallel implementation of TVD scheme at the interface nodes and the boundaries. Ghost cell method is adopted to store the neighbor data at interfaces, because in TVD schemes data from more than one neighbor nodes in each direction is required depending on the flow direction. Also, the periodic boundary condition feature was implemented and tested in this transport solver. The whole implementation is detailed in Chapter II. The general transport equation as used in this solver is:

$$\frac{\partial \varphi}{\partial t} + \nabla.(\varphi \vec{u}) = 0\,. \tag{4.10}$$

4.3.1 Test of Periodic transport

Test simulations were performed to validate the parallel implementation of TVD scheme and periodic boundary condition for the transport solver. Both, 2D and 3D runs were undertaken with multiple number of MPI processes. In the 2D case, a square domain with periodic boundaries on both pairs of boundaries (X and Z direction) was taken. A smaller square area in the domain is provided with value $\varphi = 1$ which extends from 0.2 to 0.5 in both directions, Fig. 4.3.1 (a). The constant velocity components in the two directions are given as U = 1.0 and W = 1.0, which will direct the overall velocity along the diagonal of the square domain. Under the influence of this diagonal velocity the scalar quantity φ, will get transported (advected) in the diagonal direction. As we have set the periodic conditions on both pairs of the boundaries, we will see the smaller square corresponding to $\varphi = 1$, will go out of the domain from upper right corner and it will re-enter the domain from the bottom-left corner. This is depicted in Fig. 4.3.1 (b).

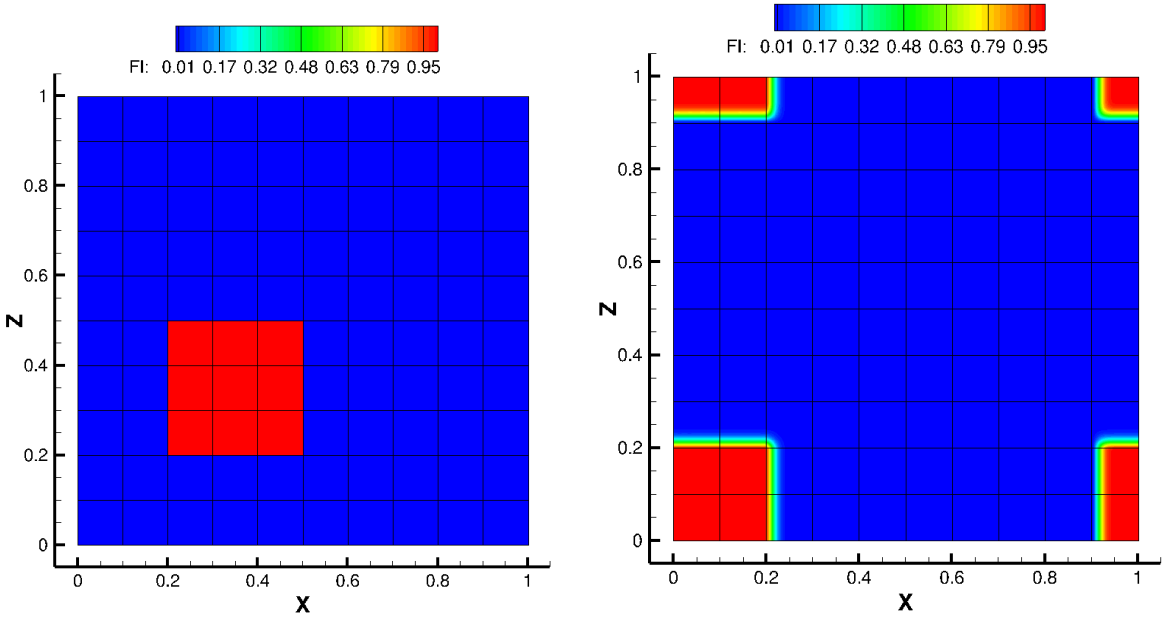


Fig. 4.3.1 a) smaller square with $\varphi = 1$, b) Periodic transport of φ, along X and Z boundaries

This test was carried out with a fine grid (500 X 500) to have enough load for all the MPI processes used. In total, 100 MPI processes were used for this problem, 10 processes in each direction. The thin black lines in the domain represents the boundaries for the MPI sub-domains, which are handled by different processes. An animation was created to better visualize the transport mechanism through periodic boundaries. We observe that there are some portions of the smaller square visible at the upper-left and bottom-right corners also, which is perfectly valid, as the vertical side of upper-right corner is periodically paired with the vertical side of upper-left corner. And similarly, the vertical side of bottom-left corner is periodically paired with the vertical side of bottom-right corner. Same arguments are given for the horizontal sides of these corners also, which is the reason for the appearance of square portions at the other two corners.

Same case is repeated in 3D domain, where we have set all three pairs of the boundaries as periodic. Grid for this case was 200 X 200 X 200, and 32 MPI processes were used. As shown in Fig. 4.3.2, the domain in X was partitioned with 4 processes, in Y with 2 processes and in Z with 4 processes, making the total count of processes to 32 for this case. The velocity vector as given in this case was $\vec{u} = (1, 1, 1)$, which again provides the diagonal overall velocity. The location of cube before and after the periodic transport is shown in Fig. 4.3.2. Thus, the cube goes out of the domain at the corner location (1, 1, 1) and re-enters the domain from bottom corner at (0, 0, 0). These two tests show the capability of the TVD scheme and the periodic boundary implementation in the parallel code, which are the most important features for the EHD problems.

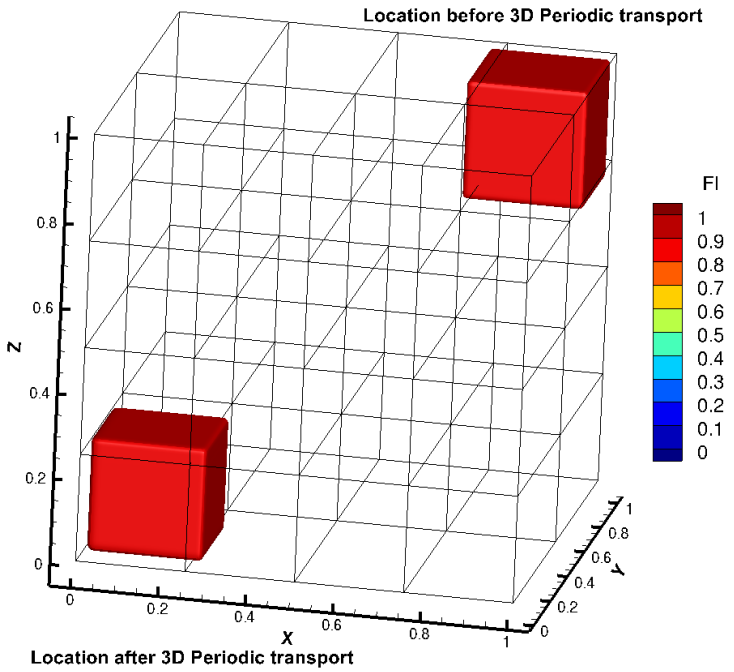


Fig. 4.3.2 3D periodic transport of a cube with $\varphi = 1$, location after and before the periodic transport

4.3.2 Test of rotational transport

Rotational deformation allows to test the efficacy of convective schemes by presenting greater number of discontinuities during the motion. And, also for the test efficacy of parallel implementation this case presents higher degree of complexity in data communication at MPI domain interfaces. In this test, a square-shaped scalar field extends from 0.4 to 0.6 in both x and y directions, as shown in Fig. 4.3.3. The square scalar field is subjected to rotational velocity field given by:

$$u = -u_0 * \cos(\pi(x - x_0) * sin(\pi(y - y_0) \quad (4.11)$$

$$v = v_0 * \sin(\pi(x - x_0) * cos(\pi(y - y_0) \quad (4.12)$$

Here u_0 and v_0 are the magnitude of velocity components which are set to 1. The coordinate (x_0, y_0) corresponds to the point (0.5, 0.5), which is the center of the square and also the center of the domain. This velocity field introduces a clock-wise motion in the square scalar field. The level of numerical diffusion of the TVD scheme depends highly on the grid refinement. We carried out this test with three different grid sizes: 1) 100 X 100, 2) 200 X 200 and 3) 1000 X 1000. Naturally the coarse grids are expected to produce higher degree of false diffusion than the refined grid, which was observed during this test, Fig. 4.3.4. It is also observed that at the corners of the square scalar field the edges have been curved a little in case of the refined grid. This test was simulated with 16 processes, however the square scalar field was limited to only central 4 processes, as seen in Fig. 4.3.3. It is evident with the solution that the implementation of TVD scheme at the interface of the MPI domains is well done, as the rotational motion was observed to be smooth and continuous at the interfaces.

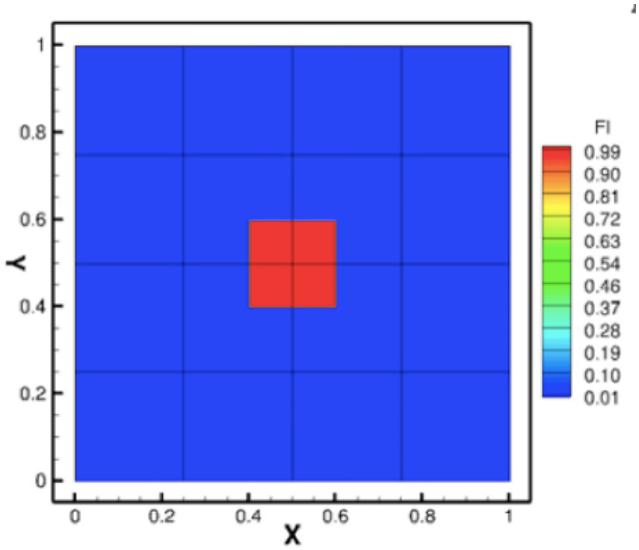
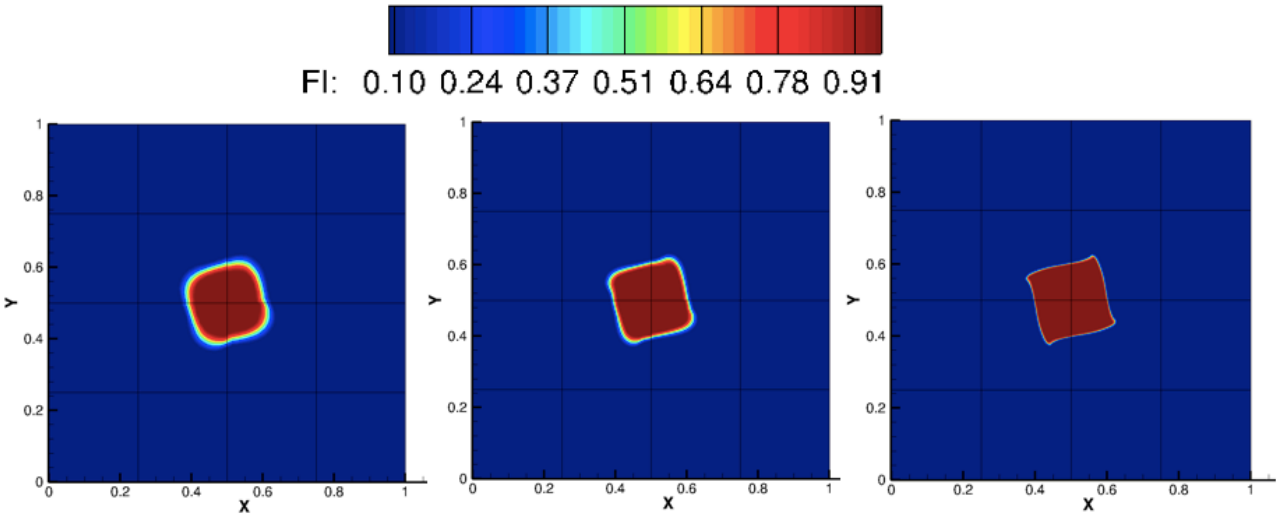


Fig. 4.3.3 Location of square scalar field before rotational deformation test.



Grids: a) 100 X 100 b) 200 X 200 c) 1000 X 1000

Fig. 4.3.4 Rotational deformation comparison with 3 grids

Bibliography

[1] Y. M. Ahipo, P. Traore, "A robust iterative scheme for finite volume discretization of diffusive flux on highly skewed meshes," J. Computational and Appl. Mathematics, 231, 478-491(2009)

[2] P. Traore, Y. M. Ahipo, C. Louste, "A robust and efficient finite volume scheme for the discretization of diffusive flux on extremely skewed meshes in complex geometries," J. Computational Physics, 228, 5148-5159 (2009)

[3] P. Traore, A. T. Perez, "Two-dimensional numerical analysis of electroconvection in a dielectric liquid subjected to strong unipolar injection," Phys. Fluids 24, 037102 (2012).

[4] J. Wu, P. Traore, P. A. Vazquez, A. T. Perez, "Onset of convection in a finite two-dimensional container due to unipolar injection of ions," Phys. Rev. E 88, 053018 (2013).

[5] P. Traroe, J. Wu, "On the limitation of imposed velocity field strategy for Coulomb-driven electroconvection flow simulations," J. Fluid Mech. 727, R3 (2013).

[6] W. Gropp, E. Lusk, A. Skjellum, "Using MPI: Portable Parallel Programming with the Message-Passing Interface," 2nd edition, The MIT Press, 1999

[7] www.idris.fr/formations/mpi

[8] https://comptuting.llnl.gov/tutorials/mpi

[9] https://www.mpi-forum.org

[10] J. Wu, "Contribution to numerical simulation of electrohydrodynamic flows: application to electro-convection and electro-thermo-convection between two parallel plates." PhD. Thesis, University of Poitiers (2012)

[11] S. V. Patankar, "Numerical Heat Transfer and Fluid Flow,", 1980, Taylor & Francis Publisheres

[12] D. K. Gartling, "A test problem for outflow boundary conditions- flow over a Backward-Facing step," Int. J. for Numerical Methods in Fluids, 11, 953-967 (1990)

[13] B. F. Armaly, F. Durst, J. C. F. Pereira, B. Schonung, "Experimental and theoretical investigation of backward-facing step flow," J. Fluid Mech., 127, 473-496 (1983)

[14] H. Le, P. Moin, J. Kim, "Direct Numerical simulation of turbulent flow over a backward-facing step," J. Fluid Mech., 330, 349-374 (1997)

[15] J. L. Sohn, "Evaluation of FIDAP on some classical laminar and turbulent benchmarks," Int. J. for Numerical Methods in Fluids, 8, 1469-1490 (1988)

[16] U. Ghia, K. N. Ghia, C. T. Shin, " High-Re solutions for incompressible flow using the Navier-Stokes equations and a Multigrid method," J. Computational Physics, 48, 387-411 (1982)

[17] P. Kundu, I. M. Cohen, "Fluid Mechanics," 4th edition, Elsevier (2008)

[18] R. K. Bansal, "A textbook of Fluid Mechanics," 1st edition, Laxmi Publications Ltd. (2008)

Chapter 5
EHD Unipolar Injection

Oracle3D is mainly an Electrohydrodynamic (EHD) solver which has been used to perform several studies including Unipolar Injection, EHD conduction, electro-separation, electro-thermo convection, flow control studies using plasma actuators, etc. [1-11]. All the necessary and important details regarding the code Oracle3D have been described in previous chapters, including the validation and performance tests. This chapter presents the important Electro-convection (EC) related numerical work carried out with Oracle3D during the course of this PhD. Oracle3D consists of a Poisson solver, a Navier-Stokes solver, and a scalar transport solver for several species. A combination of these three physical models makes the complete Oracle3D code.

In terms of transport solvers, three separate modules are available now which individually solve the Unipolar Injection (one charge species), Electro-conduction pumping (two charge species), and Plasma discharge (three charge species) models. The studies with Oracle3D, until now, mainly dealt with two dimensional domains, limiting its use for three dimensional aspects associated with the EHD problems. Now, as the parallel version is prepared, we revisit some classical EHD problems focusing mainly on their 3D aspects. An outline for this chapter is as follows:

The first part (5.1) of this chapter covers the EHD problem of unipolar injection in dielectric liquids between parallel plates. Some initial tests with some 2D cases are provided to validate the implementation in code, and then 3D electro-convection is discussed in some details. Second part (5.2) deals with EHD plumes induced by ion injection in blade-plane configuration. Simulated results are compared with some similar available studies.

5.1 Unipolar Injection between parallel plate electrodes

The phenomenon of unipolar injection in dielectric liquids is well reported both experimentally and numerically. Several two-dimensional numerical studies are available to compare the qualitative and quantitative results with this model [1,2,9,27,28]. A good amount of publications on this problem were carried out with the previous version of Oracle3D (Fortran 77 baseline version) [1-11]. This presents us with an opportunity to validate the unipolar injection module of parallel Oracle3D code against the scalar code results. Some 2D simulations with established results are provided after explaining electroconvection in unipolar injection with the mathematical model. And, a detailed 3D analysis of the problem will follow afterwards.

5.1.1 Introduction

Electroconvection between planar parallel electrodes has been investigated widely in last few decades [12-23]. A dielectric liquid, confined between two parallel plate metallic electrodes, feels significant impact of the electric field produced by the two electrodes, when supplied with an electric potential difference. High electric field between these electrodes leads to complex

electrochemical reactions at the electrode surfaces. In such situations, charge particle injection may occur at the interface of liquid and electrode surface, on one or both electrodes [2,25]. When the ion injection occurs at only one of the electrode surfaces then it is commonly termed as unipolar injection [1].

The injected ions gain momentum due to the Coulomb force, and they start migrating according to the electric field direction. A strong enough electric field develops an instability which also sets the liquid into motion, affecting the overall convection of the injected ions [6]. Such motion of liquid, with low enough conductivity, can be compared with a liquid motion due to the difference of temperature between liquid layers (thermo-convection) [13]. The two ways of convection in liquids: thermo-convection and electro-convection are often compared according to their similarities of induced flow patterns (rolls, hexagons etc.), and, dissimilarities in the underlying mechanisms [12, 37].

Unipolar injection phenomena, which is fundamental to EHD, makes a definitive analogy with the classical Rayleigh-Benard convection (RBC) [12, 25]. The liquid motion is hindered by the viscosity (for weak forces), and the state of liquid is potentially unstable in both phenomena. Atten et al. (1979) explained differences in transport mechanisms in both types of flows. In thermo-convection, heat is transferred by liquid convection and diffusion; however, in electroconvection the diffusion of charge particles is mostly negligible, and charges mainly move under the influence of liquid convection and electric field with a significant mobility. From numerical point of view, electroconvection is a system of non-linear coupled set of equations, and, in RBC the energy equations (heat convection) are of linear nature for the temperature with the Oberbeck-Boussinesq approximation [12,38]. Studies on specific pattern formations, both numerical and experimental, in these two phenomena have been reported since the conception of RBC in 1900s. Cellular flow patterns like parallel rolls, rectangles, hexagons etc. are numerically investigated in RBC by several authors [40-47]. Getling et al. (2003) presented an evolution study on three-dimensional patterns in RBC and noted that the flow seeks optimal scale to stabilize. Rapaport (2006) performed a molecular dynamics simulation to attain the hexagonal convective patterns in RBC.

In unipolar injection, several two-dimensional studies have been published by several groups using different numerical methods to solve the set of EC equations [7,21,23,27]. Perez et al. (1989) discussed the role of diffusion and Coulomb repulsion with Flux corrected transport (FCT) algorithms in finite amplitude EC. They highlighted that unsteady charge distribution is mainly dependent on advection terms, and diffusion needs to be included only if steady-state solutions are simulated [20]. Castellanos (1990) investigated injection induced instabilities and highlighted chaotic flow in unipolar injection at high electric fields. Vazquez et al. (2008) did a stability analysis and obtained the two roll structure with Finite element – FCT and Particle-in-cell (PIC) methods. Traore et al. (2013) investigated the evolution of EC flow from one convection cell structure to two convection cell structure; and then finally the chaotic regime above T = 1500. Wu et al. (2013) addressed the stability issue in wall bounded cavities with different aspect ratios, with Finite Volume method (FVM). All of these studies were carried out in two dimensional cavities.

Naturally, a large amount of research work in EC is accompanied by a lot of industrial applications also. Heat transfer enhancement [48,49], electrostatic precipitations [51,53], flow control application [52], EHD drag pumps [50], atomization technology and EHD turbulent mixer [53], are just a few to mention here. Coulomb force by the external electric field is the main driving force for the injected charge particles in gases. The high mobilities of ions in gases

make them move much faster than the neutral gas molecules. However, in liquids, the ions having rather low mobilities are also convected by the bulk liquid motion except their drift velocities due to the Coulomb force. Thus, the charge particles in liquids experience both the drift and the strong coupling with the liquid motion. Both the electrical and the hydrodynamic effects are strongly felt by the injected species in liquid.

5.1.2 Mathematical Formulation

The motion of liquid greatly modifies the charge distribution within the bulk and, consequently, the electric field due to the charge particles is also modified. This electro-hydrodynamic coupling, along with the mathematical instability, presents great difficulties for those seeking to solve this EC problem numerically in complex settings. Atten and Lacroix (1979) studied the unipolar injection problem theoretically and experimentally, and successfully predicted the characteristics of this EC flow with Galerkin-type method [12,2]. Their model was later verified with several numerical studies [9,19,23]. The complete set of non-dimensional, coupled EHD equations governing unipolar injection is as follows:

$$\nabla.(\vec{u}) = 0 \tag{5.1}$$

$$\frac{\partial \vec{u}}{\partial t} + \nabla.(\vec{u}\,\vec{u}) = -\nabla p + \frac{1}{R_{el}}\nabla.(\nabla\vec{u} + (\nabla\vec{u})^T + C\,M^2 q\vec{E} \tag{5.2}$$

$$\frac{\partial q}{\partial t} + \nabla.\left(\mathrm{q}\left(\vec{u} + \vec{E}\right)\right) = 0\,. \tag{5.3}$$

$$\nabla^2 V = -C\,q \tag{5.4}$$

$$\vec{E} = -\nabla V \tag{5.5}$$

We assume the liquid to be incompressible, Newtonian and completely insulating, satisfying eq. (5.1). Here, $\vec{u} = [u, v, w]$ is the velocity vector for the bulk fluid. Generalized pressure in the bulk liquid is expressed with p, which contains both the hydrostatic pressure and the electrostriction part. The transport of charge due to diffusion is neglected in our unipolar model from the eq. (5.3). The external electric field $\vec{E}$, is induced by the potential difference, $\Delta V = V_0 - V_1$, between the two electrodes. The charge density, q, is acted upon by the Coulomb force, $q\vec{E}$, which acts as the only source for fluid motion in eq. (5.2). The fluid is considered homogeneous and isothermal, leaving the effects of temperature on system to be negligible. The charge is always assumed to be injected by the lower electrode in our cases, and the other electrode works as a collector of charges. We find four non-dimensional numbers to be explained with above equations: T, R_{el}, M and C.

$$T = \frac{\varepsilon\,\Delta V}{\rho_0 \nu\, K}\ ,\ C = \frac{q_0 L^2}{\varepsilon\,\Delta V}\ ,\ M = \frac{1}{K}\left(\frac{\varepsilon}{\rho_0}\right)^{1/2}\ ,\ R_{el} = \frac{T}{M^2} \tag{5.6}$$

Here, T is the electric Rayleigh number which is described as the ratio of Coulomb force to viscous forces. C is termed as the injection strength which is a dimensionless measure of the injection level. M is the ratio between the hydrodynamic mobility and actual ionic mobility,

and R_{el} is the electric Reynolds number. The fluid properties are denoted with ρ_0 (fluid density), ν (kinematic viscosity) and ε (electric permittivity).

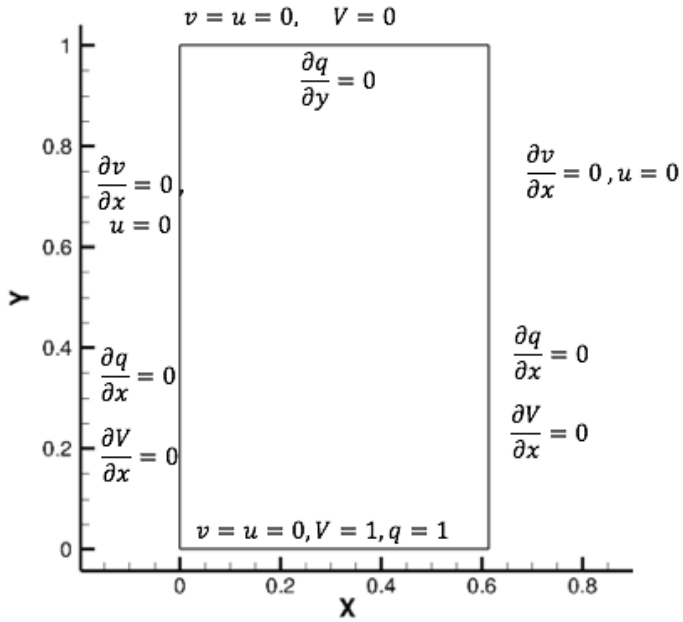


Fig. 5.1.1 Computational field and the boundary conditions

5.1.3 Unipolar injection 2D cases

The 2D problem configuration is provided in Fig. 5.1.1. The bottom wall is used as an injecting electrode and the top wall behaves as a collecting electrode. Dirichlet boundary conditions for electric potential and charge density are set for these two walls. The side walls are given with Neumann boundary conditions as shown. The electric potential and charge is initialized with an analytical solution, to quickly achieve the desired solution.

A) T= 200 case

Several values of T parameter are considered to validate the 2D solution against available data in literature. The solution with T=200, C=10 and M=10 is well established. In this configuration and parameter setting, a single convection cell is observed with T=200. We took a mesh of 100 X 200 control volumes in X and Y direction respectively. The domain width of 0.614 corresponds to a wave length of one convection cell (cell size) in this setting.

Firstly, it is important for us that our MPI implementation gives same results with any number of MPI processes used. We considered 4 cases: 1)1 block, 2) 2 symmetrical blocks, 3) 4 symmetrical blocks, and 4) 4 asymmetrical blocks. And, also simulated these cases with 8 processes and 16 processes. Charge density contour results are provided in Fig. 5.1.2 for these four cases, in which one convection cell is observed for all the four cases, as it was reported with T=200 in [6,9].

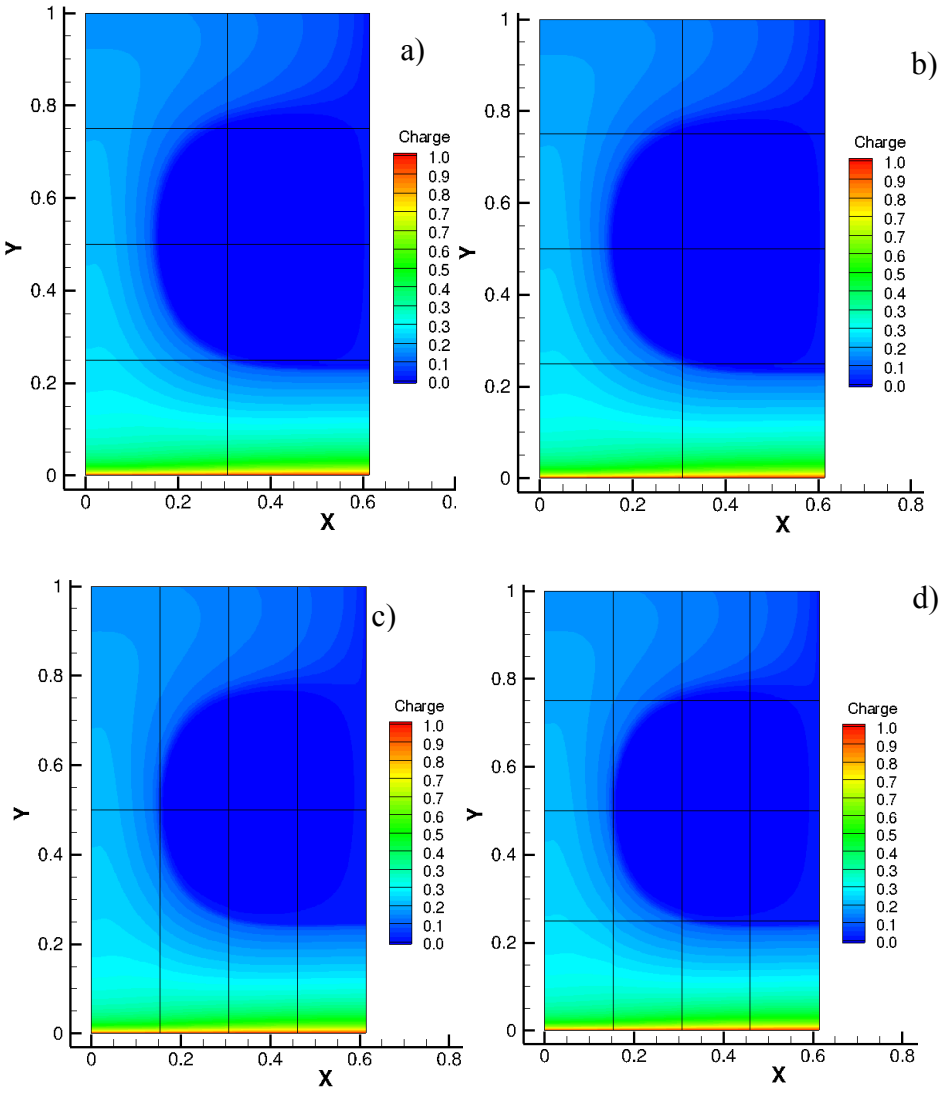


Fig. 5.1.2 Charge density contour showing one convection cell with T = 200, for 4 cases of different block configurations. a) 1 block (8 processes), b) 2 blocks (8 processes), c) 4 symmetrical blocks (8 processes), d) 4 asymmetrical blocks (16 processes)

For quantitative comparison, different variables were plotted on a section at X = 0.4. Four variables, charge density, electric potential, U velocity and V velocity components were obtained as shown in Fig. 5.1.3 and Fig. 5.1.4. All the plots show a very good match among all the four cases, which justifies the parallel implementation, mainly the message passing at the interfaces of different processes. Electric potential contours with the 4th case (4 asymmetrical blocks) are depicted in Fig. 5.1.5.

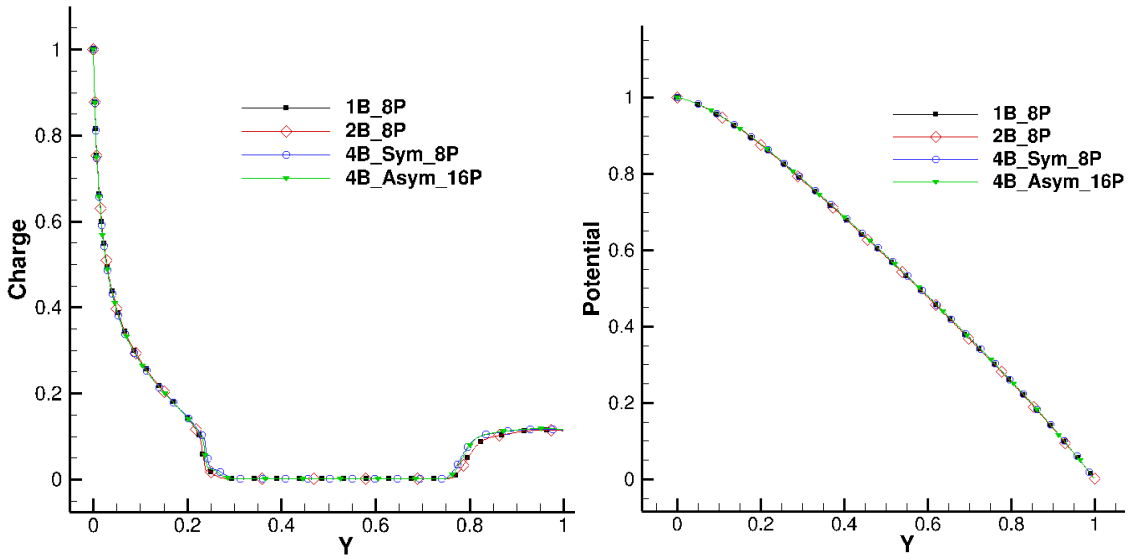


Fig. 5.1.3 Charge density and electric potential plots for different block configurations.

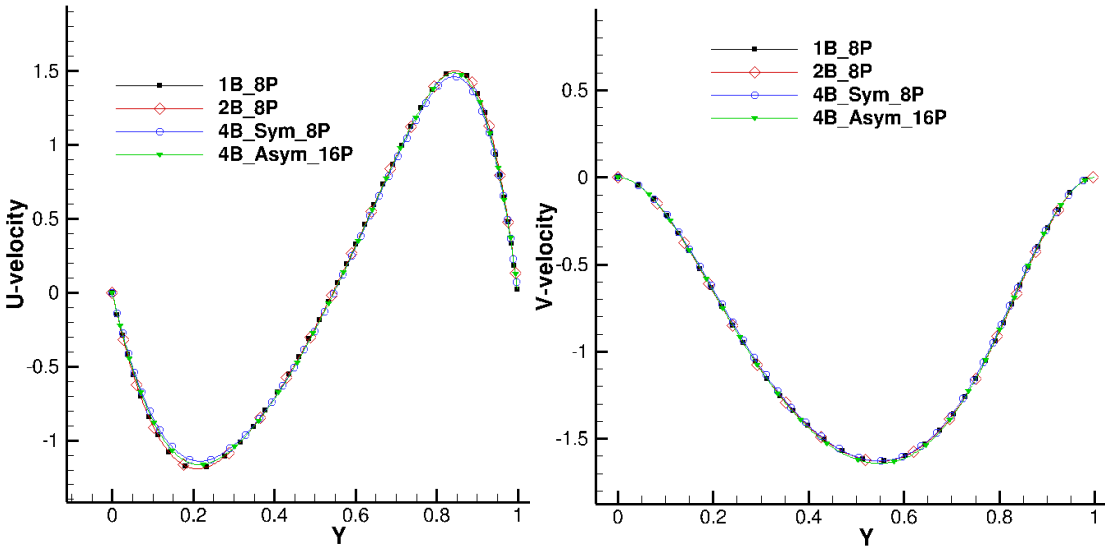


Fig. 5.1.4 X and Y components of velocity plots for different block configurations.

Results with parallel Oracle3D, were also compared with the baseline Fortran 77 code, which has been validated previously [1-7]. Maximum velocity profile obtained with 1 block case (8 processes) was compared with the profile obtained from baseline code. Fig. 5.1.6 shows that the obtained maximum velocity is same with the baseline code, however, there is a small delay with the parallel code in the onset of instability. It was observed that this problem of unipolar injection was numerically very sensitive to several factors like grid, time step, number of processes etc., for the onset of instability [9]. However, the magnitude of the maximum velocity obtained was always comparable for same conditions. Plot on the right in Fig. 5.1.6 is the charge density profile comparison on the section (X=0.4), which shows a very good match with the previously validated code results.

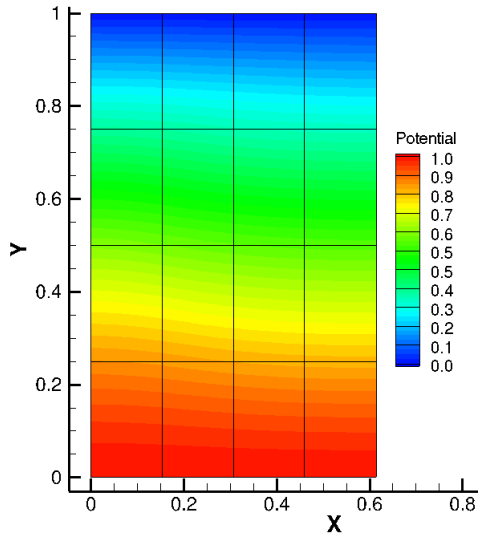


Fig. 5.1.5 Electric potential contours with 16 processes *(4 block asymmetric case)*

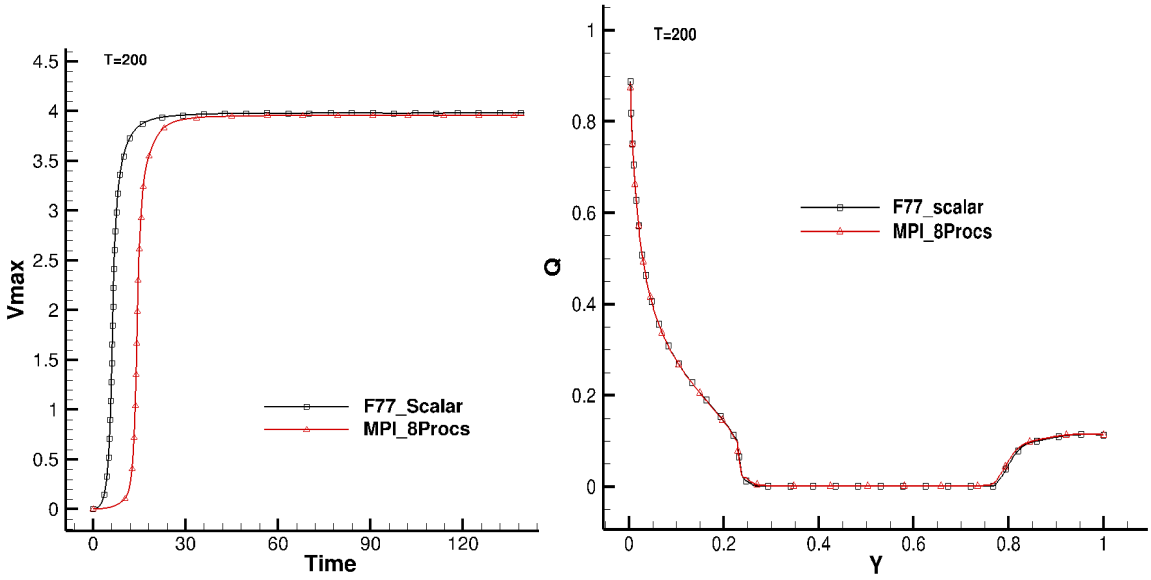


Fig. 5.1.6 Maximum velocity and charge density profiles comparison with baseline code results

B) T > 200 cases

As we increase the electric Rayleigh number (T), keeping the M parameter constant (here M=10), we increase the electric Reynolds number (R_{el}) of the induced EHD flow. The flow remains in stationary laminar regime, with one convection cell, until T=250 [1]. With the higher values of T the flow structure between the two parallel plates changes significantly. The flow reportedly starts to oscillate periodically in time and space at T=260 [9]. It was reported in [1,9] that the one convection cell EHD flow changes to a two-convection cell regime at T=300. It was observed on the maximum velocity profile that firstly, the flow sets into motion with the onset of instability following the exponential growth in maximum velocity. At the maximum velocity state, the flow seems steady for some time. However, as the flow is allowed to evolve further in time, flow starts oscillating and there appears a sudden fall in the maximum velocity which corresponds to the onset of second instability in the flow. At this point, the one cell steady regime turns into a two steady convective cells regime. Traore and Perez (2012) showed two

convective cells flow structure with T=300, which was obtained by Jian (2012) also. They obtained these results with Finite Volume method, however, Vazquez et al. (2008) had reported the two-cell regime at T=400 with particle in cell method (PIC).

Our simulation was carried out with 8 processes on same settings as Traore et al. (2012). Fig. 5.1.7 shows the profile of maximum velocity evolution with time, which clearly indicates the second instability at roughly 60 non-dimensional time units. The contour of charge density shows the two convective cell structure of the EHD flow, as reported by others. Fig. 5.1.7 should be compared with Fig. 15 in Traore et al. (2012). The magnitude of maximum velocity is also a good match with mentioned studies.

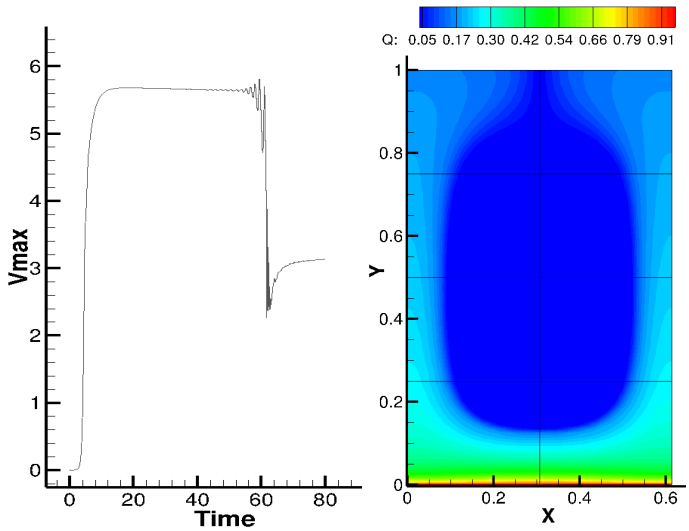


Fig. 5.1.7 Maximum velocity time evolution and charge density contour after occurrence of second instability, for T=300 with 8 processes

Simulations were also performed with T=400 with 4, 10, 16 and 20 processes. All of these tests showed the steady two-cell flow structure. Increasing further the T parameter, leads to the chaotic regime of flow, which has been well reported in [6,9]. The steady two cells are observed till T=500 and above that periodic oscillations start again. Above T = 1500 the flow is characterized by the EHD plumes [9]. The flow becomes fully unsteady and the plumes occur more frequently as we increase T further. The destabilization of laminar sub-layer near the emitter electrode gives rise to the formation of charge particle plumes [1]. Simulation results with T=4000 are depicted in Fig. 5.1.8. The Vmax plot explains the unsteady nature of flow and charge density contour indicates a rising charge plume from the emitter electrode surface. A simulation was carried out with 10 times longer domain, results of which are shown in Fig. 5.1.9. This simulation had 600 X 100 grid cells and it was simulated with 32 processes. This figure shows larger number of frequent EHD plumes arising from the emitter surface.

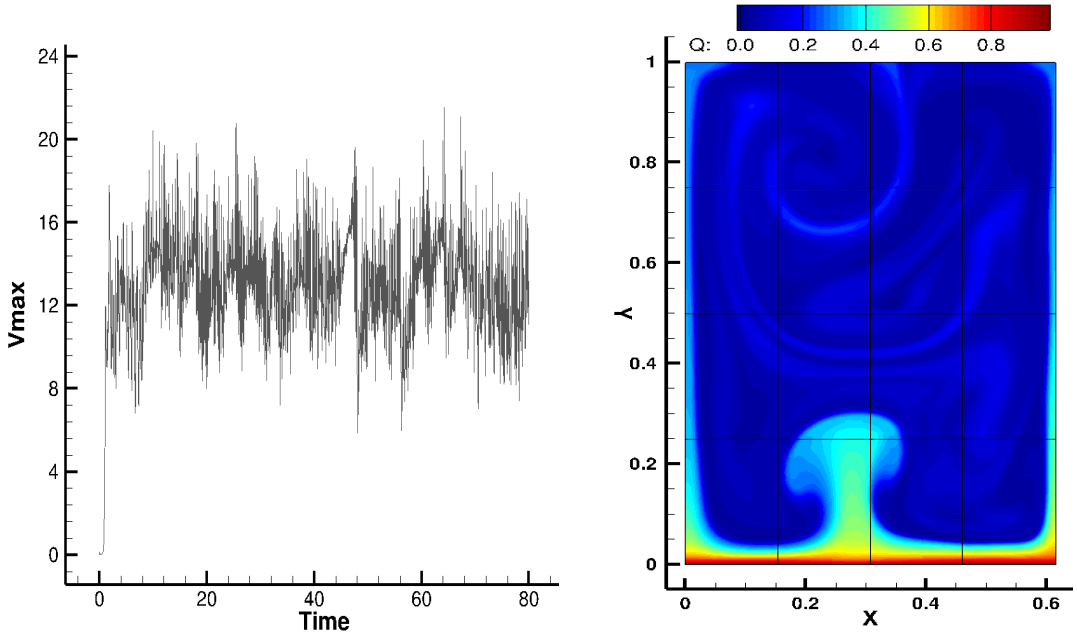


Fig. 5.1.8 Maximum velocity evolution with time and the contour of charge density showing EHD plume formation.

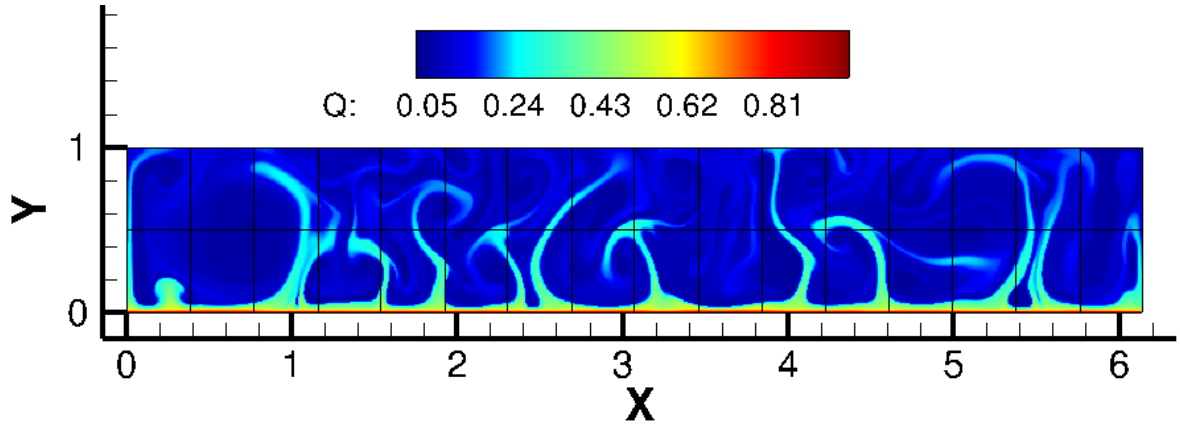


Fig. 5.1.9 Charge density plumes with longer domain at T=4000, [grid (600 X 100), 32 processes]

5.1.4 Unipolar injection 3D cases

A full 3D EC problem is computationally very demanding due to the presence of high gradients of charge density, and the complex coupling of non-linear equations of charge density, electric potential and flow variables. The complex nature of governing equations, has restricted EHD community to a very few 3D numerical studies. For such problems, a parallel code with desired scalability is indispensable. Vazquez et al. (2011) performed a stability analysis of a 3D EC between parallel plate electrodes using PIC method. They simulated weak injection regime and obtained the critical values of T parameter for onset of instability and compared the results with the values obtained from linear stability analysis. Kourmatzis et al. (2012) discussed mainly the turbulent regime of EC flow between two parallel plates.

Demekhin et al. (2014) used Finite Difference method to simulate 3D coherent structures with direct numerical simulation (DNS), and described the evolution of patterns in EC. They observed three characteristic patterns: two-dimensional EC rolls, 3D regular hexagons and

other 3D structures of spatiotemporal chaos (a combination of hexagons, quadrangles and triangles). They showed that the transition from 3D regular patterns to chaos was accompanied by interacting two dimensional solitary pulses. A 3D flow pattern study is most recently reported by Luo et al. (2018), where they described the use of Lattice Boltzmann method (LBM) to obtain stable hexagonal cells in plane parallel plate electrode configuration. Their study covered the most comprehensive stability analysis of hexagonal cells with a range of non-dimensional parameters (T, M etc.) [37,38].

After validating the injection module of Oracle3D with 2D cases, we also investigated 3D flow pattern evolution in the unipolar injection case in planar parallel plate electrode setting, with our Finite Volume approach. The occurrence of hexagonal cells is a core feature of such flows, which has been observed experimentally and theoretically. In pure, isotropic liquids there is no favoured direction for the convective cells to develop in horizontal plane, and hexagonal convective cells has higher symmetry to stabilize than other polygonal patterns (squares, rectangles etc.) [12,13]. However, before Luo el al. (2018) stable hexagonal cells were not reproduced by any numerically study in EC.

We started our study with symmetry boundary conditions (zero gradients for all variables) on the four vertical boundaries (Fig. 5.1.10). No-slip condition for velocities on the high voltage electrode, and zero gradient on the grounded electrode. Electric potential was set to Dirichlet boundary conditions on high voltage (V=1) and grounded electrodes (V=0). Charge density was set to Dirichlet value (q = 1) on high voltage electrode and zero gradient on grounded electrode. The non-dimensional parameters were taken as T=170, M=10, C=10.

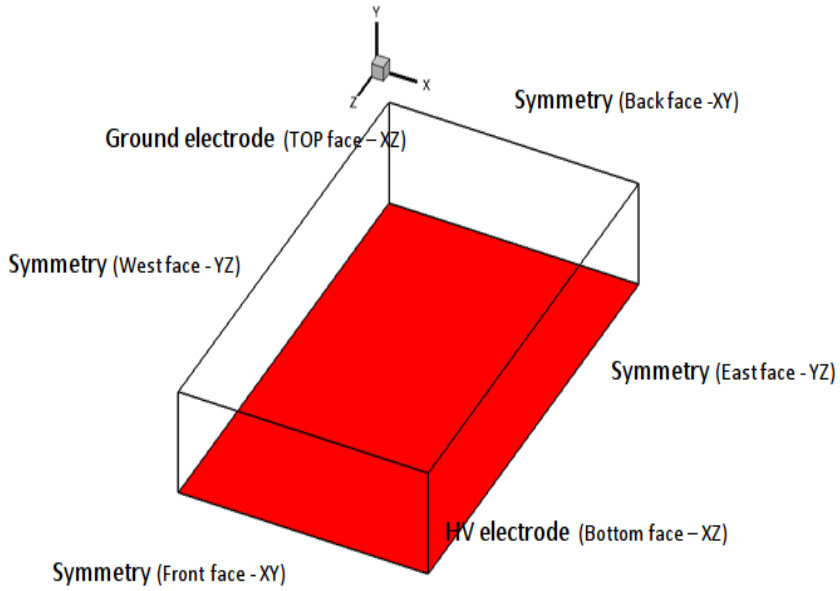


Fig. 5.1.10 Boundary conditions for 3D EC case with symmetry on four sides

In experiments, hexagons were observed in cylindrical cavity with large aspect ratio [12]; and in stability analysis an infinite domain with fully developed flow is usually assumed [34, 37]. Larger domain sizes (L=W=5, 10) were simulated first to reduce the impact of symmetry boundaries on the convection cell formation. The effects of domain size, grid, time step and initial velocity perturbations were analysed to obtain steady hexagonal cells. Table 5.1 provides

the details of three tests cases presented here with symmetry boundaries on lateral walls. In first case, no initial velocity perturbations were provided, and evolution of Vmax was plotted with time (Fig. 5.1.11). The instability occurred around 50 non-dimensional time units, and it was observed that the flow started evolving with initial hexagonal cells. All the convection cells which did not have direct contact with the lateral walls were observed to be in hexagonal shapes at the onset of instability.

Tab. 5.1 Cases details ran with symmetry BC on vertical boundaries

	Grid (L_x , L_y , L_z)	**Control Volumes total (million)**	**$V_{initial}$**	**Time step (dt)**	**T**
1.	(5,1,5)	6.25 (250 X 100 X 250)	0	10^{-3}	170
2.	(5,1,5)	6.25	Hexa	10^{-3}	170
3.	(10, 1, 10)	25	Hexa	10^{-3}	170

In Fig. 5.1.11, a top view of charge density iso-surfaces at q=0.19 is shown. The observed hexagonal cells were of random shapes at the onset of instability, as they were evolving. The cells were pushing each other in all directions to achieve an optimal scale to stabilize themselves as also reported by Getling et al. (2003) for RBC and by others in EC [37,38]. The charge density iso-surfaces at q=0.02 in Fig. 5.1.12 depict a bun like shape of convection cell as reported by Luo et al. (2018), the charge density inside this cell was observed to be zero. This explains the 3D charge transport mechanism in such EC flow as explained by Atten et al. (1979) and Luo et al. (2018), where the charge particles mainly migrate under the drift velocity effect and this drift velocity compete with the liquid velocity after the instability occurs. The domain regions where the drift velocity of charges is balanced by the liquid velocity component in electric field direction, remain void of charge particles [12], and these regions (convection cell centres) are strictly free of charge.

