Titre: Title:	Modélisation CFD 2D de la valve mitrale et de son influence sur l'écoulement dans le ventricule gauche pour le post-traitement de données échocardiographiques
Auteur: Author:	Aurélien Lacourt
Date:	2025
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Lacourt, A. (2025). Modélisation CFD 2D de la valve mitrale et de son influence sur l'écoulement dans le ventricule gauche pour le post-traitement de données échocardiographiques [Master's thesis, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/66536/

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Modélisation CFD 2D de la valve mitrale et de son influence sur l'écoulement dans le ventricule gauche pour le post-traitement de données échocardiographiques

AURÉLIEN LACOURT
Institut de génie biomédical

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*

Génie biomédical

Juillet 2025

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Modélisation CFD 2D de la valve mitrale et de son influence sur l'écoulement dans le ventricule gauche pour le post-traitement de données échocardiographiques

présenté par **Aurélien LACOURT**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

L'Hocine YAHIA, président
Delphine PÉRIÉ-CURNIER, membre et directrice de recherche
Lyes KADEM, membre

DÉDICACE

"Ce que l'on conçoit bien s'énonce clairement,
Et les mots pour le dire arrivent aisément."
Boileau, L'Art poétique (1669-1674)

REMERCIEMENTS

Je remercie Delphine Périé-Curnier pour son accueil au sein de son laboratoire et pour son soutien au cours de ce projet de maîtrise. Je suis reconnaissant de votre confiance et de l'indépendance que vous m'avez accordé pour mener à bien ce projet.

Je souhaite également remercier l'ensemble des intervenants qui ont permis d'enrichir ce mémoire par des échanges ponctuels ou réguliers, sans pour autant tous les nommer, Lyès Kadem, Jean Provost, Damien Garcia.

Un immense merci à Sarah, ma collègue de bureau et amie à qui, en bon Français, j'ai pu râler lorsqu'après des heures de travail, mon modèle ne fonctionnait toujours pas.

Mon expérience à Montréal aurait été moins enrichissante sans ta rencontre et, par extension, celle des autres loustiques, Thomas, Lilian, Romain, Lola, Justin et j'en passe.

Merci également à Saïd Bichali et Daniel Curnier pour m'avoir donné l'opportunité d'observer des acquisitions échocardiographiques à l'effort et de me familiariser avec le logiciel EchoPAC.

Enfin, je dédie ces derniers mots de remerciement à ma compagne, Victorine Sirveaux, pour m'avoir supporté et soutenu lorsque, grincheux, je ne voyais pas la finalité de mon travail.

RÉSUMÉ

L'étude de l'hémodynamique intraventriculaire permettrait de faire un diagnostic précoce de pathologies cardiaques. Au cours des dernières décennies, pour palier à la résolution limitée de l'imagerie médicale, des modèles numériques ont été développés pour améliorer la compréhension de l'écoulement dans le cœur. Ce terrain fertile a permis la découverte de nombreux marqueurs diagnostics caractérisant la capacité cardiaque. En se rapprochant de la réalité *in vivo*, la complexification des modèles numériques, en terme de ressources informatiques et de spécificité des données utilisées, a rendu impossible leur utilisation dans un contexte clinique. La démarche régressive de ce mémoire est de concevoir un modèle numérique de complexité modérée pour le post-traitement d'images médicales. L'hypothèse de recherche est qu'un modèle CFD 2D ALE patient-spécifique à géométrie prescrite basé sur des données Échocardiographiques (EchoCG) mode B reconstruit de manière satisfaisante l'écoulement dans le Ventricule Gauche (VG) et permettrait la comparaison inter-individuelle.

Ce projet de maîtrise se consacre uniquement à la conception du modèle numérique et s'articule autour des objectifs suivants : l'implémentation en langage Python et l'automatisation basé sur l'écosystème API PyAnsys ; le développement d'un modèle paramétrique de Valve Mitrale (VM) ; l'étude de sensibilité des paramètres géométriques β_t et β_n, respectivement d'asymétrie et de prolapsus de la VM et finalement la validation des performances par comparaison à la méthode iVFM et aux mesures EchoCG Doppler.

Contrairement à β_n qui ne montre pas d'influence significative, une valeur physiologique de $\beta_t \sim 0.7$ est critique pour la formation d'un vortex diastolique donnant lieu à une recirculation efficace dans le VG. Des valeurs $\beta_t = 0.5$ (ie symétrie de la VM) et $\beta_t = 0.3$ (ie asymétrie inversée) sont défavorables au remplissage diastolique et l'éjection systolique.

La validation du modèle révèle l'importance de l'estimation du flux diastolique entrant ϕ_e. En recalibrant ϕ_e avec l'EchoCG Doppler, le modèle simule un écoulement proche du cas iVFM et de la référence *in vivo* ; là où l'utilisation de la variation géométrique dans le plan sous estime ϕ_e et fait échouer le modèle. Par comparaison, le champ simulé présente une structure vorticale plus complexe qui proviendrait d'un manque de modélisation de sources dissipatives atténuant les vortex de cycle en cycle : dissipation turbulente et rugosité de la paroi.

Finalement, en simulant l'écoulement physiologique permanent (i.e. 3 cycles cardiaques) en 45 min et au vu de ses performances, la formulation finale semble prometteuse. Bien que moins performant en terme de réalisme, ce modèle reconstruit l'écoulement patient-spécifique en un

temps limité en se basant sur des acquisitions communément réalisées en clinique.

ABSTRACT

The presymptomatic study of intraventricular hemodynamics could allow early diagnosis of cardiac pathologies. Over the past decades, numerical models have been developed to challenge the limited resolution of medical imaging and to provide a better understanding of blood flow within the heart. Within this framework, numerous diagnostic markers were discovered to assess the cardiac function. As models have become closer to *in vivo* reality, their increasing complexity in terms of computational resources and data specificity has made their use in clinical settings impractical. The regressive approach claimed in this thesis is to design a numerically moderate model as a tool for the post-processing of medical images. The research hypothesis is that a 2D ALE patient-specific CFD model with prescribed geometry, based on B-mode echocardiographic (EchoCG) data, should provide satisfactory reconstruction of the flow in the Left Ventricle (LV) and enable inter-individual comparison.

This master's project mainly focuses on the design of this numerical model and is structured around the following objectives: the implementation in Python and the automation based on the PyAnsys API ecosystem; the development of a parametric model of the Mitral Valve (MV); a sensitivity study of the influence of geometric parameters β_t and β_n, representing respectively the asymmetry and the prolapse of the MV; and the validation of the model and its performance compared to the iVFM method and Doppler EchoCG measurements.

In contrast to β_n, which shows little to no significant influence, a physiological value of $\beta_t \sim 0.7$ seems critical to the development of the diastolic vortex evolving into an efficient ventricular recirculation. Values of $\beta_t = 0.5$ (i.e. MV symmetry) and $\beta_t = 0.3$ (i.e. reversed asymmetry) induce both an adverse flow for the diastolic filling and the systolic ejection.

The validation process reveals the importance of accurately estimating the magnitude of the inlet diastolic flow rate ϕ_e. By recalibrating ϕ_e on the Doppler EchoCG data, the model succesfully simulates a flow close to the iVFM result and the *in vivo* reference, whereas the use of in-plane geometric variation underestimates ϕ_e and causes the model to fail. In comparison, the simulated velocity field shows a more complex vortical structure, likely due to the lack of modeled dissipative sources that would attenuate vortices cycle after cycle such as turbulent dissipation and wall roughness.

Eventually, by its ability to reconstruct a physiological steady flow in the LV in agreement with EchoCG measurements in only 45 minutes, the last formulation appears to be a promising choice for future studies. While being less realistic than other high-end models, this one simulates patient-specific flow in a limited time based on commonly acquired in clinical data.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

ALE	*Arbitrary Lagrangian-Eulerian*
AMA	Angle Mitral Antérieur
AMP	Angle Mitral Postérieur
CFD	*Computational Fluid Dynamics* (traduction Mécanique des Fluides Numériques)
EchoCG	Echocardiographie
FDA	Flux Diastolique Amplifié
GP	Géométrie Prescrite
ICC	Insuffisance Cardiaque Congestive
IFS	Interactions Fluide-Structure
IFSI	Interactions Fluide-Structure Implicite
IRM	Imagerie par Résonance Magnétique
iVFM	*intraventricular Vector Flow Mapping*
LLA	Leucémie Lymphoblastique Aiguë
LVA	Longueur Valvule Antérieure
LVP	Longueur Valvule Postérieure
MEF	Méthode des Eléments Finis
MFN	Mécanique des Fluides Numériques
MVF	Méthode des Volumes Finis
nID	Numéro d'identification d'un nœud de la frontière du domaine
OMC	Ouverture Mitrale Contrainte
OMNC	Ouverture Mitrale Non Contrainte
PV	Pression-Volume
UDF	*User Defined Function* (vocabulaire ANSYS Fluent)
US	Ultrasons
VA	Valve Aortique
VG	Ventricule Gauche
VM	Valve Mitrale
VMA	Valvule Mitrale Antérieur
VMP	Valvule Mitrale Postérieur
VTD	Volume Télédiastolique
VTS	Volume Télésystolique
2D	Deux dimensions (bidimensionel)

3D	Trois dimensions (tridimensionel)
4D	Trois dimensions spatiales + une dimension temporelle

LISTE DES ANNEXES

CHAPITRE 1 INTRODUCTION

Les pathologies cardiaques touchent une large population et changent drastiquement la qualité de vie des personnes atteintes. Au Canada, l'insuffisance cardiaque touche environ 20% de la population, elle représente la première cause d'hospitalisation et une cause principale de décès, avec plus de 20.000 cas recensés par an [18, 19]. Il existe une grande variété de causes pouvant compromettre le système cardiovasculaire. Dans ce mémoire, nous nous concentrerons sur celles qui touchent à l'intégrité et au bon fonctionnement du muscle cardiaque. Une dégradation de la fonction cardiaque, c'est à dire sa capacité à pomper efficacement le sang, survient lorsque les tissus musculaires du cœur s'affaiblissent, se rigidifient ou se détendent. Il y a autant de types de cardiomyopathies que d'origines à cette altération du muscle cardiaque. Les symptômes de l'insuffisance cardiaque (essoufflement, fatigue, tachycardie, etc.) sont souvent mal reconnus de la population générale, ce qui conduit à des diagnostics tardifs, après que le muscle ait subi des altérations irréversibles. L'anomalie devient alors observable par Échocardiographie (EchoCG) ou par Imagerie à Résonance Magnétique (IRM).

Pour palier à ce problème de santé, il est impératif de créer de nouveaux outils de diagnostic. Une idée de plus en plus répandue au sein de la communauté médicale et scientifique est que l'étude d'anomalies de l'écoulement sanguin permettrait de faire un diagnostic précoce de pathologies cardiaques avant le remodelage du myocarde et donc de la dégradation de la fonction cardiaque. Les méthodes d'imageries permettant l'observation de l'écoulement intraventriculaire sont l'EchoCG Doppler et le ciné IRM. Bien qu'en développement depuis des années, ces moyens d'observations offrent des résolutions spatiales et temporelles technologiquement limitées. Les séquences cliniques d'acquisitions IRM offrent généralement une meilleure résolution spatiale et temporelle que celles dédiées à l'EchoCG. Toutefois des études dévoilent des mesures EchoCG atteignant des résolutions de $\sim (0.5\ mm, 2\ ms)$ [17].

De nombreux modèles numériques ont vu le jour à des fins de recherche sur la compréhension de l'écoulement intraventriculaire. Ces modèles, parfois patient-spécifiques, ont l'avantage de permettre une résolution temporelle et spatiale accrue. En améliorant la compréhension du flux sanguin, ils mettent en avant des phénomènes hémodynamiques communs entre les individus, ce qui est propice à la découverte de bons marqueurs diagnostics comme le gradient de pression ou le développement d'une recirculation lors de la diastole ventriculaire [20–26]. Malheureusement, si ces modèles deviennent trop complexes en terme de spécificité des données utilisées ou de la quantité de ressources informatiques demandée, cela limite grandement leur capacité à être employés pour des études statistiques ou en tant qu'outil

clinique.

Ce mémoire considère une approche différente, où le modèle numérique est conçu dans l'optique d'être intégré à un processus de post-traitement d'images médicales pour reconstruire l'écoulement intraventriculaire. L'hypothèse de recherche est qu'un modèle de faible complexité numérique est suffisant pour reconstruire un écoulement physiologique et permettre la comparaison inter-individuelle. Plus particulièrement, ce projet s'intéresse à la conception et à la simulation de l'écoulement sanguin dans le ventricule gauche à partir d'image EchoCG.

La **Section 1.1** présentera rapidement le contexte clinique dans lequel s'inscrit le mémoire. Les **Sections 1.2** et **1.3** présenteront respectivement le but et les objectifs spécifiques de ce projet par rapport au contexte global. Enfin la **Section 1.4** propose une vue d'ensemble sur les différents chapitres de ce mémoire.

1.1 Contexte Clinique

Cette section a pour but de situer le contexte clinique de l'étude. Elle est divisée en deux parties, la première est dédiée à une introduction aux cardiomyopathies, la seconde renvoie à l'étude clinique dont les données utilisées dans ce manuscrit sont extraites.

1.1.1 Introduction aux cardiomyopathies

Comme mentionné précédemment, l'étude de l'écoulement sanguin permettrait de poser un diagnostic précoce de cardiomyopathies et donc de prévenir dans la mesure du possible l'apparition de l'insuffisance cardiaque. Dans cette optique, il est nécessaire de définir ce qu'est une cardiomyopathie et les différentes pathologies qu'elles regroupent. Cette sous-section fait référence dans sa globalité à l'article du NIH sur les cardiomyopathies [27].

Une cardiomyopathie est un diagnostic associé à une dysfonction électrique ou musculaire du cœur. Elle recouvre une grande diversité de pathologies à forte mortalité et morbidité, dont la plupart conduisent progressivement à une insuffisance cardiaque accrue.

Il existe différents types de cardiomyopathies, de diverses origines, les principales étant répertoriées dans le **Tableau 1.1**.

Finalement, les cardiomyopathies sont en général asymptomatiques dans les premiers temps, puis présentent par la suite les symptômes typiques des insuffisances cardiaques : essoufflement, fatigue, toux, orthopnée, œdème. Le diagnostic repose généralement sur des tests sanguins, l'électrocardiographie et l'EchoCG. Les traitements ont souvent pour unique objectif de réduire les symptômes. Les principales options de traitements sont les thérapies

Type	*Phénomène Macroscopique*	*Phénomène Microscopique*	*Conséquence*
Cardiomyopathie dilatée	Dilatation ventriculaire	Diminution de la vascularisation du myocarde	Diminution de la force contractile
Cardiomyopathie hypertrophique	Hypertrophie ventriculaire	Prolifération de tissu amorphe situé entre les fibres musculaires	Augmentation de la rigidité
Cardiomyopathie restrictive	Diminution de la compliance cardiaque	Fibrose endomyocardique, processus infiltratif ou métabolique	Augmentation de la résistance au flux
Dysplasie ventriculaire droite arythmogène	Transformation du tissu cardiaque en tissu fibro-graisseux	Remplacement des cellules musculaires du ventricule droit par des cellules adipeuses	Trouble du rythme ventriculaire

Tableau 1.1 Types de Cardiomyopathies - Étiologie

pharmacologiques, la pose d'un défibrillateur automatique, les thérapies de resynchronisation cardiaque et la transplantation cardiaque.

1.1.2 Introduction au projet PETALE

Ce projet se situe dans le prolongement d'un précédent travail exploratoire [1] sur la conception d'un modèle numérique patient-spécifique pour la simulation du flux dans le Ventricule Gauche (VG) pour la détection de nouveaux indicateurs d'Insuffisance Cardiaque Congestive (ICC) au sein d'une population de survivants de Leucémie Lymphoblastique Aiguë (LLA). Il s'agit désormais de développer un nouveau modèle, comme amélioration de cette version antérieure vers un outil de post-traitement de données EchoCG. Les données utilisées dans ce mémoire sont issues de la population de contrôle du projet Prévenir les Effets TArdifs de la Leucémie lymphoblastique aïgue chez l'Enfant (PETALE), initié au Centre Hospitalier Universitaire Sainte-Justine (Montréal, Canada) en 2013, pour l'étude des effets à long terme des traitements chez les survivants du cancer pédiatrique.

Les anthracyclines sont une famille d'agents de chimiothérapie utilisés pour le traitement de cancer pédiatrique telle que la leucémie lymphoblastique. Ce traitement, spécialement lors de la croissance des individus, augmente le risque de développer une ICC au cours de leur vie. Le projet Pétale se base sur une population de 250 patients de moins de 19 ans lors du diagnostic de LLA et en rémission depuis un minimum de 5 ans. Ces sujets ont fait l'objet de mesures EchoCG et IRM pour l'évaluation de leur santé cardiaque. Basés sur cette étude, de nombreux modèles numériques ont déjà été conçus pour déterminer les propriétés mécanique du VG [28–30].

À l'inverse de beaucoup de travaux publiés qui utilisent le ciné IRM, considéré comme le *gold standard* en imagerie cardiaque, les données utilisées ici seront issues de l'EchoCG. La motivation principale est que cette méthode d'imagerie est largement utilisée en clinique pour son aspect peu invasif et son faible coût. De plus, les données étant plus bruitées et moins

résolues avec l'EchoCG qu'avec l'IRM, si la performance de notre outil est avérée, alors elle le sera également pour des données IRM. Enfin, l'EchoCG est un examen médical régulièrement pratiqué et il existe de grandes bases de données hospitalières qu'il serait possible de valoriser sans avoir recours à de nouvelles acquisitions.

1.2 Éléments de la problématique

L'étude de l'écoulement intraventriculaire permettrait la détection pré-symptomatique d'anomalies conduisant à l'insuffisance cardiaque. Il est donc impératif de se doter des moyens d'observer et de quantifier à la fois l'écoulement sanguin mais aussi la réponse mécanique du myocarde. Cette démarche peut se faire d'un point vue purement cinématique avec de l'imagerie médicale ou bien de manière dynamique en modélisant le phénomène cardiaque. De manière analogique, l'approche cinématique consiste à regarder des images d'un film uniformément réparties dans le temps et l'approche dynamique consiste à reconstruire le film à partir de quelques images et d'hypothèses. D'une part, nous avons une description discrète mais exacte d'un phénomène et, d'autre part, une description continue mais approchée de la réalité.

De part son exactitude, l'imagerie médicale est la méthode de référence pour le diagnostic, mais cela n'a pas empêché le développement de modèles numériques.

1.2.1 Évolution des moyens de simulation intraventriculaire et apparition de nouveaux marqueurs de diagnostic

En parallèle de l'avancée technologique de l'imagerie médicale, les ressources informatiques ont permis le développement de modèles numériques de plus en plus complexes pour décrire le comportement mécanique du myocarde, mais aussi l'écoulement sanguin intraventriculaire.

Ci-dessous, quelques aspects non exhaustifs permettant d'évaluer la complexité d'un modèle numérique :

- **Dimension spatiale** : 2D $\underset{+}{\longrightarrow}$ 3D
- **État de l'écoulement** : Stationnaire $\underset{+}{\longrightarrow}$ Transitoire
- **Résolution spatiotemporelle** : Faible $\underset{+}{\longrightarrow}$ Haute
- **Géométrie** : Simplifiée $\underset{+}{\longrightarrow}$ Spécifique
- **Dynamique de la géométrie** : Fixe $\underset{+}{\longrightarrow}$ Mobile
- **Couplage fluide-structure** : Non Couplé $\underset{+}{\longrightarrow}$ Couplage unilatéral $\underset{+}{\longrightarrow}$ Couplage Bilatéral

Il est attendu que plus un modèle cherche à se rapprocher de la réalité, plus il doit modéliser les différents aspects du phénomène global et donc plus sa complexité augmente. La complexité numérique et conceptuelle va de paire avec le coût numérique et le temps nécessaire à la mise en place du modèle. On peut retracer dans la littérature, ce qui sera fait ultérieurement, la complexification des modèles CFD aux cours des années. Leur objectif commun est de produire un écoulement simulé le plus proche possible de la réalité, et moyennant

validation sur des données cliniques, de pouvoir caractériser le phénomène cardiaque [31, 32]. En comparant des simulations faites sur cas pathologiques et sur cas sains, il est possible de déterminer des marqueurs permettant le diagnostic. Ces marqueurs diagnostics, une fois validés par une étude statistique, deviennent utiles à la communauté médicale pour le diagnostic de la pathologie étudiée.

1.2.2 Conséquences et heuristique du mémoire

Cependant, cette complexification comporte deux principaux effets négatifs : Premièrement, en devenant spécifique dans les données utilisées et coûteux numériquement, ces modèles ont perdu leur capacité à être mis en place sur de larges cohortes. De ce fait, l'étude statistique des biomarqueurs ne pouvant plus être réalisée directement avec les modèles numériques, ces études deviennent donc des travaux exploratoires. La validation incombe donc à des études statistiques par imagerie médicale, perdant ainsi la capacité d'utiliser une résolution spatio-temporelle élevée. Deuxièmement, en se complexifiant, ces modèles numériques ont perdu également leur capacité à être utilisés en clinique pour la reconstruction de l'écoulement. Le temps de calcul trop élevé et la spécificité des données d'entrée font qu'actuellement il n'est pas envisageable de concevoir un outil clinique à partir des modèles numériques les plus performants.

Il devient donc intéressant d'envisager une approche régressive. La réalité se trouvant au sein des images médicales, il est pertinent de penser un modèle numérique comme un moyen d'extraire plus d'information de l'image brute en améliorant sa résolution spatiale et temporelle. L'imagerie serait alors utilisée à la fois comme donnée d'entrée pour la mise en place du modèle, mais aussi pour l'optimisation des paramètres numériques de simulation, assurant ainsi une reconstruction de l'écoulement de haute résolution et proche de la réalité physiologique. Le second point, à contre pied de la tendance actuelle, est qu'un modèle de complexité faible à modérée serait suffisamment performant pour assurer la simulation d'un écoulement physiologique et permettre la comparaison inter-individuelle. En devenant léger, ce modèle peut alors être transformé en outil de post-traitement d'images médicales et donc être utilisé en clinique à des fins d'études statistiques de marqueurs diagnostics ou pour le diagnostic lui-même.

1.3 Énoncé de la thèse du mémoire et des objectifs de recherche

L'hypothèse de recherche de ce mémoire est qu'un modèle numérique de faible complexité permettrait une reconstruction de l'écoulement intraventriculaire de qualité suffisante pour être robuste à la comparaison inter-individuelle, tout en assurant un coût numérique suffisamment faible pour s'intégrer dans un *workflow* de post-traitement d'imagerie médicale.

De manière plus spécifique, ce mémoire se situe dans le prolongement d'un travail préliminaire [1] de réalisation d'un modèle CFD 2D patient-spécifique de l'écoulement dans le VG basé sur des données échocardiographiques via la suite de logiciels Ansys. Toute référence à un "modèle précédent" sera à associer à ce travail préliminaire.

Dans ce mémoire, le but se situe autour de l'élaboration du modèle numérique et l'évaluation de ses performances. L'élaboration du *workflow* de post-traitement, bien qu'en essence présente, ne fait pas partie des attendus du projet. De même, la mise en pratique de l'outil sur une large cohorte et l'étude statistique qui en découlerait, devront faire l'objet d'études ultérieures.

Les sous-sections suivantes présentent les différents objectifs attendus de ce projet.

1.3.1 Objectif n°1 : Modification de l'implémentation numérique : vers un *workflow* automatisé de post-traitement en Python

La première étape est le changement de langage de programmation du précédent modèle. L'implémentation est réalisée en Python, et l'interaction avec les softwares de la suite Ansys anciennement faite via l'exécution de scripts en IRONPython est désormais gérée via l'écosystème API PyAnsys. Ces modifications ont un impact mineur sur les résultats des simulations en tant que telles, mais elles offrent de réels avantages en terme d'accessibilité, de lisibilité et de versatilité du code. En d'autres termes, ces changements rendront possibles et faciles l'intégration future de ce modèle numérique dans des *workflows* plus généraux.

1.3.2 Objectif n°2 : Développement et intégration d'un modèle paramétrique de valve mitrale

En accord avec la littérature, le modèle précédent, dépourvu de valve mitrale, a mis en avant une sensibilité accrue au choix du profil de vitesse d'entrée au niveau de la valve mitrale afin d'assurer un flux physiologique au cours de la diastole ventriculaire. Plutôt que de trouver un profil de vitesse mimant l'effet de la valve mitrale sur l'écoulement, le choix a été fait de concevoir un modèle paramétrique de valve mitrale et de l'intégrer au modèle.

1.3.3 Objectif n°3 : Étude de sensibilité des paramètres géométriques du modèle paramétrique de la valve mitrale

De part son caractère paramétrique, le modèle de valve mitrale requiert une étude de sensibilité. De manière plus spécifique, l'étude de sensibilité est concentrée sur l'influence des paramètres géométriques de reconstruction de la valve mitrale. Dans l'optique potentielle d'optimisation de ces paramètres, les plages testées sont extrêmes et sortent du domaine physiologique.

1.3.4 Objectif n°4 : Comparaison et validation avec la méthode iVFM

Enfin, les performances relatives de notre modèle sont évaluées par comparaison à la méthode iVFM [17, 33–35] de reconstruction de l'écoulement à partir d'EchoCG Doppler. Cette méthode a été validée et a fait l'objet de diverses publications. Bien qu'utilisée uniquement en recherche, cette méthode a pour intention, dans le futur, d'être employée en clinique.

1.4 Plan du mémoire

Le **Chapitre 2** est constitué d'une brève introduction aux connaissances nécessaires pour bien appréhender le manuscrit, ainsi qu'une revue de littérature scientifique de la modélisation numérique du VG. Les sections seront respectivement dédiées à l'EchoCG (voir **Section 2.1**), à la segmentation d'image médicale (voir **Section 2.2**), à l'anatomie du cœur et à la physiologie de la fonction cardiaque (voir **Section 2.3**), à la revue de littérature (voir **Section 2.4**) et enfin à une brève synthèse (voir **Section 2.5**).

Les chapitres suivants sont dédiées aux objectifs 2 à 4, le premier objectif étant diffus au sein des divers chapitres.

Le **Chapitre 3** est consacré à la conception du modèle patient-spécifique de l'écoulement dans le VG. Il est divisé en deux temps. Le premier temps est une description générale du processus de modélisation pour mieux aborder dans un second temps la présentation de différentes variantes du modèle et de leurs performances relatives.

Le **Chapitre 4** présente l'étude de sensibilité des paramètres géométriques de reconstruction de la Valve Mitrale (VM). En plus des moyens conventionnels d'étude de l'écoulement (amplitude de la vitesse, vitesse selon l'axe apex-base), une nouvelle méthode de cartographie et catégorisation de l'écoulement est développée. Les résultats sont présentés en deux temps, d'abord en se basant sur les méthodes conventionnelles, puis sur la méthode de cartographie et de catégorisation de l'écoulement.

Le **Chapitre 5** met en avant la validation des performances de notre modèle. Une première comparaison est faite avec les résultats de la méthode iVFM, puis une seconde comparaison avec la référence *in vivo* acquise par EchoCG Doppler. Une conclusion est apportée pour mettre en perspective les forces et les faiblesses de notre modèle et de ses différentes variantes.

Le **Chapitre 6** est une conclusion des travaux réalisés suivi d'une réflexion sur les limitations de l'approche proposée et enfin une ouverture sur les perspectives et améliorations futures.

CHAPITRE 2 REVUE DE LITTÉRATURE

2.1 Échocardiographie

L'EchoCG est une technique d'imagerie présente en clinique et en recherche. À la différence de son principal concurrent, l'IRM, elle se distingue par son faible coût et sa facilité d'utilisation. Cette technologie a vu de grandes avancées technologiques au fil des années [36–38] repoussant toujours les limites de la résolution spatiale et temporelle de l'acquisition.

Cette section sera divisée en une introduction aux Principes Physiques de l'EchoCG (Sous-Section 2.1.1) avec la présentation des modes B et M, puis à l'utilisation de l'EchoCG Doppler pour l'évaluation de l'hémodynamique (Sous-Section 2.1.2).

2.1.1 Principes Physiques

L'ouvrage de Nihoyannopoulos et Kisslo [2], sauf mention contraire, est utilisé comme référence pour les connaissances présentées ci-dessous.

L'EchoCG repose sur l'utilisation d'ondes à hautes fréquences pour reconstruire des images du cœur et de l'écoulement sanguin à travers ses différentes cavités (ex. atriums et ventricules) [37–40].

Des vagues de pression sont issues de la vibration de cristaux piézoélectriques en céramique et se propagent à travers le corps. La vitesse de propagation des ondes est principalement déterminée par la densité du milieu traversé. Par exemple, les vagues de pression se propagent à $\sim 300 m.s^{-1}$ dans l'air et en moyenne à $\sim 1540 m.s^{-1}$ dans les tissus mous du corps humain.

Il y a une relation fondamentale entre la vitesse de propagation d'une onde V, sa fréquence ν et sa longueur d'onde λ qui est décrite à l'**Eq. 2.1**. Afin de visualiser le cœur, il est nécessaire que la longueur d'onde soit du même ordre de grandeur que la plus petite de ses structures. Sachant que l'épaisseur de certaines valves sont de quelques mm, du fait de l'**Eq. 2.1**, cela donne des fréquences d'ondes émises de l'ordre du MHz. Ce spectre de fréquence appartient à la famille des ultrasons (US).

$$V = \nu.\lambda \tag{2.1}$$

Une autre propriété importante des ondes sonores est leur capacité à être réfléchies à l'interface entre des milieux de densité différentes. Ainsi une onde se propageant dans une direction

est réfléchie à chaque changement de milieu (tissus, sang, muscle etc.) emportant une partie de l'énergie incidente.

Le principe de diffraction s'applique également aux ondes sonores. L'onde incidente peut subir de la diffraction au passage d'un obstacle de taille inférieure à sa longueur d'onde. Ainsi, contrairement à un laser, un faisceau ultrasonore est constitué d'une partie centrale qui diverge à mesure que l'on s'éloigne du transducteur. Ce faisceau est utilisé pour viser une région d'intérêt, dans le cas de l'EchoCG, le cœur [38].

La conception d'une sonde EchoCG dépasse le cadre de cette introduction. Il faut retenir qu'une sonde se compose d'une multitude de cristaux ordonnés qui, sous l'effet d'une impulsion électrique, vibrent. La fréquence relative de vibration des cristaux des uns par rapport aux autres donnent lieu à différentes stratégies pour cartographier la zone d'intérêt. La fréquence de vibration se situe entre 1 et 7 Mhz. Ces cristaux vibrent à nouveau lorsqu'une onde réfléchie revient à la sonde, créant un signal électrique unique.

Une vague d'émission est définie comme une série d'ondes ultrasonores envoyées à travers le thorax. Chaque vague d'ondes ne dure que quelques μs. Les réflexions des ondes aux différentes interfaces étant spéculaires, autrement dit en miroir, il y a une perte importante de la qualité du signal de retour pour la majorité des structures cardiaques.

De manière théorique, le temps de propagation τ est une fonction de la distance entre l'interface et la sonde d, et la vitesse de propagation V donnée par l'**Eq. 2.2**. De manière numérique, pour chaque onde réfléchie, d est calculée par l'**Eq. 2.2** avec une mesure de τ et la valeur de V approximée à $\overline{V}$ donnée par l'**Eq. 2.1** pour des tissus mous.

$$\tau = 2.\frac{d}{V} \tag{2.2}$$

Les interfaces proches sont donc déterminées en premier, puis celles plus profondes. Il faut attendre le retour de l'onde réfléchie sur la structure d'intérêt la plus profonde avant d'envoyer une nouvelle série d'ondes. Enfin, le signal électrique des ondes réfléchies permet de créer un visuel. Plus spécifiquement, la sonde balaye une plage angulaire couvrant toute la zone d'intérêt. Pour chaque angle, l'intensité du signal réfléchie et la profondeur d des interfaces déterminent une image de la structure d'intérêt, voir **Figure 2.1**.

Le nombre d'images acquises par seconde, i.e. la fréquence d'acquisition, est ce qui définit la résolution temporelle de la mesure. Beaucoup d'aspects viennent influencer cette fréquence mais principalement, la profondeur d'acquisition, la plage angulaire et la résolution angulaire. Il convient donc de trouver un compromis pertinent pour chaque utilisation.

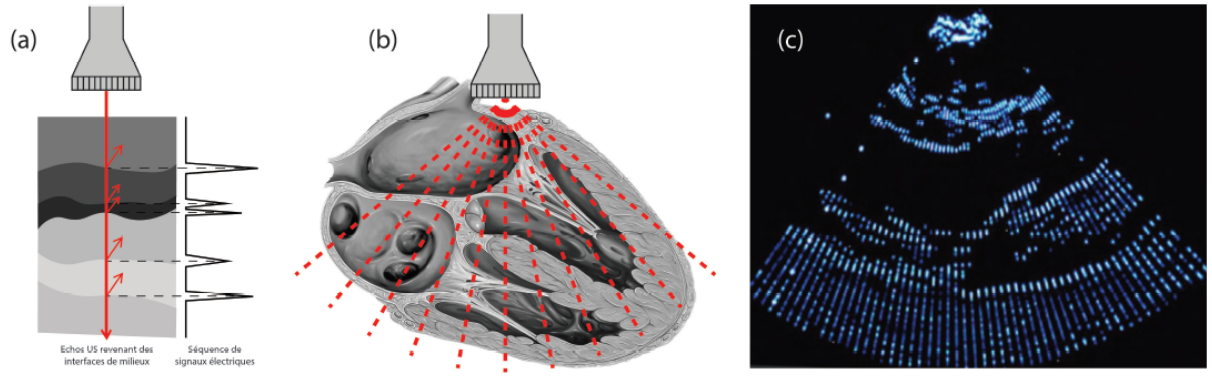


FIGURE 2.1 Concept de construction d'une image échocardiographique : (a) propagation des ondes ultrasonores à travers différents milieux, (b) divergence des ondes, (c) visualisation des données brutes. (Nihoyannopoulos et Kisslo (2009) [2] *©Springer-Verlag London Limited 2009* adapté par Dubois (2023) [1])

Des séquences dites "Ultra-Rapide" permettent désormais de faire des acquisitions avec une résolution spatiale et temporelle accrue en traitant plusieurs plans d'ondes simultanément [37, 38]. Ces avancées technologiques vont de pair avec l'amélioration des puissances de calculs et l'essor des GPU pour le traitement de large quantité de données. Désormais, des acquisitions 3D sont également rendues possibles avec des sondes à deux directions d'ondes permettant de reconstruire des volumes directement [41]. Cette technique 3D peut être utilisée en parallèle avec des séquences "Ultra-Rapide" pour avoir une reconstruction volumique de haute résolution temporelle [42]

L'image 2D reconstruite est souvent appelée mode B pour la différencier d'autres modes d'imagerie EchoCG. La principale alternative est l'EchoCG mode M. Dans ce cas, la structure est étudiée dans une direction fixe. Cette représentation spatiale 1D est balayée dans le temps pour obtenir la variation temporelle de la structure dans l'axe considéré au cours du temps, voir **Figure 2.2**. Bien que de prime à bord difficile à interpréter, cette méthode a l'avantage de fournir une résolution temporelle accrue avec ~ 1000 line.s^{-1} contre ~ 25 image.s^{-1} pour le mode B. Cette caractéristique donne accès à une information pseudo-continue en temps, ce qui est crucial pour des structures oscillantes comme des valves cardiaques.

2.1.2 Échographie Doppler

La suite de cette section est dédiée au principe d'EchoCG Doppler pour l'évaluation de l'écoulement sanguin dans le cœur.

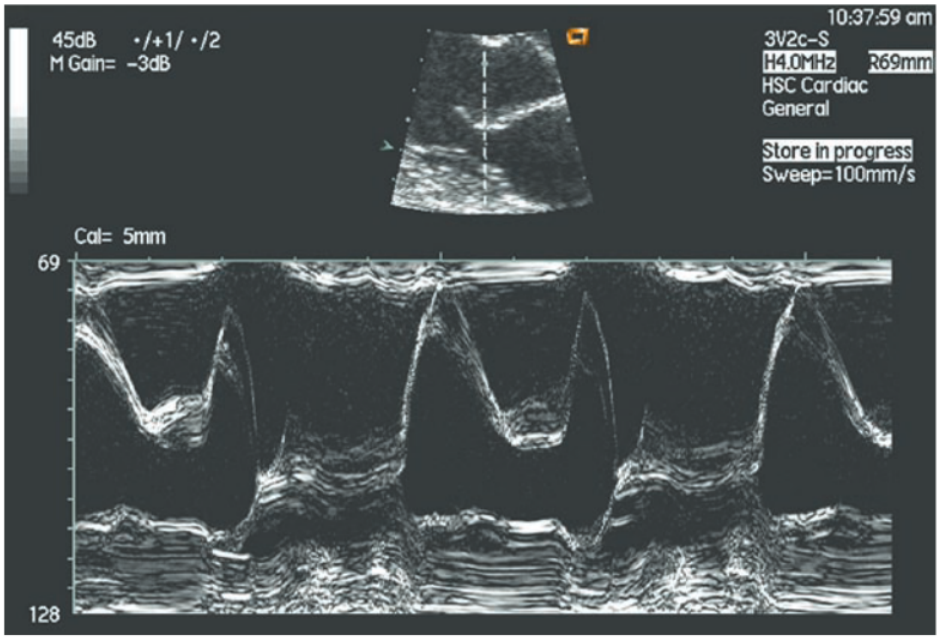


FIGURE 2.2 Exemple d'EchoCG mode M pour l'évaluation du mouvement de la valve mitrale. (Nihoyannopoulos et Kisslo (2009) [2] ©*Springer-Verlag London Limited 2009*)

Lorsque une onde ultrasonore rencontre un obstacle de taille significativement plus petite en comparaison à sa longueur d'onde, il y a alors diffusion de l'onde incidente qui se sépare de manière isotrope. Ce phénomène a deux conséquences, la première est qu'une fraction de l'énergie incidente est retournée à la sonde, la seconde est que si l'obstacle était en mouvement, la fréquence de l'onde retournée change selon le principe de Doppler. Si une onde rentre en contact avec un globule rouge, la fraction d'énergie à la sonde ne sera pas détectée. Mais si une vague d'ondes rencontre un écoulement sanguin alors la somme des ondes issues de diffusions multiples est détectable et par extension la différence de fréquence est mesurable.

Le principe de Doppler est le suivant, dans la situation où un récepteur fixe reçoit une onde issue d'un émetteur en mouvement alors l'onde reçue subit une "compression" si l'émetteur se rapproche du récepteur ou une "expansion" si l'émetteur s'éloigne, voir **Figure 2.3**. La compression et l'expansion sont caractérisées respectivement par une augmentation de la fréquence ν et une diminution de la longueur d'onde λ d'une part et une diminution de ν et une augmentation de λ d'autre part.

Dans le même référentiel galiléen, le récepteur et l'émetteur se déplaçant respectivement à la vitesse v_{rec} et v_{em} sur une même droite, si ν_{rec} et ν_{em} sont les fréquences des ondes émises et reçues, se propageant à une vitesse V dans le milieu considéré alors ν_{rec} et ν_{em} sont

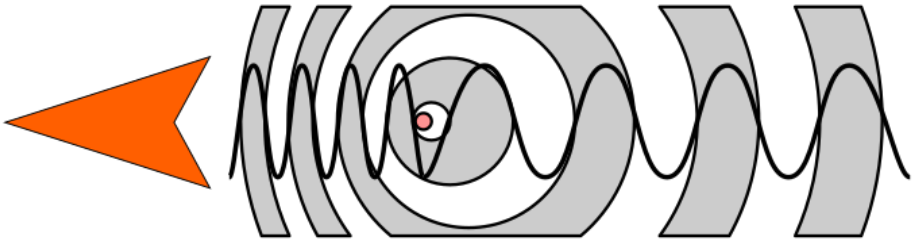


FIGURE 2.3 Principe de Doppler. (Wikimedia Commons [3] *CC 3.0*)

reliées par l'**Eq.** (2.3). Dans le cas spécifique de l'EchoCG Doppler, alors le récepteur est fixe et l'émetteur se situe au niveau de la zone de diffusion et se déplace à une vitesse égale à l'écoulement sanguin v_s de sorte que $v_{em} = 2.v_s.cos(\theta)$ avec θ l'angle entre l'onde émise et l'écoulement. Le décalage de fréquence entre l'onde émise et l'onde reçue est donné par l'**Eq. 2.4**.

La présence de la fonction cosinus dans l'**Eq. 2.4** implique que l'angle entre l'écoulement étudié doit être soit connu, soit approximé par $cos(\theta) \approx 1$. Pour un écoulement à 1 $m.s^{-1}$ et une fréquence d'ultrasons de 2.5 MHz, alors le décalage de fréquence à détecter se calcule par l'**Eq. 2.4** et vaut 3.3 kHz.

$$\nu_{rec} = \frac{V - v_{rec}}{V - v_{em}} \cdot \nu_{em} = \frac{1 - (v_{rec}/V)}{1 - (v_{em}/V)} \cdot \nu_{em} \tag{2.3}$$

$$\Delta\nu = \nu_{em} - \nu_{rec} = \frac{v_{rec}}{V} \cdot \nu_{em} = \frac{2.cos(\theta).v_s}{V} \cdot \nu_{em} \tag{2.4}$$

Dans les faits, le signal reçu est complexe du fait de la variation spatiale de l'écoulement. Le déplacement des tissus induit également une variation de la fréquence reçue. Cela implique que $\Delta\nu$ correspond à une plage de fréquence plutôt qu'à une fréquence fixe.

Ce principe a donné lieu à plusieurs techniques : Doppler à Émission Continue (*DEC*, *Continuous-Wave Spectral Doppler*), Doppler à Émission Pulsée (*DEP*, *Pulsed Wave Doppler*) et Doppler Couleur (*DC*, Color Doppler).

Doppler à Émission Continue (*DEC*) *DEC* repose sur l'utilisation d'une émission continue d'ondes US plutôt que de courtes vagues d'ondes. D'un point de vue technique, il faut différencier deux parties au sein de la sonde, une pour l'émission et l'autre pour la

réception. Les avantages de cette technique est la mesure d'une vitesse d'écoulement non ambiguë et sans phénomène d'*aliasing*. Le suivi de cette vitesse dans le temps permet par la suite d'estimer des volumes d'écoulement ainsi que des gradients de pression. Par exemple le *stroke volume* comme indice de performance cardiaque [43] ou le gradient de pression pour l'évaluation de sténose valvulaire [44] sont pertinents d'un point de vue clinique.

Doppler à Émission Pulsée (*DEP*) A la différence de *DEC*, *DEP* permet la localisation dans l'espace de la vitesse mesurée. Le principe est illustré à la **Figure 2.4**. Il faut définir un volume de contrôle, par un axe et une largeur, au sein duquel l'écoulement est étudié. Le principal inconvénient de *DEP* est l'*aliasing*. Puisque l'émission est discrète, en accord avec le Théorème de Nyquist, alors les décalages temporels correctement discernables doivent être de valeurs inférieures ou égales en valeur absolue à la moitié de la fréquence de pulsation du système. Cela se traduit par une troncature des vitesses au delà de la vitesse limite de Nyquist dont la partie tronquée sera représentée avec un signe opposé, rendant le champ de vitesse brut, bruité et ambiguë. La fréquence de pulsation est déterminée par la profondeur de la zone d'intérêt. De manière général, une mesure *DEP* du VG dans sa globalité montre de l'*aliasing* durant la totalité du cycle cardiaque, voir **Figure 2.5**.

Les applications cliniques de *DEP* sont reliées au besoin de localiser la mesure de vitesse dans l'espace, par exemple l'étude de l'écoulement sanguin au niveau des valves cardiaques et des ventricules.

Doppler Couleur (*DC*) *DC* est l'extention de *DEP* appliquée à une grille de pixels prédéfinis. Cette grille de pixels englobe à la fois les zones sanguines et tissulaires. A chaque pixel est associée une couleur correspondant à la vitesse radiale approximée. L'image finale est la superposition de la grille de couleur sur l'image structurelle mode B. La convention standard veut qu'une vitesse négative en bleu corresponde à un flux s'éloignant de la sonde et une vitesse positive rouge à un flux se rapprochant de la sonde, voir **Figure 2.5**.

Comme le ciné IRM, cette technique d'imagerie a permis une meilleur description des écoulements intraventriculaires. Elle permet d'un point de vue clinique de détecter rapidement des anomalies de l'écoulement et la pathologie en cause, par exemple une maladie dégénérative de la Valve Mitrale (VM) ou hypertrophie du VG.

Les principales limitations de cette technique sont d'une part l'*aliasing*, mais aussi sa faible résolution temporelle et sa mesure partielle de la vitesse.

Tout comme *DEP*, *DC* souffre d'*aliasing*, mais à un degré supérieur puisque la fréquence de pulsation est réduite. Dans la mesure où plusieurs pulsations sont requises pour reconstruire

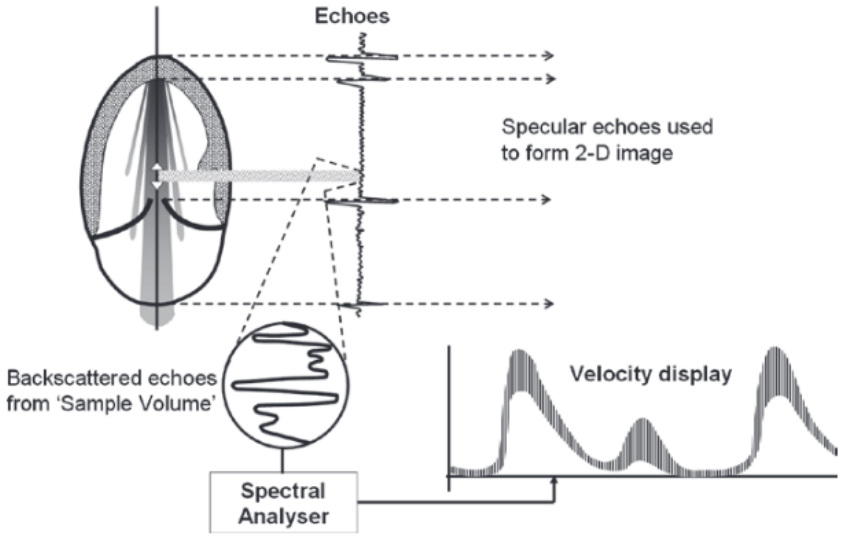


FIGURE 2.4 Principe du *DEP*. (Nihoyannopoulos et Kisslo (2009) [2] ©*Springer-Verlag London Limited 2009*)

la grille de vitesse, la résolution temporelle chute de sorte que le clinicien doit généralement réduire sa fenêtre d'observation pour garder des résultats cohérents. De la même manière, afin d'observer une large zone comme l'entièreté du VG, alors la résolution temporelle chute drastiquement. Le développement informatique et les nouvelles capacités de calculs ont permis de mitiger ce problème mais la fréquence de mesure reste de l'ordre de $\sim 50\ Hz$, ce qui est relativement faible pour obtenir une description précise de l'écoulement sanguin. Enfin, par nature la vitesse mesurée est une vitesse radiale. Bien qu'on puisse obtenir des résultats qualitatifs sans cette composante angulaire de la vitesse, il est impossible d'avoir des conclusions quantitatives.

Les critères hémodynamiques issus de la mécanique des fluides reposent tous sur un champ de vitesse continue en temps et en espace. Des travaux ont donc visé à palier aux limites du *DC*. De nombreux papiers ont utilisé des EchoCG Mode B et *DC* pour créer des modèles de Mécanique des Fluides Numériques (MFN, CFD) patient-spécifiques reconstruisant ainsi un champ de vitesse continu avec une forte résolution spatiale et temporelle [12, 13, 45–48]. Une autre approche a été de reconstruire la composante angulaire de la vitesse à partir du *DC* par minimisation algébrique d'une fonction mathématique assurant des lois physiques comme la conservation de la masse et partiellement les lois de Navier-Stokes [17, 33–35]. Enfin, une vague récente d'articles souhaite mettre à profit l'utilisation de PINNS (*Physics Induced Neural Networks*), des réseaux de neurones spécialement conçus pour approximer

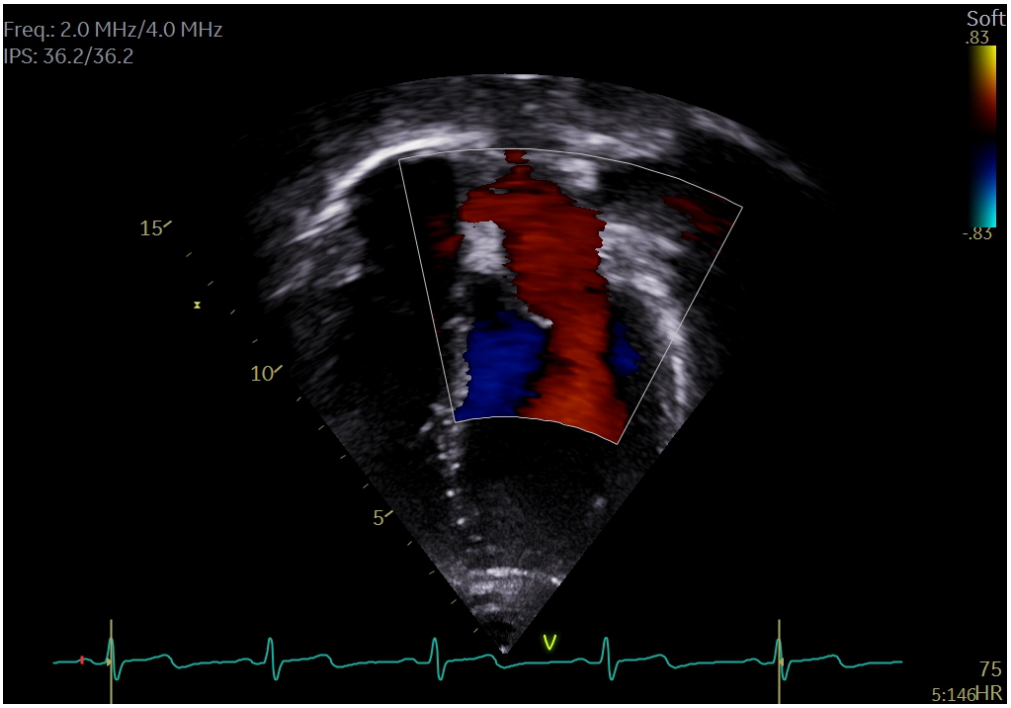


FIGURE 2.5 Mesure Doppler Couleur du ventricule gauche issue du projet Pétale.

des lois physiques, ici celles de la mécanique des fluides [49,50]. Ces réseaux de neurones sont souvent entraînés sur des données EchoCG cliniques ou artificielles issues de modèles CFD. L'avantage de ces deux dernières approches en comparaison avec les modèles CFD purs sont leur rapidité de calcul, un aspect critique pour l'application clinique.

2.2 Segmentation d'images médicales

Comme présenté au chapitre précédent, l'EchoCG est une technique d'imagerie non invasive pour la visualisation des structures cardiaques et de leur fonction. Cette méthode permet d'obtenir des images en temps réel lors d'un examen clinique. Une étape clé pour la réalisation de mesures quantitatives de la fonction cardiaque est la délimitation du VG. Parmi les paramètres évalués se trouvent par exemple le Volume Télédiastolique (VTD) et le Volume Télésystolique (VTS) dont l'importance est cruciale pour évaluer la capacité du cœur à pomper. En clinique, intégré au *workflow* d'imagerie, cette étape de segmentation se fait en général de manière semi-automatique. Le clinicien doit placer des marqueurs spécifiques sur le VG, puis l'algorithme propose une solution qui sera alors validée ou modifiée par le clinicien. En intégrant le praticien, une part de subjectivité est ajoutée et donc fait place à

de la variabilité entre les mesures. De plus, la segmentation peut se trouver être dispendieuse en temps. Donc des publications ont essayé de proposer des solutions pour automatiser la segmentation du VG. Bien que les méthodes générales de segmentation restent communes aux différents modes d'imagerie, chacune possède ses spécificités. La difficulté principale dans le cadre de l'EchoCG est la qualité de l'image qui varie significativement d'un individu à un autre rendant difficile de détecter les frontières du VG. De plus l'image EchoCG est caractérisée par un bruit inhérent. Ainsi le développement d'un outil de segmentation du VG à partir d'EchoCG mode B est un défi. Récemment, l'utilisation de Deep Learning pour la segmentation du VG a fourni de résultats prometteurs.

Méthodes semi-automatiques Dans une démarche d'automatisation, il s'est avéré pertinent d'ajouter des moyens de contrôle à la génération des contours segmentés pour minimiser les dérives à cause du bruit de l'image [20]. Il ressort également la nécessité de réaliser une étape de pré-traitement de l'image. Au moyen de filtres, l'image est modifiée de sorte à diminuer le bruit, améliorer le contraste et ainsi faciliter le repérage des contours [51].

Les méthodes semi-automatiques se basent sur l'utilisation de marqueurs sur le contour du VG donnés par l'utilisateur comme base pour déterminer l'ensemble de la segmentation. Ces points peuvent être des points physiologiques d'intérêt comme l'apex ou la base du VG [52] mais aussi les bords des valves mitrale et aortique. Ces marqueurs peuvent être aussi issus de protocoles plus arbitraires à base de recherche radiale ou normale à la droite apex-base [51].

Une fois ces points fixés, l'objectif est d'approximer le VG par une courbe paramétrique en se basant sur ces marqueurs.

Le principal avantage de ces méthodes est le contrôle de l'utilisateur sur la segmentation, la faible sensibilité au bruit et l'acquisition simultanée de points physiologiques d'intérêt. Mais elles pèchent à régler le problème de subjectivité de l'opérateur et restent relativement lentes.

Méthodes Automatiques La segmentation automatique du VG à partir de données EchoCG est un challenge pour diverses raisons. Les connaissances requises pour apprécier les différences entre les différents modèles de segmentation par *Deep Learning* dépassant les attendus de cette revue de littérature, nous présenterons seulement les difficultés pour développer ces méthodes, leurs performances et leurs limites.

Du point de vue de l'image, la variabilité de la qualité de l'image, la grande déformation morphologique au cours du cycle cardiaque et la mauvaise définition des frontières à cause du mouvement cardiaque, requièrent de l'algorithme d'être très robuste à une grande diversité d'image. Il est aussi crucial d'incorporer une cohérence spatio-temporelle entre les différentes

acquisitions d'un même cycle cardiaque pour préserver la cohérence physiologique de la segmentation. Enfin, l'apprentissage de ce genre de modèle requiert de larges bases de données annotées, alors que les bases de données médicales sont souvent peu annotées [53].

Le coefficient de Dice, publié par Lee Raymond Dice en 1945, est utilisé pour évaluer la performance des algorithmes de segmentation. D'un point de vue statistique, ce coefficient mesure la similarité de deux échantillons. Dans le cas de l'évaluation de segmentation, ce coefficient mesure la similarité entre la segmentation issue de l'algorithme et souvent une segmentation de référence réalisée manuellement par un clinicien. Des performances allant de 0.91 à 0.965 sont mentionnées dans la littérature [53–57], ce qui prouve la capacité de ces méthodes à fournir des segmentations de qualité, en un temps de l'ordre de la seconde pour certaines.

La limite principale est liée à la qualité de l'image et la diversité morphologique présentes au sein des acquisitions EchoCG. De manière générale, ces recherches restent préliminaires dans le sens où ces outils n'ont pas encore intégré les systèmes d'imagerie médicale commerciaux. Il est attendu qu'alors au sein du *workflow* d'acquisition d'image et de post-traitement, une vérification de la segmentation soit demandée au praticien.

2.3 Anatomie du cœur et physiologie de la fonction cardiaque

Cette section est dédiée à la présentation anatomique du cœur (**Sous-Section 2.3.1**) et la description du cycle cardiaque (**Sous-Section 2.3.2**).

2.3.1 Anatomie du cœur

Cette sous-section se base sur les connaissances fournies dans le livre *Gray's Anatomie : Les Fondamentaux* [58] sauf mention explicite.

Description générale Le cœur est enveloppé par le péricarde, une enveloppe externe constituée d'une couche de tissu conjonctif externe et d'une fine couche qui adhère au cœur.

Le cœur est une pyramide couchée, dont il est possible de distinguer la base de l'apex. La base du cœur est formée de l'atrium gauche, une partie de l'atrium droit et des grosses veines du cœur (cave et pulmonaire). La face antérieure du cœur est formée principalement du ventricule droit, d'une partie de l'atrium droit et du ventricule gauche. La face orientée vers le diaphragme se compose principalement du ventricule gauche, et séparé par le sillon interventriculaire, une partie du ventricule droit. La face orientée vers les poumons est formée

du côté gauche du ventricule gauche et d'une partie de l'atrium gauche et du côté droit de l'atrium droit.

Le cœur se compose de deux pompes séparées par une cloison médiane. La pompe droite reçoit le sang désoxygéné du corps et l'envoie aux poumons tandis que la pompe gauche reçoit le sang oxygéné des poumons et l'envoie dans la circulation générale. Chaque pompe se décompose d'un atrium et d'un ventricule séparés par une valve. L'atrium reçoit le sang arrivant au cœur alors que le ventricule éjecte le sang hors du cœur. Le circuit pulmonaire offrant une résistance hydraulique inférieure au circuit générale du corps, la paroi du ventricule gauche est significativement plus épaisse que celle du ventricule droit.

Spécificité du ventricule gauche Le ventricule gauche, voir **Figure 2.6**, est caractérisé par une cavité conique, plus longue que celle du ventricule droit, et est entourée par la partie la plus épaisse du myocarde. L'entrée du sang dans le ventricule gauche se fait depuis l'atrium gauche à travers la valve mitrale. L'éjection du sang se fait par la Valve Aortique (VA) donnant accès à l'aorte. La partie interne du myocarde est tapissée de trabécules charnues, i.e. des saillies musculaires arrondies ou irrégulières. Des muscles papillaires antérieurs et postérieurs, ainsi que des cordages tendineux, permettent l'actuation coordonnée des valvules de la valve mitrale. La valve mitrale rentre dans la catégorie des valves atrio-ventriculaires bicuspides. La cuspide antérieure ou Valvule Mitrale Antérieure (VMA) est la plus grande et aussi la plus mobile des deux cuspides, par comparaison avec la cuspide postérieure ou Valvule Mitrale Postérieure (VMP).

2.3.2 Description du cycle cardiaque

Cette sous-section se base sur les connaissances fournies au chapitre *Mechanical Aspects of Cardiac Performance* [59] du le livre *Handbook of Cardiac Anatomy, Physiology, and Devices* [60] sauf mention explicite.

Phases cardiaques Une description de l'activité électrique, du volume et de la pression du ventricule gauche au cours d'un cycle cardiaque est présentée à la **Figure 2.7**. Seulement l'interprétation mécanique des événements arrivant au cours du cycle cardiaque est donnée.

Au cours d'un cycle cardiaque, il faut différencier la phase de contraction de l'atrium et celle du ventricule, la première faisant place à la seconde.

La diastole, ou phase diastolique, commence avec l'ouverture de la valve mitrale lorsque la pression dans le ventricule passe sous la pression de l'atrium et que le ventricule gauche commence à se remplir. Après la phase de remplissage passif "Onde E" du ventricule, l'atrium

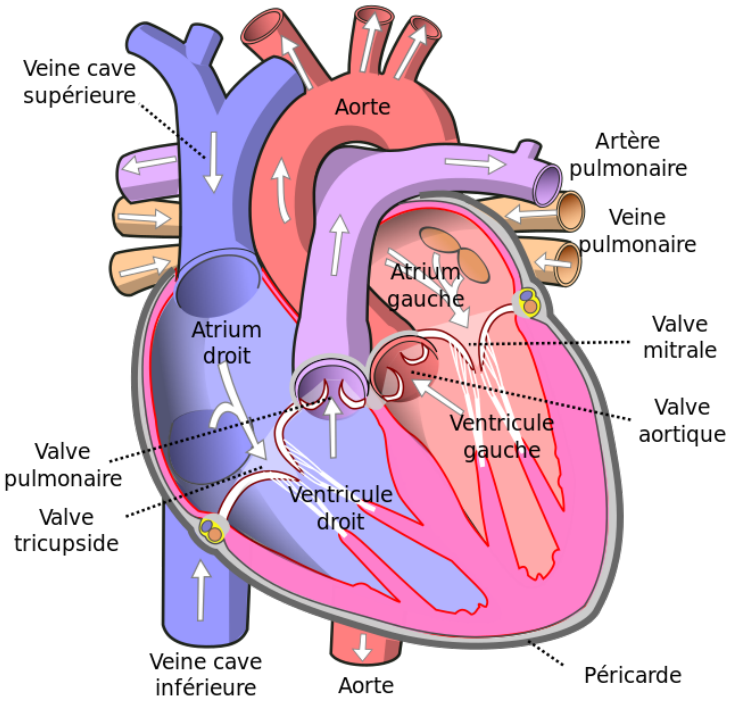


FIGURE 2.6 Schéma anatomique du cœur. (Wikimedia Commons [4] *CC 3.0*)

se contracte en fin de diastole, augmentant temporairement la différence de pression, forçant d'avantage de sang à pénétrer dans le ventricule, c'est le remplissage actif "Onde A". Cette dernière phase de remplissage représente 20% du volume entrant à un rythme cardiaque de repos. Tout au long de la diastole, le ventricule et l'atrium présentent des pressions quasi-identiques.

La systole, ou phase systolique, commence lorsque la contraction du ventricule augmente la pression intraventriculaire. Cela implique la fermeture instantanée de la valve mitrale et, à force d'augmenter, permet l'ouverture de la valve aortique. La période entre la fermeture de la valve mitrale et l'ouverture de la valve aortique est définie comme la phase de contraction isovolumique. Au début de la systole, après l'ouverture de la valve aortique, la pression continue d'augmenter dans le ventricule gauche et l'aorte du fait de la contraction du myocarde ventriculaire. Après avoir atteint un pic de pression, la force contractile du ventricule diminue progressivement. Finalement, la pression passe sous le seuil où la valve aortique se referme. Durant toute la phase d'éjection, il n'y a qu'une faible chute de pression au travers de la valve aortique. Après la fermeture de la valve aortique, le myocarde se relâche et la pression intraventriculaire diminue rapidement. Cette période de relâchement où les valves mitrale et

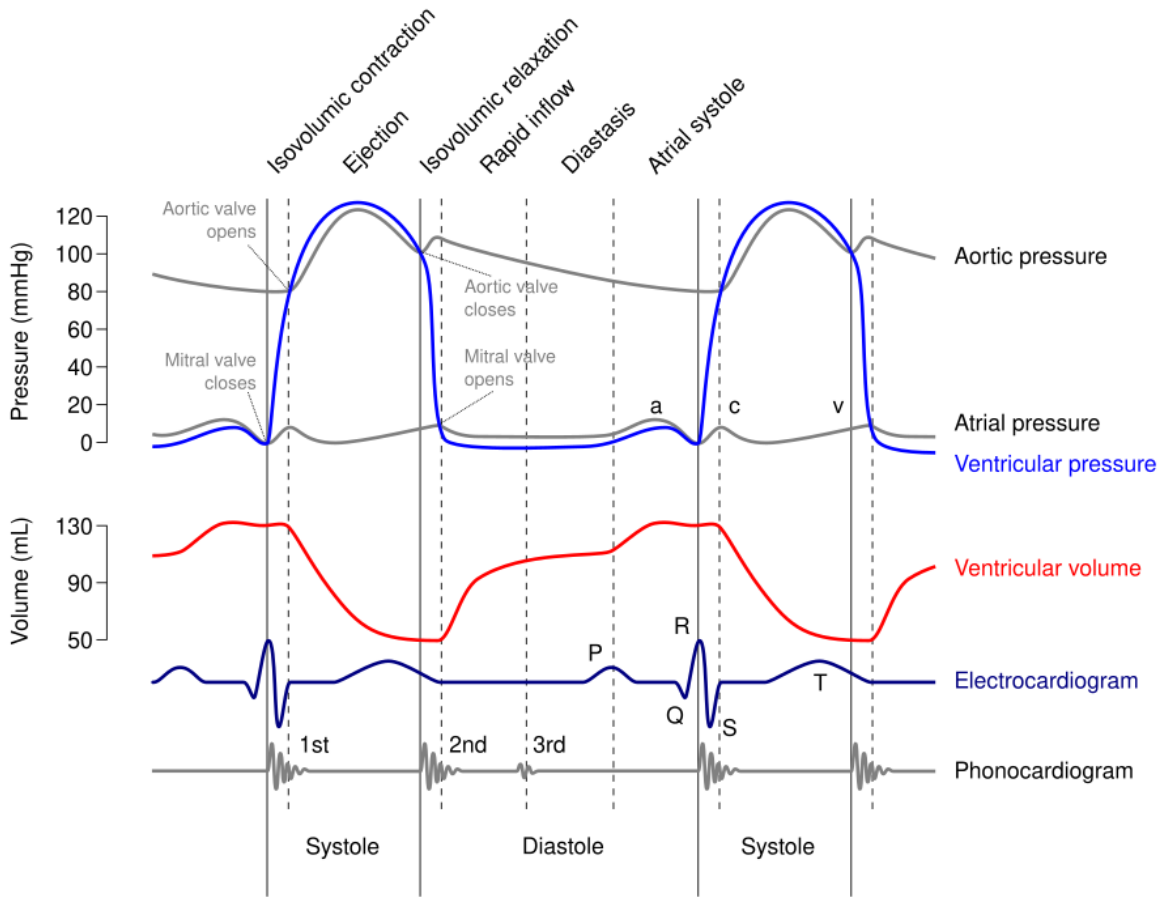


FIGURE 2.7 Cycle cardiaque du ventricule gauche. (Wikimedia Commons [5] *CC 2.5*)

aortique sont fermées définit la phase de relaxation isovolumique.

Lorsque la pression passe sous le seuil où la valve mitrale s'ouvre, alors un nouveau cycle s'enclenche.

Courbe PV Le fonctionnement ventriculaire est souvent étudié à travers une courbe Pression-Volume (PV). Cette courbe donne accès graphiquement à des paramètres quantitatifs pour évaluer la santé du ventricule gauche. Cette courbe est présentée à la **Figure 2.8**.

L'aire à l'intérieur de la courbe PV donne accès à l'énergie nécessaire au fonctionnement du ventricule au cours d'un cycle cardiaque. Le débit systolique est défini comme la différence entre le VTD et le VTS. La contractilité du myocarde est représentée par la pente de la relation P(V) en fin de systole. La compliance ventriculaire est l'inverse de la pente de la

relation P(V) en fin de diastole.

Chaque portion de la courbe PV, voir **Figure 2.8**, représente une phase précise du cycle cardiaque :

- (M) ouverture de la valve mitrale, début de diastole.
- (M) fermeture de la valve mitrale, fin de diastole.
- (A) ouverture de la valve aortique, début de systole.
- (A) fermeture de la valve aortique, fin de systole.
- A diastole, remplissage ventriculaire.
- B contraction isovolumique.
- C systole, éjection ventriculaire.
- D relaxation isovolumique.
- ESPVR *End-Systolic Pressure-Volume Relationship* pente de la contractilité ventriculaire.
- EDPVR *End-Diastolic Pressure-Volume Relationship* pente inverse de la compliance ventriculaire.

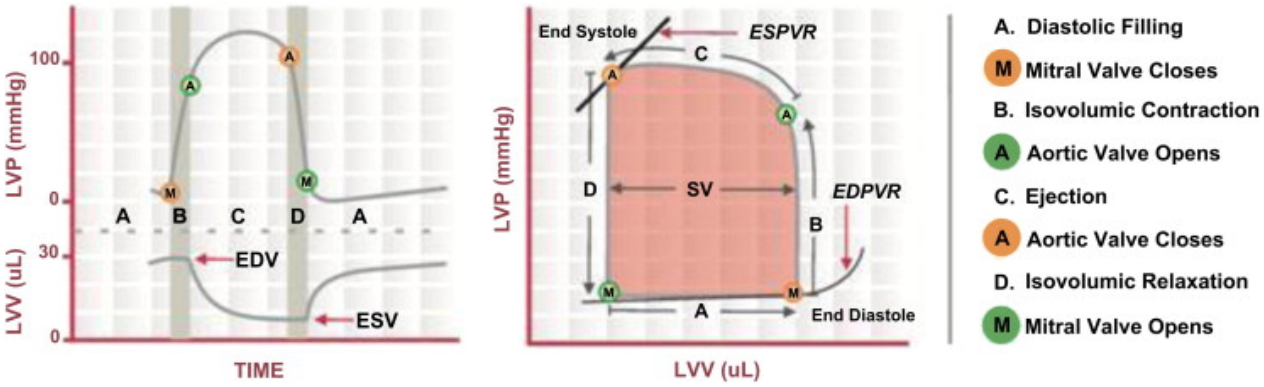


FIGURE 2.8 Courbe PV. (Wikimedia Commons [6] *CC 3.0*)

2.4 Modélisation numérique du VG

Cette dernière section s'intéresse à l'état de l'art des modèles numériques pour la modélisation mécanique du cœur, et plus particulièrement à celle de l'écoulement dans le VG à partir d'image EchoCG.

Depuis les années 1990, avec McQueen et Peskin [48,61,62], les modèles in silico de la fonction cardiaque ont été développés dans l'objectif de fournir un outil pour la compréhension de

l'écoulement cardiovasculaire.

L'étude de la dynamique de l'écoulement ventriculaire est un moyen pour évaluer la fonction cardiaque [31]. L'idée soutenue est que des anormalités dans l'écoulement sanguin causant un remodelage du myocarde apparaissent parfois avant les premiers symptômes de l'insuffisance cardiaque [63]. Par exemple, des études ont montré que la caractérisation du vortex issue de la valve mitrale lors de la diastole semble être un marqueur précoce de diagnostic de l'insuffisance cardiaque [21, 22, 26, 64–66]. Ainsi, en évaluant des conditions physiologiques variées [67], il est possible d'évaluer la sensibilité de ces marqueurs en fonction des pathologies. Des modèles fluides basés sur des images EchoCG ou IRM ont augmenté la précision de la solution numérique [20,68–70]. Cependant, la résolution au niveau de la base du VG a souvent requis l'utilisation de modèles complexes d'interaction fluide-structure [32,71] et l'intégration du comportement de valves cardiaques [72, 73].

Avant de faire une revue critique des différents modèles existants (**Sous-Section 2.4.2**), il faut d'abord faire une introduction à la mécanique des fluides et aux différentes catégories de modèles existants (**Sous-Section 2.4.1**).

2.4.1 Bases théoriques de la simulation fluide

Cette partie vise à introduire les principales connaissances en mécanique des fluides pour comprendre le comportement de l'écoulement sanguin dans le système cardiovasculaire. Il faut pour cela introduire les hypothèses physiques et les équations qui régissent la mécanique des fluides et ses implications d'un point de vue macroscopique. Cette introduction est complétée alors par une présentation de différentes catégories de modèles existants.

Toutes les équations et les connaissances théoriques de mécanique des fluides ont été validées auprès d'ouvrages de référence [64, 74].

Conservation de la masse La première loi physique qui gouverne l'écoulement sanguin est la conservation de la masse. Cette loi s'exprime de deux manière : localement **Eq. 2.5** et globalement **Eq. 2.6**.

$$\frac{\partial \rho}{\partial t} + \vec{\nabla} \cdot (\rho \cdot \vec{u}) = 0 \tag{2.5}$$

$$\frac{d}{dt} \int_{\Omega(t)} \rho(\vec{x}, t) \cdot d\Omega(t) = 0 \tag{2.6}$$

Ici, le domaine étudié est $\Omega = \Omega(t)$ avec $\rho = \rho(\vec{x}, t)$ la masse volumique du fluide et $\vec{u} =$

$\vec{u}(\vec{x},t) = (u,v,w)(\vec{x},t)$ le champ de vitesse.

Dans le cas du sang, il est légitime d'affirmer que ce fluide est incompressible et donc que sa masse volumique ne varie pas de manière significative, i.e. $\rho \approx \overline{\rho}$. Alors **Eq. 2.5**, se récrit en **Eq. 2.7**.

$$\vec{\nabla} \cdot (\vec{u}) = \frac{\partial u_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial w_z}{\partial z} = 0 \tag{2.7}$$

Cette caractéristique de divergence nulle est très importante car elle contraint fortement les trajectoires possibles de l'écoulement sanguin. Plus particulièrement, les vortex ne pourront se former qu'en contact avec un mur [64].

Conservation de la quantité de mouvement La viscosité μ est une propriété intrinsèque d'un fluide. Elle induit des contraintes de cisaillement visqueux dans l'écoulement. Intuitivement, l'énergie de dissipation visqueuse est l'énergie dissipée par l'écoulement, du fait du déplacement relatif des particules de fluides les unes par rapport aux autres. Cette dissipation visqueuse est la principale source de dissipation énergétique du fluide.

A la différence des fluides Newtoniens, où la viscosité est considérée constante, pour les fluides non-Newtoniens, la viscosité dépend du taux de cisaillement local du fluide.

Sans rentrer dans les détails, dans le cas d'un fluide incompressible et Newtonien, en l'absence de forces conservatrices, la conservation de la quantité de mouvement pour le fluide donne lieu à l'équation de Navier-Stokes, voir l'**Eq. 2.8**, une équation aux dérivées partielles non linéaire. Cette non linéarité rend la résolution numérique de cette équation un réel défi. Aucune solution analytique n'est possible, sauf cas particulier où une hypothèse est faite sur le champ de vitesse.

$$\underbrace{\frac{\partial u}{\partial t}}_{\text{variation temporelle}} + \underbrace{u.\nabla u}_{\text{terme inertiel}} = \underbrace{-\frac{1}{\rho}\nabla p}_{\text{force externe}} + \underbrace{\nu\nabla^2 u}_{\text{terme de viscosité}} \tag{2.8}$$

La viscosité cinématique $\nu = \frac{\mu}{\rho}$ est introduite et s'exprime en $m^2.s^{-1}$.

Couche limite et *Wall Shear Stress* En contact avec une frontière solide, la conservation de la masse impose que la vitesse normale à la paroi soit nulle. Et, du fait de la viscosité, il y a adhérence du fluide à la paroi, donc la vitesse tangentielle est elle aussi nulle. Cela engendre la création d'une zone appelée couche limite où le fluide est perturbé par la présence du mur. La taille de la couche limite dépend de la viscosité, une viscosité élevé implique une

couche limite épaisse. Le *wall shear stress* τ_w défini à **Eq. 2.9** est la contrainte de cisaillement exercée par le fluide sur la frontière solide. Dans le cas du VG, cette valeur indique l'effort appliqué par le fluide sur la barrière endothéliale du myocarde.

$$\tau_w = \mu \left. \frac{d\vec{u}}{d\vec{n}} \right|_{\partial\Omega} \tag{2.9}$$

Les couches limites ont une importance cruciale, elles localisent les zones de contrainte dans le fluide dues aux frottements avec les parois du domaine. Ces régions limites sont souvent instables et peuvent se détacher du mur. C'est le seul mécanisme de formation de vortex dans le cadre d'un fluide incompressible [64].

Mécanique des Fluides Numériques (MFN) Comme énoncé précédemment, l'**Eq. 2.8** ne pouvant pas être résolu analytiquement, il faut faire appel à un schéma numérique pour trouver une solution.

La Méthode des Volumes Finis (MVF) ou la Méthode des Eléments Finis (MEF) reposent sur la discrétisation mathématique de l'espace afin de résoudre localement les équations physiques. Cette discrétisation donne lieu à un maillage composé de sous-unités d'espace connectées les unes aux autres, c'est une construction topologique. Les équations physiques sont elles aussi discrétisées de sorte à pouvoir être résolues. Enfin, une solution globale est intégrée de manière itérative à partir des solutions locales [74]

Une étape de discrétisation temporelle est aussi nécessaire. L'écoulement, si résolu dans un régime transitoire, est calculé de manière discrète dans le temps. Un pas de temps inférieur aux variations locales du flux est nécessaire pour reconstruire de manière satisfaisante l'écoulement.

Du fait de la discrétisation et de la continuité imposée entre les éléments, les équations ne sont pas résolues parfaitement, il reste un résidu. Ce résidu quantifie l'erreur faite lors de la résolution locale, puis globale des équations physiques.

Enfin, il existe une grande diversité de stratégies numériques [74, 75] pour résoudre les équations physiques avec une complexité et un coût de calcul qui leurs sont propres. En fonction de la situation modélisée, certaines méthodes sont à privilégier.

La résolution d'un problème de MFN est souvent divisée en plusieurs étapes [75] :

- Pré-traitement ou *Set-Up* - définition du système étudié : géométrie, propriétés du fluide, régime d'écoulement, modèles physiques et méthode numérique
- Discrétisation - maillage du domaine

- Résolution - calcul itératif ou direct des solutions sur le domaine discrétisé.
- Post-traitement - exploitation de la solution : vérification de la cohérence, représentation graphique, calcul de paramètres quantitatifs (critère-Q, vorticité, etc.)

Des logiciels commerciaux et *open-source* existent pour la simulation numérique de phénomènes mécaniques complexes. La plupart ont intégré le *workflow* décrit ci-dessus au sein de leur produit. Détaillé au **Chapitre 3**, c'est la suite de logiciels *Ansys* qui est utilisée dans le cadre de ce projet. A titre d'exemple, *Ansys SpaceClaim* sert à créer la géométrie, *Ansys Meshing* permet de réaliser le maillage, *Ansys Fluent* sert à faire le *set-up* et résoudre le problème fluide et enfin *Ansys CFD-Post* sert au post-traitement de la solution.

2.4.2 Synthèse des performances relatives de différents modèles numériques issus de la littérature

La MFN regroupe plusieurs catégories de modèles numériques. Cette catégorisation des modèles se fait en accord avec la stratégie utilisée pour prendre en compte la variation géométrique au cours du temps et l'interaction entre le fluide et les parois du ventricule. Seront donc présentés les modèles à Géométrie-Prescrite (GP), les modèles à Interaction Fluide Structure Implicite ($IFSI$) et les modèles à Interaction Fluide-Structure (IFS). Chacune de ses catégories englobe des formulations 2D ou 3D. La complexité des ces modèles varie également en fonction du caractère simplifié ou réaliste de la géométrie. Une pratique courante est d'utiliser des données d'imagerie médicale EchoCG ou IRM comme référence pour réaliser des géométries patient-spécifiques. Une question se pose également quant à la modélisation des valves mitrale ou aortique, et la simulation d'autres cavités connexes au VG tel que l'atrium gauche et l'aorte. Les articles de Khalafvand et al. (2011) [13] et Hirschhorn et al. (2020) [32] ont permis de répertorier la majorité des articles mentionnés ci-dessous.

Géométrie-Prescrite (GP)

Les méthodes à Géométrie Prescrite (GP) reposent sur une stratégie où le mouvement de la paroi interne du VG est imposé comme condition limite. Cela implique l'utilisation d'une formulation *Arbitrary Lagrangian-Eulerian* (ALE) pour résoudre les équations de Navier-Stokes avec un maillage qui se déforme au cours de la simulation.

L'introduction de cette catégorie de modèle est faite par Vierendeels et al. (2000) [7]. Le remplissage diastolique du VG est modélisé en 2D axisymétrique. Comme énoncé précédemment, c'est une résolution par MVF avec une formulation ALE des équations de Navier-Stokes. La simulation commence avec la relaxation isovolumique, où le champ de vitesse est nul. La

relaxation est modélisée par un modèle de Meisner modifiant le module d'Young de la paroi. Lorsque la pression ventriculaire passe sous la pression atriale, alors la VM s'ouvre. Le profil de vitesse à la VM est calculé en amont pour prendre en compte l'effet de l'atrium sur le flux entrant. À la fois la relaxation du VG et la contraction de l'atrium sont modélisées, permettant d'observer respectivement l'apparition de l'onde E et l'onde A. Il y a observation de la formation de vortex et des distributions de pression associées, la **Figure 2.9** montre ses résultats en fin de diastole. Se distinguent deux vortex distincts. C'est une géométrie simplifiée en forme d'ellipse tronquée.

Saber et al. (2003) [76] propose un raffinement de la géométrie en utilisant des images IRM au cours du cycle cardiaque. Les flux mitral et aortique sont approximés par la variation du volume ventriculaire. En introduisant la géométrie des valves aortique et mitrale, l'article montre la sensibilité de l'écoulement à la forme des orifices d'entrée et de sortie du ventricule. Une comparaison avec des observations cliniques met en avant la cohérence des résultats obtenus. La précision du champ reconstruit est limitée par le manque d'information sur la géométrie des valves.

Baccani et al. (2002 et 2003) [77,78] propose une géométrie simplifiée dont la déformation est déterminée par une loi temporelle de la variation du volume basée sur des études cliniques. La géométrie est simplifiée par l'hypothèse d'axisymétrie. L'intérêt principal de l'étude est porté sur la sensibilité au diamètre de la VM qui est modélisé comme un cylindre d'épaisseur infinitésimale. L'analyse porte également sur la formation de vortex par adhésion à la paroi interne de la VM. Cette structure est ensuite éjectée dans le ventricule et créer des couches de vorticité à la paroi postérieure. Plus le diamètre diminue, plus la production de vortex est importante. La propriété mécanique des parois et la géométrie du ventricule semblent avoir une influence limitée sur le champ de vitesse étant principalement le résultat du transport des vortex dans le ventricule.

Long et al. (2003) [8, 79] propose une approche hybride pour la condition au niveau de la VM. Une partie de l'entrée est déterminée par un profil de vitesse et le reste par la pression pour satisfaire la conservation de la masse. La géométrie 3D et le déplacement des parois sont obtenus via des images IRM. Durant la systole, la condition à la VA est une pression nulle et la VM est considérée comme un mur. Durant la diastole, la VA est considérée comme un mur et la VM est déterminée par la condition hybride énoncée ci-dessus. Plusieurs cycles sont répétés. L'avantage et la conclusion de cette condition hybride est de pouvoir se rapprocher d'un flux physiologique tout en validant la conservation de la masse, voir **Figure 2.9**. L'importance de la condition limite à la VM sur l'écoulement simulé y est clairement visible.

Domenichini et al. (2005, 2007) [9, 47] présente un modèle 3D de l'écoulement dans le VG

durant la diastole et une étude expérimentale pour valider les résultats. La géométrie est basée sur une ellipse de révolution dont la déformation est déterminée par la forme du flux entrant, voir **Figure 2.9**. Dans cette étude, des paramètres physiques, tels que le nombre de Strouhal et de Stokes, semblent avoir une influence sur la progression du vortex dans la cavité ventriculaire. En essayant plusieurs variantes de géométrie, cela ne semble pas influencer significativement l'écoulement simulé. Contrairement à Long et al. (2003) [79], leurs résultats indiquent que l'utilisation d'un profil de vitesse qui tienne compte de la présence de la VM sans ajouter de condition sur la pression permet aussi d'obtenir des solutions physiologiquement cohérentes. Une limitation de l'étude est son initialisation avec un champ de vitesse nul alors que, physiologiquement, plusieurs cycles auraient déjà mis le fluide en mouvement dans la cavité ventriculaire. En collaboration avec Pedrizzeti (2005) [65], ils observent que le plan 3-chambres est un plan de quasi-symétrie de l'écoulement pour des patients sains.

Schenkel et al. (2009) [80] propose un modèle 3D patient-spécifique sur un cycle cardiaque complet à partir de données IRM en utilisant une formulation ALE. Ils introduisent une reconstruction surfacique des VM et VA adaptée au patient grâce aux mesures IRM. Le déplacement de la géométrie est interpolé en temps pour palier à la résolution temporelle de l'IRM. Les cavités aortique et mitrale sont aussi présentes. Les conditions d'entrée et de sortie sont définies par un choix de pression. Encore une fois, cette étude montre une cohérence globale avec des données physiologiques mais est extrêmement sensible au choix de la condition à la VM durant la diastole.

D'autres études de la catégorie *GP* patient-spécifique avec formulation *ALE* ont été répertoriées, chacune se différenciant par des hypothèses de dimension (2D ou 3D), le type de données d'imagerie utilisé (IRM ou EchoCG) et l'intégration ou non de valves. Sont répertoriés Doost et al. (2017 et 2016) [46, 81] avec un modèle 2D à partir d'images IRM avec un modèle de valves linéaires ; De Vecchi et al. (2014) [20] avec un modèle 3D sans valves à partir d'EchoCG 3D ; Moosavi et al. (2014) [82] avec un modèle 3D sans valves à partir d'images IRM ; Chnafa et al. (2014) [72, 73] avec un modèle 3D multi-cavités à partir d'images IRM intégrant un modèle de valves surfaciques par la méthode des frontières immergées.

Finalement, Bavo et al. (2016 et 2017) [45, 69] présente l'utilisation d'un modèle FSI 3D à partir d'images EchoCG 3D dont sont extraites la géométrie du VG et de la VM. Réalisé sur *Ansys Fluent*, le déplacement de la paroi est imposé et résolu avec une formulation ALE. Seule la diastole est étudiée, et la VA est considérée comme un mur. L'objectif est de réaliser une comparaison du développement des vortex entre 3 cas cliniques pathologiques. Les résultats montrent des écoulements distinctifs entre les cas cliniques.

Toutes ces études montrent le développement de cette catégorie de modèle numérique et

l'étendue de la personnalisation possible.

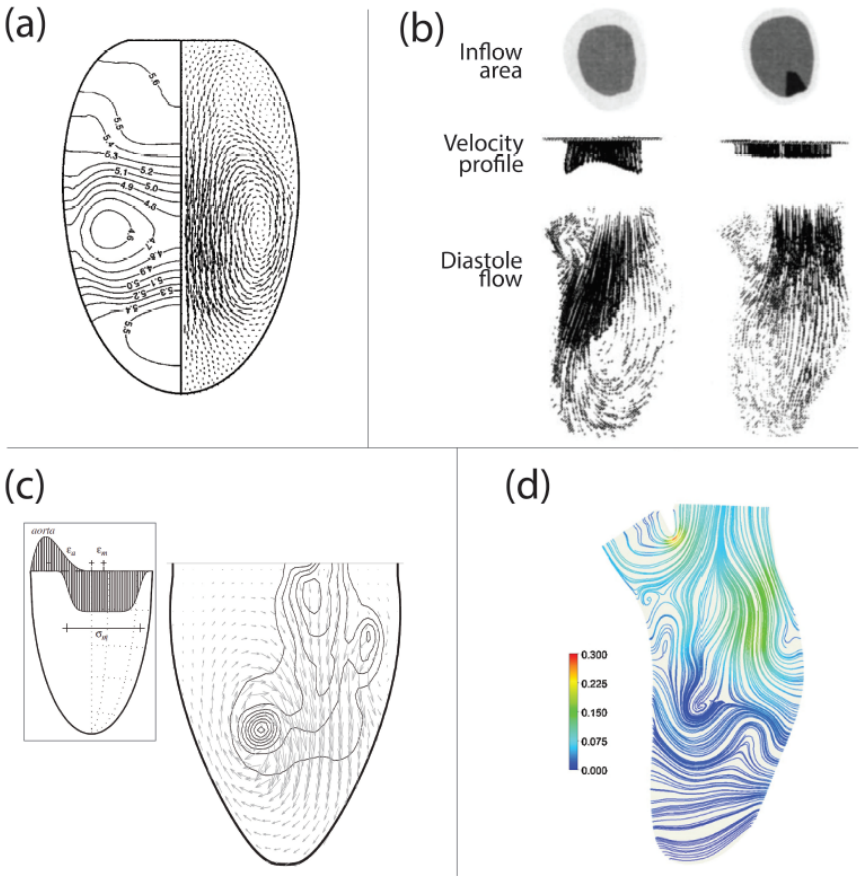


FIGURE 2.9 Exemples des écoulements simulés par différents modèles du VG basé sur une géométrie prescrite. (a) Répartition des pressions (gauche du VG) et des vortex (droite du VG) sur un modèle simple (Vierendeels et al. (2000) [7] *CCC License Agreement*). (b) Validation de conditions d'entrée couplées vitesse-pression (Long et al. (2003) [8] *CCC License Agreement*). (c) Validation de la personnalisation du profil de vitesse d'entrée du sang dans le VG (Domenichini et al. (2005) [9] *© 2005, Cambridge University Press*). (d) Profil de vitesse dans un VG patient-spécifique reconstruit à partir de l'IRM (Doost et al. (2016) [10] *CC 4.0*). Adapté de Dubois (2023) [1].

Interaction Fluide Structure Implicite (*IFSI*)

Les méthodes à Interaction Fluide Structure Implicite (*IFSI*) font principalement référence à la méthode des frontières immergées *(Immersed Boundary Method)*. Cette méthode est une simplification d'un couplage total, c'est une interaction fluide structure "fictive". Présentée par Mcqueen et Peskin (2000) [48, 61, 62], le fluide est représenté par une grille eulérienne de la pression et des vitesses, et la frontière est représentée par une structure élastique libre

de se déplacer par dessus la grille eulérienne. L'influence de la structure sur le fluide se fait par le calcul d'une force issue de sa déformation et est appliquée à la grille eulérienne dans le voisinage de la structure en utilisant la fonction Dirac. Il faut alors à nouveau utiliser le Dirac pour calculer la vitesse au niveau de la frontière sur la nouvelle grille eulérienne et la structure est déplacée à sa nouvelle position. Ce cycle est répété pour chaque pas de temps. La seule information nécessaire à ce modèle est la connaissance des propriétés du fluide, des propriétés élastiques de la frontière et de la géométrie initiale. Plus d'informations sur les raffinements présents dans la littérature sur cette méthode peuvent être trouvées dans le travail de Domenichini (2008) [83]. Le papier de Bao et al. (2017) [84] rentre en détails sur la conception mathématique d'une version récente de la méthode des frontières immergées, décrivant une formulation améliorant le respect de la conservation de la masse.

Bien que peu utilisé dans littérature, des cas d'application de la méthode pour intégrer la VA et/ou la VM ou bien pour des études cliniques, ont fait l'objet de publications. Tai et al. (2007) [85] propose un raffinement de la méthode originelle pour l'appliquer à des cas 3D d'écoulements complexes avec une prothèse mécanique de valve. Dans Chnafa et al. (2014) [72, 73], la méthode est employée pour intégrer l'influence de la VA et la VM à l'écoulement sans créer de barrière physique. De la même manière, dans Gao et al. (2017) [86], la méthode des frontières immergées est utilisée pour coupler la VM, le VG et le fluide. La géométrie y est reconstruite grâce à des images IRM au cours du cycle cardiaque. La spécificité est l'utilisation du tenseur de Piola–Kirchoff pour déterminer les forces élastiques de la structure avec des comportements non-linéaires. Les résultats concernent principalement les pressions atriales d'ouverture et les fractions d'éjection en fonction de la géométrie mitrale. Mangual et al. (2013) [11, 87] utilise ce couplage fictif pour l'étude de la diastole sur une géométrie extraite de données EchoCG en l'agrémentant d'une formulation pour la turbulence et le caractère non-Newtonien du fluide. Le modèle ne possède pas de VM ou de VA, l'entrée et la sortie ont un comportement ouvert-fermé et les flux entrant et sortant sont calculés par conservation de la masse. Le résultat de cette simulation directe est visible à la **Figure 2.10**.

Théoriquement exacte, cette approche est sensible à la qualité de l'information sur la géométrie et son déplacement. Des données de résolution spatiale et temporelle limitée fournissent en résultat une reconstruction approximative. Il faut théoriquement fournir les propriétés élastiques de la frontière, seulement en pratique, elle est approximée par un solide infiniment rigide dont seul le déplacement importe, créant un couplage partiel. La frontière agit sur le fluide, mais le fluide n'agit pas sur la frontière, c'est un couplage unilatéral.

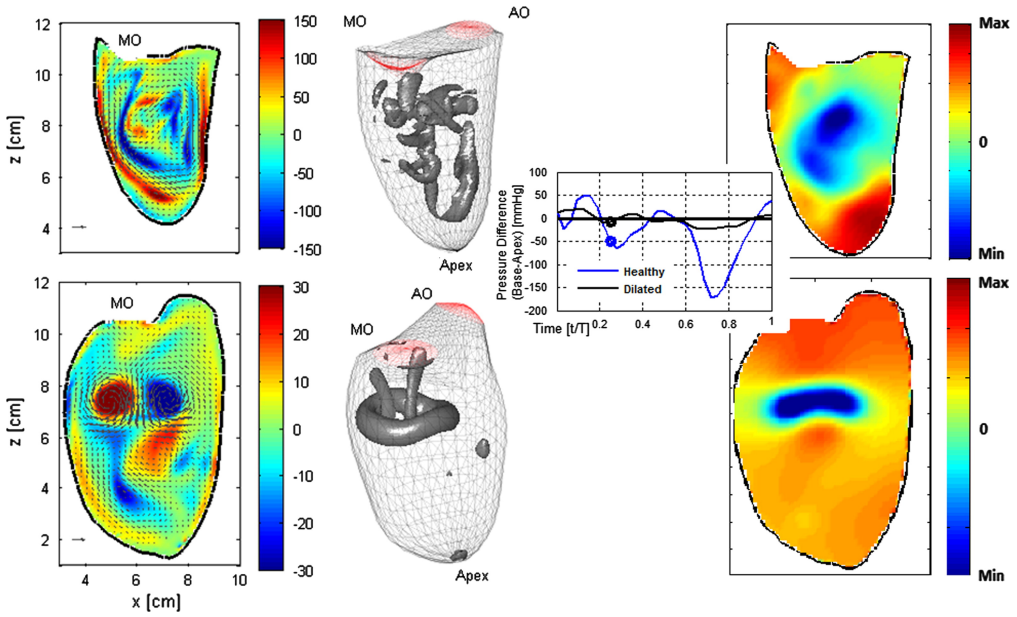


FIGURE 2.10 Écoulement dans le VG à mi-diastole. (Mangual et al. (2013) [11] *CCC License Agreement*)

Interaction Fluide-Structure (*IFS*)

Les méthodes à Interaction Fluide Structure (*IFS*) reposent sur la modélisation de la paroi solide par MEF et du fluide par MFN en assurant un schéma de couplage entre ces deux entités. Dans la quête du réalisme, cette approche semble la plus pertinente. Les propriétés mécaniques du myocarde étant complexes, cette méthode permet de mieux modéliser l'interaction entre le fluide et ses parois. Ces dernières années, des initiatives pour créer un jumeau numérique du cœur ont été lancées, et seule cette méthode permettrait une modélisation globale prenant en compte l'anatomie cardiaque, l'activation électrique, la réponse mécanique et l'hémodynamique. L'utilisation de la *IFS* s'applique à divers sujets comme l'étude des anévrismes, des prothèses de valves, des dispositifs d'assistance ventriculaire ou bien de l'athérosclérose. Cependant, ces perspectives ont un coût numérique très élevé et rendent son application généralement plus difficile. Pour certaines études, cela représenterait plusieurs mois de calcul avec des ressources disponibles en clinique.

La *IFS* se regroupe entre les couplages dits unilatéraux (*one-way*) ou totaux (*two-way*). Dans le premier cas, le solide a une influence sur le fluide, mais seulement, dans le couplage total, le fluide a une action réciproque sur le solide. De le cas unilatéral, la mécanique du solide est souvent mesurée ou calculée en premier, puis sert de condition limite au modèle fluide.

En améliorant la précision de l'écoulement au niveau de la paroi, les études *IFS* permettent, entre autres, d'étudier des problématiques où le taux de cisaillement est important, telles que la thrombogénicité et l'endommagement endothélial. En comparaison avec les catégories précédentes, celle-ci fait plus souvent l'objet de descriptions des contraintes à la paroi, puisqu'elles sont mieux résolues.

Rapidement, des travaux comme Watanab et al. (2004) [88] ont incorporé des mécanismes moléculaires et de stimulation électrique au comportement tissulaire dans des modèles 3D. L'objectif réussi est de modéliser le comportement physiologique globale du cœur.

Cheng et al. (2005) [12], dans un travail pour valider un logiciel commercial, présente un cas de résolution *IFS* du VG. La formulation est *ALE* et le solide est résolu par MEF. Le VG est approximé à une ellipsoïde de révolution avec un module d'Young variant dans le temps. Les caractéristiques de l'écoulement se rapprochent de données existantes. La description de la formation des vortex est en accord avec les études présentées précédemment. Le champ de vitesse à différents moments de la diastole peut être trouvé à la **Figure 2.11**.

La mécanique des valves, comme la VM ou la VA, est très difficile à modéliser. Or il a été montré que leur influence sur l'écoulement en aval est primordiale, tout particulièrement l'influence de la VM sur l'écoulement dans le VG. Dans cette optique, des articles ont proposé l'utilisation de solveur *IFS* pour résoudre l'interaction entre l'écoulement et la valve. En effet avec ce couplage la valve peut se fermer et s'ouvrir en fonction du gradient de pression entre ses faces. Cette modélisation de la dynamique des valves mime la réalité physiologique.

Par exemple, Dahl et al. (2012) [71, 89] propose la modélisation *IFS* entre deux solides et un fluide : la VMA, la VMP et l'écoulement. Le déplacement de la paroi est imposé et correspond à des acquisitions EchoCG. Dans cette étude, la VM est considérée comme axi-symétrique, et en l'absence de l'atrium, certaines caractéristiques attendues de l'écoulement ne sont pas visibles. Ceci marque l'importance pour l'étude du VG de prendre en compte l'asymétrie de la VM et la modélisation de l'atrium.

Contrairement aux approches *ALE*, Mao et al. (2017) [14] propose un modèle couplé *Smooth Particle Hydrodynamics* (*SPH*) incompressible pour le fluide et MEF pour le VG et la VM. La géométrie du VG est acquise sur image IRM et le modèle de VM généraliste est adapté à la physiologie du patient. Les résultats se concentrent sur la cinématique et les contraintes appliquées à la VM, voir **Figure 2.12**. Les vitesses de remplissage diastolique sont en accord avec des acquisitions EchoCG Doppler. Il remarque la formation de vortex lors de la fermeture des valves, soit disant accélérant le processus de fermeture. Une comparaison est faite avec un modèle similaire sans la présence de valves. Les résultats montrent un écoulement plus ordonné sans les valves avec l'absence de formation de vortex sur la VMA.

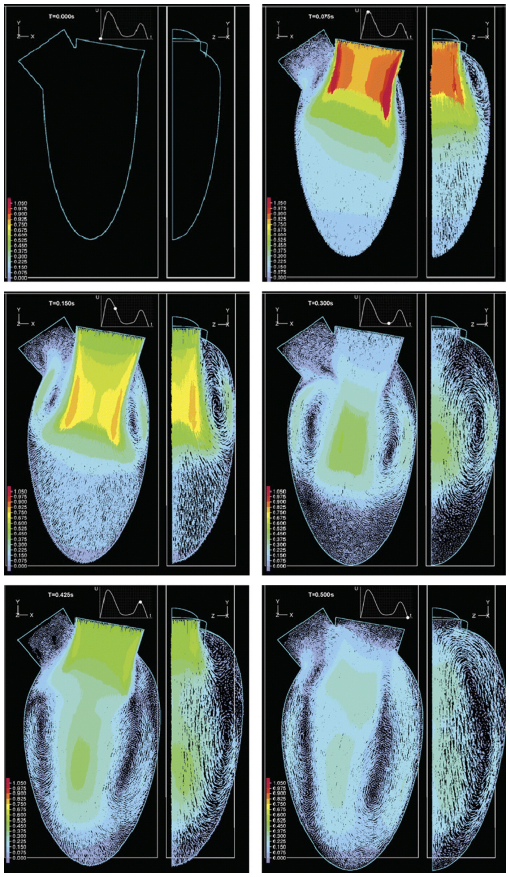


FIGURE 2.11 Champ vectoriel de la vitesse à divers moments de la diastole. (Cheng et al. (2005) [12] *Extrait de Khalafvand et al. (2011) [13] © 2011, reprinted by permission of Informa UK Limited, trading as Taylor & Francis Group, https ://www.tandfonline.com*)

Govindarajan et al. (2018) [15] propose une modélisation complexe 3D *two way IFS* patient-spécifique sur des données EchoCG. Le couplage concerne seulement la VM et le fluide. L'étude se concentre sur la phase diastolique. Durant l'onde E, ils ont répertorié des vitesses équivalentes à un Reynolds local de ∼5500 caractéristique d'un régime d'écoulement en transition de laminaire à turbulent, contredisant l'hypothèse laminaire généralement faite. Le flux d'entrée est imposé en utilisant la variation de volume du VG estimée par les images médicales et une condition de ratio de 1.35 entre les ondes E et A. Il y a observation de la formation d'un vortex annulaire se propageant dans la cavité donnant lieu à un schéma de recirculation dans la partie antérieure du VG en fin de diastole. La formation du vortex annulaire prend lieu dès l'ouverture de la valve avec une augmentation progressive du Reynolds local le long des valvules, atteignant un maximum local lorsque la position ouverte est atteinte. Dès lors, les couches de cisaillement se détachent et sont projetées dans l'écoulement principal, voir

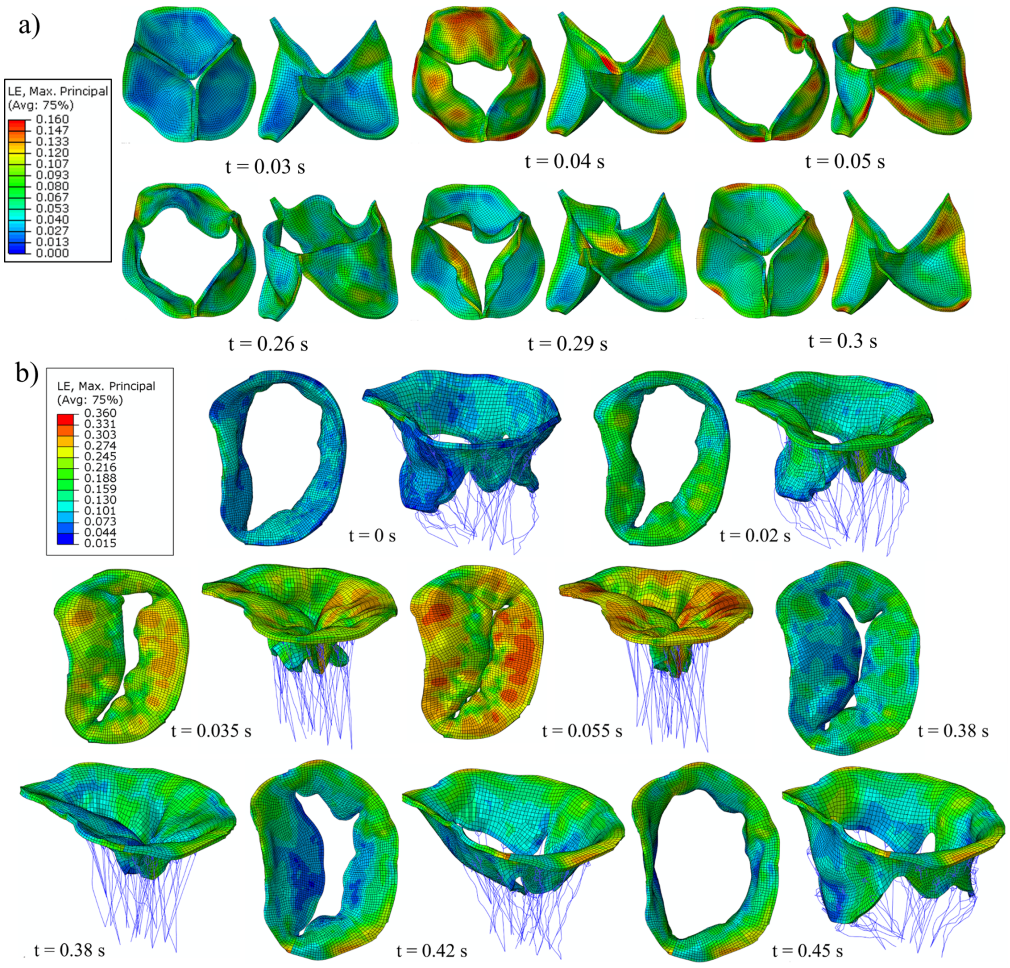


FIGURE 2.12 Déformation maximale principale sur les positions ouverte et fermée de la VA a) et VM b). (Mao et al. (2017) [14]) *CC 4.0*)

Figure 2.13. Ils observent une vitesse à l'extrémité de la valve proche de celle mesurée par EchoCG Doppler. Le travail rentre en détails sur la propagation du vortex principal au cours du temps et de son influence sur l'orientation du fluide vers la trajectoire de sortie du VG. Une limitation est que le fluide est initialement au repos dans le VG, et que seule la diastole est étudiée.

Enfin, dans la même approche que Govindarajan et al. (2018) [15], Khalafvand et al. (2019) [90] ajoute en plus le déplacement de l'anneau mitral. La VM est modélisée à partir de la mesure des angles d'ouverture dans le temps et de la position de l'anneau mitral. La simulation prend place sur un cycle cardiaque complet. Les résultats à propos des vortex sont en accord avec Govindarajan et al. (2018) [15]. Ce papier présente une comparaison entre une simulation avec et sans valves pour insister sur l'interaction de la VM sur le flux dans le VG. Sans valve,

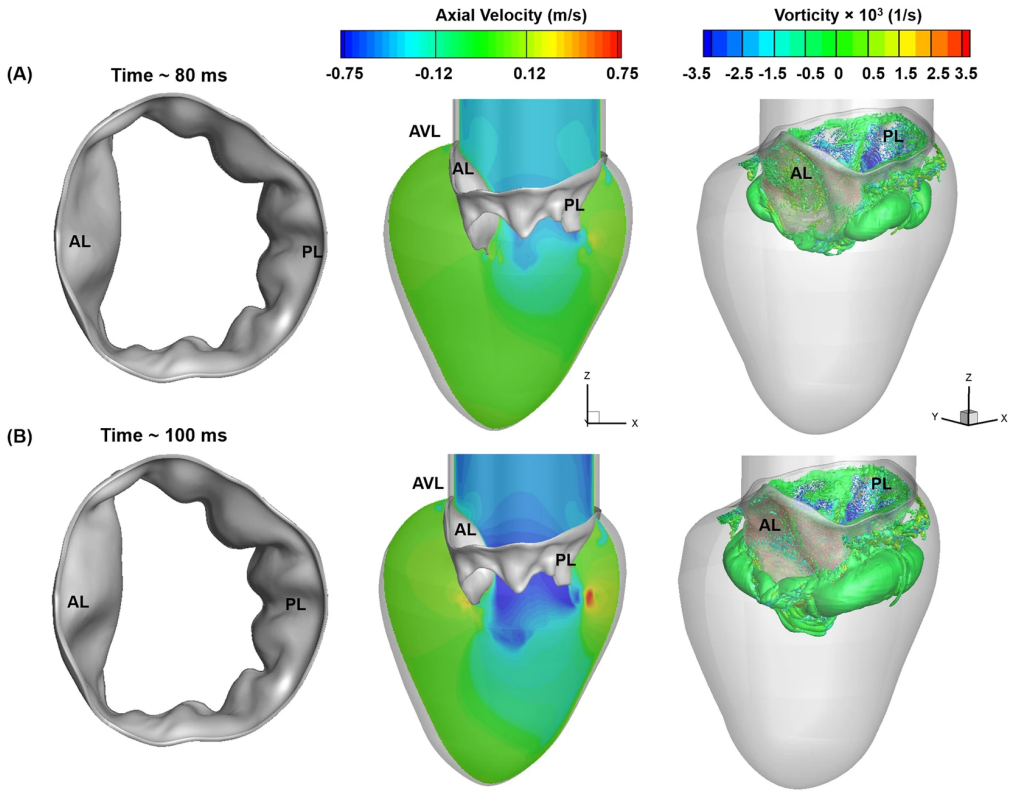


FIGURE 2.13 Déplacement de la VM, vitesse axiale et iso-surfaces de vorticité durant l'onde E à 80 ms et 100 ms. (Govindarajan et al. (2018) [15] *CC 4.0*)

ils observent l'apparition d'un vortex annulaire principal à l'onde E et un vortex secondaire à l'onde A. En l'absence de la VM, ils observent une faible dissipation des structures résultant en un écoulement dominé par ces deux vortex. La présence de la VM favorise la formation d'un vortex principal par la présence de zones de cisaillement sur les valvules mais aussi la propagation du vortex principal vers l'apex. L'asymétrie de la valve joue un rôle crucial dans l'orientation du vortex vers la face antérieure du VG. La valve augmente également le gradient de pression entre la base et l'apex favorisant le remplissage du VG.

Une limitation de Govindarajan et al. (2018) [15] et Khalafvand et al. (2019) [90] est l'absence de cordages tendineux pour la dynamique de la VM et de trabéculaires charnues sur la partie interne de la paroi ventriculaire.