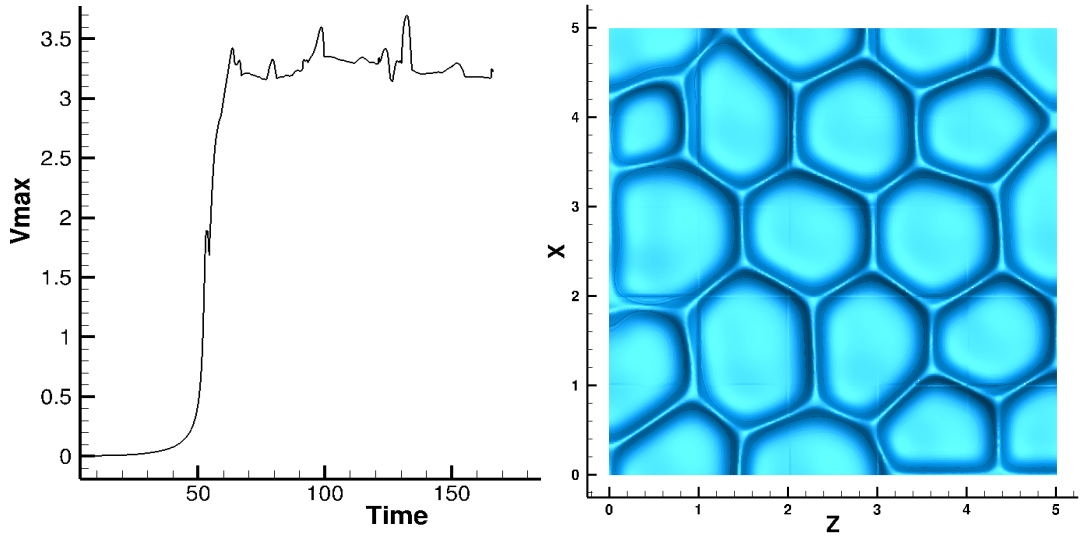


Fig. 5.1.11 Vmax evolution plot (case 1), and the top view with charge density iso-surfaces at q=0.19 at the onset of instability.

The charge void regions are a characteristic feature of these EC flows which is completely different from RBC. In these EC flows, the fluid always descends in the centre of cells, however, in thermal convection both upflow and downflow hexagonal cells have been observed to be coexisting [44,42]. In Fig. 5.1.13, we present the vertical velocity iso-surfaces at Vy=1 (in yellow) and Vy=-1 (blue), which show that the flow velocities in the peripheral regions of the cells are in upward direction and in the centre of the cells it is a downward motion. We observed that the flow did not reach a steady state in this simulation, as the hexagonal cells could not stabilize until 150-time units, and the maximum velocity evolution plot shows the unsteady nature of flow. However, it was observed that as the convection cells pattern was evolving the flow in the cell centres was always downwards.

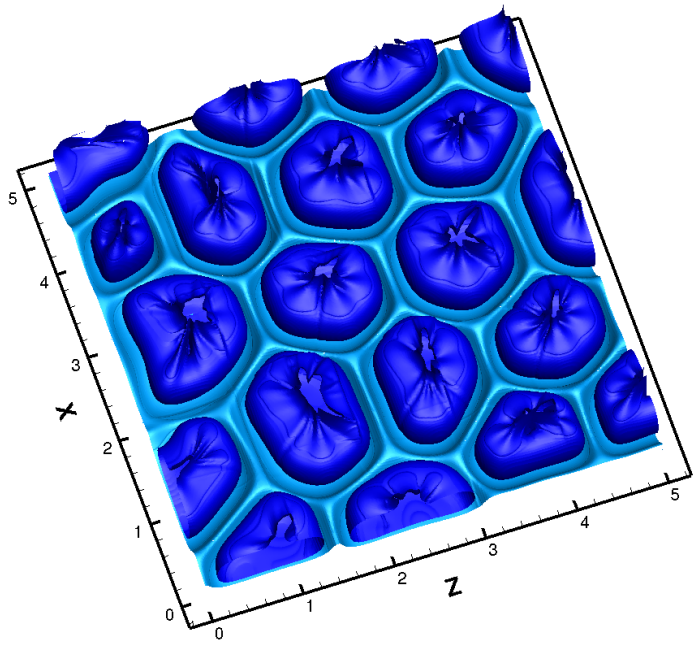


Fig. 5.1.12 Charge density iso-surfaces at q=0.19 (hexagonal cells), and q=0.02 (bun-shaped cells), at the onset of instability.

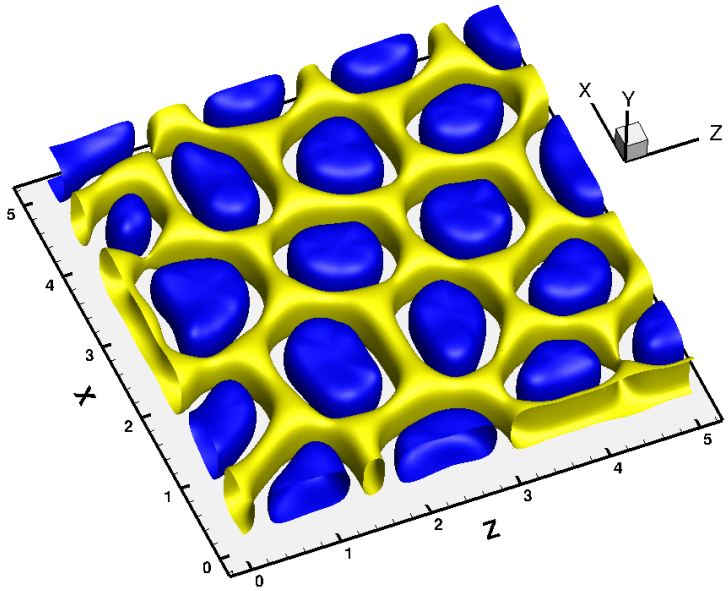


Fig. 5.1.13 Vertical velocity (Vy) iso-surfaces at Vy=1 (yellow) showing upward flow at convection cell perimeters, and Vy = -1 (blue) showing downward motion at the centre of hexagonal cells

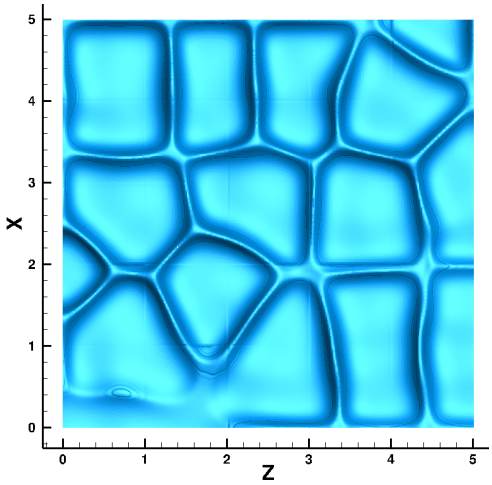


Fig. 5.1.14 Charge density iso-surfaces at 150 non-dimensional time (q=0.19)

In this simulation, the convective cells seemed to be evolving more towards attaining rectangular shapes as observed in the charge density iso-surface plot taken at the last time step (Fig. 5.1.14). According to the theoretical stability analysis [12], perfect hexagonal cells are preferred in infinite domain. Due to the limited numerical domain size and the constraint of numerical boundary conditions, we could see flow evolving with irregular hexagons and finally after a long time trying to stabilize in rectangular cells. We also noticed that the instability occurred around 50-time units for case 1 (Tab. 5.1), which was computationally expensive without any gain. To expedite the occurrence of instability and obtaining a stable hexagonal pattern, we initialized the flow with an artificial perturbation in vertical velocity component, in all following simulations.

Luo et al. (2018) reportedly used a special initialization technique to reproduce the regular hexagonal pattern, where they introduced a spatially periodic small perturbation in vertical velocity component. This perturbation is a hexagonal pattern which is given by Chandrasekhar function [37] as:

$$v = \frac{1}{3} V(y) \left[\cos(k_o z) + \cos\left(\frac{\sqrt{3} k_o x}{2}\right) \cos\left(\frac{k_o z}{2}\right) \right] \tag{5.7}$$

Where v is the vertical velocity, V(y) is the amplitude of the perturbation, k_o is the wave number corresponding the wavelengths of hexagonal cells; and z, x are the coordinate points on the boundary. Fig. 5.1.15 shows a sample domain with the contours of vertical velocity initialized with eq. (5.7). This function was used for all further simulations, as depicted in column 4 (Vin) of Tab. 5.1.

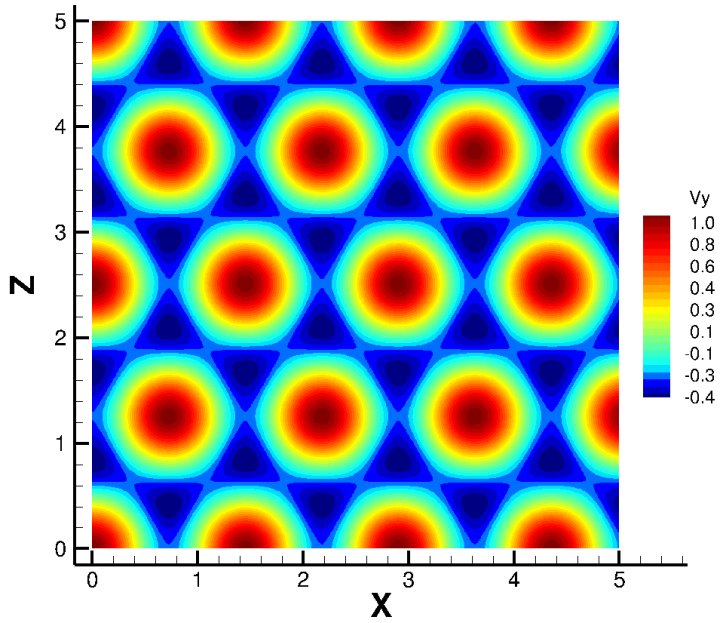


Fig. 5.1.15 Spatially periodic hexagonal perturbation contour

Case 2 (Tab. 5.1) was simulated with this perturbation in vertical velocity (eq. 5.7). Maximum velocity evolution and charge density iso-surfaces are shown in Fig. 5.1.16. The instability occurred within 20-time units in this case, which is the clear effect of using the initial perturbation. Near the onset of instability, the convection cells started developing with the initial hexagonal cells as shown in Fig. 5.1.16. The bun-shaped cells with iso-surfaces (q=0.02) were observed in this case also. Fig. 5.1.17 depicts, the charge density iso-surfaces with the MPI sub-domain interfaces, showing that one convection cells is spanned over many MPI processes, in all three directions; verifying that the code communicated data among many processes effectively to produce a continuous convection cell pattern. This simulation was carried out with 200 MPI processes. Although, the maximum velocity plot shows that the maximum velocity was much more stable than Case 1, however, the hexagonal cell pattern was not stable, and the flow evolved with irregular patterns as observed in Case 1.

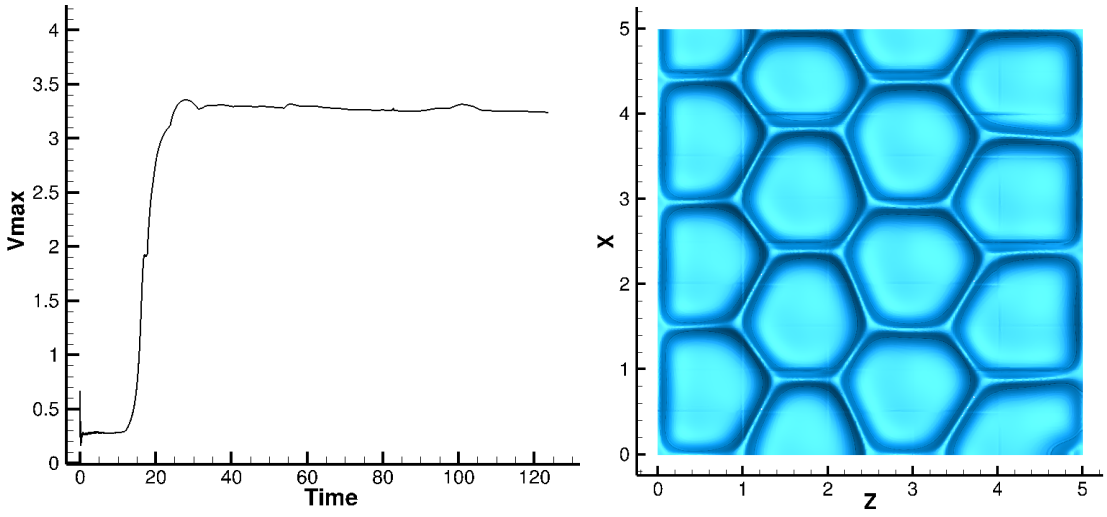


Fig. 5.1.16 Vmax evolution plot (case 2), and the top view with charge density iso-surfaces at q=0.19 at the onset of instability.

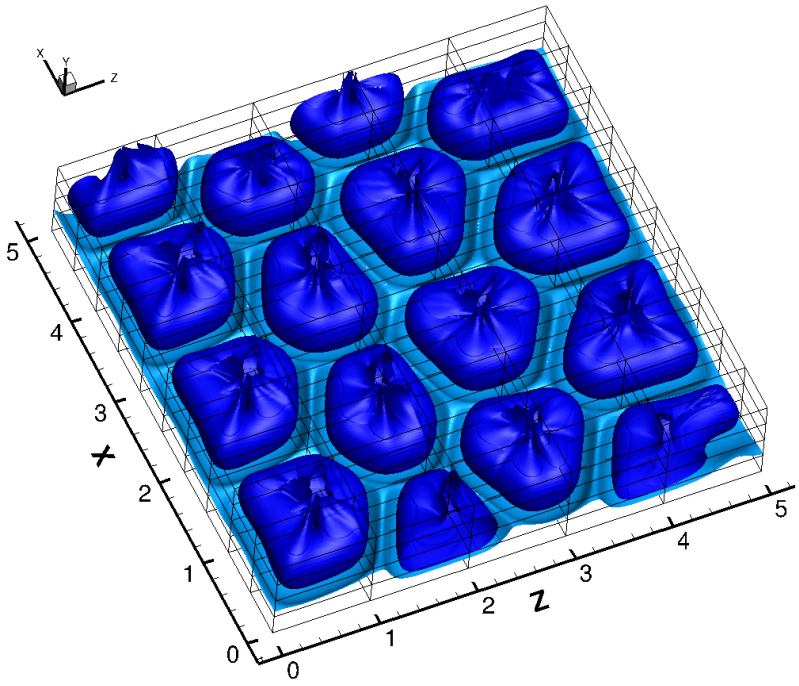


Fig. 5.1.17 Charge density iso-surfaces at q=0.19 (hexagonal cells), and q=0.02 (bun-shaped cells), at the onset of instability, with MPI domain decomposition.

.

Attempt was also made to simulate this flow in a domain with L=10, W=10, H=1. It was expected that the convection cells which are farther from the boundaries will be more stable than the ones near the boundaries. However, the hexagonal pattern did not stabilize even for the cells which were farther away from the boundaries. Fig. 5.1.18 shows the convection cell pattern at the onset of instability, where many irregular shaped cells are observed. The state of flow at the last time step was observed to be more chaotic.

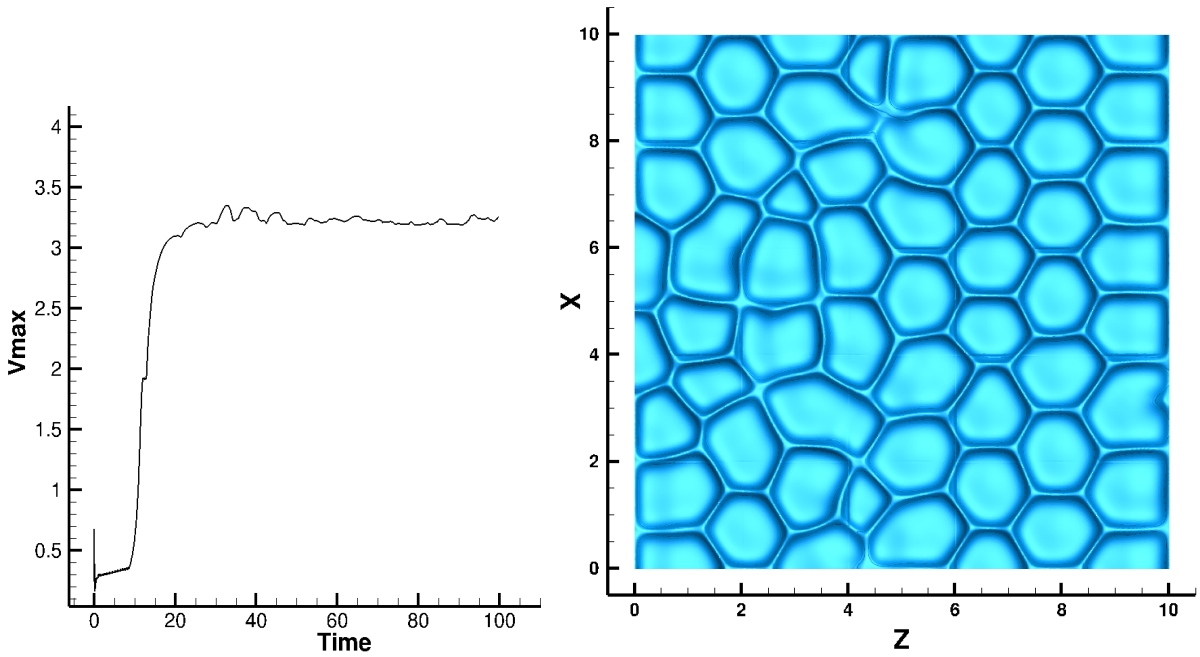


Fig. 5.1.18 Vmax evolution plot (case 3), and the top view with charge density iso-surfaces at q=0.19 at the onset of instability.

With all the simulations done with symmetry boundaries, we observed that there was a strong influence of the applied boundary conditions on the stability of the convective cells. The hexagonal pattern as observed in other studies is a spatially periodic pattern [12,37-38], where it is seen that the wavelength of hexagonal cells in z direction (λ_z) is $\sqrt{3}$ times the wavelength in x direction (λ_x) [38]. It was observed that in numerical simulations the domain size in multiples of wavelengths of the periodic pattern in both directions (x and z) is important to stabilize the convective cells [37,38]. In other words, if the domain size is such that it can fit a certain number of convection cells completely, then the whole flow will gain its optimal scale rather quickly and stabilize sooner than within a random sized domain. Also, with such domain sizes it becomes important to use the periodic boundary conditions.

As mentioned, the EC flow resembles with the RBC. The cellular pattern studies in RBC [41-47] have been available in literature for a longer period of time than the EC studies. Getling et al. (2003) have reported a numerical pattern study with RBC, where they discussed their numerical schemes and described the transition of pattern among various cell types. In EC flows, two studies reported in 2018 have used similar numerical techniques (mainly periodic BC), as Getling et al. (2003), and in the most recent one they observed the stable hexagonal pattern [38]. In our study, we did observe the hexagonal cell patterns even without using the hexagonal initial vertical velocity perturbations (Case 1), which Luo et al. (2018) reported as essential to obtain a fully developed state, but they were not stable. As Perez et al. (1989) noted that the role of diffusion is negligible in these injection phenomena, but to attain a steady state in numerical simulations we should consider diffusion of charge also. Luo et al. (2018) have also used a diffusion coefficient in the range of $10^{-3} – 10^{-4}$.

We attempted similar values of diffusion coefficient with our zero gradient boundary conditions but with diffusion also we could not stabilize the hexagonal pattern. Finally, we note that the zero gradient boundary conditions are not suitable to realise the stable hexagonal patterns, and periodic boundary conditions, with and without diffusion, should be examined with Oracle3D to obtain a stable EC flow. Some initial simulations with periodic boundaries were performed but for satisfactory results a complete dedicated study is required which in the constraint of time limit of this PhD could not be achieved.

5.2 Unipolar Injection between blade and plane electrodes

Electrohydrodynamically induced flow phenomena in dielectric liquids in case of blade-plane electrode configuration has been studied, both numerically [48, 53-55] and experimentally [56-58]. The EHD flow occurs in both injection and conduction mechanisms of charge transport in dielectric liquids. In this section, we mainly deal with the injection mechanism, which requires a certain threshold of electric field to occur and below this threshold value of electric field conduction dominates in dielectric liquids. Above this critical value of electric field, injection of charge particles occurs at the blade-fluid interface. In injection phenomena, the blade works as an emitter of charge particles, and the plane electrode behaves as a collector. The injected charged particles bring the surrounding fluid also into motion by transferring their momentum to neutral fluid particles. This sets the fluid motion like a jet towards the plane electrode. This jet like flow is commonly referred as an electrohydrodynamic plume [54].

This type of jet flow has been investigated for applications like heat transfer enhancement, mixing of fluid, flow control etc. Thus, like the thermal plumes, describing these EHD plumes is important from the industrial point of view. Vazquez et al. (1995) undertook a comparative study of thermal and EHD plumes, analysing axisymmetric plumes for various Prandtl numbers. Several following numerical studies by Vazquez et al. were performed with Finite element based numerical approaches to adequately describe the EHD plumes and their characteristics [59-61]. Perez et al. (2009) analysed the EHD plumes in blade-plane setting, with Finite Volume method using a TVD scheme (SMART) and addressed different flow regimes and characteristic flow structures in such EHD flows.

Recently, Traore et al. (2013) numerically analysed the EHD plume flows with different blade configurations, incorporating various injection laws [54]. They found significant impact of blade shape and injection laws on flow structure, specially the transition of flow from steady to unsteady regimes. Critical Reynolds number for the hyperbolic bade configuration was noted for considered injection laws and it was shown that for Reynolds number of 10000 the flow turns turbulent with development of a Kelvin-Helmoltz instability [54]. Wu et al. (2014) carried out a numerical heat transfer enhancement study with EHD plume flows. They also used a hyperbolic blade design in strong injection conditions. A dramatical increase in heat transfer rates from the plate electrode surface was reported with the application of impinging EHD jet flows [48]. Traore et al. (2015) also briefly reported an unsteady injection case, with electric Reynolds number of 200, describing the formation of Von Karman street of vortices emanating from the blade tip [55].

5.2.1 Problem Definition

Most of the above-mentioned studies were two dimensional. However, it is important to analyse the turbulent EHD plume cases in 3D, which are of immense practical use. A 3D blade-plane injection phenomenon in a dielectric liquid was studied with parallel Oracle3D. In blade-plane setting, the blade is used as a high voltage electrode, and the plane electrode is grounded. Both electrodes are separated by a distance d, as shown in Fig. 5.1.1. A voltage difference of $V_0 - V_1$, is applied between the two electrodes to generate an electric field in the gap, towards the plane.

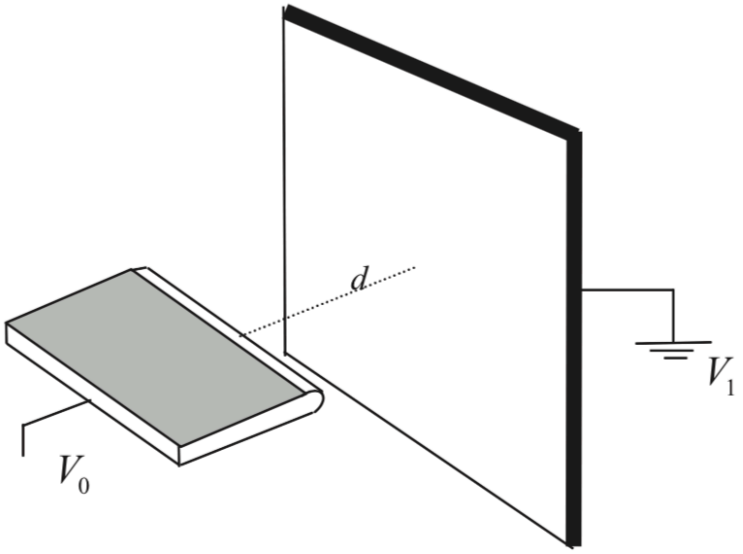


Fig. 5.2.1 Blade – plane configuration sketch

A hyperbolic blade-plane configuration was used for this study, which corresponds to blade 1 of Traore et al. (2013). The computational grid consists of 100 X 196 X 100 control volumes in X, Y and Z directions respectively, making it roughly a 2 million cells grid. Fig. 5.1.2 depicts the non-dimensional domain extents in all three directions and provides 2D and 3D views of grids. The grid near the blade tip and the surroundings was refined to capture the sharp gradients of charge in those regions. Also, this grid was created following the iso-potential and iso-electric field lines. It should be noted that the convergence rate improves in the grids which have grid lines aligned with the iso-potential and iso-electric field lines [54].

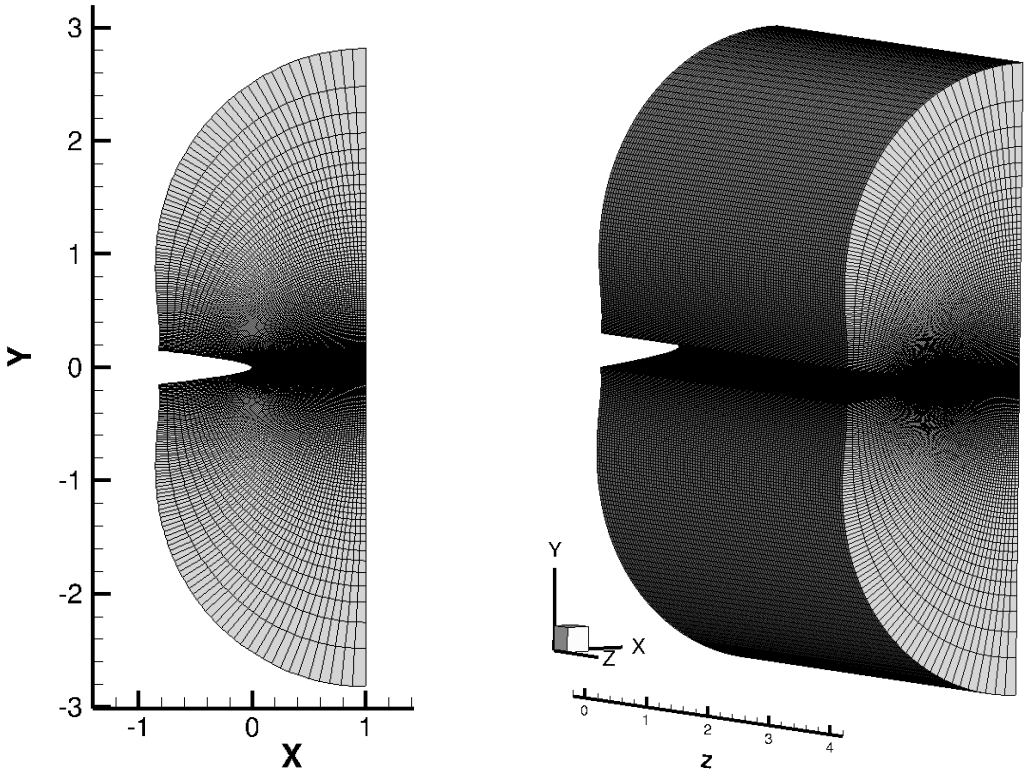


Fig. 5.2.2 Blade-plane computational grid: 2D and 3D views

Boundary conditions for this problem are described in Fig. 5.2.3, with a 2D sketch, zero gradient conditions were used on the front and back boundaries (z planes). The fluid is considered at rest at the start, and all the boundaries are set to no-slip condition for the velocity components. For electric potential, Dirichlet conditions of $V = 1$ and $V = 0$ are set on the blade and plane electrodes respectively. Electric charge density is set to $q = 1$, on the blade and rest of the boundaries are set to Neumann condition of zero charge density gradient. As detailed earlier, a 2nd order TVD scheme (SMART) was incorporated to discretize charge density, so that the sharp gradients could be preserved without numerical oscillations. The set of governing equations detailed in section 5.1.1 applies to this case also.

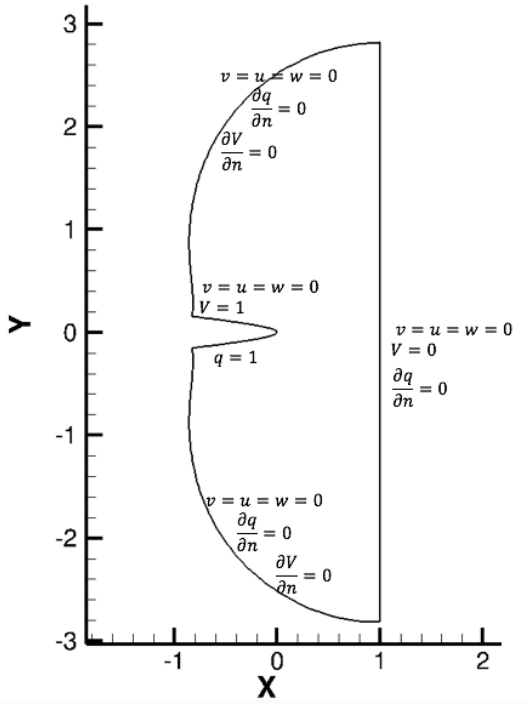


Fig. 5.2.3 Boundary conditions for blade-plane injection case

5.2.2 Initial simulation with autonomous injection law

An unsteady case, with R_{el}=5000, C=10 and M=10 was simulated for this study. Three injection laws were taken into consideration. Simulation with a classical autonomous injection law, in which the charge density on the blade surface is independent of the electric field on the blade surface, was performed as a reference case. It is observed in this case that the injection of charge extends over the whole surface of blade, Fig. 5.2.4. An isometric view of the charge density iso-surfaces with two values (q=0.1, 0.3) is presented in Fig. 5.2.4, which is taken at non-dimensional time t=0.003. An iso-surface plot showing the charge density distribution, on XY plane, at the end of simulation is shown in Fig. 5.2.5. The iso-surface at value q=0.3 (green) shows that the charge transports towards the plane electrode as a jet, and with time it expands quite a lot into the surrounding which leads to a good overall mixing of the neutral fluid and charge itself.

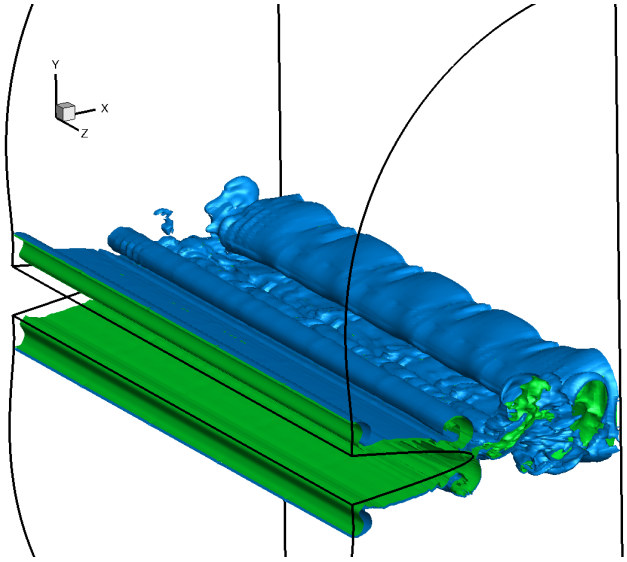


Fig. 5.2.4 Isometric-view of charge density iso-surfaces at q= 0.1 (blue) and q=0.3(green) with autonomous injection law

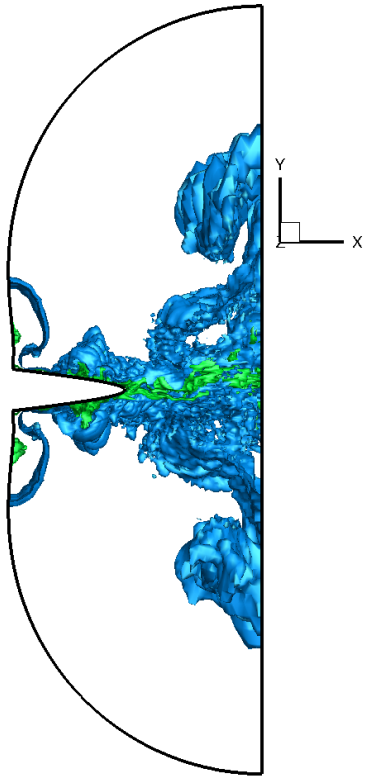


Fig. 5.2.5 Iso-surfaces of charge density (q=0.1, 0.3) as seen on XY plane

In second case, the charge density on blade surface was set to a Dirichlet value (q = 1) only where the local electric field was 60% of the maximum electric field. The 60% condition is arbitrarily set as suggested in [54]. It is noticed in experimental studies that the injection does not occur from whole surface of the blade as set with the Dirichlet boundary condition in the autonomous law, it is rather confined to the tip of the blade where the electric field is concentrated. Thus, in this case we restricted the charge injection from only those location of

blade where the electric field reached 60% of the maximum electric field. It has also been reported that blades with sharper tips inject more charge than the hyperbolic blade [54].

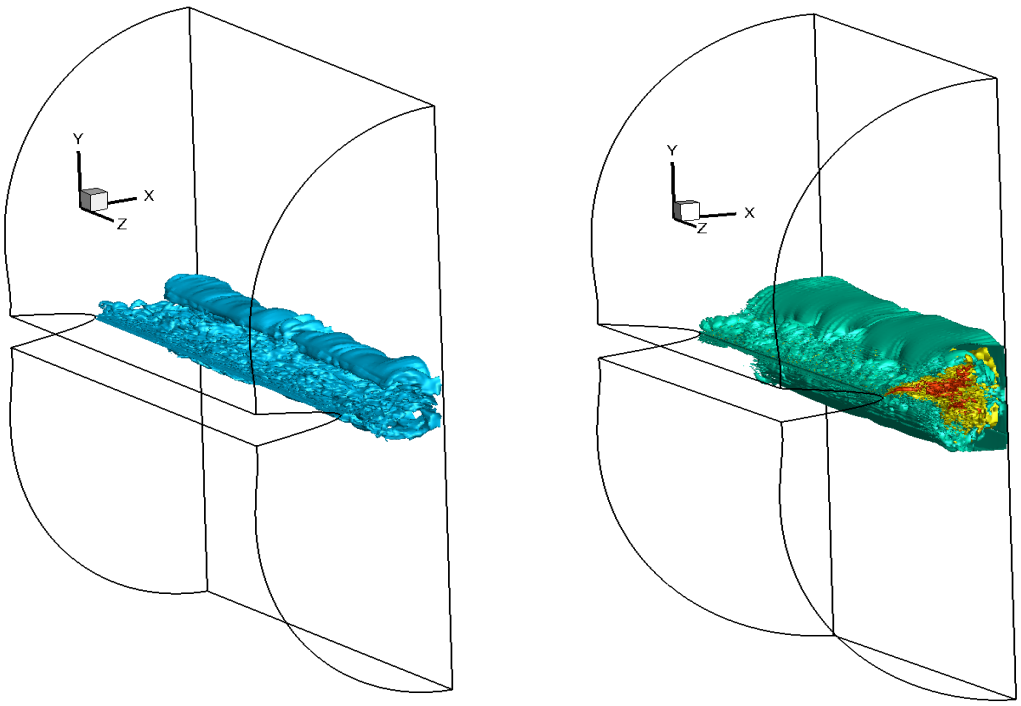


Fig. 5.2.6 a) Charge density iso-surfaces (q=0.1), and, b) Velocity magnitude iso-surfaces (Vmag= 12, 8, 3) at t=0.047

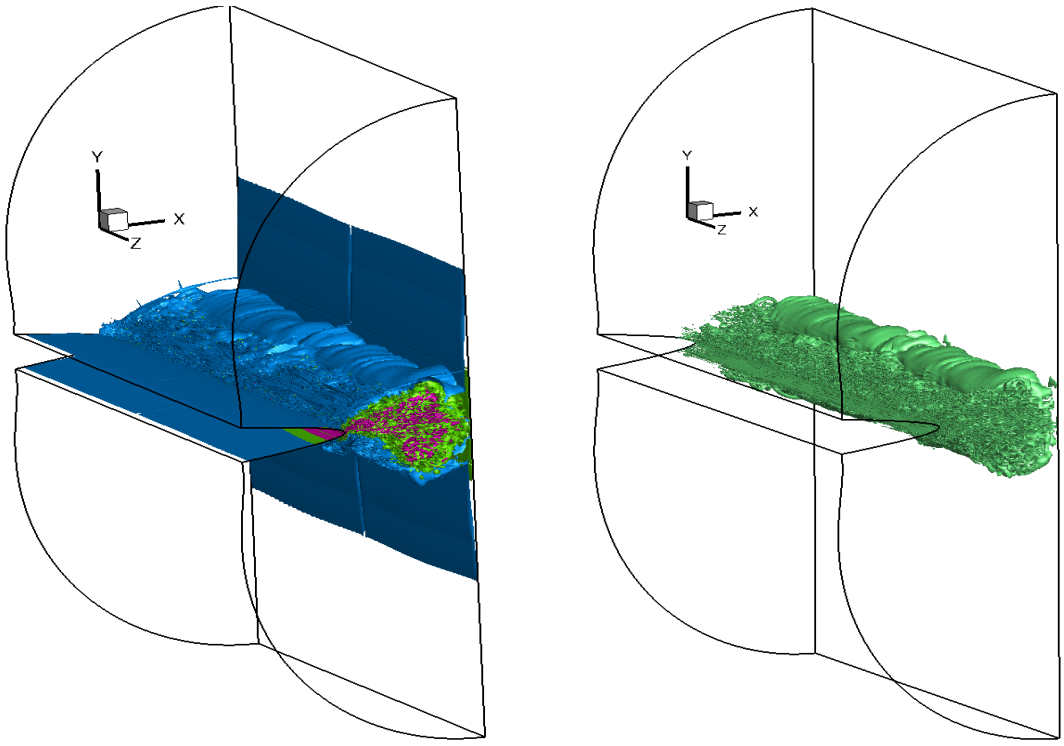


Fig. 5.2.7 a) Vorticity magnitude iso-surfaces (400, 200, 10), and, b) Q-criteria iso-surfaces (Q=100) at t=0.047 with 1st injection law.

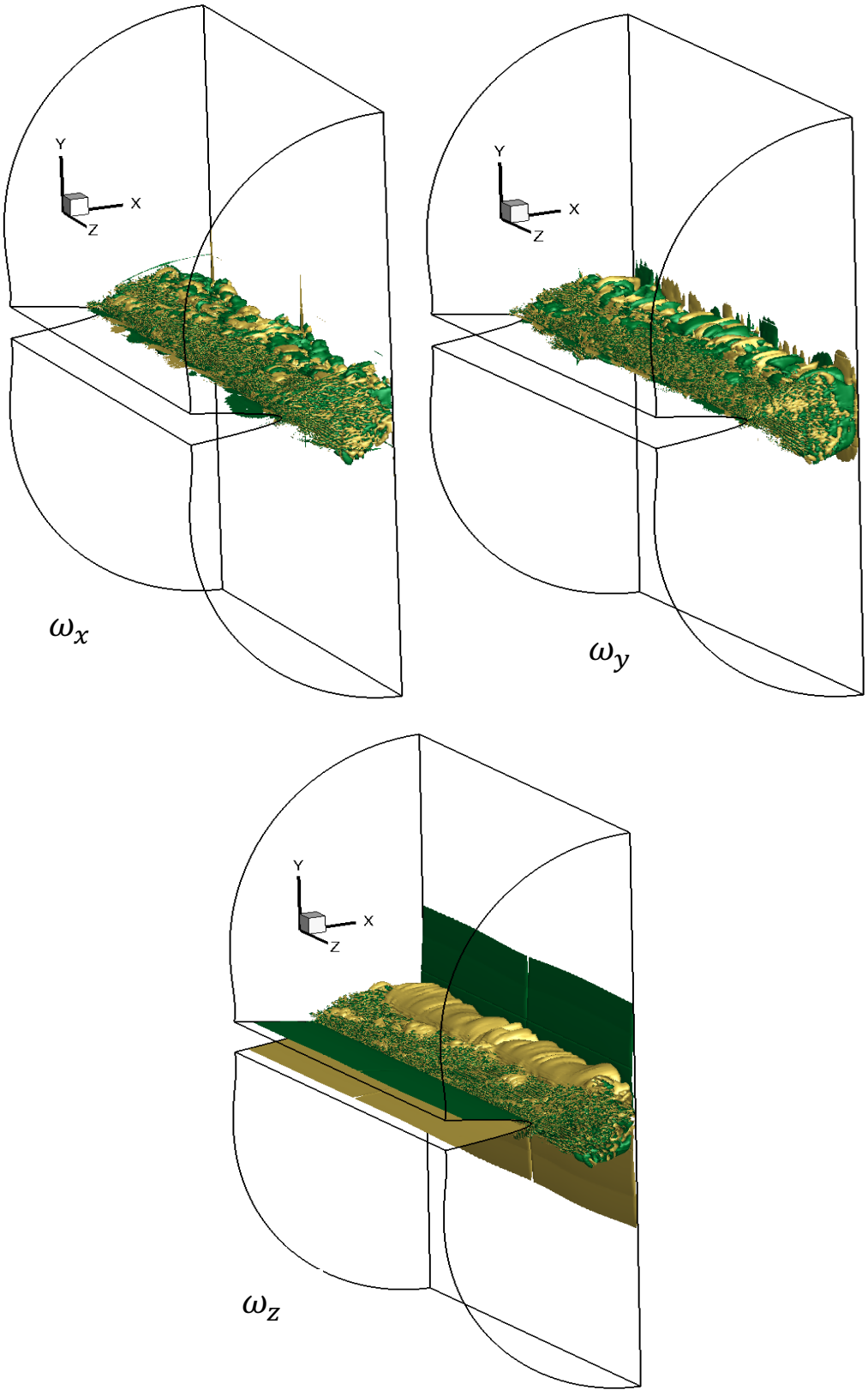


Fig. 5.2.8 Vorticity component iso-surfaces at values -20 (green) and +20 (yellow) at t=0.047

The results with our second case depicts that the injection is now limited from the tip of the blade electrode only. Fig. 5.2.6 a) shows that the injected charge density iso-surfaces at value q=0.1 are limited at the tip of the blade alone, and there is no charge present on the blade surface as predicted with the autonomous injection law. The velocity magnitude iso-surfaces with three values, Vmag = 12 (red), 8 (yellow), 3 (green) are shown in Fig. 5.2.6 b); which depicts that the charge injection plume is like a jet of velocity towards the plane electrode. Fig. 5.2.7 provides the information on rotational nature of the induced flow.

We have shown iso-surfaces of vorticity magnitude of three values: 400 (pink), 200 (green) and 10 (blue); which tells that adjacent to the plume the velocity gradients are of higher magnitude and vortical structures of several magnitudes are present in the flow describing it as a turbulent flow. Q-criteria iso-surfaces (Q=100) are also shown to describe the real 3D nature of the flow, which aids to the mixing behaviour of the flow very well. Fig. 5.2.8 depicts the iso-surfaces of three components of vorticity, at magnitudes -20 and +20, which are completely distinct from each other. It suggests that the flow rotation is random in all three directions which promotes the idea of turbulent mixing in such configurations, leading to efficient heat transfer mechanism from the plane electrode surface by EHD plumes [48].

5.2.3 Simulations with non-autonomous Injection laws

After understanding the basic idea of the phenomena, we incorporated two injection laws as simulated by Traore et al. (2013). With injection laws stronger coupling between the variables like charge, electric field, velocity etc. is induced. We simulated with 1st and 3rd injection laws from ref. [54], in which the boundary condition for charge density on the blade electrode is set as:

1st injection law: $q = C$, where $E \geq 0.6 * E_{max}$

3rd injection law: $q = \frac{E_{local}}{E_c} C$

In both of these injection laws the injection zone of the charge is limited to the part of blade electrode where the local electric field reaches certain percentage of the maximum electric field (E_{max}). Here this threshold electric field (E_c) is taken as 60 % of the E_{max}. For 3rd law $E_c = 0.6 * E_{max}$ is the threshold electric field to initiate injection. E_{local} is the local electric field on the nodes of the blade, and C represents the injection strength as given in eq. (5.6). Thus, each node on the blade surface which satisfies $E \geq 0.6 * E_{max}$ will inject charge density in 1st law, and in 3rd law injection depends on E_c value. We shall provide a comparison of these two laws with relevant flow variables and also comment on the studies previously done in two-dimensional settings. In the following section, the comparison figures on left hand side will correspond to the 1st law and on the right side will correspond to 3rd law, unless mentioned specifically.

Firstly, we analyse the 3D evolution of charge density with both laws. Next three figures (5.2.9-5.2.11) provide instantaneous charge density iso-surfaces with two values (q=0.1, 0.5). Fig. 5.2.9 gives a 3D isometric view at non-dimensional time 0.0017, which expresses that the 1st law charge distribution is rather smoother than 3rd law, and also the charge has diffused more with the 3rd law, with a higher rate of turbulent mixing. Smooth surfaces of charge as seen in 1st law distribution is not visible in 3rd law figure. Fig. 5.2.10 is also taken at the same instant of time which shows a view perpendicular to the XY plane. It can be seen that the amount of charge diffused in the domain is also higher in case of 3rd law.

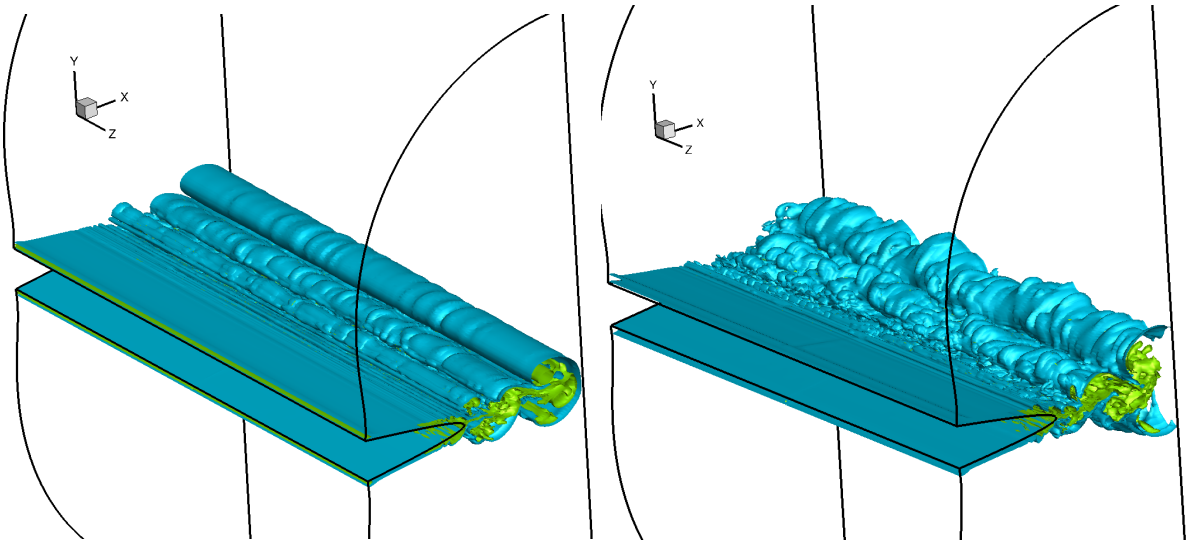


Fig. 5.2.9 3D c*harge density iso-surfaces (q = 0.1 (blue) and q = 0.5 (green)) at non-dimensional time 0.0017, for 1st and 3rd laws*

A tendency to form initial coherent vortices is seen in both flows at the onset of injection, but the injection strength is such that the initial vortices are quickly diffused due to the upcoming jet of charge just behind them. Three-dimensional mixing diffuses the charge so quickly that initial and upcoming vortices are not at all sustained in the flow, as seen in some 2D studies performed with Rey=10000 [54]. Sustained vortex shedding with Kelvin-Helmholtz instability was observed in the 2D study [54] which was not observed in our 3D cases. Fig. 5.2.11 shows the charge density distribution at t=0.02 which also suggests that 3rd injection has higher amount of charge injected than with 1st law. Fig. 5.2.12 is taken at t=0.01, the jet of charge impinging on the plane electrode with green coloured iso-surface (q=0.5) is visible with a corresponding velocity magnitude iso-surface plot at Vmag=8 (yellow) and Vmag=15 (red).