Conclusion

Les catégories présentées ci-dessus sont à considérer comme des approches distinctes ayant leurs avantages et leurs limites. Les modèles les plus complexes semblent être les modèles

IFS à couplage total avec l'intégration de l'interaction entre les valves et l'écoulement. L'interaction du myocarde et du fluide est souvent utilisée pour la modélisation des mécanismes cellulaires et de l'activation électrique du tissu cardiaque. Les principales caractéristiques de l'écoulement dans le VG ont été discernées grâce à l'analyse des gradients de pression et des vortex intraventriculaires. Le gradient de pression entre la base et l'apex semble être un indicateur de l'efficacité du remplissage. Tandis que la formation d'un vortex lors de l'onde E se propageant au cours de la diastole en direction de l'apex, permet la mise en place d'une recirculation le long de la face antérieure du VG favorisant l'éjection lors de la systole et donc la fonction cardiaque. Il y a une tendance notable à utiliser des données d'imagerie médicale pour la reconstruction de géométrie physiologique. Moins utilisée que l'IRM, l'EchoCG trouve sa place surtout lorsqu'il s'agit d'obtenir une description fidèle de la VM par mesure transoesophagienne. Son principal défaut est sa résolution limitée pour décrire la géométrie du VG, cela est souvent pallié par une phase d'interpolation ou l'adaptation d'un modèle généraliste. Une liste non exhaustive de modèles utilisants l'EchoCG pour la modélisation patient-spécifique est disponible au **Tableau A.1** de l'**Annexe A**. Des hypothèses généralement acceptées dans le cadre d'étude 2D sont le caractère symétrique de l'écoulement par rapport au plan 3-chambres et l'absence de rotation du vortex annulaire autour de l'axe passant par le centre de la VM et l'apex du VG. Le régime laminaire est accepté dans la plupart des études [10, 12, 81, 89]. Le sang est largement considéré comme incompressible et Newtonien à l'exception de quelques études [10, 11, 87].

2.5 Synthèse de la revue de littérature

Cette revue de la littérature passe à travers les différents pans de connaissances nécessaires à l'élaboration d'un MFN patient-spécifique à partir d'image EchoCG, c'est-à-dire, une présentation de l'EchoCG, une introduction aux méthodes de segmentation disponibles et en développement pour le VG, un rappel de l'anatomie du cœur et de la description du cycle cardiaque, une revue détaillée des différentes catégories de modèles (*GP*, *IFSI*, *IFS*) et des principaux travaux de chacune d'entre elles.

Il ressort que l'étude de l'hémodynamique est un domaine complexe ayant fait l'objet de beaucoup de publications au cours des deux dernières décennies. Les stratégies de modélisation de l'écoulement du ventricule sont variées et les approches parfois distinctes. La complexité de la simulation est souvent corrélée avec une précision accrue de la simulation, le *gold standard* semblant être l'utilisation d'un modèle 3D *IFS* à couplage total patient-spécifique basé sur images IRM sur plusieurs cycles avec modélisation de la VM, la VA, l'atrium et l'aorte. Un tel modèle n'a pas été répertorié mais est la conclusion de la tendance générale des différentes

publications. Un tel modèle donnerait accès à une description de haute résolution de paramètres hémodynamiques dont l'importance clinique a été prouvée, par exemple le gradient de pression intraventriculaire et la dynamique de formation du vortex diastolique.

L'avantage des modèles numériques repose dans leur capacité à décrire précisément le phénomène cardiaque et à simuler des conditions pathologiques. Mais la complexité a un coût numérique important rendant l'utilisation de ces modèles sur de large cohorte difficile, voire impossible. En comparaison, des études cliniques sont plus largement menées directement au moyen d'imagerie médicale.

Certaines publications donnent à penser que ces modèles numériques seraient robustes à la comparaison inter-individuelle mais une étude statistique de grande ampleur serait nécessaire pour le prouver. À ce titre, l'utilisation de données EchoCG, facilement disponible en clinique, semble être un choix judicieux pour la reconstruction du VG. Contrairement à un couplage, l'intérêt étant porté sur l'écoulement sanguin uniquement, la méthode *GP* avec une formulation *ALE* semblerait un bon compromis entre complexité et robustesse. Du fait de l'hypothèse de symétrie de l'écoulement dans le VG, dans un premier temps, il est légitime de faire une étude 2D vue 3-chambres sur plusieurs cycles. De nombreuses études mettent en avant leur modèle numérique et leur application à l'étude de l'écoulement dans le VG, mais pas ou peu ont envisagé la conception de leur modèle comme un outil pouvant être utilisé en clinique.

En réponse aux objectifs énoncés au **Chapitre 1**, **Section 1.3**, les prochains chapitres porteront au **Chapitre 3** sur la structure générale de notre modélisation et de ses différentes variantes, au **Chapitre 4** sur l'étude de sensibilité des paramètres géométriques du modèle de valve mitrale et au **Chapitre 5** sur la validation du modèle vis-à-vis de la méthode iVFM et de l'EchoCG Doppler.

CHAPITRE 3 MODÉLISATION PATIENT-SPÉCIFIQUE DE L'ÉCOULEMENT DANS LE VG

Ce chapitre s'intéresse à la conception du modèle numérique, et est divisé en deux parties. La première partie à la **Section 3.1**, fait une description de la structure générale de la modélisation étape par étape. La seconde partie à la **Section 3.2**, vient présenter les différentes variantes étudiées, les compare et conclue sur celles retenues pour les études réalisées aux **Chapitres 4** et **5**.

Comme énoncé à la fin du **Chapitre 2**, le modèle réalisé est un modèle à *GP* 2D patient-spécifique à partir d'images EchoCG mode B 3-chambres avec intégration d'un modèle paramétrique de VM. Afin de remplir l'objectif n°1 énoncé à la **Sous-Section 1.3** de changement d'implémentation numérique, les modifications faites sont énoncées au fur et à mesure avec des références aux codes disponibles à l'**Annexe D**.

3.1 Structure générale de la modélisation

La conception du modèle suit la structure présentée à la **Figure 3.1**.

La **Sous-Section 3.1.1** présente le passage d'une géométrie extraite au préalable des images EchoCG à un modèle géométrique numérique du domaine étudié. La **Sous-Section 3.1.2** présente la stratégie utilisée pour obtenir un maillage mobile ainsi que les différentes techniques de remaillage et de lissage utilisées pour préserver la qualité du maillage tout au long de la simulation. Ensuite, le modèle paramétrique de la VM est défini à la **Sous-Section 3.1.3**. Les différentes conditions limites pour créer un problème numérique bien défini sont développées à la **Sous-Section 3.1.4**. Enfin les paramètres du solveur numérique sont évoqués à la **Sous-Section 3.1.5**.

L'**Annexe B** est un complément des informations présentées dans cette section, qui rentre plus en détails de l'implémentation numérique.

3.1.1 Création de la géométrie numérique

L'extraction de la segmentation du VG à partir des images EchoCG du projet Pétale est présentée dans une autre publication [1]. Le résultat de la segmentation est une série de courbes 2D formant le contour du VG aux différents pas de temps du cycle cardiaque. Cette partie s'intéresse à la transformation de cette segmentation en une géométrie numérique représentant notre domaine fluide pour le problème MFN. Le processus de création de la

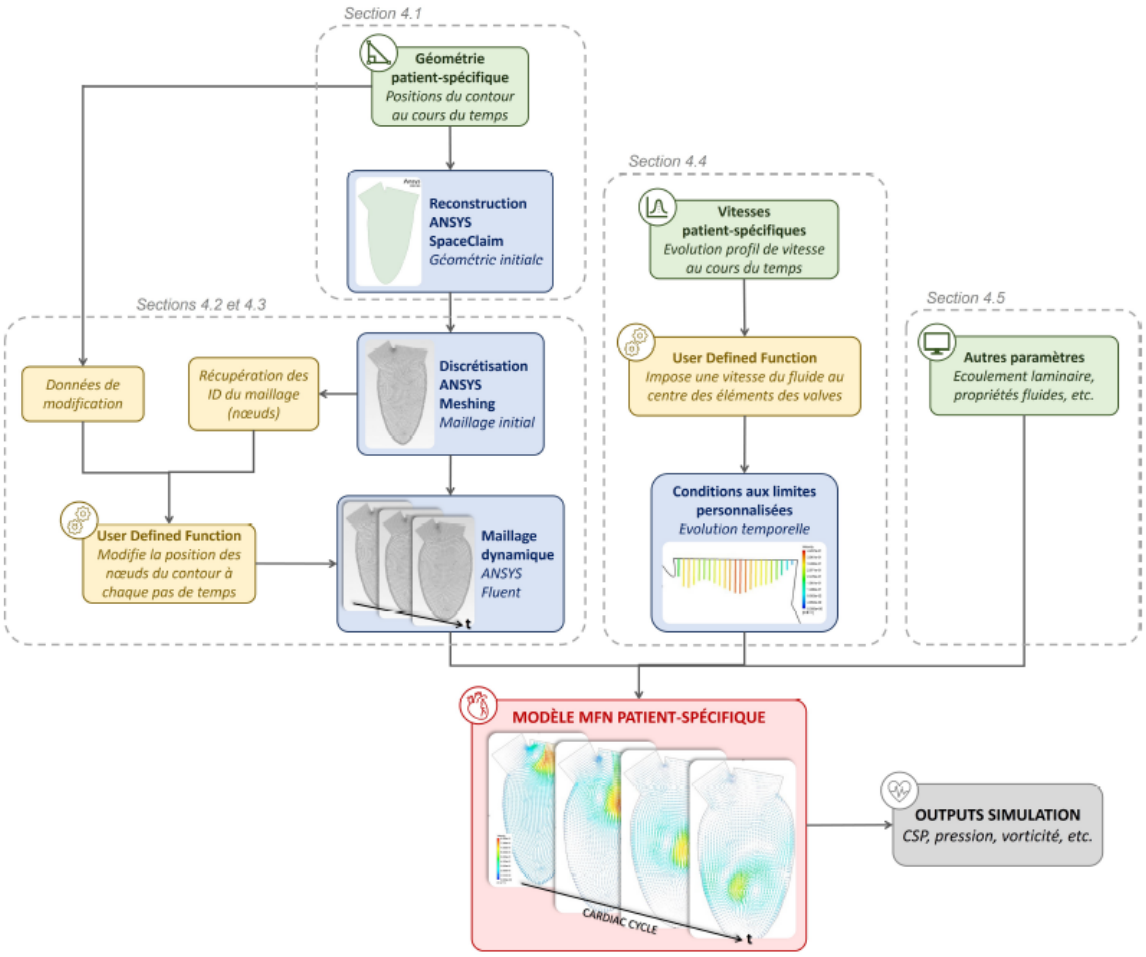


FIGURE 3.1 Démarche de conception du modèle numérique du VG sous ANSYS. (Dubois (2023) [1])

géométrie numérique est représentée à la **Figure 3.2**.

Le domaine initial correspond au VG au début de diastole (i.e. le VTS), autrement dit lorsque son volume est le plus faible. Le domaine sera déformé ultérieurement à la Sous-**Section 3.1.2** pour prendre en compte les mouvements du cœur.

Les courbes de contour de la segmentation initiale sont traitées et modifiées via une routine Python (**Code 5**) pour intégrer ultérieurement le modèle de VM et produire un fichier *curve_init.txt* lisible par *Ansys SpaceClaim*. L'en-tête du fichier correspond aux paramètres de reconstruction de *SpaceClaim*, puis l'ensemble des points des courbes définissant la frontière sont écrits selon la structure visible à la **Figure 3.3**.

L'ensemble des opérations faites par *SpaceClaim* sont ordonnées via un script Python (**Code 1**)

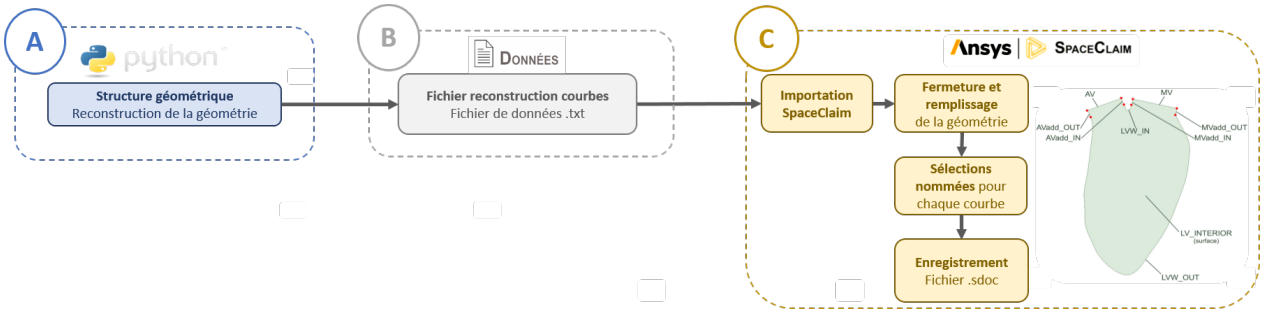


FIGURE 3.2 Processus de création du domaine numérique depuis la segmentation EchoCG dans *SpaceClaim* via PyAnsys. (Adapté de Dubois (2023) [1])

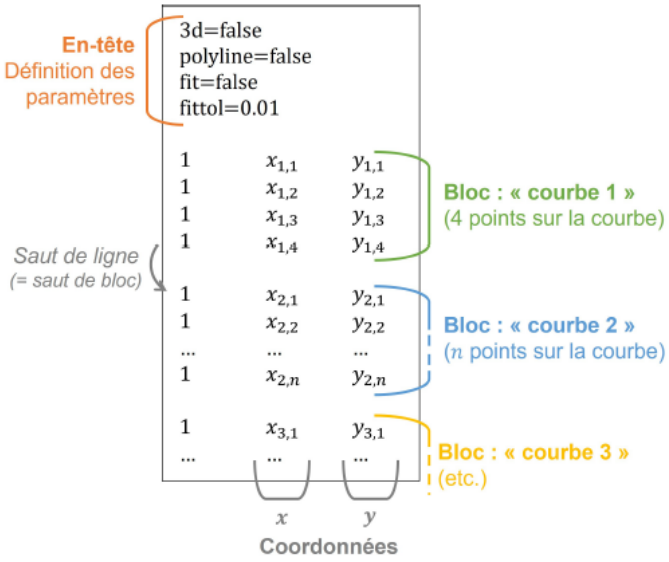


FIGURE 3.3 Structure d'un fichier (.txt) de "Points de Courbes". (Dubois (2023) [1])

utilisant le package API `from ansys.geometry.core import launch_modeler`. La frontière est importée, puis des "sélections nommées", voir **Figure 3.4**, sont créées pour identifier chaque portion de la frontière ce qui facilitera les étapes de maillage et de définition des conditions limites. Il s'agit finalement de combler l'espace interne connexe pour réaliser notre géométrie 2D. La géométrie est finalement sauvegardée en un fichier natif *geom_SCreconstruction_LL.sdoc*. Dans la version précédente, une routine MATLAB écrivait le fichier *curve_init.txt*, ouvrait *SpaceClaim* depuis le terminal et exécutait un script IRONPython regroupant l'ensemble des opérations afin de produire la géométrie finale et

l'enregistrer. Désormais avec PyAnsys, une seule routine Python modulable permet de gérer à la fois l'écriture des coordonnées de la frontière et l'interaction avec *SpaceClaim* pour réaliser la géométrie. Chaque opération est associée à une ligne de code et ne demande plus d'être regroupée.

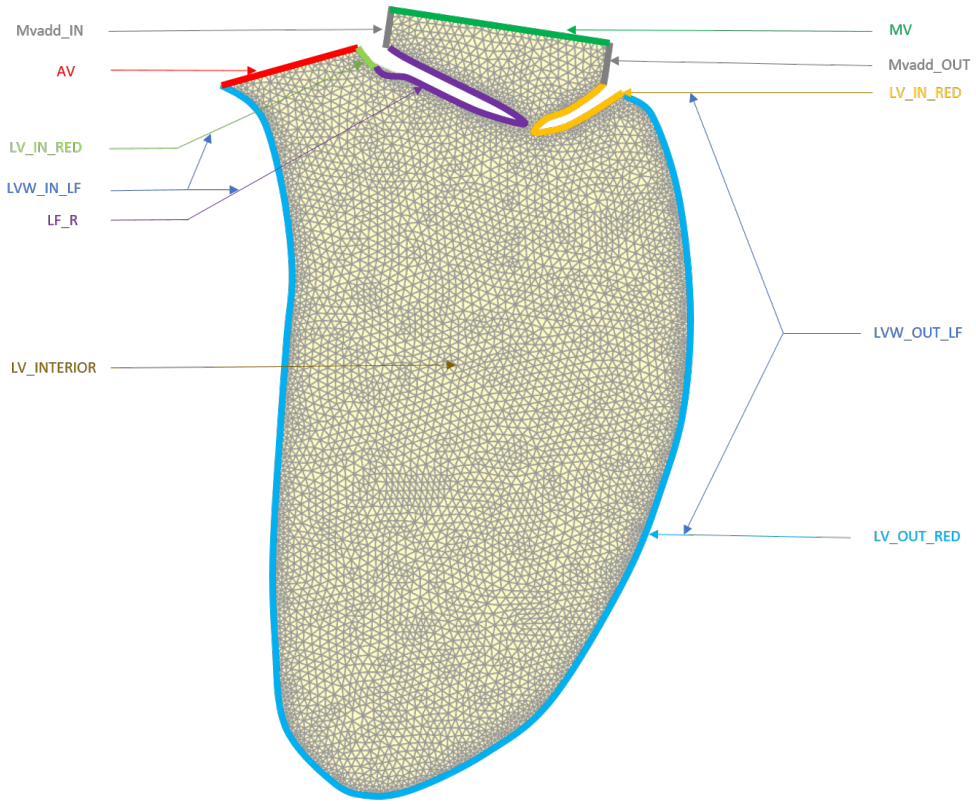


FIGURE 3.4 Schéma de présentation de la géométrie et des sélections nommées

3.1.2 Maillage dynamique

Le principe de la méthode à *GP* consiste à venir déformer le domaine géométrique au cours du temps. Le domaine initial est maillé, puis les nœuds de la frontière sont déplacés au moyen d'une *User Defined Function* (*UDF*). A chaque pas de temps, le domaine se déforme et le maillage est actualisé pour préserver sa qualité.

La première étape consiste à mailler le domaine initial créé à la **Sous-Section 3.1.1**. La routine PyAnsys (**Code 2**) fonctionne de la manière suivante :

- importer la géométrie *.sdoc* ;
- définir la taille caractéristique c_{interior} des mailles dans le domaine ;

— définir le type d'élément ;
— discrétiser le contour ;
— générer le maillage ;
— convertir au format natif *Fluent .msh* ;

La taille caractéristique c_{interior} du maillage est choisie de sorte à faire un compromis entre temps de calcul et qualité de la résolution, ce qui est détaillé ultérieurement.

Le contour est discrétisé en éléments de tailles égalent à c_{interior}, sauf "LV_IN_LF" et "LV_OUT_LF" qui sont de taille égale à $c_{\text{interior}}/2$. Ainsi, les bords de la VM et du myocarde font l'objet d'un raffinement local, voir **Figure 3.5**. Le choix est fait de ne pas utiliser de couches d'inflation aux parois à cause de la tendance des cellules quadratiques à produire des volumes négatifs lorsque le maillage se déforme.

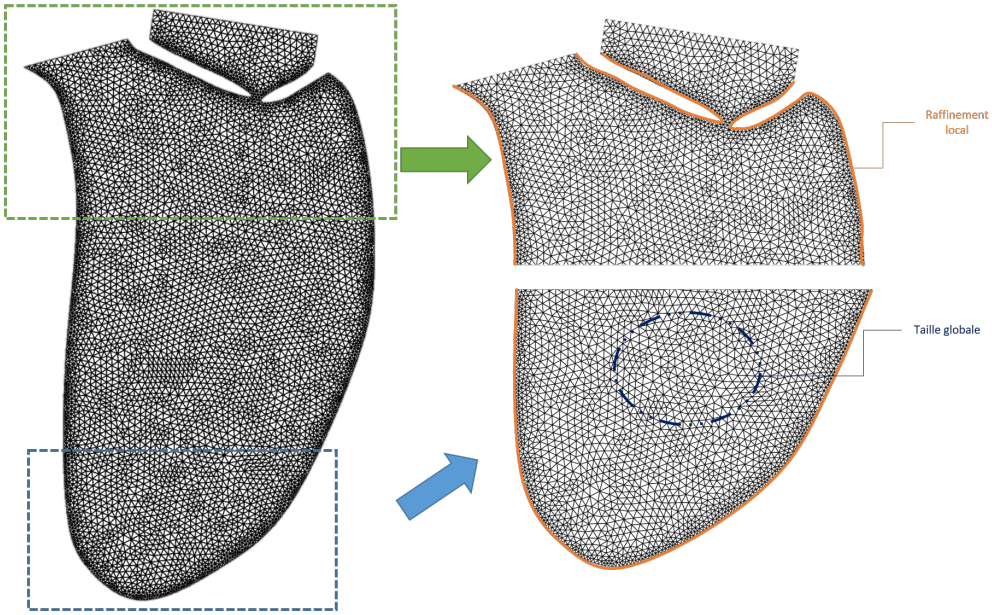


FIGURE 3.5 Exemple de maillage initial du VG. Présentation des stratégies de contrôle du maillage avec zoom des zones basale et apicale.

Cette étape est conduite en utilisant `from ansys.mechanical.core import launch_mechanical`.

Afin de déplacer les nœuds du contour, il est nécessaire de récupérer les identifiants des nœuds (nID) de la frontière. Ces nID permettent par la suite de prescrire une position à chaque nœud de la frontière. Puisque les nID sont initialisés lors de la résolution du maillage, il faut passer par **Fluent** via `import ansys.fluent.core as pyfluent` afin de les récupérer. Une *UDF* est utilisée pour récupérer le nID et les coordonnées de chaque nœud. L'implémentation

(**Code 3**) est détaillée à l'**Annexe D**.

Une fois les nID et la position initiale des nœuds exportée de *Fluent*, cette information est importée dans Python. La valeur de contour s, allant de 0 à 1 est calculée avec les nœuds du maillage. L'association entre le maillage et la segmentation se fait sur l'hypothèse que $s = \overline{s}$ est constante au cours du temps. Ainsi, chaque nœud est associé à son déplacement temporel extrait de la segmentation. Il est aussi possible d'artificiellement augmenter ou diminuer le nombre de pas de temps en réalisant une interpolation temporelle. Enfin, les fichiers *LV_OUT_motion.dat* et *LV_IN_motion.dat* détaillant les déplacements de chaque nœud à chaque pas de temps sont écrits. L'*UDF* permettant de lire ces fichiers et d'imposer le déplacement souhaité est également écrit à cette étape de la mise en place. L'implémentation (**Code 6**) est détaillée à l'**Annexe D**.

Déformer le maillage ne suffit pas, il faut le modifier à chaque pas de temps de sorte à éviter des anomalies géométriques pouvant altérer la résolution numérique. Il y a deux grandes catégories de modifications : le lissage et le remaillage. Le lissage consiste à faire pénétrer la déformation de la paroi au sein du domaine et éviter que seules les cellules du bords soient déformées. Ce lissage est fait en utilisant un schéma de diffusion défini par un paramètre de nombre maximum d'itérations et un coefficient de diffusion. Ces paramètres sont sensibles au ratio entre le déplacement de la paroi sur un pas de temps et la taille de la cellule, et doivent donc être choisis en fonction de c_{interior}. Le remaillage consiste à diviser ou à rassembler des cellules en fonction de critères sur la taille ou la dissymétrie des cellules. Le remaillage est effectué à chaque pas de temps, et les paramètres pour lesquels une cellule est remaillée sont la taille minimale, la taille maximale et la dissymétrie maximale de la cellule. Bien entendu, les tailles maximale et minimale dépendent linéairement de c_{interior} et des règles empiriques existent pour les estimer.

Le lissage est très important du fait du mouvement de compression et d'extension du VG. Le remaillage est critique pour l'ouverture et la fermeture de la VM. Cela permet de contrôler la densité de cellules entre les valvules au cours de la diastole. Les paramètres de remaillage sont optimisés empiriquement pour chaque valeur de c_{interior}, par exemple $c_{\text{interior}} = 1\ mm$, donne une taille minimale et maximale de la cellule valant respectivement 0.1 et 2 mm et une dissymétrie de 0.8. La taille minimale est critique lorsque le maillage se resserre et qu'il faut diminuer la densité de cellules, c'est ce qui se passe en amont de la VM en fin de diastole. La taille maximale permet de combler le vide lorsque le maillage est en extension, c'est ce qui se passe entre les valvules de la VM au début de la diastole. Une présentation de l'influence du remaillage et du lissage est disponible à la **Figure 3.6**. A force d'être modifié, le maillage est de plus en plus sujet à échouer lors de l'étape de remaillage. Des paramètres trop restrictifs

provoquent des erreurs fatales après un certain nombre de cycles.

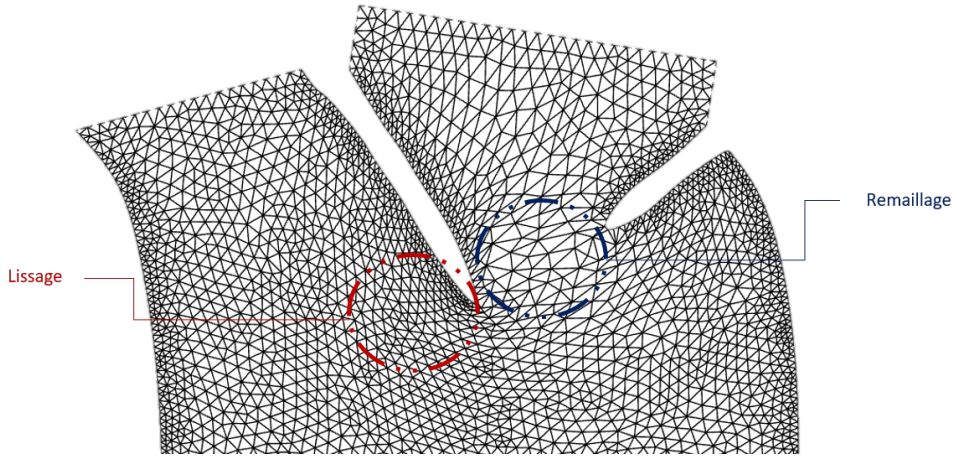


FIGURE 3.6 Exemple de l'influence du lissage et remaillage.

3.1.3 Modèle paramétrique de valve mitrale

Cette sous-section est dédiée à la conception du modèle paramétrique de VM.

Le modèle paramétrique se divise en deux parties : la reconstruction géométrique et la dynamique angulaire.

Les paramètres géométriques adimensionnels principaux sont $\beta_t = \frac{L_t^l}{L_t^l+L_t^r}$ et $\beta_n = \frac{L_n}{L_t^l+L_t^r}$ où L_t^l, L_t^r, L_n sont définis dans la **Figure 3.7**. Ces paramètres peuvent être extraits d'une image EchoCG à partir de 3 points, deux aux extrémités antérieure et postérieure de l'anneau mitral et un au point de fermeture de la VM à l'intersection des VMA et VMP. Les paramètres secondaires $(\theta_l, \theta_r, l_l, l_r)$, respectivement, Angle Mitral Antérieur (AMA), Angle Mitral Postérieur (AMP), Longueur Valvule Antérieure (LVA) et Longueur Valvule Postérieure (LVP) sont calculés à partir de (β_t, β_n). L'étude de sensibilité des paramètres (β_t, β_n) est réalisée au **Chapitre 4**. Les valeurs prises sont imposées et non extraites d'une mesure EchoCG comme ce sera le cas au **Chapitre 5** pour l'étude de performance et la validation du modèle. Les épaisseurs de la VMA et la VMP (w_l, w_r) sont en pratique difficiles à déterminer à partir de l'EchoCG et seront approximées à 2.5 mm. Il n'est pas attendu que l'épaisseur des valvules ait un impact avec l'utilisation de la méthode à GP, puisque le comportement mécanique de la VM n'est pas simulé. Un paramètre $\alpha \sim 0.9$ est posé arbitrairement de sorte que $l_j = \alpha.\tilde{l_j}$ où $\tilde{l_j}(\beta_t, \beta_n)$ est la longueur valvulaire calculée. Cela permet de créer un espace entre les

feuillets mitraux et de ne pas créer de chevauchements lors de l'ouverture et la fermeture de VM.

Il est possible de définir θ^l_{max} et θ^r_{max} qui sont les valeurs maximales admissibles pour AMA et AMP. Celles-ci sont établies à $\theta^l_{max} = \theta^r_{max} = 90°$ au **Chapitre 4**, et mesurées sur l'EchoCG au **Chapitre 5**.

Les détails de la reconstruction des valvules est disponible à l'**Annexe B**. Le principe repose sur l'utilisation de courbes de Bézier avec conditions sur les gradients aux extrémités pour créer une courbe guide qui est ensuite épaissie. Cette courbe centrale sert de référence pour créer des points de contrôle à une distance égale à la moitié de l'épaisseur. Les points de contrôle sont ensuite reliés en utilisant des splines d'ordre 3 pour reconstruire le contour de la valvule. Le point d'attache à l'extrémité de l'anneau mitral est fixe, mais le point d'attache au ventricule est mobile dans le temps.

Un couple (t, θ_l, θ_r) détermine la géométrie. Le choix des profils temporels $\theta_l(t)$ et $\theta_r(t)$ est crucial puisqu'ils définissent le comportement dynamique de la VM. Différents choix pour $(\theta_l(t), \theta_r(t))$ sont étudiés à la **Section 3.2**.

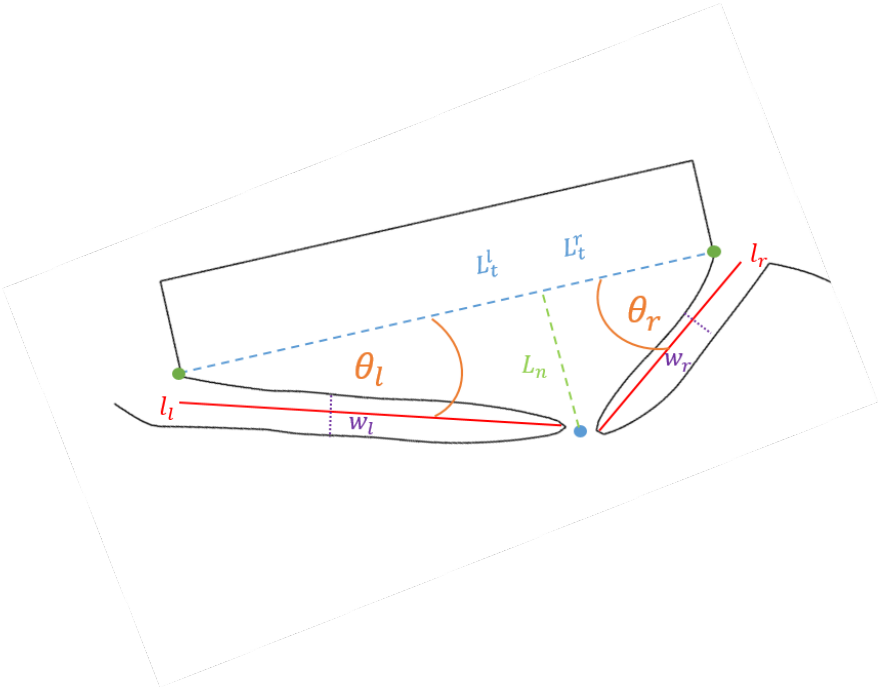


FIGURE 3.7 Schéma de reconstruction géométrique du modèle paramétrique de VM.

3.1.4 Conditions aux limites

En plus de la contrainte de déplacement, une condition de non glissement est appliquée à la paroi et sur les valvules mitrales. D'autres conditions manquent afin que le problème mécanique soit bien posé, principalement au niveau des zones d'entrée et de sortie du fluide, i.e. la VM et la VA.

À la différence de certains modèles, plusieurs cycles cardiaques sont simulés, les uns à la suite des autres. Il faut donc traiter séparément les phases de diastole et de systole. Dans la mesure où le fluide est supposé incompressible et que la mécanique du myocarde n'est pas modélisé, il est impossible de simuler la relaxation et la contraction isovolumique. Comme évoqué dans la revue de littérature, le VG est supposé être en quasi-équilibre de pression avec l'aorte pendant la systole et avec l'atrium pendant la diastole. La valeur de la pression n'a que peu d'importance puisque l'équation de Navier-Stokes (**Eq. 2.8**) est déterminée par le gradient de pression uniquement. Une gauge de pression dans le VG est donc établie à $P = 0\ Pa$.

Le choix a été fait de commencer le cycle cardiaque par la systole. Les phases systolique et diastolique requièrent des conditions aux limites différentes. L'entrée du fluide, i.e. l'*inlet*, du domaine est représentée par la sélection nommée "MV" située en amont du modèle paramétrique de valve. La sortie, i.e. l'*outlet*, est représentée par la sélection nommée "AV".

Systole Physiologiquement durant la systole, le VG se vide par la VA et la VM est fermée. La fermeture de la VM est assurée par la fonction *Gap Model* de *Fluent*. Elle prend en entrée une longueur d_g en-deçà de laquelle deux frontières sont reliées et dont la connexion est assimilée à un mur, voir **Figure 3.8**. La fermeture est parfaite dans le sens où aucun fluide n'est capable de passer, contrairement à d'autres modélisations faisant intervenir une résistance hydraulique artificielle pour bloquer l'écoulement.

Au début de la systole, le fluide dans la portion supérieure n'est pas au repos du fait de la fermeture de la VM en fin de diastole. Ce mouvement inertiel rend impossible d'assimiler "MV" à un mur, cela induirait des singularités de pression et augmenterait significativement les résidus de simulation. L'alternative utilisée est d'imposer une condition de pression d'entrée $P_{\text{inlet}} = 0$ similaire à la gauge de pression dans le VG. Ainsi, le fluide en amont de la VM est séparé du VG mais se comporte comme s'il communiquait avec une cavité externe lui permettant de revenir au repos.

Puisqu'aucune information est disponible sur la forme du flux de sortie au niveau de la sélection nommée "VA" et que l'écoulement est supposé se faire à quasi-égalité de pression, il

est retenu d'utiliser la condition de pression de sortie $P_{\text{outlet}} = 0$. Par respect de la conservation de la masse au cours du temps, le flux de sortie correspond à la variation géométrique du VG. Écrit mathématiquement, avec Q le débit volumique de sortie, $\Delta A = A(t + \Delta t) - A(t)$ la variation d'aire du VG et Δt le pas de temps :

$$Q(t) = \frac{\Delta A}{\Delta t}$$

Cette méthode s'est prouvée plus efficace d'un point de vue numérique que l'imposition d'un débit massique de sortie comme dans le modèle précédent [1]. Dans ce cas, le problème mécanique était surimposé ce qui ralentissait grandement la convergence de la solution à chaque itération.

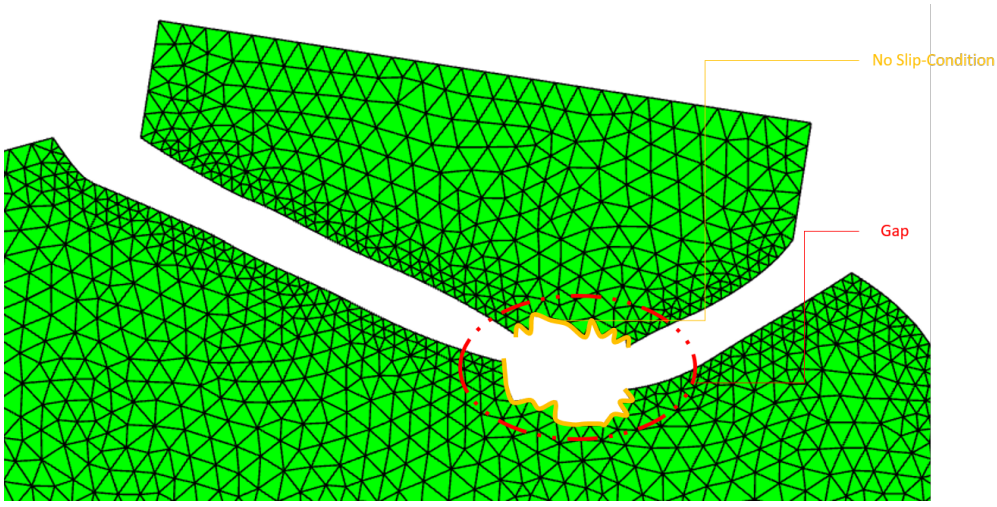


FIGURE 3.8 Illustation de la fermeture de la VM via la fonction *Gap Model* durant la systole.

Diastole Physiologiquement durant la diastole, la VA est fermée, la VM s'ouvre et le VG peut se remplir. Pour se faire, la sélection nommée "VA" est considérée comme une paroi avec condition de non glissement. Le *Gap Model* est désactivé, les valvules mitrales sont séparées et peuvent bouger naturellement. Il manque finalement une condition sur le flux d'entrée. La condition appliquée à la sélection nommée "MV" est un profil de vitesse constant de norme $U = Q(t)/d_{MV}$ avec d_{MV} le diamètre de la VM. Puisque la variation géométrique du VG est connue d'avance, $Q(t)$ est calculée d'avance pour tout le cycle cardiaque. Ainsi la loi temporelle $U(t)$ est écrite dans un tableau au format *.dat* et elle est importée en début de simulation dans *Fluent*. Ce fichier contenant la loi temporelle appelée Tableau Transitoire suit le formatage issue du *Fluent User's Guide 2024R1* illustré à la **Figure 3.9**.

L'implémentation (**Code 4**) de ces conditions limites est automatisée et les détails sont disponibles à l'**Annexe B**. La résolution se fait donc en deux étapes, d'abord la systole puis la diastole. La transition entre les phases est gérée par *Fluent* sans perturbation du flux.

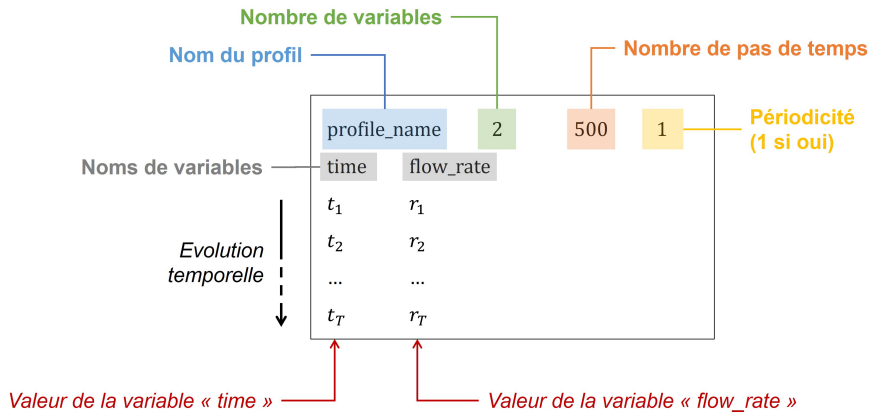


FIGURE 3.9 Exemple du formatage d'un Tableau Transitoire. (Dubois (2023) [1])

3.1.5 Paramètres de simulation

Cette dernière sous-section concerne l'ensemble des paramètres nécessaires à la configuration du solveur numérique. Ils définissent les équations résolues, le régime d'écoulement du fluide, le type de solveur utilisé, etc.

Cette configuration est réalisée en amont de la simulation comme décrit au **Code 4** en utilisant le langage Python et la libraire PyAnsys. Puisque les instructions sont réalisées en direct sans passer par un script, cela permet de créer un *workflow* naturel et flexible. Cela se reflète dans la configuration du solveur qui est séparée en plusieurs sous-parties, chacune faisant intervenir des quantités calculées en amont. Cette façon de faire rend le code beaucoup plus adaptable et versatile, ce qui est critique dans l'optique d'en faire un outil.

Les principaux paramètres concernant le solveur sont les suivants :

- Dimension : 2D ;
- Type de solveur : *Pressure-based* ;
- Schéma numérique : *Coupled* ;
- Précision : *Double* ;

Contrairement au modèle précédent [1], qui utilisait le schéma numérique SIMPLE, nous utilisons une formulation couplée. En simplifiant, cela veut dire que la pression et la vitesse

ne sont pas résolues séparément puis couplées, mais que les deux sont résolues ensembles. Ce schéma numérique est plus robuste et présente une meilleure convergence que le schéma SIMPLE lors de grandes transformations du domaine géométrique. Tout particulièrement, le schéma SIMPLE ne permettait pas de résoudre l'écoulement au début de la diastole lors de la phase d'ouverture de la VM. Les autres paramètres de base de *Fluent* ont été gardés, soient : un gradient de discrétisation spatiale basé sur les moindres carrés, une résolution de la pression et de la quantité de mouvement d'ordre 2 et un schéma implicite d'ordre 1 pour la formulation transitoire. Seul le paramètre *Flow Courant Number* du schéma numérique *Coupled* a été modifié. Plus celui-ci est grand, plus la convergence est atteinte rapidement en supposant que l'écoulement se rapproche du cas stationnaire. Une optimisation empirique de ce paramètre à été conduite pour trouver un compromis entre résilience et rapidité de convergence. Le choix de la précision *Double* est recommandé lors d'utilisation de maillage dynamique pour éviter les erreurs d'arrondies. Le nombre de pas de temps et la taille du pas de temps sont déterminés par la fréquence cardiaque et la résolution temporelle, ce qui fera l'objet d'une discussion ultérieure. Le nombre d'itérations maximal est posé à 100 itérations. Cependant, en général la convergence est satisfaite au bout de $\sim 20 - 50$ itérations, sauf lors de l'ouverture et la fermeture de la VM.

Les principales caractéristiques de l'écoulement sont les suivantes :

- Type d'écoulement : Écoulement transitoire ;
- Régime d'écoulement : Laminaire ;
- Modèle visqueux : Constant $\nu_s = 0.0035\ kg.m^{-1}.s^{-1}$
- Fluide : Newtonien, incompressible, masse volumique constante $\rho_s = 1050 kg.m^{-3}$;

Les résultats sont principalement exportés au format *.csv* puisque le post-traitement se fait via des routines Python pour des raisons de transparence et de cohérence du langage numérique. Certaines valeurs sont directement calculées par le solveur, voir les détails de l'implémentation (**Code 4**).

Pour conclure, les aspects qui n'ont pas été évoqués dans cette section ou n'apparaissent pas dans le **Code 4** prennent la valeur par défaut proposée par *Fluent*.

3.2 Présentation des différents modèles

Cette section est dédiée à la présentation de différentes variations du modèle initial concernant les conditions limites au niveau de la VM durant la diastole. Un premier modèle est présenté où la dynamique angulaire de la VM est non contrainte utilisant une loi angulaire arbitraire en échelon. Le second modèle est un raffinement du premier au sens où la

loi angulaire est désormais contrainte pour assurer une dynamique mitrale physiologique. Finalement, le dernier modèle est une extension du second modèle avec une augmentation artificielle du flux diastolique d'entrée dans le VG. Après les avoir présentés, les écoulements simulés sont comparés. Une conclusion est apportée quant à quels modèles seront utilisés aux **Chapitres 4** et **5**.

3.2.1 Modèle n°1 : Ouverture Mitrale Non Contrainte (*OMNC*)

Comme évoqué précédemment à la **Sous-Section 3.1.3**, le modèle paramétrique de VM se divise en la partie géométrique et le comportement angulaire. Le comportement angulaire est décrit par une loi temporelle pour chacune des valvules mitrales. Il est à l'origine de la dynamique de l'écoulement lors de l'ouverture et la fermeture de la VM. La première approche consiste à appliquer un profil en échelon simultané pour la VMA et VMP. Un tel profil temporel est disponible à la **Figure 3.10**. Ce profil est caractérisé par des périodes d'ouverture et de fermeture $dt_{\text{opening}} = dt_{\text{closing}} = \frac{\Delta T_{\text{diastole}}}{5}$, et un angle maximal et minimal d'ouverture ($\theta_{max} = 90,\ \theta_{min} = 10$). Les périodes d'ouverture et de fermeture sont une approximation empirique faite sur l'observation de plusieurs enregistrements EchoCG. La valeur de θ_{max} est la limite géométrique de l'algorithme pour reconstruire la VM. La valeur de θ_{min} est volontairement non nulle, puisque physiologiquement, lorsque la VM est fermée, les feuillets mitraux ne sont pas alignés.

Par la suite, cette formulation est qualifiée d'Ouverture Mitrale Non Contrainte (*OMNC*).

La particularité de cette formulation est que le mouvement mitral est superposé au mouvement ventriculaire. Elle est par essence arbitraire puisque la dynamique mitrale n'essaye pas d'être synchronisée avec la dynamique ventriculaire.

3.2.2 Modèle n°2 : Ouverture Mitrale Contrainte (*OMC*)

La seconde formulation à Ouverture Mitrale Contrainte (*OMC*) est le prolongement de *OMNC*. La principale hypothèse est que la dynamique mitrale $(\theta_l, \theta_r)(t)$ est reliée au flux entrant dans le VG ϕ_e et par conséquent à sa variation géométrique au cours du temps $\Delta A(t)$. Comme illustré à la **Figure 3.11**, l'onde E (*E-Wave*) est définie par un premier maximum local d'aire, suivit d'un minimum local à la diastase (*Diastasis*) et à nouveau un maximum local à l'onde A (*A-Wave*). Puisque $\Delta A \propto U \propto \phi_e$, la diastole se caractérise par une première accélération de l'écoulement durant l'onde E, suivie d'un ralentissement à la diastase puis d'une seconde accélération avec l'onde A.

À partir des images EchoCG, une description réaliste est l'ouverture totale avant le pic de

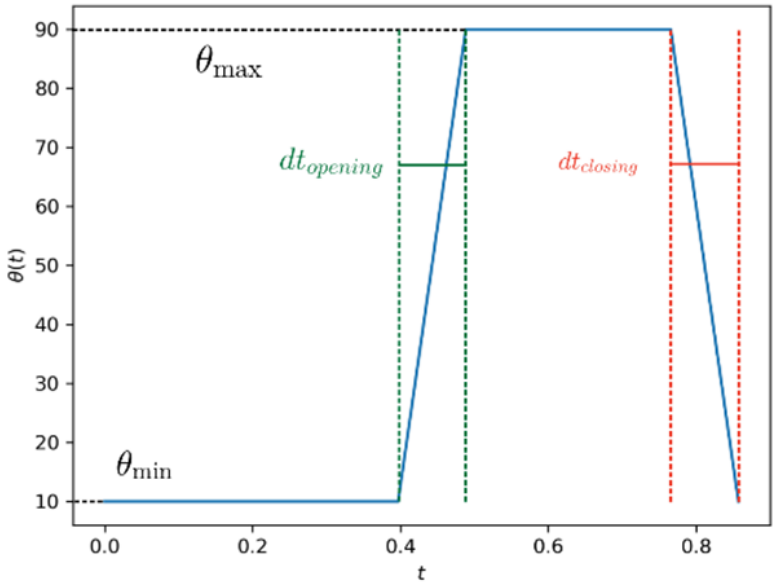


FIGURE 3.10 Profil Angulaire du modèle *OMNC*.

l'onde E, une fermeture partielle durant la diastase, une réouverture au pic de l'onde A et une fermeture totale après l'onde A. Une description simplifiée est que la VM s'ouvre du début de la diastole jusqu'au pic de l'onde E, reste ouverte jusqu'au pic de l'onde A et se referme.

Un premier critère est d'assurer que la dynamique mitrale coïncide avec l'une ou l'autre de ces descriptions.

Le second critère contraint l'ouverture angulaire de sorte qu'il n'y ait pas d'écoulement du VG vers l'atrium durant l'ouverture de la VM. Divisons le volume ventriculaire en deux sous-domaines, V_{up} la partie en amont de la VM et V_{down} en aval, voir **Figure 3.12**. De manière géométrique, $V_{up} = V_{up}(\theta_l, \theta_r)$, i.e. le volume du sous-domaine, est une fonction des angles des feuillets mitraux.

Par conservation de la masse dans le sous domaine V_{up}, alors :

$$\Delta V_{up} = (\phi_e - \phi_s).dt = \phi_e(1 - f_c(\phi_e)).dt \tag{3.1}$$

Où ΔV_{up} est la variation de volume du sous-domaine, $\phi_e = \frac{\Delta A}{\Delta t}$ le flux entrant, ϕ_s le flux sortant. Puisque ϕ_s est inconnu, la fonction f_c est définie de sorte que $\phi_s = \phi_e.f_c(\phi_e)$. Une

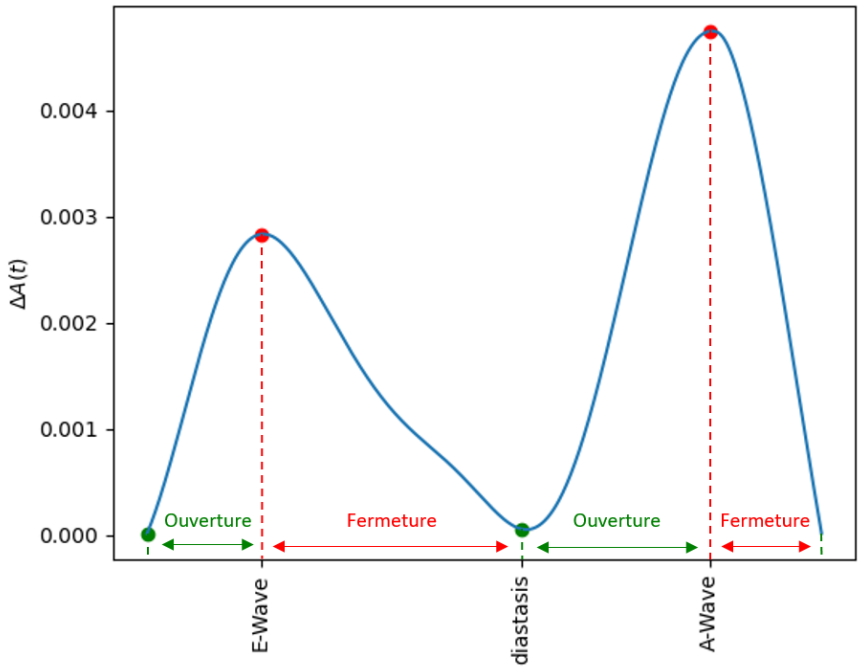


FIGURE 3.11 Description des phases d'ouverture et de fermeture de la VM durant la diastole basé sur la variation géométrique.

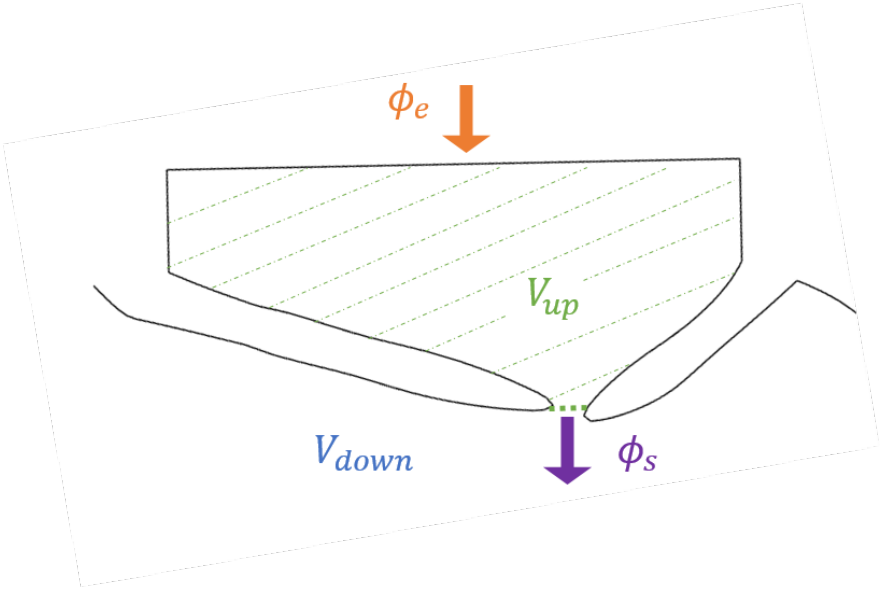


FIGURE 3.12 Schéma de définition des sous-domaines ventriculaires.

première observation est que la condition qui évite l'apparition d'un flux régurgitant est $\phi_s \geq 0$ ce qui par l'**Eq. 3.1**, donne une borne supérieure $\Delta V_{up} \leq \phi_e.dt$. La borne maximale $\Delta V_{up} = \phi_e.dt$ pour l'ouverture de la VM est atteinte avec $\phi_s = 0$. Cela se traduit par le fait que tout le flux entrant dans le sous-domaine ne sert qu'à compenser sa variation de volume. La seconde observation est qu'en cas de fermeture de la VM, i.e. $\Delta V_{up} \leq 0$, alors par l'**Eq. 3.1** $\phi_s \geq \phi_e$, ce qui se traduit par une accélération du flux de sortie lors de la fermeture de la VM. Dernière observation, si $\Delta V_{up} \leq 0$ alors $0 \leq f_c(\phi_e) \leq 1$ et si $\Delta V_{up} \geq 0$ alors $f_c(\phi_e) \geq 1$.

Donc, afin d'avoir une ouverture mitrale la plus rapide sans flux régurgitant durant l'ouverture, le critère $f_c(\phi_e) = 0$ est imposé. Durant la fermeture, afin de contrôler l'accélération du flux sortant, i.e. $\phi_s - \phi_e \leq \alpha.\phi_e$, alors $f_c(\phi_e) = 1 + \alpha$.

Ce qui se réécrit par :

— Ouverture : $\Delta V_{up} = \phi_e.dt$
— Fermeture : $\Delta V_{up} = -\alpha\phi_e.dt$
— Sinon : $\Delta V_{up} = 0$

Le problème étant que les phases d'ouverture et de fermeture ne sont pas intrinsèquement définies.

Stratégie *Naturelle* La première stratégie dite *Naturelle* se base sur l'hypothèse n°1 que la VM s'ouvre du début de la diastole jusqu'au pic de l'onde E, se referme jusqu'à la diastase, s'ouvre à nouveau jusqu'au pic de l'onde A et se referme ensuite progressivement, voir **Figure 3.11**. Ainsi, V_{up} augmente jusqu'à l'onde E, où $V_{up} = V_{up}^E$, diminue jusque la diastase où $V_{up} = V_{up}^D$, augmente à nouveau jusqu'à atteindre à nouveau $V_{up} = V_{up}^E$, puis reste constante avant de redescendre après l'onde A au volume initial $V_{up} = V_{up}^0$. Une borne maximale $V_{max} = V_{up}(\theta_{max}^l, \theta_{max}^r)$ est rajoutée, qui correspond à l'ouverture maximale admissible pour la reconstruction du modèle paramétrique de VM. La valeur V_{up}^D est donnée par une combinaison linéaire qui mime la tendance extraite de la variation géométrique du VG. Autrement dit, si $\lambda = \frac{\Delta A^D - \Delta A_0}{\Delta A^E - \Delta A_0}$ alors $V_{up}^D = \lambda V_{up}^E + (1-\lambda)V_{up}^0$ avec ΔA^D,ΔA^E et ΔA_0 qui sont respectivement les variations d'aire intraventriculaire au moment de la diastase, de l'onde E et du début de la diastole. Physiquement, cela impose que la fermeture mitrale est du même ordre de grandeur que le ralentissement du flux entrant durant la diastase. Enfin, le paramètre α est calculé pour chaque phase de sorte à ce que la fermeture soit progressive entre V_{up}^E et V_{up}^D puis entre V_{up}^E et V_{up}^0. La variation temporelle de V_{up} est illustrée à la **Figure 3.13**.

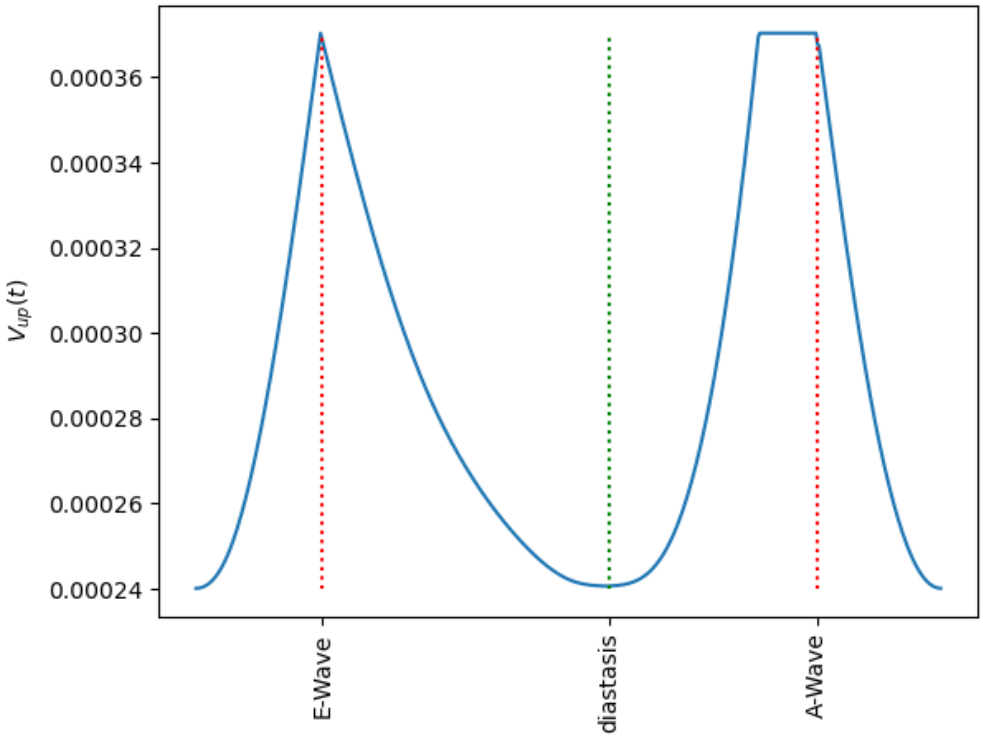


FIGURE 3.13 Profil temporel de V_{up} obtenu avec la stratégie *Naturelle*.

Stratégie *Forcée* La seconde stratégie repose sur l'hypothèse que la VM s'ouvre du début de la diastole jusqu'au pic de l'onde E, reste ouverte jusqu'au pic de l'onde A et se referme ensuite. Autrement dit, la variation angulaire lors de la diastase est omise. La valeur de V_{up}^{E} est calculée de manière similaire à la stratégie précédente. Seul un paramètre α est calculé de sorte à ce que la fermeture soit progressive entre V_{up}^{E} et V_{up}^{0} après l'onde A. La variation temporelle de V_{up} est illustrée à la **Figure 3.14**.

Une fois l'ouverture mitrale contrainte calculée, il faut procéder à la résolution du problème inverse $V((\theta_l(t), \theta_r(t)) = V_{up}(t)$, autrement dit trouver les angles des feuillets mitraux donnant le volume souhaité. Une optimisation est faite à chaque pas de temps avec comme fonction de coût :

$$L(\theta_l^{t+1}, \theta_r^{t+1}) = V(\theta_l^{t+1}, \theta_r^{t+1}) - V_{up}^{ref} + \epsilon * \left\| \begin{pmatrix} \theta_l^{t+1} \\ \theta_r^{t+1} \end{pmatrix} - \begin{pmatrix} \theta_l^{t} \\ \theta_r^{t} \end{pmatrix} \right\|^2$$

On rajoute un second membre pour assurer une continuité temporelle des mouvements angulaires de la VM. L'optimisation est bornée et repose sur la méthode *L-BFGS-B*. Une illustration du résultat de cette étape est disponible à la **Figure 3.15**.

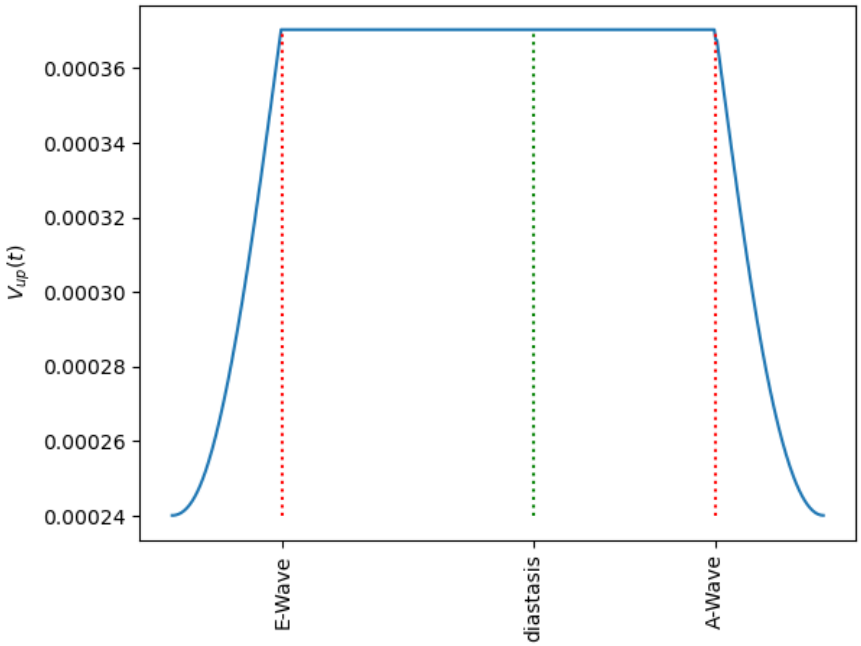


FIGURE 3.14 Profil temporel de V_{up} obtenu avec la stratégie *Forcée*.

L'implémentation du **Code 9** pour la loi temporelle de V_{up} et du **Code 10** pour l'optimisation de (θ_l, θ_r) est détaillée en **Annexe D**.

3.2.3 Modèle n°3 : Flux Diastolique Amplifié (*FDA*)

Les modèles *OMC* et *OMNC* reposent sur l'hypothèse que le flux diastolique entrant et la variation géométrique dans le plan sont proportionnels, i.e. $\phi_e \propto \Delta A$. Or, il a été observé que cette approche tend à sous estimer le flux entrant. En effet, physiologiquement un écoulement transverse au plan de symétrie 3-chambres prend place et n'est donc pas pris en compte. Cette sous-estimation a un impact direct sur l'écoulement.

Pour le modèle à Flux Diastolique Amplifiée (*FDA*), nous supposons que le flux entrant est sous-estimé mais que la dynamique angulaire de la VM telle que traitée dans *OMC* et *OMNC* reste valide. Autrement dit, l'ouverture et la fermeture sont bien reliées à la déformation dans le plan 3-chambres mais le flux d'entrée est lui sous-estimé. Ce flux ϕ_e est artificiellement augmenté, et donc l'est aussi l'amplitude du profil de vitesse constant entrant U par la VM durant la diastole. Cette augmentation est réalisée par un facteur multiplicatif a_U de sorte que la nouvelle amplitude est donnée par $\tilde{U} = a_U.\frac{\phi_e}{d_{MV}}$. La dynamique angulaire est la même que celle évoquée dans le modèle *OMC*.

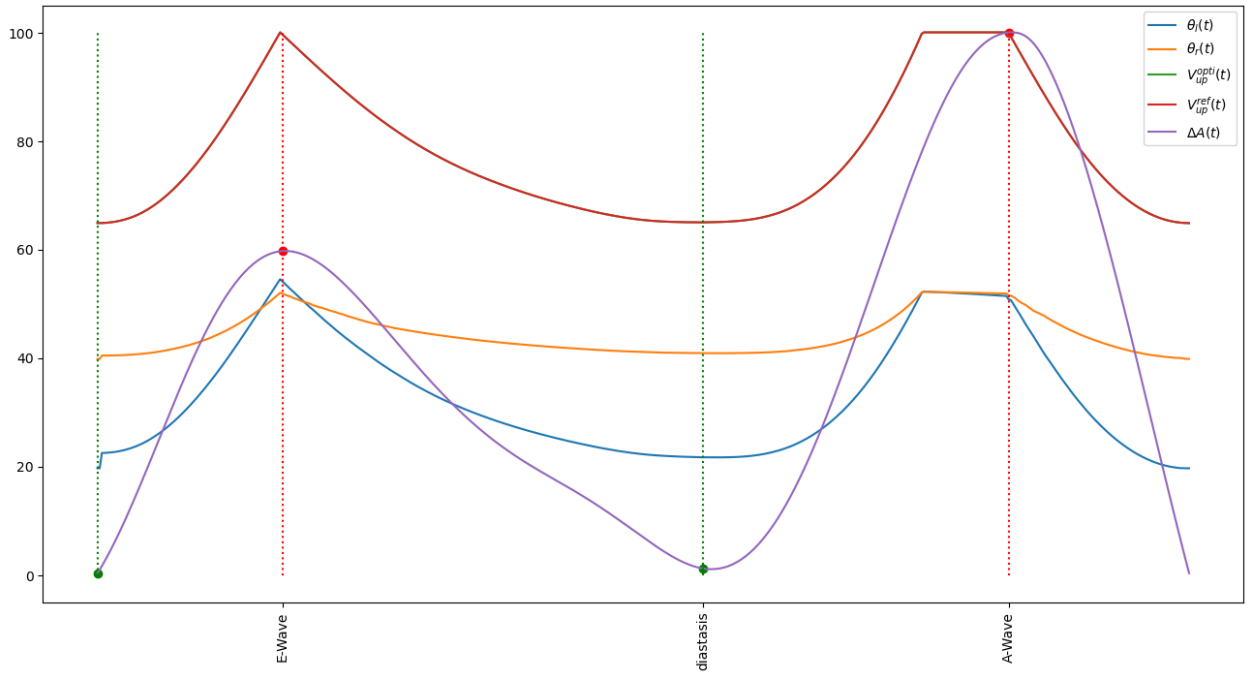


FIGURE 3.15 Problème inverse d'optimisation des angles dans le cas de la stratégie *Naturelle*. $\Delta A; V_{up}^{ref}; V_{up}^{opti}$ sont rendues adimensionnelles.

Cette amplification sera principalement utilisée dans le **Chapitre 5**. Le paramètre d'amplification rend possible de recaler le flux entrant avec la mesure de l'EchoCG Doppler, autrement dit $a_U = \frac{U_{max}^{Doppler}}{U}$ où $U_{max}^{Doppler}$ est le maximum de la vitesse radiale mesurée sur l'EchoCG Doppler *unaliased* durant tout le cycle cardiaque.

Le problème de cette technique est qu'elle viole ouvertement le principe de conservation de la masse. En imposant un tel flux, le solveur doit rétablir l'équilibre local en perdant artificiellement de la masse de manière diffuse dans le domaine. L'impact sur les résidus reste assez faible (inférieur à 10^{-3}), même pour de grosses amplifications du flux. L'influence est mesurée par la valeur moyenne de la variable *mass imbalance*.

3.3 Comparaison des performances des différents modèles et conclusion

Cette sous-section est dédiée à la comparaison des écoulements simulés par les différents modèles *OMNC*, *OMC* et *FDA*.

Une étude préliminaire est réalisée avec le modèle *OMNC* afin de déterminer la résolution spatiale et temporelle adéquate pour obtenir une reproduction fidèle des principales caracté-

ristiques de l'écoulement tout en préservant un temps de calcul limité.

Tout d'abord, l'étude spatiale est réalisée en fonction de la taille caractéristique du maillage dans le domaine géométrique c_{interior} pour les valeurs suivantes $[0.2, 0.5, 1, 2]$ mm. L'analyse qualitative est faite sur les résultats présentés à la **Figure 3.16**. L'augmentation de la résolution spatiale induit une augmentation locale de la vitesse, ce qui est particulièrement visible à l'onde E et la fermeture de VM en fin de systole. Mise à part $c_{\text{interior}} = 2$ mm, les autres cas montrent un écoulement semblable et ce tout au long du cycle cardiaque. La variabilité des structures vorticales est visible à l'onde A mais elles restent majoritairement de faible amplitude de vitesse. La valeur $c_{\text{interior}} = 1$ mm semble être le seuil à partir duquel l'influence de la dynamique mitrale sur l'écoulement est résolue. Dans un soucis de gain de temps de calculs et dans l'objectif d'obtenir un modèle robuste à la comparaison et non le plus réaliste possible, la valeur $c_{\text{interior}} = 1$ mm est utilisée pour le reste du projet.

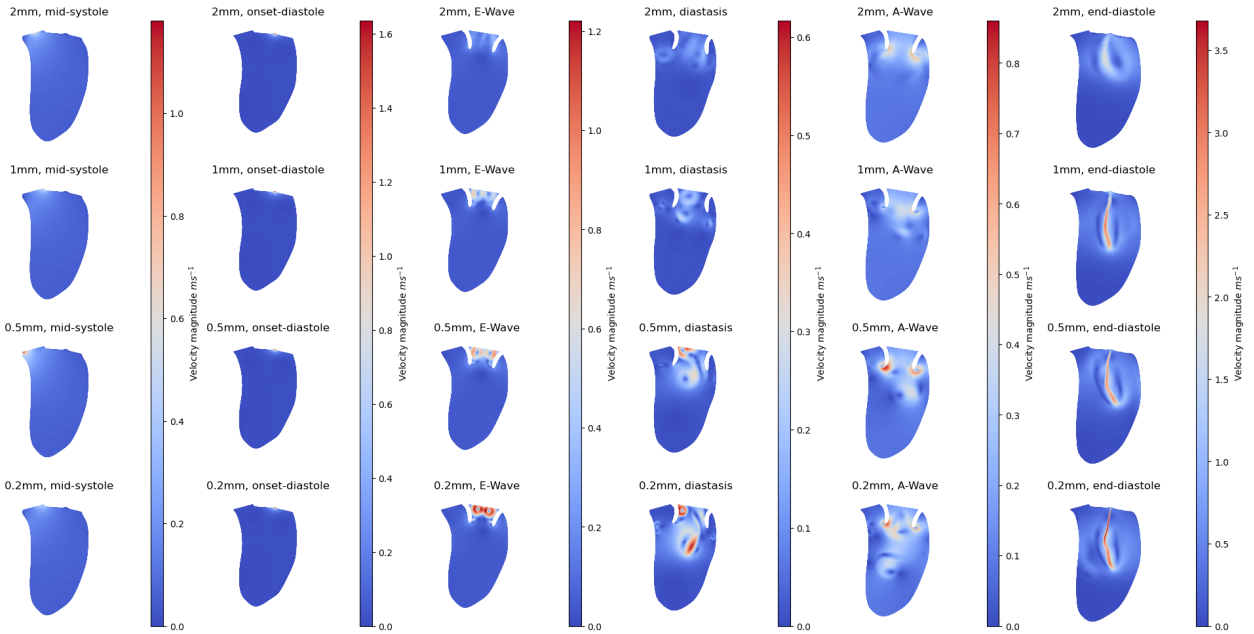


FIGURE 3.16 Étude de l'influence de la résolution spatiale en fonction de c_{interior} à des moments clés du cycle cardiaque.

Une étude temporelle est réalisée en fonction du pas de temps Δt. Un pas de temps initial de $\Delta t = 943$ μs issu du modèle précédent [1] est utilisé comme référence. Les pas de temps simulés sont donc $[\frac{\Delta t}{2}, \Delta t, 2\Delta t, 5\Delta t]$. Les résultats de cette étude sont présentés à la **Figure 3.17**. Une observation rapide est que le cas $2\Delta t$ montre d'ores et déjà des altérations de l'écoulement et $5\Delta t$ présente un flux totalement différent. La formation des vortex au moment de l'onde E est totalement modifiée par comparaison au cas de référence Δt. Contrairement à ce

qu'on pourrait s'attendre, le cas $\frac{\Delta t}{2}$ montre un flux moins bien résolu que le cas de référence Δt, avec des résidus plus élevés et des difficultés à converger. Ce phénomène doit être lié à une accumulation d'erreurs lors de l'interpolation temporelle de la segmentation donnant lieu à un déplacement irrégulier et donc un écoulement perturbé. La conclusion est qu'il n'y a pas d'intérêts à diminuer la résolution temporelle au risque d'altérer complètement l'écoulement. Un calcul rapide basé sur cette étude montre que $\Delta t \sim \frac{T}{1000}$ est une règle empirique cohérente pour estimer la résolution temporelle nécessaire pour notre simulation.

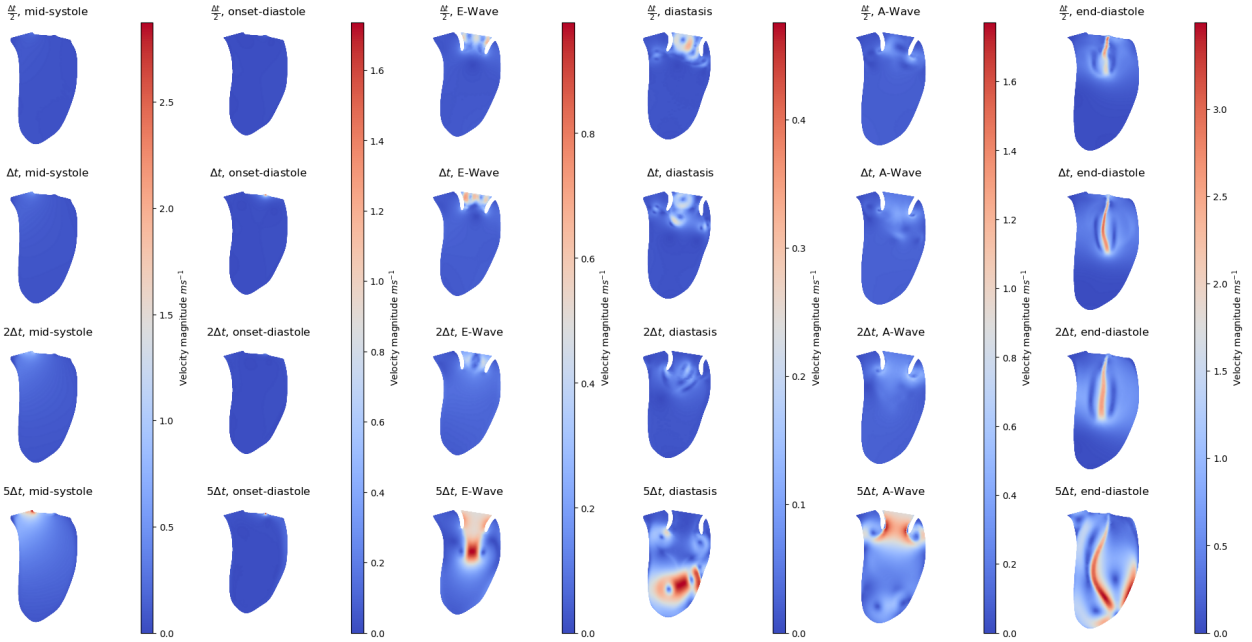


FIGURE 3.17 Étude de l'influence de la résolution temporelle en fonction de Δt à des moments clés du cycle cardiaque.

3.3.1 Ouverture Mitrale Non Contrainte (*OMNC*)

L'étude de ce modèle débute par une étude d'hystérésis pour estimer le nombre de cycles cardiaques nécessaires pour atteindre un régime permanent et périodique. Cette étude est illustrée à la **Figure 3.18**. Comme attendu, le premier cycle simulé met en mouvement le fluide qui était initialement au repos. Les second et troisième cycles sont similaires, ce qui indique qu'on se rapproche dès le second cycle du régime permanent. Mais il faut attendre le troisième cycle pour voir s'installer totalement une recirculation à deux vortex dans le VG, un vortex principal au centre du VG et un secondaire au niveau de l'apex. La conclusion est que 3 cycles semblent nécessaires pour atteindre le régime permanent.

Une étude plus approfondie montre que l'approche d'ouverture non contrainte de la VM donne lieu à des flux non physiologiques. Comme illustré à **Figure 3.19**, au premier cycle cardiaque, lorsque la VM s'ouvre, il y a une décompression en amont de la valve et une compression en aval donnant lieu à un flux régurgitant vers l'atrium (sur la gauche de l'image) et, lorsque la VM se referme, une décompression apparaît en aval le long des feuillets mitraux et une compression en amont créant une recirculation importante et un jet exagéré. Physiologiquement, le déplacement de la VM est déterminé par les différences de pression entre l'atrium et le VG. Tout au long de la diastole, la pression au niveau de l'atrium est supérieure à celle du VG qui se remplit, de fait aucun flux régurgitant n'est possible mis à part à la fin de la diastole lorsque le gradient de pression s'inverse et que la VM se referme. Ceci est observé cliniquement [65]. De la même manière, lorsque la VM se referme, cela indique que les pressions entre l'atrium et le VG se rapprochent et donc que le gradient de pression qui maintenait la VM ouverte diminue. De ce fait, la fermeture se fait à pression égale, rendant impossible l'apparition d'un jet de l'ampleur de celui simulé. Le caractère anormal n'est pas la présence des recirculations le long des feuillets mitraux en aval ou le jet mais leurs amplitudes. Ces défauts proviennent du fait que le mouvement de la VM est découplé de la dynamique ventriculaire et donc du champ de pression simulé dans le VG.

De plus, il a été observé que la recirculation créée dans le VG à la fin du troisième cycle

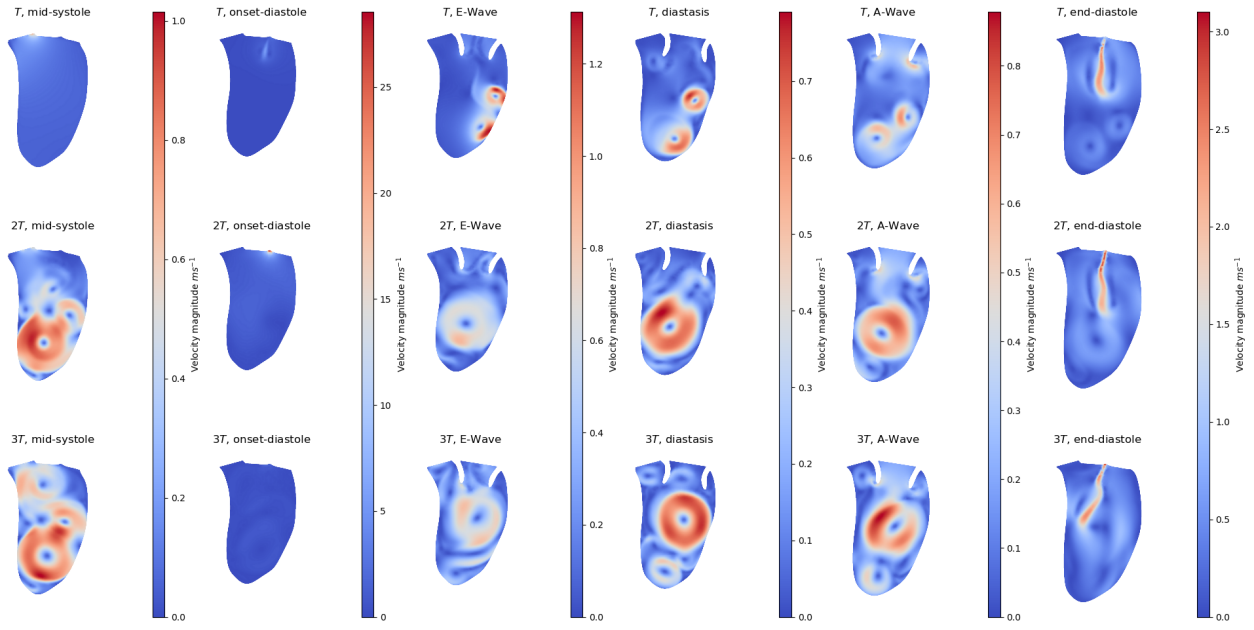


FIGURE 3.18 Étude d'hystérésis du modèle *OMNC* basée sur l'amplitude de la vitesse à des moments clés du cycle cardiaque.

va dans le sens inverse de celle observée cliniquement et discutée dans la littérature, i.e. en rotation de la VM vers la VA au lieu de l'inverse. Une origine possible est l'utilisation dans cette étude préliminaire d'un profil symétrique pour la VM, i.e. $\beta_t = 0.5$. Cette condition géométrique ne pousse pas le flux entrant vers la paroi postérieure mais plutôt vers le centre du VG où il est dévié vers la paroi antérieure.

Les résultats indiquent toutefois que le modèle de valve paramétrique est intégré avec succès.

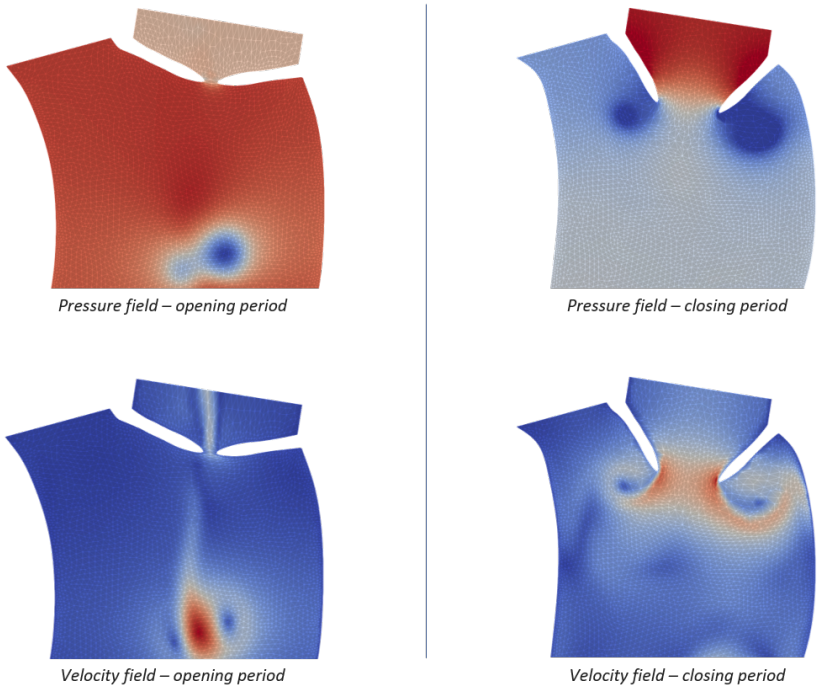


FIGURE 3.19 Illustration de la présence de flux non physiologiques simulés par le modèle *OMNC* basée sur les champs de vitesse et de pression.

3.3.2 Ouverture Mitrale Contrainte (*OMC*)

L'initiative derrière le développement du modèle *OMC* est de fournir une dynamique mitrale physiologique pour éviter les anomalies de l'écoulement observé avec *OMNC*, c'est à dire un flux régurgitant à l'onde E et un jet exagéré à la fermeture de la VM. Une reconstruction géométrique physiologique ($\beta_t = 0.7, \beta_n = 0.25$) est désormais utilisée. Sont également testées les deux stratégies angulaires *Naturelle* et *Forcée* comme présentées à la **Figure 3.20**. L'ensemble des résultats présentés ci-dessous sont basés sur le champ de vitesse simulé au troisième cycle cardiaque en accord avec les résultats présentés sur l'étude d'hystérésis faite

avec le modèle *OMNC* et confirmée par la **Figure 3.21**.

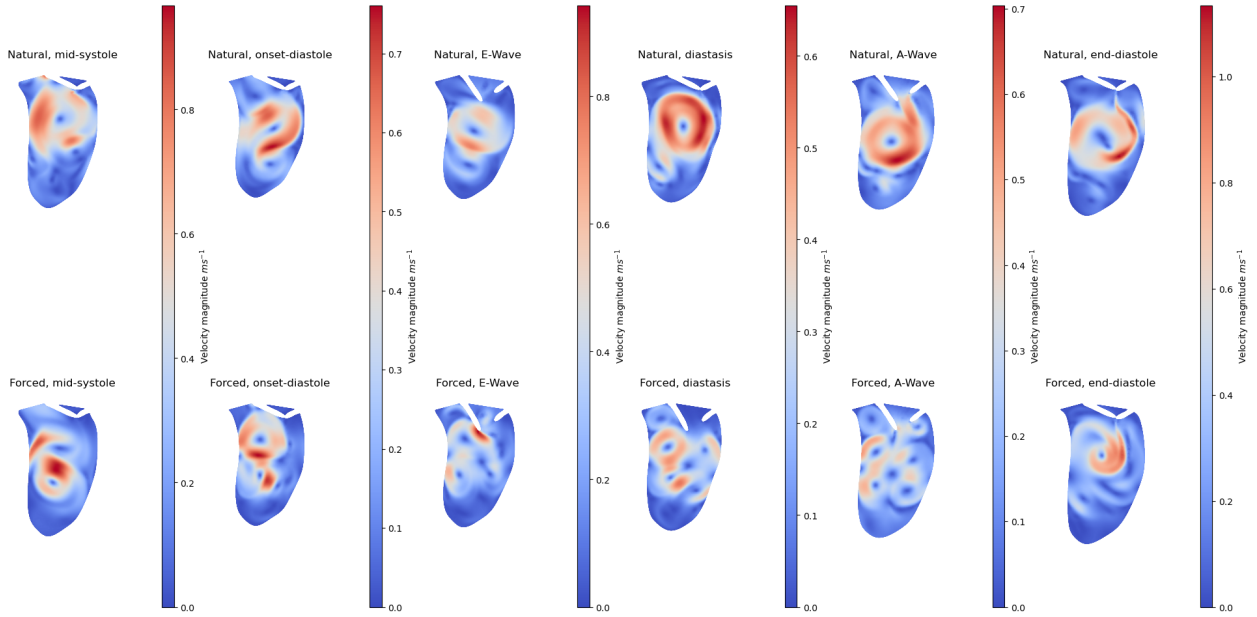


FIGURE 3.20 Comparaison des stratégies *Naturelle* et *Forcée* avec l'utilisation du modèle *OMC*.

Premièrement, les deux stratégies ont permis de totalement éradiquer le flux régurgitant.

La stratégie *Forcée* produit un champ vectoriel désordonné avec la présence d'une multitude de sous-structures vorticales. Contrairement à la stratégie *Naturelle*, aucune recirculation globale ne se met en place. En l'absence d'un modèle de turbulence, ces vortex ne sont pas dissipés et conduisent à un écoulement ventriculaire chaotique et non physiologique. La conclusion est que la fermeture partielle de la VM qui n'est pas prise en compte dans la stratégie *Forcée* est cruciale pour la modélisation de la dynamique ventriculaire.

Dans la cas de la stratégie *Naturelle*, le jet de fermeture de la VM reste la première source de mise en mouvement de la recirculation dans le VG. Contrairement à la littérature [15, 73, 90], ici c'est l'entrée du jet diastolique de vitesse supérieure à celle présente dans le fluide lors de la fermeture de la VM qui induit l'apparition d'un vortex principal et non le cisaillement sur les feuillets mitraux. C'est tout particulièrement visible à la **Figure 3.21** durant le premier cycle cardiaque. Le résultat montre un vortex principal qui est peu propagé vers l'apex.

Contrairement à *OMNC*, celui-ci tourne dans le sens physiologique, comme illustré à la **Figure 3.22**. Cet indice suggère que l'asymétrie de la VM joue un rôle important dans l'écoulement intraventriculaire, comme l'indique certains travaux [71, 90]. Cette influence est étudiée plus en détails au **Chapitre 4**.

Encore une fois, des petites structures vorticales sont disséminées dans le VG et ne sont pas toutes éjectées durant la systole. En temps normal, ces vortex devraient être dissipés au niveau de la paroi du VG, tout spécialement au niveau de l'apex. Cette dissipation turbulente est le fruit du phénomène de cascade tourbillonnante énoncé par Kolmogorov mais aussi de la rugosité apparente des trabécules charnues qui se sont pas modélisées. Cependant, ces résultats illustrent le caractère complexe des structures vorticales observées durant l'écoulement intraventriculaire, allant de petits vortex à des vortex globaux. *In vivo*, cette complexité est

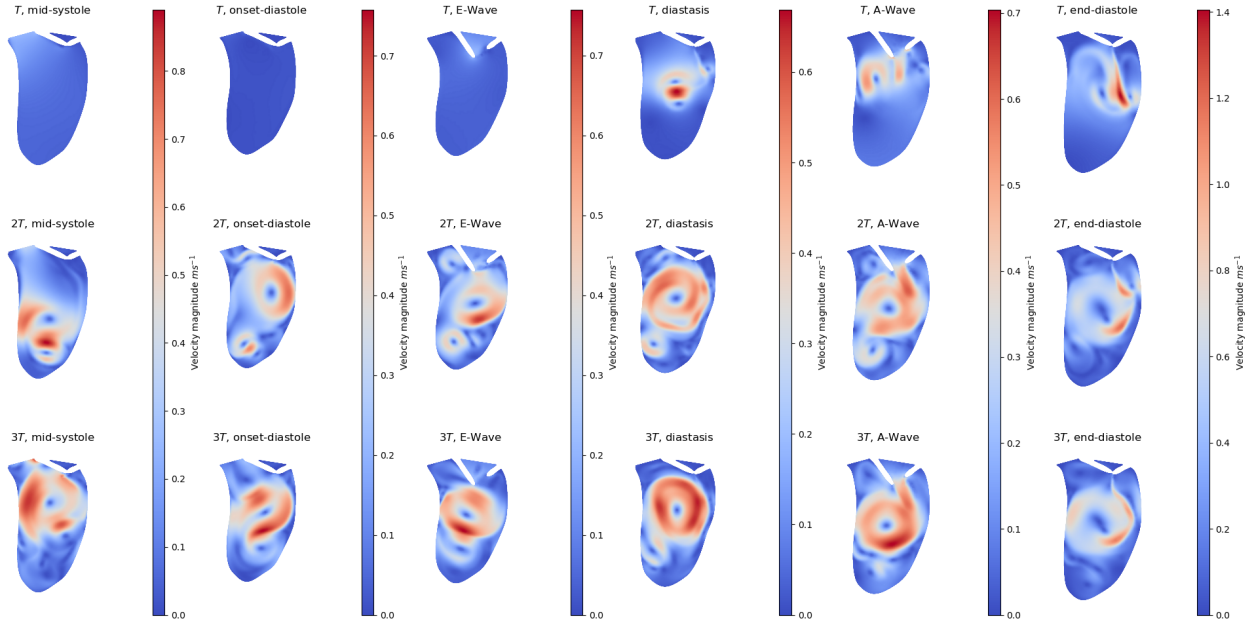


FIGURE 3.21 Étude d'hystérésis du modèle *OMC* avec stratégie *Naturelle* basée sur l'amplitude de la vitesse à des moments clés du cycle cardiaque.

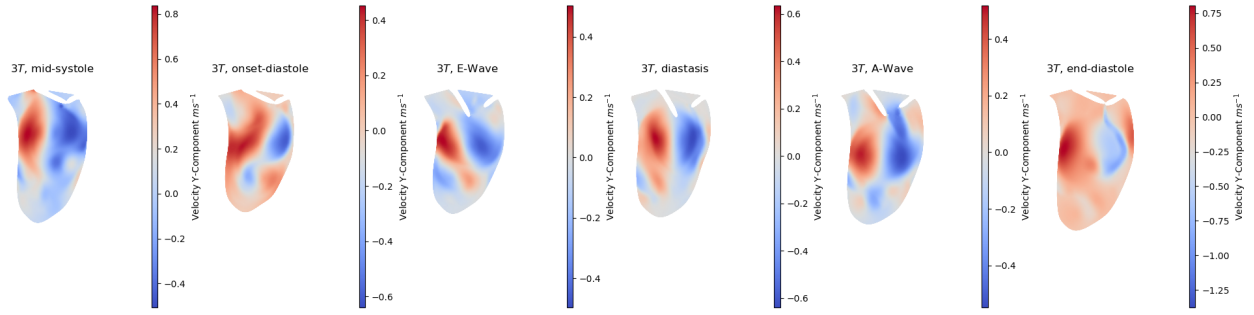


FIGURE 3.22 Étude d'hystérésis du modèle *OMC* avec stratégie *Naturelle* basée sur la vitesse axiale (apex-base) à des moments clés du cycle cardiaque.

d'une part héritée de l'atrium mais aussi induite par la dynamique ventriculaire [73].

En conclusion, l'introduction d'une dynamique mitrale physiologique a permis d'éviter les principales anomalies non physiologiques. En comparaison avec la stratégie *Forcée*, la stratégie *Naturelle* est plus prometteuse en se rapprochant de la description de l'écoulement faite dans la littérature [12,15,47,71,73,90]. On observe d'ores et déjà plusieurs limitations, la première étant le manque de dissipation des vortex du au manque de modélisation de la turbulence et de la rugosité des parois ventriculaires. Cependant, cette structure vorticale non triviale tend à appuyer le fait que la mécanique du VG induit un écoulement complexe, en plus de la complexité *in vivo* héritée de l'atrium, ce qui contredit l'image lissée répandue par l'imagerie médicale.

Finalement, deux indices tendent à corroborer l'hypothèse que le flux entrant est sous-estimé. Le premier indice est que le vortex principal ne se propage pas suffisamment en profondeur par rapport à ce qui est décrit dans d'autres publications [12, 15, 47, 71, 73, 90]. Le second indice est que la formation de ce vortex principal n'est pas induit par le cisaillement sur les valvules mitrales, contrairement aux résultats de modèles *FSI* [15, 90]. Il semblerait que la déformation géométrique sous-estime le flux entrant, ou autrement dit qu'il existe un flux transverse au plan dans le VG.

3.3.3 Flux Diastolique Amplifié (*FDA*)

Le modèle *FDA* a pour objectif d'augmenter artificiellement le flux entrant dans le VG pour voir le potentiel impact sur l'écoulement. Nous reprenons pour ce fait les éléments validés précédemment, c'est-à-dire la reconstruction géométrique ($\beta_t = 0.7, \beta_n = 0.25$) et la stratégie angulaire *Naturelle*. Le facteur d'amplification est $a_U = 5$ qui se rapproche des valeurs physiologiques estimées en comparant l'amplitude du profil de vitesse issue de la déformation géométrique et l'EchoCG Doppler.

Une première étude sur plusieurs cycles révèle que le régime permanent est atteint au troisième cycle comme pour les modèles précédents, voir **Figure 3.23**. L'analyse du premier cycle cardiaque montre que la formation du vortex principal prend place dès l'onde E, puis s'accélère lors de la fermeture partielle durant la diastase. À nouveau, la formation du vortex est du à l'entrée d'un fluide dans un milieu de vitesse inférieure et non du fait du cisaillement le long des faces internes des valvules, ce qui est également observé dans d'autres modèles à *GP* avec modélisation de la VM [45,73]. Pour obtenir la formation de vortex par détachement de la zone de cisaillement, il faudrait une résolution spatiale accrue le long des valvules, ainsi qu'un modèle de turbulence. Par conséquence, contrairement à la description faite dans la majorité des publications, ici, la naissance du vortex annulaire ne se fait pas sur l'extérieur

de la VM mais directement à l'intérieur du VG, ce qui est visible à l'onde E du premier cycle cardiaque. Avec un flux entrant amplifié, nous obtenons également dès la fin du premier cycle cardiaque une structure à deux vortex.

On réalise donc une comparaison au troisième cycle des modèles *FDA* et *OMC*, les deux avec la stratégie *Naturelle* pour estimer l'influence d'augmenter le flux diastolique entrant sur l'écoulement dans le VG. Comme illustré à la **Figure 3.25**, le modèle *FDA* donne un écoulement intraventriculaire beaucoup plus structuré, avec un second vortex de vitesse et de taille plus grande qu'avec *OMC*. Dans les deux cas, le vortex principal semble être maintenu de cycle en cycle. Un second vortex de cet ampleur proche de l'apex n'a pas été reporté dans le cadre de modèles utilisant une formation turbulente [15, 73], mais peu d'entre eux, si ce n'est aucun n'a réalisé d'étude sur plusieurs cycles. Ceci tend à confirmer qu'un modèle permettant la dissipation turbulente est intéressant pour décrire de manière satisfaisante l'atténuation des vortex et par conséquent de l'écoulement dans le VG.

L'étude de la vitesse axiale estimée selon l'axe apex-base positive vers la base, disponible aux **Figures 3.22** et **3.24**, révèle peu de différences entre les modèles *FDA* et *OMC*. Il est possible de remarquer la présence plus marquée d'un vortex secondaire au niveau de l'apex, particulièrement visible en fin de diastole et pendant la systole. Le flux entrant est plus projeté le long de la face postérieure du VG dans le cas du modèle *FDA*. Le vortex principal semble s'y propager légèrement plus en profondeur.

Dans le cas du modèle *FDA*, en fin de diastole nous constatons l'apparition de deux vortex, un principal au milieu du VG et un secondaire au niveau de l'apex. Cette structure est altérée par la systole, le vortex principal se déplace vers l'atrium, connectant les deux vortex de sorte à créer 2 bandes distinctes dans le VG, une bande négative vers l'apex le long de la face antérieure et une bande positive le long de la face postérieure. L'onde E correspond à une phase de transition pour retrouver un système à deux vortex dès la diastase. Le vortex principal est poussé vers l'apex durant l'onde A. Cette formation de la recirculation en deux temps, i.e. formation pendant l'onde E et propagation pendant l'onde A, a déjà été observée via des modèles *FSI*.

En conclusion, le modèle *FDA* met en évidence le fait que le vortex principal n'est pas uniquement du au cisaillement contre la face interne de la VM mais existe aussi grâce à la différence de vitesse entre le flux entrant et le volume résiduel présent dans le VG. La naissance du vortex annulaire contre la VM n'est pas observé ici, il se développe directement dans la cavité ventriculaire. Le modèle *FDA* présente un développement et une propagation amplifiée du vortex diastolique. En essence, la structure de l'écoulement telle que simulée avec *OMC* reste la même, mais est plus marquée avec ce dernier modèle. Un modèle avec

dissipation turbulente semble préférable pour atténuer la propagation de vortex secondaires dans la cavité, et plus particulièrement celui situé au niveau de l'apex.

Quant à la pertinence d'un tel modèle, il faudra voir si le fait de recaler la vitesse du profil de vitesse entrant par la vitesse maximal de l'EchoCG Doppler permet d'obtenir une meilleure adéquation entre notre simulation et la réalité. Cette étude est faite au **Chapitre 5**.

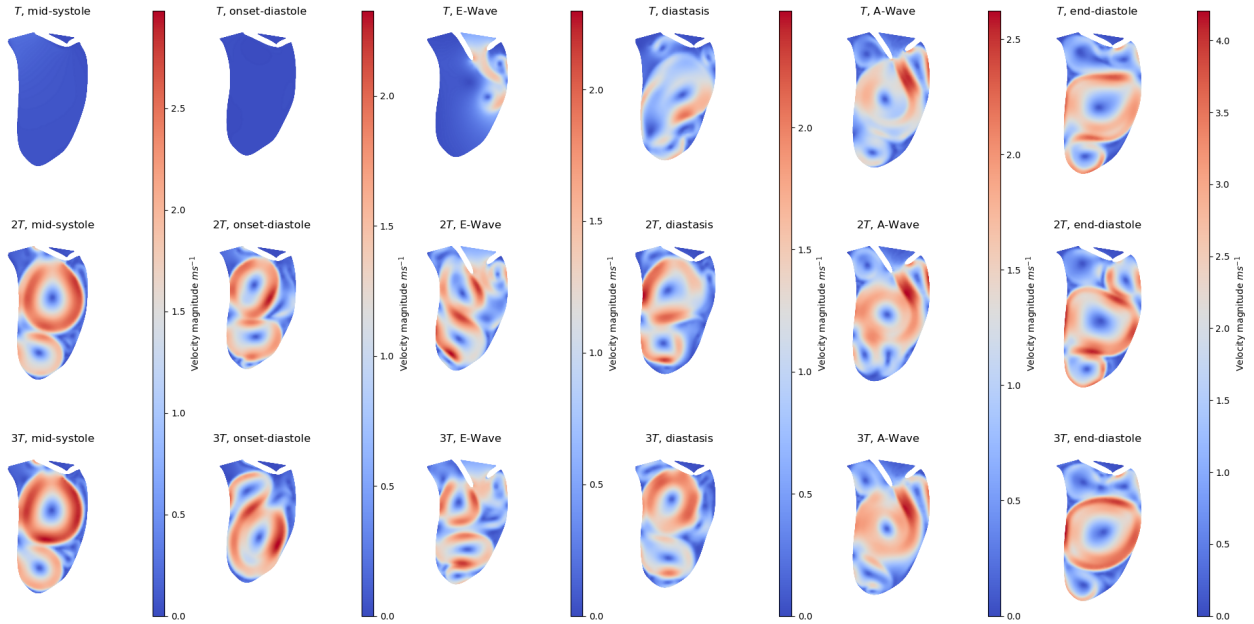


FIGURE 3.23 Étude d'hystérésis du modèle *FDA* avec stratégie *Naturelle* basée sur l'amplitude de la vitesse à des moments clés du cycle cardiaque.

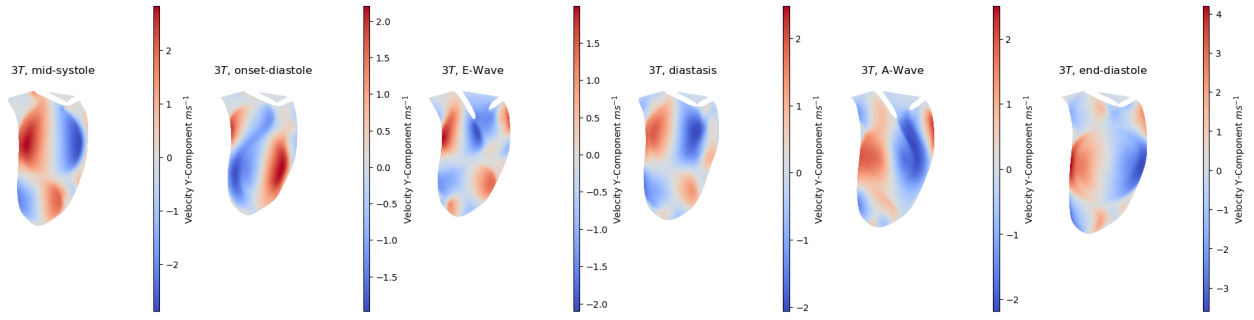


FIGURE 3.24 Étude d'hystérésis du modèle *FDA* avec stratégie *Naturelle* basée sur la vitesse axiale (apex-base) à des moments clés du cycle cardiaque.

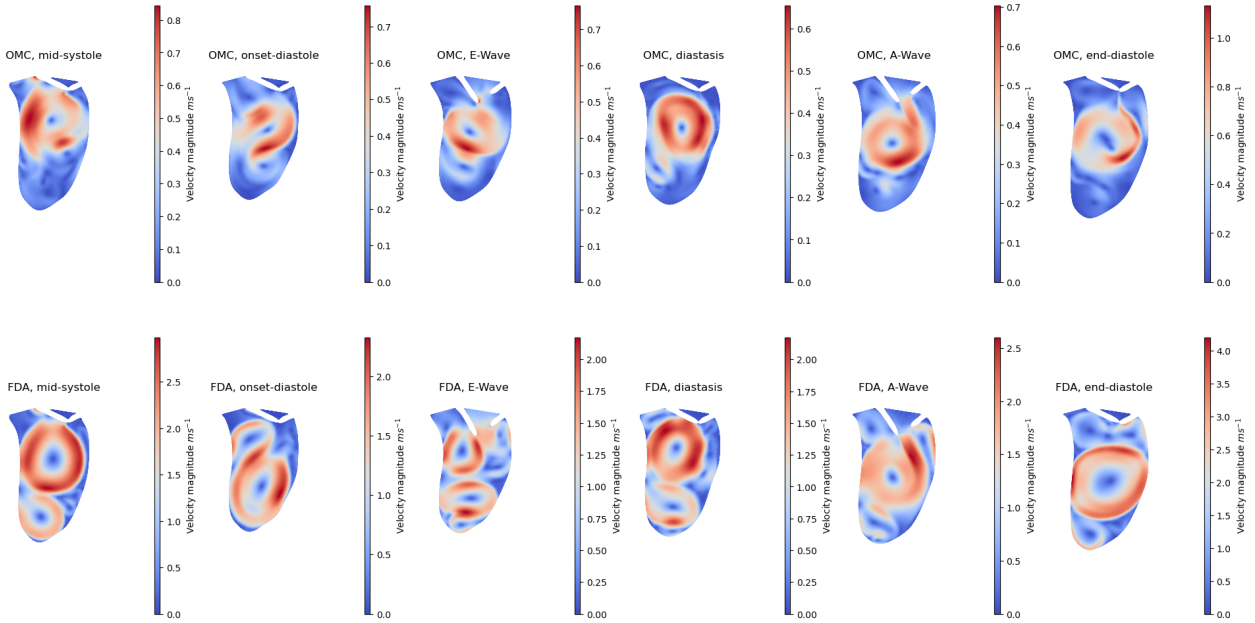


FIGURE 3.25 Comparaison des écoulements simulés par les modèles *OMC* et *FDA* avec la stratégie *Naturelle* basée sur l'amplitude de la vitesse.

CHAPITRE 4 ÉTUDE DE SENSIBILITÉ DES PARAMÈTRES GÉOMÈTRIQUES DU MODÈLE DE VALVE MITRALE

Ce chapitre est dédié à l'étude de l'influence des paramètres géométriques β_t et β_n sur l'écoulement dans le VG. La première **Section 4.1** est dédiée à la présentation de la méthodologie de comparaison. Les résultats de sensibilité sont ensuite présentés et interprétés à la **Section 4.2**. Une conclusion à la **Section 4.3** est finalement apportée pour donner une vue globale sur l'influence de la forme de la VM sur l'écoulement dans le VG. L'objectif est de répondre à l'Objectif n°3 énoncé à la **Section 1.3**.

4.1 Méthodologie de la comparaison

Cette section reprend les points de méthodologie et plus particulièrement les quantités calculées pour comparer qualitativement et quantitativement l'influence des différents paramètres géométriques β_t et β_n sur l'écoulement intraventriculaire.

Étude du champ de vitesse Comme vue précédemment au **Chapitre 3**, l'étude du champ de vitesse au troisième cycle cardiaque donne des informations cruciales sur le mode de formation des vortex et, par extension, la recirculation dans la cavité ventriculaire. Cette méthode est donc reconduite pour l'étude de sensibilité. Les résultats concernent l'amplitude de la vitesse, mais aussi la vitesse axiale selon l'axe apex-base. La vitesse axiale offre une description plus fine concernant le sens de rotation des structures vorticales. Enfin, la représentation axiale se rapproche de l'image EchoCG Doppler acquise en clinique dans le sens où la principale composante participant à la vitesse radiale est cette vitesse axiale. La norme est faite de sorte qu'une vitesse axiale positive se dirige vers la base, tandis qu'une vitesse négative pointe vers l'apex du VG. Des exemples de tels résultats sont disponibles au **Chapitre 3** aux **Figures 3.23** et **3.24**. Des variables comme la vorticité ou le *Q-criterion* ne sont pas utilisés, pour ne pas surcharger la lecture du manuscrit dans la mesure où le comportement de l'écoulement peut être décrit en utilisant les moyens décrits ci-dessus.

Reconstruction des trajectoires - Cartographie et classification du VTD Basé sur les travaux de Carlhäll et Bolger (2010) [91], Bolger et al. (2007) [92] et Eriksson et al. (2010, 2011 et 2013) [16,93,94], nous introduisons une nouvelle méthode pour caractériser l'écoulement diastolique. Le principe repose sur la catégorisation de trajectoires de particules imaginaires calculées à partir du champ de vitesse pour obtenir une représentation statistique

et visuelle de l'écoulement. Les trajectoires sont divisées en 4 catégories [16, 91, 92] :

- *Direct Flow* : Les trajectoires qui entrent par la VM et sortent par la VA durant le cycle cardiaque.
- *Retained Inflow* : Les trajectoires qui entrent par la VM et ne sortent pas durant le cycle cardiaque.
- *Delayed Ejection Flow* : Les trajectoires qui débutent dans le VG et sortent par la VA durant le cycle cardiaque.
- *Residual Volume* : Les trajectoires qui débutent dans le VG et ne sortent pas durant le cycle cardiaque.

Deux méthodes pour calculer ses trajectoires semblent disponibles, la première méthode [91, 92] est de définir un plan émetteur dont sont émises en continue des particules durant la diastole, la seconde méthode [16] consiste à émettre une unique salve de trajectoires sur une grille uniforme basée sur le VTD. La particularité de la seconde méthode est qu'il faut procéder à une intégration temporelle des trajectoires vers l'avant (*forward integration*) de la systole et une intégration vers l'arrière (*backward integration*) de la diastole, voir illustration à la **Figure 4.1**.