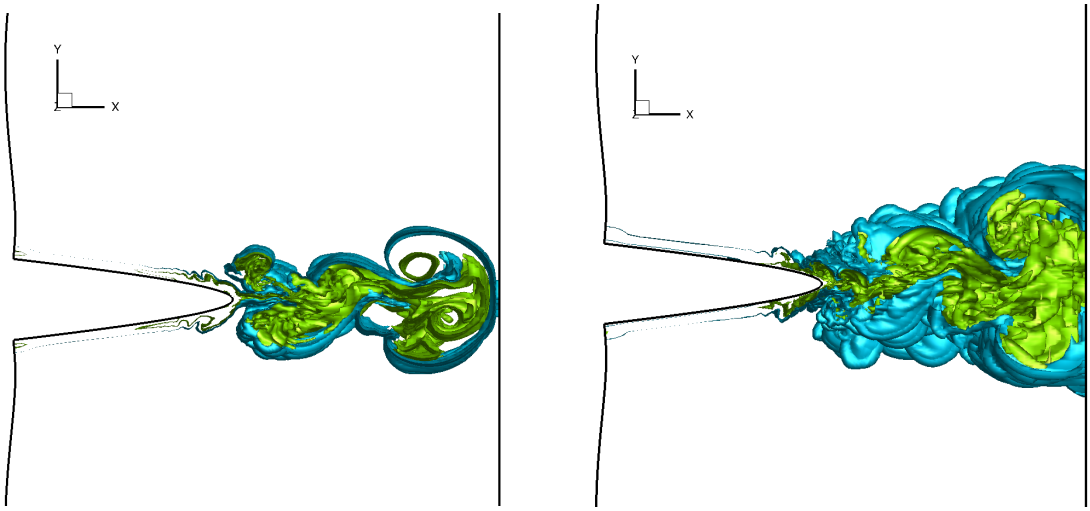


Fig. 5.2.10 Two-dimensional c*harge density iso-surfaces (q = 0.1 (blue) and q = 0.5 (green)) at non-dimensional time 0.0017, for 1st and 3rd laws*

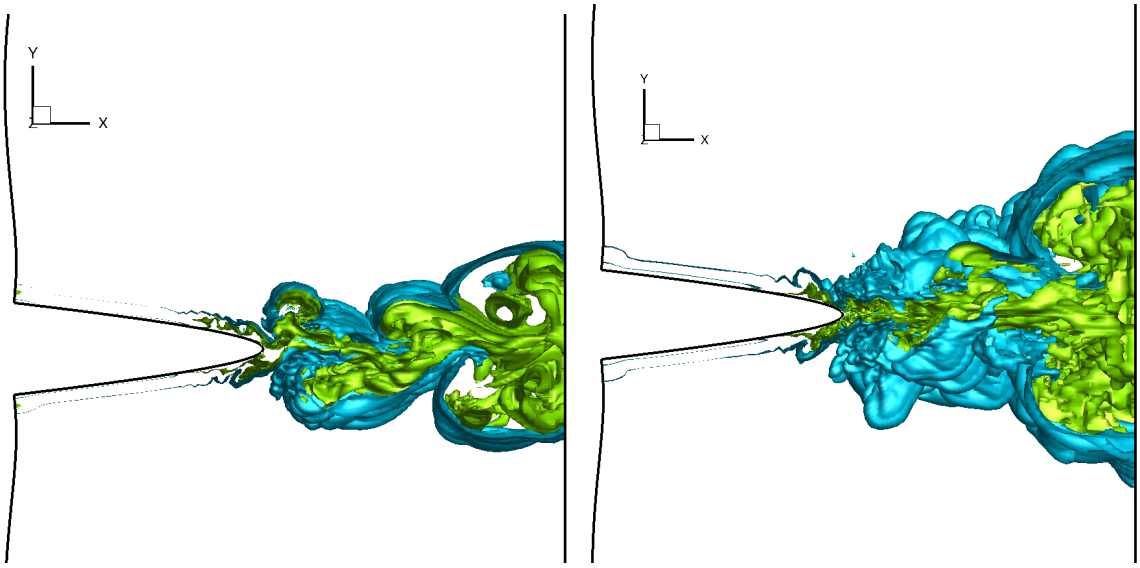


Fig. 5.2.11 2D slice of charge density iso-surfaces *(q = 0.1 (blue) and q = 0.5 (green)) at non-dimensional time 0.002, for 1st and 3rd laws*

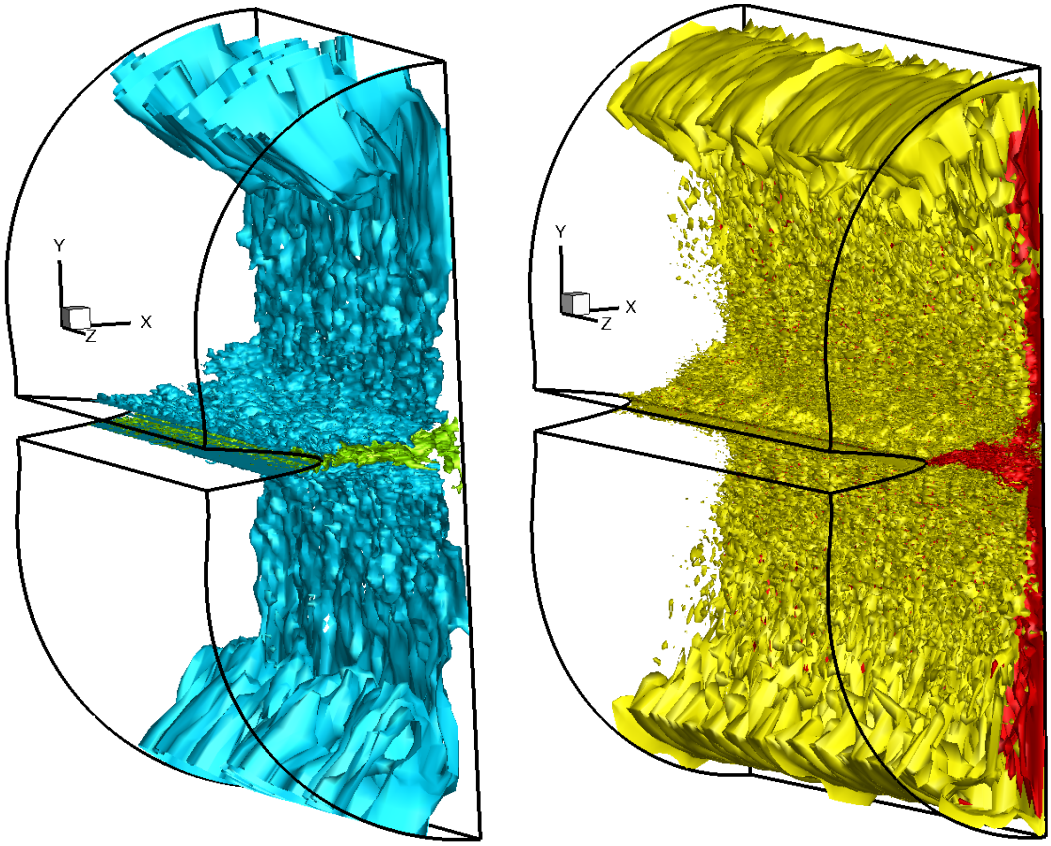


*Fig. 5.2.12 Iso-metric view of a)*charge density iso-surfaces *(q=0.1,0.5), and, b) velocity magnitude (Vmag=8, 15) , at t=0.01 (3rd law)*

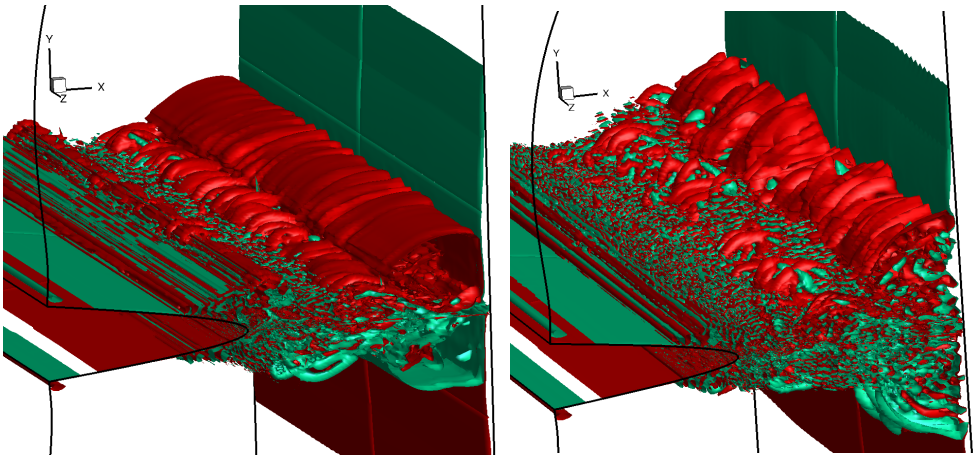


Fig. 5.2.13 3D z-vorticity iso-surfaces ($\omega_z = -50 \text{ and } +50$) at non-dimensional time 0.002, for 1st and 3rd laws

Two initial vortices rotating in clockwise and counter-clockwise directions are observed before the flow sets into complete turbulent motion. The jet moves in +x direction and z-vorticity shows the prominent rotating vortices along the z direction. Fig. 5.2.13 shows z-vorticity iso-surfaces at -50(green) and +50(red) values which are present clockwise and anti-clockwise rotation respectively. The plume contains vortices of many sizes as seen in Fig. 5.2.13, and we also notice that with 3rd law the vortices are diffused faster than the 1st law.

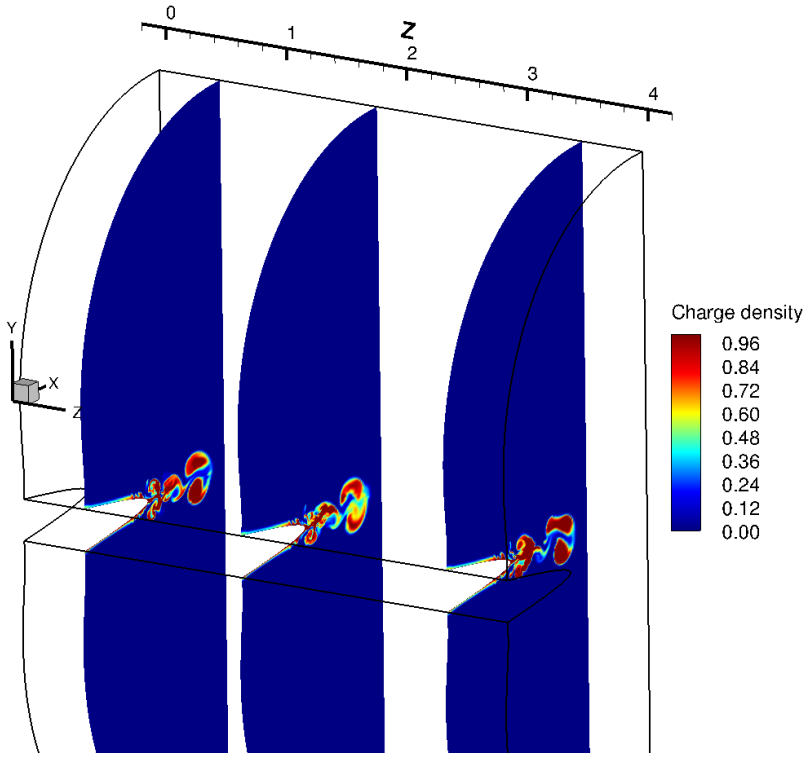


Fig. 5.2.14 2D Charge density contours in three different planes in Z direction at t=1.2 X 10^{-3} with 3rd injection law.

Two-dimensional charge density contours were plotted in three different planes (Z= 0.5, 1.8, 3.5) to observe the 3D nature of charge propagation. Fig. 5.2.14 shows that in 3 planes the charge density contours are distinct with each other. Fig. 5.2.15 proves that the flow has strong 3D features as the z-component of velocity has a significant magnitude and the contours of this velocity (W) are also different in 3 planes. Fig. 5.2.16 the 3D vortical structures of the flow with Q-criteria and vorticity magnitude variables.

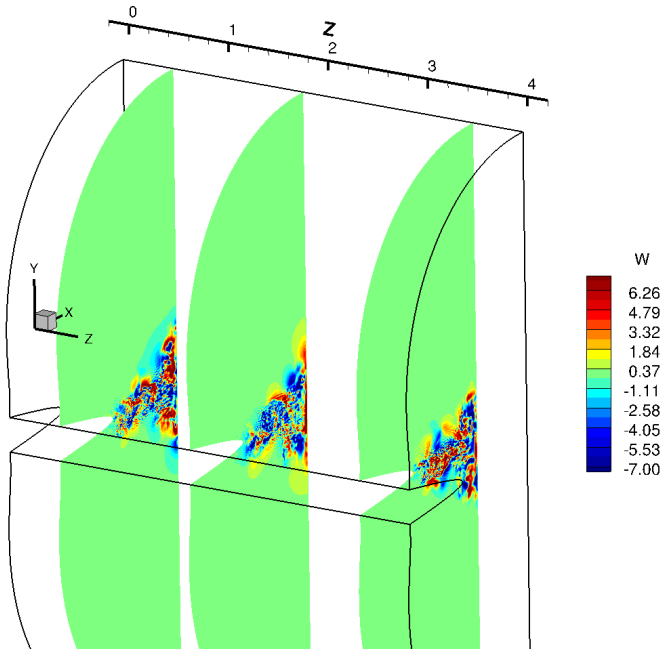


Fig. 5.2.15 Z-component of velocity contours in three different planes in Z direction at t=1.2 X 10^{-3} with 3rd injection law.

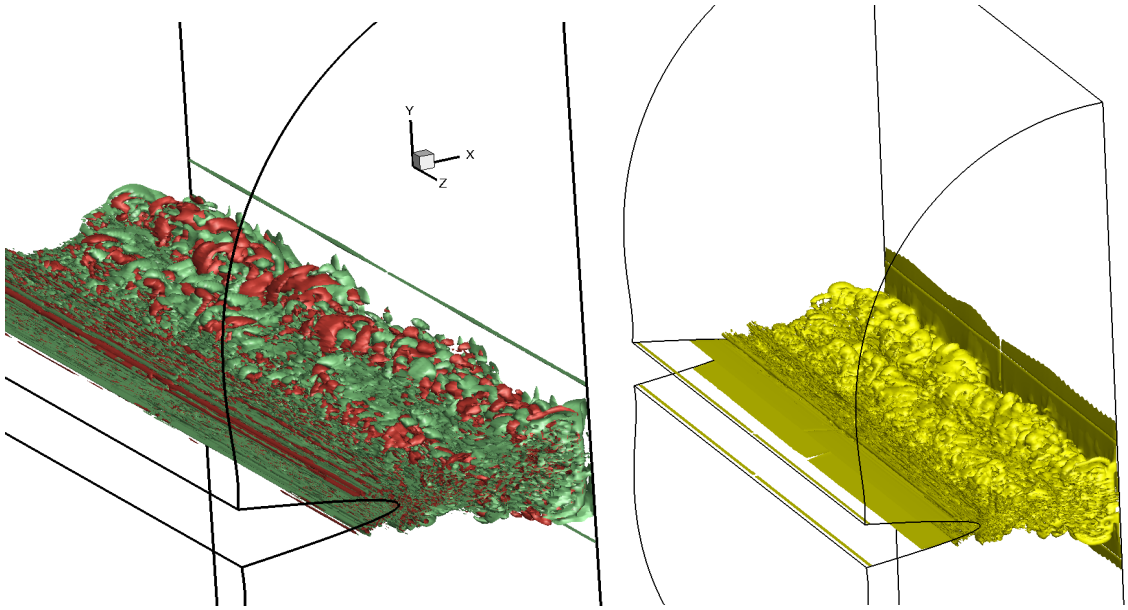


Fig. 5.2.16 a) Q-criteria iso-surfaces (Q=-20,000 (green), +20,000 (red)), and, b) Vorticity magnitude iso-surfaces($|\omega| = 300$), at t=1.6 X 10^{-3} with 3rd injection law

Bibliography

[1] P. Traore, A. T. Perez, "Two-dimensional numerical analysis of electroconvection in a dielectric liquid subjected to strong unipolar injection," Phys. Fluids 24, 037102 (2012).

[2] J. Wu, P. Traore, P. A. Vazquez, A. T. Perez, "Onset of convection in a finite two-dimensional container due to unipolar injection of ions," Phys. Rev. E 88, 053018 (2013).

[3] P. Traroe, J. Wu, "On the limitation of imposed velocity field strategy for Coulomb-driven electroconvection flow simulations," J. Fluid Mech. 727, R3 (2013).

[4] J. Wu, P. Traore, "A finite-volume method for electro Thermoconvective phenomena in a plane layer of dielectric liquid," Numer. Heat Transfer, Part A 68, 471 (2015).

[5] J. Wu, A. T. Perez, P. Traore, P. A. Vazquez, "Complex flow patterns at the onset of annular electroconvection in a dielectric liquid subjected to an arbitrary unipolar injection," IEEE Trans. Dielectr. Insul. 22, 2637 (2015).

[6] J. Wu, P. Traore, A. T. Perez, P. A. Vazquez, "On two-dimensional finite amplitude electro-convection in a dielectric liquid induced by a strong unipolar injection," J. Electrost. 74, 85 (2015).

[7] J. Wu, P. Traore, C. Louste, "An efficient finite volume method for electric field-space charge coupled problems,", J. Electrost. 71, 319-325, (2013).

[8] J. Wu, P. Traore, F. Tian, "Three-dimensional numerical simulation of electro-convection due to strong unipolar charge injection in a cubic cavity," Annual Report Conference on Electrical Insulation And Dielectric Phenomena, IEEE, (2013)

[9] J. Wu, "Contribution to numerical simulation of electrohydrodynamics flows: application to electro-convection and electro-thermo-convection between two parallel plates." PhD. Thesis, University of Poitiers (2012)

[10] D. Koulova, P. Traore, H. Romat, "Numerical study of the heat transfer and electro-thermo-convective flow patterns in dielectric liquid layer subjected to unipolar injection," J. Elecctrost. 71, 970-975, (2013)

[11] U. K. Seth, P. Traore, F. J. Duran-Olivencia, E. Moreau, P. A. Vazquez, "Parametric study of a DBD plasma actuator based on the Suze-Huang model," J. Electrost. 93, 1-9, (2018)

[12] P. Atten, J. C. Lacroix, "Non-linear hydrodynamic stability of liquids subjected to unipolar injection," J. Mec. 18 (1979).

[13] P. Atten, J. C. Lacroix, "Electrohydrodynamic stability of liquids subjected to unipolar injection: Non-linear phenomena," J. Electrost. 5, 439-452, (1978).

[14] B. Malraison, P. Atten, "Chaotic behavior of instability due to unipolar ion injection in a dielectric liquid," Phys. Rev. Lett. 49, 10, 723-726 (1982).

[15] P. Atten, T. Honda, "The electroviscous effect and its explanation I- the electrohydrodynamic origin; study under unipolar D. C. injection," J. Electrost. 11, 225-245 (1982)

[16] P. Atten, B. Malraison, S. A. Kani, "Electrohydrodynamic stability of dielectric liquids subjected to A. C. Fields," J. Electrost. 12, 477-488 (1982)

[17] P. Atten, L. Elouadie, "EHD convection in a dielectric liquid subjected to unipolar injection: coaxial wire/cylinder geometry," J. Electrost. 34, 279-297 (1995)

[18] P. Atten, "Electrohydrodynamic instability and motion induced by injected space charge in insulating liquids," IEEE Trans. Dielect. Elect. Insulation, Vol 3, 1, 1996

[19] A. Perez, A Castellanos, "Laminar chaotic transport of charge in finite amplitude electroconvection," Phys. Rev. A 44, 6659 (1991).

[20] A. Perez, A Castellanos, "Role of charge diffusion in finite-amplitude electroconvection," Phys. Rev. A 40, 10 (1989)

[21] A. Castellanos, P. Atten, Numerical modeling of finite amplitude convection of liquids subjected to unipolar injection," IEEE Trans. Ind, Appl. IA-23, 825 (1987).

[22] A. Castellanos, "Injection induced instabilities and chaos in electrohydrodynamics," J. Phys.: Condens. Matter 2, 499-503 (1990)

[23] R. Chicon, A. Castellanos, E. Martin, "Numerical modelling of Coulomb-driven convection in insulating liquids," J. Fluid Mech. 344, 43-66 (1997)

[24] K. Adamiak, P. Atten, "Simulation of corona discharge in point-plane configuration," J. Electrost. 61, 85 (2004).

[25] A. Castellanos, Electrohydrodynamic, Springer Publication, (1998).

[26] R. Chicon, A. Perez, A. Castellanos, "Modelling the Finite Amplitude Electroconvection in Cylindrical Geometry: Characterization of Chaos," Annual Report Conference On Electrical Insulation And Dielectric Phenomena, IEEE, (2002)

[27] P. A. Vazquez, G. E. Georghiou, A. Castellanos, "Numerical analysis of the stability of the electrohydrodynamic (EHD) electroconvection between two plates," J. Phys. D: Appl. Phys. 41 (2008)

[28] P. A. Vazquez, A. Castellanos, "Stability analysis of the 3D Electroconvective Charged Flow Between Parallel Plates Using the Particle-in-Cell Method," IEEE Internt. Conference on Dielectric Liquids (2011)

[29] A. T. Perez, P. A. Vazquez, J. Wu, P. Traore, "Electrohydrodynamic linear stability analysis of dielectric liquids subjected to unipolar injection in a rectangular enclosure with rigid sidewalls," J. Fluid Mech. 758, 586-602 (2014)

[30] P. A. Vazquez, A. Perez, A Castellanos, “Thermal and electrohydrodynamic plumes. A comparative study,” Phys. Fluids 8, 2091 (1996)

[31] E. A. Demekhin, V. S. Shelistov, S. V. Polyanskikh, “Linear and nonlinear evolution and diffusion layer selection in electrokinetic instability,” Phys. Rev. E 84, 036318 (2011).

[32] E. A. Demekhin, N. V. Nikitin, V. S. Shelistov, “Direct numerical simulation of electrokinetic instability and transition to chaotic motion,” Phys. Fluids 25, 122001 (2013)

[33] E. A. Demekhin, N. V. Nikitin, V. S. Shelistov, “Three-dimensional coherent structures of electrokinetic instability,” Phys. Rev. E 90, 013031 (2014).

[34] M. Zhang, F. Martinelli, J. Wu, P. J. Schmid, M. Quadrio, “Model and non-model stability analysis of electrohydrodynamic flow with and without cross-flow,” J. Fluid Mech. 770, 319-349 (2015)

[35] K. Luo, J. Wu, H. Yi, H. Tan, “Lattice Boltzmann model for Coulomb-driven flows in dielectric liquids,” Phys. Rev. E 93, 023309 (2016)

[36] K. Luo, J. Wu, H. Yi, H. Tan, “Lattice Boltzmann modelling of electro-thermo-convection in a planar layer of dielectric liquid subjected to unipolar injection and thermal gradient,” Intern. J. Heat and Mass Transf. 103, 832-846 (2016)

[37] K. Luo, J. Wu, H. Yi, H. Tan, “Three-dimensional finite amplitude electroconvection in dielectric liquids,” Phys. Fluids 30, 023602 (2018)

[38] K. Luo, J. Wu, H. Yi, L. Liu, H. Tan, “Hexagonal convective patterns and their evolutionary scenarios in electroconvection induced by a strong unipolar injection,” Phys. Rev. Fluids 3, 053702 (2018)

[39] F. H. Busse, “Non-linear properites of thermal convection,” Rep. Prog. Phys. 41, 1978

[40] H. S. Greenside, W. M. Coughran Jr., N. L. Schryer, “Nonlinear Pattern Formation near the Onest of Rayleigh-Benard Convection,” Phys. Rev. Lett. 49, No. 10 (1982).

[41] V. Steinberg, G. Ahlers, D. S. Cannell, “Pattern formation and wave-number selection by Rayleigh-Benard convection in a cylindrical container,” Physica Scripta. T9, 97-110 (1985)

[42] F. H. Busse, R. M. Clever, “Asymmetric Squares as Attracting Set in Rayleigh-Benard Convection,” Phys. Rev. Lett. 81, No. 2 (1998).

[43] R. M. Clever, F. H. Busse, “Hexagonal convection cells under conditions of vertical symmetry,” Phys. Rev. E 53, No. 3 (1996).

[44] Michel Assenheimer, Victor Steinberg, “Observation of Coexisting Upflow and Downflow Hexagons in Boussinesq Rayleigh-Benard Convection,” Phys. Rev. Lett. 76, No. 5 (1996).

[45] E. Bodenschatz, W. Pesch, G. Ahlers, “Recent developments in Rayleigh-Benard convection,” Annu. Rev. Fluid Mech. 32, 709-778 (2000)

[46] A. V. Getling, O. Brausch, "Cellular flow patterns and their evolutionary scenarios in three-dimensional Rayleigh-Benard convection," Phys. Rev. E 67, 0463133 (2003)

[47] D. C. Rapaport, "Hexagonal convection patterns in atomistically simulated fluids," Phys. Rev. E 73, 025301 (R) (2006).

[48] J. Wu, P. Traore, C. Louste, A. T. Perez, P. A. Vazquez, "Heat transfer enhancement by an electrohydrodynamic plume induced by ion injection from a hyperbolic blade," IEEE Int. Conference on Liquid Dielectrics, (2014).

[49] J. E. Bryan, J. S. Yagoobi, "Electrohydrodynamically enhanced convective boiling: Relationship between electrohydrodynamic pressure and momentum flux rate," J. Heat Transf. 122, 266 (1999)

[50] J. S. Yagoobi, "Electrohydrodynamic pumping of dielectric liquids." J. Elecctrost. 63, 861 (2005)

[51] P. Traore, M. Daaboul, C. Louste, "Numerical simulation and PIV experimental analysis of electrohydrodynamic plumes induced by a blade electrode." J. Phys. D: Appl. Phys. 43, (2010)

[52] C. Louste, Z. Yan, P. Traore, R. Sosa, "Electroconvective flow induced by dielectric barrier injection in silicone oil," J. Electrost. 71, 504 (2013)

[53] J. H. Davidson, P. J. McKinney, "Turbulent mixing in a barbed plate-to-plate electrostatic precipitator," Atmosp. Environ. 23, 2093 (1989)

[54] P. Traore, J.Wu, C. Louste, Q. Pelletier, L. Dascalescu, "Electro-Hydro-Dynamic plumes due to autonomous and non-autonomous charge injection by a sharp blade electrode in a dielectric liquid," IEEE Transc. Industry Applications, (2013)

[55] P. Traore, J. Wu, P. A. Vazquez, C. Louste, C. Gouriou, A. Perez, "Numerical simulation of electrohydrodynamically induced dielectric liquid flow through pure conduction blade-plane geometry," 11th Int. Conference on Modern Problems of Electrophysics and Electrohydrodynamics (MPEE-2015)

[56] F. M. J. McCluskey, A. T. Perez, "The electrohydrodynamic plume between a line source or ions and a flat plate," IEEE Trans. Electr. Insulation 2(27), 334 (1992)

[57] B. Malraison, P. Atten, A. T. Perez, Panches charges resultant de l'injection d'ions dans un liquid isolant par une lame ou une pointe placee en face d'un plan, J. Phys. III France 4, 75 (1994)

[58] A. T. Perez, P. A. Vazquez, A. Castellanos, "Dynamics and linear stability of charged jets in dielectric liquids," IEEE Trans. Industry Appl. 31, 761 (1995)

[59] P. A. Vazquez, E. C. Vera, A. Castellanos, T. C. Rebollo, "Finite element-particle method calculation of EHD plumes," Annual Report Conference on Electrical Insulation and Dielectric Phenomena, IEEE, 208-211 (2002)

[60] P. A. Vazquez, E. C. Vera, A. Castellanos, T. C. Rebollo, “Finite element-particle method calculation of EHD plumes,” Annual Report Conference on Electrical Insulation and Dielectric Phenomena, IEEE, 706-709 (2003)

[61] P. A. Vazquez, C. Soria, A. Castellanos, “Numerical simulation of two-dimensional EHD plumes mixing finite element and particle methods,” Annual Report Conference on Electrical Insulation and Dielectric Phenomena, IEEE, 122-125 (2004)

Chapter 6
EHD Conduction

In this chapter we present the electro-conduction model and some initial case studies performed with some new implementations in the conduction model as available in Oracle3D. Mainly, some validation cases with flow feature analysis in a conduction channel configuration were compared with COMSOL solutions and results are reported. Results with newly implemented Robin and non-homogeneous Neumann boundary conditions are presented and their physical significance is discussed. Impact of mathematical formulations: implicit and explicit in case of FVM discretization of transport of species, and effect of Onsager effect on EHD conduction is briefly highlighted. In last section, flow pattern with blade-plane electrode configuration with a case is discussed in both 2D and 3D.

6.1 Introduction

The electric charges present in Electrohydrodynamic (EHD) conduction are created by dissociation and recombination of a weak electrolyte in a non-polar or mildly polar liquid. When an external electric field is applied, layers with a net electric charge appear next to each electrode. These are the heterocharge layers, with a polarity opposed to the one of the electrodes. The motion of charged species in the bulk liquid is due to the electric force density, $\vec{F_e}$, which results from three different physical components. The first and the most dominant component is the Coulomb force which is the first term on right side in eq. (1). The second term is the dielectric force which is present only when the permittivity gradient ($\nabla\varepsilon$) exists. The third term is known as the electrostrictive force which, being the gradient of a scalar, can be incorporatd in pressure [1,2].

$$\vec{F_e} = q\vec{E} - \frac{1}{2} E^2 \nabla\varepsilon + \nabla\left[\rho \frac{E^2}{2}\left(\frac{\partial\varepsilon}{\partial\rho}\right)_T\right] \quad (1)$$

Thus, the Coulomb force alone is left which sustain a permanent EHD motion in such conduction phenomena. In most of the EHD applications, this Coulomb force sets the liquid in motion which is utilized for the intended purposes like pumping, wall jets etc. The net Coulomb force is generated only if there is an imbalance in the densities of positive and negative charge carriers. Asymmetrical electrode configurations play an important role in creating this imbalance in charge carriers' densities, which have been explored in many studies [2-4]. The mechanism of EHD conduction provides a non-mechanical and low-power consumption approach to generate or control an active flow, which can be used for applications targeted for terrestrial and microgravity conditions [6].

6.2 Mathematical model

The electro-conduction model as available in Oracle3D is based on the original model proposed by Atten et al. (2003). This model considers chemical dissociation and recombination of neutral liquid molecules within a reversible reaction. The concentrations of produced ions in the parent liquid are controlled by the dissociation (k_D) and recombination (k_R) rates. A general reversible chemical equation can be given as:

$$AB \underset{k_R}{\overset{k_D}{\rightleftharpoons}} A^+ + B^-$$

If c is the concentration of neutral molecules and N_p and N_n are the charge densities of positive and negative ion species, then at the thermodynamic equilibrium one can write:

$$k_D c = k_R N_{p0} N_{n0} = k_R N_{p0}^2 = k_R N_{eq}^2 = k_R N_{eq}^2$$

Where '0' refers to the values at equilibrium, and $N_{p0} = N_{n0} = N_{eq}$ follows from the equilibrium condition, which results in $N_{p0} = \sqrt{k_D c / k_R}$. The transport mechanism of the species under the influence of external electric field is governed by following set of equations:

$$\frac{\partial N_p}{\partial t} + \nabla . \vec{J}_+ = k_D c - k_R N_p N_n \tag{6.1}$$

$$\frac{\partial N_n}{\partial t} + \nabla . \vec{J}_- = k_D c - k_R N_p N_n \tag{6.2}$$

$$\frac{\partial c}{\partial t} + \nabla . \Phi_{AB} = k_R N_p N_n - k_D c \tag{6.3}$$

Where the current density fluxes are provided by:

$$\vec{J}_+ = \vec{u} N_p + K_+ N_p \vec{E} - D_+ \nabla N_p \tag{6.4}$$

$$\vec{J}_- = \vec{u} N_n - K_- N_n \vec{E} - D_- \nabla N_n \tag{6.5}$$

$$\Phi_{AB} = D_{AB} \nabla c \tag{6.6}$$

In the above equations the mobility and diffusion coefficients of positive and negative ion species are given by K_+, K_-, D_+ , D_- respectively. The overall liquid velocity is denoted by $\vec{u}$.

We can write the Gauss law to obtain the electric field ($\vec{E}$) due to the species as:

$$\nabla . (\varepsilon \vec{E}) = N_p - N_n \tag{6.7}$$

Here ε refers to the permittivity of liquid. The whole system can be explained as a combination of hydrodynamics and electrostatics. We consider the effects of motion of charged species on the neutral fluid by adding the force on the charged species in the Navier-Stokes equations as a source term. The transport equations of the liquid can be written as:

$$\nabla . \vec{u} = 0 \tag{6.8}$$

$$\rho\left(\frac{\partial \vec{u}}{\partial t} + \nabla.(\vec{u}\,\vec{u})\right) = -\nabla\tilde{P} + \nabla.(\mu\,(\nabla\vec{u} + (\nabla\vec{u})^T) + \,(N_p - N_n)\vec{E} \tag{6.9}$$

The pressure term $\tilde{P}$ contains the contribution from electrostriction force and the liquid pressure. The dynamic viscosity of liquid is denoted by μ. The recombination rate constant is $k_R = (K_+ + K_-)/\varepsilon$, as given by Langevine's approximation for dielectric liquids [4, 11]. We work in the universal framework of non-dimensional equations by taking the reference variables as:

$$x, y \propto d\ ; \qquad t \propto \frac{d^2}{K_{0+}\,(V_0 - V_1)}\ ; \qquad \vec{u} \propto \vec{u}_{ref} = \frac{K_{0+}\,(V_0 - V_1)}{d}$$

$$\tilde{P} \propto \ \rho\, u_{ref}^2\ ; \qquad N_p, N_n \ \propto N_{eq}$$

$$K_+, K_- \ \propto \ K_{0+}\ ; \quad D_+, D_- \ \propto \ D_{0+}\ ; \quad \varepsilon \ \propto \varepsilon_0\ ; \quad \rho \ \propto \rho_0$$

$$\mu \ \propto \mu_0\ ; \qquad V \propto (V_0 - V_1) \quad ; \qquad \vec{E} \ \propto \frac{(V_0 - V_1)}{d}$$

Here K_{0+} and D_{0+} are the mobility and diffusion coefficients of the positive ions at equilibrium. $\rho_0, \mu_0, \varepsilon_0$ are the reference values of density, dynamic viscosity and permittivity for liquid respectively. The length reference is the distance between the electrodes (d), $V_0 - V_1$ is the potential difference between the two electrodes. These reference variables lead to the non-dimensional set of equations for the transport of species in which the non-dimensional values are represented with star (*):

$$\frac{\partial N_p^*}{\partial \mathrm{t}^*} + \nabla^*.\left(\left(\overrightarrow{u^*} + K_+^*\overrightarrow{E^*}\right)N_p^*\right) = \alpha\nabla^*.(D_+^*\,\nabla^* N_p^*) + \left(\frac{K_+^* + K_-^*}{\varepsilon^*}\right) C_0\ \left(1 - N_p^* N_n^*\right) \tag{6.10}$$

$$\frac{\partial N_n^*}{\partial \mathrm{t}^*} + \nabla^*.\left(\left(\overrightarrow{u^*} - K_-^*\overrightarrow{E^*}\right)N_n^*\right) = \alpha\nabla^*.(D_-^*\,\nabla^* N_n^*) + \left(\frac{K_+^* + K_-^*}{\varepsilon^*}\right)\ C_0\ \left(1 - N_p^* N_n^*\right) \tag{6.11}$$

$$\nabla^*.(\varepsilon^*\,\nabla^*\mathrm{V}^*) = \ -C_0\left(N_p^* - N_n^*\right) \tag{6.12}$$

$$\overrightarrow{E^*} = \ -\nabla^*\mathrm{V}^* \tag{6.13}$$

$$\nabla^*.\overrightarrow{u^*} = 0 \tag{6.14}$$

$$\frac{\partial \rho^*\overrightarrow{u^*}}{\partial \mathrm{t}^*} + \nabla^*.\left(\rho^*\overrightarrow{u^*}\,\overrightarrow{u^*}\right) = -\nabla^*\tilde{P}^* + \frac{1}{R_{el}}\nabla^*.\left(\mu^*(\nabla^*\overrightarrow{u^*} + \left(\nabla^*\overrightarrow{u^*}\right)^T)\right) + \ C_0\,M^2\left(N_p^* - N_n^*\right)\overrightarrow{E^*} \tag{6.15}$$

This set of non-dimensional equations introduces the following non-dimensional parameters which characterize our conduction problem:

$$R_{el} = \frac{\rho_0 K_{0+}\,(V_0 - V_1)}{\mu_0}\ ; \qquad M = \frac{1}{K_{0+}}\left(\frac{\varepsilon}{\rho_0}\right)^{\frac{1}{2}} \quad ; \quad C_0 \ = \ \frac{N_{eq} d^2}{\varepsilon_0 (V_0 - V_1)}\ ;$$

$$\alpha = \ \frac{D_{0+}}{K_{0+}(V_0 - V_1)}$$

C_0 is the ratio of the ionic transit time $\frac{d^2}{K_{0+}\,(V_0-V_1)}$ and the relaxation time ε_0/N_{eq}. M is the ratio between the hydrodynamic mobility and actual ionic mobility, and R_{el} is the electric Reynolds number. The non-dimensional diffusion coefficient is denoted by α. We write the final set of non-dimensional equations by removing the star

$$\frac{\partial N_p}{\partial t} + \nabla.\left(\left(\vec{u} + K_+\vec{E}\right)N_p\right) = \alpha\nabla.\left(D_+\nabla N_p\right) + \left(\frac{K_+ + K_-}{\varepsilon}\right)C_0\ \left(1 - N_pN_n\right) \tag{6.16}$$

$$\frac{\partial N_n}{\partial t} + \nabla.\left(\left(\vec{u} - K_-\vec{E}\right)N_n\right) = \alpha\nabla.\left(D_-\nabla N_n\right) + \left(\frac{K_+ + K_-}{\varepsilon}\right)C_0\ \left(1 - N_pN_n\right) \tag{6.17}$$

$$\nabla.\left(\varepsilon\,\nabla V\right) = -C_0\ \left(N_p - N_n\right) \tag{6.18}$$

$$\vec{E} = -\nabla V \tag{6.19}$$

$$\nabla.\vec{u} = 0 \tag{6.20}$$

$$\frac{\partial \rho\vec{u}}{\partial t} + \nabla.\left(\rho\vec{u}\,\vec{u}\right) = -\nabla\tilde{P} + \frac{1}{R_{el}}\nabla.\left(\mu(\nabla\vec{u} + (\nabla\vec{u})^T)\right) + C_0\,M^2(N_p - N_n)\vec{E} \tag{6.21}$$

6.2.1 The Onsager effect

We explain EHD conduction in dielectric liquids based on the heterocharge layers of species generated due to the dissociation and recombination of ions under the influence of external electric field. Several mechanisms of charge generation are explained with different mathematical models in literature [1,7,9]. Ionogenic impurities (ion-pairs with zero net-charge) also induce conductivity by dissociation, producing oppositely charged free ion species [7].

In absence of external electric field, the rates of dissociation (k_D) and recombination (k_R) are considered constant, as provided in last section. Onsager proposed his model based on the concept of enhancing the dissociation process by increasing the external electric field [7,9]. An important length scale in this model, the Onsager distance ($l_O = \sqrt{e/4\pi\varepsilon E}$), is described as the distance in non-polar liquids where the force due to the external electric field is balanced by the Coulombic force of the ion pair. Thus, the Onsager distance decreases with the increase in external electric field and it makes the separation of ion-pairs (dissociation) easier. The proposed formula for dissociation states:

$$k_D(E) = k_{D0}\,F(b) \tag{6.22}$$

With $b^2 = (l_B/l_O)^2 = 4\gamma E$, $\gamma = e^3/(16\pi\varepsilon k_B^2T^2)$ and $F(b) = I_1(2b)/b$. Here k_D is the dissociation constant and k_{D0} refers to its value at thermodynamic equilibrium in absence of electric field. Also, $l_B = e^2/4\pi\varepsilon k_BT$ is the Bjerrum distance [7], k_B is the Boltzmann constant, T is the temperature of the species and e is the elementary electric charge. The function I_1 is the modified Bessel function of first kind and order one. More details on this model can be obtained from references [7,9,10]. This model is implemented in Oracle3D which updates the

dissociation process in the transport equations of the species. By definition, this model will enhance the dissociation and there will be an increase in the ion concentrations.

Let's consider the transport equation of positive ions as given by eq. (6.1). We replace the dissociation constant in eq. (6.1) by the dissociation constant given with eq. (6.22), and rewrite the equation:

$$\frac{\partial N_p}{\partial t} + \nabla.\vec{J}_+ = k_{D0}\, F(b)\, c - k_R N_p N_n \tag{6.23}$$

Replacing the value of dissociation constant with the recombination constant, as described at thermodynamic equilibrium, we get:

$$\frac{\partial N_p}{\partial t} + \nabla.\vec{J}_+ = k_R\, (N_{eq}^2\, F(b) - N_p N_n) \tag{6.24}$$

We take the same scaling parameters as shown above to get the corresponding non-dimensional equations for eq. (6.24). The function $F(b)$ is already a non-dimensional number as explained above. The non-dimensional equation for the transport of positive ions after considering the Onsager effect will be given by:

$$\frac{\partial N_p}{\partial t} + \nabla.\left(\left(\vec{u} + K_+\vec{E}\right)N_p\right) = \alpha\nabla.(D_+\nabla N_p) + \left(\frac{K_+ + K_-}{\varepsilon}\right) C_0\, \left(F(b) - N_p N_n\right) \tag{6.25}$$

Similarly, we can write the transport equation for the negative ions also, which is:

$$\frac{\partial N_n}{\partial t} + \nabla.\left(\left(\vec{u} - K_-\vec{E}\right)N_n\right) = \alpha\nabla.(D_-\nabla N_n) + \left(\frac{K_+ + K_-}{\varepsilon}\right) C_0\, \left(F(b) - N_p N_n\right) \tag{6.26}$$

This field enhanced dissociation model for neutral species as given by Onsager was implemented in Oracle3D. The impact of this model on the species' density and flow parameters is presented in section 6.4.2 of this thesis, by comparing some variables computed with and without this model.

6.2.2 The electric current (I)

Electric current is one of the most important parameters while discussing the electro-conduction phenomenon. In the following sections, we have plotted the electric current several times to compare and analyze the results obtained with our simulations. The electric current manifests mainly due to the combined flux of the charge species' densities through the electrode surfaces. We rewrite here the equation of current density fluxes (eq. 6.4-6.5) due to both positive and negative ions, and the overall current density flux ($\vec{J}$) is given by eq. (6.27).

$$\vec{J}_+ = \vec{u}N_p + K_+ N_p\vec{E} - D_+\nabla N_p \tag{6.4}$$

$$\vec{J}_- = \vec{u}N_n - K_- N_n\vec{E} - D_-\nabla N_n \tag{6.5}$$

$$\vec{J} = \vec{J}_d + \vec{J}_+ - \vec{J}_- \tag{6.27}$$

$$\vec{J} = \frac{\partial \varepsilon_r \varepsilon_0 \vec{E}}{\partial t} + \left(K_+ N_p + K_- N_n\right)\vec{E} + \left(N_p - N_n\right)\overrightarrow{u} - \left(D_+ \nabla N_p - D_- \nabla N_n\right) \qquad (6.28)$$

In eq. 6.27, we also include the factor of displacement current ($\vec{J}_d$), which arises due to the time-varying electric field in the system. Eq. 6.28 is the dimensional expression of combined net current density flux through any control volume of the system. In our cases, we have mainly computed the non-dimensional electric current on the high-voltage electrode. The non-dimensionalization of eq. 6.28 can be carried out with the previously mentioned reference parameters for the respective variables. Eq. 6.29 is obtained after using the reference parameters for all the variables in eq. 6.28.

$$\begin{aligned}\vec{J} = {} & K_{0+}\varepsilon_r\varepsilon_0 \frac{(V_0 - V_1)^2}{d^3}\frac{\partial \overrightarrow{E^*}}{\partial t^*} + K_{0+}N_{eq}\frac{(V_0 - V_1)}{d}\left(K_+^* N_p^* + K_-^* N_n^*\right)\overrightarrow{E^*} \\ & + K_{0+} N_{eq}\frac{(V_0 - V_1)}{d}\left(N_p^* - N_n^*\right)\overrightarrow{u^*} \\ & - \frac{N_{eq} D_{0+}}{d}\left(D_+^* \nabla^* N_p^* - D_-^* \nabla^* N_n^*\right)\end{aligned} \qquad (6.29)$$

$$\begin{aligned}\frac{\vec{J}\, d}{K_{0+}\, N_{eq}\, (V_0 - V_1)} = {} & \varepsilon_r\varepsilon_0 \frac{(V_0 - V_1)}{N_{eq}\, d^2}\frac{\partial \overrightarrow{E^*}}{\partial t^*} + \left(K_+^* N_p^* + K_-^* N_n^*\right)\overrightarrow{E^*} + \left(N_p^* - N_n^*\right)\overrightarrow{u^*} \\ & - \frac{D_{0+}}{K_{0+}\,(V_0 - V_1)}\left(D_+^* \nabla^* N_p^* - D_-^* \nabla^* N_n^*\right)\end{aligned} \qquad (6.30)$$

$$\begin{aligned}\overrightarrow{J^*} = {} & \frac{\varepsilon_r}{C_0}\frac{\partial \overrightarrow{E^*}}{\partial t^*} + \left(K_+^* N_p^* + K_-^* N_n^*\right)\overrightarrow{E^*} + \left(N_p^* - N_n^*\right)\overrightarrow{u^*} \\ & - \alpha\left(D_+^* \nabla^* N_p^* - D_-^* \nabla^* N_n^*\right)\end{aligned} \qquad (6.31)$$

Eq. 6.31 represents the final non-dimensional expression for the net current density flux ($\overrightarrow{J^*}$). We rewrite this expression without the * sign to simplify the notations. The non-dimensional numbers C_0 and α are the same as mentioned previously.

$$\vec{J} = \frac{\varepsilon_r}{C_0}\frac{\partial \vec{E}}{\partial t} + \left(K_+ N_p + K_- N_n\right)\vec{E} + \left(N_p - N_n\right)\overrightarrow{u} - \alpha\left(D_+ \nabla N_p - D_- \nabla N_n\right) \qquad (6.32)$$

$$C_0 = \frac{N_{eq} d^2}{\varepsilon_0 (V_0 - V_1)} \;;\; \alpha = \frac{D_{0+}}{K_{0+}(V_0 - V_1)}$$

Now, the electric current (I_{total}) is computed on the high voltage electrode, which is obtained by the integration of the net current density flux over the surface of the electrode as:

$$I_{total} = \int_s \vec{J}.\vec{n}\, dS \qquad (6.33)$$

The equation for current (eq. 6.33) is solved with Finite volume approach by adding the contribution of the current through all the boundary nodes of the high voltage electrode surface. The displacement current is negligible in such electro-conduction situations in comparison with the current density fluxes, and thus it is not included in our final computation of electric current. All the current plots shown in following sections of this chapter are computed as described here.

In this thesis, we carried out most of our studies with a rectangular channel as shown by the sketch in Fig. 6.2.1. All the relevant boundary conditions (BC) for all the variables are provided in this figure. East and west faces of the channel are treated as periodic for all variables. Electrode configuration is asymmetric where the high voltage electrodes is 3 times smaller than the grounded electrode in our test cases. N_p and N_n represent the number densities of positive and negative species respectively. Symmetry boundary on the top face refers to zero gradient for all the variables. For the substrate, we have the options of zero gradient Neumann BC and non-homogeneous Neumann ($\nabla V.\vec{n} = \sigma$) for electric potential, here 'σ' refers to the accumulated charge on the substrate. For the charge species, we have zero gradient Neumann BC and Robin BC to account for the charge accumulation on the substrate.

Channel bottom represents no-slip conditions for velocities. In Fig. 6.2.1, the high voltage electrode is shown in red, which is defined by Dirichlet BC (V=1) for electric potential. The grounded electrode is shown in blue which is also set to Dirichlet BC (V=0) for electric potential. Boundary conditions for charge species on the electrodes are set as shown in Fig. 6.2.1. In all the studies in this thesis, we assumed that the mobility coefficients of both ion species are equal to 1 which states $K_+ = K_- = 1$. We have also taken constant values of liquid properties like permittivity, density and dynamic viscosity for our studies.

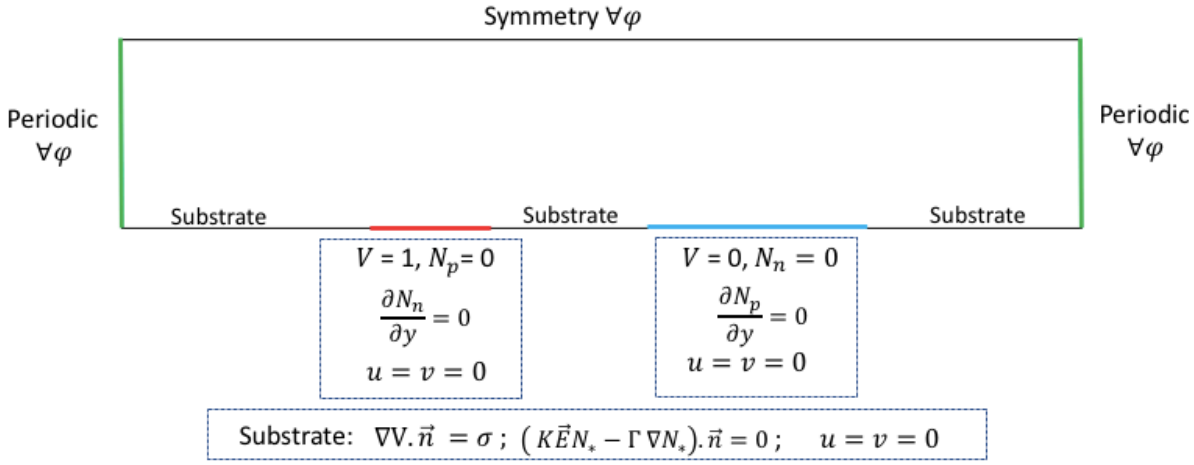


Fig. 6.2.1 Overall boundary conditions for the channel case with conduction model

6.3 Validation and flow analysis studies

This section presents the very initial case studies which were performed to validate the EHD conduction model as implemented in Oracle3D. Effects of boundary conditions such as Neumann BC, Robin BC, non-homogeneous Neumann and Periodic BC are studied, and results were compared with available COMSOL results. The flow with electro-conduction was simulated in a rectangular channel with non-dimensional domain as shown in Fig. 6.3.1. The high voltage electrode is 3 times smaller than the grounded electrode making it an asymmetrical configuration of electrodes. Fig. 6.3.2 shows the location of sections on which we shall plot variables for comparison. A grid size of 400 X 200 was taken for the channel and the model non-dimensional numbers were taken as:

$C_0 = 1.25599$
$M = 3.74195$
$\alpha = 5.0503 \times 10^{-4}$
$R_{el} = 7.752$

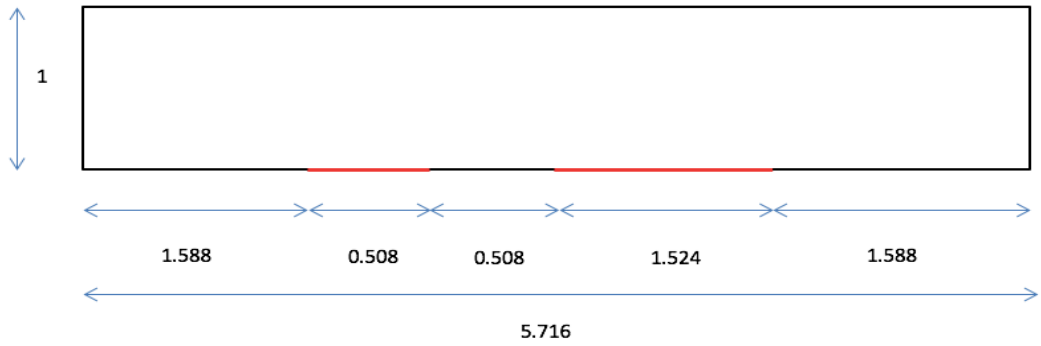


Fig. 6.3.1 Channel configuration with non-dimensional lengths

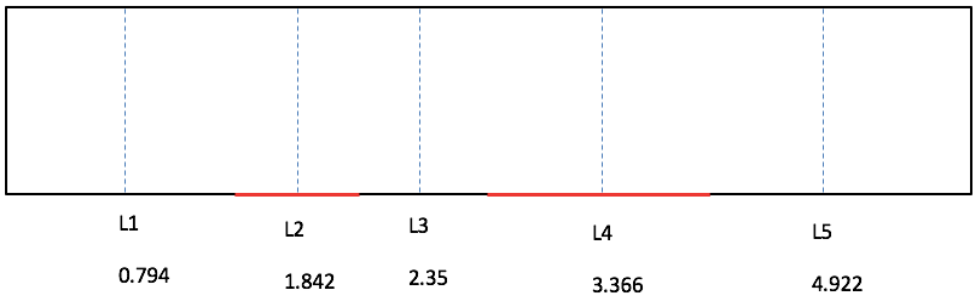


Fig. 6.3.2 Section locations for plotting profiles

We subdivide this section in several sub-sections for clarity. In section 6.3.1 we present validation cases with COMSOL considering 3 boundary conditions combinations, the corresponding flow structure with these cases are also explained after the validation results. Section 6.3.2 explains the impact of Onsager effect with relevant variables. The impact and the underlying physics behind the Robin BC on the substrate are presented in section 6.3.3.

Summary of this section:

6.3.1 Validation cases

Several combinations of boundary conditions were simulated with Oracle3D while developing the new features for conduction model. Here we present three cases which were compared with COMSOL results for validation. Corresponding flow structure with cases I and II are also presented within these sections.