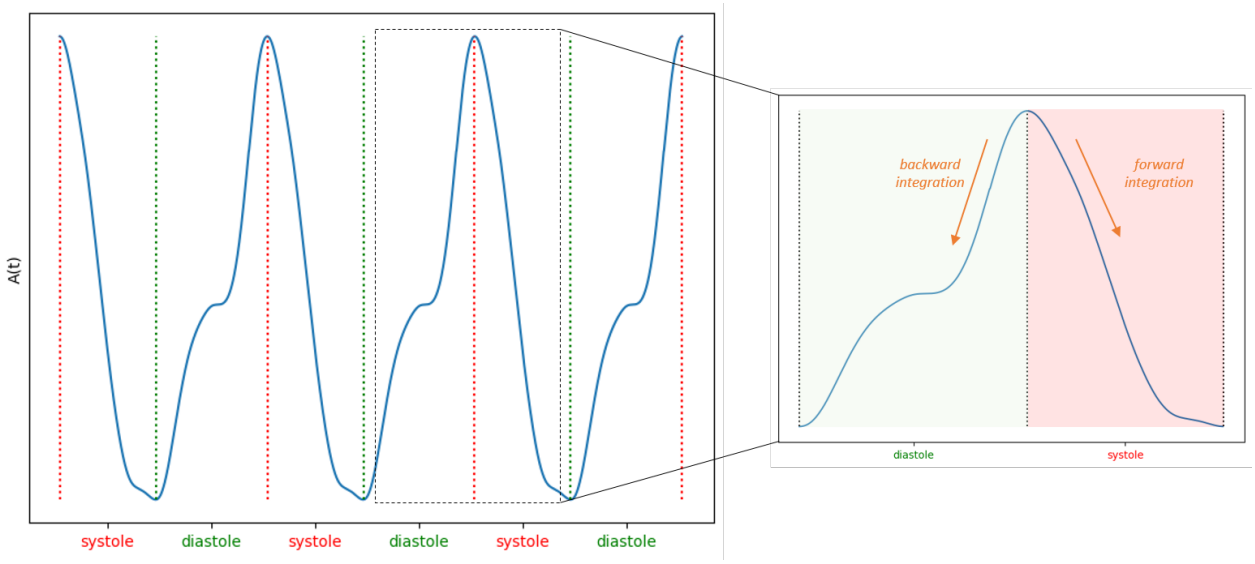


FIGURE 4.1 Schéma explicatif de l'intégration spatio-temporelle des trajectoires particulaires.

Une fois les trajectoires calculées, il est possible d'obtenir une représentation statistique et visuelle de l'écoulement, voir **Figures 4.2** et **4.3**.

La seconde méthode est appliquée pour sa facilité d'implémentation. Le modèle numérique

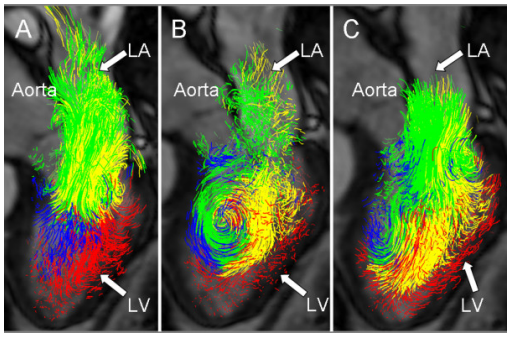


FIGURE 4.2 Représentation des trajectoires calculées de l'écoulement du VG (A) durant l'onde E, (B) durant la diastase, (c) durant l'onde A ; Code Couleur *Direct Flow* : vert, *Retained Inflow* : jaune, *Delayed Ejection Flow* : bleu, *Residual Volume* : rouge. (Eriksson et al. (2010) [16] *CC 4.0*)

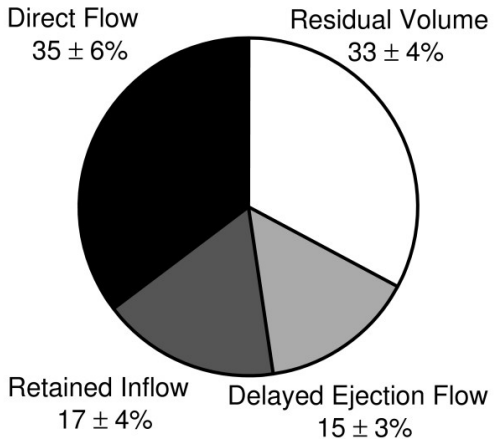


FIGURE 4.3 Représentation des composantes de l'écoulement en pourcentage du VTD. (Eriksson et al. (2010) [16] *CC 4.0*)

OMC Naturelle est utilisé pour assurer un écoulement physiologique qui respecte la conservation de la masse. Le troisième cycle cardiaque est utilisé comme champ de vitesse de référence pour l'intégration des trajectoires. Le champ de vitesse simulé est interpolé sur une grille structurée de résolution $d = 0.1\ mm$. La grille d'initialisation est l'intersection d'une grille de résolution $d_{init} = \alpha^2 d$ avec $\alpha = 0.1$ et le VTD. La valeur de α a été sélectionnée empiriquement pour limiter le temps de calcul tout en préservant la véracité des résultats. Un exemple de grille d'initialisation est visible à la **Figure 4.4**. L'intégration fait intervenir un schéma numérique Runge-Kutta d'ordre 4. La stratégie des articles cités précédemment est de retirer les trajectoires sortant de la segmentation du VG. Ici, dans le cas où une particule sortirait du VG, au lieu de la retirer, elle est déplacée au point de la frontière le plus proche au début du pas de temps suivant. Cette modification a pour but d'améliorer la conservation de la masse. Des vidéos couvrant l'ensemble du cycle cardiaque ont été produites mais, par soucis d'efficacité, seule une représentation visuelle du VTD avec le code couleur correspondant aux catégories énoncées précédemment est présentée par la suite. Ce visuel est accompagné du résultat statistique qui lui est associé.

L'ensemble de l'implémentation (**Code 11**) est disponible à l'**Annexe D**.

4.2 Résultats de l'étude de sensibilité

Cette section a pour objectif de présenter les résultats et de les interpréter. Les valeurs de paramètres géométriques sont $\beta_t \in [0.3, 0.5, 0.7]$ et $\beta_n \in [0.15, 0.25]$. Le paramètre β_t mesure

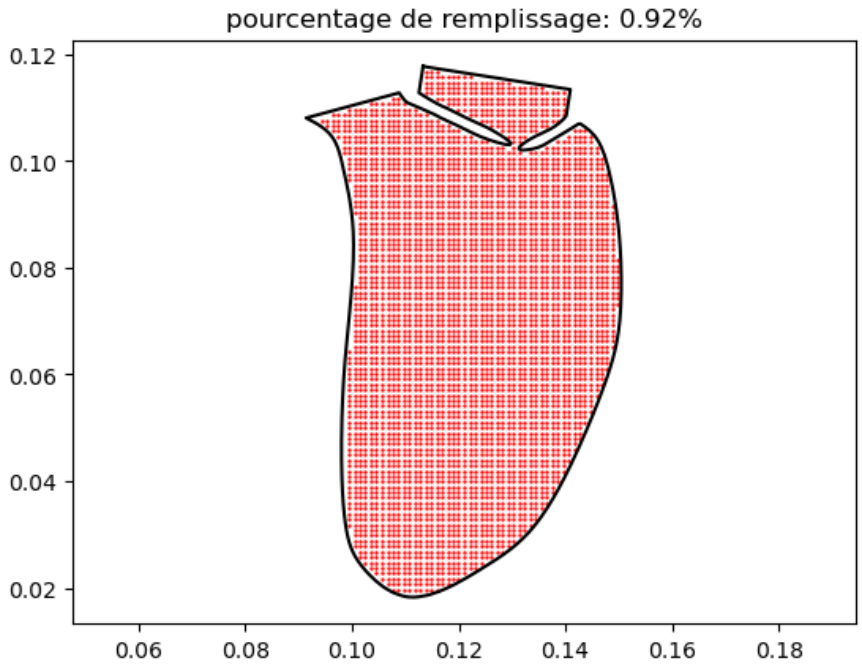


FIGURE 4.4 Exemple de grille d'initialisation des trajectoires avec le taux de remplissage par rapport à la grille d'interpolation du champ de vitesse.

la direction d'asymétrie de la VM et le paramètre β_n le phénomène de prolapsus mitral. Grâce aux données EchoCG, il est estimé que d'un point de vue physiologique une personne en bonne santé possède un couple de paramètre se rapprochant de $\beta_t \sim 0.7$ et $\beta_n \sim 0.25$. La valeur de $\beta_t = 0.3$ représente un cas pathologique. Le cas $\beta_t = 0.5$ représente l'hypothèse d'une valve symétrique. Le cas $\beta_n = 0.15$ simule un léger prolapsus mitral. Il est attendu que le paramètre β_t ait une influence sur l'écoulement et l'efficacité de la fonction cardiaque. Aucune hypothèse ne peut être faite sur l'influence de β_n.

4.2.1 Résultat de l'analyse du champ de vitesse

Cette section s'intéresse aux résultats et interprétations issues de l'analyse du champ de vitesse de l'écoulement, présentés aux **Figures 4.5** et **4.6**.

L'influence de β_t est examinée en premier.
À la fin de la diastole, lors de la fermeture finale de VM, le jet change d'orientation en fonction de β_t avec un jet vers la face antérieure pour $\beta_t = 0.3$, un jet plutôt central pour $\beta_t = 0.5$ et un jet vers la face postérieure pour $\beta_t = 0.7$. Cela induit pour les cas $\beta_t = 0.3$ et $\beta_t = 0.5$ une inversion du sens de rotation du vortex principal. À $\beta_t = 0.5$, avec une valve

symétrique, l'écoulement est en phase de transition avec $\beta_n = 0.25$, mais en cas de prolapsus avec $\beta_n = 0.15$, la rotation est d'ores et déjà inversée. C'est tout particulièrement visible en étudiant la vitesse axiale à la diastase.

De manière générale, $\beta_t = 0.5$ donne lieu à un écoulement plus chaotique avec de multiples structures vorticales qui n'arrivent pas à s'organiser. Ce phénomène est plus marqué pour $\beta_n = 0.25$ que $\beta_n = 0.15$. Dans le cas $\beta_n = 0.25$, le champ de vitesse à l'onde E, la diastase et l'onde A sont non structurés. À l'inverse de $\beta_t = 0.3$ ou $\beta_t = 0.7$, ici le flux diastolique entrant n'est pas bloqué le long d'une paroi et donc les deux vortex induits peuvent se propager librement dans le VG. Il est attendu qu'avec un modèle de dissipation turbulente, ces sous-structures vorticales se dissipent au cours du temps, mais ici elles survivent de cycle en cycle. Physiologiquement, un écoulement chaotique de ce genre donne lieu à des contraintes de cisaillement élevées dans la cavité pouvant provoquer des phénomènes de thromboses, mais aussi engendrer une perte énergétique importante demandant alors un effort supplémentaire au cœur pour assurer la fonction cardiaque.

Dans le cas $\beta_t = 0.3$, peu importe β_n, la recirculation en sens inverse bloque une partie de l'écoulement dans la partie inférieure de la cavité lors de la systole. Une discontinuité de la vitesse axiale au niveau de la face antérieure du VG à la mi-systole se forme avec une valeur positive proche de la VA, une valeur négative au milieu de la cavité et une valeur positive au niveau de l'apex. Cette complexité des lignes de courant indique que le fluide remonte le long de la face antérieure au niveau de l'apex, passe du côté postérieur au milieu de la cavité et rejoint ensuite la VA. Ce motif entraîne d'une part une résistance à l'éjection mais aussi une perte de charge par frottement visqueux.

Finalement, le cas $\beta_t = 0.7$ propose des résultats en accord avec la littérature, avec un écoulement organisé qui favorise l'éjection systolique. Cela se traduit par un vortex principal qui tourne de la VA vers la VM, facilitant à la fois le remplissage diastolique mais aussi à l'expulsion du fluide situé au niveau de l'apex.

Désormais, nous étudions l'influence de β_n sur l'écoulement.
De manière générale, peu importe la valeur de β_t, l'écoulement semble plus marqué et organisé avec $\beta_n = 0.15$ que $\beta_n = 0.25$. C'est particulièrement visible sur les images d'amplitude de la vitesse au moment de l'onde A. Le prolapsus semble limiter la distance à laquelle le jet de fin de diastole se propage dans la cavité, donnant lieu à une simplification de l'écoulement dans la partie basse de la cavité (i.e. au niveau de l'apex) et ce peu importe la valeur de β_t. Cet effet est visible d'une part à la fin de la diastole mais aussi au moment de l'onde E.

Concernant la performance à l'éjection, le paramètre β_n ne semble pas avoir d'influence directe significative.

En conclusion, l'étude de l'amplitude de vitesse et la vitesse axiale a révélé le caractère critique du paramètre β_t sur l'orientation du vortex diastolique, mais aussi sur la capacité de l'écoulement à s'organiser au cours des cycles. D'un point de vue physiologique, tout semble indiquer qu'une asymétrie de la VM vers la face antérieure du VG est défavorable pour la fonction cardiaque, demandant un effort supérieur au muscle cardiaque pour assurer l'éjection systolique.

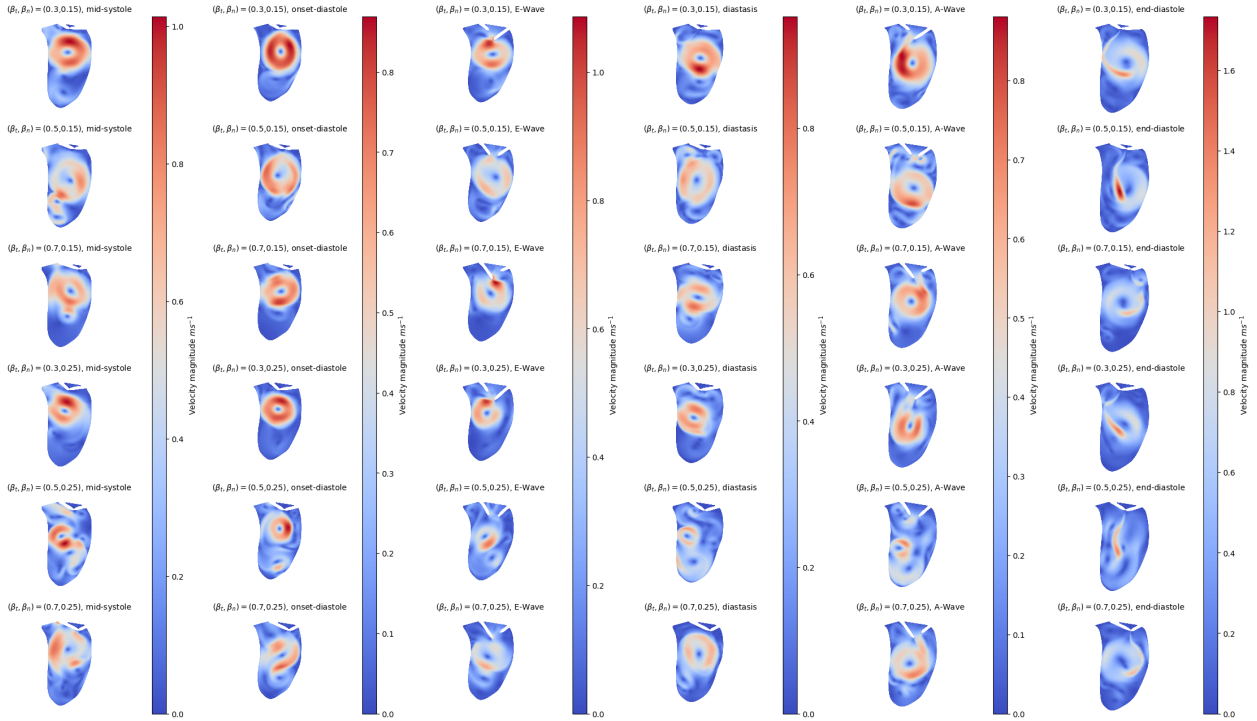


FIGURE 4.5 Étude de l'influence des paramètres $\beta_t \in [0.7, 0.5, 0.3]$ et $\beta_n \in [0.15, 0.25]$ sur l'écoulement dans le VG basée sur l'amplitude de la vitesse à divers moments du cycle cardiaque.

4.2.2 Résultat de la cartographie et catégorisation de l'écoulement

Cette section s'intéresse aux résultats et interprétations issues de la technique de cartographie et catégorisation de l'écoulement, présentés aux **Figures 4.7** et **4.8**.

Les valeurs issues de Eriksson et al. (2010) [16] sont données comme référence :

- *Direct Flow* : $30 \pm 10\%$
- *Retained Inflow* : $18 \pm 4\%$
- *Delayed Ejection Flow* : $17 \pm 4\%$
- *Residual Volume* : $36 \pm 6\%$

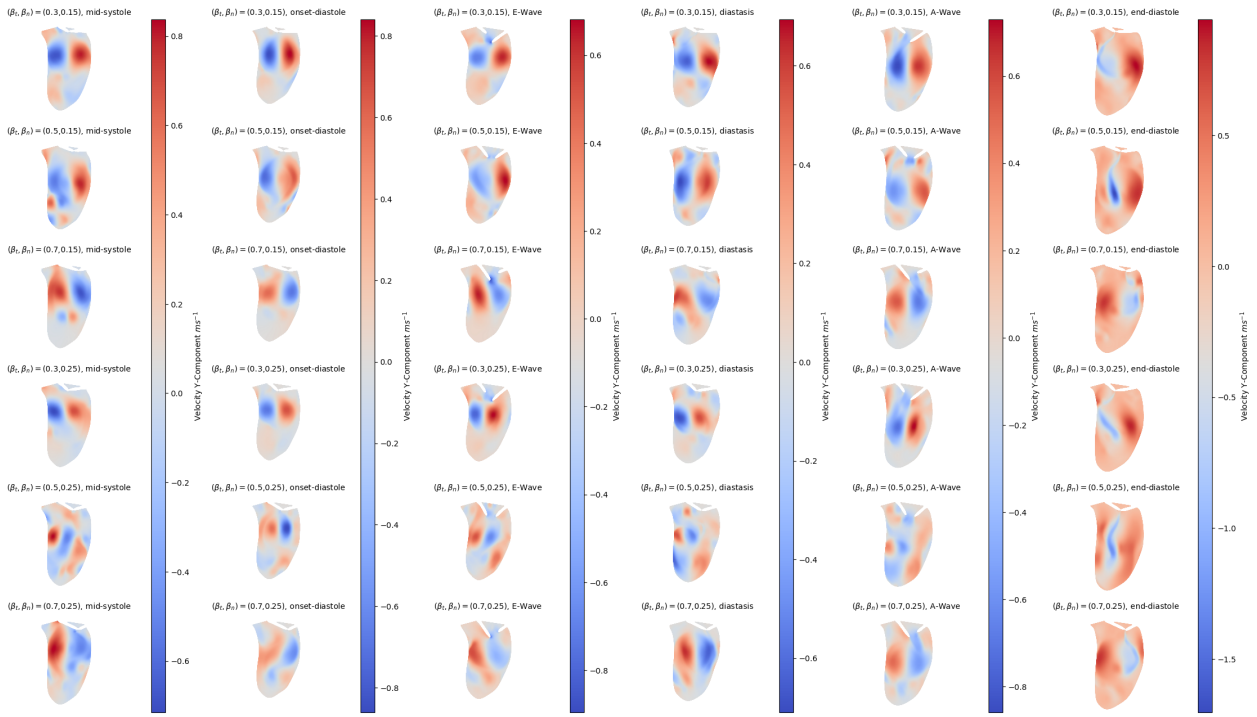


FIGURE 4.6 Étude de l'influence des paramètres $\beta_t \in [0.7, 0.5, 0.3]$ et $\beta_n \in [0.15, 0.25]$ sur l'écoulement dans le VG basée sur la vitesse axiale (apex-base) à divers moments du cycle cardiaque.

Une première comparaison indique que *Retained Inflow* et *Delayed Ejection Flow* sont proches de la valeur de référence et sont quasi-identiques, ce qui indique que la condition de conservation de la masse au cours d'un cycle cardiaque est respectée. On observe que *Direct Flow* de $\sim 2 - 6.5$ % et *Residual Volume* de $\sim 59 - 63.5$ % sont respectivement plus petit et plus grand que les valeurs de référence. Dans la mesure où ces résultats de simulation sont réalisés avec le modèle *OMC*, i.e. sans amplification du flux entrant, cela constitue une preuve supplémentaire que le flux entrant est sous-estimé par rapport à la réalité physiologique. En effet, les valeurs de référence sont issues de champs de vitesses mesurées par EchoCG et reflètent donc la projection 2D d'un flux 3D *in vivo*. Avec un flux entrant minimisé, la recirculation dans le VG est moins efficace, donnant lieu à une baisse du *Direct Flow* et une augmentation du *Residual Volume*.

La cartographie sur VTD montre, dans tous les cas simulés, des accumulations de particules de la catégorie *Retained Inflow* en amont de la VM et de la catégorie *Delayed Ejection Flow* au niveau de la VA, ce qui est attendu. Les particules de la catégorie *Direct Flow* sont disséminées sauf pour $\beta_t = 0.3$, où un amas est observé le long de la face antérieure du VG. La catégorie *Residual Volume* occupe la majorité du volume.

Il est à noter que chaque cas possède une cartographie qui lui est spécifique. Les particules de la catégorie *Retained Inflow* permettent de distinguer également la forme du flux entrant dans le VG. En effet, la direction du jet diastolique est à nouveau visible avec une direction vers la face postérieure du VG pour le cas $\beta_t = 0.7$, une direction centrale pour le cas $\beta_t = 0.5$ et une direction vers la face antérieure pour le cas $\beta_t = 0.3$. Ces caractéristiques semblent plus marquées pour une valeur de $\beta_t = 0.25$ que pour $\beta_t = 0.15$.

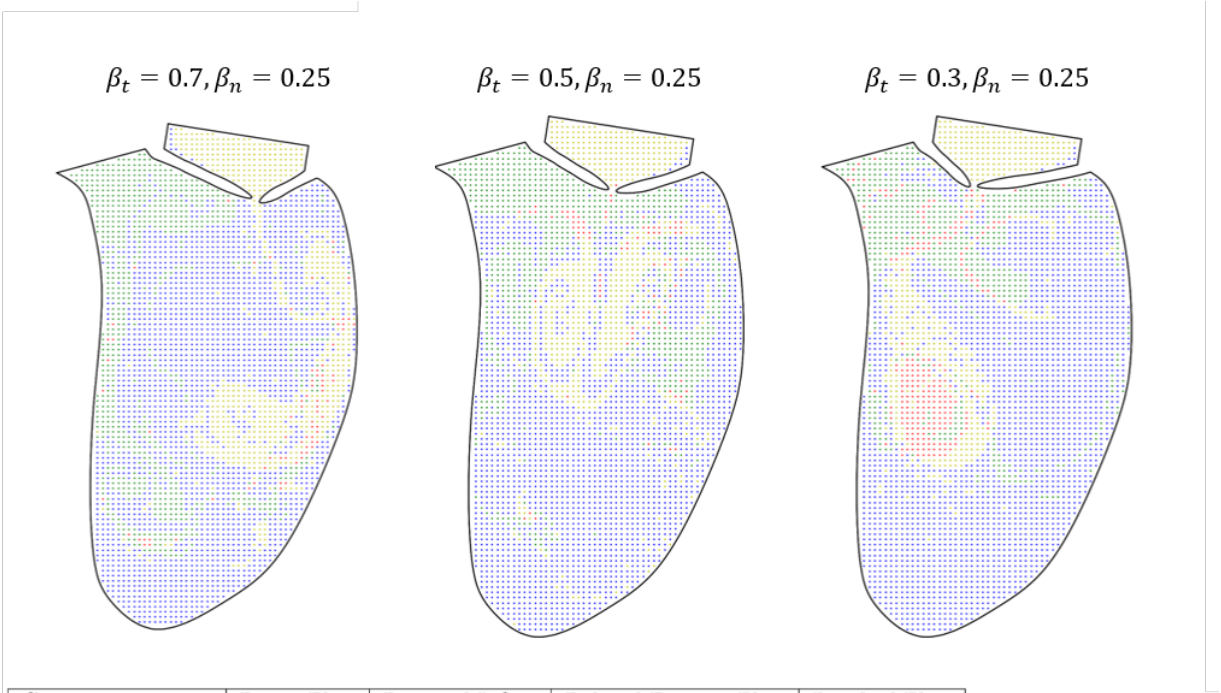
En terme de statistiques, peu importe β_n, peu de différences existent entre les cas $\beta_t = 0.5$ et $\beta_t = 0.7$. Quant à $\beta_t = 0.3$, il se distingue par une augmentation significative du *Residual Volume* mais aussi du *Direct Flow*.

En conclusion, ces résultats montrent que la cartographie de l'écoulement est sensible aux paramètres géométriques de la VM, chaque cas possède son unique cartographie. Mais, même si des changements dans l'écoulement sont discernables, ces résultats ne permettent pas de montrer une différence significative dans la performance cardiaque. La sous-estimation du *Direct Flow* et la surestimation du *Residual Volume* par rapport à des valeurs de référence semblent indiquer à nouveau une sous-estimation du flux entrant dans le VG. De ce fait, la méthode de cartographie et catégorisation de l'écoulement n'est pas la plus adaptée dans le cas 2D.

4.3 Conclusion

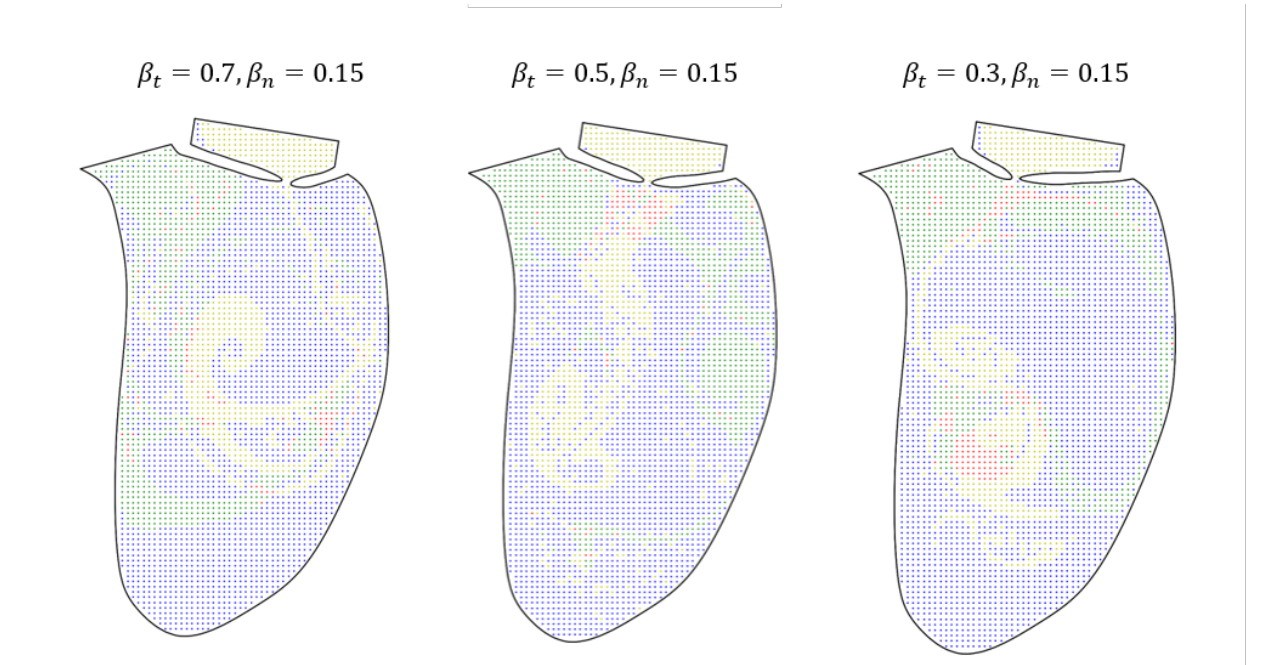
Ce chapitre donne lieu à l'introduction d'un nouvel outil d'observation de l'écoulement dans le VG en cartographiant et catégorisant les trajectoires de particules fictives au cours d'un cycle. Cette méthode, en parallèle des moyens conventionnels d'étude de l'écoulement, est utilisée pour étudier l'influence des paramètres β_t et β_n du modèle paramétrique sur l'écoulement dans le VG et ses conséquences en terme de performance cardiaque. Ces paramètres géométriques peuvent être interprétés physiquement comme une mesure directionnelle de l'asymétrie et du phénomène de prolapsus de la VM. L'étude révèle l'effet critique de l'asymétrie de la VM sur l'écoulement et tout particulièrement sur le développement de recirculations dans la cavité ventriculaire. Une valve symétrique (i.e. $\beta_t = 0.5$) se traduit par un écoulement désorganisé avec de multiples recirculations locales. Une asymétrie inversée (i.e. $\beta_t = 0.3$) induit une inversion du vortex diastolique limitant fortement le déplacement du fluide de l'apex vers la base, entravant l'éjection systolique. Le cas physiologique (i.e. $\beta_t = 0.7$) se rapproche de la description faite dans la littérature, avec la présence d'un écoulement organisé autour d'un vortex principal favorisant le remplissage diastolique et l'éjection systolique. L'influence du phénomène de prolapsus semble toutefois limitée et demanderait une étude complémentaire. Il est notable toutefois qu'un prolapsus léger (i.e. $\beta_n = 0.15$) semble limiter le jet diastolique

dans la partie supérieure de la cavité ventriculaire, limitant la formation de structures annexes au niveau de l'apex. Cela se traduit par une tendance à créer un écoulement plus organisé. Cette influence reste beaucoup moins marquée que pour β_n. La cartographie et catégorisation de l'écoulement ont révélé peu de différences en terme d'efficacité cardiaque mais confirment les modifications faites à l'écoulement. Cette méthode est capable de distinguer les différents cas étudiés en produisant une cartographie qui leur sont propres. Elle permet de souligner la direction du jet diastolique et l'organisation du fluide dans la cavité. La comparaison avec des valeurs de référence a permis de mettre en avant une sous-estimation du flux diastolique donnant lieu à une augmentation du volume résiduel et une diminution de l'éjection directe au cours d'un cycle. Il s'avère donc que l'utilisation de cette méthode n'est pas adaptée pour l'étude 2D, mais est prometteuse pour la comparaison inter-individuelle dans le cas 3D.



Cas	Direct Flow	Retained Inflow	Delayed Ejection Flow	Residual Flow
$\beta_t = 0.7,\ \beta_n = 0.25$	2.03	17.69	20.54	59.76
$\beta_t = 0.5,\ \beta_n = 0.25$	2.27	17.47	20.18	60.09
$\beta_t = 0.3,\ \beta_n = 0.25$	6.51	13.68	16.34	63.47

FIGURE 4.7 Résultats de la cartographie et catégorisation de l'écoulement dans le VG pour des valeurs de $\beta_t \in [0.7, 0.5, 0.3]$ et $\beta_n = 0.25$.



Cas	Direct Flow	Retained Inflow	Delayed Ejection Flow	Residual Flow
$\beta_t = 0.7,\ \beta_n = 0.15$	2.73	17.25	20.16	59.86
$\beta_t = 0.5,\ \beta_n = 0.15$	2.26	17.94	20.71	59.09
$\beta_t = 0.3,\ \beta_n = 0.15$	4.24	14.36	19.32	62.08

FIGURE 4.8 Résultats de la cartographie et catégorisation de l'écoulement dans le VG pour des valeurs de $\beta_t \in [0.7, 0.5, 0.3]$ et $\beta_n = 0.15$.

CHAPITRE 5 VALIDATION DU MODÈLE NUMÉRIQUE

Ce chapitre est dédié à la validation de la méthode numérique en comparant l'écoulement simulé à la reconstruction issue de la méthode iVFM [17, 33, 35] à la **Section 5.1** et à l'EchoCG Doppler à la **Section 5.2**. Deux modèles numériques seront étudiés : le modèle *OMC* et le modèle *FDA* recalé sur l'EchoCG Doppler, avec une stratégie angulaire *Naturelle*. Une conclusion est proposée à la **Section 5.3** afin de répondre à l'objectif n°4 énoncé à la **Section 1.3**.

La méthode iVFM, autrement appelée *Intraventricular Vector Flow Mapping* sert à reconstruire un champ vectoriel complet à partir de la vitesse radiale issue de l'EchoCG Doppler. Son application d'abord 2D [17] a été étendue au 3D [35]. Le calcul du champ de vitesse se fait par la résolution d'un problème d'optimisation contraint par des lois physiques de la mécanique des fluides. Autrement dit, l'objectif est de produire un champ de vitesse dont la vitesse radiale est égale à celle mesurée à l'EchoCG Doppler tout en vérifiant une série de contraintes physiques telles que des conditions limites, la condition de divergence nulle et partiellement l'équation de Navier-Stokes. L'attrait principal de cette méthode est de pouvoir fournir une reconstruction du champ de vitesse en un temps inférieur à la minute. Son utilisation en recherche a fait l'objet de nombreuses publications et pourrait s'étendre à la clinique. Un schéma explicatif du principe de fonctionnement de la méthode iVFM est disponible à la **Figure 5.1**.

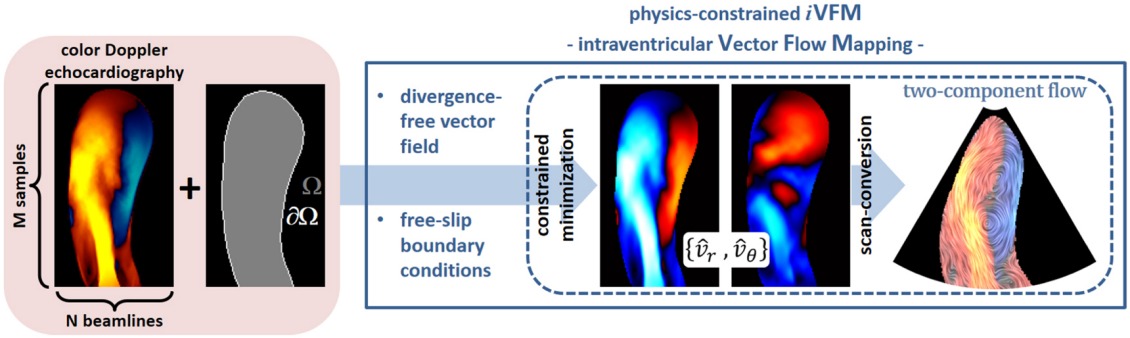


FIGURE 5.1 Schéma explicatif de la méthode iVFM. (Vixège et al. (2021) [17] *CCC License Agreement*)

Deux principales différences existent entre cette méthode et notre modèle numérique. Premièrement, la méthode iVFM demande l'acquisition d'un EchoCG Doppler de résolutions temporelle et spatiale élevées peu pratiquée en clinique pour pouvoir fonctionner correcte-

ment, là où notre modèle se base uniquement sur l'EchoCG mode B largement disponible. Deuxièmement, la méthode iVFM permet une reconstruction à un moment figé dans le temps, là où notre modèle numérique reconstruit l'écoulement dans sa globalité.

En se comparant à cette méthode, cela évalue la capacité de notre modèle à simuler un écoulement physiologique dans le VG. L'intérêt est aussi de situer la performance de notre modèle en tant qu'outil de reconstruction de l'écoulement à partir de données disponibles en clinique, par comparaison à d'autres moyens nécessitant des données plus spécifiques.

Le cas simulé jusqu'à maintenant était issu de la cohorte Pétale et la reconstruction de sa géométrie avait été réalisée lors d'une étude précédente [1]. Ce chapitre requiert de simuler un nouveau cas pour lequel les résultats de la méthode iVFM sont disponibles. La partie inférieure de la cavité ventriculaire est extraite par le logiciel EchoPAC, mais la partie supérieure (i.e. la base) avec la délimitation de la VA et de la VM fait intervenir une routine créée pour l'occasion. Ce travail étant annexe à l'objectif principal de ce chapitre, il est détaillé à l'**Annexe C** et son implémentation numérique (**Code 12**) à l'**Annexe D**. Cette routine fournit également les paramètres géométriques $\beta_t = 0.736$ et $\beta_n = 0.249$ et les paramètres dynamiques $\theta^l_{max} = 80.6°$ et $\theta^r_{max} = 83.3°$.

Le paramètre d'amplification a_U du modèle *FDA* permettant de recaler l'amplitude du flux entrant durant la diastole est calculé à partir de la vitesse radiale maximale enregistrée avec la mesure EchoCG Doppler *unaliased*. Il est estimé à $a_U = 4.69$.

La stratégie angulaire *Naturelle* donne lieu à la dynamique mitrale illustrée à la **Figure 5.2**. Le ratio $E : A$ y est relativement faible, ce qui conduit à une fermeture progressive de la VM au cours de la diastole avec une légère réouverture au moment de l'onde A.

Les **Figures 5.3** et **5.4** présentent l'écoulement ventriculaire à travers l'amplitude de la vitesse et la vitesse axiale à des moments clés du cycle cardiaque sur les trois premiers cycles. Une mise en mouvement plus rapide du fluide est constaté avec le modèle *FDA* que *OMC*, ce qui est attendu. Les deux modèles semblent avoir atteint une certaine périodicité à partir du troisième cycle. Les résultats présentés dans les sections suivantes sont donc réalisés au troisième cycle cardiaque pour limiter l'effet de l'initialisation du fluide au repos.

5.1 Comparaison avec la méthode iVFM

Cette section est dédiée à la comparaison entre les deux modèles numériques et la méthode iVFM. Les principaux résultats pour le modèle *OMC* sont présentés aux **Figures 5.5** et **5.6**, et pour le modèle *FDA* aux **Figures 5.7** et **5.8**.

OMC Tout d'abord, nous nous apercevons d'une différence significative entre l'écoulement simulé et les résultats de l'iVFM. La principale différence est la présence moins marquée d'un flux diastolique au moment de l'onde E. Il semble que le modèle *OMC* sous-estime le flux entrant dans le VG. Tout deux montrent un fluide au quasi-repos à la fermeture de la VA en fin de systole et un écoulement dirigé vers la VA durant la systole. En regardant la vitesse axiale de l'écoulement simulé au moment de l'onde A, en accord avec l'iVFM, le ventricule se divise en trois bandes de vitesse alternativement positive et négative. Finalement, le vortex dans la cavité ventriculaire est plus marqué et tourne dans le sens inverse de celui observé avec la référence iVFM. En conclusion, contrairement à ce que présageaient les résultats vus à la **Section 3.2**, le modèle OMC échoue ici à simuler de manière satisfaisante un écoulement physiologique du VG.

FDA De manière générale, en regardant l'amplitude de vitesse, l'écoulement simulé est globalement plus rapide que la référence iVFM, avec une recirculation principale plus marquée. Cette différence de vitesse rend difficile l'appréciation des différences et ressemblances entre les écoulements en se basant sur l'amplitude de la vitesse.

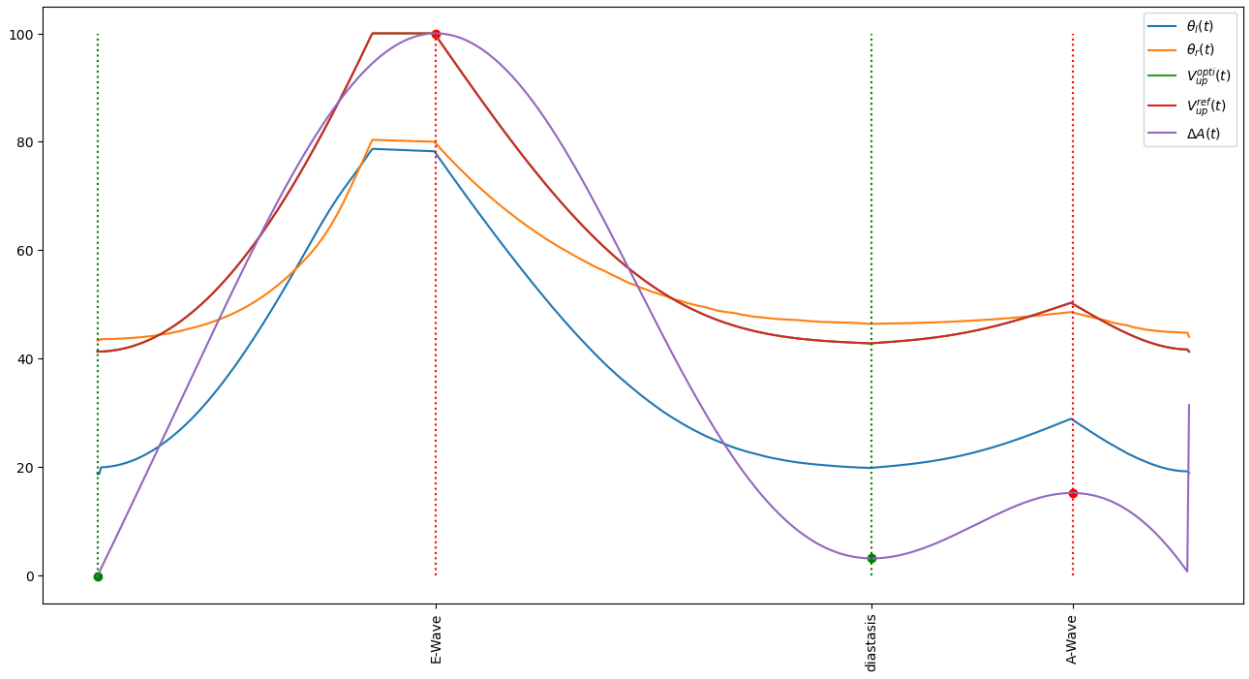


FIGURE 5.2 Présentation de la dynamique mitrale obtenue avec la stratégie *Naturelle* dans le cas iVFM. $\Delta A; V_{up}^{ref}; V_{up}^{opti}$ sont rendues adimensionnelles.

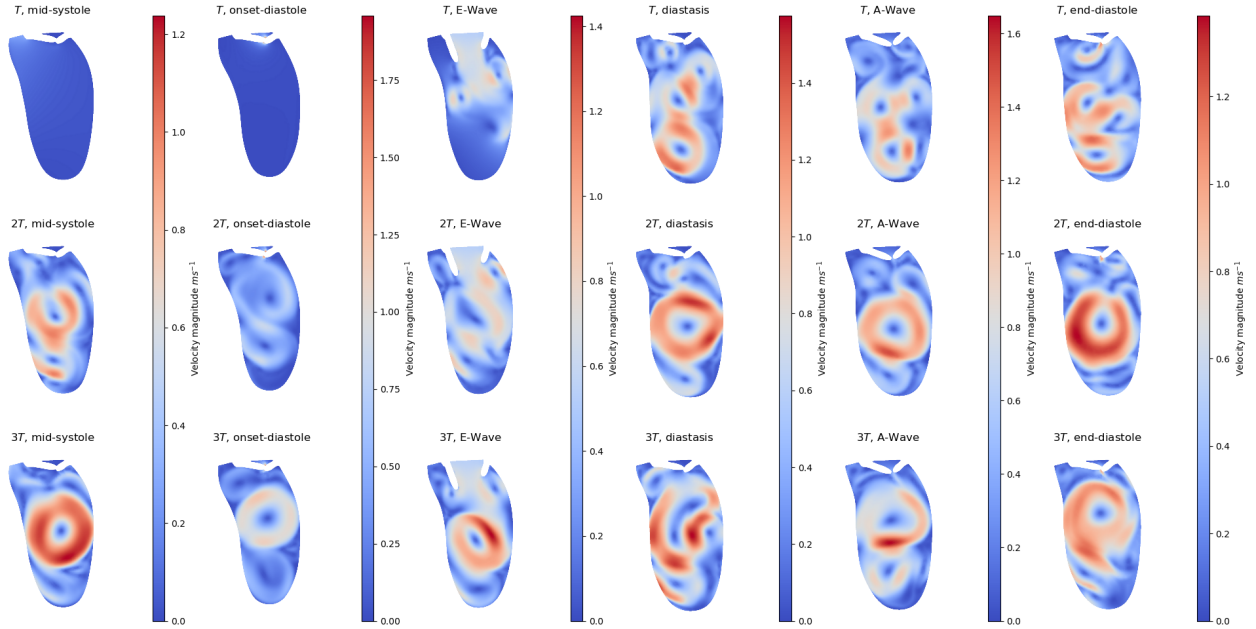


FIGURE 5.3 *Validation* - Étude d'hystérésis du modèle *FDA* avec stratégie *Naturelle* dans le cas iVFM basée sur l'amplitude de la vitesse à des moments clés du cycle cardiaque.

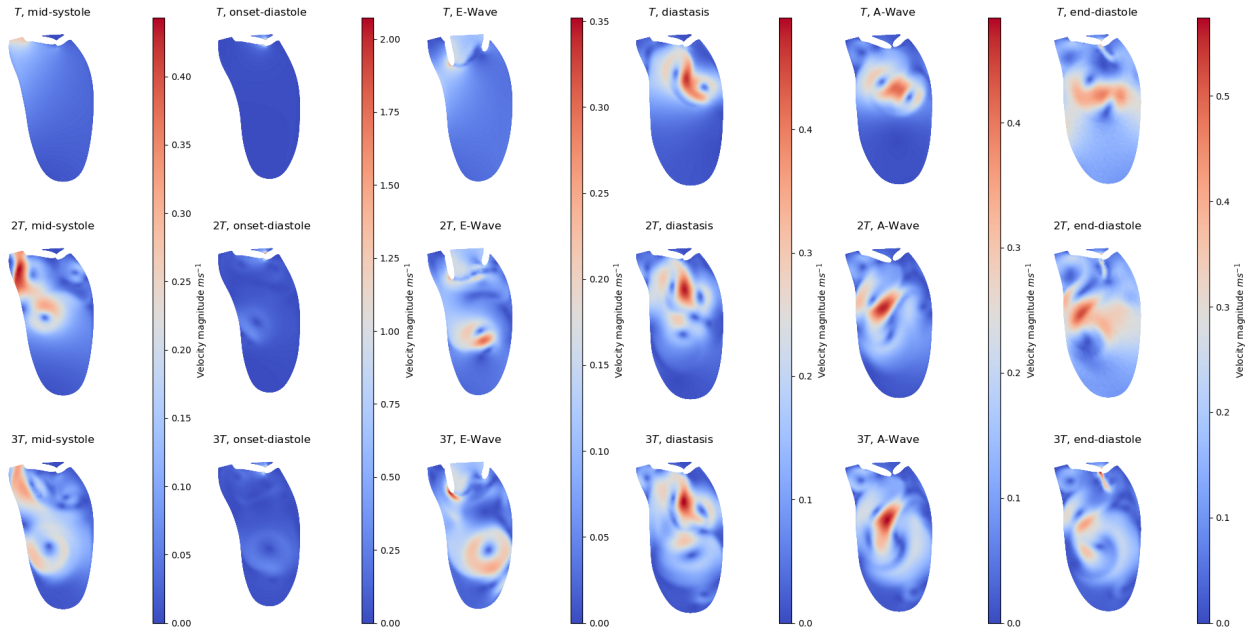


FIGURE 5.4 *Validation* - Étude d'hystérésis du modèle *OMC* avec stratégie *Naturelle* dans le cas iVFM basée sur l'amplitude de la vitesse à des moments clés du cycle cardiaque.

La vitesse axiale révèle quant à elle des similitudes telles qu'un flux diastolique marqué à l'onde E, une recirculation couvrant l'ensemble de la cavité en fin de diastole, la présence d'une vitesse négative le long de la face antérieure et positive le long de la face postérieure au moment de la diastole, et la présence d'une couche limite de vitesse positive à l'onde A et à l'onde E au niveau de la VA le long de la face antérieure du fait de la direction du flux entrant. En accord avec iVFM, l'écoulement à la systole montre une continuité de la vitesse sur la trajectoire de sortie entre le flux sortant et la recirculation le long de la face postérieure. Cependant, la structure des vortex présents dans le VG est beaucoup plus complexe dans le cas du modèle numérique. Plus spécifiquement, la simulation donne lieu à des recirculations secondaires locales au niveau de l'apex et de la VA, invisibles avec iVFM.

Ces indices démontrent une performance accrue du modèle à flux entrant amplifié. Il paraît donc critique de correctement évaluer ce flux entrant si l'on souhaite définir une condition limite en entrée permettant la simulation d'un écoulement physiologique.

En conclusion, concernant les deux modèles, il existe une différence notable entre le résultat de la méthode iVFM et l'écoulement mesuré. Cependant, les résultats issus du modèle *FDA* semblent prometteurs, tout spécialement lorsque l'on compare les vitesses radiales. Il ressort que la méthode iVFM produit un écoulement plus organisé et lisse que celui simulé via les modèles numériques.

5.2 Comparaison avec le Couleur Doppler

Cette section est dédiée à la comparaison entre l'écoulement simulé et la vitesse radiale issue de l'EchoCG Doppler. L'EchoCG Doppler constitue notre référence pour évaluer dans quelle mesure l'écoulement simulé se rapproche de l'écoulement *in vivo*. Deux mesures de la vitesse radiale sont illustrées aux **Figures 5.9** et **5.10**, une mesure issue de l'EchoCG Doppler brute présentant de l'*aliasing* et une issue de l'EchoCG Doppler *unaliased* extraite de l'iVFM. La méthode iVFM requiert une étape de *dealiasing* de la mesure EchoCG Doppler avant de calculer le champ de vitesse puisque la vitesse radiale doit concorder avec celle mesurée. De ce fait, il suffit donc de calculer la vitesse radiale du résultat de la méthode iVFM pour avoir la vitesse radiale extraite de l'EchoCG Doppler unaliased.

OMC En accord avec les résultats présentés à la **Section 5.1**, *OMC* présente une différence marquée avec la vitesse radiale *in vivo*. Cela est particulièrement visible au moment de l'onde E avec un flux entrant minimisé et à la diastole avec une recirculation inversée.

Cependant, si l'on compare la vitesse radiale brute et la vitesse radiale simulée, au moment

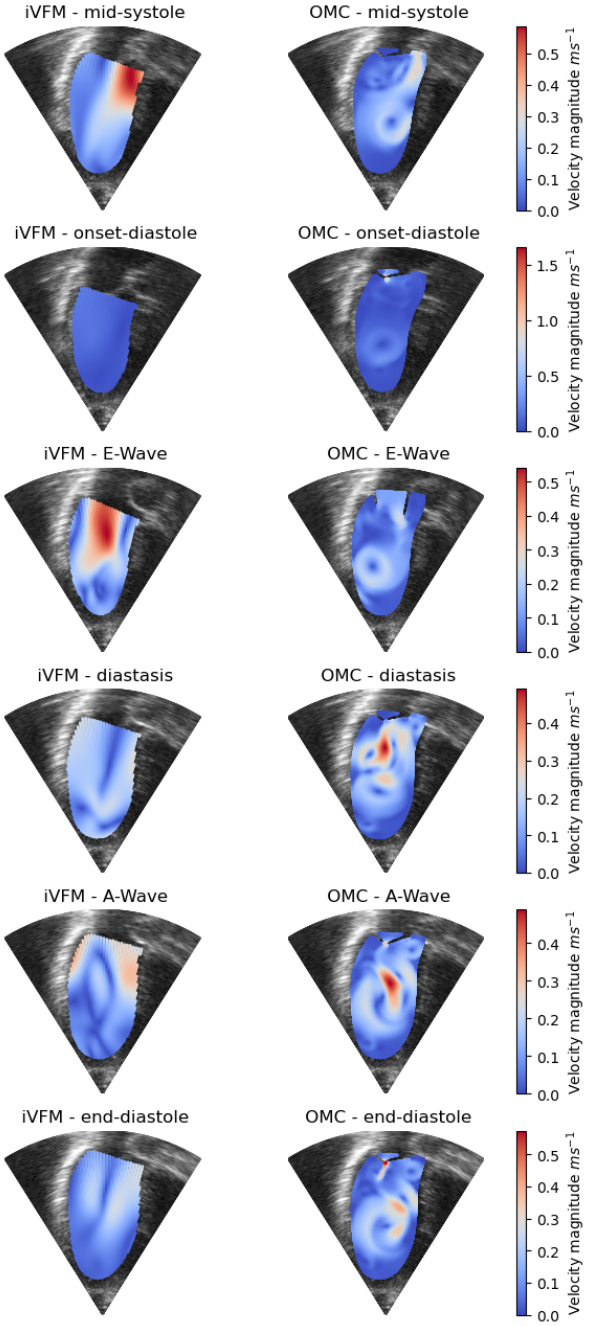


FIGURE 5.5 Comparaison entre les écoulements obtenus avec la méthode iVFM et avec le modèle *OMC* stratégie *Naturelle* basée sur l'amplitude de la vitesse à des moments clés du cycle cardiaque.

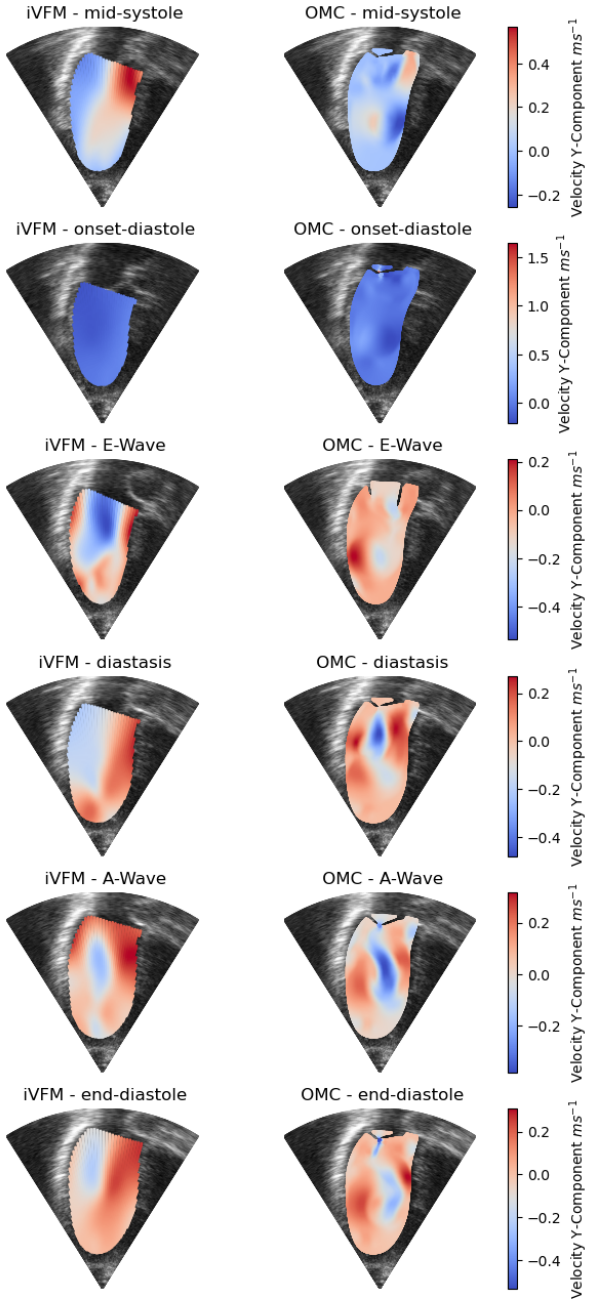


FIGURE 5.6 Comparaison entre les écoulements obtenus avec la méthode iVFM et avec le modèle *OMC* stratégie *Naturelle* basée sur la vitesse axiale (apex-base) à des moments clés du cycle cardiaque.

de la diastase et l'onde A, il est possible de remarquer la présence commune d'un jet central. La vitesse radiale unaliased, lisse la zone le long de la face antérieure et ne permet pas de

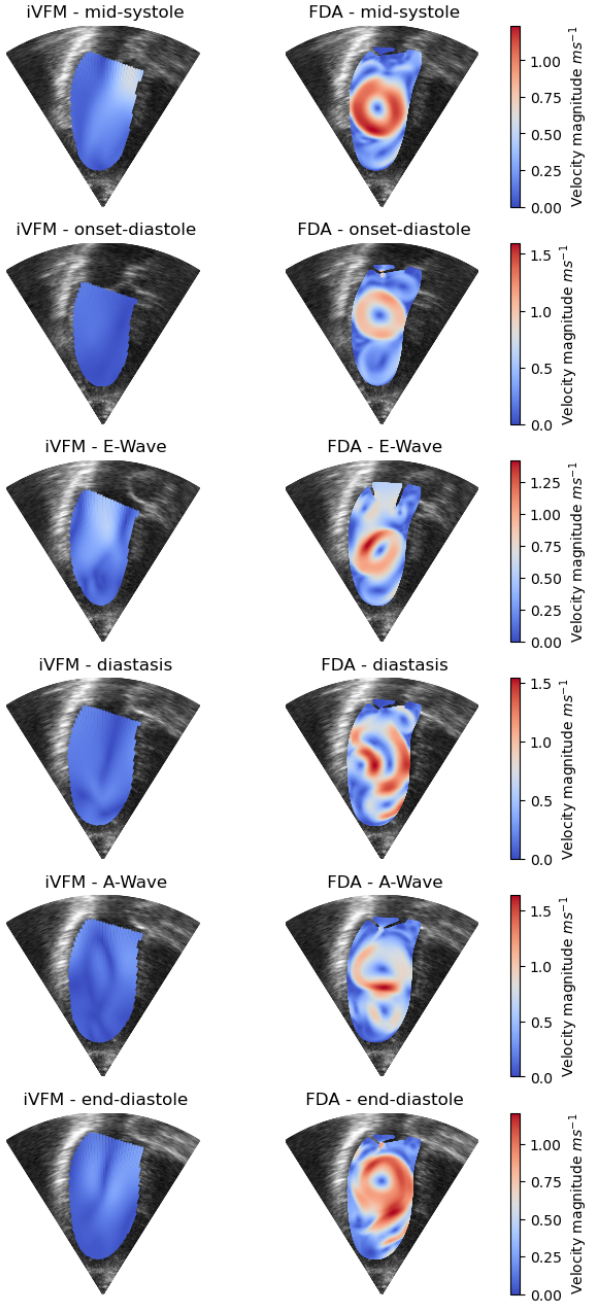


FIGURE 5.7 Comparaison entre les écoulements obtenus avec la méthode iVFM et avec le modèle *FDA* stratégie *Naturelle* basée sur l'amplitude de la vitesse à des moments clés du cycle cardiaque.

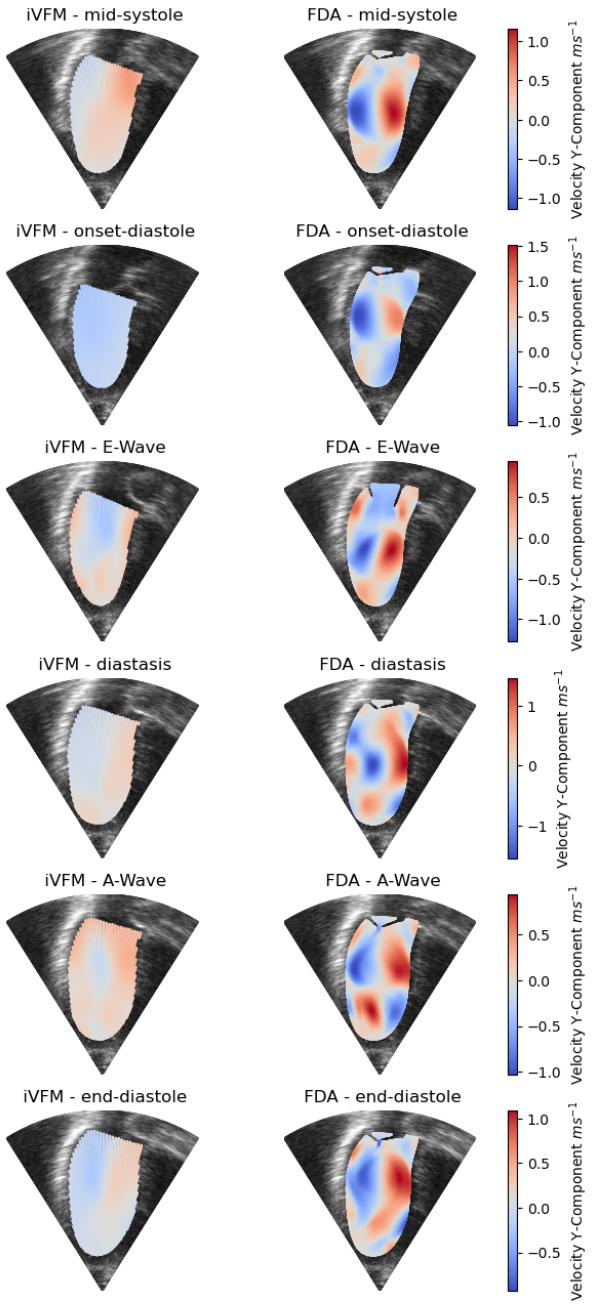


FIGURE 5.8 Comparaison entre les écoulements obtenus avec la méthode iVFM et avec le modèle *FDA* stratégie *Naturelle* basée sur la vitesse axiale (apex-base) à des moments clés du cycle cardiaque.

visualiser les 3 bandes de vitesses. Le modèle *OMC* sous-estime le remplissage à l'onde E mais se comporte mieux au moment de la diastase et l'onde A. Cette indice semble indiquer

que la dynamique mitrale se rapproche de la réalité *in vivo*. La comparaison durant la systole et juste avant la diastole montre un accord entre l'écoulement simulé et la référence *in vivo*. Finalement, la recirculation inversée en fin de diastole va à l'encontre de la mesure EchoCG, ce qui renforce la conclusion que ce modèle échoue globalement à simuler un écoulement physiologique.

FDA Concernant le modèle *FDA*, la vitesse radiale semble globalement supérieure à celle mesurée dans le VG. Cela semble indiquer que la recalibration du flux entrant sur la vitesse radiale maximale mesurée sur l'EchoCG Doppler unaliased n'est pas optimale. Cependant, le modèle *FDA* se comporte mieux que le modèle *OMC* et présente des similarités claires avec l'imagerie médicale.

Visible à la fin de la diastole et en accord avec iVFM, la recirculation ventriculaire couvre l'ensemble de la cavité et tourne dans le même sens. En regardant la vitesse radiale avec *aliasing*, une couche limite se distingue le long de la face antérieure à côté de la VA, celle-ci est également présente dans l'écoulement simulé mais pas sur l'écoulement unaliased. Contrairement à *OMC*, durant la fin de la systole, l'écoulement dans le ventricule n'est pas au repos par opposition aux mesures *in vivo*, ce qui indique que le flux entrant est surestimé et que la recalibration est perfectible. Avec le modèle *FDA*, au moment de l'onde E et l'onde A, la partie supérieure de la cavité montre une bonne ressemblance avec la mesure EchoCG. Plus précisément, que ce soit avec l'écoulement simulé ou mesuré, sont observables les éléments communs suivants : un jet à l'onde E de taille supérieure à celui de l'onde A et des recirculations de vitesses négatives de part et d'autre du jet à l'onde E et A.

La principale différence est la présence de sous-structures annexes, rendant l'écoulement simulé plus complexe que l'écoulement mesuré par l'imagerie. Cela est particulièrement visible à la diastase, mais aussi tout au long du cycle cardiaque avec un vortex secondaire significatif au niveau de l'apex.

En conclusion, le modèle *FDA* a permis de simuler un écoulement qui reprend les principaux critères de l'écoulement *in vivo* mais diffère par l'amplitude moyenne de la vitesse et la présence de recirculations locales qui ne sont pas visibles sur les données mesurées par imagerie. Au vu de la ressemblance au moment de l'onde E et l'onde A, la dynamique mitrale avec stratégie *Naturelle* semble globalement concorder avec la réalité physiologique.

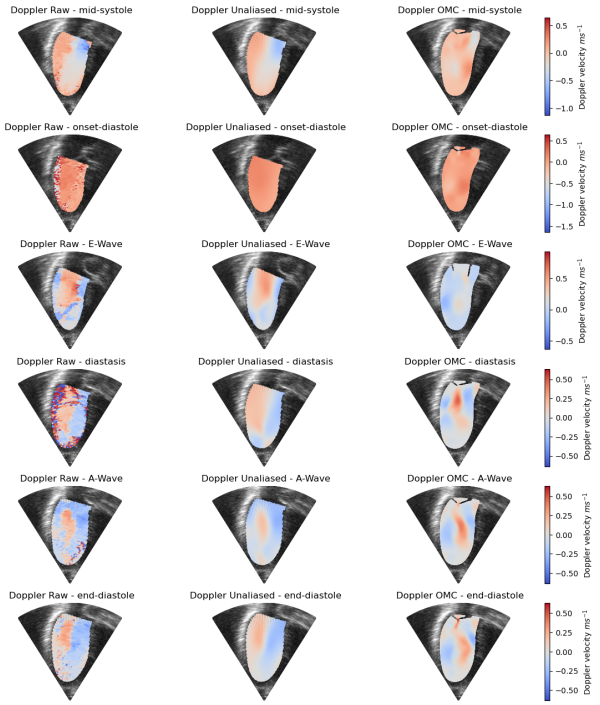


FIGURE 5.9 Comparaison entre les vitesses radiales brute mesurée avec EchoCG Doppler, *unaliased* calculée avec les résultats de la méthode iVFM et simulée avec le modèle *OMC* stratégie *Naturelle* à des moments clés du cycle cardiaque.

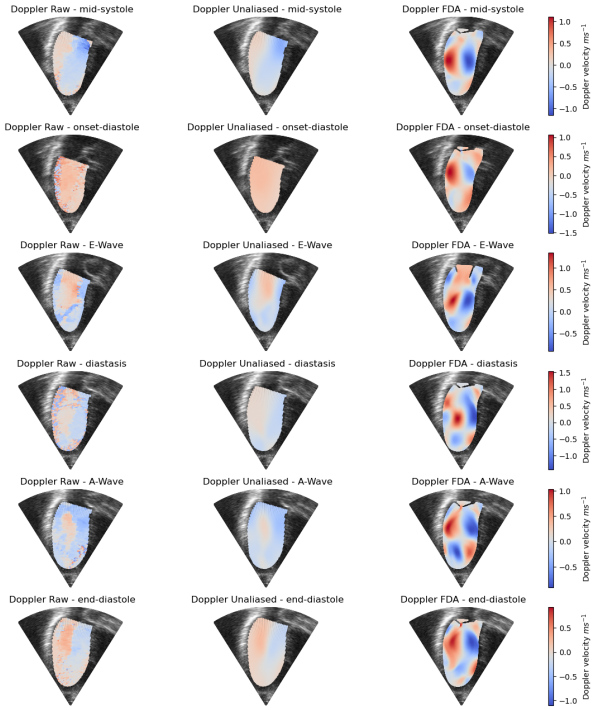


FIGURE 5.10 Comparaison entre les vitesses radiales brute mesurée avec EchoCG Doppler, *unaliased* calculée avec les résultats de la méthode iVFM et simulée avec le modèle *FDA* stratégie *Naturelle* à des moments clés du cycle cardiaque.

5.3 Conclusion

Pour conclure, la comparaison à la vitesse radiale extraite de l'EchoCG Doppler révèle un écoulement simulé plus complexe avec la présence de vortex locaux, là où l'écoulement EchoCG mesuré est plus lisse et dissipé. Cette observation est également valable avec la comparaison à la méthode iVFM, ce qui est attendu puisqu'elle utilise l'EchoCG Doppler comme donnée d'entrée pour calculer le champ de vitesse. Cette différence peut être liée à un déficit de dissipation par frottement à la paroi ou par phénomène de turbulence. Il se pourrait également que la mesure EchoCG tende à fournir un écoulement lissé ne permettant pas d'observer des variations locales de la vitesse, comme celles simulées par le modèle numérique. Finalement, le modèle *FDA* se comporte mieux que le modèle *OMC*, confirmant l'importance de l'amplitude du flux entrant pour simuler un flux physiologique. Il devient donc crucial pour les études 2D de modéliser le flux transverse au plan 3-chambres afin de rétablir la conservation de la masse. De plus, le modèle *FDA* tend à surestimer l'amplitude de la vitesse dans le VG, laissant à penser que la méthode de recalibration a surestimé l'am-

plitude du profil de vitesse entrant. Un critère plus précis que celui basé sur la vitesse radiale maximale enregistrée par l'EchoCG Doppler durant la diastole devrait être exploré. Finalement, de nombreuses similarités entre l'écoulement simulé avec le modèle *FDA* et la vitesse radiale mesurée par l'EchoCG Doppler existent et semblent démontrer que la stratégie *Naturelle* de la dynamique angulaire de notre modèle paramétrique de valve concorde avec la réalité *in vivo*. Ce modèle de faible complexité permet de reconstruire un écoulement tout au long du cycle cardiaque qui se rapproche des reconstructions discrètes dans le temps de la méthode iVFM. Dans la mesure où le temps de calcul nécessaire à la simulation de l'écoulement permanent (i.e. 3 cycles cardiaques) est de $\sim 45\ min$, et au vu de ses performances, la formulation avec le modèle *FDA* à stratégie *Naturelle* recalibrée avec l'EchoCG Doppler, semble prometteuse, comparée à d'autres modèles plus complexes et plus lourds.

CHAPITRE 6 CONCLUSION

Le dernier chapitre de ce mémoire permet de faire une synthèse des travaux réalisés (voir **Section 6.1**) et d'estimer si les objectifs présentés en Introduction ont été atteints. Les limitations de notre modèle numérique sont également discutées (voir **Section 6.2**). Les perspectives et améliorations futures de ce projet sont finalement évoquées pour conclure ce projet de maîtrise (voir **Section 6.3**).

6.1 Synthèse des travaux

Il est important de rappeler que ce projet a pour but de développer un modèle numérique de reconstruction de l'écoulement dans le VG de faible complexité présentant un compromis entre la résolution satisfaisante des principales caractéristiques de l'écoulement et le faible coût numérique. La perspective finale est d'intégrer cet outil dans un *workflow* de post-traitement de données d'imagerie médicale pouvant être utilisé sur de larges cohortes pour des études statistiques de biomarqueurs diagnostics basés sur l'hémodynamique du VG. Ce projet d'envergure est limité au sein de ce projet de maîtrise à tout ce qui a attrait à la conception du modèle numérique, et quatre principaux objectifs à atteindre sont énoncés en Introduction à la **Section 1.3**.

Le **Chapitre 2** rassemble d'une part les connaissances nécessaires à la compréhension du manuscrit et d'autre part une revue de littérature sur la modélisation numérique de l'hémodynamique du VG. Le rappel de connaissances concerne l'EchoCG, la segmentation d'images médicales, l'anatomie du cœur et la physiologie de la fonction cardiaque, et enfin les bases théoriques de la mécanique des fluides et de son implémentation numérique. La revue de littérature passe en revue différentes philosophies de modélisation que sont les modèles à Géométrie Prescrite (*GP*), à Interaction Fluide Structure Implicite (*IFSI*) et à Interaction Fluide Structure (*IFS*). Cette revue de littérature permet de valider la problématique de complexification des modèles numériques et la perte de leur capacité à constituer un outil applicable en clinique. Ce chapitre se conclue sur l'ensemble des hypothèses extraites de la littérature nécessaire à la conception du modèle numérique.

Le **Chapitre 3** couvre la présentation du modèle numérique avec une description de sa structure générale à la **Section 3.1** pour mieux apprécier à la **Section 3.2** les différentes formulations explorées. Ce modèle numérique se trouve dans la continuité d'un travail exploratoire précédent [1] dont les principales modifications concernent l'implémentation numé-

rique globale, ainsi que la conception et l'intégration d'un modèle paramétrique de VM. Ces modifications sont des enjeux principaux de ce projet et constituent les objectifs n°1 et n°2 du projet. La forme finale du modèle prend la forme d'un modèle fluide 2D à *GP* patient-spécifique avec formulation *ALE* basée sur des données EchoCG mode B avec intégration d'un modèle paramétrique de VM.

La conception du modèle paramétrique de VM comporte deux aspects, respectivement la reconstruction géométrique et la dynamique angulaire des valvules mitrales. Chacun de ses aspects est présenté à la **Section 3.1.3**. La reconstruction géométrique fait intervenir les paramètres β_t et β_n quantifiant respectivement l'asymétrie et le prolapsus de la VM, faisant l'objet d'une étude de sensibilité au **Chapitre 4** pour répondre à l'objectif n°3.

Le choix de la dynamique mitrale donne lieu à deux variantes, un modèle à Ouverture Mitrale Non Contrainte (*OMNC*) où un profil angulaire arbitraire est utilisé et un modèle à Ouverture Mitrale Contrainte (*OMC*) où l'application d'une contrainte permet d'obtenir un profil angulaire physiologique. Deux stratégies de contrainte sont développées, une stratégie dite *Naturelle* prenant en compte la fermeture partielle de la VM durant la diastase et une stratégie dite *Forcée* où la VM reste en position ouverte durant toute la diastole. Un dernier modèle à Flux Diastolique Amplifiée (*FDA*) qui amplifie artificiellement le flux entrant est proposé. Suite à des résultats préliminaires de ces modèles à la **Section 3.2**, seuls les modèles *FDA* et *OMC* avec stratégie *Naturelle* sont retenus pour la validation des performances faites au **Chapitre 5**. La validation du modèle qui constitue l'objectif n°4 est faite par comparaison avec la méthode iVFM et la mesure de la vitesse radiale issue de l'EchoCG Doppler.

L'objectif n°1 est réalisé tout au long du manuscrit en fournissant l'implémentation numérique Python permettant de réaliser l'ensemble des étapes nécessaires à la mise en place et l'utilisation de ce modèle numérique. Les codes annotés sont répertoriés à l'**Annexe D**. Ce répertoire a pour objectif de fournir une trace écrite assurant la transparence du travail réalisé et pouvant être utilisé comme référence par autrui pour la construction d'un *workflow* avec l'éco-système PyAnsys.

L'objectif n°2 est atteint en deux étapes. D'abord, le **Chapitre 3** introduit le développement détaillé du modèle paramétrique de VM et des différentes variations qui en découlent, puis présente des résultats préliminaires validant que l'implémentation à été faite avec succès.

L'objectif n°3 concerne l'étude de sensibilité des paramètres géométriques β_t et β_n sur l'écoulement dans le VG et fait l'objet du **Chapitre 4**. Pour ce faire, une nouvelle méthode de cartographie et catégorisation du VTD est développée et présentée, voir **Annexe C**. Cette étude paramétrique révèle que le paramètre β_t a une influence critique sur l'écoulement tandis que le paramètre β_n ne montre pas d'influence significative. En conclusion, une valeur

physiologique de $\beta_t \sim 0.7$ permet la formation d'un vortex diastolique donnant lieu à une recirculation efficace dans le VG, tandis que la valeur $\beta_t = 0.5$ (i.e. symétrie de la VM) produit un écoulement désordonné et la valeur $\beta_t = 0.3$ (i.e. asymétrie inversée) donne lieu à une recirculation inversée, tous deux étant défavorables au remplissage diastolique et à l'éjection systolique. La cartographie et la catégorisation du VTD, bien que sensibles aux variations de l'écoulement, ne fournissent pas de différences quantitatives concernant la performance cardiaque. En comparant avec des valeurs de références, elles mettent en lumière une sous-estimation du flux entrant basé sur la déformation géométrique 2D du VG. Cela rend la méthode peu adaptée à l'utilisation 2D, mais pertinente et prometteuse dans le cas 3D.

L'objectif n°4 réalisé au **Chapitre 5** est l'ultime étape de la conception d'un modèle numérique : la validation. Cette validation est faite en comparaison avec la méthode iVFM et la vitesse radiale issue de l'EchoCG Doppler. La comparaison avec la méthode iVFM situe notre travail par rapport à d'autres outils de reconstruction de l'écoulement utilisés en recherche, là où la mesure par imagerie définit la référence *in vivo*. En accord avec la conclusion du **Chapitre 4** sur la sous-estimation du flux entrant dans le VG, le modèle *OMC* se comporte mal tandis que le modèle *FDA* avec flux entrant recalibré sur l'EchoCG Doppler simule un écoulement proche à la fois de iVFM et de l'EchoCG Doppler. Que ce soit le modèle *FDA* ou *OMC*, le champ de vitesse simulé présente une structure plus complexe avec des recirculations locales au niveau de l'apex et de la VA, en comparaison avec un écoulement plus lisse qu'il soit calculé par iVFM ou mesuré par EchoCG Doppler. Cette différence pourrait venir d'un manque de dissipation des vortex en l'absence de modélisation de la turbulence et de rugosité de la paroi ventriculaire. La corrélation entre les résultats et la réalité *in vivo* en utilisant la stratégie de dynamique mitrale *Naturelle* confirme l'importance de modéliser la fermeture partielle de la VM durant la diastase et valide notre modèle paramétrique. Finalement, ce modèle de faible complexité reconstruit un écoulement tout au long du cycle cardiaque qui se rapproche des reconstructions discrètes dans le temps de la méthode iVFM. Finalement, dans la mesure où le temps de calcul nécessaire à la simulation de l'écoulement permanent (i.e. 3 cycles cardiaques) est de 45 min, et au vu de ses performances, la formulation avec le modèle *FDA* à stratégie *Naturelle* recalibrée avec l'EchoCG Doppler, semble prometteuse comparée à d'autres modèles plus complexes et plus lourds. Bien que moins performant en terme de réalisme, ce modèle simule avec succès un écoulement physiologique patient-spécifique en un temps limité basé sur des mesures EchoCG mode B communément réalisées en clinique.

6.2 Limitations de l'approche proposée

Comme l'ensemble des modèles à *GP*, ce modèle est sensible à la reconstruction géométrique du VG dans le temps. Des approximations sont faites autant au moment de la segmentation du VG qu'à la création de la géométrie numérique. Enfin, il est nécessaire de réaliser une étape d'interpolation temporelle puisque la résolution temporelle de l'EchoCG n'est pas suffisante. Dans la mesure où l'écoulement est sensible aux mouvements du VG, ces approximations forment une limite à la simulation de l'écoulement. Avec les nouvelles méthodes de segmentation et l'augmentation de la résolution temporelle de l'EchoCG, cette limitation sera de moins en moins significative.

D'autres limitations sont des conséquences de l'utilisation d'hypothèses de simplification afin de limiter la complexité numérique et le temps de calcul. Plus précisément, l'écoulement a été supposé laminaire, ce qui semble valide si l'on réalise une étude sur un seul cycle cardiaque. Si l'on souhaite simuler plusieurs cycles et limiter l'influence de l'initialisation du fluide, une formulation turbulente dissiperait ces vortex secondaires et faciliterait l'organisation de l'écoulement. De plus, au vu des résultats, l'hypothèse que le flux diastolique entrant dans le VG est égal à la variation de surface du VG dans le plan 3-chambres donne lieu à une sous-estimation de la vitesse de l'écoulement au niveau de la VM et altère l'écoulement simulé. Des résultats plus probants ont été observés en augmentant artificiellement le flux entrant, au détriment de la conservation de la masse et en augmentant par la-même les résidus de simulation.

Il est à noter aussi que les paramètres d'entrée du modèle paramétrique de VM (β_t, β_n) pour la reconstruction géométrique et $(\theta_l^{max}, \theta_r^{max})$ pour la dynamique angulaire sont estimés à partir des images mode B. L'estimation de ces paramètres est donc sensible au bruit de l'image mais aussi à la subjectivité de l'opérateur du dispositif d'imagerie dans le choix du plan d'acquisition de l'image.

Finalement, dans le cadre de ce projet, il n'a pas été possible, par manque de temps, de vérifier la robustesse de notre approche sur une cohorte de patient. Il n'a donc pas été possible de vérifier que le modèle numérique soit suffisamment robuste pour assurer la comparaison inter-individuelle.

6.3 Perspectives et améliorations

Ce manuscrit se clôture avec des pistes d'améliorations et les perspectives qu'ouvre ce projet de maîtrise.

Les **Chapitres 4** et **5** ont mis en évidence que la sous-estimation du flux diastolique entrant en le supposant égal à la variation géométrique dans le plan 3-chambres a des conséquences directes sur l'écoulement et qu'il est critique de fournir un flux entrant correct pour obtenir des résultats physiologiques. Dans ce travail, cela a été réalisé en augmentant artificiellement la vitesse d'entrée au dépend de la conservation de la masse. C'est une problématique qui doit être adressée. Une solution serait de procéder à l'extrusion transverse (selon l'axe e_z) de la géométrie 2D (dans le plan xy) sur une distance d_e de quelques millimètres et d'imposer une condition de symétrie par rapport au plan médian $z = d_e/2$ tout en autorisant le fluide à sortir au niveau des faces de vecteur normal dirigé selon $\pm e_z$. Le flux entrant serait modifié avec un coefficient d'amplification $a_U = a_U(t)$ d'abord constant puis variable dans le temps. Cette loi temporelle doit être estimée à partir d'un EchoCG Doppler limité à la zone proche de la VM, puis généralisée.

Au vu des résultats, il serait également pertinent de modifier le régime de simulation de laminaire à turbulent avec la modélisation de la dissipation turbulente. L'objectif est de limiter la propagation de cycle en cycle de vortex annexes. Dans la même logique, il faudrait mesurer l'influence sur l'écoulement de la rugosité de la paroi du VG. À condition que le coût numérique n'en soit pas trop augmenté, il serait possible de créer un modèle poreux le long de la paroi pour modéliser les trabécules charnues.

Dans l'objectif d'augmenter la vitesse de calcul et réduire le temps de simulation, il est envisageable d'utiliser un solveur utilisant le GPU (*Graphics Processing Unit*) plutôt qu'un solveur utilisant le CPU (*Central Processing Unit*). Cette avancée technique a permis dans d'autres domaines de la mécanique des fluides numériques de réduire drastiquement le temps de calcul. Au sein de la suite Ansys, l'option de maillage dynamique n'est pas encore disponible avec un solveur GPU, mais devrait l'être dans les prochaines années.

Concernant la limitation liée à l'estimation des paramètres du modèles paramétriques de la VM, il serait intéressant d'envisager une optimisation de ces paramètres en comparant l'écoulement simulé avec une acquisition EchoCG Doppler de la cavité complète. Il s'agirait alors d'un problème inverse où l'on cherche la configuration qui permet de simuler un écoulement similaire à celui mesuré *in vivo*. Même si la VM reconstruite diffère dans sa géométrie et sa dynamique de celle présente sur l'EchoCG, cela aurait une influence uniquement locale au niveau de la VM. Il faudrait utiliser une stratégie d'optimisation globale qui limite le nombre d'essais puisque le temps de simulation d'un essai est long. Une optimisation Bayésienne utilisant des processus Gaussiens semble toute indiquée.

Finalement, une étude complémentaire avec l'utilisation du modèle sur une cohorte réduite semble nécessaire pour vérifier la robustesse de la méthode. Dans l'objectif final de créer un

outil de post-traitement d'images EchoCG, il est critique que la méthode soit applicable à une large population. Il faut alors tester autant les routines de *set-up* en amont de la simulation que la simulation elle-même. L'attention devra être portée à l'unicité au sein de la cohorte des écoulements simulés pour estimer sa performance à la comparaison inter-individuelle.

RÉFÉRENCES

[1] P. Dubois, "Modélisation hémodynamique du ventricule gauche à partir d'images échocardiographiques," Mémoire de maîtrise, Polytechnique Montréal, août 2023. [En ligne]. Disponible : https://publications.polymtl.ca/54856/

[2] P. Nihoyannopoulos et J. Kisslo, édit., *Echocardiography*. London : Springer, 2009. [En ligne]. Disponible : http://link.springer.com/10.1007/978-1-84882-293-1

[3] O. . T. V. . Tatoute, "English : Waves emitted by a source moving from the right to the left. The frequency is higher on the left (ahead of the source) than on the right." janv. 2006. [En ligne]. Disponible : https://commons.wikimedia.org/wiki/File:Doppler_effect_diagrammatic.svg?uselang=fr

[4] Wapcaplet, "Diagram of the human heart, created by Wapcaplet in Sodipodi. Cropped by Yaddah to remove white space (this cropping is not the same as Wapcaplet's original crop)." juin 2006. [En ligne]. Disponible : https://commons.wikimedia.org/wiki/File:Diagram_of_the_human_heart_(cropped).svg

[5] D. r. o. w. o. D. R. a. S. b. xavax, "English : A Wiggers diagram, showing the cardiac cycle events occuring in the left ventricle." mars 2012. [En ligne]. Disponible : https://commons.wikimedia.org/wiki/File:Wiggers_Diagram.svg?uselang=fr

[6] Andyhenton83, "English : Pressure-Volume Loop with stages of cardiac cycle." janv. 2008. [En ligne]. Disponible : https://commons.wikimedia.org/wiki/File:Cardiac_Pressure_Volume_Loop.jpg

[7] J. A. Vierendeels, K. Riemslagh, E. Dick et P. R. Verdonck, "Computer simulation of intraventricular flow and pressure gradients during diastole," *Journal of Biomechanical Engineering*, vol. 122, n°. 6, p. 667–674, déc. 2000.

[8] Q. Long, R. Merrifield, G. Z. Yang, P. J. Kilner, D. N. Firmin et X. Y. Xu, "The influence of inflow boundary conditions on intra left ventricle flow predictions," *Journal of Biomechanical Engineering*, vol. 125, n°. 6, p. 922–927, déc. 2003.

[9] F. Domenichini, G. Pedrizzetti et B. Baccani, "Three-dimensional filling flow into a model left ventricle," *Journal of Fluid Mechanics*, vol. 539, p. 179–198, sept. 2005. [En ligne]. Disponible : https://www.cambridge.org/core/journals/journal-of-fluid-mechanics/article/threedimensional-filling-flow-into-a-model-left-ventricle/98751DA4CA4D1016697C65ACB1645A0E

[10] S. N. Doost, L. Zhong, B. Su et Y. S. Morsi, "The numerical analysis of non-Newtonian blood flow in human patient-specific left ventricle," *Computer Methods*

and Programs in Biomedicine, vol. 127, p. 232–247, avr. 2016. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0169260715300742

[11] J. O. Mangual, E. Kraigher-Krainer, A. De Luca, L. Toncelli, A. Shah, S. Solomon, G. Galanti, F. Domenichini et G. Pedrizzetti, "Comparative numerical study on left ventricular fluid dynamics after dilated cardiomyopathy," *Journal of Biomechanics*, vol. 46, n°. 10, p. 1611–1617, juin 2013.

[12] Y. Cheng, H. Oertel et T. Schenkel, "Fluid-structure coupled CFD simulation of the left ventricular flow during filling phase," *Annals of Biomedical Engineering*, vol. 33, n°. 5, p. 567–576, mai 2005.

[13] S. S. Khalafvand, E. Y. K. Ng et L. Zhong, "CFD simulation of flow through heart : a perspective review," *Computer Methods in Biomechanics and Biomedical Engineering*, vol. 14, n°. 1, p. 113–132, 2011.

[14] W. Mao, A. Caballero, R. McKay, C. Primiano et W. Sun, "Fully-coupled fluid-structure interaction simulation of the aortic and mitral valves in a realistic 3D left ventricle model," *PLOS ONE*, vol. 12, n°. 9, p. e0184729, sept. 2017, publisher : Public Library of Science. [En ligne]. Disponible : https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0184729

[15] V. Govindarajan, J. Mousel, H. S. Udaykumar, S. C. Vigmostad, D. D. McPherson, H. Kim et K. B. Chandran, "Synergy between Diastolic Mitral Valve Function and Left Ventricular Flow Aids in Valve Closure and Blood Transport during Systole," *Scientific Reports*, vol. 8, n°. 1, p. 6187, avr. 2018, publisher : Nature Publishing Group. [En ligne]. Disponible : https://www.nature.com/articles/s41598-018-24469-x

[16] J. Eriksson, C. J. Carlhäll, P. Dyverfeldt, J. Engvall, A. F. Bolger et T. Ebbers, "Semi-automatic quantification of 4D left ventricular blood flow," *Journal of Cardiovascular Magnetic Resonance*, vol. 12, n°. 1, p. 9, févr. 2010. [En ligne]. Disponible : https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2831022/

[17] F. Vixège, A. Berod, Y. Sun, S. Mendez, O. Bernard, N. Ducros, P.-Y. Courand, F. Nicoud et D. Garcia, "Physics-constrained intraventricular vector flow mapping by color Doppler," *Physics in Medicine and Biology*, vol. 66, n°. 24, déc. 2021.

[18] "Insuffisance cardiaque." [En ligne]. Disponible : https://icm-mhi.org/soins-et-services/insuffisance-cardiaque/

[19] J. Milette, "Données statistiques sur l'insuffisance cardiaque." [En ligne]. Disponible : https://sqic.org/donnees-statistiques-linsuffisance-cardiaque/

[20] A. De Vecchi, A. Gomez, K. Pushparajah, T. Schaeffter, D. Nordsletten, J. Simpson, G. Penney et N. Smith, "Towards a fast and efficient approach for modelling the patient-

specific ventricular haemodynamics," *Progress in biophysics and molecular biology*, vol. 116, août 2014.

[21] G. Pedrizzetti, G. La Canna, O. Alfieri et G. Tonti, "The vortex–an early predictor of cardiovascular outcome ?" *Nature Reviews. Cardiology*, vol. 11, n°. 9, p. 545–553, sept. 2014.

[22] A. Kheradvar, C. Rickers, D. Morisawa, M. Kim, G.-R. Hong et G. Pedrizzetti, "Diagnostic and prognostic significance of cardiovascular vortex formation," *Journal of Cardiology*, vol. 74, n°. 5, p. 403–411, nov. 2019. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0914508719301431

[23] A. Kheradvar et M. Gharib, "On Mitral Valve Dynamics and its Connection to Early Diastolic Flow," *Annals of Biomedical Engineering*, vol. 37, n°. 1, p. 1–13, janv. 2009. [En ligne]. Disponible : https://doi.org/10.1007/s10439-008-9588-7

[24] ——, "Influence of Ventricular Pressure Drop on Mitral Annulus Dynamics Through the Process of Vortex Ring Formation," *Annals of Biomedical Engineering*, vol. 35, n°. 12, p. 2050–2064, déc. 2007. [En ligne]. Disponible : https://doi.org/10.1007/s10439-007-9382-y

[25] A. Kheradvar, R. Assadi, A. Falahatpisheh et P. P. Sengupta, "Assessment of Transmitral Vortex Formation in Patients with Diastolic Dysfunction," *Journal of the American Society of Echocardiography*, vol. 25, n°. 2, p. 220–227, févr. 2012. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0894731711007619

[26] M. Gharib, E. Rambod, A. Kheradvar, D. J. Sahn et J. O. Dabiri, "Optimal vortex formation as an index of cardiac health," *Proceedings of the National Academy of Sciences*, vol. 103, n°. 16, p. 6305–6308, avr. 2006, publisher : Proceedings of the National Academy of Sciences. [En ligne]. Disponible : https://www.pnas.org/doi/full/10.1073/pnas.0600520103

[27] R. Wexler, T. Elton, A. Pleister et D. Feldman, "Cardiomyopathy : An Overview," *American family physician*, vol. 79, n°. 9, p. 778–784, mai 2009. [En ligne]. Disponible : https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2999879/

[28] M. Ghafarinatanzi, "Identification of Mechanical Properties for Patient-Specific Left Ventricle Models from CMR using Virtual Fields Method," phd, Polytechnique Montréal, juin 2023. [En ligne]. Disponible : https://publications.polymtl.ca/54161/

[29] M.-O. Lapointe, "Évaluation des propriétés mécaniques du tissu cardiaque par échocardiographie," Mémoire de maîtrise, École Polytechnique de Montréal, août 2016. [En ligne]. Disponible : https://publications.polymtl.ca/2201/

[30] M. Gamba, "Subtle Changes in Hyperelastic Properties of Myocardium With Cardiotoxicity Remodeling From Cardiac Magnetic Resonance," Mémoire de

maîtrise, Polytechnique Montréal, mai 2019. [En ligne]. Disponible : https://publications.polymtl.ca/3846/

[31] P. Soni, S. Kumar, B. V. R. Kumar, S. K. Rai, A. Verma et O. Shankar, "A comprehensive review on CFD simulations of left ventricle hemodynamics : numerical methods, experimental validation techniques, and emerging trends," *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, vol. 46, nº. 5, p. 301, avr. 2024. [En ligne]. Disponible : https://doi.org/10.1007/s40430-024-04875-1

[32] M. Hirschhorn, V. Tchantchaleishvili, R. Stevens, J. Rossano et A. Throckmorton, "Fluid–structure interaction modeling in cardiovascular medicine – A systematic review 2017–2019," *Medical Engineering & Physics*, vol. 78, p. 1–13, avr. 2020. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S1350453320300199

[33] F. Vixege, "Écoulement intraventriculaire 4-D par échocardiographie Doppler," phdthesis, Université de Lyon, mars 2022. [En ligne]. Disponible : https://theses.hal.science/tel-03827041

[34] K. C. Assi, E. Gay, C. Chnafa, S. Mendez, F. Nicoud, J. F. P. J. Abascal, P. Lantelme, F. Tournoux et D. Garcia, "Intraventricular vector flow mapping-a Doppler-based regularized problem with automatic model selection," *Physics in Medicine and Biology*, vol. 62, nº. 17, p. 7131–7147, août 2017.

[35] F. Vixège, A. Berod, P.-Y. Courand, S. Mendez, F. Nicoud, P. Blanc-Benon, D. Vray et D. Garcia, "Full-volume three-component intraventricular vector flow mapping by triplane color Doppler," *Physics in Medicine and Biology*, vol. 67, nº. 9, avr. 2022.

[36] H. Feigenbaum, "Evolution of Echocardiography," *Circulation*, vol. 93, nº. 7, p. 1321–1327, avr. 1996, publisher : American Heart Association. [En ligne]. Disponible : https://www.ahajournals.org/doi/10.1161/01.CIR.93.7.1321

[37] M. Tanter et M. Fink, "Ultrafast imaging in biomedical ultrasound," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 61, nº. 1, p. 102–119, janv. 2014. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/6689779

[38] M. Cikes, L. Tong, G. R. Sutherland et J. D'hooge, "Ultrafast Cardiac Ultrasound Imaging : Technical Principles, Applications, and Clinical Benefits," *JACC : Cardiovascular Imaging*, vol. 7, nº. 8, p. 812–823, août 2014. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S1936878X14004410

[39] N. S. Anavekar et J. K. Oh, "Doppler echocardiography : a contemporary review," *Journal of Cardiology*, vol. 54, nº. 3, p. 347–358, déc. 2009.

[40] A. S. Omran, A. A. Arifi et A. A. Mohamed, "Echocardiography of the mitral valve," *Journal of the Saudi Heart Association*, vol. 22, nº. 3, p. 165–170, juill. 2010. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S1016731510004161

[41] A. Falahatpisheh, G. Pedrizzetti et A. Kheradvar, “Three-dimensional reconstruction of cardiac flows based on multi-planar velocity fields,” *Experiments in Fluids*, vol. 55, n°. 11, p. 1848, nov. 2014. [En ligne]. Disponible : https://doi.org/10.1007/s00348-014-1848-8

[42] J. Provost, C. Papadacci, J. E. Arango, M. Imbault, J.-L. Gennisson, M. Tanter et M. Pernot, “3D Ultrafast Ultrasound Imaging In Vivo,” *Physics in medicine and biology*, vol. 59, n°. 19, p. L1–L13, oct. 2014. [En ligne]. Disponible : https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4820600/

[43] J. H. Goldman, N. B. Schiller, D. C. Lim, R. F. Redberg et E. Foster, “Usefulness of stroke distance by echocardiography as a surrogate marker of cardiac output that is independent of gender and size in a normal population,” *The American Journal of Cardiology*, vol. 87, n°. 4, p. 499–502, A8, févr. 2001.

[44] H. Baumgartner, J. Hung, J. Bermejo, J. B. Chambers, A. Evangelista, B. P. Griffin, B. Iung, C. M. Otto, P. A. Pellikka, M. Quiñones, American Society of Echocardiography et European Association of Echocardiography, “Echocardiographic assessment of valve stenosis : EAE/ASE recommendations for clinical practice,” *Journal of the American Society of Echocardiography : Official Publication of the American Society of Echocardiography*, vol. 22, n°. 1, p. 1–23 ; quiz 101–102, janv. 2009.

[45] A. M. Bavo, A. M. Pouch, J. Degroote, J. Vierendeels, J. H. Gorman, R. C. Gorman et P. Segers, “Patient-specific CFD simulation of intraventricular haemodynamics based on 3D ultrasound imaging,” *Biomedical Engineering Online*, vol. 15, n°. 1, p. 107, sept. 2016.

[46] S. N. Doost, L. Zhong, B. Su et Y. S. Morsi, “Two-dimensional intraventricular flow pattern visualization using the image-based computational fluid dynamics,” *Computer Methods in Biomechanics and Biomedical Engineering*, vol. 20, n°. 5, p. 492–507, avr. 2017.

[47] F. Domenichini, G. Querzoli, A. Cenedese et G. Pedrizzetti, “Combined experimental and numerical analysis of the flow structure into the left ventricle,” *Journal of Biomechanics*, vol. 40, n°. 9, p. 1988–1994, 2007.

[48] C. S. Peskin et D. M. McQueen, “A three-dimensional computational method for blood flow in the heart I. Immersed elastic fibers in a viscous incompressible fluid,” *Journal of Computational Physics*, vol. 81, n°. 2, p. 372–405, avr. 1989. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/0021999189902131

[49] H. S. Wong, W. X. Chan, W. Mao et C. H. Yap, “3D velocity and pressure field reconstruction in the cardiac left ventricle via physics informed neural network from echocardiography guided by 3D color Doppler,” *Computer Methods and*

Programs in Biomedicine, vol. 263, p. 108671, mai 2025. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0169260725000884

[50] H. J. Ling, S. Bru, J. Puig, F. Vixège, S. Mendez, F. Nicoud, P.-Y. Courand, O. Bernard et D. Garcia, "Physics-Guided Neural Networks for Intraventricular Vector Flow Mapping," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 71, nº. 11, p. 1377–1388, nov. 2024, arXiv :2403.13040 [eess]. [En ligne]. Disponible : http://arxiv.org/abs/2403.13040

[51] S. G. Lacerda, A. F. da Rocha, D. F. Vasconcelos, J. L. A. de Carvalho, I. G. Sene et J. F. Camapum, "Left ventricle segmentation in echocardiography using a radial-search-based image processing algorithm," *Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Annual International Conference*, vol. 2008, p. 222–225, 2008.

[52] D. Krishnaswamy, A. R. Hareendranathan, T. Suwatanaviroj, P. Boulanger, H. Becher, M. Noga et K. Punithakumar, "A New Semi-automated Algorithm for Volumetric Segmentation of the Left Ventricle in Temporal 3D Echocardiography Sequences," *Cardiovascular Engineering and Technology*, vol. 13, nº. 1, p. 55–68, févr. 2022.

[53] Z. Qian, T. Hu, J. Wang et Z. Yang, "U-shape-based network for left ventricular segmentation in echocardiograms with contrastive pretraining," *Scientific Reports*, vol. 14, nº. 1, p. 29689, nov. 2024, publisher : Nature Publishing Group. [En ligne]. Disponible : https://www.nature.com/articles/s41598-024-81523-7

[54] Y. Lei, Y. Fu, J. Roper, K. Higgins, J. D. Bradley, W. J. Curran, T. Liu et X. Yang, "Echocardiographic image multi-structure segmentation using Cardiac-SegNet," *Medical Physics*, vol. 48, nº. 5, p. 2426–2437, 2021, _eprint : https ://onlinelibrary.wiley.com/doi/pdf/10.1002/mp.14818. [En ligne]. Disponible : https://onlinelibrary.wiley.com/doi/abs/10.1002/mp.14818

[55] Y. Ali, F. Janabi-Sharifi et S. Beheshti, "Echocardiographic image segmentation using deep Res-U network," *Biomedical Signal Processing and Control*, vol. 64, p. 102248, févr. 2021. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S1746809420303761

[56] M. Liao, Y. Lian, Y. Yao, L. Chen, F. Gao, L. Xu, X. Huang, X. Feng et S. Guo, "Left Ventricle Segmentation in Echocardiography with Transformer," *Diagnostics*, vol. 13, nº. 14, p. 2365, juill. 2023. [En ligne]. Disponible : https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10378102/

[57] T. Kim, M. Hedayat, V. V. Vaitkus, M. Belohlavek, V. Krishnamurthy et I. Borazjani, "Automatic segmentation of the left ventricle in echocardiographic images using convolutional neural networks," *Quantitative Imaging in Medicine*

and Surgery, vol. 11, n°. 5, p. 1763–1781, mai 2021. [En ligne]. Disponible : https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8047352/

[58] R. L. Drake, F. Duparc, A. W. M. Mitchell et A. W. Vogl, *Gray's Anatomie - Les fondamentaux*. Elsevier Health Sciences, août 2018, google-Books-ID : oBZnDwAAQBAJ.

[59] M. K. Loushin, J. L. Quill et P. A. Iaizzo, "Mechanical Aspects of Cardiac Performance," dans *Handbook of Cardiac Anatomy, Physiology, and Devices*, P. A. Iaizzo, édit. Totowa, NJ : Humana Press, 2009, p. 271–296. [En ligne]. Disponible : https://doi.org/10.1007/978-1-60327-372-5_18

[60] P. A. Iaizzo, édit., *Handbook of Cardiac Anatomy, Physiology, and Devices*. Cham : Springer International Publishing, 2015. [En ligne]. Disponible : https://link.springer.com/10.1007/978-3-319-19464-6

[61] D. McQueen et C. Peskin, "Shared-Memory Parallel Vector Implementation of the Immersed Boundary Method for the Computation of Blood Flow in the Beating Mammalian Heart," *The Journal of Supercomputing*, vol. 11, n°. 3, p. 213–236, nov. 1997. [En ligne]. Disponible : https://doi.org/10.1023/A:1007951707260

[62] D. M. McQueen et C. S. Peskin, "A three-dimensional computer model of the human heart for studying cardiac fluid dynamics," *ACM SIGGRAPH Computer Graphics*, vol. 34, n°. 1, p. 56–60, févr. 2000. [En ligne]. Disponible : https://dl.acm.org/doi/10.1145/563788.604453

[63] G. Pedrizzetti, A. R. Martiniello, V. Bianchi, A. D'Onofrio, P. Caso et G. Tonti, "Cardiac fluid dynamics anticipates heart adaptation," *Journal of Biomechanics*, vol. 48, n°. 2, p. 388–391, janv. 2015.

[64] A. Kheradvar et G. Pedrizzetti, *Vortex Formation in the Cardiovascular System*. London : Springer, 2012. [En ligne]. Disponible : http://link.springer.com/10.1007/978-1-4471-2288-3

[65] G. Pedrizzetti et F. Domenichini, "Nature optimizes the swirling flow in the human left ventricle," *Physical Review Letters*, vol. 95, n°. 10, p. 108101, sept. 2005.

[66] S. Sperlongano, A. D'Andrea, D. Mele, V. Russo, V. Pergola, A. Carbone, F. Ilardi, M. Di Maio, R. Bottino, F. Giallauria, E. Bossone et P. Golino, "Left Ventricular Deformation and Vortex Analysis in Heart Failure : From Ultrasound Technique to Current Clinical Application," *Diagnostics*, vol. 11, n°. 5, p. 892, mai 2021, number : 5 Publisher : Multidisciplinary Digital Publishing Institute. [En ligne]. Disponible : https://www.mdpi.com/2075-4418/11/5/892

[67] A. A. Bakir, A. Al Abed, N. H. Lovell et S. Dokos, "Multiphysics Computational Modelling of the Cardiac Ventricles," *IEEE reviews in biomedical engineering*, vol. 15, p. 309–324, 2022.

[68] V. Vasudevan, A. J. J. Low, S. P. Annamalai, S. Sampath, K. K. Poh, T. Totman, M. Mazlan, G. Croft, A. M. Richards, D. P. V. de Kleijn, C.-L. Chin et C. H. Yap, "Flow dynamics and energy efficiency of flow in the left ventricle during myocardial infarction," *Biomechanics and Modeling in Mechanobiology*, vol. 16, n°. 5, p. 1503–1517, oct. 2017. [En ligne]. Disponible : https://doi.org/10.1007/s10237-017-0902-x

[69] A. M. Bavo, A. M. Pouch, J. Degroote, J. Vierendeels, J. H. Gorman, R. C. Gorman et P. Segers, "Patient-specific CFD models for intraventricular flow analysis from 3D ultrasound imaging : Comparison of three clinical cases," *Journal of Biomechanics*, vol. 50, p. 144–150, janv. 2017.

[70] S. S. Khalafvand, E. Y. K. Ng, L. Zhong et T. K. Hung, "Fluid-dynamics modelling of the human left ventricle with dynamic mesh for normal and myocardial infarction : preliminary study," *Computers in Biology and Medicine*, vol. 42, n°. 8, p. 863–870, août 2012.

[71] S. K. Dahl, J. Vierendeels, J. Degroote, S. Annerel, L. R. Hellevik et B. Skallerud, "FSI simulation of asymmetric mitral valve dynamics during diastolic filling," *Computer Methods in Biomechanics and Biomedical Engineering*, vol. 15, n°. 2, p. 121–130, 2012.

[72] C. Chnafa, S. Mendez et F. Nicoud, "Image-Based Simulations Show Important Flow Fluctuations in a Normal Left Ventricle : What Could be the Implications ?" *Annals of Biomedical Engineering*, vol. 44, n°. 11, p. 3346–3358, nov. 2016.

[73] ——, "Image-based large-eddy simulation in a realistic left heart," *Computers & Fluids*, vol. 94, p. 173–187, mai 2014. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0045793014000486

[74] J. F. Wendt, édit., *Computational Fluid Dynamics*. Berlin, Heidelberg : Springer, 2009. [En ligne]. Disponible : http://link.springer.com/10.1007/978-3-540-85056-4

[75] A. W. Date, *Introduction to Computational Fluid Dynamics*. Cambridge : Cambridge University Press, 2005. [En ligne]. Disponible : https://www.cambridge.org/core/books/introduction-to-computational-fluid-dynamics/DAE7132CD57EA9D487BECEE529B86A7E

[76] N. R. Saber, N. B. Wood, A. D. Gosman, R. D. Merrifield, G.-Z. Yang, C. L. Charrier, P. D. Gatehouse et D. N. Firmin, "Progress towards patient-specific computational flow modeling of the left heart via combination of magnetic resonance imaging with computational fluid dynamics," *Annals of Biomedical Engineering*, vol. 31, n°. 1, p. 42–52, janv. 2003.

[77] B. Baccani, F. Domenichini et G. Pedrizzetti, "Vortex dynamics in a model left ventricle during filling," *European Journal of Mechanics - B/Fluids*, vol. 21, n°. 5, p.

527–543, sept. 2002. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0997754602012001

[78] ——, “Model and influence of mitral valve opening during the left ventricular filling,” *Journal of Biomechanics*, vol. 36, n°. 3, p. 355–361, mars 2003. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0021929002004207

[79] Q. Long, R. Merrifield, X. Xu, P. Kilner, D. Firmin et G. Yang, “Intra-ventricular blood flow simulation with patient specific geometry,” dans *4th International IEEE EMBS Special Topic Conference on Information Technology Applications in Biomedicine, 2003.*, avr. 2003, p. 126–129. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/1222489

[80] T. Schenkel, M. Malve, M. Reik, M. Markl, B. Jung et H. Oertel, “MRI-based CFD analysis of flow in a human left ventricle : methodology and application to a healthy heart,” *Annals of Biomedical Engineering*, vol. 37, n°. 3, p. 503–515, mars 2009.

[81] S. N. Doost, D. Ghista, B. Su, L. Zhong et Y. S. Morsi, “Heart blood flow simulation : a perspective review,” *Biomedical Engineering Online*, vol. 15, n°. 1, p. 101, août 2016.

[82] M.-H. Moosavi, N. Fatouraee, H. Katoozian, A. Pashaei, O. Camara et A. F. Frangi, “Numerical simulation of blood flow in the left ventricle and aortic sinus using magnetic resonance imaging and computational fluid dynamics,” *Computer Methods in Biomechanics and Biomedical Engineering*, vol. 17, n°. 7, p. 740–749, mai 2014.

[83] F. Domenichini, “On the consistency of the direct forcing method in the fractional step solution of the Navier–Stokes equations,” *Journal of Computational Physics*, vol. 227, n°. 12, p. 6372–6384, juin 2008. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0021999108001617

[84] Y. Bao, A. Donev, B. E. Griffith, D. M. McQueen et C. S. Peskin, “An Immersed Boundary method with divergence-free velocity interpolation and force spreading,” *Journal of Computational Physics*, vol. 347, p. 183–206, oct. 2017. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0021999117304953

[85] C. H. Tai, K. M. Liew et Y. Zhao, “Numerical simulation of 3D fluid–structure interaction flow using an immersed object method with overlapping grids,” *Computers & Structures*, vol. 85, n°. 11, p. 749–762, juin 2007. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0045794907000363

[86] H. Gao, L. Feng, N. Qi, C. Berry, B. E. Griffith et X. Luo, “A coupled mitral valve—left ventricle model with fluid–structure interaction,” *Medical Engineering & Physics*, vol. 47, p. 128–136, sept. 2017. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S1350453317301819

[87] J. O. Mangual, F. Domenichini et G. Pedrizzetti, “Three dimensional numerical assessment of the right ventricular flow using 4D echocardiography boundary data,” *European Journal of Mechanics - B/Fluids*, vol. 35, p. 25–30, sept. 2012. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0997754612000234

[88] H. Watanabe, S. Sugiura, H. Kafuku et T. Hisada, “Multiphysics simulation of left ventricular filling dynamics using fluid-structure interaction finite element method,” *Biophysical Journal*, vol. 87, n°. 3, p. 2074–2085, sept. 2004.