I. Neumann BC substrate without Onsager effect

We have Periodic boundary condition for the east and west boundary faces for all variables in this case. Boundary conditions for electrodes are same for all the cases as mentioned in Fig. 6.2.1. Substrate was set to no-slip boundary for velocities, and zero gradient Neumann for potential and charge species. The Onsager effect was not considered in this case. Plotting contours of various variables provide important insight into the effects of boundary conditions used. Fig. 6.3.3 depicts the influence of periodic BC at west and east face on electric potential. Fig. 6.3.4-6.3.5 show the electric field components. We observe that the high magnitudes of electric field components correspond to having peaks at the edges of electrodes, which is expected due to sudden jump in potential between the electrodes and the substrate, at the edges.

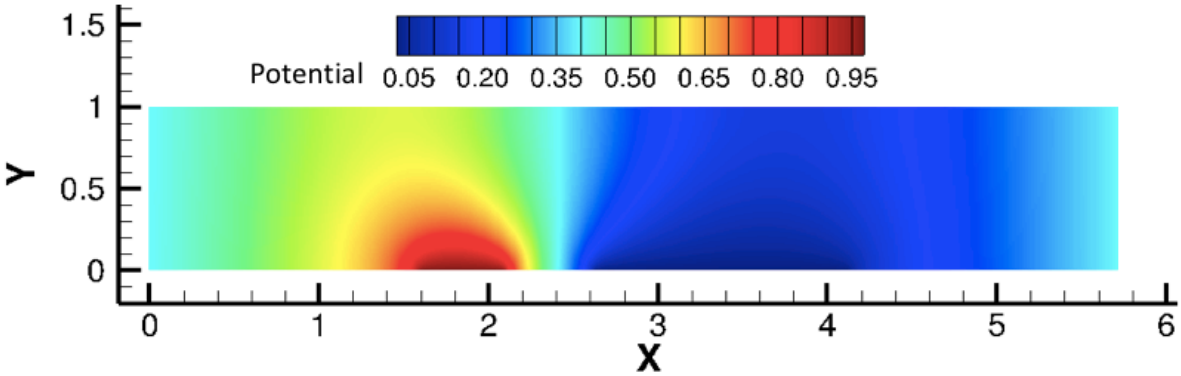


Fig. 6.3.3 Electric Potential contours with periodic BC at east and west faces

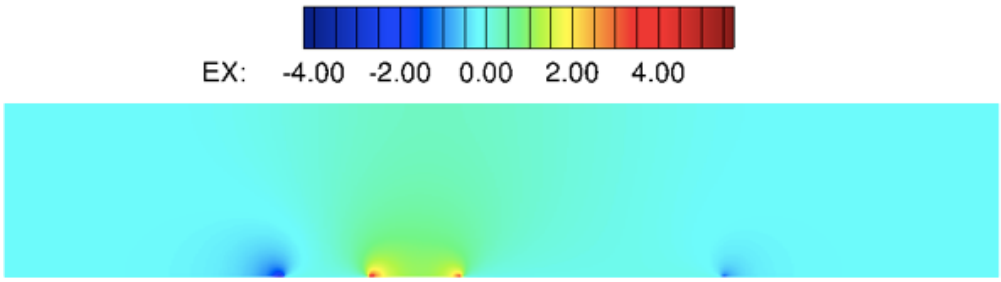


Fig. 6.3.4 X-component of Electric Field vector

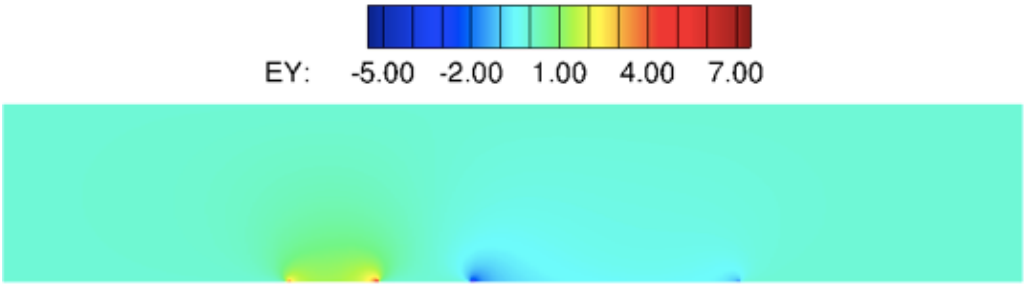


Fig. 6.3.5 Y-component of Electric Field vector

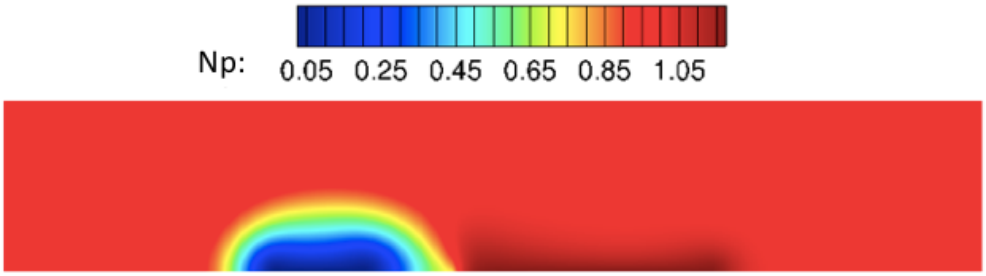


Fig. 6.3.6 Positive charge density contours

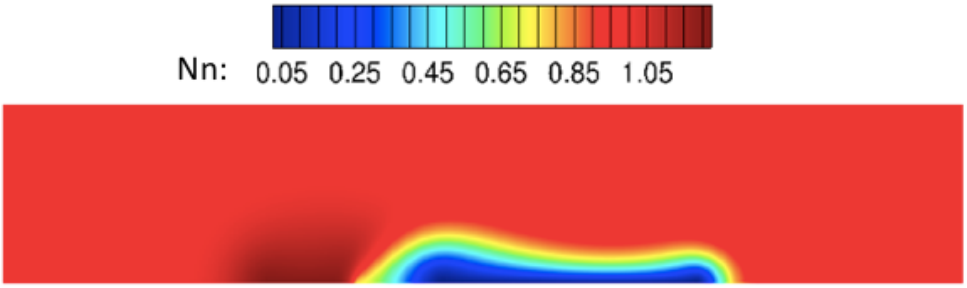


Fig. 6.3.7 Negative charge density contours

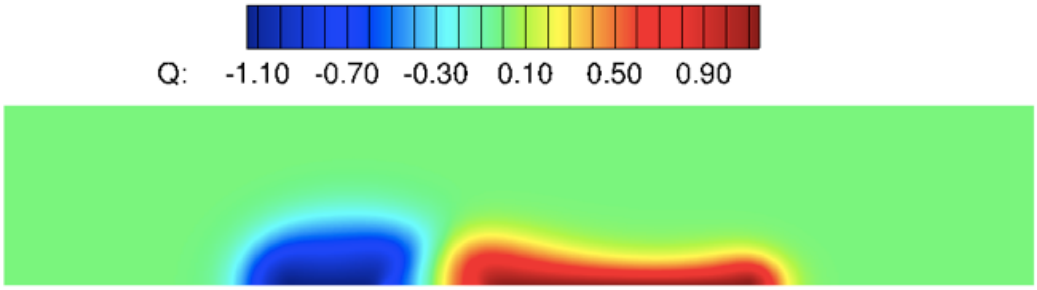


Fig. 6.3.8 Net charge density contours

Contours of charge densities of species present their distribution in the domain under the influence of electric field and according to the polarity of electrodes, Fig. 6.3.6-6.3.7. We observe that due to the electric field heterocharge layers are formed over both the electrodes, which show that ions of opposite polarity are attracted by both the electrodes. The proximity of the two electrodes leads to considerably high electric field intensity in the inter-electrode region, as a consequence of that thickening of the heterocharge layers are observed in the inter-electrode region. Numerically, this behavior can be explained with the C_0 parameter which incorporates the impact of voltage difference (V_0-V_1), distance (d) and the equilibrium charge density(N_{eq}).

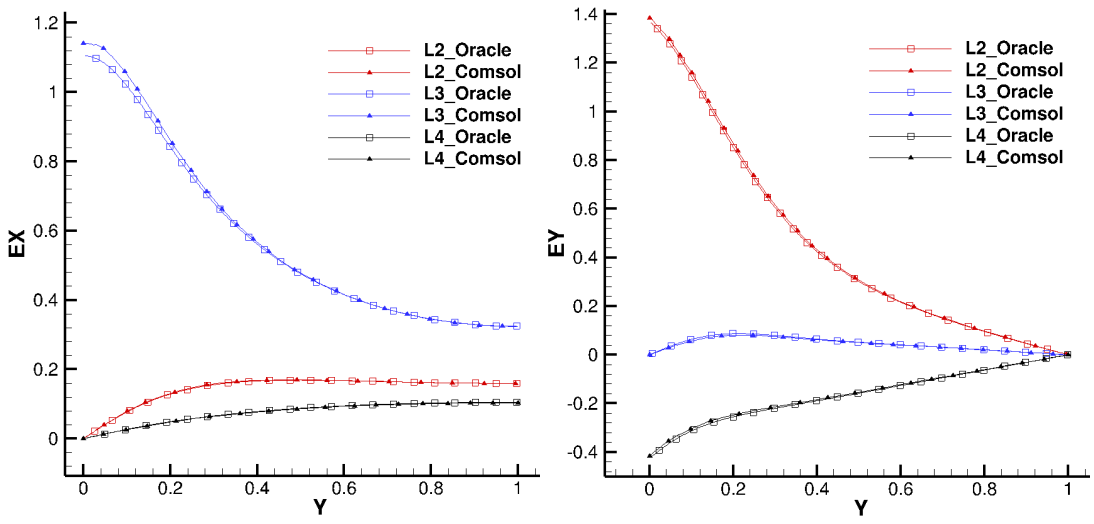


Fig. 6.3.9 Electric field components at X = 1.842, 2.35 and 3.366

Fig. 6.3.8 shows the net charge ($N_p - N_n$) density in the domain. We plot the profiles of electric field components and charge species on three section L2, L3, L4, as shown in Fig. 6.3.2, and compare our results with COMSOL results. Fig. 6.3.9 and 6.3.10 show that we have a very good match with the COMSOL results on these profiles. Results were also compared on sections L1 and L5, and a good match was observed on those sections too.

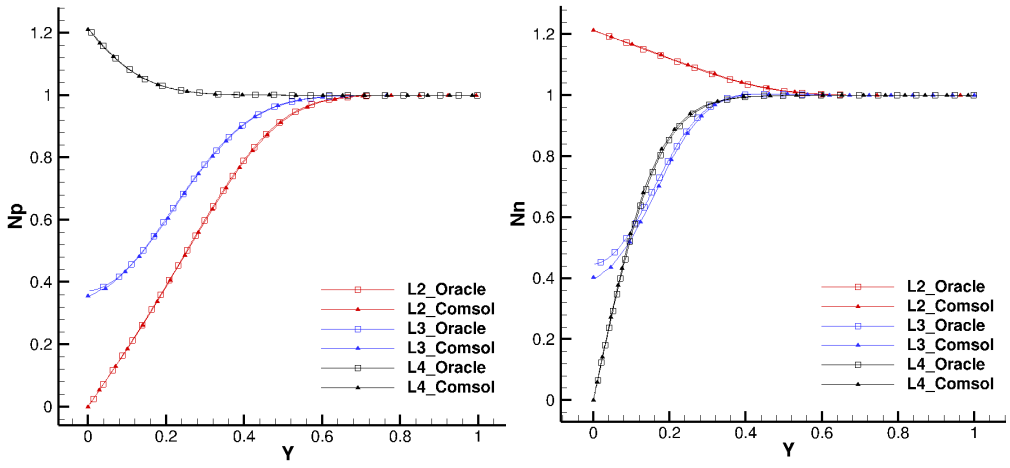


Fig. 6.3.10 Np and Nn at X = 1.842, 2.35 and 3.366

Fig. 6.3.11(a) shows the electric current evolution as computed on the HV electrode with Oracle3D, and the evolution of the integration of force vectors in whole domain. The electric current was computed as explained in section 6.2.2. Fig. 6.3.11 (a) represents the total current computed on the HV electrode by the fluxes of both positive and negative ion densities combined as given by eq. 6.33. The force components represent the integration of force vectors on all the control volumes of the domain. It should be noted that the results until Fig. 6.3.11 were computed without the Navier-Stokes computations, only the transport of species was simulated under the electric field influence to verify our implementations of the electrostatic part of the solver.

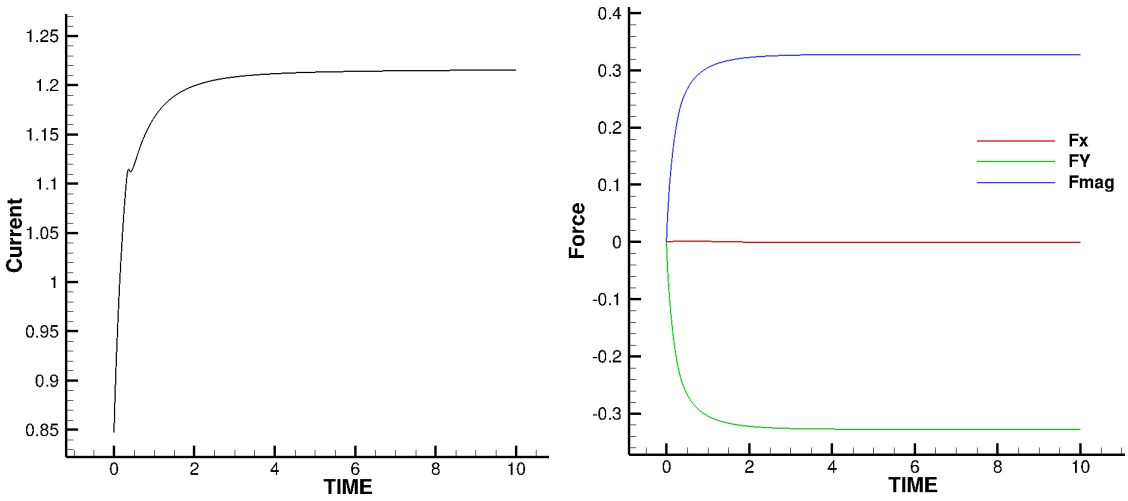


Fig. 6.3.11 a) electric current evolution on HV electrode, and b) Force vector evolution

Flow analysis with case I

After validating the species transport results with COMSOL, we simulated the species transport with the neutral fluid in channel. Fig. 6.3.12-6.3.13 show the steady state results with the flow simulations accounting for Navier-Stokes equations coupled with species transport. We observe

in the streamlines plot that the fluid is pulled within the inter-electrode region in the negative y direction. As explained above, the inter-electrode region has higher electric field intensity and the heterocharge layers are also thicker near the edges of the electrodes leading to net charge density in this region. This produces a larger electric body force in this region which is directed towards both the electrode surfaces.

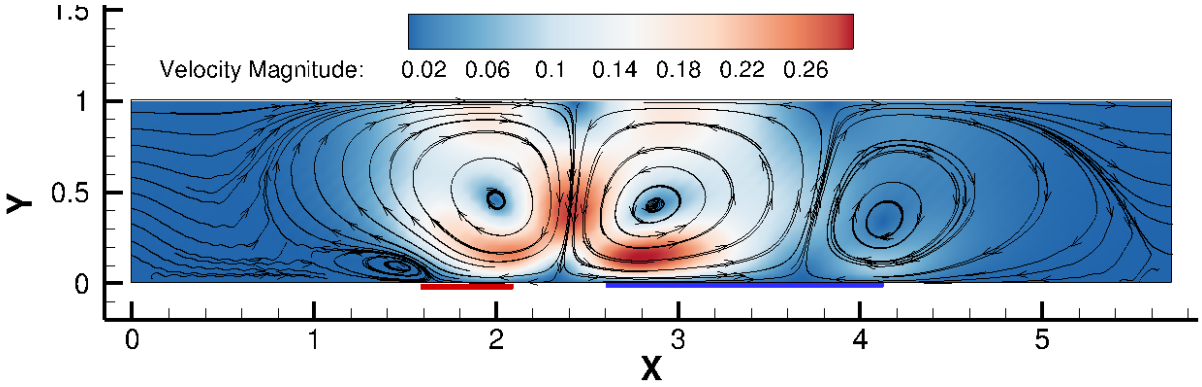


Fig. 6.3.12 Streamlines with non-dimensional velocity contours in conduction channel at steady state

Two counter rotating vortices are formed over the electrodes, due to the locally high electric body forces in the inter-electrode region. This shows that the generation of electric body force is mainly confined to this inter-electrode region, and thus it is non-uniform in the domain. The asymmetric electrode configurations are important in generating a net charge density and consequently a net electric body force, which sets the whole fluid in channel into motion [3,8]. We observe the overall movement of the fluid in channel is in the direction from shorter electrode (HV electrode) to the longer electrode (grounded electrode), making the flow going out from east face and due to periodic BC entering again on the west face of channel (Fig. 6.312). Yazdani et al. (2009) reported that in EHD conduction channel the flow is always generated from the shorter electrode to the longer electrode independent of their polarity.

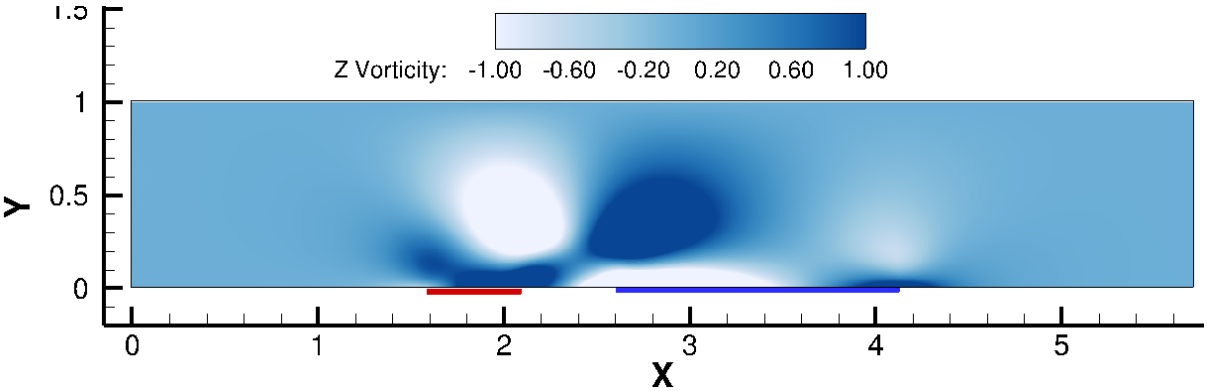


Fig. 6.3.13 Z-vorticity contours in conduction channel at steady state

Similar flow pattern was observed with COMSOL simulations also. Fig. 6.3.13 shows the z-vorticity contours complementing the streamlines pattern shown in Fig. 6.3.12. The counterclockwise rotating vortex over the grounded electrode is represented with positive value of z-vorticity in Fig. 6.3.13 with dark blue patch. The evolution of net mass flow rate obtained with this case is shown in Fig. 6.3.14, which shows the initially stationary flow has reached a steady mass flow rate after nearly 10 non-dimensional time units. This confirms the pumping effect induced due to the electro-conduction mechanism in dielectric liquids. Micro-scale channels with asymmetric electrode designs have been used in the electrokinetics domain to pump electrolyte liquids utilizing this concept of electro-conduction pumping [3].

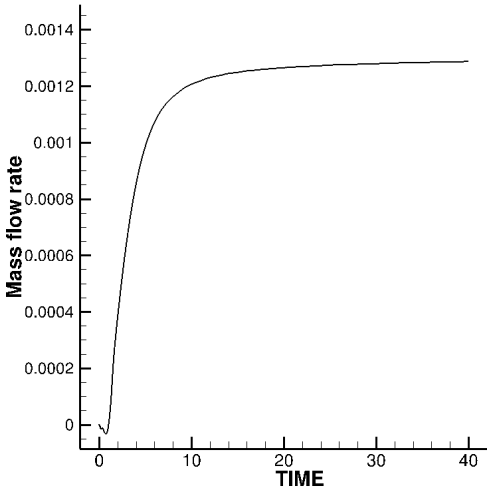


Fig. 6.3.14 Evolution of the non-dimensional net mass flow rate

II. Robin BC substrate with Onsager effect

For this validation case, we incorporated the Robin boundary conditions for the species on the substrate and non-homogeneous Neumann for the electric potential as mentioned in Fig. 6.2.1. In this section, we took this case for the validation purposes only, a brief discussion on Robin BC follows in section 6.3.3 of this thesis. Mainly, the Robin BC accounts for the accumulation of charge on the substrate. We also considered the Onsager effect for this case. Onsager model mainly provides a formulation for the enhancement of dissociation of species with increasing electric field [7,9]. We simulated the transport of species with Navier-Stokes equations and compared the results with COMSOL solution for this case.

The comparison of force components evolution is shown in Fig. 6.3.15. We observe that the magnitude of y-component of force is higher in than the x-component, these force components represent the integrated value over whole the domain. We notice a magnitude difference of ~ 0.002 in x-component against the value obtained with COMSOL, which is significantly lower in comparison with the dominant y-component. On the other hand, the y-component shows a very good agreement with COMSOL results. Thus, the overall force magnitude showed a good agreement in our validation tests. The evolution of electric current, which is an important indicator of species distribution, shows a very good agreement with COMSOL current

evolution plot (Fig. 6.3.16). As mentioned above, this current represents the total current computed on the HV electrode by the fluxes of both positive and negative ion densities combined.

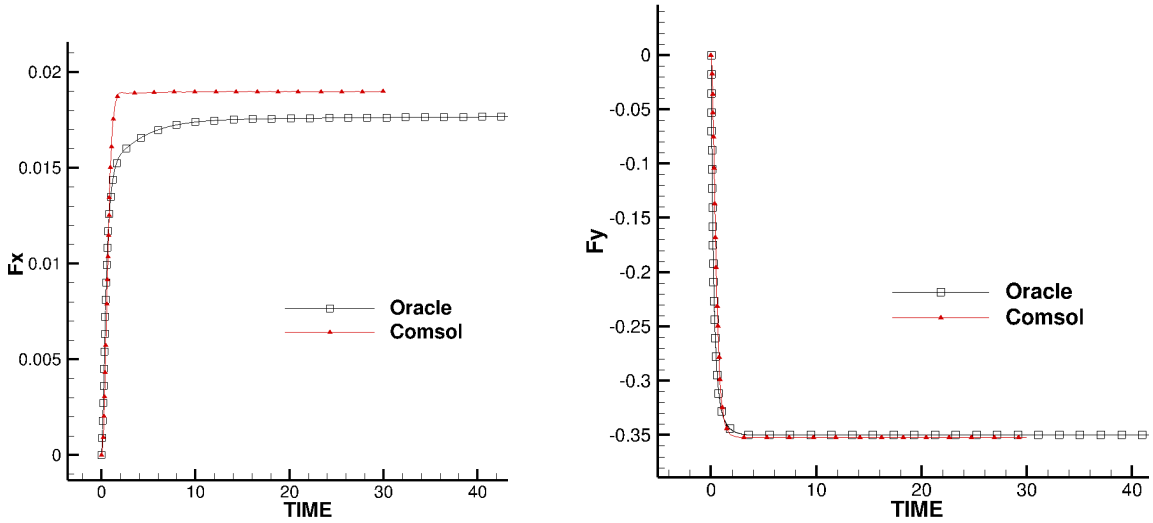


Fig. 6.3.15 Evolution of x and y components of force with Oracle and COMSOL

As we also incorporated Robin BC condition for the charge species on the substrate, we compared electric field components and species densities on the bottom surface of channel (Y=0). Electric field components show peaks at the electrode edges as expected. We observe that except the peak values at edges we match very well over the entire surface of electrodes and substrate (Fig. 6.3.17). Positive peaks of Ex also match perfectly but other peaks have discrepancies with COMSOL results. During our comparisons we observed that the order of accuracy of approximations used to compute the electric field on the boundaries have significant impact on the peak values, and this factor led to this mismatch in results here.

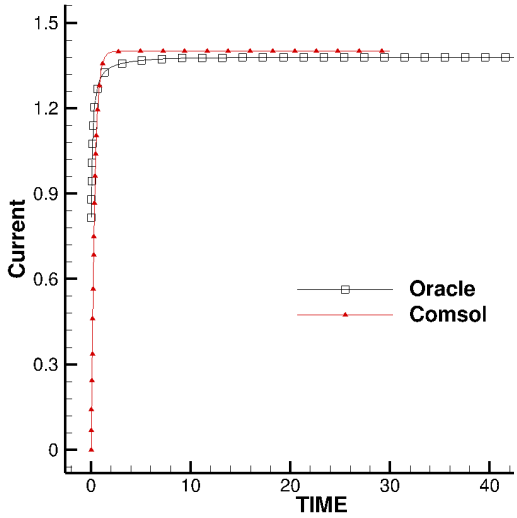


Fig. 6.3.16 Evolution of electric current on HV electrode with Oracle and COMSOL

In Oracle3D, we used 2nd order interpolation for computing electric field on surface, however, it could not be confirmed which method is used by COMSOL. Similarly, as the Robin BC ($\left(K\vec{E}\varphi - \Gamma \nabla\varphi \right).\vec{n} = 0$) accounts for the electric field on the surface, we see that there is a mismatch in the peak values of species densities also, Fig. 6.3.18, however on rest of the channel bottom surface the species density shows a very good match. We also note that COMSOL uses the Finite Element discretization as compared to the Finite Volume approach used in Oracle3D. Additionally, in section 6.4.1 we will show that the differences in the peak values have no significant impact on the overall flow variables.

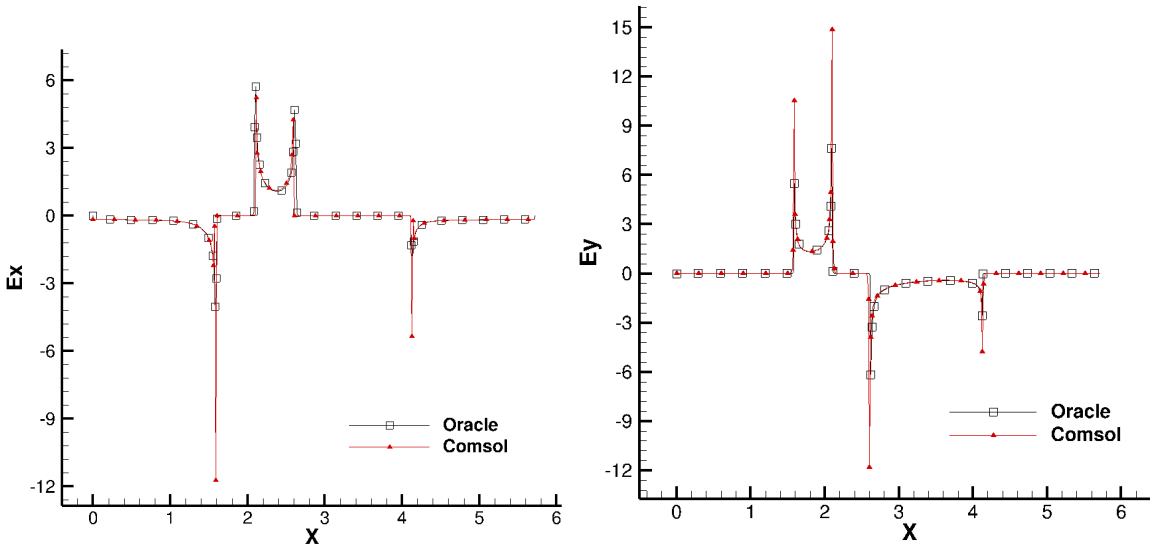


Fig. 6.3.17 Electric field components Ex and Ey on the channel bottom surface (Y=0)

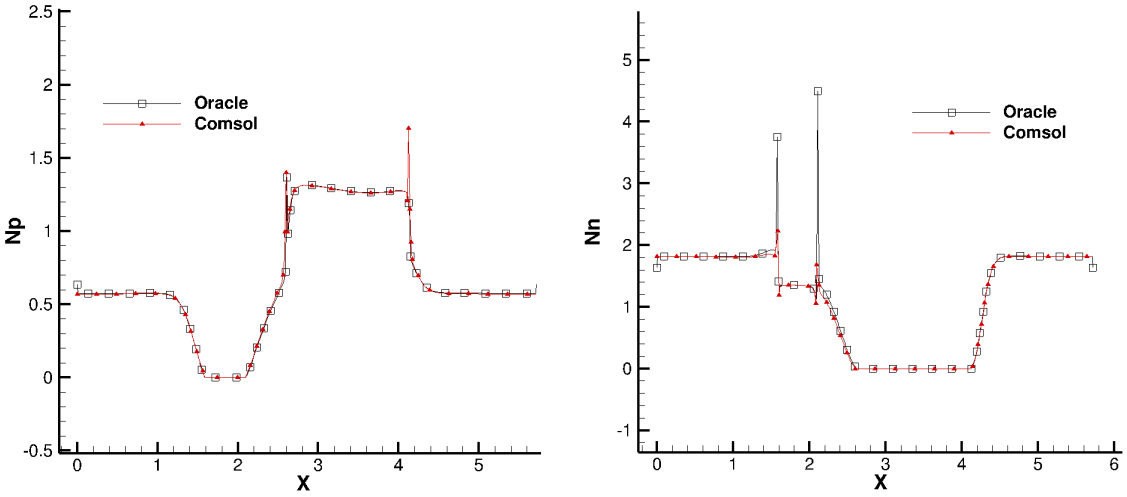


Fig. 6.3.18 Positive (Np) and Negative (Nn) species density on the channel bottom surface (Y=0)

Flow analysis with case II

We plot the streamlines of flow with the velocity contours in Fig. 6.3.19. The explanation of flow features as discussed in last section applies here also. In this case, we observe that the net flow is in negative x direction. This flow is from the longer to the shorter electrode, opposite to what we observed in the previous section. The flow clearly shows periodic behavior at east and

west boundaries. During our tests we observed that the flow direction remained same for the zero gradient Neumann and Robin boundary conditions for species, having Neumann also for the electric potential on substrate. However, we found that with the implementation of non-homogenous Neumann BC ($\nabla V.\vec{n} = \sigma$) for electric potential and Robin BC for species, the flow changed the direction. Both of these boundary conditions account for the accumulation of charge densities on the substrate which is not considered in the zero gradient Neumann BC.

The value of non-dimensional surface charge 'σ' was taken as 2.115×10^{-2} for this case, which was obtained with some previous experimental measurements. This surface charge is defined as $\sigma = \zeta d\ /(\Delta V\ \lambda)$. Where zeta potential (ζ) is an electric characteristic of the liquid-solid couple, which has a typical value of 15 mV in such cases. The σ charge develops at the surface of the solid substrate which is in contact with the liquid dielectric. ΔV is the voltage difference, d is the channel width and λ is the Debye length.

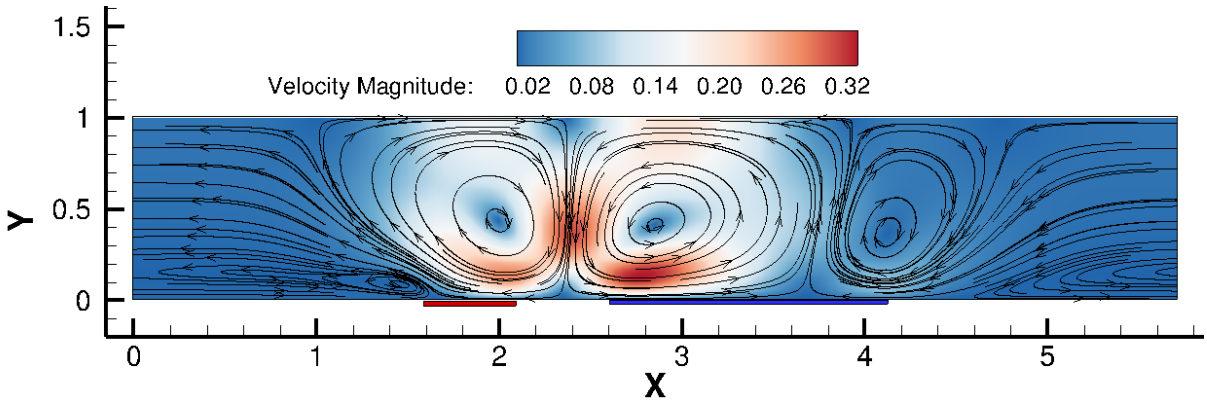


Fig. 6.3.19 Streamlines with non-dimensional velocity contours in conduction channel at steady state

Accounting for the Onsager effect or not, which mainly affected the charge densities, did not change the underlying flow direction. In reality, there are many parameters which affect the overall phenomenon in EHD conduction: liquid properties, electrical inputs, electrode configuration, impurities, temperature etc [3-5,9]. Numerically, we see that the boundary conditions and overall EHD models have significant impacts on the electrostatic and flow features. More studies with same and different boundary conditions will have to be undertaken to generalise the explanation of the overall phenomenon.

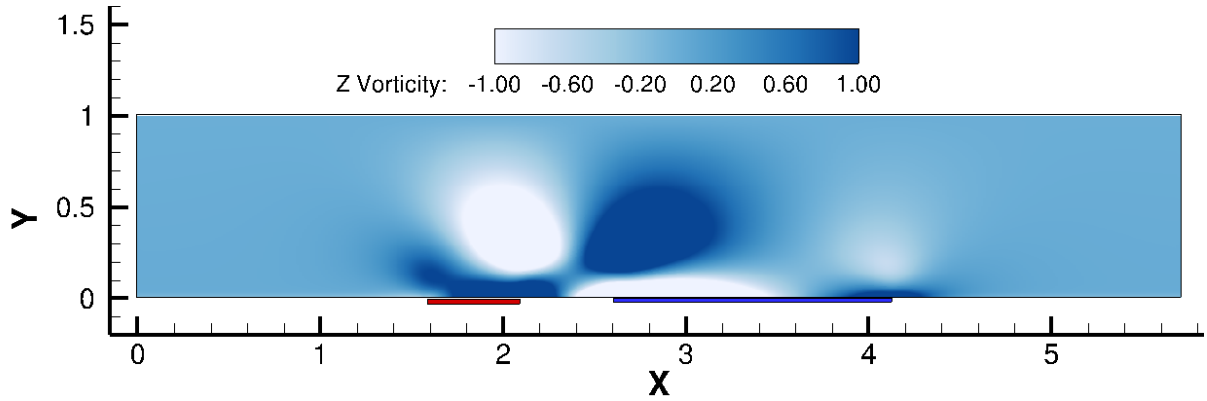


Fig. 6.3.20 Z-vorticity contours at steady state

III. Without Periodic BC on east and west faces

We have used zero gradient Neumann boundary condition for the east and west boundary faces for all variables in this case. Boundary conditions for electrodes are same as previous cases, Fig. 6.2.1. Substrate was set to no-slip boundary for velocities, and zero gradient Neumann for potential and charge species. This case shows us the effect of not having periodic BC at the east and west faces, where zero gradients are used for potential and species. This was also a test to validate the implementation of conduction model without the periodic BC. Fig. 6.3.21 should be viewed in comparison with Fig. 6.3.3 to appreciate the use of periodic BC. We show the comparison of electric field components and species density on the sections L2, L3, L4 in Fig. 6.3.22-6.3.23. We observe a very good agreement with COMSOL results for these variables.

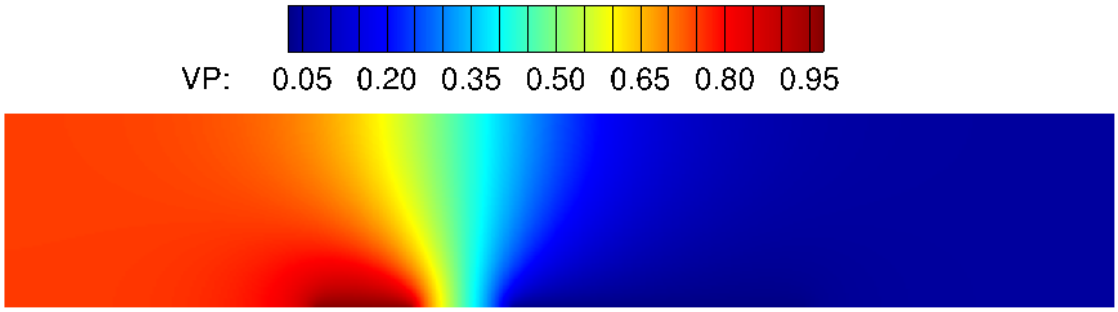


Fig. 6.3.21 Electric Potential contours without periodic BC at east and west faces

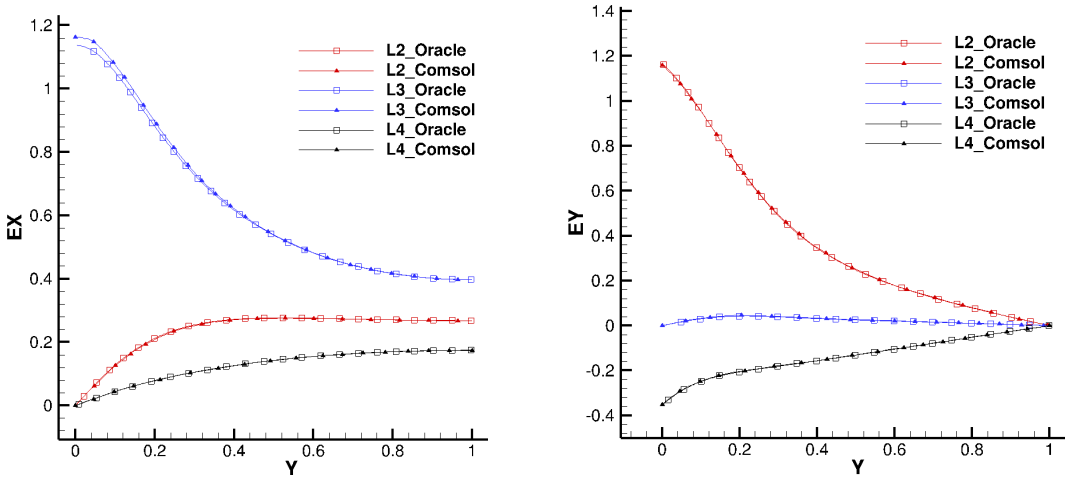


Fig. 6.3.22 Electric field components at X = 1.842, 2.35 and 3.366

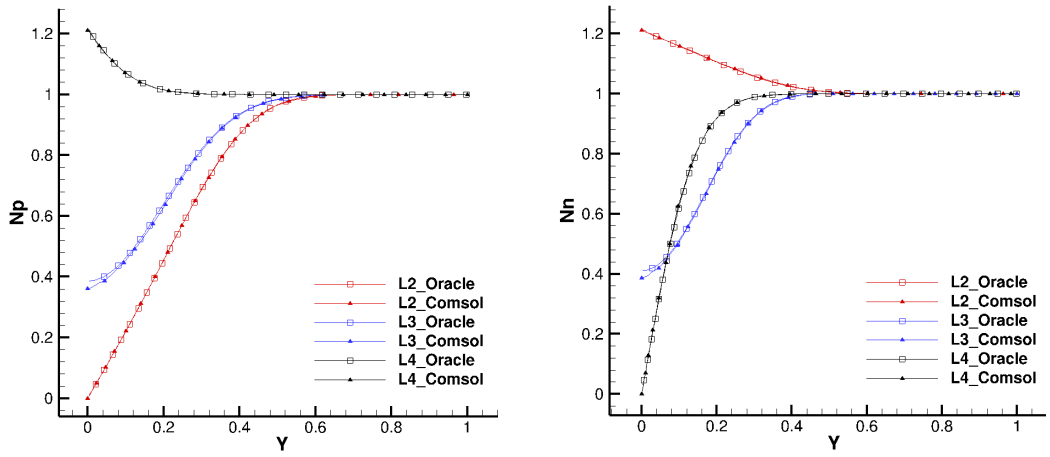


Fig. 6.3.23 Np and Nn at X = 1.842, 2.35 and 3.366

We also simulated the same case with Robin BC for the species at the substrate and compared results with COMSOL. The x-component of force showed minor discrepancy in comparison with COMSOL results, however, the dominant component (F_y) showed a good agreement with COMSOL values (Fig. 6.3.24). The electric current evolution on HV electrode also showed a good agreement with COMSOL (6.3.25 (b)). The force components and the electric current are computed as mentioned in the previous cases.

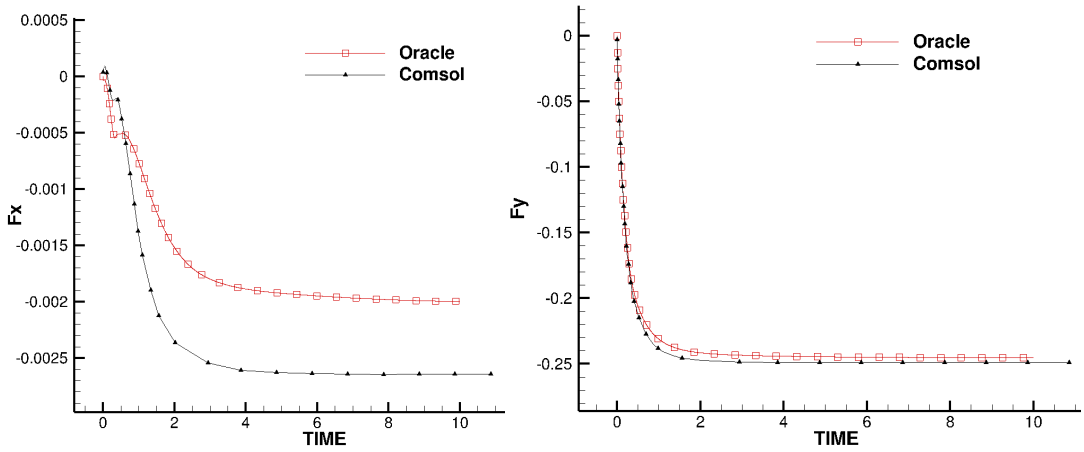


Fig. 6.3.24 Evolution of x and y components of force with Neumann BC at east and west faces

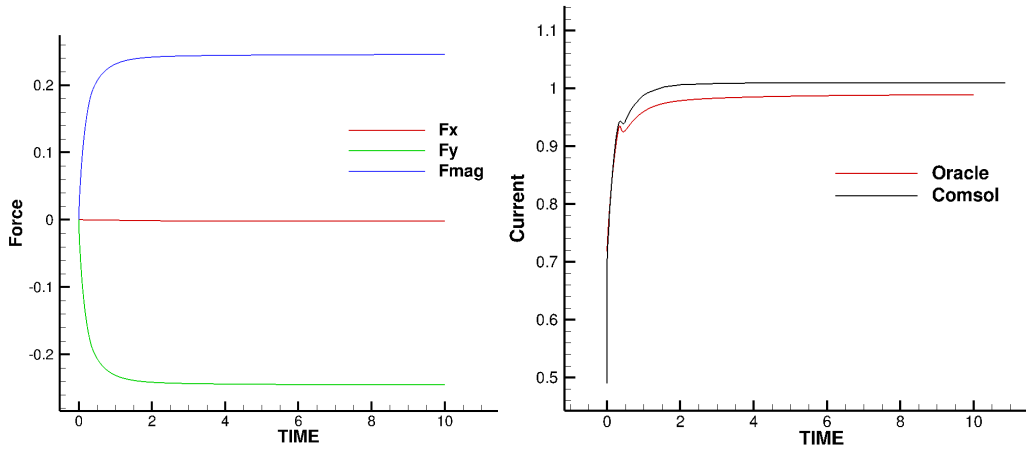


Fig. 6.3.25 a) Evolution of force components and force magnitude, b) evolution of electric current

As we used Robin BC on the substrate, we plotted the Ex, Ey, Np, Nn also on the channel bottom surface. In Figs. 6.3.26-6.3.27 we observe a very good match of quantities on the entire bottom surface of channel except some values of peaks at the electrode edges. As explained above, the differences with the peak values are mainly due to the methods used by two solvers to approximate the values on the boundary surfaces. We can say that overall our results with COMSOL are in good agreement.

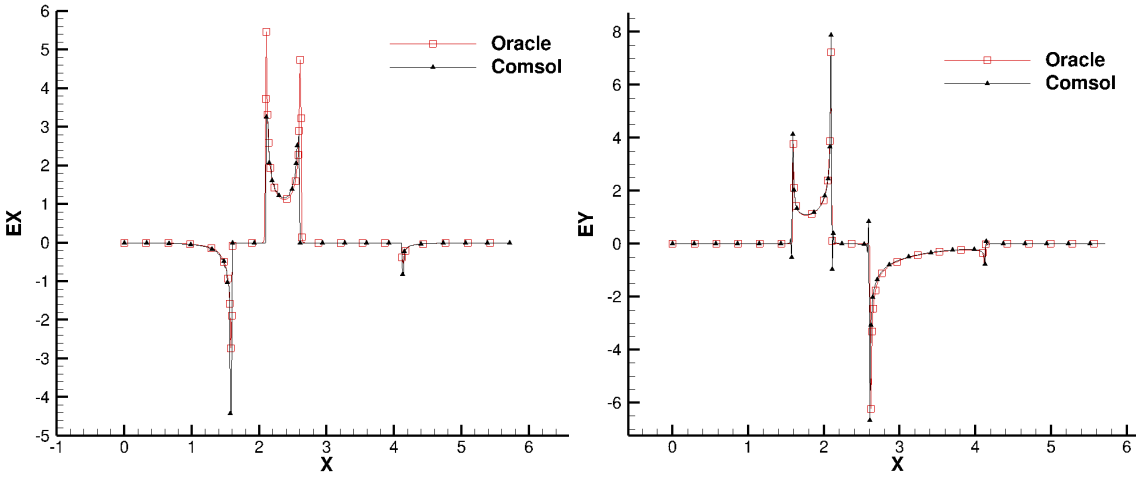


Fig. 6.3.26 Electric field components Ex and Ey on the channel bottom surface (Y=0)

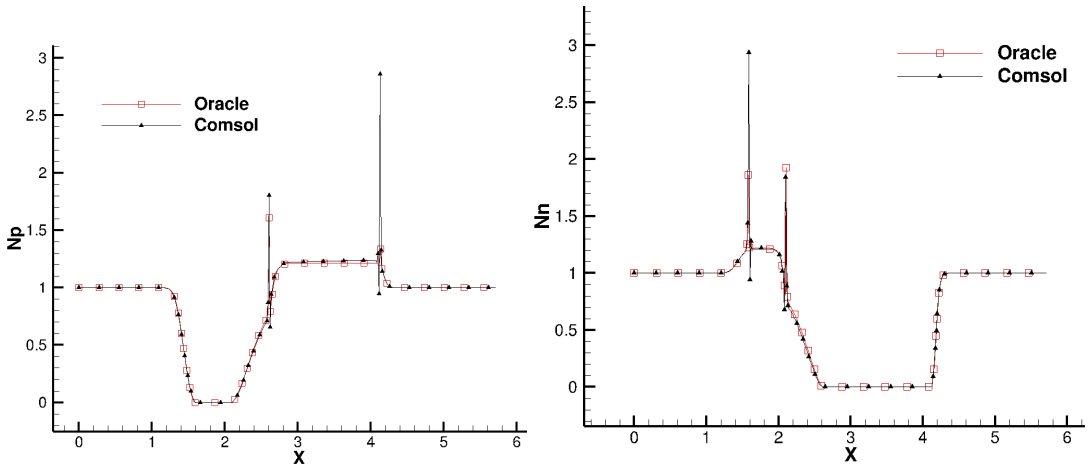


Fig. 6.3.27 Positive (Np) and Negative (Nn) species density on the channel bottom surface (Y=0)

6.3.2 Impact of the Onsager effect

As explained in previous sections, the charge species in electro-conduction are generated by dissociation and recombination mechanisms occurring during the reversible chemical reactions in the liquid electrolytes. It has been established that with the strong electric field intensity the conductance of electrolytes increases [9,10]. This supports the fact that the dissociation and recombination rate coefficients clearly depend on the external electric field. Onsager proposed his field-enhanced dissociation model for incorporating the effect of electric field on the dissociation rate coefficients in such chemical reactions. We have provided a brief overview of Onsager's model and the corresponding dimensional and non-dimensional conduction equations in section 6.2.1. We recall eq. 6.23 here, in which we introduced the dissociation rate coefficient based on the Onsager model.

$$\frac{\partial N_p}{\partial t} + \nabla.\vec{J}_+ = k_{D0}\, F(b)\, c - k_R N_p N_n \tag{6.23}$$

If the dissociation rate coefficient (k_D) is considered independent of the external electric field, then in eq. 6.23 we use $F(b) = 1$. According to the Onsager model $F(b) > 1$ for the electrolytes in high external electric fields, and in that case $k_D > k_{D0}$. k_{D0} refers to the dissociation rate coefficient at thermodynamic equilibrium in the absence of electric field. This theoretical model was proposed by Onsager and he verified the model findings with available experimental measurements as reported in [10]. Our aim here is to verify the implementation of this model in Oracle3D and observe its impact on the electro-conduction as studied here.

We compared two simulation results preformed with our EHD channel configuration (Fig. 6.3.1), with and without using the Onsager effect, and plotted some variables to investigate its impact. On substrate, we used Robin BC for the species and non-homogeneous Neumann for the electric potential. Rest of the BCs are set as mentioned in Fig. 6.2.1. Our results with Onsager effect show an increase in the electric current on the HV electrode, Fig. 6.3.28 (a), which is a result of increased ion concentrations on the electrode surface. It should be noted that in our model of conduction new charge is generated by dissociation, and with Onsager effect we increase the dissociation rate of the neutral species in the liquid. Consequently, the charge fluxes on the electrodes increase leading to higher current values.

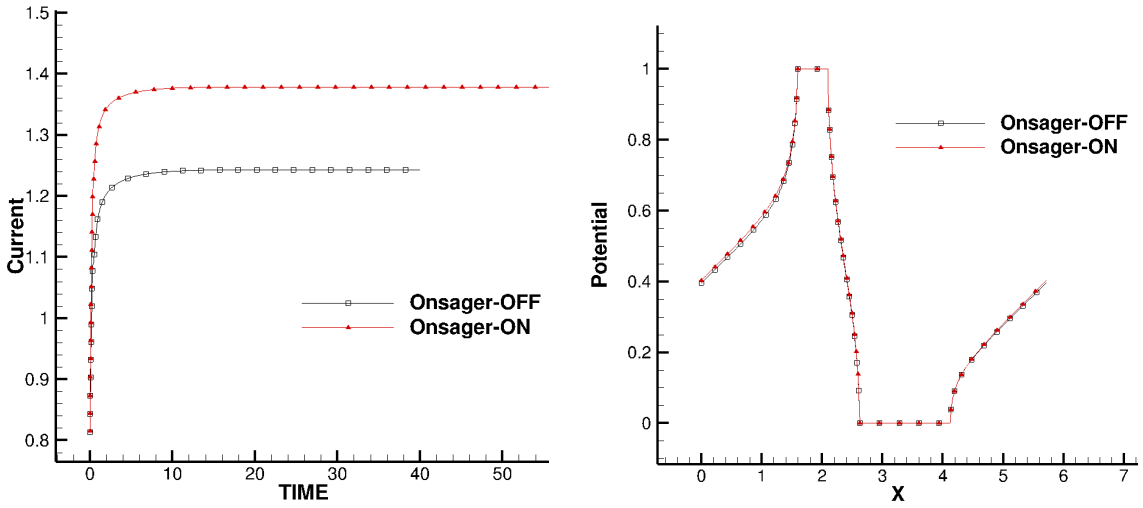


Fig. 6.3.28 a) Evolution of electric current on HV electrode, and b) Variation in electric potential along Y=0 with and without electric potential

The comparison of force components is shown in Fig. 6.3.29, where we see an increment in x-component of force but there is no significant change in the y-component of force. Higher density of charge due to enhanced dissociation modifies the electric field also, both of these factors lead to increment in the x-component of force. Charge distribution next to bottom surface of channel (substrate and electrodes) is also affected by the Onsager effect as observed in Fig.6.3.30, it can be said that the accumulation of charge is increased near the substrate with Onsager model. The density of charge species is especially higher near the edges of the electrodes which are the regions of high electric field intensity leading to higher dissociation occurring in those regions according to the Onsager model (Fig. 6.3.30). However, negligible difference in electric potential on the channel bottom was observed in our plots, Fig. 6.3.28 (b), which is due to the fact that in non-homogeneous Neumann BC for potential we used a fixed value of charge density which will be updated with future implementation regarding this BC.