[89] S. K. Dahl, “Numerical Simulations of Blood Flow in the Left Side of the Heart,” Doctoral thesis, Norges teknisk-naturvitenskapelige universitet, Fakultet for ingeniørvitenskap og teknologi, Institutt for konstruksjonsteknikk, 2012, accepted : 2014-12-19T12 :00 :07Z ISBN : 9788247135532. [En ligne]. Disponible : https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/236914

[90] S. S. Khalafvand, F. Xu, J. Westenberg, F. Gijsen et S. Kenjeres, “Intraventricular blood flow with a fully dynamic mitral valve model,” *Computers in Biology and Medicine*, vol. 104, p. 197–204, janv. 2019. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0010482518303949

[91] C. J. Carlhäll et A. Bolger, “Passing strange : flow in the failing ventricle,” *Circulation. Heart Failure*, vol. 3, n°. 2, p. 326–331, mars 2010.

[92] A. F. Bolger, E. Heiberg, M. Karlsson, L. Wigström, J. Engvall, A. Sigfridsson, T. Ebbers, J.-P. E. Kvitting, C. J. Carlhäll et B. Wranne, “Transit of blood flow through the human left ventricle mapped by cardiovascular magnetic resonance,” *Journal of Cardiovascular Magnetic Resonance : Official Journal of the Society for Cardiovascular Magnetic Resonance*, vol. 9, n°. 5, p. 741–747, 2007.

[93] J. Eriksson, P. Dyverfeldt, J. Engvall, A. F. Bolger, T. Ebbers et C. J. Carlhäll, “Quantification of presystolic blood flow organization and energetics in the human left ventricle,” *American Journal of Physiology. Heart and Circulatory Physiology*, vol. 300, n°. 6, p. H2135–2141, juin 2011.

[94] J. Eriksson, A. F. Bolger, T. Ebbers et C.-J. Carlhäll, “Four-dimensional blood flow-specific markers of LV dysfunction in dilated cardiomyopathy,” *European Heart Journal. Cardiovascular Imaging*, vol. 14, n°. 5, p. 417–424, mai 2013.

[95] Astropy Collaboration, A. M. Price-Whelan, P. L. Lim, N. Earl, N. Starkman, L. Bradley, D. L. Shupe, A. A. Patil, L. Corrales, C. E. Brasseur, M. Nöthe, A. Donath, E. Tollerud, B. M. Morris, A. Ginsburg, E. Vaher, B. A. Weaver, J. Tocknell, W. Jamieson, M. H. v. Kerkwijk, T. P. Robitaille, B. Merry, M. Bachetti, H. M. Günther, T. L. Aldcroft, J. A. Alvarado-Montes, A. M. Archibald, A. Bódi, S. Bapat, G. Barentsen, J. Bazán, M. Biswas, M. Boquien, D. J. Burke, D. Cara, M. Cara, K. E. Conroy, S. Conseil, M. W. Craig,

R. M. Cross, K. L. Cruz, F. D'Eugenio, N. Dencheva, H. A. R. Devillepoix, J. P. Dietrich, A. D. Eigenbrot, T. Erben, L. Ferreira, D. Foreman-Mackey, R. Fox, N. Freij, S. Garg, R. Geda, L. Glattly, Y. Gondhalekar, K. D. Gordon, D. Grant, P. Greenfield, A. M. Groener, S. Guest, S. Gurovich, R. Handberg, A. Hart, Z. Hatfield-Dodds, D. Homeier, G. Hosseinzadeh, T. Jenness, C. K. Jones, P. Joseph, J. B. Kalmbach, E. Karamehmetoglu, M. Kałuszyński, M. S. P. Kelley, N. Kern, W. E. Kerzendorf, E. W. Koch, S. Kulumani, A. Lee, C. Ly, Z. Ma, C. MacBride, J. M. Maljaars, D. Muna, N. A. Murphy, H. Norman, R. O'Steen, K. A. Oman, C. Pacifici, S. Pascual, J. Pascual-Granado, R. R. Patil, G. I. Perren, T. E. Pickering, T. Rastogi, B. R. Roulston, D. F. Ryan, E. S. Rykoff, J. Sabater, P. Sakurikar, J. Salgado, A. Sanghi, N. Saunders, V. Savchenko, L. Schwardt, M. Seifert-Eckert, A. Y. Shih, A. S. Jain, G. Shukla, J. Sick, C. Simpson, S. Singanamalla, L. P. Singer, J. Singhal, M. Sinha, B. M. Sipőcz, L. R. Spitler, D. Stansby, O. Streicher, J. Šumak, J. D. Swinbank, D. S. Taranu, N. Tewary, G. R. Tremblay, M. d. Val-Borro, S. J. V. Kooten, Z. Vasović, S. Verma, J. V. d. M. Cardoso, P. K. G. Williams, T. J. Wilson, B. Winkel, W. M. Wood-Vasey, R. Xue, P. Yoachim, C. ZHANG et A. Zonca, "The Astropy Project : Sustaining and Growing a Community-oriented Open-source Project and the Latest Major Release (v5.0) of the Core Package," *The Astrophysical Journal*, vol. 935, nº. 2, 2022.

ANNEXE A MODÈLES NUMÉRIQUES BASÉS SUR L'ÉCHOCARDIOGRAPHIE

Le **Tableau A.1** présente les travaux répertoriés utilisant l'EchoCG dans leur modélisation du VG. Les aspects principaux des modèles y sont présentés.

Auteur	**Mode EchoCG**	**Géométrie**			**Caractéristique du sang**	**Méthode numérique**	**Phase du cycle simulé**	**Solveur**	**Cas**	**Méthode de Validation**
		Reconstruction du myocarde	Reconstruction du flux sanguin	Type						
McQueen al. (2000) [62]	3D	Structure géodésique fibres myocardiques	Vitesse uniforme section VM	3D	Newtonien, incompressible	IBM	Complet	X	X	X
Baccani et al. (2005) [78]	2D	Ellispe de révolution tronquée VM cylindre diamètre fixe	Vitesse uniforme ou parabolique section VM	2D	Newtonien, incompressible	FSI	Diastole	X	X	Doppler
Cheng et al. (2005) [12]	2D	Ellispe de révolution tronquée à épaisseur variable	Vitesse uniforme section VM variation temporelle empirique	3D	Newtonien, laminaire	FSI	Diastole	ADINA-FSI	X	X
Dahl et al. (2010) [71, 89]	2D	Speckle tracking (EchoPAC)	X	2D	Newtonien, laminaire, incompressible	FSI[1] + GPM	Diastole	Ansys Fluent	Sain	Qualitatif
de Vecchi et al. (2014) [20]	B-mode, Color Doppler	Segmentation manuelle (MITK) des contours (Doppler) et des valves (B-mode).	Extrait du Color Doppler	3D	X	GPM[2]	Complet	Cheart	Pathologique	X
Mangual et al. (2014) [11]	2D, 3D	Feature tracking (4DLVA)	X	3D	Newtonien, laminaire	IBM[3]	Complet	Cheart	Sain	Speckle tracking
Bavo al. (2017) [69]	TEE[4] 3D	Segmentation semi-automatique (ITK-Snap). NCS and BS time interpolation	Vitesse uniforme section VM	3D	Newtonien, incompressible	GPM	Diastole	Ansys Fluent	Pathologique	X
Govindarajan al. (2018) [15]	TEE 3D	X	Vitesse Profil Constant variation volume mesuré	3D	Newtonien, incompressible	FSI	Diastole	FEAP	Sain	Doppler

TABLEAU A.1 Répertoire d'études utilisant l'EchoCG pour la modélisation numérique de l'écoulement dans le VG. (Adapté de Dubois 2023 [1])

1. *FSI = Fluid-Structure Interaction*
2. *GPM = Geometry-Prescribed Method*
3. *IBM = Immersed Boundary Method*
4. *TEE = Transesophageal Echocardiography*

ANNEXE B COMPLÉMENTS - CONCEPTION DU MODÈLE NUMÉRIQUE

Cette annexe a pour objectif de rentrer dans les détails de l'implémentation du modèle numérique.

Afin de créer le domaine géométrique de la résolution MFN, il faut lire, traiter et reconstruire les courbes formant le contour du VG à partir de la segmentation du VG. Cette étape est réalisée avec le **code 5**. La segmentation est divisée en portions, voir **Figure 3.4**, `['MV','AV','LVW_in','LVW_out','MVadd_in','MVadd_out','AVadd_in','AVadd_out']` chacune étant reliée à un fichier *.csv* des coordonnées de ces points au cours du temps. Des portions intermédiaires telles que `['LV_in_LF','LV_in_red','LV_out_LF','LV_out_red','AV_mod','LF_L','LF_R']` sont créées. Elles correspondent respectivement à la portion allant de l'extrémité antérieure de la VM à l'extrémité postérieure de la VA en passant par la VMA ; la portion allant de l'extrémité antérieure de la VMA à l'extrémité postérieure de la VA ; à la portion allant de l'extrémité postérieure de la VM à l'extrémité antérieure de la VA en passant par la VMP ; la portion allant de l'extrémité postérieure de la VMP à l'extrémité antérieure de la VA ; à la portion représentant la VA sans décalage géométrique comme dans le modèle précédent [1] ; la VMA ; la VMP.

Afin de créer ces courbes de contour, il faut procéder à la reconstruction du modèle paramétrique de valve. L'implémentation (**Code 8**) disponible dans le répertoire de fonction en **Annexe D** se divise en une première partie où les paramètres géométriques sont calculés puis une seconde partie où la géométrie est reconstruire. Le calcul des paramètres est défini à la **Sous-Section 3.1.3**. Concernant la reconstruction géométrique elle prend en entrée un fichier précisant la valeur de AMA et AMP pour chaque pas de temps du cycle. Le calcul de ces lois temporelles vu à la **Section 3.2** fait l'objet des routines (**Codes 9** et **10**). La reconstruction de la VM est définie dans la fonction `def leaflets(step,angle_l,angle_r)`, elle concerne d'abord la valvule antérieure, puis postérieure. Un point est extrait de la géométrie à une distance égale à l'épaisseur valvulaire du point à l'extrémité annulaire. Le point milieu est défini comme le milieu de ces derniers. Le point milieu est relié à l'extrémité de la valve via une courbe de Bézier créant une courbe milieu. À partir de cette courbe milieu, des points de contrôle sont uniformément répartis à une distance égale à la moitié de l'épaisseur valvulaire. Enfin, trois splines sont reconstruites, l'extrémité de la valvule et les raccordements à la paroi afin d'assurer une continuité du raccord.

Le fichier *curve_init.txt* est ensuite écrit en accord avec le formatage disponible dans le

Fluent User's Guide 2024R1. La routine (**Code 1**) est divisée en l'écriture locale d'une série d'opérations qui est ensuite exécutée dans *SpaceClaim* au sein d'une instance API. Elle comprend l'importation des courbes issues de *curve_init.txt*, la création des sélections nommées, le fait de combler l'espace interne et l'enregistrement au format natif *.scdoc*.

Une fois le domaine créé, le maillage est généré en utilisant une routine (**Code 2**) faisant intervenir *Ansys Meshing*. Le code est divisé en deux étapes, la première étape consiste à créer le maillage, puis à l'exporter au format *.msh*. Cette étape est réalisée directement en utilisant l'éco-système PyAnsys. L'exportation nécessite l'écriture et l'exécution d'un script.

Afin de déplacer le maillage, il faut récupérer les identifiants des nœuds (nID) de la frontière, or ceux-ci sont initialisés lors de l'étape de résolution avec *Fluent*. Il suffit donc de lancer une simulation d'un seul pas de temps avec une *UDF* qui, au début du pas de temps, écrit dans un fichier séparé le nID et les coordonnées de chacun des nœuds. Le **Code 3** montre l'écriture de l'*UDF*, ouvre une session, compile l'*UDF* et lance la simulation.

L'information exportée est ensuite importée et triée. Les points du maillage et de la segmentation ne coïncidant pas exactement, une association est faite sur la valeur du contour s. Cette valeur est unique pour chaque point et permet de faire une bijection entre le contour du maillage et le contour de la segmentation. Ensuite, via l'hypothèse que cette valeur reste constante au cours du temps, il est possible de calculer la position au cours du temps de chaque point du maillage initial. Connaissant la position des points au cours du cycle cardiaque, nous pouvons artificiellement, par interpolation temporelle, réduire ou augmenter le pas de temps. Une fois satisfait, l'information est écrite sous la forme d'un fichier *.dat* pour chaque sélection nommée n'étant pas fixée dans l'espace. L'*UDF* permettant de lire le fichier *.dat* et aussi de déplacer la frontière à chaque pas de temps est également écrite à ce moment là de la mise en place. Cette sub-routine (**Code 6**) fait appel à des fonctions utilitaires écrites dans un fichier (**Code 7**) séparé et à importer.

Cette description en espace et en temps de la frontière est de nouveau importée et sert de base au calcul des conditions limites du modèle. Il est possible de détecter les transitions de la systole à la diastole et de la diastole à la systole respectivement par un minimum et un maximum de l'aire $A(t)$ du VG. La variation d'aire du VG, ΔA, est également calculée à chaque pas de temps. Elle permet de calculer l'amplitude du profil de vitesse constant imposé durant la diastole qui respecte la conservation de la masse. Le Tableau Transitoire est ensuite écrit en respectant un formatage propre à *Fluent*. Au début de la simulation, le fluide est initialisé avec une vitesse nulle et une pression égale à la pression de gauge (i.e. nulle). Ensuite, chaque cycle cardiaque est divisé en deux, d'abord la systole puis la diastole. La simulation résout l'écoulement systolique, puis résout l'écoulement diastolique. À la transition,

les anciennes conditions aux limites sont modifiées, et ainsi de suite à chaque changement de phase cardiaque. Le détail des commandes PyAnsys sont annotées et présentées dans le **Code 4**.

ANNEXE C COMPLÉMENTS - RECONSTRUCTION DE LA GÉOMÉTRIE DU VG À PARTIR DES IMAGES ECHOCG MODE B

Contrairement aux **Chapitre 3** et **4** où les images EchoCG sont extraites de la cohorte Pétale, le **Chapitre 5** fait appel à des données externes sur lesquelles ont été réalisées des reconstructions de l'écoulement avec la méthode iVFM. Dans le cadre de la cohorte Pétale, la reconstruction de la géométrie du VG avait été réalisée lors d'une étude précédente [1]. Dans le cas de la comparaison avec le cas iVFM, la partie inférieure de la cavité ventriculaire est extraite par le logiciel EchoPAC, mais la partie supérieure (i.e. la base) avec la délimitation de la VA et la VM doit être réalisée via une routine créée pour l'occasion. Cette annexe rend compte de la méthodologie utilisée pour reconstruire la géométrie du VG dans le cas iVFM.

Suite à l'examen médical, le clinicien a réalisé une segmentation de la cavité du VG via le logiciel EchoPAC. Cette segmentation discrète en espace et en temps est approximée au moyen de splines d'ordre 5 et 4 respectivement pour les paramètres $t \in [0, T]$ le temps et $s \in [0, 1]$ le contour. Le résultat est présenté aux **Figures C.1** et **C.2**.

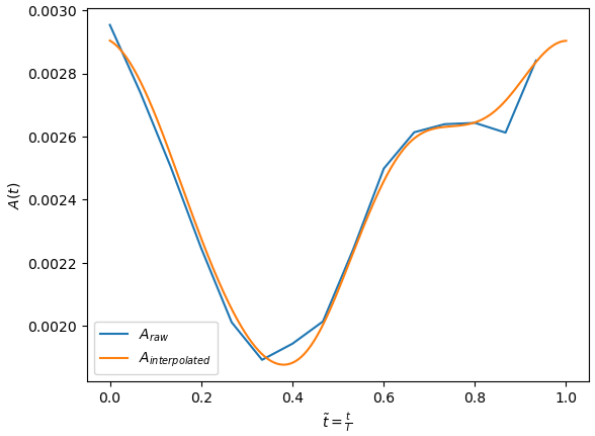


FIGURE C.1 Résultat de l'interpolation spatio-temporelle de la segmentation issue de Echo-PAC - $A(t)$

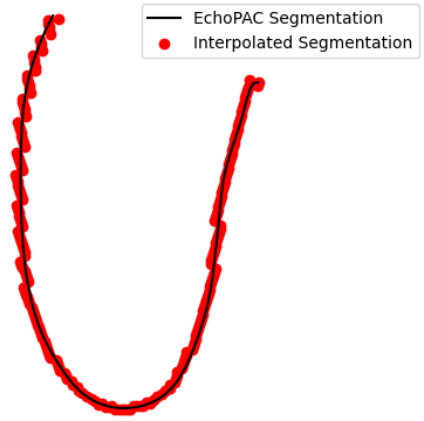


FIGURE C.2 Résultat de l'interpolation spatio-temporelle de la segmentation issue de Echo-PAC - $X_s(t)$

La suite est dédiée à la détection de la VA et de la VM et à la reconstruction de la partie supérieure de la cavité. Le contour de la cavité est d'abord produit grâce à une série d'opérations de traitement d'image (voir **Figure C.3**), puis les positions de la VA et de la VM sont détectées puis approximées pour reconstruire la partie supérieure du VG (voir **Figure C.4**). L'ensemble de ce processus se fait sur l'EchoCG mode B du VTS.

Ce paragraphe détaille les étapes (A à H) présentes dans le schéma explicatif du traitement d'image visible à la **Figure C.3**.

- A : Une première phase de filtrage est appliquée à l'image brute pour augmenter le contraste de l'image, elle fait intervenir un filtre *Butterworth* (`skimage.filters.butterworth()`) passe-bas d'ordre 2 avec une fréquence de seuil de 0.5 et un filtre *Sigmoïd Correction* (`skimage.exposure.adjust_sigmoid()`) de gain 10 et de fréquence de seuil de 0.5.
- B et C : Une nouvelle itération de filtrage fait intervenir un filtre *Butterworth* passe-bas d'ordre 2 avec une fréquence de seuil de 0.5, puis respectivement un filtre *Sigmoïd Correction* de gain 10 et de fréquence de seuil de 0.5 pour B et 0.15 pour C. La paroi du ventricule est rajoutée à l'image pour clore la cavité.
- D : L'image est utilisée pour faire une première segmentation avec une technique de *Flood* (`skimage.morphology.flood()`) appliquée à la zone de haute intensité. La segmentation est ensuite modifiée avec des transformations morphologiques d'ouverture et de fermeture (`skimage.morphology.binary_opening()`,`skimage.morphology.binary_closing()`) de sorte à relier la VM et la VA, mais aussi restreindre la tendance à la segmentation à rentrer dans le VG le long de la face antérieure. Cette étape sert de première estimation.
- E : L'image est modifiée en utilisant un filtre d'inversion du gradient d'intensité issue d'un lissage gaussien (`skimage.segmentation.inverse_gaussian_gradient()`). Cette étape accentue les gradients d'intensité et par conséquent la paroi du VG mais aussi les feuillets de la VA et la VM.
- F et G : La première estimation issue de D et l'image inversée de E sont utilisées dans une méthode plus sophistiquée de segmentation basé appelée *Morphological Geodesic Active Contours* (`skimage.segmentation.morphological_geodesic_active_contour()`). Le squelette de cette segmentation ((`skimage.morphology.skeletonize()`) est produit dans l'objectif d'extraire une première approximation de contour du VG.
- H : Le contour issue de F et G présente des branches annexes qui doivent être retirées. Cela est fait en deux temps, d'abord avec le package Python Astropy [95], puis en appliquant un seuil sur la longueur des branches. Ce contour est coupé au niveau de extrémités de la segmentation de la partie inférieure du VG.

Une fois ce contour brut de la partie supérieur produit, il faut détecter la position de la VA et la VM, puis produire une géométrie simplifiée pour former la partie supérieure de la cavité. Le schéma explicatif de la **Figure C.4** illustre les étapes (A à D) pour obtenir la géométrie finale.

— A : Représente le contour brut de la partie supérieure du VG.
— B : À partir de cette portion de contour, un vecteur normal n et un vecteur tangent t au segment qui relie ces deux extrémités sont calculés. La composante normale X_n de la position des points du contour projetée sur le segment s_t est utilisée pour détecter la position de la VA (en rouge sur la **Figure C.4**) et la VM (en vert sur la **Figure C.4**). La VA correspond à la portion répondant au critère $|X_n(s_t) - X_n^{max}| \leq 0.15.|X_n^{max} - X_n^{min}|$. Le bord antérieur de la VM est déterminé par un maximum local. La portion qui relie ce bord à la VA constitue la VM.
— C : Afin d'estimer plus tard les paramètres β_t et β_n, il faut détecter le point de fermeture de la VM, autrement dit l'endroit où les feuillets mitraux se rejoignent. Une première approximation est faite avec le critère $X_n(s_t) - X_n^{min}$. L'intensité $I(X)$ de l'image le long de la portion projetée sur le segment fait apparaître les variations locales du contour. L'hypothèse est que le point de fermeture se trouve au niveau d'un minimum local de l'intensité dans la direction de l'extrémité antérieure de la VM. En reliant les bords de la VM à ce point de fermeture, il est possible d'en déduire les paramètres β_t et β_n.
— D : L'étape précédente a permis de mettre en évidence les extrémités de la VA et de la VM. Une jonction est créée entre ces deux structures au moyen d'un profil parabolique de 3 mm dans la direction de l'apex ($+n$). La position de la VM est artificiellement décalée vers l'atrium ($-n$) de 3 mm pour faciliter la simulation plus tard. La géométrie finale est formée de segments rectilignes et d'une jonction parabolique.

Finalement, la segmentation de la partie inférieure est interpolée à nouveau en contraignant la position de ces extrémités à coïncider avec l'extrémité antérieure de la VM et l'extrémité postérieure de la VA. Les géométries de la partie supérieure sont discrétisées. Chaque portion de la géométrie finale est exportée séparément au format *.csv* prête à être lue par la routine de simulation. Pour obtenir le détail des paramètres des fonctions Python utilisées précédemment veuillez-vous référer au **Code 12** disponible à l'**Annexe D**.

Il est à noter que cette reconstruction de la géométrie fait intervenir de nombreux paramètres empiriques qui peuvent limiter la robustesse de la méthode. De nouveaux outils faisant intervenir le *machine learning* ont prouvé leur efficacité pour la segmentation de cavité ventriculaire. Il est donc envisageable d'utiliser ce genre de technique pour segmenter la base en plus de la cavité inférieure du VG.

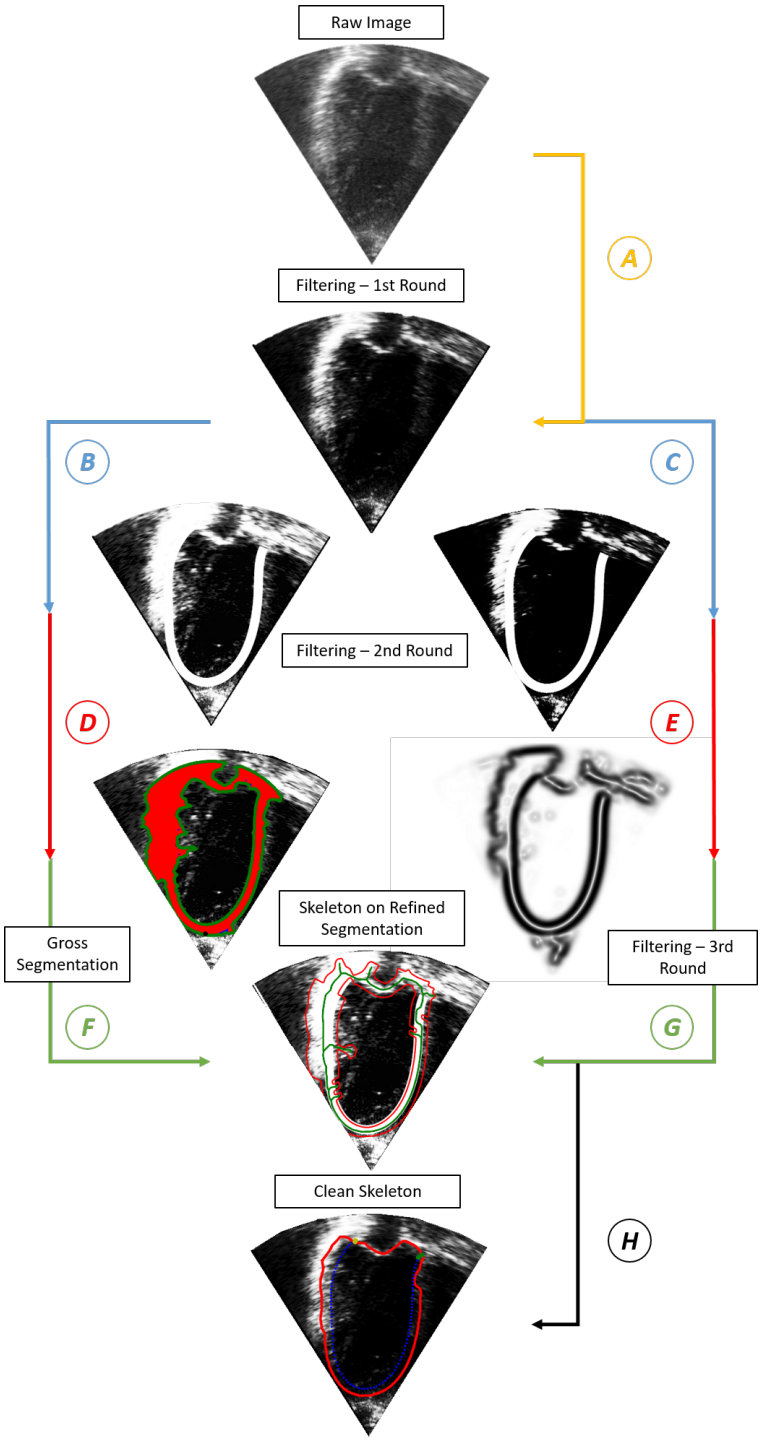


FIGURE C.3 Schéma explicatif du processus de traitement d'image pour obtenir le contour du VG.

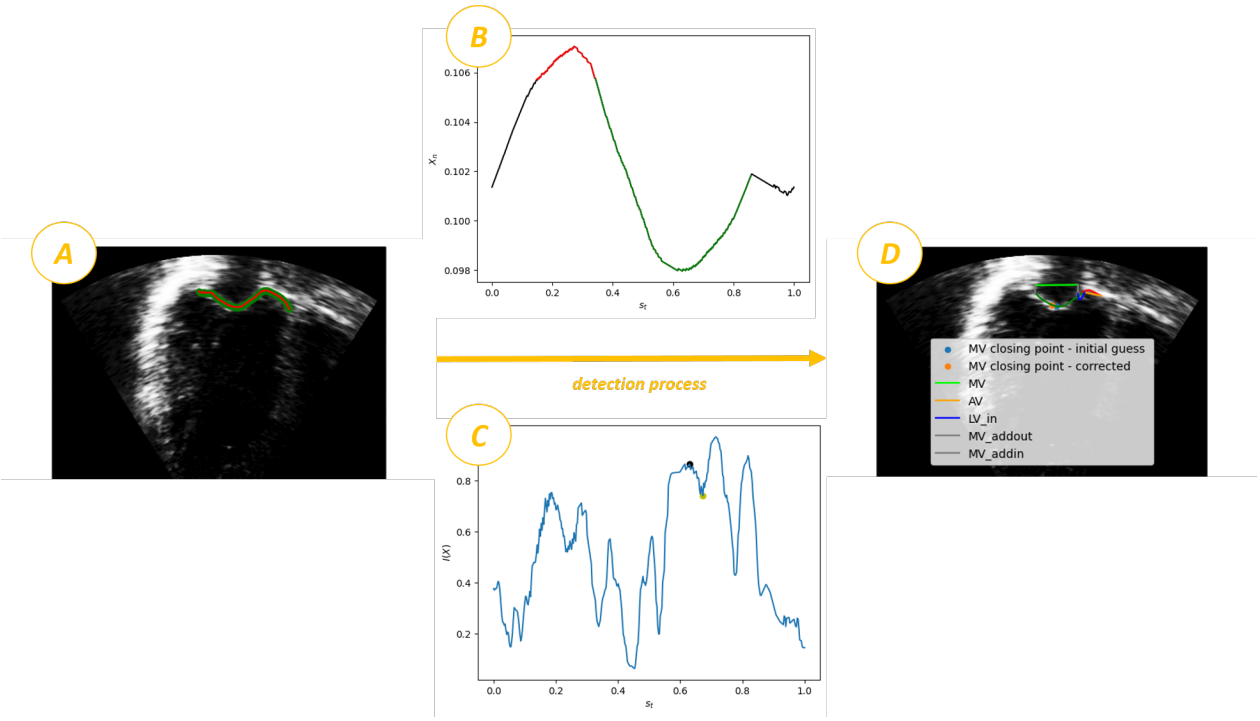


FIGURE C.4 Schéma explicatif de la détection et de la reconstruction de la VA et la VM.

ANNEXE D RÉPERTOIRE - CODE

Cette annexe a pour vocation de répertorier l'ensemble des codes sources nécessaires pour la réalisation du modèle numérique.

Liste des Codes Sources

```
from ansys.geometry.core import launch_modeler

curvesPath = dataFolder + "geom\\curve_init.txt"
geoPath = dataFolder + "geom\\geom_SCreconstruction_LL.scdocx"

script =  f"""# Python Script, API Version = V24
# Author : Aurélien Lacourt, M.Sc.A, Polytechnique Montreal

# Configurer le plan d esquisse
sectionPlane = Plane.PlaneXY
result = ViewHelper.SetSketchPlane(sectionPlane, None)
# EndBlock

# Inserer depuis un fichier
importOptions = ImportOptions.Create()
DocumentInsert.Execute(r'{curvesPath}', importOptions, GetMaps("59b224e7"))
# EndBlock

# Solidifier l esquisse
mode = InteractionMode.Solid
result = ViewHelper.SetViewMode(mode, None)
# EndBlock


selection = Selection.Create([
        GetRootPart().Curves[0],
        GetRootPart().Curves[1],
        GetRootPart().Curves[2],
        GetRootPart().Curves[3],
        GetRootPart().Curves[4],
        GetRootPart().Curves[5],
        GetRootPart().Curves[6],
    GetRootPart().Curves[7],
])
secondarySelection = Selection.Empty()
options = FillOptions()
result = Fill.Execute(selection, secondarySelection, options, FillMode.ThreeD, None)

# Selection nommee "LF_R"
primarySelection = EdgeSelection.Create(GetRootPart().Bodies[0].Edges[1])
secondarySelection = Selection.Empty()
result = NamedSelection.Create(primarySelection, secondarySelection)
result = NamedSelection.Rename("Groupe1", "LF_R")

# Selection nommee "LVW_OUT_RED"
primarySelection = EdgeSelection.Create(GetRootPart().Bodies[0].Edges[2])
secondarySelection = Selection.Empty()
result = NamedSelection.Create(primarySelection, secondarySelection)
result = NamedSelection.Rename("Groupe1", "LVW_OUT_RED")

# Selection nommee "LVW_OUT_LF"
primarySelection = EdgeSelection.Create(GetRootPart().Bodies[0].Edges[2],GetRootPart().Bodies[0].Edges[1])
secondarySelection = Selection.Empty()
```

```
result = NamedSelection.Create(primarySelection, secondarySelection)
result = NamedSelection.Rename("Groupe1", "LVW_OUT_LF")

# Selection nommee "LF_L"
primarySelection = EdgeSelection.Create(GetRootPart().Bodies[0].Edges[5])
secondarySelection = Selection.Empty()
result = NamedSelection.Create(primarySelection, secondarySelection)
result = NamedSelection.Rename("Groupe1", "LF_L")

# Selection nommee "LVW_IN_RED"
primarySelection = EdgeSelection.Create(GetRootPart().Bodies[0].Edges[4])
secondarySelection = Selection.Empty()
result = NamedSelection.Create(primarySelection, secondarySelection)
result = NamedSelection.Rename("Groupe1", "LVW_IN_RED")

# Selection nommee "LVW_IN_LF"
primarySelection = EdgeSelection.Create(GetRootPart().Bodies[0].Edges[5],GetRootPart().Bodies[0].Edges[4])
secondarySelection = Selection.Empty()
result = NamedSelection.Create(primarySelection, secondarySelection)
result = NamedSelection.Rename("Groupe1", "LVW_IN_LF")

# Selection nommee "MVadd_OUT"
primarySelection = EdgeSelection.Create(GetRootPart().Bodies[0].Edges[0])
secondarySelection = Selection.Empty()
result = NamedSelection.Create(primarySelection, secondarySelection)
result = NamedSelection.Rename("Groupe1", "MVadd_OUT")

# Selection nommee "MVadd_IN"
primarySelection = EdgeSelection.Create(GetRootPart().Bodies[0].Edges[6])
secondarySelection = Selection.Empty()
result = NamedSelection.Create(primarySelection, secondarySelection)
result = NamedSelection.Rename("Groupe1", "MVadd_IN")

# Selection nommee "AV"
primarySelection = EdgeSelection.Create(GetRootPart().Bodies[0].Edges[3])#4
secondarySelection = Selection.Empty()
result = NamedSelection.Create(primarySelection, secondarySelection)
result = NamedSelection.Rename("Groupe1", "AV")

# Selection nommee "MV"
primarySelection = EdgeSelection.Create(GetRootPart().Bodies[0].Edges[7])
secondarySelection = Selection.Empty()
result = NamedSelection.Create(primarySelection, secondarySelection)
result = NamedSelection.Rename("Groupe1", "MV")

# Selection nommee "LV_INTERIOR"
primarySelection = FaceSelection.Create(GetRootPart().Bodies[0].Faces[0])
secondarySelection = Selection.Empty()
result = NamedSelection.Create(primarySelection, secondarySelection)
result = NamedSelection.Rename("Groupe1", "LV_INTERIOR")

# Selection nommee "LV_INTERIOR_OBJ"
primarySelection = BodySelection.Create(GetRootPart().Bodies[0])
secondarySelection = Selection.Empty()
result = NamedSelection.Create(primarySelection, secondarySelection)
```

```
result = NamedSelection.Rename("Groupe1", "LV_INTERIOR_OBJ")

# Enregistrer le fichier
options = ExportOptions.Create()
DocumentSave.Execute(r'{geoPath}', options)
# EndBlock

"""

# Start a modeler session
modeler = launch_modeler()
print(modeler)

fname = "Y:\\Recherche\\DEV\\PyAnsys\\script_geom_reconstruct.py"
with open(fname, 'w') as f:
    f.write(script)

output = modeler.run_discovery_script_file(file_path=fname)

os.remove(fname)

modeler.close()
```

Listing 1 PyAnsys : Création Domaine Géométrique

```
from ansys.mechanical.core import launch_mechanical, App, find_mechanical, global_variables

meshPath_temp = dataFolder + "mesh\\geom_mesh.mechdb"
meshPath_temp_folder = dataFolder + "mesh\\geom_mesh_Mech_Files"
meshPath = dataFolder + "mesh\\geom_mesh.msh"

# to use the first we launch mechanical
app = App()
print(app)
#globals().update(global_variables(app))
app.update_globals(globals())

geomImport = Model.GeometryImportGroup.AddGeometryImport()
geomImport_format = geomImport.GetFormat()
geomImport_preferences = geomImport.GetPreferences()

geomImport_preferences.ProcessSolids = True
geomImport_preferences.ProcessSurfaces = True
geomImport_preferences.ProcessLines = True

geomImport_preferences.ProcessNamedSelections = True
geomImport_preferences.NamedSelectionKey = ""

geomImport.Import(geoPath, geomImport_format, geomImport_preferences)

#### meshing parameters
```

```
27
28  cellsize = 1 #mm
29
30  inflationMaximumLayers = 1
31  inflationGrowthRate = 2
32
33  NbOfRefinements = 1
34
35  #meshing instructions
36
37  sizingtype  = Ansys.Mechanical.DataModel.Enums.SizingType
38  sizingbehavior = Ansys.Mechanical.DataModel.Enums.SizingBehavior
39  methodtype = Ansys.Mechanical.DataModel.Enums.MethodType
40
41  mesh_1 = Model.Mesh
42
43
44  # Domain mesh sizing
45  sizing_1 = mesh_1.AddSizing()
46  sizing_1.Location = DataModel.GetObjectsByName("LV_INTERIOR")[0]
47  sizing_1.Type = sizingtype.ElementSize
48  sizing_1.ElementSize = Quantity(cellsize,"mm")
49
50  # Methode triangles
51  automatic_method = mesh_1.AddAutomaticMethod()
52  automatic_method.Location = DataModel.GetObjectsByName("LV_INTERIOR_OBJ")[0]
53  automatic_method.Method = methodtype.AllTriAllTet
54
55  # MV edge sizing
56  sizing_2 = mesh_1.AddSizing()
57  sizing_2.Location = DataModel.GetObjectsByName("MV")[0]
58  sizing_2.Type = sizingtype.ElementSize
59  sizing_2.ElementSize = Quantity(cellsize,"mm")
60  sizing_2.Behavior = sizingbehavior.Hard
61
62  # MVadd_IN edge sizing
63  sizing_3 = mesh_1.AddSizing()
64  sizing_3.Location = DataModel.GetObjectsByName("MVadd_IN")[0]
65  sizing_3.Type = sizingtype.ElementSize
66  sizing_3.ElementSize = Quantity(cellsize,"mm")
67  sizing_3.Behavior = sizingbehavior.Hard
68
69  # AVadd_OUT edge sizing
70  sizing_4 = mesh_1.AddSizing()
71  sizing_4.Location = DataModel.GetObjectsByName("MVadd_OUT")[0]
72  sizing_4.Type = sizingtype.ElementSize
73  sizing_4.ElementSize = Quantity(cellsize,"mm")
74  sizing_4.Behavior = sizingbehavior.Hard
75
76
77  # AV edge sizing
78  sizing_3 = mesh_1.AddSizing()
79  sizing_3.Location = DataModel.GetObjectsByName("AV")[0]
80  sizing_3.Type = sizingtype.ElementSize
81  sizing_3.ElementSize = Quantity(cellsize,"mm")
```

```
sizing_3.Behavior = sizingbehavior.Hard



## LVW_OUT edge sizing
sizing_5 = mesh_1.AddSizing()
sizing_5.Location = DataModel.GetObjectsByName("LVW_OUT_LF")[0]
sizing_5.Type = sizingtype.ElementSize
sizing_5.ElementSize = Quantity(cellsize/2,"mm")
sizing_5.Behavior = sizingbehavior.Hard

## LVW_IN edge sizing
sizing_5 = mesh_1.AddSizing()
sizing_5.Location = DataModel.GetObjectsByName("LVW_IN_LF")[0]
sizing_5.Type = sizingtype.ElementSize
sizing_5.ElementSize = Quantity(cellsize/2,"mm")
sizing_5.Behavior = sizingbehavior.Hard


# Generation of mesh
mesh_1.GenerateMesh()

app.save(meshPath_temp)

app.close()

# saving to.msh

#convert mesh object to .msh

mechanical = launch_mechanical(batch=True, cleanup_on_exit=False)

command =f"""DataModel.Project.Open("{meshPath_temp}")

NS = ['LF_R','LVW_OUT_RED','LF_L','LVW_IN_RED']


for ns in ExtAPI.DataModel.GetObjectsByType(DataModelObjectCategory.NamedSelection):
    if ns.Name == NS[0] :
        ns.Delete()

for ns in ExtAPI.DataModel.GetObjectsByType(DataModelObjectCategory.NamedSelection):
    if ns.Name == NS[1] :
        ns.Delete()

for ns in ExtAPI.DataModel.GetObjectsByType(DataModelObjectCategory.NamedSelection):
    if ns.Name == NS[2] :
        ns.Delete()

for ns in ExtAPI.DataModel.GetObjectsByType(DataModelObjectCategory.NamedSelection):
    if ns.Name == NS[3] :
        ns.Delete()

"""

```

```
mechanical.run_python_script(command)

save_command = f'ExtAPI.DataModel.Project.Model.Mesh.InternalObject.WriteFluentInputFile("{meshPath}")'
mechanical.run_python_script(save_command)

mechanical.exit(force=True)

os.remove(meshPath_temp)
shutil.rmtree(meshPath_temp_folder)
```

Listing 2 PyAnsys : Maillage du domaine

```
import ansys.fluent.core as pyfluent
from ansys.fluent.core import examples
from ansys.fluent.visualization.pyvista import Graphics

get_nodesPath = dataFolder + "udf\\GET_NODES_ID_write.c"

nodesPath_LV_IN = dataFolder + "fluent\\node_positions_LV_IN.txt"
nodesPath_LV_IN = nodesPath_LV_IN.replace("\\","\\\\")
nodesPath_LV_OUT = dataFolder + "fluent\\node_positions_LV_OUT.txt"
nodesPath_LV_OUT = nodesPath_LV_OUT.replace("\\","\\\\")

motionPath_LV_IN = dataFolder + "fluent\\LV_IN_motion.dat"
motionPath_LV_OUT = dataFolder + "fluent\\LV_OUT_motion.dat"

mesh_motionPath = dataFolder + "fluent\\LV_MOTION.c"
mesh_motionPath_LV_IN = dataFolder + "udf\\LV_MOTION_LV_IN.c"
mesh_motionPath_LV_OUT = dataFolder + "udf\\LV_MOTION_LV_OUT.c"

motionPath_LV_IN = motionPath_LV_IN.replace("\\","\\\\")
motionPath_LV_OUT = motionPath_LV_OUT.replace("\\","\\\\")

outmassflowratePath =  dataFolder + 'fluent\\massflowrate.txt'
velocityMagnPath = dataFolder + 'fluent\\velocity_magn.txt'

script =  f"""#include "udf.h"

// This function is to retreive the nodes position in FLUENT in order to assign node motion and obtain a     ...
↪  dynamic geometry of the LV in 2D

DEFINE_GRID_MOTION(GET_NODES_ID_LV_IN, domain, dt, time, dtime)
{{
        Thread *tf = DT_THREAD(dt);
        face_t f;
        Node *v;
        int n;
        // Open the file in append mode so data from each time step is saved
    FILE *fp = fopen("{nodesPath_LV_IN}", "a");
    if (fp == NULL)
    {{
```

```
39          Message("Error opening file!\\n");
40          return;
41      }}
42
43          SET_DEFORMING_THREAD_FLAG(THREAD_T0(tf));
44          begin_f_loop(f, tf)
45          {{
46                  f_node_loop(f, tf, n)
47                  {{
48                          v = F_NODE(f, tf, n);
49                          if (NODE_POS_NEED_UPDATE(v))
50                          {{
51                                  fprintf(fp, "%17.16f, %17.16f, %llu \\n", NODE_X(v), NODE_Y(v),              ...
                                    ↪  NODE_ID(v));
52                                  NODE_POS_UPDATED(v);
53                          }}
54
55                  }}
56          }}
57          end_f_loop(f, tf);
58          fclose(fp);
59  }}
60
61  DEFINE_GRID_MOTION(GET_NODES_ID_LV_OUT, domain, dt, time, dtime)
62  {{
63          Thread *tf = DT_THREAD(dt);
64          face_t f;
65          Node *v;
66          int n;
67          // Open the file in append mode so data from each time step is saved
68      FILE *fp = fopen("{nodesPath_LV_OUT}", "a");
69      if (fp == NULL)
70      {{
71          Message("Error opening file!\\n");
72          return;
73      }}
74
75          SET_DEFORMING_THREAD_FLAG(THREAD_T0(tf));
76          begin_f_loop(f, tf)
77          {{
78                  f_node_loop(f, tf, n)
79                  {{
80                          v = F_NODE(f, tf, n);
81                          if (NODE_POS_NEED_UPDATE(v))
82                          {{
83                                  fprintf(fp, "%17.16f, %17.16f, %llu \\n", NODE_X(v), NODE_Y(v),              ...
                                    ↪  NODE_ID(v));
84                                  NODE_POS_UPDATED(v);
85                          }}
86
87                  }}
88          }}
89          end_f_loop(f, tf);
90          fclose(fp);
91  }}
```

```
"""

with open(get_nodesPath, 'w') as f:
    f.write(script)

shutil.copy2(get_nodesPath, workingFolder)


#find nodes ID for dynamic mesh UDF


solver = pyfluent.launch_fluent(
    precision="double",
    processor_count=1,
    mode="solver",
    dimension = pyfluent.Dimension.TWO,
    ui_mode= "gui",
    cwd=workingFolder,
)

solver.file.read_mesh(file_name=meshPath)

solver.setup.general.solver.time = "unsteady-1st-order"
solver.setup.models.viscous.model = "laminar"
solver.tui.define.boundary_conditions.zone_type('lv_interior', 'fluid')

solver.tui.define.user_defined.compiled_functions('compile', '"libudf"', 'yes',  '"GET_NODES_ID_write.c"',   ...
↪  '"', '"')
solver.tui.define.user_defined.compiled_functions('load', '"libudf"')
solver.tui.define.dynamic_mesh.dynamic_mesh('yes', 'no', 'no', 'no', 'no')
solver.tui.define.dynamic_mesh.zones.create('lvw_out_lf', 'user-defined', '"GET_NODES_ID_LV_OUT::libudf"',   ...
↪  'no', 'no', '4', 'constant', '0.', 'no', 'no', 'no')
solver.tui.define.dynamic_mesh.zones.create('lvw_in_lf', 'user-defined', '"GET_NODES_ID_LV_IN::libudf"',     ...
↪  'no', 'no', '4', 'constant', '0.', 'no', 'no', 'no')

solver.solution.initialization.initialization_type = "standard"
solver.solution.initialization.standard_initialize()
solver.solution.run_calculation.dual_time_iterate(time_step_count = 1, max_iter_per_step = 10)

solver.exit()
```

Listing 3 PyAnsys : Récupérer les identifiants

```
# compute indices of diastolic start

lv_in, lv_out = load_points(motionPath_LV_IN), load_points(motionPath_LV_OUT)

curve_mesh = np.concatenate((lv_in[:,::-1,:], lv_out), axis=1)

area = np.zeros(nb_time_step_extended)
for _ in range(nb_time_step_extended):
```

```
    pgon = Polygon(curve_mesh[_,:,:])
    area[_] = pgon.area


local_extreme_max = np.argmax(area)
local_extreme_min = np.argmin(area)


area_diff = np.zeros_like(area)
area_diff = np.diff(area,axis=0, n=1, append=area[0])
area_diff = area_diff/dt

mask_flux = area_diff >= 0

diastolic_flux = area_diff.copy()
diastolic_flux[~mask_flux] = 0
systolic_flux = area_diff.copy()
systolic_flux[mask_flux] = 0


i_dia = local_extreme_min
print(i_dia)
t_dia = t_extended[i_dia]
i_T = local_extreme_max + nb_time_step
print(i_T)
t_T = t_extended[i_T]


# UDF massflowrate profile

mfr_v2 = systolic_flux*depth*blood_dens

df = pd.DataFrame({
    'time': t_extended.flatten(),
    'massflowrate': mfr_v2.flatten()
})


# UDF velocity magnitude profile


inlet_v2 = diastolic_flux*depth #m3/s
U = inlet_v2/D_MV


# Combine into a DataFrame
df = pd.DataFrame({
    'time': t_extended.flatten(),
    'flowrate': U.flatten()
})

# Save to a text file with a header

with open(velocityMagnPath, 'w') as f:
    f.write(f'velocity_magn_profile\t2\t{nb_time_step_extended}\t0\n')
```

```
64      f.write('time\tvelocity_magn\n')
65      df.to_csv(f, sep='\t', index=False, header=False, float_format='{:.12f}'.format, lineterminator='\n')
66
67
68  #compute gap value
69
70  GAP = []
71  for _ in range(lv_in.shape[0]):
72      gap = np.min(cdist(lv_in[_,:,:], lv_out[_,:,:]))
73      GAP.append(gap)
74  GAP = np.array(GAP)
75
76  gap_value = math.ceil(np.max(GAP[:i_dia+1]) * 1e5) / 1e5
77  print(gap_value)
78
79  # simulation criterion
80
81  absolute_criteria_residuals = 1e-05
82  max_iter_per_step_value = 100
83
84  fcn = 100
85  momentum_rf = 0.75
86  pressure_rf = 0.5
87
88  #remesh parameters
89  min_len = 0.0001
90  max_len = 0.002
91  max_skew = 0.8
92  remesh_interval = 1
93  maxiter = 500
94  diff_coeff = .5
95
96
97  #simulation
98
99  solver = pyfluent.launch_fluent(
100     precision="double",
101     processor_count=16,
102     mode="solver",
103     dimension = pyfluent.Dimension.TWO,
104     ui_mode= "gui",
105     cwd=workingFolder,
106 )
107
108 solver.file.read_mesh(file_name=meshPath)
109
110 solver.setup.general.solver.time = "unsteady-1st-order"
111 solver.setup.models.viscous.model = "laminar"
112 solver.setup.materials.fluid.make_a_copy(from_ = "air", to = "blood")
113 solver.setup.materials.fluid['blood'] = {"density" : {"option" : "constant", "value" : blood_dens},          ...
    ↪  "viscosity" : {"option" : "constant", "value" : blood_visc}, "chemical_formula" : ""}
114 solver.tui.define.boundary_conditions.zone_type('lv_interior', 'fluid')
115 solver.setup.cell_zone_conditions.fluid['lv_interior'] = {"material" : "blood"}
116
117 solver.tui.solve.set.p_v_coupling(24)#20 SIMPLE 21 SIMPLEC22 PISO 24 Coupled 25 Fractional Step
```

```
118 solver.tui.solve.set.p_v_controls(fcn, momentum_rf, pressure_rf)
119
120
121 solver.tui.define.user_defined.use_built_in_compiler('yes')
122 solver.tui.define.user_defined.compiled_functions('compile', '"libudf"', 'yes', '"LV_MOTION.c"','"", '"")
123 solver.tui.define.user_defined.compiled_functions('load', '"libudf"')
124
125 solver.tui.define.dynamic_mesh.dynamic_mesh('yes', 'no', 'no', 'no', 'yes')
126 solver.tui.define.dynamic_mesh.zones.create('lvw_in_lf', 'user-defined',                                    ...
    ↪  '"Implement_Motion_LV_IN::libudf"', 'no', 'no', '4', 'constant', '0.', 'no', 'no', 'no')
127 solver.tui.define.dynamic_mesh.zones.create('lvw_out_lf', 'user-defined',                                   ...
    ↪  '"Implement_Motion_LV_OUT::libudf"', 'no', 'no', '4', 'constant', '0.', 'no', 'no', 'no')
128
129
130 solver.tui.define.user_defined.execute_on_demand('"Read_Motion_LV_IN::libudf"')
131 solver.tui.define.user_defined.execute_on_demand('"Read_Motion_LV_OUT::libudf"')
132
133
134 solver.tui.file.read_transient_table(r'".\\data\\fluent\\velocity_magn.txt"')
135
136 solver.tui.define.boundary_conditions.zone_type('lvw_in_lf', 'wall')
137 solver.tui.define.boundary_conditions.zone_type('lvw_out_lf', 'wall')
138
139 solver.tui.define.boundary_conditions.zone_type('interior-lv_interior', 'interior')
140
141
142 solver.tui.define.dynamic_mesh.controls.smoothing('yes')
143 solver.tui.define.dynamic_mesh.controls.smoothing_parameters.smoothing_method('diffusion')
144 solver.tui.define.dynamic_mesh.controls.smoothing_parameters.max_iter('%d'%(maxiter))
145 solver.tui.define.dynamic_mesh.controls.smoothing_parameters.diffusion_coeff_parameter('%.1f'%(diff_coeff))
146
147 solver.tui.define.dynamic_mesh.controls.remeshing('yes')
148 solver.tui.define.dynamic_mesh.controls.remeshing_parameters.unified_remeshing('no')
149 solver.tui.define.dynamic_mesh.controls.remeshing_parameters.remeshing_methods('yes','no','no','no')
150 solver.tui.define.dynamic_mesh.controls.remeshing_parameters.length_min('%.8f'%(min_len))
151 solver.tui.define.dynamic_mesh.controls.remeshing_parameters.length_max('%.8f'%(max_len))
152 solver.tui.define.dynamic_mesh.controls.remeshing_parameters.cell_skew_max('%.2f'%(max_skew))
153 solver.tui.define.dynamic_mesh.controls.remeshing_parameters.size_remesh_interval('1')
154
155 solver.solution.report_definitions.surface["inlet-vel-rdef"] = {}
156 solver.solution.report_definitions.surface['inlet-vel-rdef'].report_type = "surface-facetavg"
157 solver.solution.report_definitions.surface['inlet-vel-rdef'] = {"field" : "velocity-magnitude",              ...
    ↪  "surface_names" : ["mv"]}
158 solver.solution.report_definitions.surface["outlet-vel-rdef"] = {}
159 solver.solution.report_definitions.surface['outlet-vel-rdef'].report_type = "surface-facetavg"
160 solver.solution.report_definitions.surface['outlet-vel-rdef'] = {"field" : "velocity-magnitude",             ...
    ↪  "surface_names" : ["av"]}
161 solver.solution.report_definitions.volume["area-rdef"] = {}
162 solver.solution.report_definitions.volume['area-rdef'].report_type = "volume-zonevol"
163 solver.solution.report_definitions.volume['area-rdef'] = {"cell_zones" : ["lv_interior"]}
164 solver.solution.report_definitions.volume["mean-pressure-rdef"] = {}
165 solver.solution.report_definitions.volume['mean-pressure-rdef'].report_type = "volume-average"
166 solver.solution.report_definitions.volume['mean-pressure-rdef'] = {"field" : "absolute-pressure",           ...
    ↪  "cell_zones" : ["lv_interior"]}
167 solver.solution.report_definitions.volume["max-pressure-rdef"] = {}
```

```
168 solver.solution.report_definitions.volume['max-pressure-rdef'].report_type = "volume-max"
169 solver.solution.report_definitions.volume['max-pressure-rdef'] = {"field" : "absolute-pressure",
    ↪  "cell_zones" : ["lv_interior"]}
170 solver.solution.report_definitions.volume["min-pressure-rdef"] = {}
171 solver.solution.report_definitions.volume['min-pressure-rdef'].report_type = "volume-min"
172 solver.solution.report_definitions.volume['min-pressure-rdef'] = {"field" : "absolute-pressure",
    ↪  "cell_zones" : ["lv_interior"]}
173
174 solver.solution.report_definitions.surface["mean-wss-rdef"] = {}
175 solver.solution.report_definitions.surface['mean-wss-rdef'].report_type = "surface-facetavg"
176 solver.solution.report_definitions.surface['mean-wss-rdef'] = {"field" : "wall-shear", "surface_names" :
    ↪  ["lvw_out_lf"]}
177
178 solver.solution.report_definitions.surface["max-wss-rdef"] = {}
179 solver.solution.report_definitions.surface['max-wss-rdef'].report_type = "surface-facetmax"
180 solver.solution.report_definitions.surface['max-wss-rdef'] = {"field" : "wall-shear", "surface_names" :
    ↪  ["lvw_out_lf"]}
181
182 solver.solution.report_definitions.surface["massflow-rate-mv-rdef"] = {}
183 solver.solution.report_definitions.surface["massflow-rate-mv-rdef"].report_type = 'surface-massflowrate'
184 solver.solution.report_definitions.surface["massflow-rate-mv-rdef"] = {"surface_names" : ["mv"]}
185 solver.solution.report_definitions.surface["massflow-rate-av-rdef"] = {}
186 solver.solution.report_definitions.surface["massflow-rate-av-rdef"].report_type = 'surface-massflowrate'
187 solver.solution.report_definitions.surface["massflow-rate-av-rdef"] = {"surface_names" : ["av"]}
188
189 solver.solution.monitor.report_plots["velocity-rplot"] = {}
190 solver.solution.monitor.report_plots['velocity-rplot'] = {"report_defs" : ["inlet-vel-rdef",
    ↪  "outlet-vel-rdef"]}
191 solver.solution.monitor.report_plots["area-rplot"] = {}
192 solver.solution.monitor.report_plots['area-rplot'] = {"report_defs" : ["area-rdef"]}
193 solver.solution.monitor.report_plots["pressure-rplot"] = {}
194 solver.solution.monitor.report_plots['pressure-rplot'] = {"report_defs" : ["mean-pressure-rdef",
    ↪  "max-pressure-rdef", "min-pressure-rdef"]}
195
196 solver.solution.monitor.report_plots["wss-rplot"] = {}
197 solver.solution.monitor.report_plots['wss-rplot'] = {"report_defs" : ["mean-wss-rdef", "max-wss-rdef"]}
198 solver.solution.monitor.report_plots["massflow-mv-rplot"] = {}
199 solver.solution.monitor.report_plots['massflow-mv-rplot'] = {"report_defs" : ["massflow-rate-mv-rdef",
    ↪  "massflow-rate-av-rdef"]}
200 solver.solution.monitor.report_files["check-rfile"] = {}
201 solver.solution.monitor.report_files['check-rfile'] = {"report_defs" : ["flow-time", "inlet-vel-rdef",
    ↪  "outlet-vel-rdef", "area-rdef","massflow-rate-mv-rdef", "massflow-rate-av-rdef"]}
202 solver.solution.monitor.report_files["results-rfile"] = {}
203 solver.solution.monitor.report_files['results-rfile'] = {"report_defs" : ["flow-time",
    ↪  "mean-pressure-rdef", "max-pressure-rdef", "min-pressure-rdef", "mean-wss-rdef", "max-wss-rdef"]}
204
205 export_dir  = workingFolder + f'\\results\\velocity_csv'
206
207 solver.tui.file.transient_export.ascii(export_dir + "\\fluent_export_csv", 'interior-lv_interior', '()',
    ↪  'x-velocity', 'y-velocity','velocity-magnitude',
    ↪  'absolute-pressure','mass-imbalance','strain-rate-mag','vorticity-mag',
    ↪  'wall-shear','lambda2-criterion','q-criterion','stream-function', 'quit', 'no', 'y', f'"export-csv"',
    ↪  '"time-step"', '1', 'time-step')
```

```
208 solver.tui.file.transient_export.ensight_gold_transient(export_dir + "\\fluent_export_ensight",                ...
    ↪ 'interior-lv_interior', '()', 'lv_interior', '()','x-velocity', 'y-velocity','velocity-magnitude',        ...
    ↪ 'absolute-pressure','mass-imbalance','strain-rate-mag','vorticity-mag',                                   ...
    ↪ 'wall-shear','lambda2-criterion','q-criterion','stream-function', 'quit', 'no', 'no',                     ...
    ↪ f'"export-ensight"', '"time-step"', '1', 'no')
209
210
211 for _ in range(period):
212
213     solver.tui.define.gap_model.enable('yes')
214     solver.tui.define.gap_model.create('"flow_gap"','lvw_in_lf','lvw_out_lf','()',gap_value,'')
215     solver.tui.define.boundary_conditions.zone_type('av', 'pressure-outlet')
216                                                                                                             ...
    ↪ solver.setup.boundary_conditions.pressure_outlet['av']={"momentum":{"gauge_pressure":0,'p_backflow_spec_gen':'Total
    ↪ Pressure'}}
217
218     solver.tui.define.boundary_conditions.zone_type('mv', 'pressure-inlet')
219     solver.setup.boundary_conditions.pressure_inlet['mv']={"momentum":{"gauge_total_pressure":0}}
220
221
222     if _ == 0 :
223         solver.tui.solve.initialize.compute_defaults.all_zones
224         solver.solution.initialization.initialization_type = "standard"
225         solver.solution.initialization.defaults['pressure'] = 0
226         solver.solution.initialization.standard_initialize()
227         solver.solution.run_calculation.transient_controls.time_step_count = int(nb_time_step*time_factor)
228         solver.solution.run_calculation.transient_controls.time_step_size = dt/time_factor
229         solver.solution.run_calculation.transient_controls.max_iter_per_time_step = max_iter_per_step_value
230
231     solver.solution.run_calculation.dual_time_iterate(time_step_count = int(i_dia*time_factor),             ...
    ↪ max_iter_per_step = int(max_iter_per_step_value))
232
233     solver.tui.define.gap_model.enable('no')
234
235     solver.tui.define.boundary_conditions.zone_type('mv', 'velocity-inlet')
236     solver.setup.boundary_conditions.velocity_inlet['mv']
237     solver.setup.boundary_conditions.velocity_inlet['mv'] = {"momentum" : {"velocity" : {"profile_name" :    ...
    ↪ 'velocity_magn_profile', "field_name": 'velocity_magn' , "option" :                                       ...
    ↪ "profile"},'velocity_specification_method' : "Magnitude, Normal to Boundary"}}
238
239
240     solver.tui.define.boundary_conditions.zone_type('av', 'wall')
241
242     solver.solution.run_calculation.dual_time_iterate(time_step_count = int((i_T - i_dia)*time_factor),     ...
    ↪ max_iter_per_step = int(max_iter_per_step_value))
```