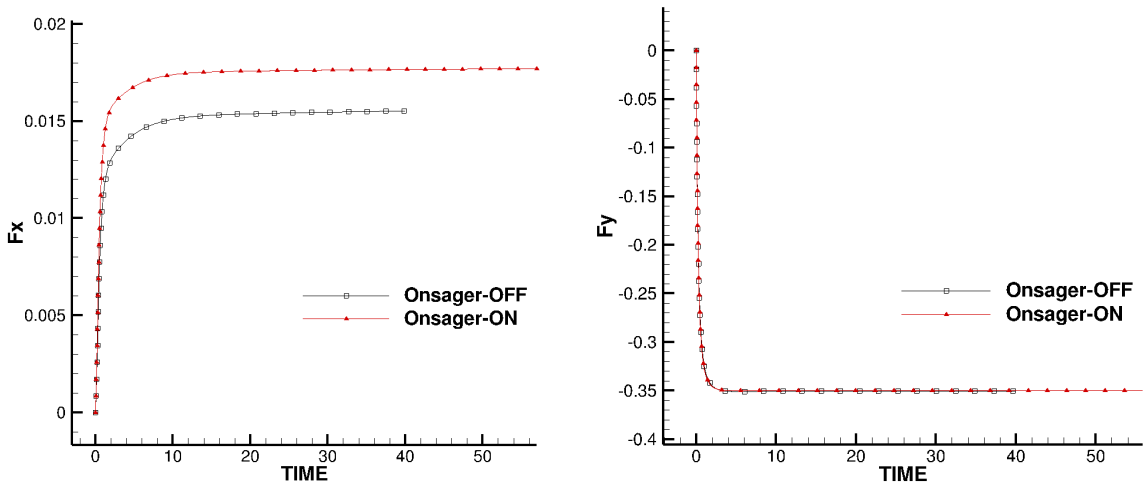


Fig. 6.3.29 Evolution of x and y components of force with and without Onsager effect

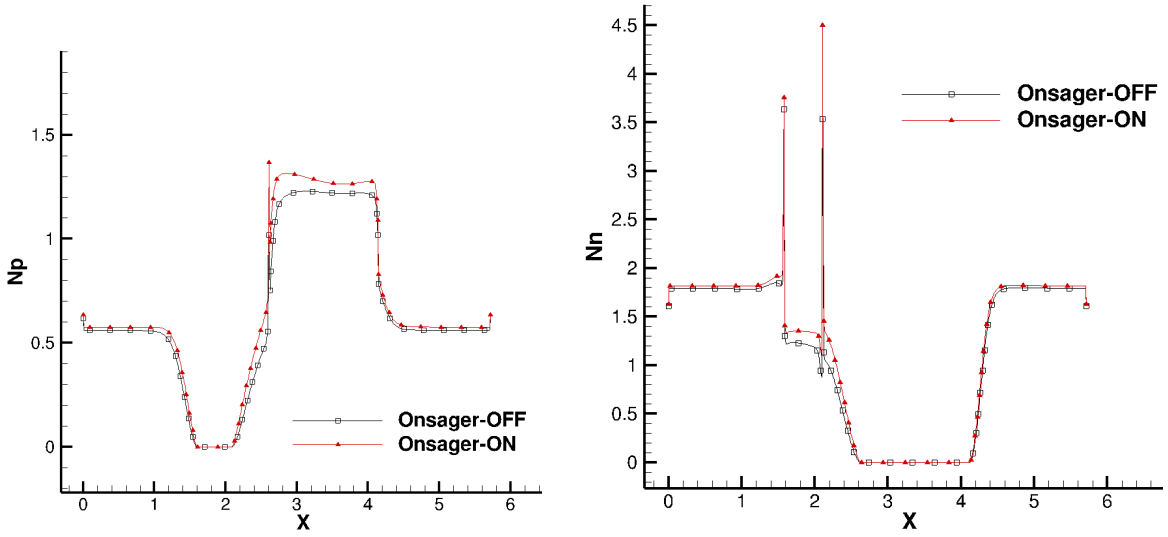


Fig. 6.3.30 Positive charge density (Np) and negative charge density (Nn) next to channel bottom surface

6.3.3 Effect of Robin boundary condition

In our EHD channel case, for conduction modelling, we have to provide some boundary condition for the species density on the substrate. This boundary condition should correspond to the physical phenomenon occurring on that surface. Typically, zero gradient Neumann boundary conditions are used, because of simplicity of implementation and an approximate charge distribution on substrate which would be equal to the charge density in the control volumes at the boundary. More complex is the Robin boundary condition which approximates the accumulation of charge on the substrate based on the species fluxes through the substrate. The charge species fluxes mainly depend upon their convection, through drift in electric field and with neutral fluid motion, and diffusion transport phenomena.

We have described the charge density fluxed with eqs. 6.4 and 6.5, as given in section 6.2 of this thesis. At the substrate surface the flow velocities are given no-slip conditions so the convection through neutral fluid is zero on the substrate. In electro-conduction phenomena as described in this work, any kind of charge generation on substrate is not considered. Thus, numerically there are no source terms to consider in the flux computations at the substrate surface. The final formulation of charge density flux through the substrate depends only on the drift and diffusion of charge species on the surface nodes. So, the Robin BC accounting for the charge accumulation on the substrate is given by the formulation:

$$\textit{Robin BC:} \quad \left(K\vec{E}\varphi - \mathrm{D}\,\nabla\varphi \right).\vec{n} = 0$$

here $\vec{n}$ is the unit normal vector to the surface, φ denotes the positive or negative charge density, and rest of the variables are as provide in section 6.2. On the other hand, the zero gradient Neumann BC is given by:

$$\textit{Neumann BC:} \quad \nabla\varphi.\vec{n} = 0$$

We performed two simulations changing only this charge density BC on the substrate and plotted the results on the line Y=0 (bottom surface of channel). The BC for electric potential on

the substrate was taken as zero gradient Neumann, and the velocities were set to no-slip conditions. The inlet and outlet of the channel were set to periodic BC. The conditions on the electrodes were set as given in Fig. 6.2.1.

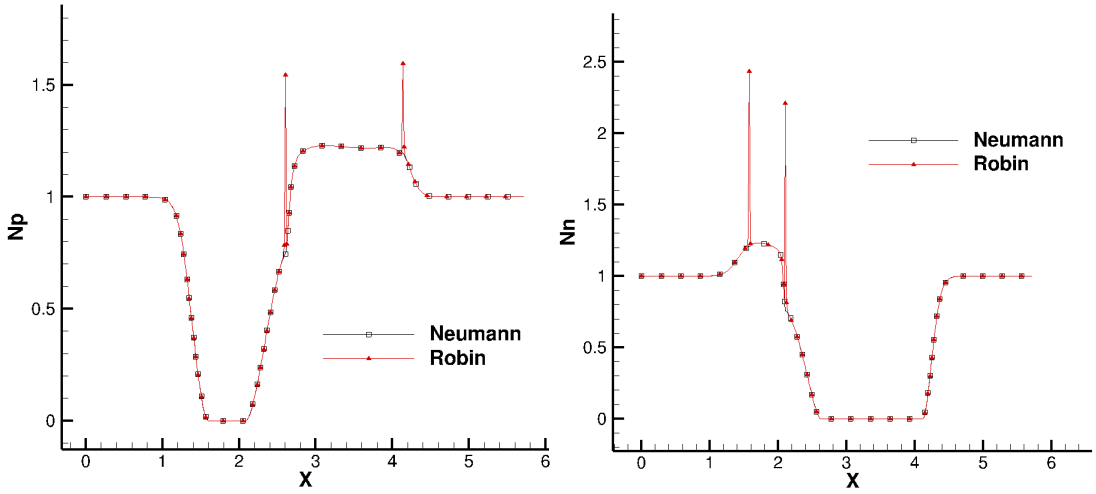


Fig. 6.3.31 Positive charge density (Np) and negative charge density (Nn) along Y=0 line

Fig. 6.3.31 explains an important electrostatic feature with the peaks of charge densities as computed by Robin BC. We know that the electric field magnitude is the highest at the edges of the electrodes, and this should correspond to the highest number density of corresponding charge species. To account for this physical condition, the Robin boundary condition has incorporated the electric field vector in the equation of BC. This leads to the Np and Nn peaks at grounded and HV electrodes' edges respectively (Fig. 6.3.31).

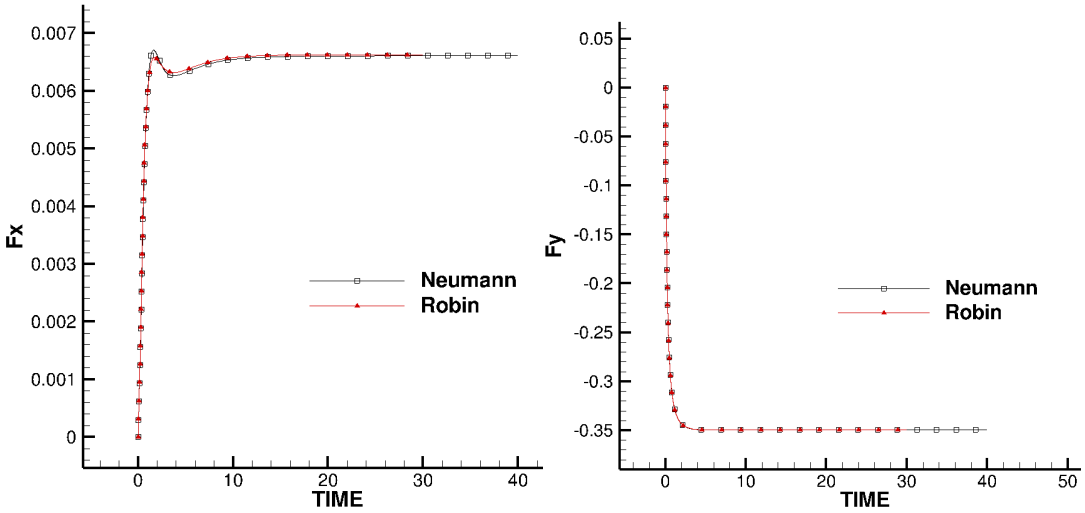


Fig. 6.3.32 Evolution of x and y components of force with Neumann and Robin BC

The zero gradient Neumann BC cannot account for this physical aspect of these configurations, as there is no term which could consider the electric field vector on the surface. And, hence we do not see any peaks of the charge densities with this Neumann BC. We plotted several variables to understand the impact of these charge density peaks on the overall flow. However, the overall force magnitude, force components, electric current on the electrodes, electric potential along Y=0 did not show significant difference with these two boundary conditions, Fig. 6.3.32-6.3.33. Electric field components and mass flow rates with these two boundary conditions also did not show any significant difference. But, as seen in Fig. 6.3.31, the Robin boundary condition is physically more accurate than Neumann BC.

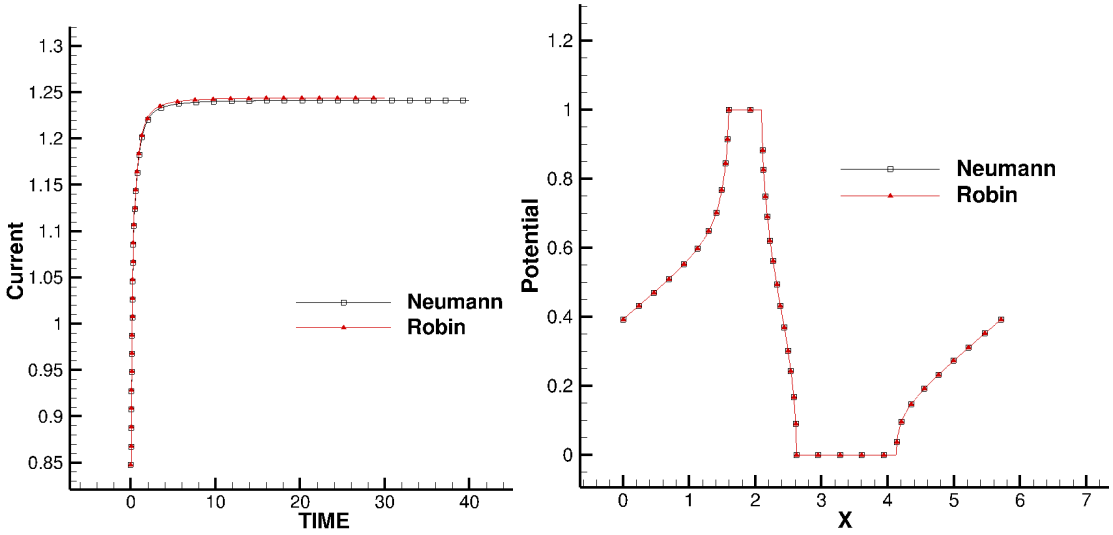


Fig. 6.3.33 a) Evolution of electric current over HV electrode, b) Variation in electric potential along Y=0 with Neumann and Robin BC

6.4 Conduction in blade-plane geometry

Asymmetrical electrode configurations are important in generating a net Coulomb force in EHD flows. In this section we revisit the blade-plane electrode setting with a conduction dominated flow. It is generally believed that the occurrence of conduction and injection phenomenon in dielectric liquids is dependent on electric voltage. Conduction is supposed to manifest when the applied electric voltage is below a threshold voltage which is required for injection to occur [2]. However, experiments show that both conduction and injection can coexist when a strong electric field is present in a liquid. The other factors like the liquid properties, electrode configurations, working temperature, amount of impurities in liquid etc. cannot be neglected while deciding on the question of which phenomena occurs.

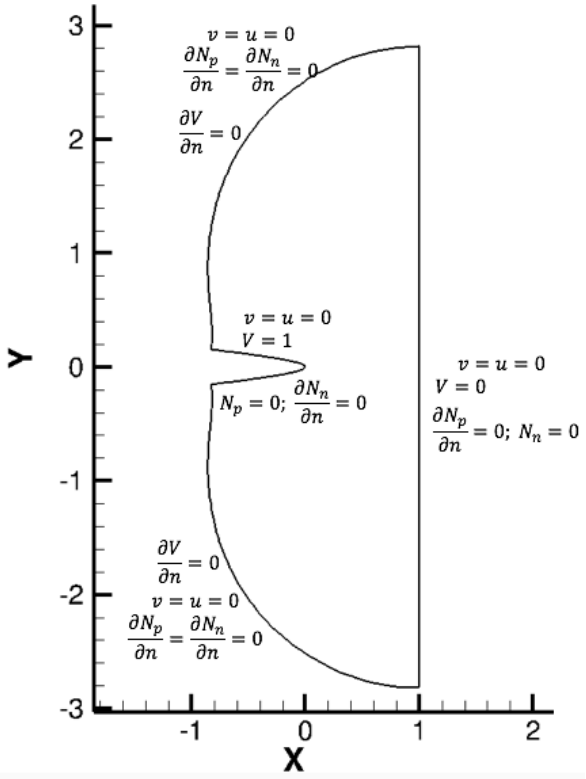


Fig. 6.4.1 Boundary conditions for blade-plane injection case

6.4.1 General Conduction phenomenon

Understanding the flow features is critical in completely describing the EHD phenomenon which generates them. It has been previously observed that in conduction dominant settings the overall flow is from the plane electrode towards the blade electrode [6], which is opposite to the injection cases. Traore et al. (2015) reported a pure conduction case with M=10, Co=6.5, R_{el} = 2.5; they observed a steady flow was rapidly developed with two counter rotating vortices on either side of the blade electrode [6]. Their findings followed the accepted theoretical flow pattern in which the steady state flow was from plane electrode towards the blade electrode. We also simulated the same non-dimensional case with our parallel code to understand the classical behavior of conduction in this setting. The boundary conditions for our problem for all surfaces are provided in Fig. 6.4.1.

Fig. 6.4.2 depicts the velocity vectors and z-vorticity contours for this case (M=10, Co=6.5, R_{el}=2.5) after reaching the steady state, as obtained with our simulation. Two counter rotating vortices are observed in both sides of the blade electrode with velocity vectors. The overall direction of the flow in front of the blade region is from plane electrode to the blade electrode, same as reported in [6]. The z-vorticity contours reaffirm the counter rotating flow in both sides, but the magnitude of the z-vorticity is visibly the same in both vortices. The high magnitude zones of z-vorticity along the blade electrode surface on both sides are due to the steep velocity gradients in the boundary layer region.

The initial non-dimensional values of both positive and negative charge densities were set to 1 in whole domain. Fig. 6.4.3 show the contours of positive, negative and net charge densities at the steady state. We observe the heterocharge layers on both the electrodes, which are typical with conduction phenomenon. The electric field direction is from blade to plane electrode, as blade electrode is set to electric potential V=1 and the plane electrode is grounded (V=0). Thus, the negative charges in the domain drift towards the blade electrode, opposite to the electric field, and form a layer around this electrode. Similarly, the positive charge density heterocharge layer is formed over the plane electrode.

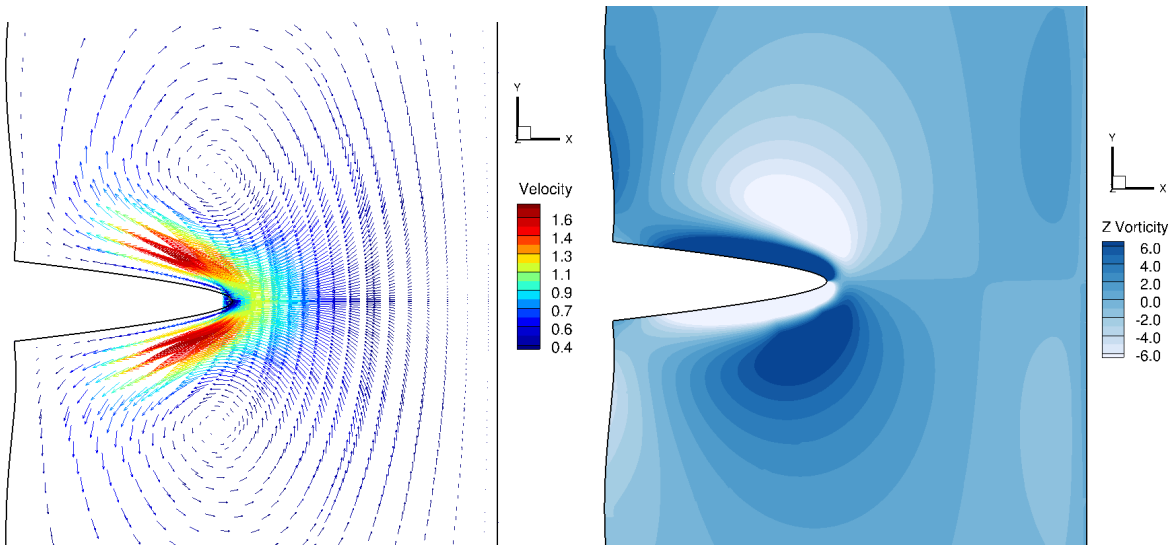


Fig. 6.4.2 Velocity vectors and z-vorticity contours at steady state, with R_{el}=2.5

The overall flow of the liquid is due to this motion of charge species, which transfer their momentum to the neutral liquid particles. This effect is represented in the momentum equations of the liquid with the added source term due the Coulomb force. This kind of flow is observed by us in other conduction dominated cases, and this is the same classical conduction case as reported by Traore et al. (2015).

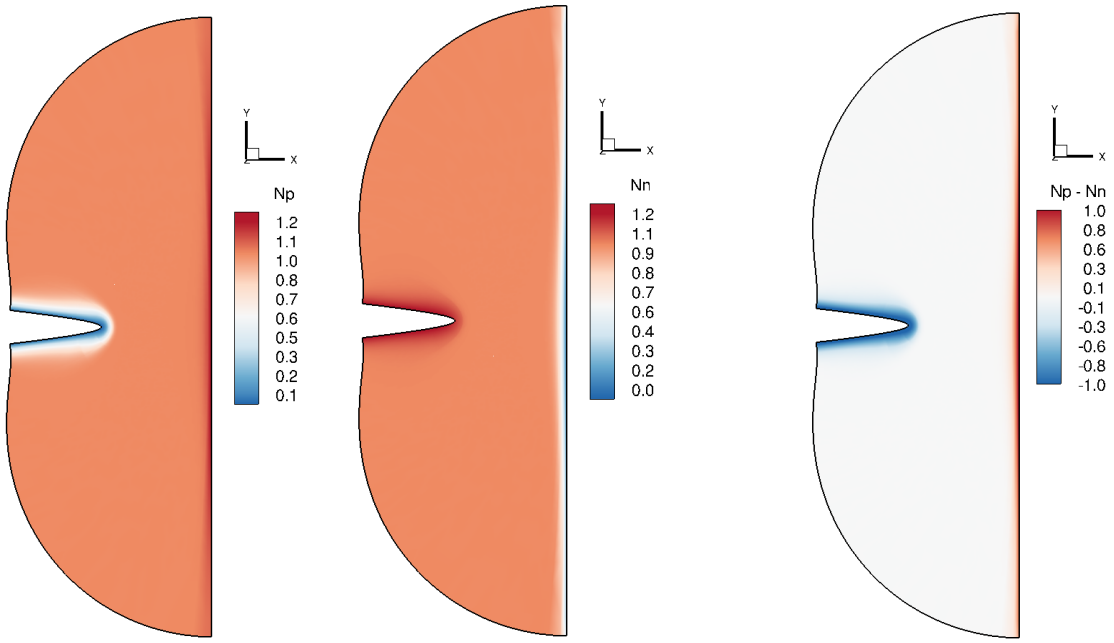


Fig. 6.4.3 Positive (Np), negative (Nn) and Net charge density (Np-Nn) contours at steady state with R_{el}=2.5

6.4.2 Reversed flow phenomenon in conduction with higher $\mathbf{R_{el}}$

We carried out some numerical tests to examine the flow features in blade-plane configurations with a different set of case parameters, in the influence of conduction. The boundary conditions for this problem are as given in Fig. 6.4.1. For the input parameters, we took M=0.2, Co=0.1 with the electric Reynolds (R_{el}) number equal to 2500. The simulation was run for 5 non-dimensional time units, and velocity vectors were recorded after every 0.05 non-dimensional time.

Flow structure

Fig. 6.4.4 shows velocity vectors with z-vorticity contours at two non-dimensional times (t=0.75, 5). We observe that initial flow, on the blade surface, is seen going from plane to blade direction (t=0.75); but at t=5 we observe that the vectors are moving from blade to plane direction. This was found to be contradictory to what was observed in our classical conduction case and ref. [6]. The observed flow behavior presented two different types of flow regimes in this case. We term the classical conduction flow regime as R1, where the flow direction in steady state is from plane to blade electrode, as reported in [6]. The flow behavior observed at steady state with this case settings is termed as regime R2, where flow direction is from blade to plane electrode, Fig. 6.4.5. In this R2 regime, flow behaves similarly as found in injection cases, where the positive charges are injected from the blade electrode and they move towards the grounded plane electrode. However, the conduction flow with two species is more complex than injection cases. We investigated the flow with more plots of vectors at different time steps to get a better understanding of the overall flow.

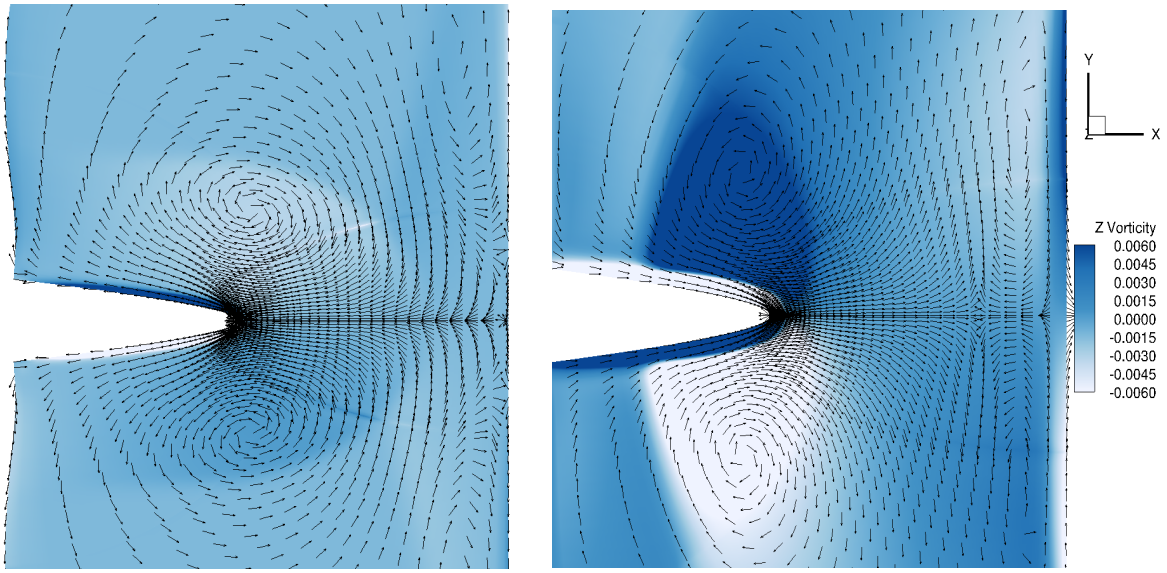


Fig. 6.4.4 Velocity vectors with z-vorticity contours at t=0.75 and 5

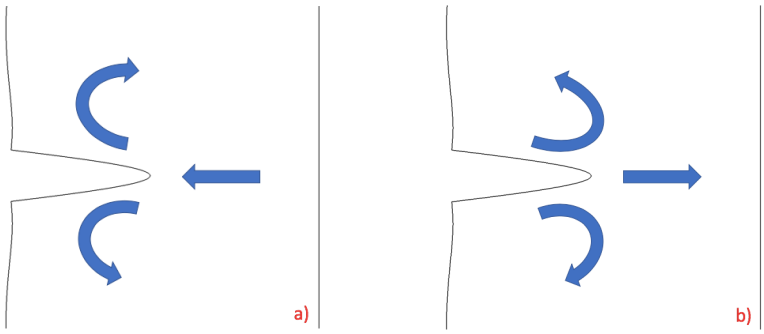


Fig. 6.4.5 Sketches representing flow structure in two regimes of flow, a) R1 and b) R2

In Fig. 6.4.6 we show some figures with velocity vectors. The figures are taken at non-dimensional time = 0.35, 1.4, 2.15, 3.3 and 5 respectively starting from top left. We observe that in initial stage the vectors are pointing from plane to blade direction along the blade surface, and two counter rotating vortices has started from the blade tip. At time 1.4 two new counter rotating vortices are seen originating from the blade tip again, but this time these vortices are rotating in opposite directions with respect to the earlier vortices at the same side of the blade electrode. This changed direction of rotation is due to the induced Coulomb force at blade tip which is seen pulling the flow vectors towards the blade tip at t=2.15.

In third figure at t=2.15 two more vortices are seen near the plane electrode surface. At this time the flow has started changing its direction at the tip of the blade but in the vicinity of the plane electrode the flow is still seen moving towards the blade. The R2 regime is observed developing in Fig. 6.4.6 (d), where the flow in front of the blade tip is observed to be from blade to plane electrode. We also notice that a stagnation line is formed in the domain between the blade tip and the plane electrode seen in Fig. 6.4.6 (d) and (e). This is due to the complex vortex

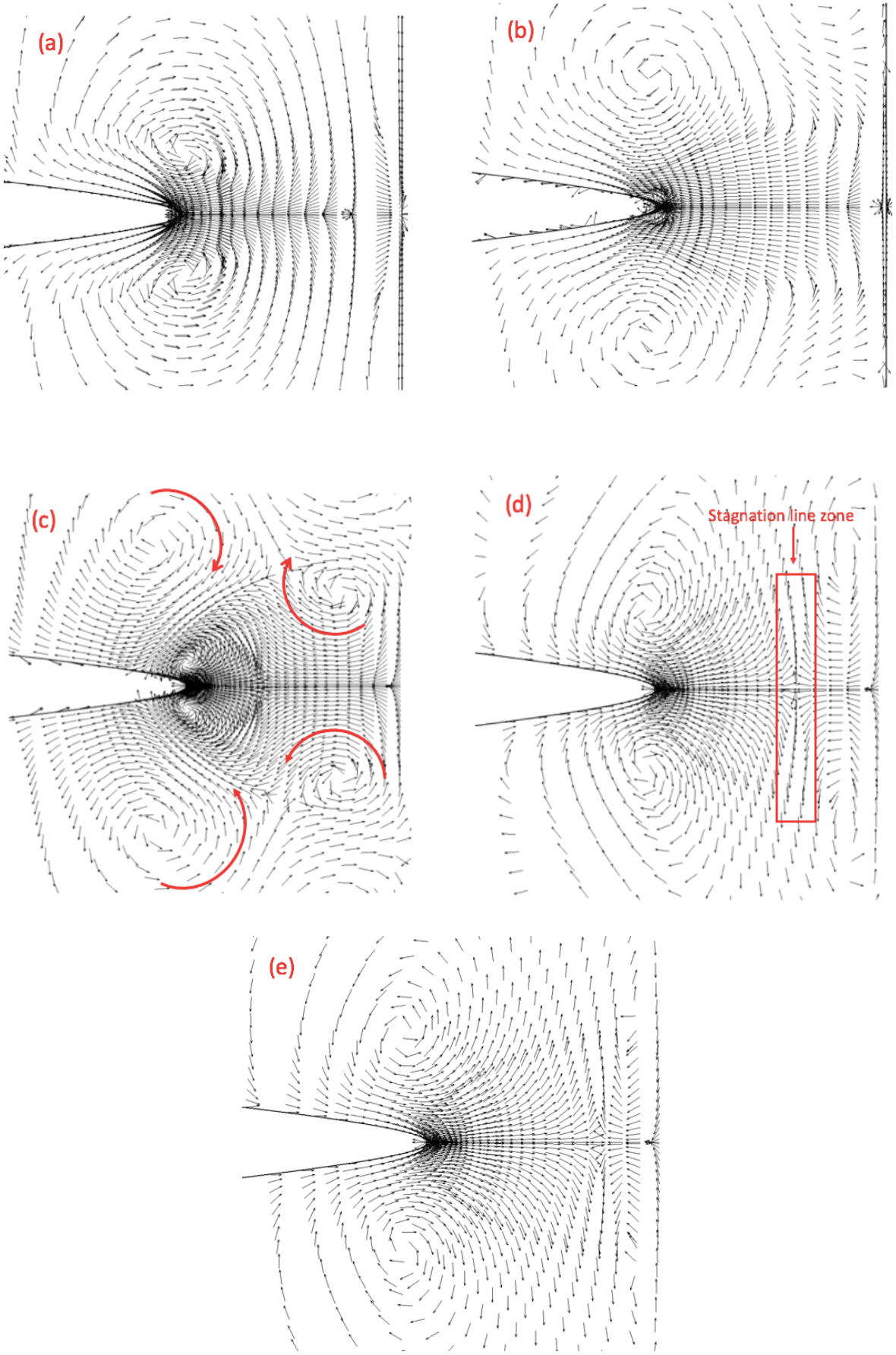


Fig. 6.4.6 Velocity vectors taken at non-dimensional time a)0.35, b) 1.4, c)2.15, d)3.3 and e)5

motion developing in the step (c) of Fig. 6.4.6. There are two vortices forming on each side (upper and lower) of the blade electrode which are rotating in the counter directions to each other. The stagnation line of the flow is observed where the liquid layers relating to these two vortices meet each other. In this small stagnation region, the vectors moving in opposite directions are passing by this common line.

At t=3.3 these vortices are seen growing and moving away from the blade tip, and more and more vectors are now pulled towards the blade tip. There is a prominent zone between the electrodes, starting from the blade surface, where the flow is now seen towards the plane electrode. The magnitude of velocity was observed to be increasing near the tip of the blade. By the time simulations reach t=5, the flow attains a steady state and the effective flow around the blade area is observed moving towards the plane electrode with the counter rotating vortices still seen on both sides of the blade electrode. The flow near the plane electrode is reduced to thinner layers than earlier but the vectors are still pointing towards the blade. We shall attempt to relate this flow motion with the movement of charge species in following paragraphs.

Transport of species

The flow pattern we discussed above is due to the complex movement of charge species under the influence of electric force. The formation of hetrocharge layers on the electrode surfaces is a well observed feature of conduction in dielectric liquids. In Fig. 6.4.7, we see that at t=0.35 there are the finite thickness hetrocharge layers of positive and negative species on the plane and blade electrodes respectively. The blade electrode is with positive polarity, so it attracts the negative species and the plane electrode attracts the positive species. However, as we progress in time we note that the charge layers are growing in thickness asymmetrically over the electrode surfaces as seen in Fig. 6.4.8. Especially at the tip of the blade and the plane electrode portion just in front for blade tip are seen loosing their charge densities with time.

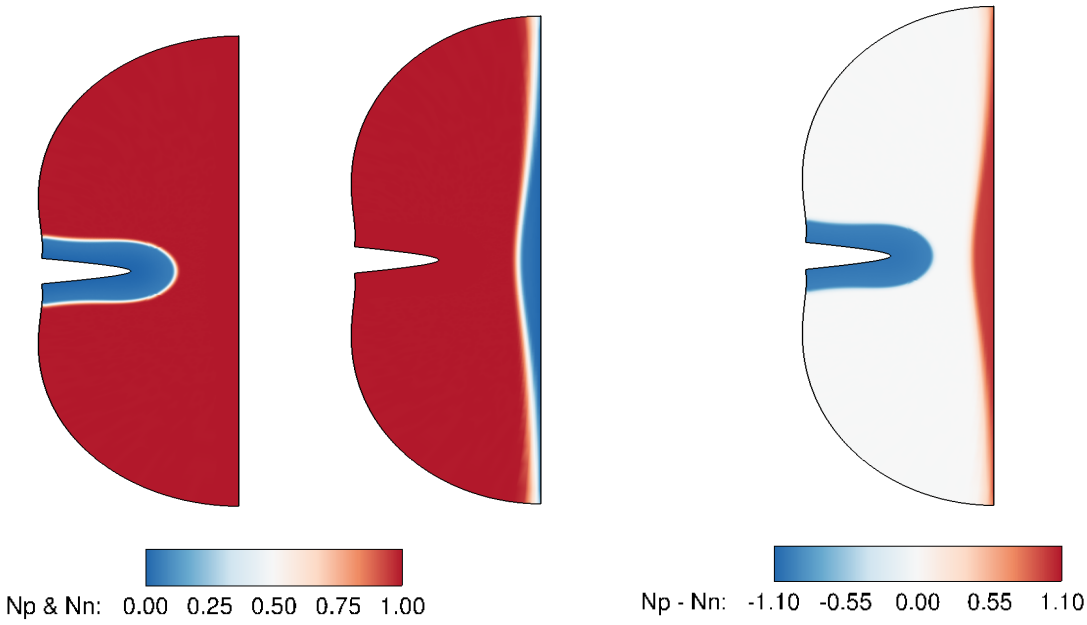


Fig. 6.4.7 Positive (Np), negative (Nn) and Net charge density (Np-Nn) contours at t=0.35

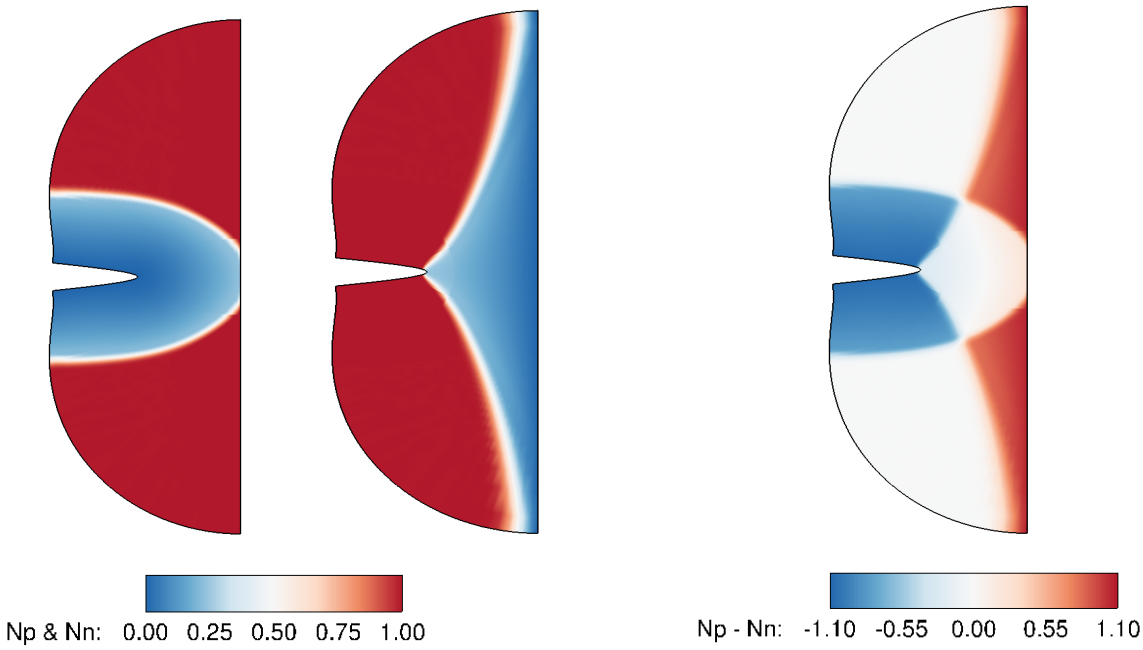


Fig. 6.4.8 Positive (Np), negative (Nn) and Net charge density (Np-Nn) contours at t=1.4

Charge density figure. 6.4.8 corresponds to the time when we start observing the second pair of vortices at the tip of the blade in Fig. 6.4.6 (b). This indicates that the second pair of vortices starts forming at the tip of the blade when the negative charges are shifting away from the tip. The domain in front of the tip seems more occupied by the overall positive charge, which moves towards the plane electrode. Till the end of the simulations the charges are moving in similar way and a steady state flow is obtained. At t=5, we do not observe hetrocharge layers of same density along full surfaces of both electrodes as seen in Fig. 6.4.9.

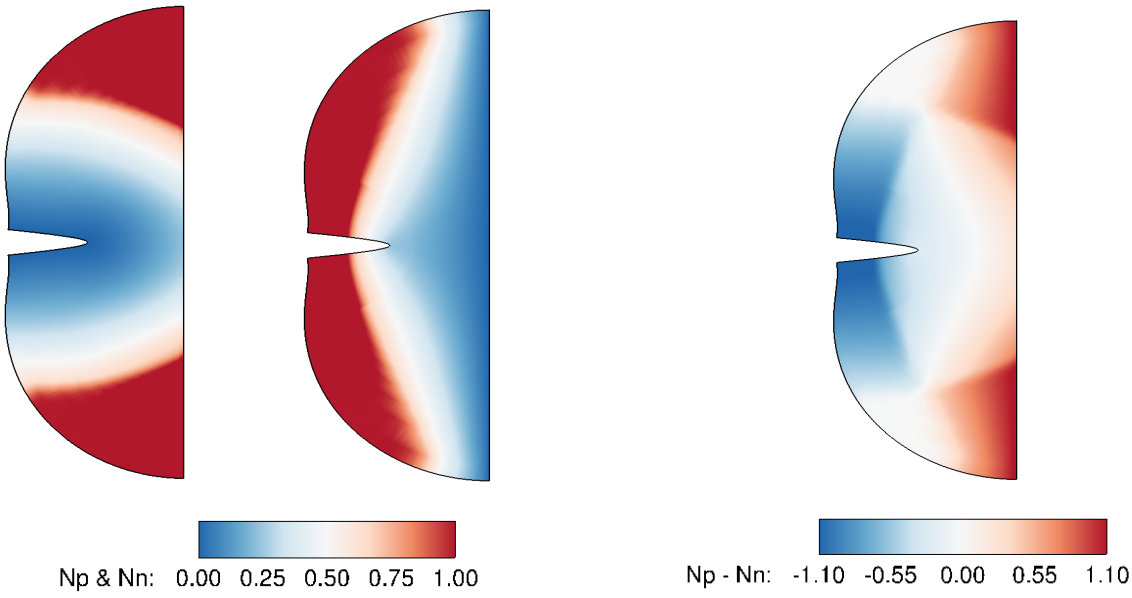


Fig. 6.4.9 Positive (Np), negative (Nn) and Net charge density (Np-Nn) contours at t=5

Analysis of the obtained flow structure

The transport of species depends on their charge and the external electric field. The electric field is mainly due to the applied voltage difference ($V_0 - V_1$) between the two electrodes which is kept constant with time in this problem. The contribution due to the unsteady charge species' densities in generating electric field is negligible and thus, the electric field vectors are not changing with time and pointing from blade electrode to the plane electrode, as shown in Fig. 6.4.10. The components of electric field are also shown in Fig. 6.4.11. It is evident that the negative charge species will drift opposite to the electric field direction and the positive charge species will drift along the electric field direction.

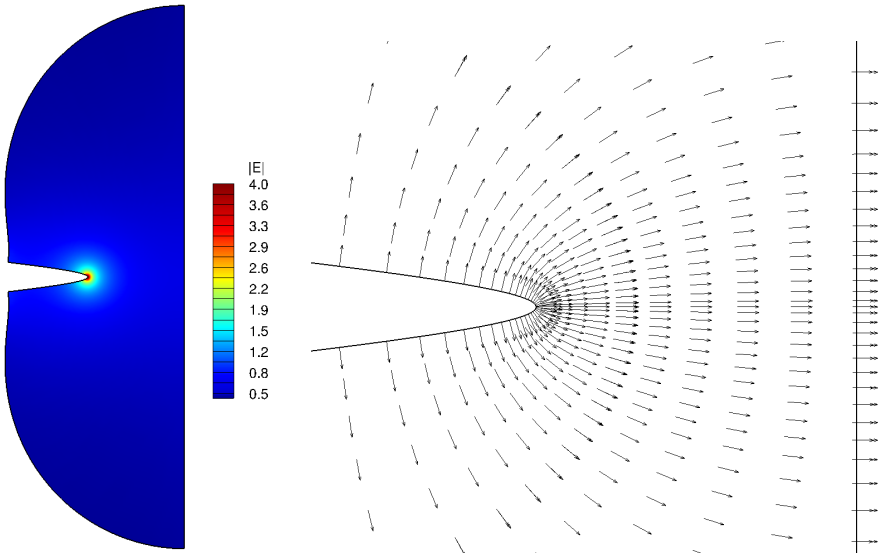


Fig. 6.4.10 Non-dimensional electric field magnitude contour and the electric field vectors.

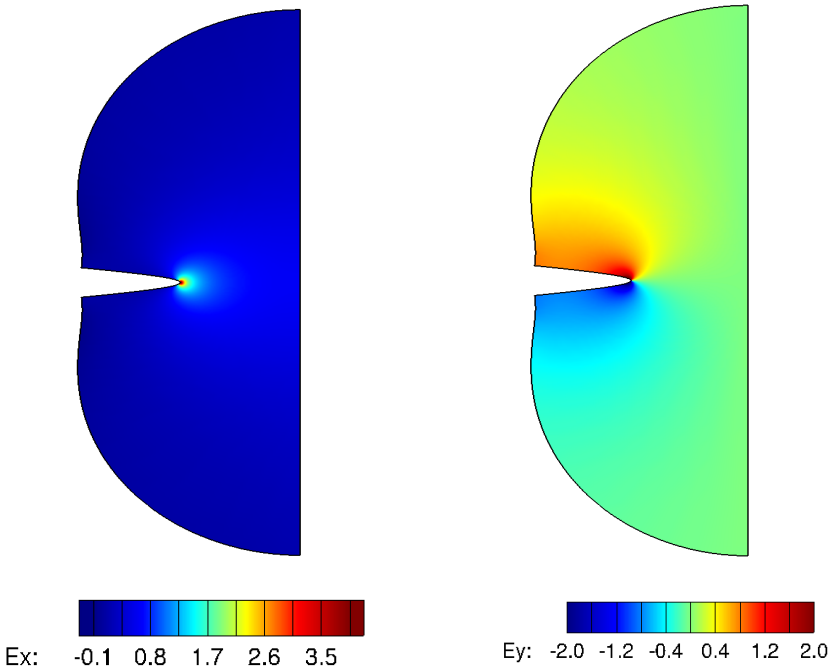


Fig. 6.4.11 The non-dimensional electric field components

The complex transport of the charge species under the influence of electric field within the domain walls gives rise to several vortices in the domain as visible in Fig. 6.4.6 (c). These vortices also promote higher degree of mixing in the developing flow and plays an important role, with the electric field, in determining the charge species overall distribution around the electrodes. In our case settings, initially, the direction of the flow around the blade electrode is mainly determined by the transport of the negative species against the electric field direction. The Coulomb force on the negative charges is such that it leads the flow surrounding the blade electrode to move from plane to blade direction, which represents the regime R1 as mentioned earlier. This regime can be explained with the Fig. 6.4.12 (a), where we show the directions of electric field components and the consequent electric force vectors' direction in region surrounding blade electrode where net charge is negative initially. A thin layer of fluid surrounding the plane electrode always moves towards the plane electrode which is driven by the positive charge species attracted by the plane electrode (V=0).

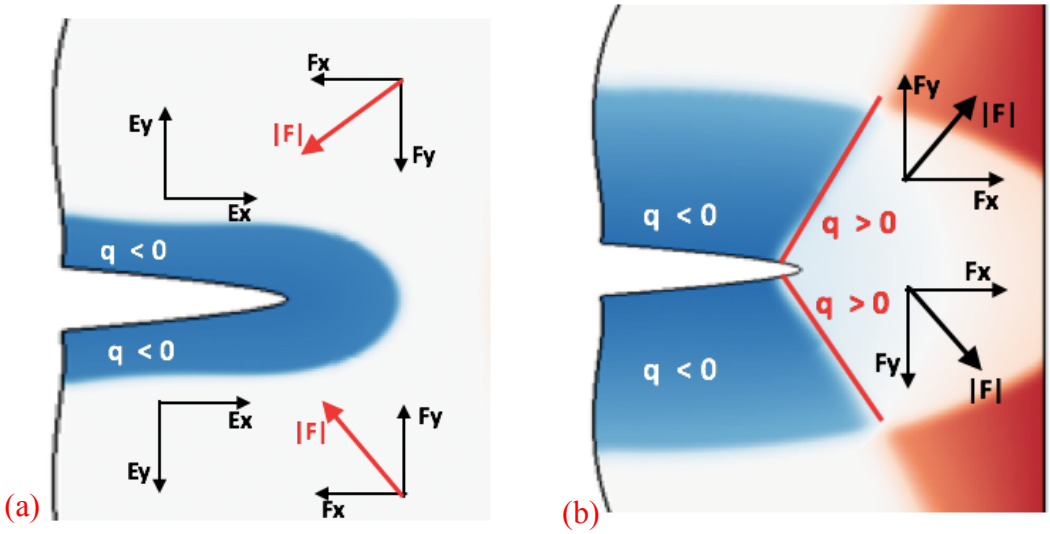


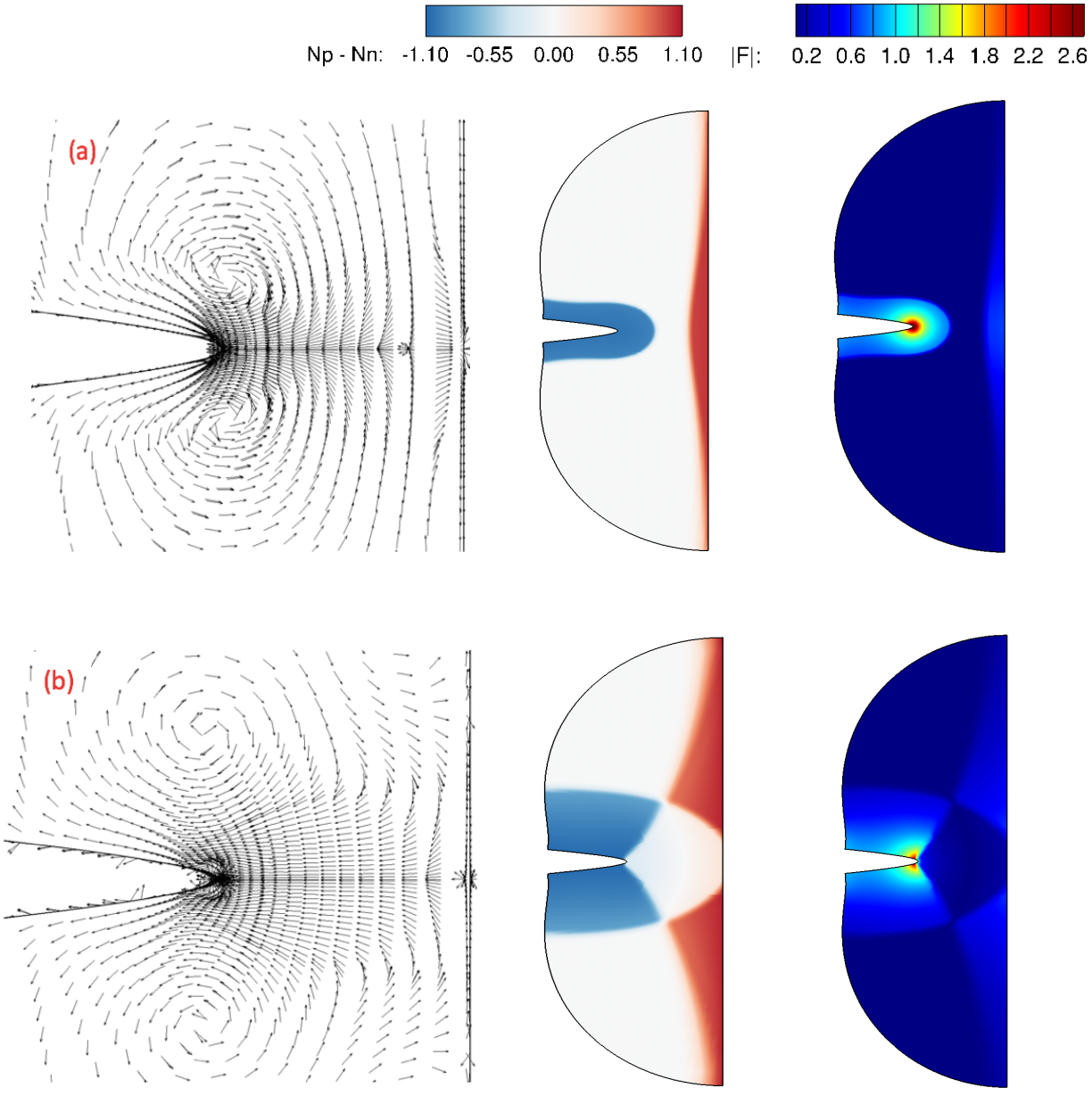
Fig. 6.4.12 Sketches depicting the directions of electric field and the electric forces in the regions of positive and negative net charge densities

After a certain amount of time, due to the drift of the species and the mixing introduced by the vortices lead to a situation where the density of charge species starts turning positive around the tip of the blade electrode. The negative species are observed to be pushed away from the tip of the blade electrode along its surface in negative x direction (Fig. 6.4.8), and due to the transport of positive species in the direction of electric field a significant amount of fluid starts moving towards the plane electrode. The overall positive charge in the region in front of the tip leads to this regime of flow which we referred by R2 previously. The overall electric force in front of the blade tip region, where we have positive charge as dominant species, is shown with sketches in Fig. 6.4.12 (b). It should be noted that the electric field direction remains the same with time, as the impact of unsteady charge distribution on electric field is not significant.

In steady state, the overall flow presents the R2 regime where the flow in front of the blade tip moves from blade to the plane electrode. Fig. 6.4.13 provides an overview of the whole flow mechanism by showing the vectors, net charge density and the electric force contours at the same time instants together. The stagnation line is also visible in the force magnitude contours

at step (d) and (e) of Fig. 6.4.13, where a dark blue line (|F|=0) is seen extending in the domain on both sides of the blade electrode, in the region between the two electrodes. We also notice that due to the symmetrical geometry the flow features are symmetric with respect to the horizontal line passing through the tip of the blade electrode.

An overall observation suggests that such conduction phenomena involve several factors to introduce such complex flow regimes. The distribution of charge density which keeps on changing due to the transport, generation and destruction of the species plays a significant role in producing the overall flow features. The electric field influences the drift of the species. And the vortex motion of fluid plays an important role in bringing and taking away fresh amount of charges to and from the electrodes. The properties of fluid and the amount of impurities should also be accounted carefully to describe such flows more precisely.



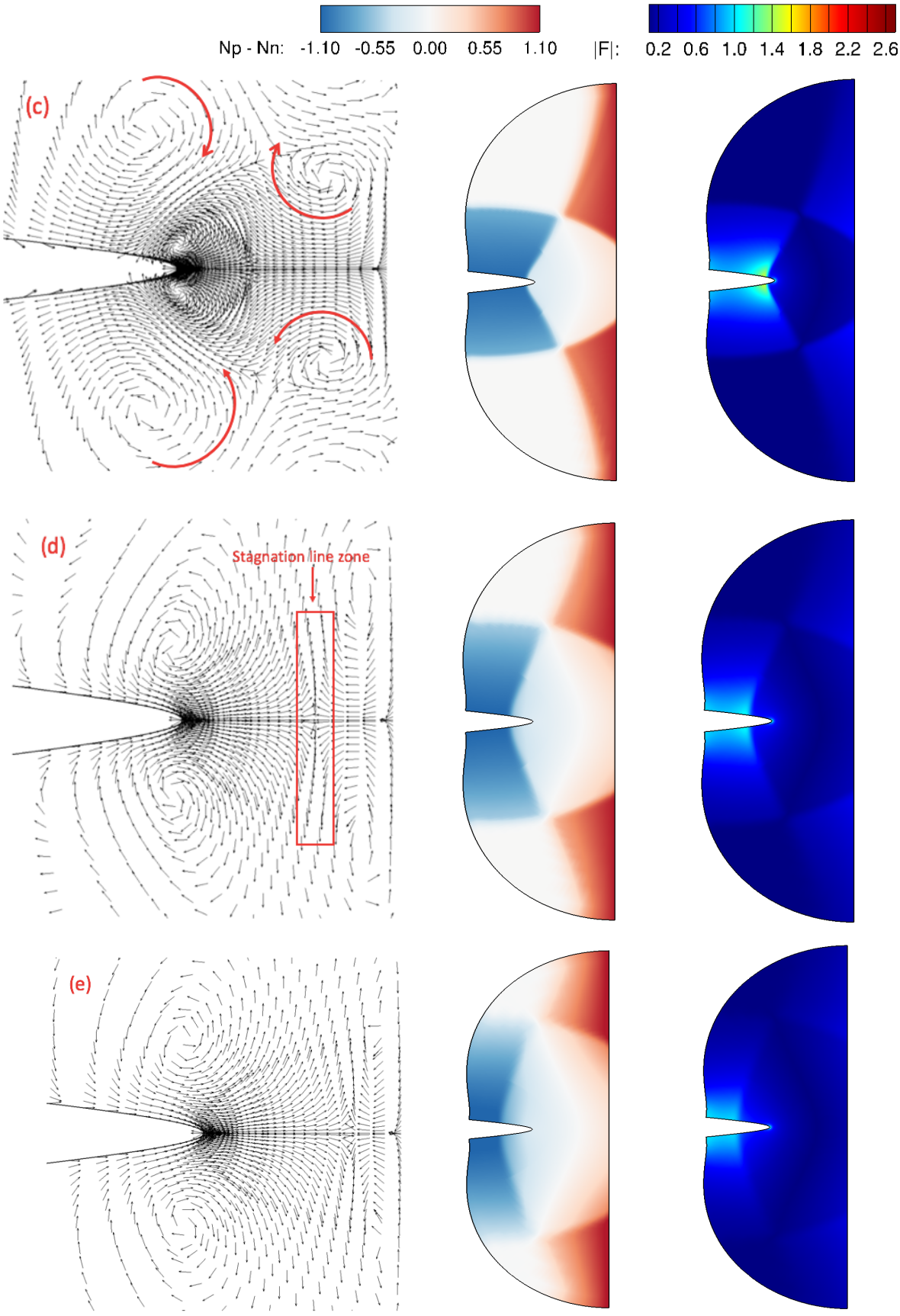


Fig. 6.4.13 Velocity vectors, Net Charge and the Force magnitude. Non-dimensional time a) 0.35, b) 1.4, c) 2.15, d) 3.3 and e) 5.0

This whole study was repeated with a 3D grid also to verify whether the flow had 3D effects. We verified that the z-component of velocity was zero, and the velocity vectors were found to be of same pattern in all the planes at different z locations in the domain. We show here the vorticity magnitude (Fig. 6.4.14) and z-vorticity (Fig. 6.4.15) contours to verify that the flow was two dimensional as these variables were found to be same in all planes along z direction. And, the flow attained the steady state as was observed with the 2D simulation.

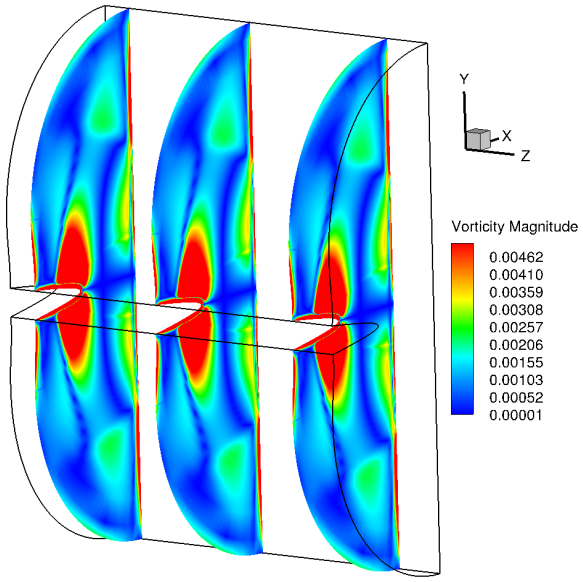


Fig. 6.4.14 Vorticity magnitude contours at t=5 on three z planes.

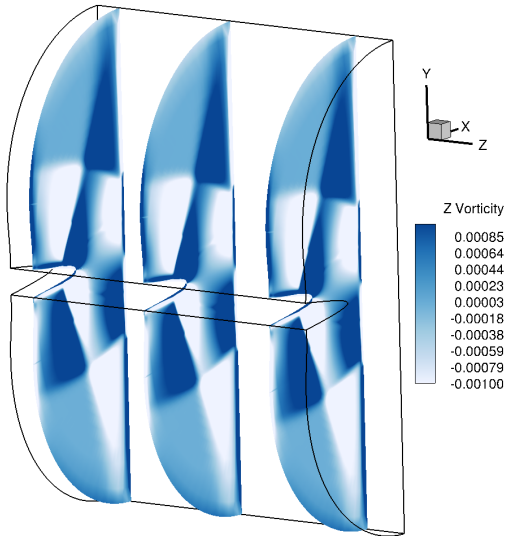


Fig. 6.4.15 Z-vorticity contours at t=2.15 on three z planes.

6.4.3 Case with 10000 R_{el}

We investigated another case with higher electric Reynolds number (R_{el} = 10000), the other set of parameters being M=0.1, Co=0.1. In this case we reduced M value by half. We first analysed the velocity vectors obtained with the 2D simulations. Fig. 6.4.16 shows the vector pattern at t=0.75 and t=5. We observed same pattern as was discussed above with M=0.2, the flow vectors were initially seen going from plane to blade electrode but after certain time the flow along the blade surface was clearly from blade to plane electrode. As the R_{el} value is four times higher in this case, we investigated the 3D case also for the same settings.

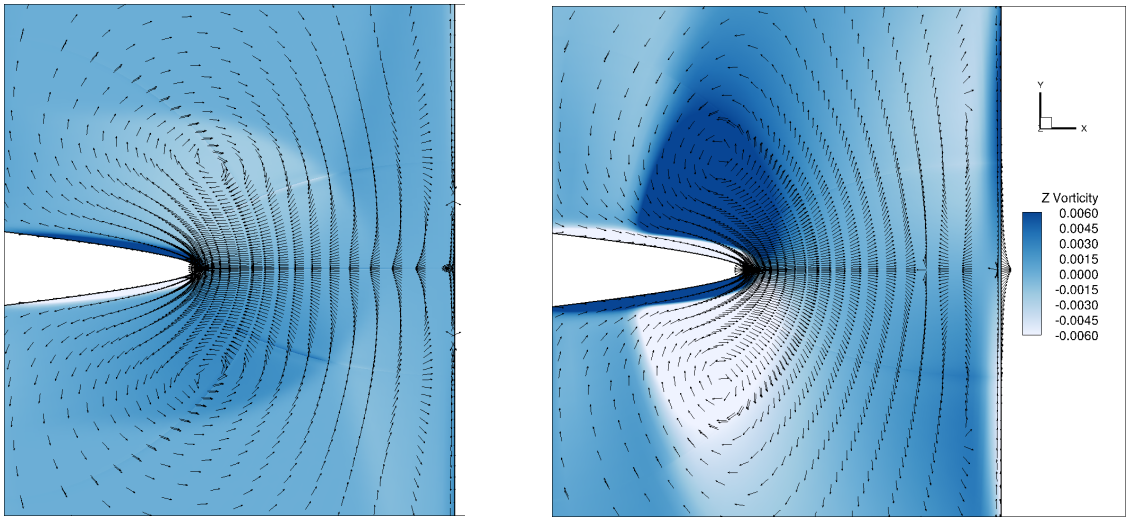


Fig. 6.4.16 Velocity vectors with z-vorticity contours at t=0.75 and t=5

We plotted the z-vorticity iso-surfaces to analyse the symmetry in vortices, if any exists. In the initial time steps we observed that the z-vorticity iso-surfaces were symmetrical along the z axis, Fig. 6.4.16 (a). However with the evolution of the flow we started observing the turbulent nature of the flow, and the symmetry of flow along the z direction was taken over by real 3D features. The z-vorticity iso-surfaces at t=1.5 suggests that the flow is evidently three dimensional where iso-surfaces of values +10 and -10 are intricatly mixed with each other, Fig. 6.4.17 (d). In the 2D simulation we observed a steady flow with two counter rotating vortices in Fig. 6.4.16 but in 3D flow the vorticity magnitude iso-surfaces plot suggests that the flow was unsteady and turbulent where steady vortices do not exist, Fig. 6.4.17. We also observed significant z component of velocity in comparison with x, y components, which confirmed that the flow was 3D.

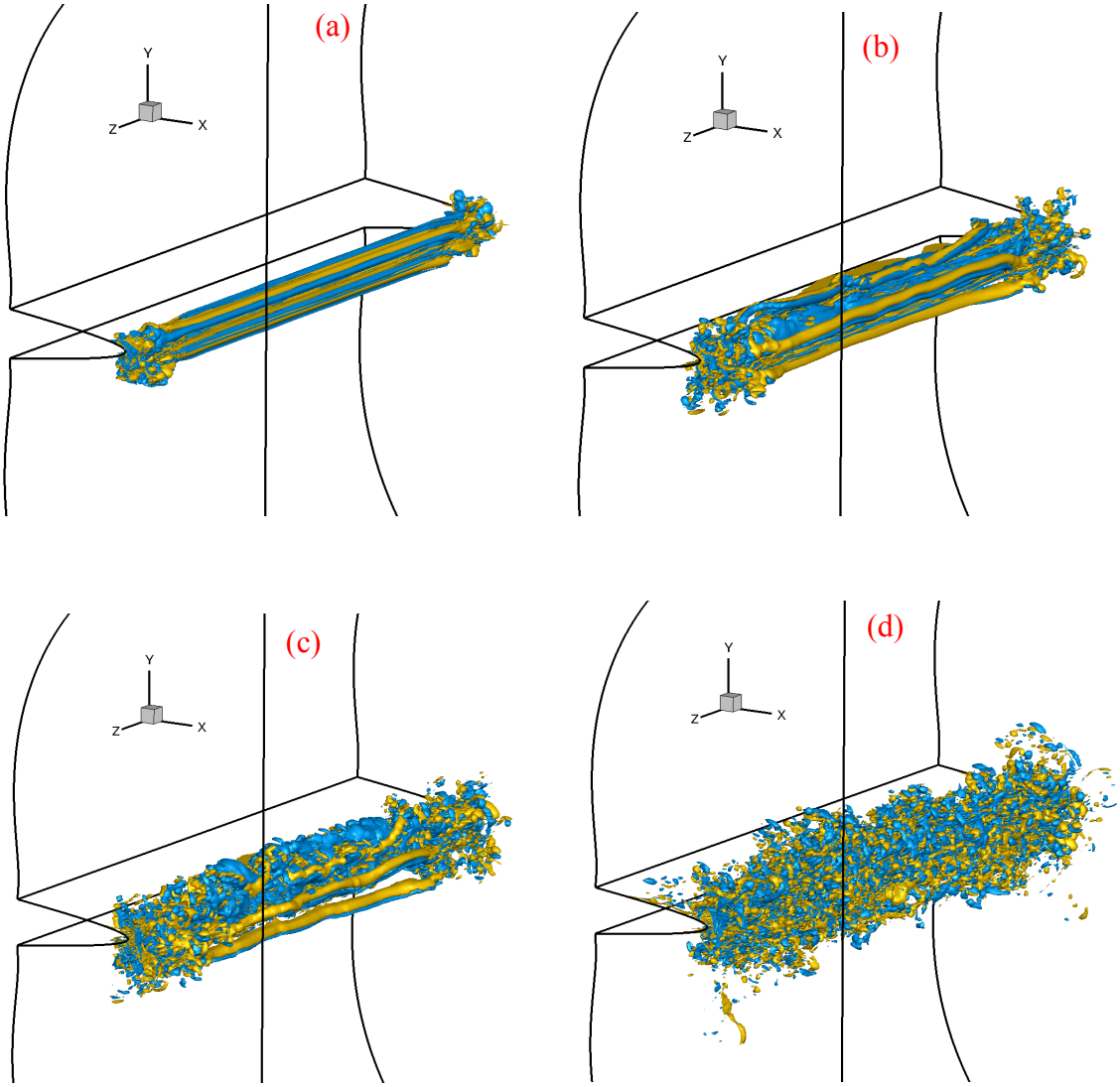


Fig. 6.4.17 Evolution of z-vorticity iso-surfaces at ω_z =-10 (blue), +10 (yellow); a) t=0.15, b) t=0.5 , c) t=0.8, d) t=1.5

Recently, Louste et al. (2018) investigated the electroconvective cavity flow pattern with a cylinder-plane electrode configuration. They observed charge injection from cylinder with positive voltage polarity on plane electrode and a conduction phenomenon when the cylinder is kept at positive voltage. They did not observe the transition from conduction to injection with an increase in positive voltage on cylinder electrode unitll 22 kV [2]. They also noted 3D effects in the flow and mentioned that the threshold electric field as described in literature could not be detected with their dielectric liquid (HFE 7600).

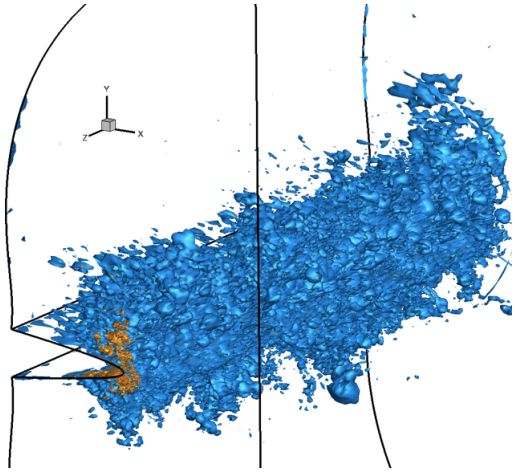


Fig. 6.4.18 Vorticity magnitude iso-surfaces at values 5(blue) and 15 (orange) at t=12

We noticed that with M=0.2 the flow was 2D and with M=0.1 we observed 3D effects. The M parameter is given by $M = \frac{1}{K}\left(\frac{\varepsilon}{\rho_0}\right)^{1/2}$, which is a fuction of liquid properties alone. It shows that the flow pattern in conduction with asymmetrical electrode configurations is strongly dependent on the liquid properties and not alone the applied voltage. More detailed analysis with different liquid properties, with impurities fraction and electrical inputs have to be investigated to further improve the understanding of such EHD flows. We also highlight that in such electro-conduction flows there are some combinations of non-dimensional parameters (M, C_0, R_{el}) for which the flow will remain in regime R1, which is considered as typical conduction behavior. However, we can have some combindations of fluid properties and input parameters which will lead to such a set of parameters (M, C_0, R_{el}) for which we shall observe the flow reversing to R2 regime. We mentioned several factors which affect this behavior and further studies are required for a broad understanding of these phenomena.

Bibliography

[1]. P. Atten, J. Seyed-Yagoobi, "Electrohydrodynamically induced dielectric liquid flow through pure conduction in point/plane geometry," IEEE Trans. Dielectrics and Electrical Insulation, 10 (1), (2003)

[2]. C. Louste, H. Romat, P. Traore, M. Daaboul, P. Vazquez, R. Sosa, "Electroconvective cavity flow patterns created by asymmetric electrode configuration," IEEE Trans. On Industry Applications, (2018)

[3]. M. Yazdani, J. Seyed-Yagoobi, "Electrically induced dielectric liquid film flow based on electric conduction phenomenon," IEEE Trans. Dielectrics and Electrical Insulation, 16 (3), (2009)

[4]. M. Yazdani, J. Seyed-Yagoobi, "Effect of charge mobility on dielectric liquid flow driven by EHD conduction phenomenon," Journal of Electrostatics, 72, 285-294, (2014)

[5]. Y. Feng, J. Seyed-Yagoobi, "Electrical charge transport and energy conversion with fluid flow during electrohydrodynamic conduction pumping," Phys. Of Fluids 19, (2007)

[6]. P. Traore, J. Wu, P. A. Vazquez, C. Louste, C. Gouriou, A. Perez, "Numerical simulation of Electrohydrodynamically induced dielectric liquid flow through pure conduction blade-plane geometry," 11th Int. Conference on Modern Problems of Electrophysics and Electrohydrodynamics (MPEE-2015)

[7]. Y. K. Suh, "Modeling and Simulation of Ion transport in dielectric liquids – fundamentals and review," IEEE Trans. Dielectrics and Electrical Insulation, 19 (3), (2012)

[8]. M. Yazdani, J. Seyed-Yagoobi, "The effect of charge injection on EHD conduction pumping," IEEE Industry Applications Society Annual Meeting, (2010)

[9]. A. Castellanos, "Electrohydrodynamics," Springer-Verlag Wein, (1998)

[10]. L. Onsager, "Deviations from Ohm's law in weak electrolytes," Journal of Chemical Physics, 2, 599-615, (1934)

[11]. P. Langevin, "Recherches sur les Gaz Ionises," Annales Chimie et Physique 28, p. 223 (1903)

Chapter 7.

Suzen-Huang model for DBD actuators

7.1 Introduction

As the natural flow of many fluids (air, water etc.) in environment has been exploited in many ways to support human causes, similarly, controlling the flow in natural or artificial scenarios has many practical applications associated with it. Especially in aerodynamics related industries, controlling or modifying the flow over a body has been of prime concern for engineers. Actuators present themselves as devices to assist in modifying the flow in different conditions. In past 15 years, the plasma-based actuators have attracted a lot of research where these actuators have been investigated as prime candidates for flow control applications in air.

A plasma-based actuator typically has two electrodes separated by a dielectric material. One electrode is generally exposed to air and the other one is grounded, Fig. 7.1.1. This configuration is known as single dielectric barrier discharge (SDBD) plasma actuator [1,2]. The dielectric material between the electrodes serves as a protecting slab against unwanted sparks. When supplied with electrical voltage, an electric field is generated between the two electrodes, which if strong enough can produce electrical discharge in surrounding air. Inside the electric discharge ions are produced by the mechanisms such as ionization, recombination, attachment, detachment, photoionization etc. which occur at picosecond time scales [2,9]. The induced electric force makes these charge species to drift under the influence of the electric field according to their polarity. While in motion, some momentum of these ions is transferred to the neutral molecules due to collisions among them and after a certain time the whole fluid surrounding the HV electrode is set into motion. This produced ionic flow phenomenon is termed as *ionic* or *electric wind*.

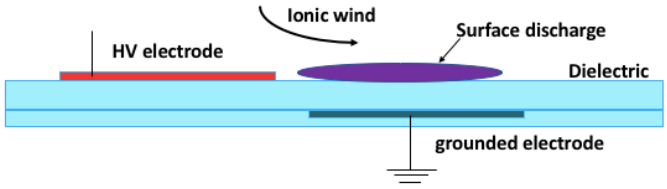


Fig. 7.1.1 A typical sketch of single dielectric barrier discharge plasma actuator

Generally, the induced electric force is such that an ionic jet flow is produced over the dielectric surface just above the grounded electrode due to the discharge phenomenon in an SDBD, Fig. 7.1.1. Because of this jet flow, the air above the right edge of the HV electrode is pulled along the dielectric wall with a significant velocity of a few m/s. This jet can be utilized to modify the overall

flow along the wall surface. Mainly, in flow control applications, this actuator is placed near the flow separation point and then the separating flow can be re-attached by the suction effect created by the DBD actuator. Considering this fundamental phenomenon with this device, several experimental [2,6] and numerical [4,5,8,10] studies with several applications have been published to explore the characteristics of DBD actuators. Moreau (2007) and Benard et al. (2014) provided a detailed review of experimental investigations done in last 15 years, similarly, a modest review of numerical studies is given by Corke et al. 2010.

These actuators place two directions for investigation in front of us: 1) study of plasma physics involved with the phenomenon, or 2) explore the practical uses with different engineering applications. Several physicists have investigated the ionic wind phenomenon numerically with one, two or multiple chemical species inside plasma [14,15,30]. More recent numerical studies used three species (electrons, positive and negative ions) and they have shown the mechanism by solving the transport equations of these species [31-34]. The numerical models which involve the computations of charge density distribution with time are termed as the self-sufficient models. Several numerical models have been proposed based on the nature of plasma phenomena [1,14,22], and in this chapter we mainly deal with the Suzen-Huang model [5].

Seth et al. (2018) provides a good review of most of the initial studies which were carried out with Suzen-Huang (SH) model. These investigations were based on the various updates [13,20,22] in the model and varied applications which showed promising results with this model [7,12,24]. SH model is basically a simple engineering model which does not incorporate the full plasma physics involved in DBD actuators. However, it approximates the charge density based on experimental results and works out the induced Coulomb force by the DBD actuator. This Coulomb force is used as the source term in Navier-Stokes equations to simulate the flow conditions. Most recently, Mahfoze et al. (2017) used two variants of SH model to study skin-friction drag reduction in a channel flow with streamwise-aligned actuators. Simplicity in implementation and low computational costs of this model have allowed the community to explore the suitability of this model in many applications. Here, SH model was also implemented in Oracle3D and we carried out a parametric study with some flow control analysis with this model, which will be presented in this chapter.

7.2 Mathematical model

SH model is considered to mimic a plasma discharge induced by a DBD actuator. The mean EHD force per unit volume is computed with SH model simulations to gain insight on the authority of DBD actuators. This force is added as a source term in Navier-Stokes equations and the impact on fluid is analyzed. We present in this section the mathematical description of the SH model derivation which is also provided in [1,4,8,13].

Electrical potential Φ can be split in two parts for the weakly ionized gas particles [1]. One is the potential due to the external electric field (ϕ), and the other part is induced by the net charge density in the plasma (φ) :

$$\Phi = \phi + \varphi \tag{7.1}$$

Debye length (λ_D) and the net charge density (ρ_c) on the dielectric wall are assumed to be small. It is considered that the distribution of charged particles mainly depends on the electric charge density near the wall. The external electric field conditions do not significantly impact on the distribution of charge species in the domain [1]. With these statements, we arrive at the two major SH model equations:

$$\nabla.\ (\varepsilon_r \nabla \phi) = 0 \tag{7.2}$$

$$\nabla.\ (\varepsilon_r \nabla\ \varphi) = -(\rho_c\ /\varepsilon_0) \tag{7.3}$$

For the derivation of equation (7.3) a case is considered where only positive ions and electrons are significantly present in discharge (e.g. discharge in noble gases). The derivation can be generalized easily for other types of gases also. Boltzmann relation for electron and positive ion densities in thermal equilibrium as given by eqs. (7.4) and (7.5), is used to approximate the net charge density ρ_c within the plasma. Applying Boltzmann expressions in equation (7.6) the exponential form of the charge density is obtained as eq. (7.7).

$$n_e = n_o \exp(e\ \varphi / kT_e) \tag{7.4}$$

$$n_i = n_o \exp(-e\ \varphi / kT_i) \tag{7.5}$$

$$\rho_c = e(n_i - n_e) \tag{7.6}$$

$$\rho_c = e n_o [\exp(-e\ \varphi / kT_i) - \exp(e\ \varphi / kT_e)] \tag{7.7}$$

Here n_i and n_e are the charge densities (number of particles per unit volume) of positive ions and electrons respectively. The overall plasma density is given by $n_o = n_i - n_e$. Boltzmann's constant is $k = 1.38 \times 10^{-23}\ m^2 kg s^{-2} K^{-1}$; T is temperature of respective charge particles in Kelvin and $e = 1.6 \times 10^{-19}\ C$ is the charge of an electron. Expanding the exponential functions in eq. (7.7) with Taylor series, assuming $e\ \varphi \ll k\ T$, we get

$$\rho_c = -e^2\ n_o\ [1/kT_i + 1/kT_e\ ]\ \varphi \tag{7.8}$$

The Debye length (λ_D) is introduced here, which is the distance over which the electric field is shielded out from a charged particle in plasma [37,38]. For case the Debye length is expressed by:

$$1/\lambda_D^2 = \frac{e^2\ n_o}{\varepsilon_0}\ [1/kT_i + 1/kT_e\ ] \tag{7.9}$$

The electric potential due to charge density in terms of the Debye length is obtained by combining equation (7.8) and (7.9):

$$\varphi = (-\rho_c \lambda_D^2 \,/\, \varepsilon_o) \tag{7.10}$$

Finally, replacing φ in eq. (7.3) by eq. (7.10) we obtain the SH model equation for charge density distribution, which is:

$$\nabla.\ (\varepsilon_r \nabla\, \rho_c\,) = {}^{\rho_c}/_{\lambda_D^2} \tag{7.11}$$

Here ε_r and ε_0 are the relative permittivity of the dielectric material and the permittivity of vacuum respectively. Neglecting the time variation of the magnetic field in case of plasma actuators, the external electric field is calculated as the gradient of the applied voltage and given by:

$$\vec{E} = -\nabla\phi \tag{7.12}$$

Thus, the Coulomb force acting on the charge species becomes:

$$\overrightarrow{F_e} = \rho_c \vec{E} \tag{7.13}$$

Incompressible Navier-Stokes equations are solved with $\overrightarrow{F_e} = \rho_c \vec{E}$ acting as the source term which induces the flow motion.

$$\nabla.(\vec{u}) = 0. \tag{7.14}$$

$$\rho\left(\frac{\partial \vec{u}}{\partial t} + \nabla.(\vec{u}\,\vec{u})\right) = -\nabla p + \nabla.(\mu\,(\nabla\vec{u} + (\nabla\vec{u})^T) + \ \rho_c \overrightarrow{E}. \tag{7.15}$$

where ρ is the fluid density, μ the dynamic viscosity, $\vec{u}$ the velocity and p the pressure. We decided to work with the non-dimensional equations to keep the universality in the description of the problem. We introduced the following scales for all the variables to transform the equations into non-dimensional ones.

$$x, y, \lambda_D \ \propto L \qquad u \propto u_0 \qquad p \propto \ \tfrac{1}{2}\,\rho\, u_0^2 \qquad t \propto L/u_0$$
$$\phi \propto \phi_{max} \qquad \rho_c \propto \rho_c^{max}$$
$$\vec{E} \propto \phi_{max}\,/L\ .$$

ϕ_{max} is the amplitude of applied voltage, and ρ_c^{max} is the maximum experimental charge density obtained. We take u_0 as the reference velocity of the flow, which can be the maximum ionic wind velocity induced in such flows, and L is the reference length for the problem. Two dimensionless numbers arise with these formulations:

$$Rey = \rho\, u_0\, L/\mu \qquad D_c = \ \rho_{c_{max}}\, \phi_{max}/\rho\, u_0^2 \tag{7.16}$$

Rey is the flow Reynolds number and D_c represents the non-dimensional number corresponding to Coulomb force. The non-dimensional set of equations is written as:

$$\nabla.\vec{u} = 0. \tag{7.17}$$

$$\frac{\partial \vec{u}}{\partial t} + \nabla.(\vec{u}\,\vec{u}) = -\nabla p + \frac{1}{Rey}\nabla.(\nabla\vec{u} + (\nabla\vec{u})^T) + D_c\rho_c\vec{E}\,f^2(t) \tag{7.18}$$

$$\nabla.\,(\varepsilon_r\nabla\phi) = 0 \tag{7.19}$$

$$\vec{E} = -\nabla\phi\,. \tag{7.20}$$

$$\nabla.\,(\varepsilon_r\nabla\,\rho_c\,) = {}^{\rho_c}/_{\lambda_D^2} \tag{7.21}$$

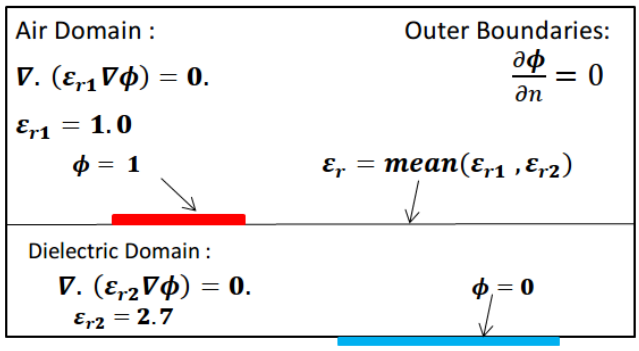


Fig. 7.2.1. Boundary conditions for the electric potential

In SH model, the steady charge density and the steady electric field are computed with eqs. 7.11 and 7.12 only once at the beginning, and the waveform of applied voltage is not considered in electrostatic computations. The fluctuations due to the waveform of the applied AC voltage and charge density are taken into account directly in the non-dimensional Coulomb force term in eq. (7.18) by multiplying it with $f^2(t)$. In SH model, the charge density is also considered to fluctuate with the same waveform as of the applied voltage [1]. The function $f(t)$ depends on the waveform of the applied high voltage. For example, if the applied voltage is given by $\phi = \phi_{max}\sin(2\pi t)$, then for this case $f(t) = \sin(2\pi t)$.

Initially, quiescent air is considered in flow domain. Initial velocities, electrical potential and charge densities are set to zero. No slip boundary conditions are set for velocities on dielectric surface, exposed electrode and all walls. Fig. 7.2.1 and Fig. 7.2.2 describe the boundary conditions for electrical potential and charge density. Dirichlet boundary conditions for non-dimensional electric potential for air exposed electrode (red in Fig. 7.2.1) is fixed to 1 and the grounded electrode (blue in Fig. 7.2.1) is fixed to 0. A half Gaussian distribution is set over the dielectric surface just above the grounded electrode as the charge density boundary condition, given by eq. (7.23).

$$\rho_c(x,t) = \rho_c^{max}\,G(x)f(t) \tag{7.22}$$

$$G(x) = exp\,[-(x-x_o)^2/(2\sigma^2)] \tag{7.23}$$

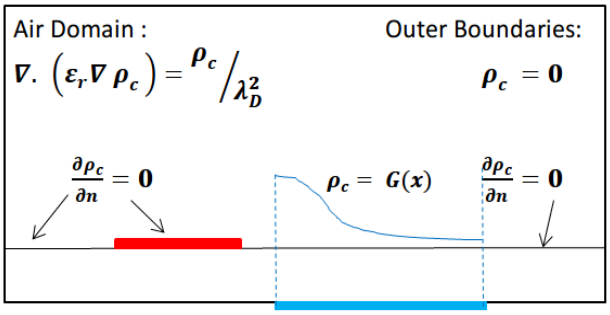


Fig. 7.2.2 Charge density boundary conditions on all surfaces

Here x_o represents the location of maximum charge in x direction and σ is the rate of decay of charge with distance. This half Gaussian distribution of charge is based on some experimental observations and is used in standard SH model [1]. It is a weakness of the SH model that we need to know the charge density distribution a priori to set as a boundary condition. And, as this charge density distribution can change with different actuator configurations, we need to calibrate this model with different actuator configurations. Maximum charge density (ρ_c^{max}) and the Debye length (λ_D) are the two scalar parameters of the SH model which depend on the experimental conditions. The numerical overall charge density distribution in SH model depends on these two parameters.

7.3 Initial investigations with SH model

In the beginning, we started with a validation of our implementation of SH model in Oracle3D. We simulated the same problem as described in Suzen et al. (2005), with same input parameters. The geometrical configuration of the DBD actuator are electrode width (10mm), gap between electrodes (0.5mm), height of electrodes (0.25mm), and thickness of dielectric is 0.125mm between the grounded electrode and the outer surface, Fig. 7.3.1. Quiescent air in the domain was considered to analyze the effect of actuator generated electric force with SH model. Other parameters used for this study were:

Maximum charge density	ρ_c^{max} = 0.0008 C/m^3
Amplitude of AC potential	Φ^{max} =5.0 kV
Frequency of applied AC voltage	f = 4.5 kHz
The Debye Length	λ_D = 0.001 m
Air density	ρ = 1.225 kg/m^3
Air viscosity	μ = 1.85e^{-5} kg/(m.s)

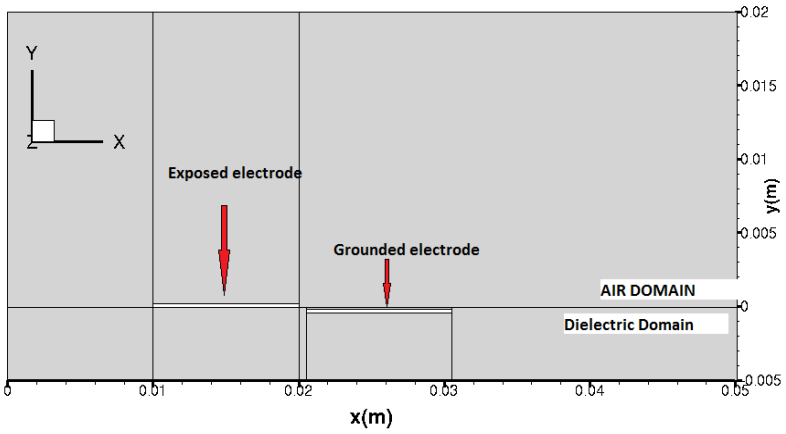
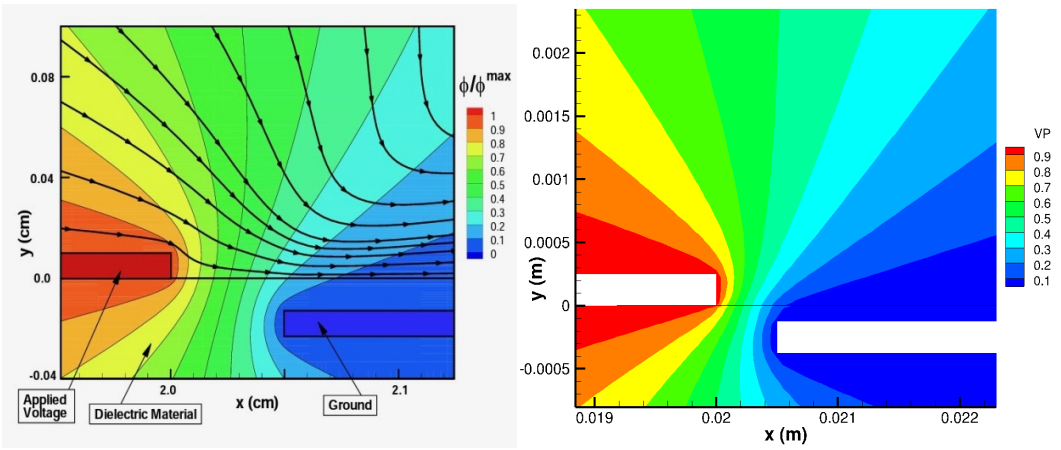


Fig. 7.3.1 The configuaration used for the initial validation test

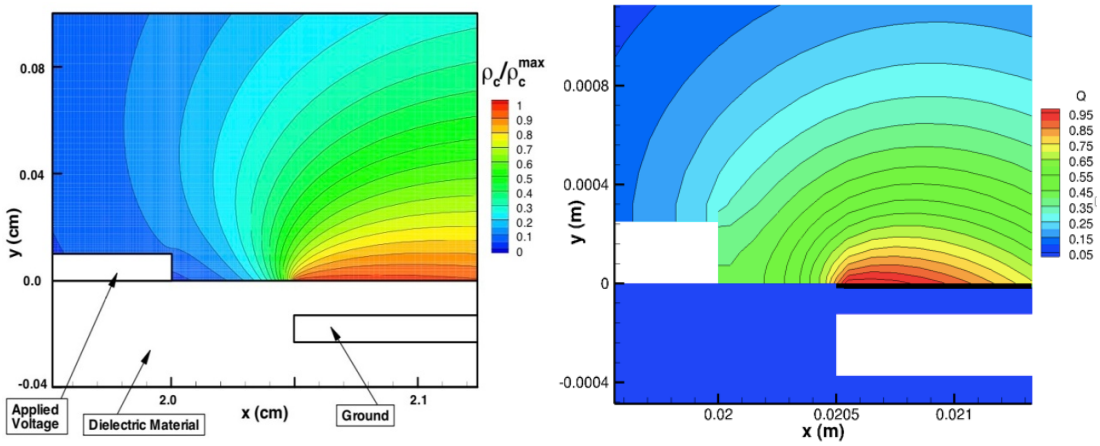
7.3.1 Comparison of electrostatic parameters

We present some qualitative and quantitative comparisons to validate our results with the baseline case reported in [1]. Some qualitative comparison results towards electrical parameters are presented in Figs. 7.3.2-7.3.4. We observe that the 2D contours of the electric potential computed with the Poisson's equation (eq. 7.2) show a very good match with the Suzen et al. (2005) and Comsol results. In the figure of electric potential, Suzen et al. also provided the streamlines of the flow which show the flow direction due to the induced jet along the dielectric wall. The electric charge density computed with the Gaussian boundary condition (eq. 7.22) also provides a qualitatively comparable match with the Suzen et al. (2005) results, Fig. 7.3.3.



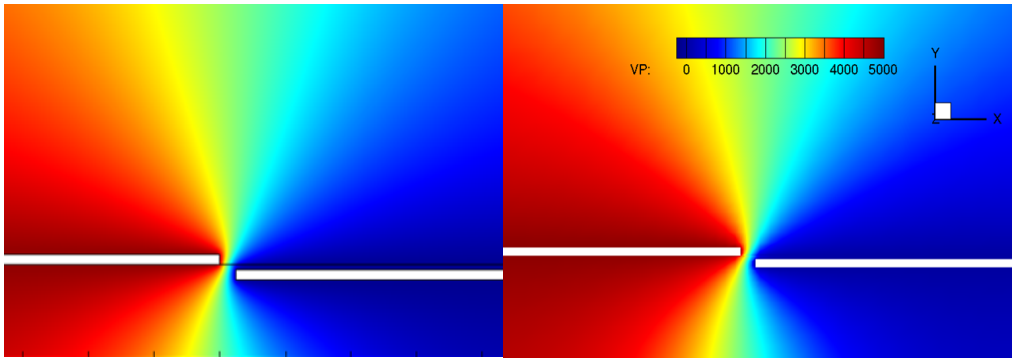
Suzen et al. (2005) Code Oracle

Fig. 7.3.2 Contours of non-dimensional electric potential with flow stream lines.



Suzen et al. (2005) Code Oracle

Fig. 7.3.3 Contours of non-dimensional electric charge density



Comsol Code Oracle

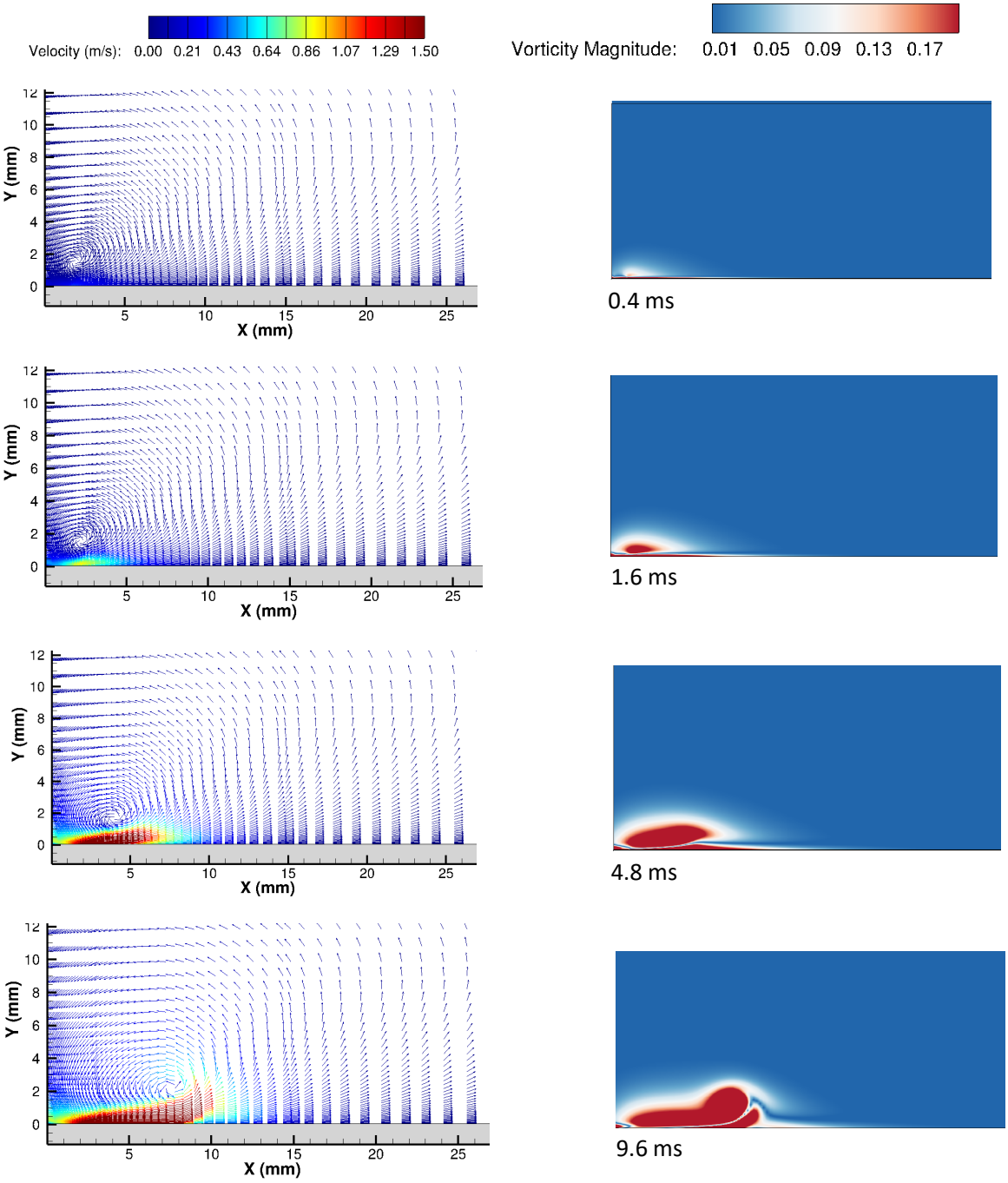
Fig. 7.3.4 Contours of non-dimensional electric potential with Comsol and Oracle

7.3.2 Analysis of the produced ionic wind

An electric field builds in the domain due to the applied voltage to the electrodes (Fig. 7.3.4). This electric field acts on the charge species and an ionic wind is induced in the domain. Formation of a starting vortex by this ionic wind is a prominent flow feature of these types of discharges which has been reported by several researchers (Timothy et al. 2016, Benard et al. 2014 etc.). Here, we present the evolution of this vortex as observed with SH model at initial time steps during the discharge in quiescent air. We show this starting vortex formation with velocity vectors and the corresponding vorticity contours in Fig. 7.3.5. Initially there is no flow in the domain and the air surrounding the actuator is inactive.

As mentioned earlier, the plasma dynamics occur at the time scales of picoseconds. Some early movement of vectors is seen in first snapshot of Fig. 7.3.5 which was taken at 0.4 ms. In this configuration, the right edge of HV electrode (10 mm width) was placed at x=0 and the gap between the electrodes was 1 mm. The grounded electrode width was 20 mm, and the dielectric was 3 mm

thick. This was a general case taken here just to show the evolution of the starting vortex as an effect of the induced ionic wind. We observed the beginning of vortex formation in the gap region between the electrodes within first 2 ms. The gap region between the electrodes experiences high electric force which decreases as we move away from the HV electrode along the dielectric wall just above the grounded electrode. In the third snapshot, taken at 4.8 ms, we see that the ionic jet is starting with increasing velocity and the center of the starting vortex shifting slowly along the dielectric wall in positive x direction.



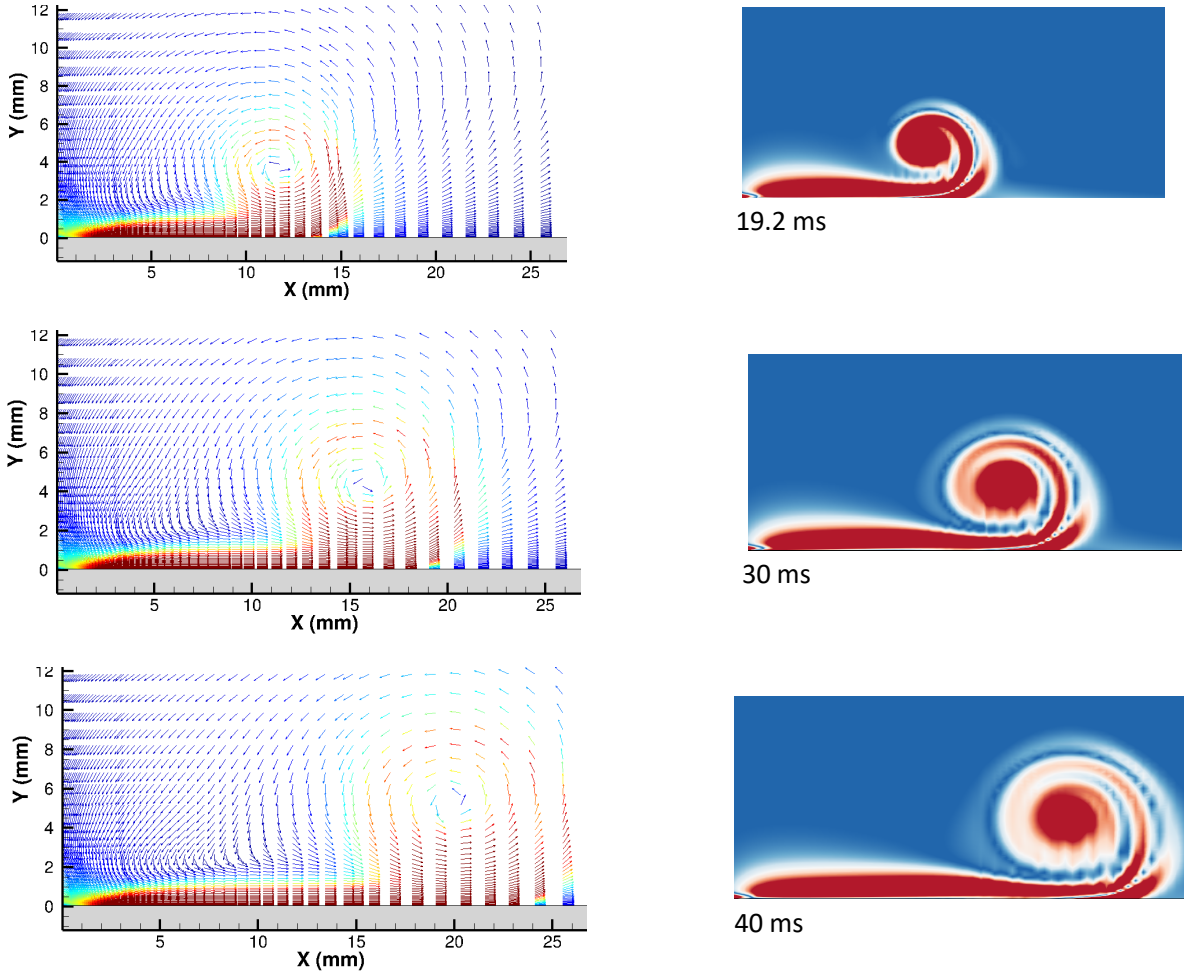


Fig. 7.3.5 Evolution of the starting vortex formed with an SDBD plasma actuator in quiescent air. Velocity vectors on left and the corresponding vorticity contour on right with the time instant

The vortex center has reached around x=12 mm in nearly 20 ms and the extension of ionic jet is seen continuously increasing along the dielectric wall as we advance in time. It should be noted here that there is no other external force except the Coulomb force induced by the discharge which is mainly concentrated in the gap between the electrodes. The vortex is drifting in the surrounding quiescent air with the induced ionic wind only, as there is no other external flow. Snapshot at 19.2 ms clearly shows a well-defined vortex center and the surrounding rotating flow with the vorticity contour.

This case was simulated only till 40 ms and by this time the vortex center has drifted till x=20 mm as seen in the last snapshot of Fig. 7.3.5. The ionic jet thickness has roughly reached a few μm in height and the jet has extended till x= ~25 mm by the end of this simulation. Velocity vectors clearly show that the velocity in the ionic jet is much higher than the surrounding flow which

depends on the input electrical parameters, mainly the applied high voltage. The impact of some geometrical and electrical parameters is reported in section 7.4 of this thesis.

7.3.3 Impact of the Debye length

In general terms, the Debye length is the range of electrostatic effect due to a charged particle in a plasma. It is used as a characteristic property for defining the plasma. In plasma, the electrons will tend to gather near a positively charged ion and this tendency of electrons will shield out the electrostatic field of the positive charge from leaking further into the plasma. Similarly, the positive charge particles will be attracted towards the negative ones. Thus, the individual electric field effect of a charged particle will be confined to a certain distance around that particular particle. This is a shielding effect, and this provides us with a fundamental property of plasma. *The distance over which the electric field is shielded out from a charged particle is termed as the Shielding length or the Debye Length* (λ_D) [50].

The Debye length is a function of the density of charge particles in a plasma and their respective temperatures. Thus, it varies with these physical properties of the plasma. We did not go further into the theoretical details of the Debye length and undertook an analysis with varying Debye length to observe its impact on the charge density distribution and the induced flow in context of the SH model.

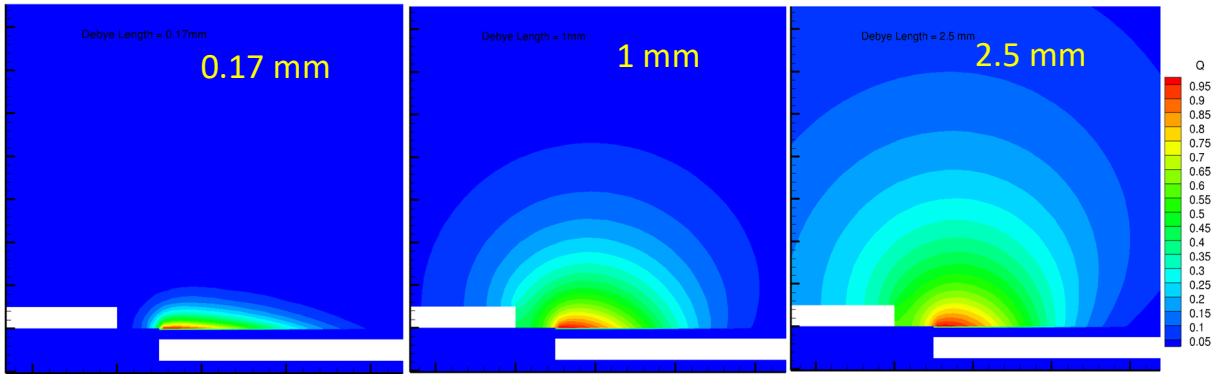


Fig. 7.3.6 Contours of non-dimensional electric charge density with $\lambda_D = 0.17$, *1 and 2.5 mm*

The effect of varying Debye length on charge density distribution and induced flow is reported by Ibrahim et al. (2013) with several values of Debye length [4]. They observed improved *u* velocities (x-components of velocity) with increasing the Debye length in the SH model, but simultaneously the accuracy with *v* velocities (y-component of velocity) was decreased. Physically the Debye length depends on plasma conditions, Laten et al. (2017) used varying Debye lengths in their simulations according to the charge density distribution obtained from experiments [8]. They propose to improve the SH model considering varying Debye lengths through the plasma based on the experimentally measured charge density and use it for multi-encapsulated DBD actuators.

Typically, the theoretical value of Debye length is of the order 10^{-4} m for gas discharge applications at atmospheric conditions as mentioned in [5, 8, 35].

We also observed significant dependence of numerically obtained charge density and flow velocities on the Debye length values with the SH model. Fig. 7.3.6 depicts the charge density contours with different values of Debye lengths. It is observed that with increasing the Debye length, the charge density distribution area is also increased. We chose these values of Debye lengths as these were explored by other researchers also in similar conditions [4,5].

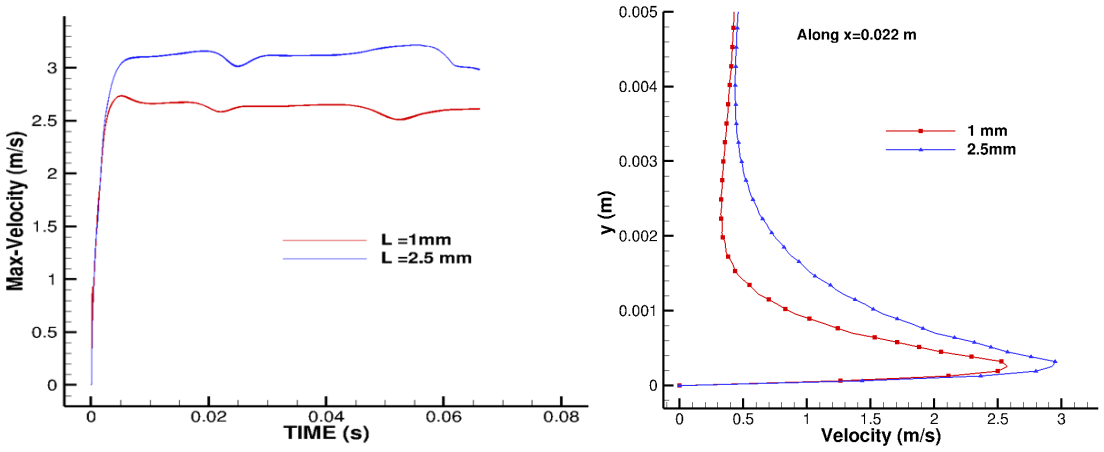


Fig. 7.3.7 a) Evolution of the maximum velocity with time (in whole domain) and b) velocity magnitude at t=37.5ms for λ_D =1 mm & 2.5 mm

The induced maximum ionic wind velocity in the whole domain is directly dependent on the produced electric force. The evolution of maximum velocity in domain was also plotted for two Debye length values, Fig. 7.3.7 (a). The maximum velocity increases from the zero value (quiescent air) and quickly reaches a steady value in both cases. It showed a difference of roughly 0.5 m/s in the steady value between the two cases. Instantaneous velocity magnitude at t=37.5 ms along a section (x = 0.022m) also confirms that the higher value of Debye length induces higher magnitude of ionic jet velocity, Fig. 7.3.7 (b). By the time t=37.5 ms, the maximum velocity has attained a steady value as seen in Fig. 7.3.7 (a). Similar conclusion can be made with Fig. 7.3.8, where the velocity magnitude contour shows a stronger vortex, in terms of velocity values, with the higher value of Debye length.

The Debye length and the maximum charge density are the unknown paramenters for the SH model which are difficult to measure and they vary with the physical conditions of plasma, mainly the temperature [8]. Thus, they are calibrated with experimental measurements to provide a matching numerical value of EHD force. Mahfoze et al. (2017) tested eighth values of λ_D between 1 mm and 4.5 mm to get an optimized value of λ_D to represent the experimental force with desired accuracy. Bouchmal (2011) proposed an approximated linear function of applied voltage to obtain the Debye length, which is also used by Omidi et al. (2017) [52,53]. However, we did not use this formula for

now as more analysis is required in this direction. Thus, there exist several studies each proposing a different method to use and improve the SH model in terms of the Debye length values. We also note that the multiple species plasma models [9,14,15,30] provide an unsteady distribution of charge density in the domain, which could be used to get an approximated EHD force. And, this force can be used as a source term to simulate the flow control applications with simpler models like the SH model. Thus, a combination of these self-sufficient models and SH model can also provide an important insight into such flows. Overall, further research and analysis is required to improve the SH model in the directions mentioned here.

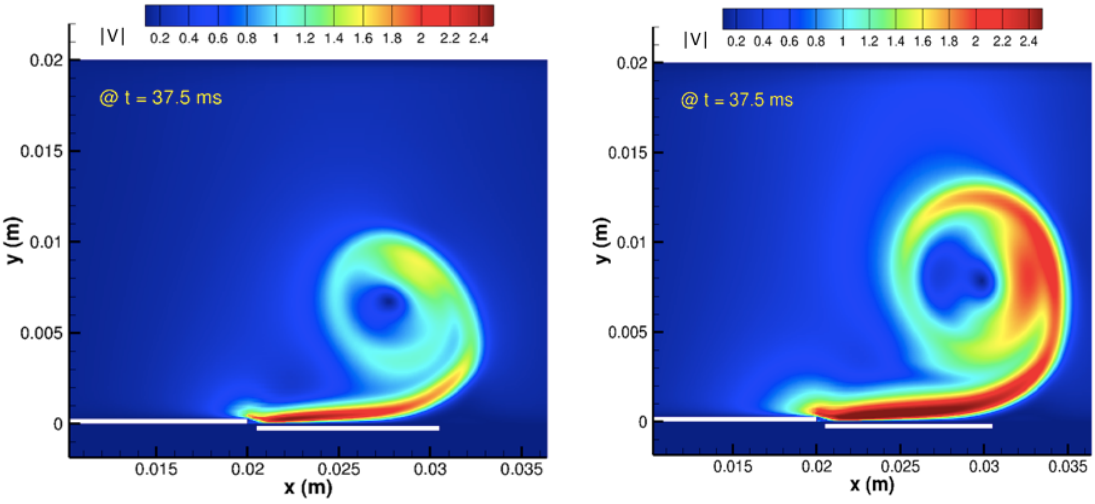


Fig. 7.3.8 Contours of velocity magnitude with $\lambda_D = 1$ mm and 2.5 mm

7.4 A Parametric Study

This section provides a study based on a parametric analysis with some geometrical and electrical parameters which characterize the DBD actuators. The impact of these parameters on the induced body force and the ionic wind velocity was investigated. This study considered thickness of dielectric and gap between the electrodes as important parameters which geometrically configure these actuators. Effect of these parameters was analyzed by varying them for each case and keeping all other parameters constant. Induced maximum velocity in the whole domain and space-averaged EHD force were determined with SH model simulations to compare the individual cases and judge the effect of these parameters on the working of the DBD actuators.

The configuration is depicted in Fig. 7.4.1. This was a multi-block grid, made especially to separate the dielectric and air domain in which we solve different sets of model equations as already mentioned in section 7.2. Mainly, the electric potential is computed in the whole domain, and rest of the equations (NS and charge density distribution) are solved only in the air domain. The domain region in $y > 0$ represents air domain and $y < 0$ is the dielectric slab. The widths of the high voltage and embedded electrodes are respectively 10 mm and 20 mm. The gap between the electrodes, and

the dielectric thickness are pointed out in the sketch, which will vary with cases. The thickness (height) of the air-exposed electrode is 50 μm. Thickness of the electrodes is much smaller than the other geometrical length scales, so infinitely thin height was used for the grounded electrode in our simulations. Thus, the grounded electrode was represented only by the nodes on the boundary surface. Likhanskii et al (2008) considered infinitely thin electrodes citing experimental studies which observed that slightly protruding and buried electrodes have no significant difference in actuator performance [33].

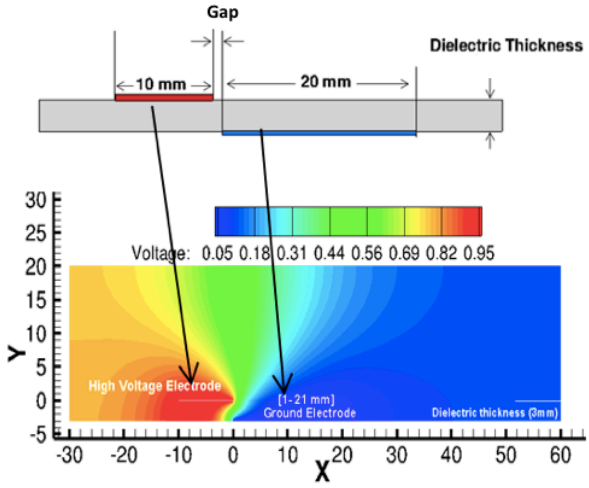


Fig. 7.4.1 Configuration of actuator with sketch of problem domain showing electric potential contours

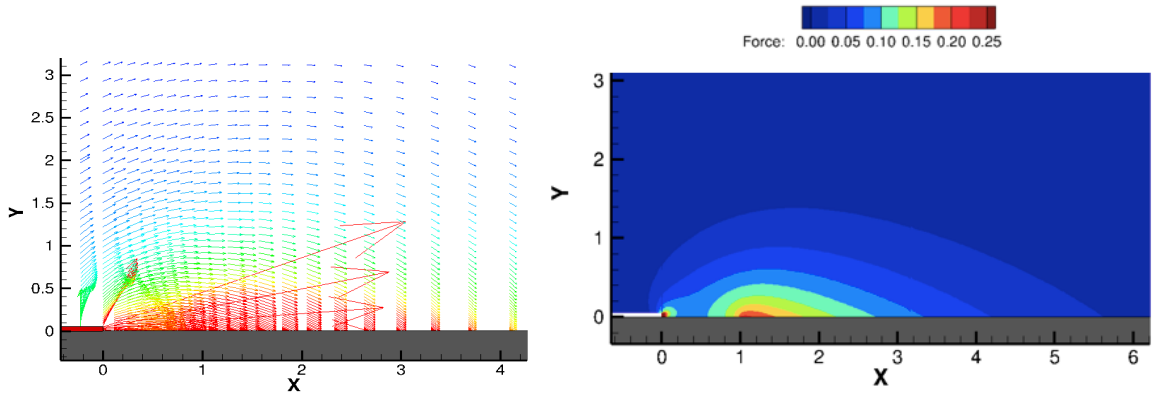


Fig. 7.4.2 Obtained body force vectors and non-dimensional body force contours as induced with such DBD actuator configurations

Quiescent air is considered in the domain, so no external flow exists other than the induced ionic wind flow. Theoretical value of Debye length (10^{-4} m) is taken for all the cases; and $\rho_{c_{max}}$ as 0.001 C/m^3 is considered in this work. Relative permittivity for the dielectric (ε_r) is taken as 2.7 which corresponds to Kapton. Reference velocity is taken as u_0= 5m/s which is nearly the maximum ionic wind velocity attainable with such DBD configurations. The reference length is selected to be L = 1mm, which corresponds to the dimensions of gap between the electrodes in our

configurations. These parameters are set for non-dimensionalization of the problem and ambient conditions are taken for flow parameters.

As mentioned in section 7.1, the SH model solves steady electrostatic equations of electric potential (eq. 7.2) and electric charge (7.11) and it determines an induced steady Coulomb force as given by eq. 7.13. In Fig. 7.4.2, we show the general nature of this body force, induced with such actuator configurations, using the steady force vectors and the non-dimensional steady body force contours.

Force vectors depict the maximum force strength concentrated in small region near the gap between the electrodes and the prominent direction of this force is along the dielectric wall. As a consequence of this force we see the ionic jet along the dielectric wall, as explained in previous sections.

7.4.1 Effect of electrical parameters

Firstly, we investigated the electrical parameters which are the important inputs for the working of DBD actuators. We dealt with two electrical parameters in this study: 1) frequency of voltage signal and 2) the waveform of the voltage signal. Actuator configuration with 3-mm-thick dielectric and 1-mm gap was simulated with four values of applied voltage frequencies (1, 2, 3 and 4 kHz). Benard et al. (2014) reported that DBD actuator configurations having a few mm thick dielectric and operating voltage frequencies around 1 kHz give the best design in terms of higher induced ionic wind velocities and robustness of actuators.

It was observed in experimental studies that the maximum induced velocity increases with voltage frequency up to ~1.2 kHz, and above this frequency the maximum velocity Vs frequency curve starts forming a plateau as provided in [2]. With higher frequencies the charges accumulated on the dielectric wall cannot fully relax in the short duration between two successive discharges, leading to filamentary surface discharges in next cycles which do not contribute to ionic wind production. Thus, the maximum velocity reaches a plateau and a saturation effect is observed with increasing frequency as reported in [2].

In our numerical study also, it was observed that the frequency in the range 1 kHz to 4 kHz brings no significant variation in the induced maximum velocity in whole domain (Fig. 7.4.3 (a)). Fig. 7.4.3 (a) was captured after roughly 100-units of non-dimensional time, by then the maximum induced velocities had reached the corresponding peak values starting from the zero initial values. Experimental studies suggest increasing the applied voltage magnitude for gaining higher ionic wind velocities rather than increasing the voltage frequency. With increasing voltage magnitude, the increment in maximum ionic wind velocities was found to follow a linear profile in our simulations (Fig. 7.4.3 (b)).

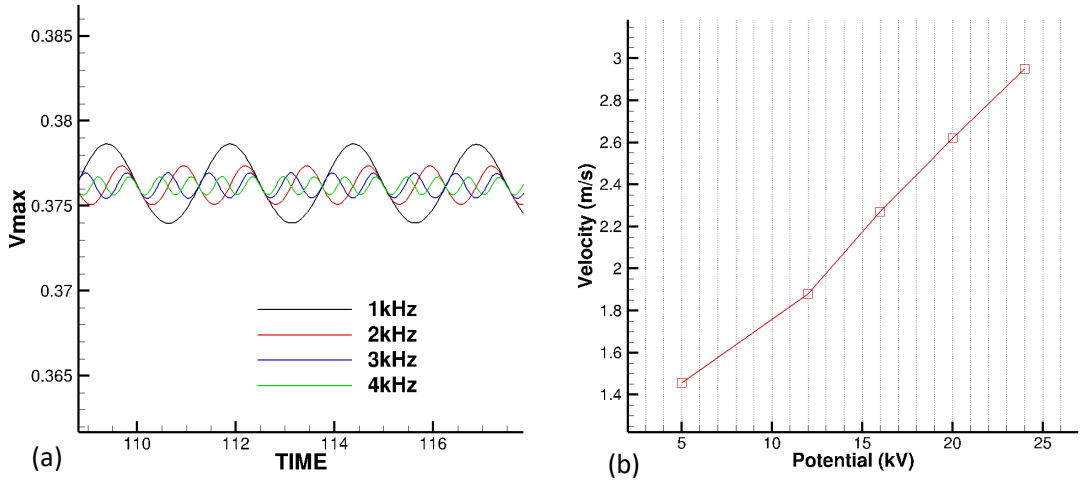


Fig. 7.4.3 a) Evolution of non-dimensional Vmax with varying applied voltage frequency, and b) the effect of increasing voltage magnitude on maximum ionic wind velocities

Impact of voltage waveforms was also analyzed with three waveforms: 1) sinusoidal wave, 2) triangular wave, and 3) square wave. We kept the voltage amplitude same for the three waveforms (20kV) and computed the maximum velocity reached in all three cases. We obtained approximately 3 m/s induced maximum velocity with square wave and ~1.5 m/s with the sinusoidal wave, and the triangular case induced even lower velocity. Our numerical predictions on the magnitudes of induced maximum velocities with different waveforms follow the experimental findings mentioned in [6], however, the square wave form consumes more electrical energy than the sinusoidal voltage.

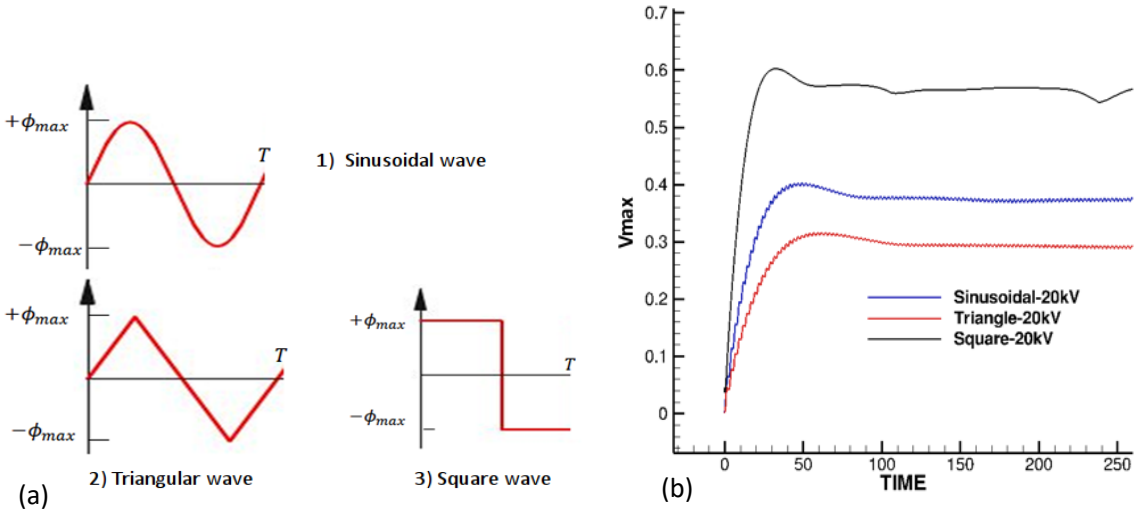


Fig. 7.4.4 a) Studied voltage waveforms, and b) evolution of induced non-dimensional maximum velocity with different voltage waveforms

Benard et al. (2012) reported in their experimental findings that for mean electrical wind production square waveform is optimal which is key for manipulating low speed flows. However, for high speed flow control applications the fluctuating wave forms (sinusoidal etc.) have their own advantages in terms of directly interacting with the natural fluctuations of the flow and manipulating it. In such cases, the voltage frequency and duty-cycles become important parameters [2,6].

Experiments performed in our lab suggest that sine waveform is perhaps more effective than other waveforms in providing unsteady forcing for controlling large scale flow structures. Moreover, for optimal use in terms of induced body force by the electrical power consumption the sinusoidal waveforms are preferred over the square waveforms. We should also note here that with SH model only limited information can be obtained in terms of the effect of voltage waveforms and frequency. In practice, during a single cycle of ac voltage there is the streamer regime in the positive-going cycle, and the glow-like discharge regime in the negative-going cycle as explained in [2,6]. The details of these discharge regimes and consequently the impact on the induced ionic wind velocities are also dependent on the waveforms. SH model, being a phenomenological model, is not a good candidate to investigate the waveform effect with the detailed physics involved.

7.4.2 Effect of dielectric thickness

Dielectric thickness is a geometrical parameter which was studied after electrical parameters. It is previously reported in experimental studies [2] that a thick dielectric (a few mm) makes the actuator robust, and the total induced body force and the resulting ionic wind velocities are also increased. To verify that, we carried out simulations with four different thicknesses (1mm, 3mm, 5mm and 7mm). As shown above (Fig. 7.4.2), the induced EHD force is dominant in x direction along the dielectric wall, so mean EHD force component (f_x) was computed with our cases within a same volume above the dielectric.

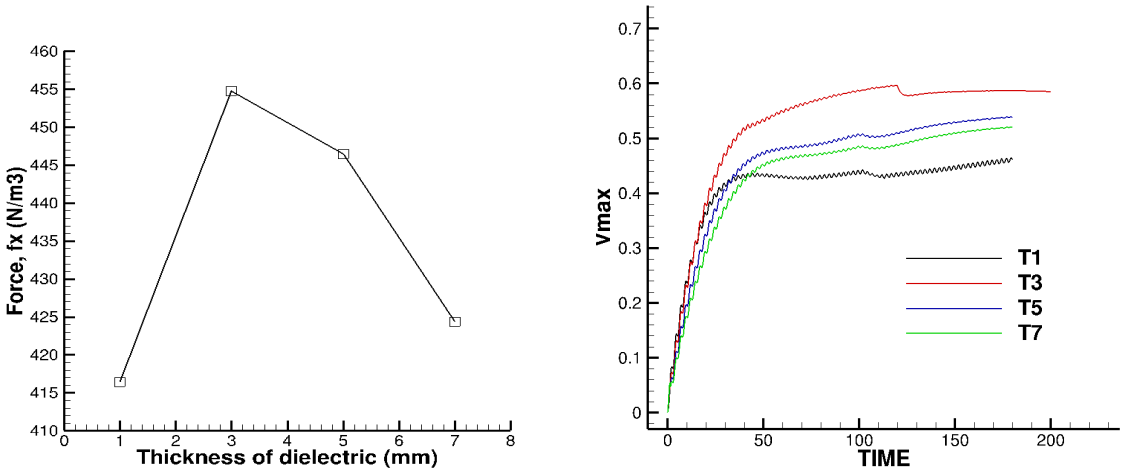


Fig. 7.4.5 a) The mean EHD force component (F_x) with varying thicknesses of dielectric, and b) evolution plot of maximum induced ionic wind velocity in the whole domain

The mean EHD force results show that the actuator configuration with 3mm thick dielectric produced the maximum EHD force. The fall in mean EHD force for the 1 mm thickness case can be attributed to the domain of influence of the electric field. For the 1mm thickness case, the electrodes get closer to each other thus increasing the electric field between them, but the impact of this higher electric field is reduced in a very small region just near the gap between the two electrodes. We observed that the mean EHD force was indeed higher in the small gap area for the 1 mm thickness but as suggested by Fig. 7.4.5 (a), the average force in the whole domain decreased for this 1-mm-thick dielectric case.

The evolution of maximum induced ionic wind velocity in the domain was plotted with the four cases. Fig. 7.4.5 (b) depicts that the velocity increases until reaching a steady value at about 50 to 100 non-dimensional times units. Maximum velocity plot follows the results of mean EHD force computations, and the maximum velocity is obtained with the 3mm thick dielectric configuration. The non-dimensional velocity magnitude of 0.55 will provide a physical velocity of ~ 2.7 m/s (reference velocity $u_0 = 5m/s$), which corresponds well to experimentally found values in [2]. In these cases, the non-dimensional Dc parameter was taken as 0.778 which corresponds to an applied voltage magnitude of ~ 22kV in our conditions. 200 units of non-dimensional time corresponds to 40 ms in our case with the reference time scale set to 0.2×10^{-3} sec. Due to the restart of simulation at 120-time units (case-T3) and at 100-time units (cases- T1, T5, T7), a slight drop in velocity is observed in Fig. 7.4.5 (b) which is not of physical nature.

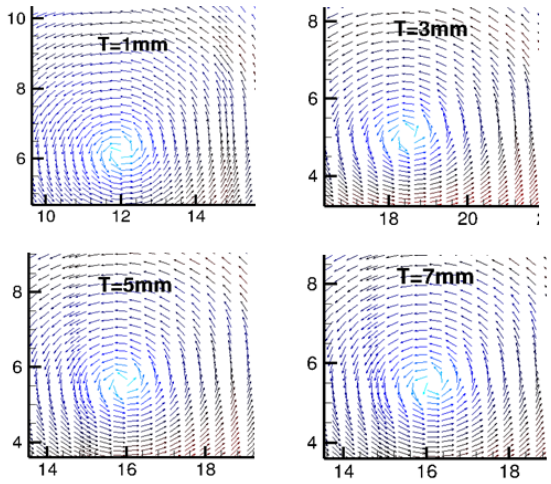


Fig. 7.4.6 Location of the vortex centers with different thickness configurations at 36 ms

The strength of the vortex and the location of its center were also studied as mentioned in reference [5]. For the same instant of time (36 ms), the center of the vortex was found to be at x= ~12, ~18.5, ~16, ~16 for cases T1, T3, T5 and T7 respectively. The vortex had shifted farther downstream of the flow for the 3-mm-thick dielectric than the other cases. Fig. 7.4.6 explains that the fluid above the ground electrode was pushed with greater strength for the 3-mm-thick dielectric, in quiescent air, than with other configurations. One more time, this analysis suggests that the actuator with 3-mm thick dielectric produced higher EHD force and velocities for this particular configuration of actuator.

7.4.2 Effect of the gap between the electrodes

Effect of the gap between the electrodes was studied with four different values (0, 1mm, 3mm and 5mm) of gap. In the previous section, the gap was kept at 1 mm to obtain the optimal thickness of 3 mm. Here, we change the gap from zero to 5 mm to see the influence of the gap, and the thickness of dielectric in these cases is set to 3mm.

Fig. 7.4.7 shows that the mean EHD force is the highest with zero-gap design. Experimental studies have also reported that with a thick dielectric, no gap designs have better performance in terms of induced body force and resulting ionic wind velocity. The induced ionic wind velocity is observed to be maximum with the zero-gap configuration. Although, the induced velocity is not significantly different with a gap of 1 mm but increasing the gap to 3 mm and 5 mm shows a significant decrease in velocity. Analysis of the location of vortex centers also led to same conclusion.

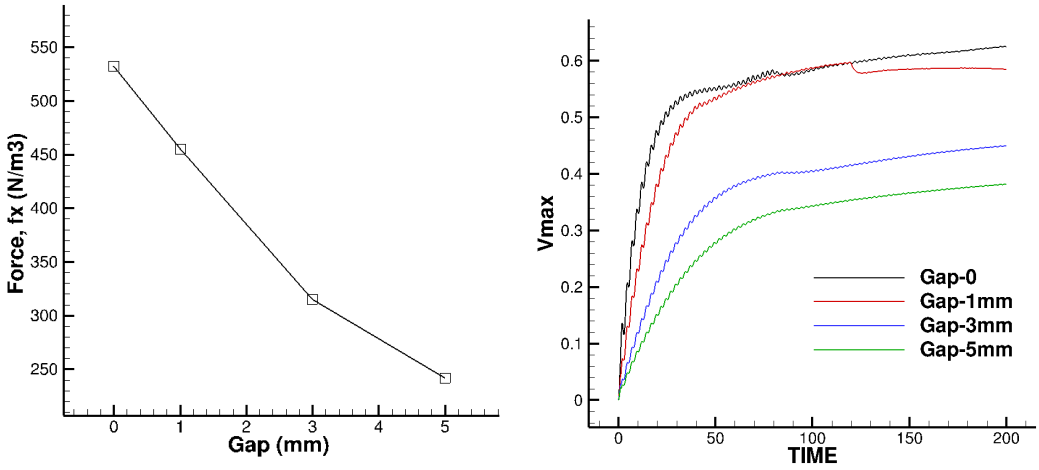


Fig. 7.4.7 a) The mean EHD force with varying gap between the electrodes, and b) evolution plot of maximum induced ionic wind velocity, and,

This parametric study was undertaken to analyze the working of DBD actuators in different electrical and geometrical configurations. The main aim with geometrically different configurations was to get some tendencies in electrically similar conditions. We observed that there is an optimal dielectric thickness for the DBD actuator configurations, above or below which the performance deteriorates. Zero gap between the electrodes was confirmed to induce higher EHD force and higher ionic wind velocities, as suggested in previous experimental studies.

7.5 Analyzing the measured force with SH model: a brief study

Detailed and precise information of EHD volume force produced by the DBD plasma actuators is of great significance, both for optimization of the actuators and improving the numerical modeling of such plasma discharges. Computationally expensive cost of plasma physics models [31-33] limit their use for practical aerodynamic purposes. Phenomenological models like Suzen-Huang model also provide an approximated charge density value during the discharge of plasma, and several studies have been proposed towards the improvements of such models [4,5,8,22]. Obtaining the charge density distribution experimentally in such actuator settings will be very interesting and relevant. However, several experimental studies have reported methods to obtain the time-resolved and steady EHD volume force by using the velocity vector fields obtained experimentally with PIV (particle image velocimetry) [11,42-44].

Kotsonis et al. (2011) firstly proposed a technique to estimate the time averaged EHD body force distribution with the spatio-temporal evolution data of the velocity field from PIV, using the Navier-Stokes equations [44]. Debien et al. (2012) computed time-resolved EHD force with two actuator designs: 1) plate-to-plate and 2) wire-to-plate surface DBD. They used a Navier-Stokes based method and reported strong unsteady nature of the EHD force which showed strong dependence on the shape of active electrode [55]. Benard et al. (2013) characterized the time-dependent topology of the EHD volume force produced by a surface DBD. They used PIV velocity measurement data in a simplified Navier-Stokes solver to compute the unsteady EHD force. They reported the accuracy of their method by comparing with other studies in literature.

In later studies, time-resolved EHD force calculations were reported by Benard et al. (2015). The mean EHD force computed in this study was compared with the force balance measurements to validate their approach. Several values of applied voltage and frequency were studied to provide a spatial distribution of the mean EHD force, which could be used directly in numerical solvers to model such DBD discharges. A reduced order model based on proper orthogonal decomposition was proposed in [43] to more accurately extract the dynamics of the discharge force.

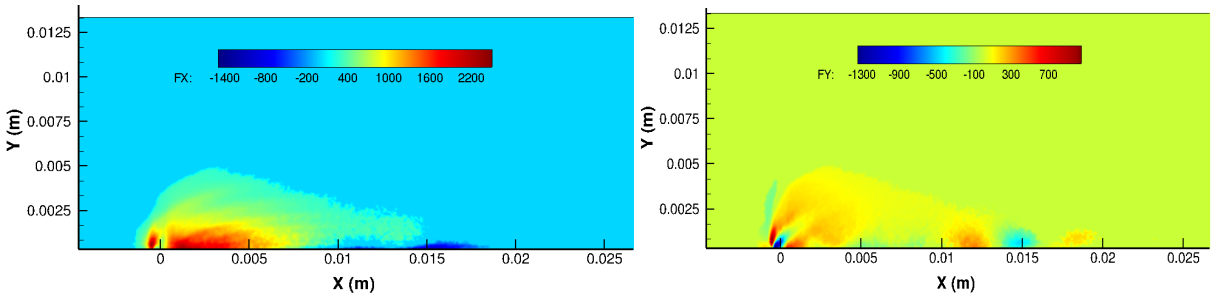


Fig. 7.5.1 Contours of 5% filtered mean EHD force (N/m^3) components a) x-component (F_x), b) y-component (F_y) with 20 kV applied voltage & 1000 Hz frequency

Our study is based on the mean EHD force computed by Benard et al. (2015). The aim here is to regenerate the velocity field numerically, by using the computed mean EHD force as the source term in Navier-Stokes (NS) solver, and then to compare it with the experimentally obtained velocity profiles. The basic idea for computing the mean force with PIV data is that this flow obeys the incompressible 2D Navier-Stokes equations. The steady formulation of equations used to compute the mean force is:

$$\vec{F} - \nabla p = \rho\, \vec{U}.\nabla\vec{U} - \mu\, \nabla^2\vec{U} \tag{7.24}$$

where $\vec{F}$ is the mean force field, p is the pressure field, $\vec{U}$ is the mean velocity field, and ρ and μ are gas density and dynamic viscosity, respectively. The terms on the right-hand side of eq. (7.24) are computed with the PIV velocity field measurements at each desired location. The main assumption considered in this approach is that the pressure contribution in source term is negligible in comparison with the EHD force. With this assumption, it is stated that the right-hand side of the eq. (7.24) corresponds to the EHD body force alone. Thus, the right side of eq. 7.24 is evaluated to obtain $\vec{F}$ with a finite difference solver with 2nd order accurate spatial scheme.

Experimental data generally contains measurement noises, which were controlled in this study by removing the values of force which were below a certain percentage of the maximum computed force. Several percentage values were tried to remove noise. Removal of force values below 5% of maximum force value was found to produce good results. The configuration of the DBD was the same as given in Fig. 7.4.1, where X=0 in our figures corresponds now to the right edge of the high voltage electrode. The input values taken for our case were 20kV voltage and 1000 Hz frequency. The computed mean force components after noise filtration with 5% are shown in Fig. 7.5.1. The direction of overall force is depicted by force vectors in Fig. 7.5.2. It confirms that the force is concentrated in a very small region along the dielectric wall and surrounding the gap between the two electrodes.

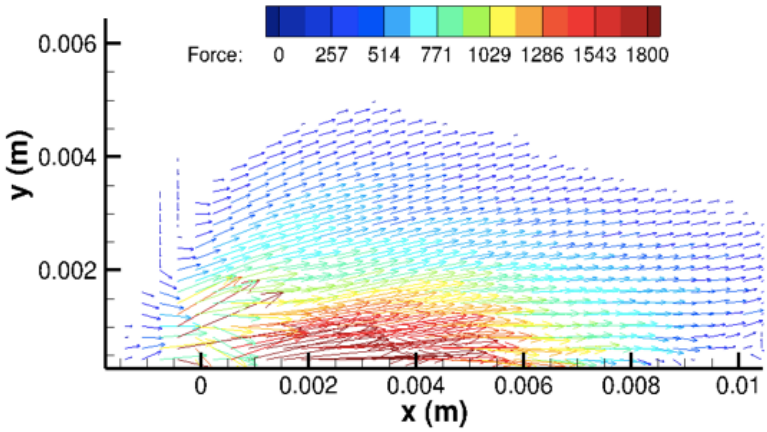


Fig. 7.5.2 Vectors of mean EHD force (N/m^3) (5 % filtered) with 20 kV applied electric potential & 1 kHz frequency

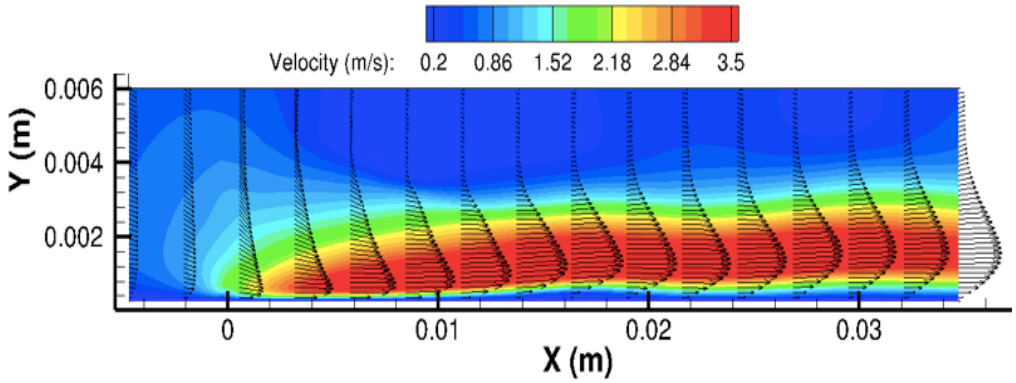


Fig. 7.5.3 Velocity contour with the velocity vectors (20kV, 1kHz)

Numerically obtained velocity vectors with the computed mean EHD force were plotted as shown in Fig. 7.5.3. A jet velocity is clearly visible along the wall. The corresponding plot of experimentally measured velocity vectors is available in ref. [11]. We took 4 sections at locations X=0, 0.001m, 0.01m and 0.02m, as shown in Fig. 7.5.4. Velocity profiles obtained along these sections with PIV and NS solver were plotted together to be compared.

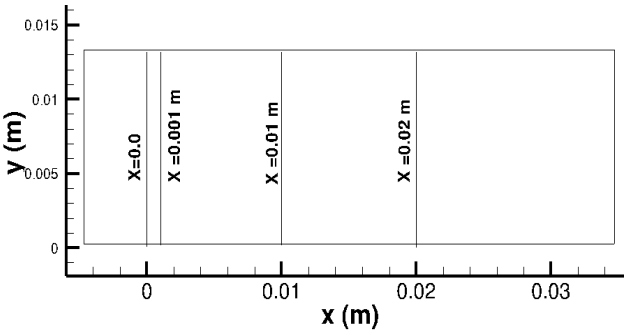


Fig. 7.5.4 Section locations for velocity profiles

The comparison results for all the four locations are provided in Fig. 7.5.5. The first two sections (X=0 and X=0.001m) represent the locations of the right and the left edges of the high voltage and grounded electrodes, respectively, as the gap between the electrodes is 1mm in this configuration. The velocity profiles show a good match within the thickness of the ionic jet, and the thickness of the ionic jet corresponds well with the force vectors as shown above in Fig. 7.5.2. Above the jet thickness (Y > 0.004m) we observe slight variation in PIV and NS results (Fig. 7.5.5 (a,b)). However, the overall shapes of the profiles show a good agreement.

As we go farther downstream from the HV electrode, the velocity in the jet slightly increases as the momentum transfer occurs along the wall and it requires some space to develop fully, as also mentioned in [43]. Thus, the peak velocities are observed slightly away from the HV electrode. Velocity profiles along X=10mm and 20mm are shown in Fig. 7.5.5 (c) and (d). They show that the numerical peak velocity magnitudes are slightly lower than the ones obtained with PIV. Also,

the difference in profiles above the jet thickness is more pronounced as we go away from the HV electrode. However, the thickness of the jet at both the locations are comparable with the PIV results.

The difference in the profiles can be attributed to the assumption of taking pressure gradient as negligible. It was discussed in literature that the pressure gradients are negligible at the beginning of the discharge propagation. However, the impact of pressure gradient increases with time [11, 42-44]. Techniques to obtain precise contribution of the pressure gradients are yet not available. However, most of these studies have demonstrated sufficient agreement with the various other types of force measurement.

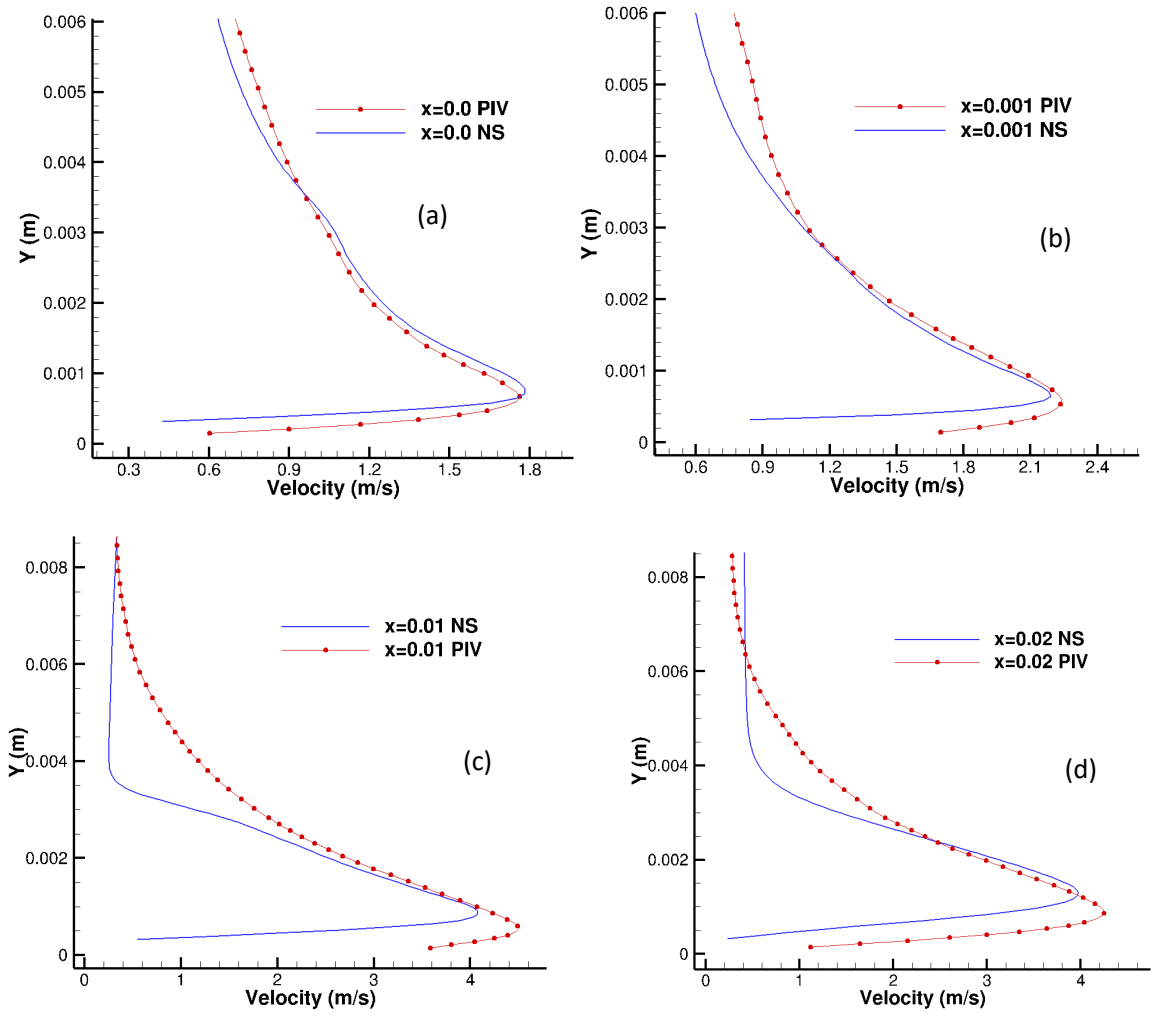


Fig. 7.5.5 Experimental and numerical velocity profiles at a) X=0, b) X=1mm, c) X=10mm, and d) X=20mm

7.6 Backward facing step Flow control

The characteristics of DBD actuators have made them primary candidate in many engineering applications. Possible flexibilities in their design and no moving parts have led to ease in placing these devices in many complex structures like turbine blades for instance [20,24,29]. Backward facing step (BFS) flow has been widely investigated as a classical flow problem arising in many practical situations where flow separation from sharp edges occur (Fig. 7.6.1). Flow control efficiency of DBD actuators have also been investigated with BFS configurations [45-49] in both laminar and turbulent flows. D'Adamo et al. (2014) reported an experimental study on BFS flow control within a transition regime (Re = 1520) by using DBD actuators. They placed the actuator on the vertical step wall to have the velocity jet perpendicular to the flow, in positive y direction, separating from the step corner. They concluded a 37% reduction in recirculation zone with the forcing frequency corresponding to the Kelvin-Helmoltz instability arising from the separation point [45].

Turbulent flow control over the BFS was reported by Pouryoussefi et al. (2014) with their experimental study within the Reynolds number range of 18000 – 54000. They investigated four actuator locations: 1) upstream of separation point, 2) perpendicular arrangement at the tip of step, and 3-4) on the channel bottom wall inside the recirculation bubble. Their results suggested that the actuator located upstream of separation point was the best in controlling the turbulent flows separating at the step edge. Natural frequency of the vortex shedding is considered to be the reference for the actuator frequency to reduce the separation bubble. Pouryoussefi et al. (2014) mentioned that the behavior of the flow in wake was similar with actuator positions 1 and 2, but the actuator at position 2 was less efficient in reducing the recirculation zone.

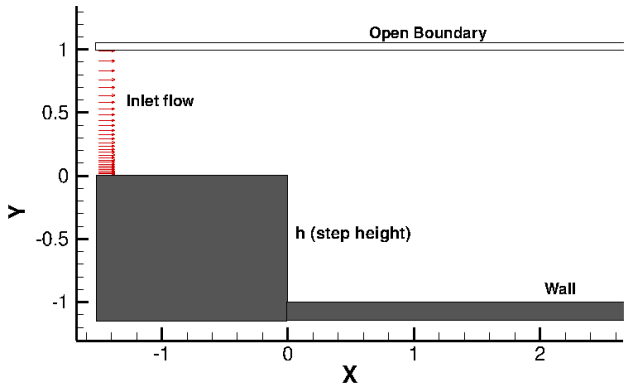


Fig. 7.6.1 Configuration of Backward-facing step

In our laboratory in the context of the EU/China project, MARS, Sujar-Garrido (2014) worked on active control of turbulent flow downstream of a backward facing step using the DBD plasma actuators. Sujar-Garrido et al. (2015) investigated performance of DBD actuators at four locations,

with Reynolds number equal to 30000. They reported that the location upstream of the separation point led to the highest reduction of the reattachment length [47]. Benard et al. (2016) carried out a parametric study involving voltage amplitude, burst frequency and duty cycle of the high voltage with the actuator located upstream of separation point. They conclude a 22% reduction in recirculation length with a frequency mode which would correspond to direct excitation of the natural shear layer instability [48].

In MARS, a numerical study was reported by Peng (2015) in which he used the Suzen-Huang model to obtain the EHD body force for the DBD actuator setting as used in [47,48]. With two hybrid RANS-LES models he simulated the turbulent flow separation control with BFS. In this section, we also aim at simulating the BFS flow separation control with different locations of actuators as shown in Fig. 7.6.2. In configuration A, both the electrodes are perpendicular to each other. The HV electrode is air-exposed and located just upstream the step corner when the grounded electrode is located along the step wall. In configuration B, the HV electrode starts from the corner of the step and extends along the step wall as shown below. The electrode heights are considered infinitely thin as mentioned previously [33,41]. Both of these configurations create an ionic wind jet in negative y direction along the step wall, as indicated by the red arrow in Fig. 7.6.2.

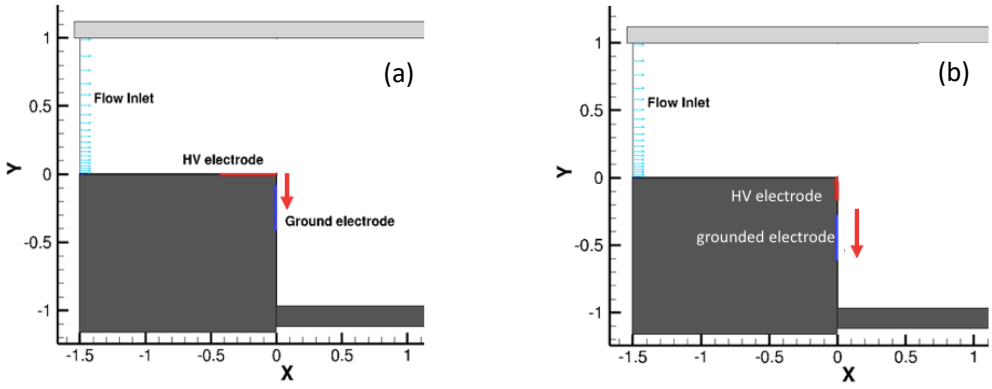


Fig. 7.6.2 Actuator locations, a) configuration A and b) configuration B; red arrow shows the ionic wind jet direction

In numerical analysis we can obtain the electric field vectors with eq. 7.12, and with SH model we have an approximated steady charge density distribution also which is based on experimental studies [1]. With this information we can plot the steady Coulomb force vectors and make an initial guess on how the flow should behave with the obtained force vectors. This simple electrostatic analysis can give some important ideas on where to place the actuators, and look for some other potential locations in complex geometries also.

We carried out a non-dimensional flow analysis with both the configurations. The body force is defined as $\overrightarrow{F_e} = \rho_c \vec{E}$, which gives us the force vectors with the two configurations as shown in Fig. 7.6.3. These force vectors correspond well to our previous understanding of the induced EHD force with DBD actuators, where we have seen the dominant force vectors along the dielectric wall

and concentrated in the gap between the electrodes. For configuration A, the force vectors start at the corner of the step, whereas in configuration B the location of force vectors is slightly shifted downwards according to the placement of the electrodes (Fig. 7.6.3). These force vectors create a suction like effect along the -y direction, as also reported by Pouryoussefi et al. (2014). This suction effect should have a direct impact on the separating flow from the corner of the step.

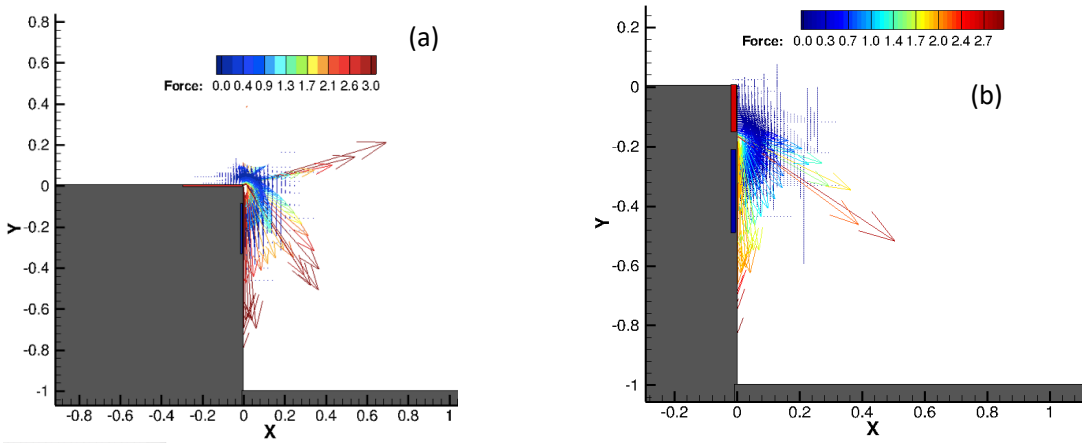


Fig. 7.6.3 Corresponding non-dimensional body force vectors produced with a) configuration A and b) configuration B

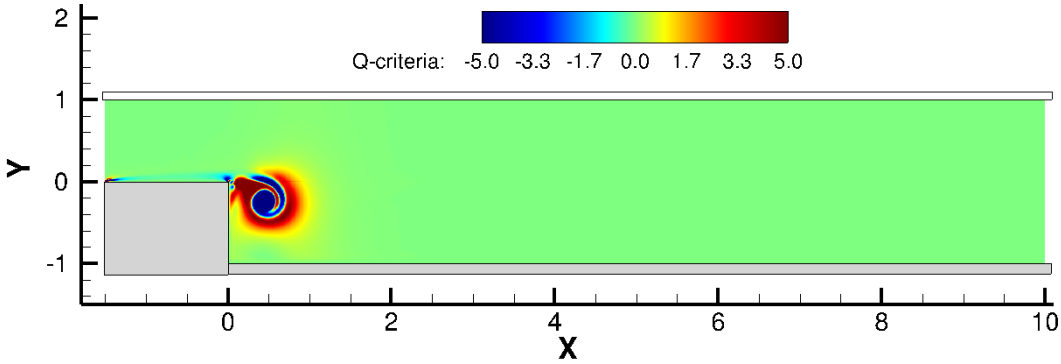


Fig. 7.6.4 Q-criterion contours with Reynolds number = 1000, at t=2.8 (non-dimensional time)

For our channel flow, we took a Reynolds number equal to 1000, based on the height of the step. First, we examined the uncontrolled flow before we switch on the DBD actuator to control the flow. In Fig. 7.6.4 the full non-dimensional domain with the step height equal to 1 unit length, and the domain extension of 10 unit lengths, after the step, is shown. We shall explain the uncontrolled flow features in this configuration with the Q-*criterion* parameter. Fig. 7.6.4 was taken at 2.8 units of non-dimensional time which depicts the first vortex forming from the corner of the step. Vortex is formed due to the velocity shear between two adjacent fluid layers just after the step corner. At

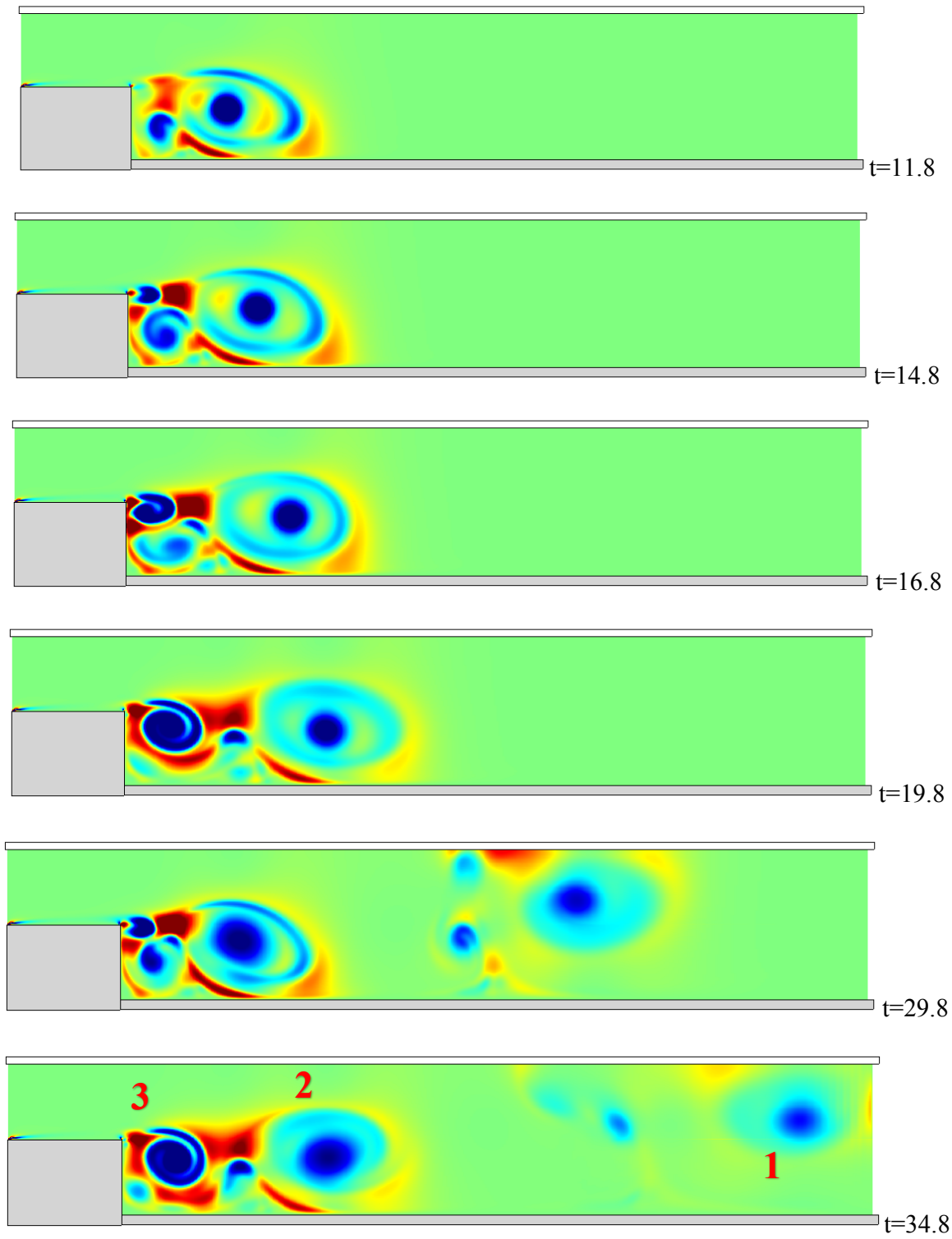
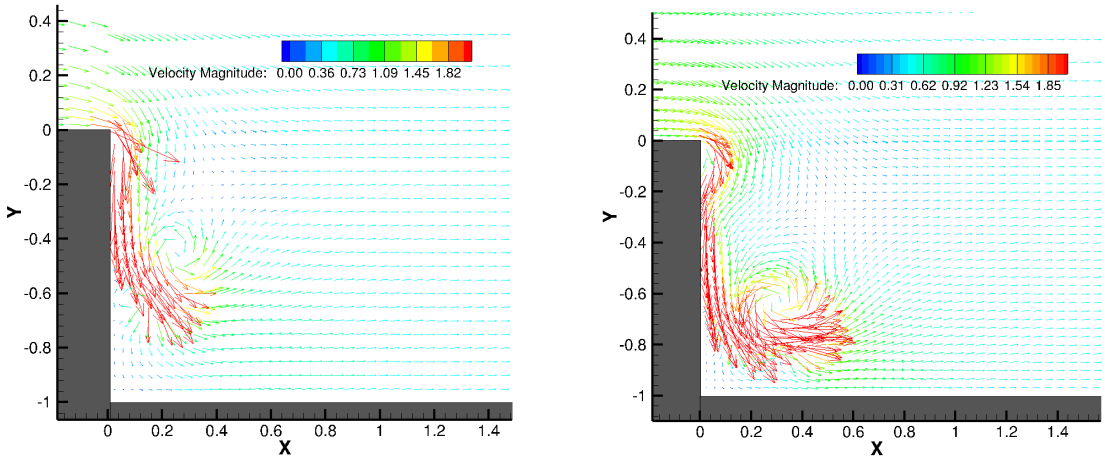


Fig. 7.6.5 Q-criterion snapshots describing evolution of the Kelvin- Helmholtz vortex shedding from the step corner in uncontrolled flow at various non-dimensional time instants

the corner of the step, the boundary layer attached to the wall breaks and the flow starts separating from that point.

The evolution of the flow features after the flow separation is described with some snapshots at various time instants (Fig. 7.6.5). The first vortex moves ahead from the step and its interaction with the bottom wall produces a secondary vortex which moves towards the step wall, as seen in the snapshot taken at t=11.8 and 14.8. However, this secondary vortex is pushed towards the bottom wall by the second vortex forming from the step corner as seen at t=16.8. The second vortex from the step corner grows in size and diminishes this secondary vortex (t=19.8). The second vortex also creates a secondary vortex moving towards step wall, but it is again pushed down by the third vortex, shedding from the step corner, as seen at t=29.8. Thus, the separating flow from the corner of step produces vortical structures forming as typically found in Kelvin-Helmholtz instability, as also mentioned by [47].

The vortex shedding from the corner of the step is clearly visible in the last snapshot taken at t=34.8. At t=34.8, the first vortex has reached near the outlet of the channel and the 2nd vortex is following after it, the 3rd vortex is seen separating from the step corner. So, we have an unsteady flow with periodic vortex shedding from the corner of the step with the flow of 1000 Reynolds number. Now, our objective is to try to control this flow separation from the step corner by placing DBD actuators in the vicinity of the separation point.



Configuration A Configuration B

Fig. 7.6.6 The velocity vectors with both configurations of DBD actuators taken during the flow control simulation at t=0.6 non-dimensional time

In non-dimensional analysis with the SH model, we consider the D_c parameter (eq. 7.16) as a measure of the input electrical power. It is a non-dimensional number given by $D_c = \rho_{c_{max}} \phi_{max} / \rho\, u_0^2$. The value of D_c parameter is proportional to the multiplication of the amplitude of the applied voltage and maximum value of charge generated. We started our flow control simulations with $D_c = 15.0$, which roughly corresponds to 5kV applied voltage magnitude

(ϕ_{max}) and an approximated maximum charge density of 0.001 C/m^3 ($\rho_{c_{max}}$). In SH model, we take an approximate value of maximum charge density which is not known to us, so we can not state the exact values of applied voltage with a given D_c parameter.

We simulated this case with both the actuator configurations (A and B) and found that this low Reynolds flow was modified by both actuator locations; even if configuration A was more efficient than configuration B in terms of reduction of the recirculation zone. It can be explained with the force vectors which show that the impact of body force in configuration A is directly concentrated at the corner of the step where the flow separation starts (Fig. 7.6.3). The consequent velocity vectors during flow control simulations, at t=0.6 non-dimensional time instant, are shown in Fig. 7.6.6. These vector plots confirm the formation of an ionic wind jet along the step wall. With configuration A the ionic wind jet starts right at the corner of the step, however its location is slightly below the corner in -y direction with configuration B.

Here we show the flow control results with configuration A only. Fig. 7.6.7 shows the Q-criterion contours of controlled flow after reaching the steady state. This figure corresponds to a non-dimensional time of 29.8 units. A comparison at this time instant with uncontrolled flow can be made with Fig. 7.6.5, which shows an unsteady vortex shedding flow at this moment. The recirculation zone is evidently reduced and the vortex shedding from the step corner has ceased. We used a square waveform of the voltage here to provide a steady fixed magnitude body force. The purpose was to analyse the working of the SH model in our code with BFS flow control cases and verify if this actuator location has any significant impact at least on laminar flows. A streamlines plot corresponding to Fig. 7.6.7 is provided in Fig. 7.6.8, which also suggests that the vortex shedding is completely absent in this case and the large recirculation zone as seen in uncontrolled flow does not exist after the actuator is switched on.

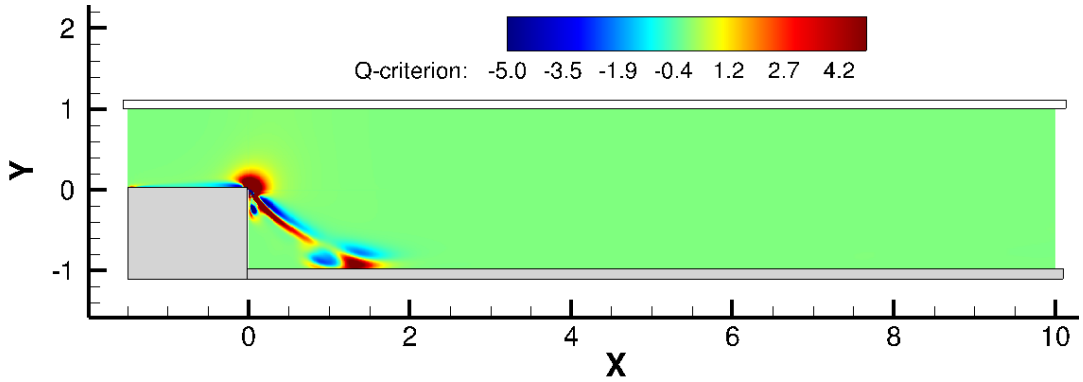


Fig. 7.6.7 Q-criterion contours with controlled flow (D_c=15.0), after reaching steady state

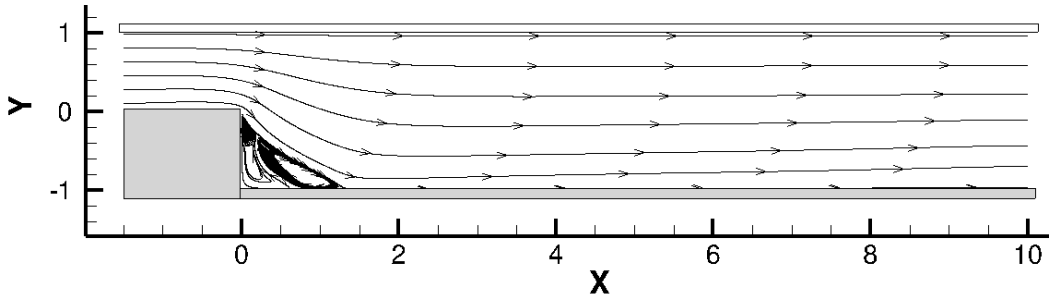


Fig. 7.6.8 Controlled streamlines with D_c=15.0, after reaching the steady state

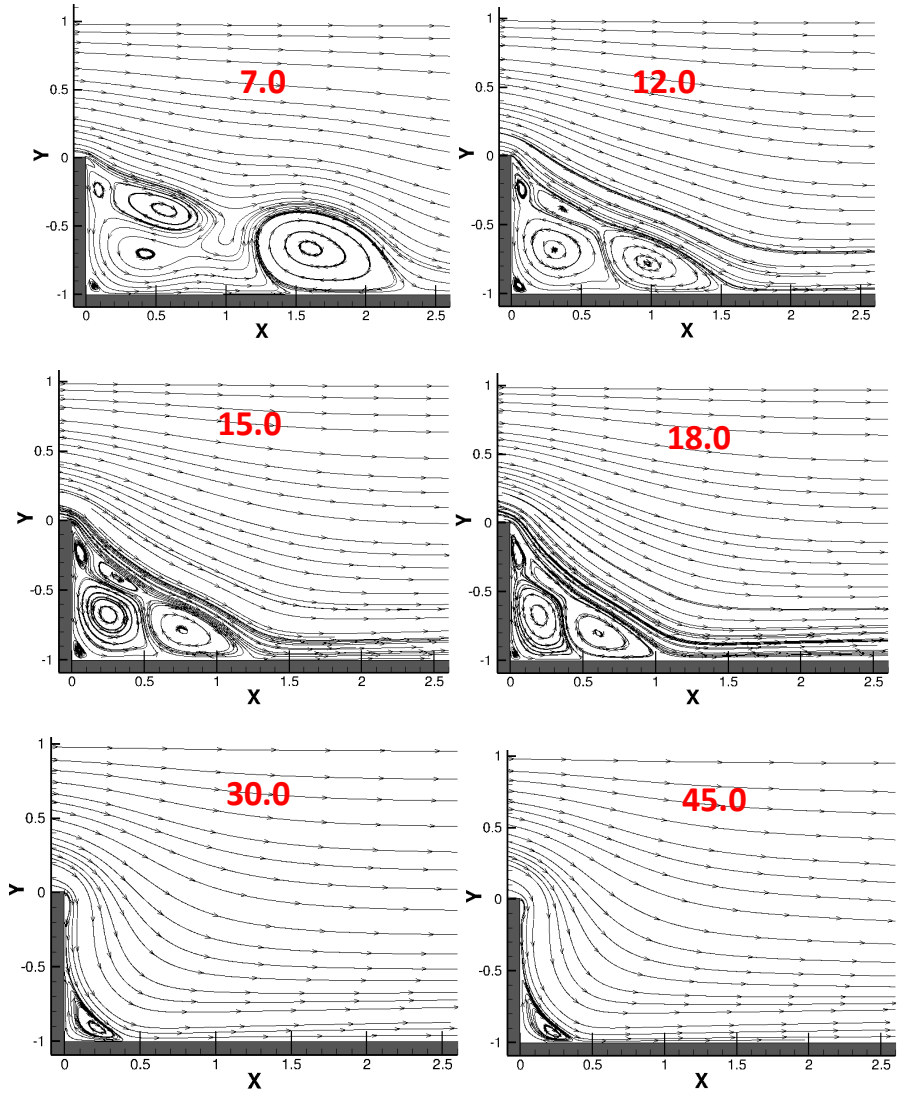


Fig. 7.6.9 Streamlines with controlled flows with increasing D_c parameter (red coloured number)

The impact of increasing and decreasing the D_c parameter was investigated on the same flow. The D_c parameter was varied from 7.0 to 45.0 and the length of recirculation zone was plotted. As seen in Figs. 7.6.9 and 7.6.10, with increasing D_c parameter the re-attachemennt length has decreased from 2.32 (D_c=7.0) to 0.37 (D_c=45.0). Reduction of the re-attachement length is visualized in Fig. 7.6.9 with the steamlines drawn after reaching a steady state in each case. It was observed that after a certain value of D_c parameter, by which the re-attachement length has significantly decreased, increasing D_c further has no impact on re-attachement length as a small corner vortex remains present at the foot of the step. Increament in D_c parameter mainly corresponds to increasing the magnitude of applied voltage and consequently the maximum charge density also, which in effect increase the overall EHD body force.

Experimental analyses suggest that the best location for turbulent flow control in BFS is at the upstream location from the step corner, however, as far as we know no numerical studies have been performed with the two locations we investigated in case of turbulent flows. Our investigation suggests that configuration A has a direct impact on the separation point as the EHD force is concentrated at the corner of the step and a suction effect is created along the step wall. It was showed that a low Reynolds number flow can certainly be controlled with this location of DBD actuator. We believe that this location should also be investigated numerically with turbulent flows to at least analyze its impact on high speed flows, which is not well documented. Time constraint associated with the PhD thesis did not allow us to test these actuator locations with turbulent flows and especially 3D flows.

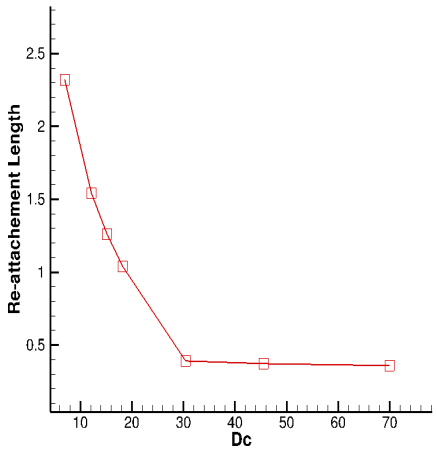


Fig. 7.6.10 Dc Vs re-attachement length (Rey = 1000)

REFERENCES

[1] Y. B. Suzen, P. G. Huang, J. D. Jacob and D. E. Ashpis, "Numerical simulations of plasma-based flow control applications," *AIAA Fluid Dynamics Conference, Toronto, Ontario Canada, 35, 2005*.

[2] N. Benard and E. Moreau, "Electrical and mechanical characteristics of surface AC dielectric barrier discharge plasma actuators applied to airflow control," *Exp Fluids, 55:1846, 2014.*

[3] T.C. Corke, C. L.Enloe and S. P. Wilkinson, "Dielectric Barrier Discharge Plasma Actuators for Flow Control," *Annu. Rev. Fluid Mech.*, 42:505-29 2010.

[4] I. H. Ibrahim and M.Skote, "Effects of the scalar parameters in the Suzen-Huang model on plasma actuator characteristics," *International Journal of Numerica Methods for Heat and Fluid Flow*, August 2013.

[5] T. Brauner, S. Laizet, N. Benard and E. Moreau, "Modelling of Dielectric Barrier Discharge Plasma Acutuators for Direct Numerical Simulatons," *AIAA Flow control conference*, June 2016.

[6] N. Benard and E. Moreau, "Role of the electric waveform supplying a dielectric barrier discharge plasma actuator," *Applied Physics Letters*, 100, 193593, 2012.

[7] M. Sato, K. Okada, H. Aono, K. Asada, A. Yakeno, T. Nonomura and K. Fujii, "LES of Separated-flow Controlled by DBD Plasma Actuator around NACA 0015 over Reynolds Number Range of 104-106," *AIAA Aerospace Sciences Meeting*, Florida. January 2015.

[8] J.B. Laten and R.P. LeBeau, "Improving the Performance of a Plasma Actuator Model for DBD and Multi-Encapsulated Electrode Actutors," *AIAA Aerospace Sciences Meeting, Texas*. January 2017.

[9] J. P. Boeuf, Y. Lagmich and L. C. Pitchford, "Contriution of positive and negative ions to the electrohydrodynamic force in a dielectric barrier discharge plasma actuator operating in air," *Journal of Applied Physics,* 106, 023115. 2009

[10] B. Jayaraman, and W. Shyy, "Modeling of dielectric barrier discharge-induced Fluid Dynamics and Heat Transfer," Progress in Aerospace Sciences, 44, pp 139-191, 2007.

[11] N. Benard, M. Caron and E. Moreau, "Evaluation of the time-resolved EHD force produced by a plasma actuator by particle image velocimetry – a parametric study," *Electrostatics, Journal of Physics: Conference Series* 646 (2015).

[12] B. A. Belson, R. E. Hanson, D. Palmeiro, P. Lavoie, K. Meidell and C. W. Rowley, "Comparison of plasma actuators in simulations and experiments for control of bypass transition," AIAA Aerospace Sciences Meeting, January 2012.

[13] I. H. Ibrahim and M.Skote, "Simulations of the linear plasma synthetic jet actuator utilizing a modified Suzen-Huang model," Physics of Fluids 24, 113602 (2012).

[14] W. Shyy, B. Jayaraman and A. Anderson, "Modeling of glow discharge-induced fluid dynamics," Journal of Applied Physics, Vol 92-Number 11, 2002.

[15] K. Kourtzanidis and L. L. Raja, "Modeling of a Sliding Nanosecond Dielectric Barrier Discharge Actuator for Flow Control," AIAA Aerospace Sciences Meeting, January 2017.

[16] S. Ahn, J. Chae, H. Kim and K. H. Kim, "Simulation of Reduced Air Plasma Reactions for Nanosecond-Pulse Dielectric Barrier Discharge," AIAA Aerospace Sciences Meeting, January 2017.

[17] D. M. Orlov and T. C. Corke, "Numerical Simulations of Aerodynamic Plasma Actuator

Effects," AIAA Aerospace Sciences Meeting, January 2005.
[18] S. Sato and N. Ohnishi, "Influence of Voltage Waveform on Electrohydrodynamic Force in a Dielectric-Barrier-Discharge Plasma Actuator," AIAA Aerospace Sciences Meeting, January 2017.
[19] Y. B. Suzen, P. G. Huang, "Simulations of Flow Separation Control using Plasma Actuators," *AIAA Aerospace Sciences Meeting and Exhibit, Nevada, 2006*
[20] Y. B. Suzen, P. G. Huang and D. E. Ashpis, "Numerical Simulations of Flow Separation Control in Low-Pressure Turbines using Plasma Actuators," *AIAA Aerospace Sciences Meeting and Exhibit, Nevada, 2007*
[21] R. P. LeBeau, D. A. Reasor, Y. B. Suzen, J. D. Jacob and P. G. Huang, "Unstructured Grid Simulations of Flow Separation Control using Plasma Actuators," *AIAA Computational Fluid Dynamics Conference, Miami, 2007.*
[22] I. H. Ibrahim and M.Skote, "Boundary Condition Modifications of the Suzen-Huang Plasma Actutor Model," International Journal of Flow Control, 2011
[23] I. H. Ibrahim and M.Skote, "Simulating plasma actuators in a channel flow configuration by utilizing the modified Suzen-Huang model," Computers & Fluids, 99, 144-155, 2014
[24] M.Skote and I. H. Ibrahim, "Utilizing the L-PSJA for controlling cylindirical wake flow," International Journal of Numerical Methods for Heat and Fluid Flow, Vol. 26 Iss 5, 2016
[25] K. Asada, "Computational Analysis of the Flow Fields Induced by a DBD Plasma Actuator toward Separated-Flow Control,", Ph. D. Thesis, 2014.
[26] M. Sato, K. Okada, T. Nonomura, H. Aono, A. Yakeno, K. Asada, Y. Abe and K. Fujii, "Massive Parametric Study by LES on Separated-flow Control around Airfoil using DBD Plasma Actuator at Reynolds Number 63000," AIAA Fluid Dynamics Conference, 2013
[27] T. C. Corke, M. L. Post and D. M. Orlov, "SDBD plasma enhanced aerodynamics: concepts, optimization and applications," Progress in Aerospace Sciences, Vol. 43, 2007, pp.193-217
[28] M. Sato, K. Asada, T. Nonomura, H. Aono, A. Yakeno, and K. Fujii, "Effective Mechanisms for Turbulent-sepatration Control by DBD Plasma Actuator around NACA0015 at Reynolds Number 1,600,000," AIAA Flow Control Conference 2014.
[29] K. Fujii, "High-performace computing-based exploration of flow control with micro devices," Phil. Trans. R. Soc. A 372:20130326, 2014
[30] C. A. Shi, K. Adamiak and G. S. P. Castle, "Numerical simulation of a DBD Actuator for Airflow Control," ISEHD Conference 2017, Canada
[31] M. Abdollahzadeh, J. C. Pascoa and P. J. Oliveira, "Implementation of the classical plasma-fluid model for simulation of dielectric barrier discharge (DBD) actuators in OpenFOAM," Computers and Fluids, 128, 77-90, 2016
[32] T. Unfer, J. P. Boeuf, F. Rogier and F. Thivet, "Modeling of Dielectric Barrier Discharge and coupling with Computational Fluid Dynamics," AIAA Aerospace Science Meeting and Exhibit, Nevada, 2008
[33] A. V. Likhanskii, M. N. Shneider, S. O. Macheret and R. B. Miles, "Modelling of dielectric barrier discharge plasma actuator in air," J. Appl. Phys. 103, 053305, 2008
[34] H. Nishida, N. Asaumi and Y. Tanaka, "Effect of Aerodynamic Body Curvature on Dielectric-Barrier-Discharge Plasma Actuator," ISEHD Conference 2017, Canada
[35] O. Mahfoze and S. Laizet, "Skin-friction drag reduction in a channel flow with streamwise-aligned plasma actuators," International Journal of Heat and Fluid Flow, 66, pp.83-94, 2017
[36] J. Wu, P. Traore and C. Louste, "An efficient finite volume method for electric field-space

charge coupled problems," Journal of Electrostatics, 71, pp- 319-325, 2013
[37] M. A. Lieberman and A. J. Lichtenberg, "Principles of Plasma Discharges and Materials Processing," , 2nd Edition, ISBN 0-471-72001-1
[38] R. J. Goldston and P. H. Rutherford, "Introduction to Plasma Physics,", ISBN 0-7503-0325-5
[39] F. Moukalled, L. Mangani and M. Darwish, "The Finite Volume Method in Computational Fluid Dynamics," Springer, ISBN 978-3-319-16873-9
[40] P. Traore, Y. M. Ahipo and C. Louste, "A robust and efficient finite volume scheme for the discretization of diffusive flux on extremely skewed meshes in complex geometries." Journal of Computational Physics, 228, 5148–5159, 2009.
[41] U. K. Seth, P. Traore, F. J. Duran-Olivencia, E. Moreau, P. A. Vazquez, "Parametric study of a DBD plasma actuation based on the Suzen-Huang model," Journal of Electrostatics, 93, 1-9 (2018)
[42] N. Benard, A. Debien, E. Moreau, Time-dependent volume force produced by a non-thermal plasma actuator from experimental velocity field," J. Phys. D: Appl. Phys. 46 (2013)
[43] N. Benard, S. Laizet, E. Moreau, "PIV-based dynamic model of EHD volume force produced by a surface dielectric barrier discharge," AIAA Aerospace Sciences meeting, Jan. (2017)
[44] M. Kotsonis, S. Ghaemi, L. Veldhuis, F. Scarano, "Measurement of the body force field of plasma actuators," J. Phys D: Appl. Phys. 44 (2011)
[45] J. D'Adamo, R. Sosa, G. Artana, "Active control of a Backward Facing step flow with plasma actuators," ASME Journal of Fluids Engineering, 136, (2014)
[46] S. G. Pouryoussefi, M. Mirzaei, M. Hajipour, "Experimental study of separation bubble control behind a backward facing step using plasma actuators," Acta Mech 226(4), 1153-1165, (2014)
[47] P. Sujar-Garrido, N. Benard, E. Moreau, J. P. Bonnet, "Dielectric barrier discharge plasma actuator to control turbulent flow downstream of a backward-facing step," Exp. Fluids, 56-70, 2015
[48] N. Benard, P. Sujar-Garrido, J. P. Bonnet, E. Moreau, "Control of the coherent structure dynamics downstream of a backward facing step by DBD plasma actuator," Int. Journal of Heat and Fluid Flow 61, 158-173 (2016)
[49] Shia-Hui Peng, "Shear-Layer manipulation of backward facing step flow with forcing: A numerical study," Swedish Defense Research Agency, FOI, Sweden
[50] M. A. Lieberman, A. J. Lichtenberg, "Principles of Plasma discharges and materials processing," 2nd edition, John Wiley & Sons Publication, 2005
[51] R. J. Goldston, P. J. Rutherford, "Introduction to Plasma Physics," Institute of Physics Publishing, 1995
[52] A. Bouchamal, "Modelling of Dielectric-Barrier Discharge actuator," Master of Science thesis, Delft University of Technology, (2011)
[53] J. Omidi, K. Mazaheri, "Improving the performance of a numerical model to simulate the EHD interaction effect induced by dielectric barrier discharge," Int. Journal of Hear & Fluid Flow, 67, 79-94 (2017)
[54] E. Moreau, "Airflow control by non-thermal plasma actuators," J. Phys. D: Appl. Phys. 40, 605-636 (2007)
[55] A. Debien, N. Benard, L. David, E. Moreau, "Unsteady aspect of the electrohydrodynamic force produced by surface dielectric barrier discharge actuators," Appl. Phys. Lett. 100,

013901 (2012)
[56] P. Sujar-Garrido, “Active control of the turbulent flow downstream of a backward facing step with dielectric barrier discharge plasma actuators,” PhD Thesis, University of Poitiers, (2014)

Conclusions & Perspectives

This thesis work advanced the capability of numerical work carried out within the EFD group at Institut Pprime by successfully upgrading and thoroughly validating the in-house EHD solver, 'Oracle3D'. The solver was parallelized with advanced MPI features and desired scalability up to available 1200 cores was reported. Several 3D simulations concerning EHD phenomena of unipolar injection and electro-conduction were undertaken. Flow control efficacy of DBD plasma actuators was studied with several test cases. In this section, we summarize the important work done during this thesis and also propose directions for further research.

Code 'Oracle3D' had been used for many important EHD studies previously, however, to prepare it for computationally larger and physically more complex problems it was imperative to upgrade it to modern Fortran framework and parallelize it. Being a principle parallelizing standard, which is widely adopted both in industry and research, Message Passing Interface (MPI) was considered as the most suited approach to parallelize Oracle3D for the distributed memory platforms. Advanced MPI features of Cartesian topology and Inter-Communicators were implemented to prepare highly scalable data exchange strategy for multi-block structured grids. Some scalability tests showed super-linear scalabilities which are attributed to favorable cache effects while increasing the core count and due to the advanced algorithms used by the modern Intel processors to operate dynamically at higher than their base frequencies. To benefit the future users of this code, the whole MPI strategy was described in detail in the thesis, which would facilitate the future development of the code also.

A dedicated roadmap was followed to validate the new code with several physical problems, and most of the results are reported to prepare a database for users. Individual physical models: Poisson's equation, Navier-Stokes equations, advective transport equations etc. were tested with independent individual solvers while preparing the whole EHD solver. The dedicated parallel Poisson solver was validated with highly skewed grids with improved deferred correction (IDC) approach using varied number of MPI processes. Working of TVD scheme, periodic boundary conditions, 3D data exchange etc. were rigorously validated in parallel context. Implementation of new features like periodic boundaries, Robin boundaries, non-homogeneous Neumann boundaries, TVD scheme updates etc. are reported with relevant Finite Volume discretization equations in the chapter dedicated to numerical methods.

Unipolar injection problem was studied with the complete EHD solver. Some 2D validation tests were performed to validate the results with unipolar injection module against the already available data in literature. This electro-convection (EC) problem is often studied considering its analogy with the Rayleigh-Benard thermal convection (RBC) problem. In experimental studies, hexagonal convection cell patterns are observed in EC between parallel plate electrodes as also found in RBC. We carried out a three-dimensional convective cell pattern study to numerically reproduce the hexagonal cells in this EC problem. We observed the formation of hexagonal cells

with our solver; however, we found significant impact of Neumann boundary conditions on the evolving convective cells which prohibited stabilization of these hexagonal cells in our studies.

We simulated different cases by changing several parameters like the time step, grid size, initial vertical velocity conditions etc. but it was observed that a zero gradient Neumann boundary was not suitable for stabilizing the convective cells. Different cases were undertaken with grid sizes varying from 2 million cells to 25 million cells, and, with 50 MPI processes to up to 400 MPI processes. The parallel efficiency of the code, in exchanging the data at process interfaces, with this instability dependent flow was as desired.

As a perspective for the future work in this direction, we propose a pattern formation study with our Finite Volume solver with parameters such as the periodic boundaries, diffusion coefficient and the domain size and shape. During our investigations we observed a favorable effect of using hexagonal initial velocity perturbations on simulation time, thus these initial velocity perturbations should be used as an approach to reduce computational time. However, having or not having this initial velocity perturbation did not impact the flow stabilization in our simulations. The 3D flow phenomenon with EC plumes was also studied with blade-plane electrode configuration considering both autonomous and non-autonomous injection laws. Differences observed with different injection laws were highlighted with several flow parameters.

Oracle3D was prepared for the electro-conduction problem with new boundary conditions and this module was validated with some Comsol results. The impact of Robin boundary condition on charge species' distribution was discussed with relevant plots. The Onsager effect was explained with relevant equations and its impact on the electro-conduction was presented in channel case. A three-dimensional channel case was simulated to observe if 3D effects were present with our conduction channel conditions. It was found that our case settings produced a perfect 2D flow as no 3D effects were observed in the channel. It was critical to verify the 2D nature as the two vortices expanding through the entire channel height could produce some 3D effects in narrow channels even though the numerical conditions suggested a laminar flow. A 3D flow introduces higher mixing in the domain which is desirable in many applications.

Some cases with electro-conduction were performed in blade-plane electrode setting. With our case settings (T=100, M=0.2, C=0.1), we observed a reversal of flow direction during the flow evolution from the stationary initial state. Initially the fluid is seen moving from plane to blade electrode, which gets reversed as the flow reaches a steady state. This reversal of flow direction was never observed in previously reported conduction studies with similar blade-plane configurations. This flow did not show any 3D effects with an electric Reynolds number equal to 2500. We observed similar flow features with M=0.1 in 2D simulations, however, this case showed clear 3D features when full 3D simulations were performed. Limited cases could be undertaken during this PhD on electro-conduction; further parametric studies should be considered to better understand the flow features in different conduction configurations.

Suzen-Huang (SH) model was implemented in Oracle3D to explore its performance in simulating the flow with DBD plasma actuator configurations. A parametric study was published based on the geometrical parameters which characterize the functioning of DBD actuators. Impact of the dielectric thickness, the gap between the electrodes, the frequency and the waveform of voltage etc. was described in terms of their effect on the induced maximum velocity and average body force. A brief study was highlighted with the experimental force used as the numerical source term in Navier-Stokes equations to reproduce the experimental velocities numerically. Flow controlling capability of DBD actuators was demonstrated with backward facing step configuration for low Reynolds number flows. It was shown that a laminar flow separation could be drastically controlled by placing the actuator at the tip of the step with both electrodes perpendicular to each other. Overall, we have prepared the parallel code for these DBD actuator simulations, so that full 3D studies could be performed now with different DBD configurations with various Reynolds number flows.

During the course of this thesis, a self-sufficient 3 species plasma discharge model was also incorporated in the parallel code, but due to the time limit exclusive simulations could not be performed. This plasma discharge model was prepared to investigate the detailed underlying physics of discharge mechanism which is not possible with experiments, given the picosecond time scales involved within the discharge at species level and the inability of experiments to measure the dynamics of individual species during the discharge. The plasma discharge model predicts the unsteady charge density distribution of individual species and which is supposed to provide a better approximation for the overall charge distribution and consequently the induced electric force. It is a clear advantage over the SH model which is limited by the requirement of experimental charge values. In fact, the SH model can benefit by the charge distribution predicted with the plasma model, which could be an interesting prospective with Oracle3D. Indeed, this problem is challenging and computationally expensive with numerical simulations also, nevertheless, with the parallel code available the group can advance in this direction now.

Appendix 1

The discretization of Deferred Correction terms for Standard Deferred Correction as used in Oracle3D. We provide here some specific details for the user of the code Oracle3D, towards the implementation of the discretization. This section follows from the same heading in chapter 2. We rewrite the eq. (2.15):

$$\left(\frac{\Delta V}{\lambda_D^2}\right)\varphi_P^m - \sum_{k=newsbt}\frac{\varepsilon_{rk}S_k}{d(P,K)}\left(\varphi_K^m - \varphi_P^m\right) \quad = \quad \left[\sum_{k=newsbt}\varepsilon_{rk}S_k\,(\nabla\varphi)_k^{\,m-1}.(\vec{n}_k - \vec{\xi}_k)\right] \tag{2.15}$$

The deferred correction terms, which were dropped in eq. (2.16) in chapter 2, are approximated and used as source terms in algebraic equations. We write here the RHS term of eq. (2.15) and expand it in two terms as follows:

$$\sum_{k=newsbt}\varepsilon_{rk}S_k\,(\nabla\varphi)_k^{\,m-1}.(\vec{n}_k - \vec{\xi}_k) = \left[\sum_{k=newsbt}\varepsilon_{rk}S_k\,(\nabla\varphi)_k^{\,m-1}.\vec{n}_k - \sum_{k=newsbt}\varepsilon_{rk}S_k\,(\nabla\varphi)_k^{\,m-1}.\vec{\xi}_k\right] \tag{2.21}$$

These two terms on the RHS of eq. (2.21) are stored in variables SUEH and SUEL in Oracle3D. Contributions in these terms from all the cell faces can be obtained similarly as detailed in section 2.2.1 of chapter 2, for the LHS of eq. (2.15). Some code relevant details corresponding these terms are provided in here. These terms are approximated in code as:

$$\varepsilon_{rk}S_k(\nabla\varphi)_k\ .\vec{n}_k = \ \varepsilon_{rk}\left[(\nabla\varphi)_{kx} * S_{kx} + (\nabla\varphi)_{ky} * S_{ky} + (\nabla\varphi)_{kz} * S_{kz}\right] \tag{2.22}$$

$$\varepsilon_{rk}S_k(\nabla\varphi)_k\ .\vec{\xi}_k = DE * \left[(\nabla\varphi)_{kx} * (PK)_x + (\nabla\varphi)_{ky} * (PK)_y + (\nabla\varphi)_{kz} * (PK)_z\right] \tag{2.23}$$

Where S_{kx}, S_{ky}, S_{kz} are the surface area vector components of the control volume face in k direction; they are defined as ARX, ARY and ARZ in the code. And $(PK)_x$, $(PK)_y, (PK)_z$ are the components of the vector which joins the nodes of two neighboring control volumes; in the code variables AKX, AKY and AKZ represent these components. We have some more variables which are used in the code as they are defined here:

$$DE = \frac{\varepsilon_{re}\left|\vec{S}\right|}{\left|\overrightarrow{PK}\right|}\ ; \qquad \vec{n}_k = \frac{\vec{S}}{\left|\vec{S}\right|} \qquad ; \qquad \vec{\xi}_k = \frac{\overrightarrow{PK}}{\left|\overrightarrow{PK}\right|}$$

According to the notation given in the code, we have:

$\vec{S} = (ARX)i + (ARY)j + (ARZ)k$

$$\left|\vec{S}\right| = \sqrt{ARX^2 + ARY^2 + ARZ^2}$$

$$\overrightarrow{PK} = (AKX)i + (AKY)j + (AKZ)k$$

$$\left|\overrightarrow{PK}\right| = \sqrt{AKX^2 + AKY^2 + AKZ^2}$$

The final algebraic equation corresponding to eq. (2.6) for a control volume cell as used in Orcale3D can be written as:

$$A_P\varphi_P + \sum_{k=newsbt} A_k\varphi_k = SUEH - SUEL \qquad (2.24)$$

Appendix 2

2.1 Some general MPI functions

Here we briefly explain some important MPI functions which are general for most of the MPI programs, and they are used in Oracle3D also.

MPI_INIT(IERR)
Initializes MPI environment

This function is the initialization function for MPI library. It sets the execution environment for the MPI communications. MPI_INIT must be called in every MPI program before beginning any MPI related task. No MPI function can be called before this function. It must also be called only once in a program.

MPI_COMM_SIZE(COMM, NPROCS, IERR)
Returns the number of MPI processes in input communicator

Here COMM is the input communicator for the function MPI_COMM_SIZE. In Oracel3D, this function is called just after the initialization function to set the variable NPROCS to the total number of available processes for the default communicator MPI_COMM_WORLD. This function can be used with any communicator. We have used it with different Cartesian and intra-communicators to assign the specific variables the number of available processes for the respective communicators.

MPI_BARRIER(COMM, IERR)
Provides a barrier for synchronization of processes

The MPI_BARRIER function is called with input communicator argument. It blocks the processes of input communicator until all the processes have reached this function. This call returns to a processor only when all the other processors of input communicator have entered this function. There are several instances in Oracle3D where we have used MPI_BARRIER to synchronize the timing of processes of a particular communicator.

MPI_FINALIZE(IERR)
Terminates MPI environment

MPI_FINALIZE terminates the executing MPI environment in the program. This function breaks the link of the program with the MPI library. This has to be called at the end of all the MPI related work. NO other MPI function can be called after the MPI environment has been terminated with MPI_FINALIZE.

2.2 Communication routines used in Oracle3D

MPI_SEND(buffer, count, datatype, destination, tag, comm, ierr)
Sends the buffer variable to the destination process with tag identifier

MPI_RECV(buffer, count, datatype, source, tag, comm, status, ierr)
Receives the buffer variable from the source process with tag identifier

The MPI_SEND and MPI_RECV are the blocking send and receive operations. Blocking operations do not return until the buffer variable is safe to reuse. In case of send, the buffer is ready to use when the buffer is transferred to the buffer of the receiver or the receiver has actually received the message. The MPI_RECV receives the message and block until the message data is transferred in its buffer. 'Count' is the number of elements in the buffer variable, and the buffer can be of any supported data type by the MPI library. A list of predefined MPI datatypes is provide in Tab. 2.2.1. 'Destination' argument is the rank of the destination process, and for receive the source argument is the rank of the sending process. The 'comm' argument is the communicator handle in whose context the message passing is being carried out.

Tag argument is an integer which is used to distinguish between different types of messages in the program, in same communication context. MPI has to provide queuing mechanism for the message-passing to be performed as desired. In this the 'tag' argument works as another screening parameter for the messages, along with the communicator and rank arguments. It is very important in a message passing system that a receive operation with a certain tag is finalized only when a matching send arrives to it. MPI also provides the wild-card tag entries that match any tag on receiver process, however, it is to be used wisely by the user after ensuring that there are no conflicting families of messages exist.

The size of the buffer variable in receive function call, must be greater than or equal to the size of the incoming message. In the overflow situations at the receive process, an error code is issued and the information pertaining to the source and tag of the overflowed message is returned in the 'status' argument. The information contained in the arguments – source, destination, tag and comm are collectively termed as the message envelope. For a receive operation to correctly perform, the message envelope provided by its arguments must match the destination, tag and comm values of the incoming message.

Table 2.2.1 MPI data types for respective Fortran data types [2]

MPI datatype	**Fortran datatype**
MPI_INTEGER	INTEGER
MPI_REAL	REAL
MPI_DOUBLE_PRECISION	DOUBLE PRECISION
MPI_COMPLEX	COMPLEX
MPI_LOGICAL	LOGICAL
MPI_CHARACTER	CHARACTER

MPI_ALLREDUCE(sendbuff, recvbuff, count, datatype, operation, comm, ierr)
Performs a reduce operation among all processes of a communicator

MPI_ALLREDUCE is a collective communication, in which all the processes of the communicator take part. All the processes provide a send buffer to the function as input argument, and everyone receives an output buffer in 'recvbuff'. The 'operation' arguments defines the nature of the reduction operation to be performed on the input buffer. There are several standard reduction operations made available by MPI library, and the users can also define their own operations. Tab. 2.2.2 shows a list of some common reduction operations given by MPI. There is a version of reduce function (MPI_REDUCE) which provides the output buffer to a root process only, in contrast with ALLREDUCE which provide the output value to all the member processes.

Table 2.2.2 MPI provided reduction operations

Operation	Definition
MPI_MAX	Maximum
MPI_MIN	Minimum
MPI_SUM	Sum
MPI_PROD	Product
MPI_LAND	Logical AND
MPI_LOR	Logical OR
MPI_MAXLOC	Maximum value and location
MPI_MINLOC	Minimum value and location

Both the input and output buffers are defined with their datatype and the number of elements (count). Both must have same datatype and count values. All the member processes of the communicator must provide the same size of send buffer of same data type, with same operation. The send buffer can be a scalar or an array variable. In Oracle3D, the MPI_MAX, MPI_MIN and MPI_LOR operation are mainly used at different location in code for different purposes. The last two operations mentioned in Tab. 2.2.2 are a little different from the rest. These two operations: MPI_MAXLOC and MPI_MINLOC, provide two elements as output. They give the maximum or minimum value, and also provide the rank of the first process which has the maximum or minimum values. Chapter 5 of the MPI standard 3.1 manual should be referred for further details on these functions.

MPI_BCAST(buffer, count, datatype, root, comm, ierr)
Broadcast a buffer from a root process to others in the communicator

This function is the standard broadcasting function of MPI. It is used when some data is to be sent from one process to all the other processes of the same communicator. It is a collective communication and all the processes of the communicator must make this call with same values

for comm and root. ‘ROOT’ argument is the rank of the process which has to broadcast the buffer variable to others.

2.3 Some additional COMMUNICATOR functions:

MPI_COMM_COMPARE(COMM_1, COMM_2, RESULT, IERR)
Compares two communicators

There are four possible results for this function.
MPI_IDENT: If the two input communicator handles refer to same communicator object.
MPI_CONGRUENT: If the underlying MPI groups in both communicators are identical in their constituent processes with the same rank order
MPI_SIMILAR: If the group processes are the same but the rank order in groups are different
MPI_UNEQUAL: If the above conditions do not match

MPI_COMM_DUP(COMM, OUT_COMM, IERR)
Creates a duplicate communicator with same topological information

MPI_COMM_CREATE(COMM, GROUP, OUT_COMM, IERR)
Create a new communicator from the processes of an existing MPI group

MPI_COMM_FREE(COMM, IERR, IERR)
Deallocates communicator object

Some additional functions associated with Cartesian Communicators:

MPI_TOPO_TEST(COMM, STATUS, IERR)
Returns the type of topology associated with comm

MPI_CARTDIM_GET(COMM, NDIMS, IERR)
Returns the nDIMS value for the input Cartesian communicator ‘comm’

MPI_CART_GET(COMM, MAXDIMS, DIMS, PERIODS, COORDS, IERR)
Returns all the information associated with the Cartesian communicator ‘comm’

MPI_CART_RANK(COMM, COORDS, RANK, IERR)
Returns the rank of the process with input COORDS value of the Cartesian communicator ‘comm’

MPI_CART_SUB(COMM, REMAIN_DIMS, OUT_COMM, IERR)
Partitions the Cartesian communicator in lower dimensional Cartesian communicators

2.4 Some functions associated with MPI Groups:

'Group' is again an MPI concept like communicators. A group defines a collection of processes which have ordered ranks in the communication pattern. Groups define the scope of process names (ranks) in collective and point-to-point communication operations in the MPI environments. In other words, a group is attached to and used within a communicator to describe the constituent processes in that communication universe. Although, groups are defined and manipulated separately from communicators but only communicators are used in all the communication operations.

MPI_COMM_GROUP(CART_COMM, CART_GROUP,IERR)
Creates an MPI group consisting of all processes of a communicator

Groups are represented with opaque group objects. MPI_GROUP_EMPTY is an MPI defined default group which has no members in it. A constant MPI_GROUP_NULL is also pre-defined by MPI which is used for invalid group handles. The base group in any MPI universe is associated with the default communicator MPI_COMM_WORLD. MPI provides a function to access groups associated with communicators which is: MPI_COMM_GROUP. The input for this function is the communicator handle for which we need a group handle. The second argument is the output which is the resulting group handle from this function. This function can be used with any communicator to get the group handle which contains all the process of the input communicator. This function is used when we create inter-communicators in next section. MPI library provides some other useful functions to manipulate and use the MPI groups. Some are mentioned here:

MPI_GROUP_UNION(in_group1, in_group2, out_group, IERR)
A new group from the union of two existing groups.

MPI_GROUP_INTERSECTION(in_group1, in_group2, out_group, IERR)
A new group from the common processes of two groups.

MPI_GROUP_DIFFERENCE(in_group1, in_group2, out_group, IERR)
A new group with those processes of first group which are not in second group

In the above-mentioned group operations, the order of processes in the output group is determined mainly by their order in the first group, and if necessary then by the order in the second group.

MPI_GROUP_INCL(GROUP,N, RANKS, OUT_GROUP, IERR)
A new group by including the processes, given by RANKS, of an existing group.

MPI_GROUP_EXCL(GROUP,N, RANKS, OUT_GROUP, IERR)
A new group by excluding the processes, given by RANKS, of an existing group.

Here ‘GROUP’ is an existing group with processes recognized by their ranks. RANKS for this function is an array which has the ranks of processes which need to be included or excluded in the output group ‘OUT_GROUP’. Argument ‘N’ is the total processes which will be used to create the new group. In MPI_GROUP_INCL, if argument ‘N’ is zero, then it will lead to the creation of new group which is empty and by default is provided by handle ‘MPI_GROUP_EMPTY’. In case of N=0 in MPI_GROUP_EXCL, the new group is identical to the input group, as the number of excluded processes is equal to zero.

MPI_GROUP_FREE(GROUP, IERR)
Deallocate the group object.

This should be called after the job of the group is finished. It returns an MPI_GROUP_NULL for the GROUP handle. If some group operations are on-going then the deallocation of group object waits until the on-going operations are finished.