Listing 4 PyAnsys : Set-up et calcul des conditions limites

```
1 workingFolder = 'Y:\\Recherche\\DEV\\PyAnsys\\case4\\'
2 dataFolder = 'Y:\\Recherche\\DEV\\PyAnsys\\case4\\data\\'
3
```

```
4
5
6   parts = ['MV','AV','LVW_in','LVW_out','MVadd_in','MVadd_out','AVadd_in','AVadd_out']
7   coord_parts = {}
8
9   for _ in range(len(parts)):
10      x = np.loadtxt(dataFolder + f'geom\\xcoord_{parts[_]}.csv', delimiter=',',skiprows=2)/1000
11      y = np.loadtxt(dataFolder + f'geom\\ycoord_{parts[_]}.csv', delimiter=',',skiprows=2)/1000
12      coord_parts[parts[_]] = [x[:,:], y[:,:]]### x(t,p); y(t,p)
13
14
15  nb_time_step = coord_parts['MV'][0].shape[0]
16
17  coord_parts['LVW_in'] = [coord_parts['LVW_in'][0][:,::-1], coord_parts['LVW_in'][1][:,::-1]]
18
19  list1 = np.column_stack((coord_parts['LVW_in'][0][0,:],coord_parts['LVW_in'][1][0,:]))
20
21  list2 = np.column_stack((coord_parts['LF_L'][0][0,:],coord_parts['LF_L'][1][0,:]))
22
23  common_points = set(map(tuple, list1)) & set(map(tuple, list2 ))
24
25  common_point = list(common_points)[-1]  # Get the last common point
26  index_list1 = np.where((list1 == common_point).all(axis=1))[0][0]
27
28  x_mod = np.column_stack((coord_parts['LVW_in'][0][:, :index_list1 + 1], coord_parts['LF_L'][0][:, :]))
29  y_mod = np.column_stack((coord_parts['LVW_in'][1][:, :index_list1 + 1], coord_parts['LF_L'][1][:, :]))
30
31  coord_parts['LV_in_LF'] = [x_mod, y_mod]
32  parts.append('LV_in_LF')
33
34  x_mod = coord_parts['LVW_in'][0][:, :index_list1 +1]
35  y_mod = coord_parts['LVW_in'][1][:, :index_list1 +1]
36
37  coord_parts['LV_in_red'] = [x_mod, y_mod]
38  parts.append('LV_in_red')
39
40  list1 = np.column_stack((coord_parts['LVW_out'][0][0,:],coord_parts['LVW_out'][1][0,:]))
41
42  list2 = np.column_stack((coord_parts['LF_R'][0][0,::-1],coord_parts['LF_R'][1][0,::-1]))
43
44  common_points = set(map(tuple, list1)) & set(map(tuple, list2 ))
45
46  common_point = list(common_points)[-1]  # Get the first last point
47  index_list1 = np.where((list1 == common_point).all(axis=1))[0][0]
48
49  x_mod = np.column_stack((coord_parts['LVW_out'][0][:, : index_list1], coord_parts['LF_R'][0][:, :]))
50  y_mod = np.column_stack((coord_parts['LVW_out'][1][:, : index_list1], coord_parts['LF_R'][1][:, :]))
51
52  coord_parts['LV_out_LF'] = [x_mod, y_mod]
53  parts.append('LV_out_LF')
54
55  x_mod = coord_parts['LVW_out'][0][:, : index_list1+1]
56  y_mod = coord_parts['LVW_out'][1][:, : index_list1+1]
57
58  coord_parts['LV_out_red'] = [x_mod[:,:], y_mod[:,:]]
```

```
parts.append('LV_out_red')

x_mod = np.column_stack((coord_parts['AVadd_in'][0][:, 0], coord_parts['AVadd_out'][0][:, 1]))
y_mod = np.column_stack((coord_parts['AVadd_in'][1][:, 0], coord_parts['AVadd_out'][1][:, 1]))

coord_parts['AV_mod'] = [x_mod, y_mod]
parts.append('AV_mod')

subparts = ['MV','MVadd_in','LF_L','LV_in_red','AV_mod','LV_out_red','LF_R','MVadd_out']
subparts_nodes = {}

# write down init geometry for surface reconstruction
fname = dataFolder + "geom\\curve_init.txt"

with open(fname, 'w') as f:
    f.write('3d=false\npolyline=false\nfit=false\nfittol=0.01\n \n')

    for _ in range(len(subparts)):
        count=0
        x = coord_parts[subparts[_]][0][0,:]*1000
        y = coord_parts[subparts[_]][1][0,:]*1000
        for j in range(x.shape[0]):
            f.write(f'1\t{round(x[j],8)}\t{round(y[j],8)}\n')
            count+=1
        print(f'{subparts[_]} a {count} noeuds')
        subparts_nodes[subparts[_]] = count
        f.write(' \n')
```

Listing 5 Reconstruction Géométrique

```
#Read and sort nodes ID fluent output

LV_IN_MESH_INIT, nb_msh_nodes_in = clean_IDs(nodesPath_LV_IN)
LV_OUT_MESH_INIT, nb_msh_nodes_out = clean_IDs(nodesPath_LV_OUT)

#Propagate nodes positions to match reference movement

LV_IN_MESH = mesh_position_homo(coord_parts['LV_in_LF'][0],coord_parts['LV_in_LF'][1],t,LV_IN_MESH_INIT )
LV_OUT_MESH = mesh_position_homo(coord_parts['LV_out_LF'][0],coord_parts['LV_out_LF'][1],t,LV_OUT_MESH_INIT )

# change time_step + write UDF node motion

LV_IN_MESH_TR = time_factor_interp(LV_MESH = LV_IN_MESH, t=t, T=T, time_factor=time_factor)
write_LV_motion(period=period,LV_MESH_INIT=LV_IN_MESH_INIT,LV_MESH=LV_IN_MESH_TR,motionPath=motionPath_LV_IN)

LV_OUT_MESH_TR = time_factor_interp(LV_MESH = LV_OUT_MESH, t=t, T=T, time_factor=time_factor)
write_LV_motion(period=period,LV_MESH_INIT=LV_OUT_MESH_INIT,LV_MESH=LV_OUT_MESH_TR,motionPath=motionPath_LV_OUT)

    script =  f"""// Authors: Aurélien Lacourt
```

```
// Polytechnique Montreal, Canada
#include "udf.h"
#include "unsteady.h"
#include "dynamesh_tools.h"
//
// Declare variables
int n_row_in = {nb_time_step*period*time_factor};
int n_col_in = {2*LV_IN_MESH.shape[1]};
int n_row_out = {nb_time_step*period*time_factor};
int n_col_out = {2*LV_OUT_MESH.shape[1]};
int i;
int row;
int col;
double **coord_in;
double **coord_out;
//
DEFINE_ON_DEMAND(Read_Motion_LV_IN)
{{
    FILE* data;
    data = fopen("{motionPath_LV_IN}","r");
    coord_in = malloc(n_row_in * sizeof *coord_in);
    coord_in[0] = malloc(n_col_in * n_row_in * sizeof **coord_in);
    for (i=1;i<n_row_in;i++)
    {{
        coord_in[i] = coord_in[i-1] + n_col_in;
    }}
    for (row = 0; row < n_row_in; ++row)
    {{
        for (col = 0; col < n_col_in; ++col)
        {{
            fscanf(data, "%lf", &coord_in[row][col]);
        }}
    }}
    fclose(data);
}}
//
DEFINE_GRID_MOTION(Implement_Motion_LV_IN, domain, dt, time, dtime)
{{
    Thread *tf = DT_THREAD(dt);
    face_t f;
    Node *v;
    int n,compt_n,compt_t;

    SET_DEFORMING_THREAD_FLAG(THREAD_T0(tf));

    begin_f_loop(f, tf)
    {{
        f_node_loop(f, tf, n)
        {{
            v = F_NODE(f, tf, n);
            if (NODE_POS_NEED_UPDATE(v))
            {{
                v = F_NODE(f, tf, n);

                for (compt_n = 0;compt_n<n_col_in;compt_n+=2)
```

```
                    {{
                        if (coord_in[0][compt_n]==NODE_ID(v))
                        break;
                    }}
                    compt_t = (int) (time/dtime)%(n_row_in-1)+1;
                    NODE_X(v) = coord_in[compt_t][compt_n];
                    NODE_Y(v) = coord_in[compt_t][compt_n+1];
                    NODE_POS_UPDATED(v);
                }}
            }}
        }}
        end_f_loop(f, tf);
    }}
    DEFINE_ON_DEMAND(Read_Motion_LV_OUT)
    {{
        FILE* data;
        data = fopen("{motionPath_LV_OUT}","r");
        coord_out = malloc(n_row_out * sizeof *coord_out);
        coord_out[0] = malloc(n_col_out * n_row_out * sizeof **coord_out);
        for (i=1;i<n_row_out;i++)
        {{
            coord_out[i] = coord_out[i-1] + n_col_out;
        }}
        for (row = 0; row < n_row_out; ++row)
        {{
            for (col = 0; col < n_col_out; ++col)
            {{
                fscanf(data, "%lf", &coord_out[row][col]);
            }}
        }}
        fclose(data);
    }}
    //
    DEFINE_GRID_MOTION(Implement_Motion_LV_OUT, domain, dt, time, dtime)
    {{
        Thread *tf = DT_THREAD(dt);
        face_t f;
        Node *v;
        int n,compt_n,compt_t;

        SET_DEFORMING_THREAD_FLAG(THREAD_TO(tf));

        begin_f_loop(f, tf)
        {{
            f_node_loop(f, tf, n)
            {{
                v = F_NODE(f, tf, n);
                if (NODE_POS_NEED_UPDATE(v))
                {{
                    v = F_NODE(f, tf, n);

                    for (compt_n = 0;compt_n<n_col_out;compt_n+=2)
                    {{
                        if (coord_out[0][compt_n]==NODE_ID(v))
                        break;
```

```
                    }}
                    compt_t = (int) (time/dtime)%(n_row_out-1)+1;
                    NODE_X(v) = coord_out[compt_t][compt_n];
                    NODE_Y(v) = coord_out[compt_t][compt_n+1];
                    NODE_POS_UPDATED(v);
                }}
            }}
        }}
        end_f_loop(f, tf);
    }}
    """

    with open(mesh_motionPath, 'w') as f:
        f.write(script)

    shutil.copy2(mesh_motionPath, workingFolder)
```

Listing 6 Déformation de la frontière

```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
import pandas as pd
from scipy.spatial import cKDTree
from scipy.interpolate import splev, splrep
import os
import glob
import time as timer
from skspatial.objects import Line
from scipy.interpolate import griddata, LinearNDInterpolator, interp1d, splrep, splev, RegularGridInterpolator
from shapely.geometry import Polygon
from scipy.signal import argrelextrema
from scipy.interpolate import SmoothBivariateSpline, RectBivariateSpline
from scipy.optimize import root_scalar, minimize, differential_evolution, shgo
from scipy.spatial.distance import cdist
import scipy.optimize as spopti
from pint import Quantity
from sklearn.metrics import mean_squared_error
from scipy.integrate import simpson
import math
import shutil

def angle_between_vectors(a, b):
    # Compute dot product
    dot_product = np.dot(a, b)
    # Compute norms (magnitudes)
    norm_a = np.linalg.norm(a)
    norm_b = np.linalg.norm(b)
    # Compute angle in radians
    angle_rad = np.arccos(dot_product / (norm_a * norm_b))
    # Convert to degrees
```

```
33      return angle_rad
34
35
36  def init_parameters(beta_t, beta_n, p_mv, alpha = 0.95, verbose = False ):
37
38      tau = p_mv[1,:] - p_mv[0,:]
39      norm_mv = np.linalg.norm(tau)
40
41      tau = tau/norm_mv
42      n = -np.array([-tau[1], tau[0]])
43
44      p_tip = beta_t*norm_mv*tau + beta_n*norm_mv*n + p_mv[0]
45
46      l_l = alpha*np.linalg.norm(p_tip-p_mv[0])
47      l_r = alpha*np.linalg.norm(p_tip-p_mv[1])
48
49      angle_off_set_l = angle_between_vectors(p_mv[1] - p_mv[0] , p_tip - p_mv[0])
50      angle_off_set_r = angle_between_vectors(p_mv[0] - p_mv[1] , p_tip - p_mv[1])
51
52      if verbose:
53          plt.plot(p_mv[:,0],p_mv[:,1])
54          plt.scatter(p_tip[0], p_tip[1])
55          plt.axis('equal')
56          plt.show()
57
58      return l_l, l_r, angle_off_set_l, angle_off_set_r
59
60
61  def quadratic_bezier_with_gradients(P0, P2, G0, G2, scale=0.3, num_points=100, verbose=False):
62      """
63      Generates a quadratic Bézier curve given the start and end points and their gradients.
64
65      Parameters:
66          P0, P2 : array-like, shape (2,)
67              Start and end points of the curve.
68          G0, G2 : array-like, shape (2,)
69              Gradients (tangent vectors) at P0 and P2.
70          scale : float
71              Scaling factor for the control point.
72          num_points : int
73              Number of points to sample on the curve.
74
75      Returns:
76          curve : array, shape (num_points, 2)
77              The computed Bézier curve points.
78      """
79      # Compute the control point P1
80      P1 = (P0 + P2) / 2 + scale * (G0 - G2) / 2
81
82      # Compute Bézier curve points
83      t = np.linspace(0, 1, num_points)[:, None]
84      curve = (1 - t)**2 * P0 + 2 * (1 - t) * t * P1 + t**2 * P2
85
86      if verbose:
87          # Define points and gradients
```

```
88          P0 = np.array([0, 0])    # Start point
89          P2 = np.array([2, 0])    # End point
90          G0 = np.array([1, 2])    # Gradient at P0
91          G2 = np.array([-1, -2])  # Gradient at P2
92
93          # Generate Bézier curve
94          curve, P1, shape = quadratic_bezier_with_gradients(P0, P2, G0, G2,scale = 10)
95
96          # Plot
97          plt.plot(curve[:, 0], curve[:, 1], label="Bézier Curve")
98          plt.scatter(*P0, color='red', label="Start Point (P0)")
99          plt.scatter(*P1, color='blue', label="Control Point (P1)")
100         plt.scatter(*P2, color='green', label="End Point (P2)")
101         plt.quiver(*P0, *G0, color='red', angles='xy', scale_units='xy', scale=5, width=0.005,
            ↪  label="Gradient at P0")
102         plt.quiver(*P2, *G2, color='green', angles='xy', scale_units='xy', scale=5, width=0.005,
            ↪  label="Gradient at P2")
103         plt.legend()
104         plt.title("Quadratic Bézier Curve with Tangent Constraints")
105         plt.grid()
106         plt.show()
107
108     return curve, P1, curve.shape
109
110
111 def with_in_bounds(P_0, dangle, angle_l_max,angle_off_set_l,angle_r_max,angle_off_set_r ):
112     angle_l, angle_r = P_0[0] + dangle, P_0[1] + dangle
113     if angle_l> angle_l_max:  angle_l = angle_l_max
114     if angle_l< angle_off_set_l: angle_l = angle_off_set_l
115     if angle_r> angle_r_max:  angle_r = angle_r_max
116     if angle_r< angle_off_set_r: angle_r = angle_off_set_r
117     return angle_l, angle_r
118
119 def project_point_on_segment(xa, xb, point):
120     """Project a point onto the line segment defined by xa and xb."""
121     xa, xb, point = np.array(xa), np.array(xb), np.array(point)
122
123     # Compute vector projections
124     #print((xb - xa))
125     #print(np.linalg.norm(xb - xa))
126     nab = (xb - xa)/np.linalg.norm(xb - xa)
127     ap = point - xa
128
129     # Compute projection scalar (clamped between 0 and 1 to keep it on the segment)
130
131     mu = np.dot(ap, nab)
132
133     # Compute projected point
134     xp = xa + mu * nab
135     return xp
136
137 def clean_IDs(file, start_left = True):
138
139     mesh_points = np.loadtxt(file, delimiter=",")
140
```

```
141     ids = mesh_points[:, 2]
142 
143     unique_ids, unique_indices = np.unique(ids, return_index=True)
144 
145     mesh_points = mesh_points[unique_indices]
146 
147     sorted_mesh_x = mesh_points[mesh_points[:, 0].argsort()]
148     sorted_mesh_y = mesh_points[mesh_points[:, 1].argsort()]
149     mask = np.ones((mesh_points.shape[0],2))
150 
151     for _ in range(mesh_points.shape[0]-1):
152         if sorted_mesh_x[_,0] == sorted_mesh_x[_+1,0]:
153             mask[_,0]=0
154         if sorted_mesh_y[_,1] == sorted_mesh_y[_+1,1]:
155             mask[_,1]=0
156     mask = mask.astype(bool)
157 
158     sorted_mesh_x = sorted_mesh_x[mask[:,0],:]
159     sorted_mesh_x = sorted_mesh_x[sorted_mesh_x[:, 0].argsort()]
160 
161     sorted_mesh_y = sorted_mesh_y[mask[:,1],:]
162     sorted_mesh_y = sorted_mesh_y[sorted_mesh_y[:, 0].argsort()]
163 
164     mi , ma = np.min(sorted_mesh_y[:,0]),np.max(sorted_mesh_y[:,0])
165 
166 
167     if start_left:  mesh_points = sorted_mesh_x
168     else: mesh_points = np.flip(sorted_mesh_x,axis=0)
169 
170     nb_nodes = mesh_points.shape[0]
171     print(nb_nodes)
172 
173     points = mesh_points[:,:2]
174 
175     idx = []
176     ordered_points = []
177     # Start with the extremity
178     start_idx = 0
179     ordered_points.append(points[start_idx,:])
180     remaining_points = np.delete(points, start_idx, axis=0)
181     while len(remaining_points) > 0:
182         # Find the closest point to the last point in the ordered list
183         tree = cKDTree(remaining_points)
184         search_points = ordered_points[-1]
185         closest_distances, closest_indices = tree.query(search_points, k=1)  # Find closest points
186         closest_point = remaining_points[closest_indices,:]
187         ordered_points.append(closest_point)
188         remaining_points = np.delete(remaining_points, closest_indices, axis=0)
189 
190     ordered_points = np.array(ordered_points)
191 
192     indexes = [np.where((points == row).all(axis=1))[0][0] for row in ordered_points]
193 
194     mesh_points = mesh_points[indexes,:]
195 
```

```
196     return mesh_points, nb_nodes
197
198 def mesh_position_homo(x,y, t,mesh_points):
199     nb_time_step = x.shape[0]
200     nb_points = x.shape[1]
201
202
203     curve = np.zeros((nb_time_step,nb_points,2))
204     curve[:,:,0] = x
205     curve[:,:,1] = y
206
207     S = np.zeros((nb_time_step,nb_points))
208     for time in range(nb_time_step):
209         s=0
210         S[time,0] = 0
211         for pts in range(nb_points - 1):
212             dx = curve[time,pts+1,0] - curve[time,pts,0]
213             dy = curve[time,pts+1,1] - curve[time,pts,1]
214             s+= np.sqrt(dx**2 +dy**2)
215             S[time,pts+1] = s
216         S[time,:] = S[time,:]/S[time,-1]
217
218     points  = np.zeros((nb_points*nb_time_step,2))
219     f = np.zeros((nb_points*nb_time_step,2))
220
221     count =0
222     for time_step in range(nb_time_step):
223         for pts in range(nb_points):
224
225             points[count,0] = S[0, pts]
226             points[count,1] = t[time_step]
227
228             f[count,0] = curve[time_step,pts,0]#### X_t = f_x(s,t)
229             f[count,1] = curve[time_step,pts,1]#### Y_t = f_y(s,t)
230
231             count+=1
232
233
234
235     S_ref = np.linspace(0,1, mesh_points.shape[0])
236     S_0, i_s0 = np.unique(S[0,:],return_index=True)
237     interp_x = RegularGridInterpolator(points=(t,S_0), values=curve[:,i_s0,0],method='linear')
238     interp_y = RegularGridInterpolator(points=(t,S_0), values=curve[:,i_s0,1],method='linear')
239
240     grid_s, grid_t = np.meshgrid(S_ref,t)
241
242
243     grid_x = interp_x((grid_t, grid_s))
244     grid_y = interp_y((grid_t, grid_s))
245
246     mesh_points_lp = np.zeros( (nb_time_step, mesh_points.shape[0], 2))
247
248     mesh_points_lp[:,:,0] = grid_x
249     mesh_points_lp[:,:,1] = grid_y
250
```

```
    mesh_points_lp[:,0,0], mesh_points_lp[:,-1,0] = x[:,0], x[:,-1]
    mesh_points_lp[:,0,1], mesh_points_lp[:,-1,1] = y[:,0], y[:,-1]

    return mesh_points_lp

def time_factor_interp(LV_MESH,t,T,time_factor):
    nb_time_step = LV_MESH.shape[0]

    #print(t)
    t_new = np.linspace(0,T,int(time_factor*nb_time_step))
    #print(t_new)
    dt = round(t_new[1] - t_new[0],8)
    t_new = np.array([k*dt for k in range(int(time_factor*nb_time_step))])
    t_new[-1] = t[-1]
    #print(t_new)
    # Perform interpolation along time axis
    interp_func = interp1d(t, LV_MESH, axis=0, kind='linear')  # Linear interpolation
    LV_MESH_TR = interp_func(t_new)

    return LV_MESH_TR

def write_LV_motion(period,LV_MESH_INIT,LV_MESH,motionPath):
    nb_time_step = LV_MESH.shape[0]
    nb_pts = LV_MESH_INIT.shape[0]
    nb_cat = LV_MESH_INIT.shape[1]

    points_coord =  np.zeros((nb_time_step, nb_pts, nb_cat))

    for time in range(nb_time_step):
        points_coord[time,:, 2] = LV_MESH_INIT[:, 2]
        points_coord[time,:, 0] = LV_MESH[time,:,0]
        points_coord[time,:, 1] = LV_MESH[time,:,1]


    MatchCoord = np.zeros((nb_time_step + 1, nb_pts*2))

    for points in range(nb_pts):
        MatchCoord[0,2*points] = points_coord[0, points, 2]
        MatchCoord[0,2*points+1] = points_coord[0, points, 2]
        MatchCoord[1:, 2*points] = points_coord[:, points, 0]
        MatchCoord[1:, 2*points+1] = points_coord[:, points, 1]

    with open(motionPath, 'w') as file:
        # Write the first line as integers (Node IDs)
        nodes_ID  = MatchCoord[0,:].astype(int)
        line = "".join(f"{nodes_ID[m]:d}\t" for m in range(nodes_ID.shape[0] - 1))
        line += "" + f"{nodes_ID[-1]:d}\n"  # Last column

        file.write(line)

        # Write the other lines as floats (Node coordinates)
        for _ in range(period):
            for n in range(1, MatchCoord.shape[0]):
                line = "".join(f"{MatchCoord[n, m]:3.16f}\t" for m in range(MatchCoord.shape[1] - 1))
                line += "" + f"{MatchCoord[n, -1]:3.16f}\n"  # Last column
```

```
306                 file.write(line)
307
308 def load_points(motionPath):
309
310     data = np.loadtxt(motionPath, delimiter='\t')
311     nb_time_step_tt, total_columns = data[1:,:].shape
312     nb_points = total_columns // 2
313     curve_mesh = data[1:,:].reshape(nb_time_step_tt, nb_points, 2)
314
315     return curve_mesh
316
317
```

Listing 7 Répertoire fonctions utilitaires

```
1  ####### Geometric parameters
2
3  D_MV = np.linalg.norm([coord_parts["MV"][0][0,0] - coord_parts["MV"][0][0,-1] ,coord_parts["MV"][1][0,0]-    ...
   ↪  coord_parts["MV"][1][0,-1]])
4  D_AV = np.linalg.norm([coord_parts["AV"][0][0,0] - coord_parts["AV"][0][0,-1] ,coord_parts["AV"][1][0,0]-    ...
   ↪  coord_parts["AV"][1][0,-1]])
5  print(f'MV diameter {D_MV*1000}, AV diameter {D_AV*1000} mm')
6
7  ##### define parameters
8  ## geometry
9
10 beta_t = .7 # proportion of anterior to posterior contribution to mitral valve area
11 beta_n = .25 # proportion of anterior to posterior contribution to mitral valve area
12
13 l_l, l_r, angle_off_set_l, angle_off_set_r = init_parameters(beta_t=beta_t, beta_n=beta_n, p_mv =            ...
   ↪  np.array([[coord_parts["MVadd_in"][0][0,0],coord_parts["MVadd_in"][1][0,0]],[coord_parts["MVadd_out"][0][0,0],coord_parts["M
   ↪  = 0.92, verbose=False)
14 print(l_l, l_r, angle_off_set_l*180/np.pi, angle_off_set_r*180/np.pi)
15
16 w_l = 2e-3 #mm
17 w_r = 2e-3 #mm
18
19 scale_factor = 0.01# for the curbature of the leaflet
20 #scale_factor = 0# for the curbature of the leaflet
21
22 ## dynamics optional
23 theta_l_max = 90 #degree
24 angle_l_max = theta_l_max*np.pi/180
25 theta_r_max = 90 #degree
26 angle_r_max = theta_r_max*np.pi/180
27
28 def leaflets(step,angle_l,angle_r):
29     nb_lf_nodes_tip = 31
30     nb_lf_nodes_term_up = 5
31     nb_lf_nodes_term_down = 5
32     nb_lf_nodes = nb_lf_nodes_tip + nb_lf_nodes_term_up + nb_lf_nodes_term_up
```

```
33
34
35  # compute leaflets
36  angles = np.loadtxt(dataFolder + "geom\\optimized_angles.txt" ,delimiter=',')
37  angle_l, angle_r = angles[:,0], angles[:,1]
38
39  for step, time in enumerate(t):
40
41      leaflet_l, leaflet_r = leaflets(step,angle_l[step],angle_r[step])
42      if step == 0:
43          x_leaflet_l = np.zeros((nb_time_step, leaflet_l.shape[0]))
44          y_leaflet_l = np.zeros((nb_time_step, leaflet_l.shape[0]))
45
46          x_leaflet_r = np.zeros((nb_time_step, leaflet_r.shape[0]))
47          y_leaflet_r = np.zeros((nb_time_step, leaflet_r.shape[0]))
48
49      x_leaflet_l[step,:], y_leaflet_l[step,:]  = leaflet_l[:,0],leaflet_l[:,1]
50      x_leaflet_r[step,:], y_leaflet_r[step,:]  = leaflet_r[:,0],leaflet_r[:,1]
51
52
53  coord_parts['LF_L'] = [x_leaflet_l[:,::-1], y_leaflet_l[:,::-1]]
54  coord_parts['LF_R'] = [x_leaflet_r[:,::-1], y_leaflet_r[:,::-1]]
55
56
57  #update coords_part
58
59  coord_parts['LVW_in'] = [coord_parts['LVW_in'][0][:,::-1], coord_parts['LVW_in'][1][:,::-1]]
60
61  list1 = np.column_stack((coord_parts['LVW_in'][0][0,:],coord_parts['LVW_in'][1][0,:]))
62
63  list2 = np.column_stack((coord_parts['LF_L'][0][0,:],coord_parts['LF_L'][1][0,:]))
64
65  # Find common points
66  common_points = set(map(tuple, list1)) & set(map(tuple, list2 ))
67
68  common_point = list(common_points)[-1]  # Get the last common point
69  index_list1 = np.where((list1 == common_point).all(axis=1))[0][0]
70
71
72  x_mod = np.column_stack((coord_parts['LVW_in'][0][:, :index_list1 + 1], coord_parts['LF_L'][0][:, :]))
73  y_mod = np.column_stack((coord_parts['LVW_in'][1][:, :index_list1 + 1], coord_parts['LF_L'][1][:, :]))
74
75
76  coord_parts['LV_in_LF'] = [x_mod, y_mod]
77  parts.append('LV_in_LF')
78
79  x_mod = coord_parts['LVW_in'][0][:, :index_list1 +1]
80  y_mod = coord_parts['LVW_in'][1][:, :index_list1 +1]
81
82  coord_parts['LV_in_red'] = [x_mod, y_mod]
83  parts.append('LV_in_red')
84
85  list1 = np.column_stack((coord_parts['LVW_out'][0][0,:],coord_parts['LVW_out'][1][0,:]))
86
87  list2 = np.column_stack((coord_parts['LF_R'][0][0,::-1],coord_parts['LF_R'][1][0,::-1]))
```

```
88
89  # Find common points
90  common_points = set(map(tuple, list1)) & set(map(tuple, list2 ))
91
92  common_point = list(common_points)[-1]  # Get the first last point
93  index_list1 = np.where((list1 == common_point).all(axis=1))[0][0]
94
95  x_mod = np.column_stack((coord_parts['LVW_out'][0][:, : index_list1], coord_parts['LF_R'][0][:, :]))
96  y_mod = np.column_stack((coord_parts['LVW_out'][1][:, : index_list1], coord_parts['LF_R'][1][:, :]))
97
98  coord_parts['LV_out_LF'] = [x_mod, y_mod]
99  parts.append('LV_out_LF')
100
101 x_mod = coord_parts['LVW_out'][0][:, : index_list1+1]
102 y_mod = coord_parts['LVW_out'][1][:, : index_list1+1]
103
104 coord_parts['LV_out_red'] = [x_mod[:,:], y_mod[:,:]]
105 parts.append('LV_out_red')
106
107
108 x_mod = np.column_stack((coord_parts['AVadd_in'][0][:, 0], coord_parts['AVadd_out'][0][:, 1]))
109 y_mod = np.column_stack((coord_parts['AVadd_in'][1][:, 0], coord_parts['AVadd_out'][1][:, 1]))
110
111
112 coord_parts['AV_mod'] = [x_mod, y_mod]
113 parts.append('AV_mod')
114
115
116 parts.append('LF_L')
117 parts.append('LF_R')
118
119
120
121     x_leaflet_l = np.zeros(nb_lf_nodes)
122     y_leaflet_l = np.zeros(nb_lf_nodes)
123
124     x_leaflet_r = np.zeros(nb_lf_nodes)
125     y_leaflet_r = np.zeros(nb_lf_nodes)
126
127
128     ### left side
129
130     tau = np.array([coord_parts['MVadd_in'][0][step,1],coord_parts['MVadd_in'][1][step,1]]) -                ...
        ↪  np.array([coord_parts['MVadd_in'][0][step,0],coord_parts['MVadd_in'][1][step,0]])
131     tau = -tau/np.linalg.norm(tau)
132     n = -np.array([-tau[1], tau[0]])
133
134     p_l = np.array([coord_parts['MVadd_in'][0][step,-1],coord_parts['MVadd_in'][1][step,-1]])
135     p_first = p_l
136     p_last =                                                                                                ...
        ↪  np.array([coord_parts['LVW_in'][0][step,len(coord_parts['LVW_in'][0][0,:])//2],coord_parts['LVW_in'][1][step,len(coord_p
137
138     MVT = np.linalg.norm(np.array([coord_parts['MVadd_in'][0][0,-1],coord_parts['MVadd_in'][1][0,-1]]) -      ...
        ↪  np.array([coord_parts['LVW_in'][0][0,len(coord_parts['LVW_in'][0][0,:])//2],coord_parts['LVW_in'][1][0,len(coord_parts['
139     #print(f'maximum valve thickness {MVT}')
```

```
140
141     p_center = (p_first + p_last)/2
142
143     tau_l = p_first - p_last
144     tau_l = -tau_l/np.linalg.norm(tau_l)
145     n_l = np.array([-tau_l[1], tau_l[0]])
146
147     p_tip = p_center + (l_l+np.dot(p_l - p_center,n))*(n*np.cos(angle_l) - tau*np.sin(angle_l)) +           ...
        ↪  (np.dot(p_l - p_center,tau))*(tau*np.cos(angle_l) + n*np.sin(angle_l))
148
149     scale = angle_l/angle_l_max * scale_factor
150     curve, P1, shape = quadratic_bezier_with_gradients(p_center, p_tip, n_l, (n*np.cos(angle_l) -           ...
        ↪  tau*np.sin(angle_l)) , scale = scale)
151
152     p = curve[int(shape[0]/2), :]
153     pp1 = curve[int(shape[0]/2) + 1 , :]
154     tau_star = pp1 - p
155     tau_star = -tau_star/np.linalg.norm(tau_star)
156     n_star = -np.array([-tau_star[1], tau_star[0]])
157     p_mid_up =  p + w_l/2*n_star
158     p_mid_down = p - w_l/2*n_star
159
160     p = curve[int(shape[0]/4), :]
161     pp1 = curve[int(shape[0]/4) + 1 , :]
162     tau_star = pp1 - p
163     tau_star = -tau_star/np.linalg.norm(tau_star)
164     n_star = -np.array([-tau_star[1], tau_star[0]])
165     p_mid_up_1 =  p + w_l/2*n_star
166     p_mid_down_1 = p - w_l/2*n_star
167
168     p = curve[int(shape[0]*3/4), :]
169     pp1 = curve[int(shape[0]*3/4) + 1 , :]
170     tau_star = pp1 - p
171     tau_star = -tau_star/np.linalg.norm(tau_star)
172     n_star = -np.array([-tau_star[1], tau_star[0]])
173     p_mid_up_2 =  p + w_l/2*n_star
174     p_mid_down_2 = p - w_l/2*n_star
175
176
177     p = curve[0, :]
178     pp1 = curve[1 , :]
179     tau_star = pp1 - p
180     tau_star = -tau_star/np.linalg.norm(tau_star)
181     n_star = np.array([-tau_star[1], tau_star[0]])
182     p_first_fixed = p + MVT*n_star/2
183     p_last_fixed = p - MVT*n_star/2
184
185
186     control_nodes_x = [p_first[0], p_mid_up_1[0], p_mid_up[0], p_mid_up_2[0], p_tip[0], p_mid_down_2[0],     ...
        ↪  p_mid_down[0], p_mid_down_1[0], p_last[0]]
187     control_nodes_y = [p_first[1], p_mid_up_1[1], p_mid_up[1], p_mid_up_2[1], p_tip[1], p_mid_down_2[1],     ...
        ↪  p_mid_down[1], p_mid_down_1[1], p_last[1]]
188     id_term_up = 1
189     id_term_down = -2
190
```

```
    s = [0]
    for _ in range(len(control_nodes_x)-1):
        dx = control_nodes_x[_+1] - control_nodes_x[_]
        dy = control_nodes_y[_+1] - control_nodes_y[_]
        ds  = np.sqrt(dx**2 + dy**2)
        s.append(s[-1]+ds)

    s = s/s[-1]

    control_nodes_x_tip = [p_first_fixed[0], p_mid_up_1[0], p_mid_up[0], p_mid_up_2[0], p_tip[0], p_mid_down_2[0], p_mid_down[0], p_mid_down_1[0], p_last_fixed[0]]
    control_nodes_y_tip = [p_first_fixed[1], p_mid_up_1[1], p_mid_up[1], p_mid_up_2[1], p_tip[1], p_mid_down_2[1], p_mid_down[1], p_mid_down_1[1], p_last_fixed[1]]

    s_tip = [0]
    for _ in range(len(control_nodes_x_tip)-1):
        dx = control_nodes_x_tip[_+1] - control_nodes_x_tip[_]
        dy = control_nodes_y_tip[_+1] - control_nodes_y_tip[_]
        ds  = np.sqrt(dx**2 + dy**2)
        s_tip.append(s_tip[-1]+ds)

    s_tip = s_tip/s_tip[-1]

    spl_x = splrep(s,control_nodes_x,k=3)
    spl_y = splrep(s,control_nodes_y,k=3)

    spl_x_tip = splrep(s_tip,control_nodes_x_tip,k=3)
    spl_y_tip = splrep(s_tip,control_nodes_y_tip,k=3)

    s_term_up = np.linspace(0, s[id_term_up], nb_lf_nodes_term_up + 1 )
    s_term_down = np.linspace(s[id_term_down], 1  , nb_lf_nodes_term_down + 1 )
    s_term_tip = np.linspace(s_tip[id_term_up], s_tip[id_term_down], nb_lf_nodes_tip)

    xi = np.concatenate((splev(s_term_up[:-1], spl_x),splev(s_term_tip, spl_x_tip),splev(s_term_down[1:], spl_x)))
    yi = np.concatenate((splev(s_term_up[:-1], spl_y),splev(s_term_tip, spl_y_tip),splev(s_term_down[1:], spl_y)))

    xi[0], xi[-1] = p_first[0], p_last[0]
    yi[0], yi[-1] = p_first[1], p_last[1]

    x_leaflet_l = xi
    y_leaflet_l = yi
    leaflet_l = np.column_stack((x_leaflet_l,y_leaflet_l))

    ### right side

    tau = np.array([coord_parts['MVadd_out'][0][step,1],coord_parts['MVadd_out'][1][step,1]]) - np.array([coord_parts['MVadd_out'][0][step,0],coord_parts['MVadd_out'][1][step,0]])
    tau = tau/np.linalg.norm(tau)
    n = np.array([-tau[1], tau[0]])

    p_r = np.array([coord_parts['MVadd_out'][0][step,0],coord_parts['MVadd_out'][1][step,0]])

    p_first = p_r
```

```
241
242     distances = np.linalg.norm(np.column_stack((coord_parts['LVW_out'][0][0,:],coord_parts['LVW_out'][1][0,:])) - np.array([coord_parts['MVadd_out'][0][0,0],coord_parts['MVadd_out'][1][0,0]]), axis=1)
243     closest_index = np.argmin(np.abs(distances - MVT))
244
245     p_last = np.array([coord_parts['LVW_out'][0][step,closest_index], coord_parts['LVW_out'][1][step,closest_index]])
246
247     p_center = (p_first + p_last)/2
248
249     tau_r = p_first - p_last
250     tau_r = -tau_r/np.linalg.norm(tau_r)
251     n_r = -np.array([-tau_r[1], tau_r[0]])
252
253     p_tip = p_center + (l_r+np.dot(p_r - p_center,n))*(n*np.cos(angle_r) - tau*np.sin(angle_r)) + (np.dot(p_r - p_center,tau))*(tau*np.cos(angle_r) + n*np.sin(angle_r))
254
255     scale = angle_r/angle_r_max * scale_factor
256     curve, P1, shape = quadratic_bezier_with_gradients(p_center, p_tip, n_r, (n*np.cos(angle_r) - tau*np.sin(angle_r)) , scale = scale)
257
258     p = curve[int(shape[0]/2), :]
259     pp1 = curve[int(shape[0]/2) + 1 , :]
260     tau_star = pp1 - p
261     tau_star = -tau_star/np.linalg.norm(tau_star)
262     n_star = np.array([-tau_star[1], tau_star[0]])
263     p_mid_up =  p + w_r/2*n_star
264     p_mid_down = p - w_r/2*n_star
265
266     p = curve[int(shape[0]/4), :]
267     pp1 = curve[int(shape[0]/4) + 1 , :]
268     tau_star = pp1 - p
269     tau_star = -tau_star/np.linalg.norm(tau_star)
270     n_star = np.array([-tau_star[1], tau_star[0]])
271     p_mid_up_1 =  p + w_r/2*n_star
272     p_mid_down_1 = p - w_r/2*n_star
273
274     p = curve[int(shape[0]*3/4), :]
275     pp1 = curve[int(shape[0]*3/4) + 1 , :]
276     tau_star = pp1 - p
277     tau_star = -tau_star/np.linalg.norm(tau_star)
278     n_star = np.array([-tau_star[1], tau_star[0]])
279     p_mid_up_2 =  p + w_r/2*n_star
280     p_mid_down_2 = p - w_r/2*n_star
281
282
283     p = curve[0, :]
284     pp1 = curve[1 , :]
285     tau_star = pp1 - p
286     tau_star = -tau_star/np.linalg.norm(tau_star)
287     n_star = np.array([-tau_star[1], tau_star[0]])
288     p_first_fixed = p + MVT*n_star/2
289     p_last_fixed = p - MVT*n_star/2
290
```

```
291
292     control_nodes_x = [p_first[0], p_mid_up_1[0], p_mid_up[0], p_mid_up_2[0], p_tip[0], p_mid_down_2[0],     ...
        ↪  p_mid_down[0], p_mid_down_1[0], p_last[0]]
293     control_nodes_y = [p_first[1], p_mid_up_1[1], p_mid_up[1], p_mid_up_2[1], p_tip[1], p_mid_down_2[1],     ...
        ↪  p_mid_down[1], p_mid_down_1[1], p_last[1]]
294     id_term_up = 1
295     id_term_down = -2
296
297     s = [0]
298     for _ in range(len(control_nodes_x)-1):
299         dx = control_nodes_x[_+1] - control_nodes_x[_]
300         dy = control_nodes_y[_+1] - control_nodes_y[_]
301         ds  = np.sqrt(dx**2 + dy**2)
302         s.append(s[-1]+ds)
303
304     s = s/s[-1]
305
306     control_nodes_x_tip = [p_first_fixed[0], p_mid_up_1[0], p_mid_up[0], p_mid_up_2[0], p_tip[0],            ...
        ↪  p_mid_down_2[0], p_mid_down[0], p_mid_down_1[0], p_last_fixed[0]]
307     control_nodes_y_tip = [p_first_fixed[1], p_mid_up_1[1], p_mid_up[1], p_mid_up_2[1], p_tip[1],            ...
        ↪  p_mid_down_2[1], p_mid_down[1], p_mid_down_1[1], p_last_fixed[1]]
308
309     s_tip = [0]
310     for _ in range(len(control_nodes_x_tip)-1):
311         dx = control_nodes_x_tip[_+1] - control_nodes_x_tip[_]
312         dy = control_nodes_y_tip[_+1] - control_nodes_y_tip[_]
313         ds  = np.sqrt(dx**2 + dy**2)
314         s_tip.append(s_tip[-1]+ds)
315
316     s_tip = s_tip/s_tip[-1]
317
318     spl_x = splrep(s,control_nodes_x,k=3)
319     spl_y = splrep(s,control_nodes_y,k=3)
320
321     spl_x_tip = splrep(s_tip,control_nodes_x_tip,k=3)
322     spl_y_tip = splrep(s_tip,control_nodes_y_tip,k=3)
323
324     s_term_up = np.linspace(0, s[id_term_up], nb_lf_nodes_term_up+1)
325     s_term_down = np.linspace(s[id_term_down], 1  , nb_lf_nodes_term_down+1)
326     s_term_tip = np.linspace(s_tip[id_term_up], s_tip[id_term_down], nb_lf_nodes_tip)
327
328     xi = np.concatenate((splev(s_term_up[:-1], spl_x),splev(s_term_tip, spl_x_tip),splev(s_term_down[1:],    ...
        ↪  spl_x)))
329     yi = np.concatenate((splev(s_term_up[:-1], spl_y),splev(s_term_tip, spl_y_tip),splev(s_term_down[1:],    ...
        ↪  spl_y)))
330
331     xi[0], xi[-1] = p_first[0], p_last[0]
332     yi[0], yi[-1] = p_first[1], p_last[1]
333
334     x_leaflet_r = xi
335     y_leaflet_r = yi
336     leaflet_r = np.column_stack((x_leaflet_r,y_leaflet_r))
337
338     return leaflet_l, leaflet_r
339
```

Listing 8 Calculs des paramètres géométriques et reconstruction de la VM

```
# shifting cycle to start with systole

area = np.zeros(nb_time_step)

for _ in range(nb_time_step):
    pgon = Polygon(np.column_stack((coord_parts['LVW_out'][0][_,:],coord_parts['LVW_out'][1][_,:])))
    area[_] = pgon.area


t_star = np.array([k*dt for k in range(2*nb_time_step)])

area_diff = np.zeros_like(area_mod)
area_diff = np.diff(area_mod,axis=0, n=1, append=area_mod[0])
area_diff = area_diff/dt

spl_area = splrep(np.array([k*dt for k in range(2*nb_time_step)]),np.concatenate((area_mod,area_mod)),k=5,  ...
↪  s=3e-10)
#spl_area = splrep(t,area_mod,k=5, s=5e-10)
area_splined = splev(t_star, spl_area)
area_splined_diff = np.diff(area_splined,axis=0, n=1, append=area_splined[0])/dt

local_extreme_max = argrelextrema(data=np.abs(area_diff), comparator=np.greater, order=100)[0].astype(int)
local_extreme_min = argrelextrema(data=np.abs(area_diff), comparator=np.less, order=100)[0].astype(int)

def V_mv(step, angle_l, angle_r, verbose=False):
    lf_l, lf_r = leaflets(step, angle_l,angle_r)

    lf_l_cut, lf_r_cut = lf_l[: lf_l.shape[0]//2 +1,:],lf_r[: lf_r.shape[0]//2 +1,:]

    x =                                                                                                  ...
    ↪  np.concatenate((lf_r_cut[::-1,0],coord_parts['MVadd_out'][0][0,:],coord_parts['MV'][0][0,:],coord_parts['MVadd_in'][0][0
    y =                                                                                                  ...
    ↪  np.concatenate((lf_r_cut[::-1,1],coord_parts['MVadd_out'][1][0,:],coord_parts['MV'][1][0,:],coord_parts['MVadd_in'][1][0

    if verbose:
        plt.plot(x,y)
        plt.scatter(lf_l[lf_l.shape[0]//2,0],lf_l[lf_l.shape[0]//2,1], color='r')
        plt.scatter(lf_r[lf_r.shape[0]//2,0],lf_r[lf_r.shape[0]//2,1], color='g')
        plt.axis('equal')
        plt.show()

    pgon = Polygon(np.column_stack((x,y)))
    area_lf = pgon.area

    return area_lf


#natural
V_0 = np.max(np.array([V_mv(step = _, angle_l = angle_off_set_l, angle_r = angle_off_set_r, verbose=False)  ...
↪  for _ in range(nb_time_step)]))

```

```
phi_i = area_diff.copy()#S2 without parameters
phi_i[:np.argmin(area_mod)+1] = 0

V_max_geo = np.min(np.array([V_mv(step = _, angle_l = angle_l_max, angle_r = angle_r_max, verbose=False)     ...
↪  for _ in range(nb_time_step)]))

V_max = min(np.cumsum(phi_i[np.argmin(area_mod)+1:local_extreme_max[1]]*dt,axis=0)[-1] + V_0, V_max_geo)

gamma = (area_diff[local_extreme_min[1]] - area_diff[np.argmin(area_mod)])/                                  ...
↪  (area_diff[local_extreme_max[1]] - area_diff[np.argmin(area_mod)])
V_mid = gamma*V_max +(1-gamma)*V_0

alpha_1  = (V_max - V_mid)/np.sum(phi_i[local_extreme_max[1]:local_extreme_min[1]])/dt
V_peak = V_mid + np.sum(phi_i[local_extreme_min[1]:local_extreme_max[2]])*dt
alpha_2  = (V_peak-V_0)/np.sum(phi_i[local_extreme_max[2]:])/dt
print(alpha_1, alpha_2)

f_s = np.zeros_like(phi_i)
f_s[local_extreme_max[1]:local_extreme_min[1]] = 1 + alpha_1
f_s[local_extreme_max[2]:] = 1 + alpha_2

DV = phi_i*(1-f_s)*dt

V=[V_0]

for dv in DV:
    V_star = V[-1]+dv

    if V_star > V_max:
        V_star = V_max
    elif V_star < V_0:
        V_star = V_0

    V.append(V_star)

V=np.array(V[1:])


#forced
V_0 = np.max(np.array([V_mv(step = _, angle_l = angle_off_set_l, angle_r = angle_off_set_r, verbose=False)   ...
↪  for _ in range(nb_time_step)]))


phi_i = area_diff.copy()#S2 without parameters
phi_i[:np.argmin(area_mod)+1] = 0

V_max_geo = np.min(np.array([V_mv(step = _, angle_l = angle_l_max, angle_r = angle_r_max, verbose=False)     ...
↪  for _ in range(nb_time_step)]))

V_max = min(np.cumsum(phi_i[np.argmin(area_mod)+1:local_extreme_max[1]]*dt,axis=0)[-1] + V_0, V_max_geo)


alpha  = (V_max-V_0)/np.sum(phi_i[local_extreme_max[2]:])/dt

f_s = np.zeros_like(phi_i)
```

```
f_s[local_extreme_max[1]:local_extreme_min[1]] = 1
f_s[local_extreme_max[2]:] = 1 + alpha

DV = phi_i*(1-f_s)*dt

V=[V_0]

for dv in DV:
    V_star = V[-1]+dv

    if V_star > V_max:
        V_star = V_max
    elif V_star < V_0:
        V_star = V_0

    V.append(V_star)

V=np.array(V[1:])

```

Listing 9 Profil temporelle de V_{up} Stratégie *Naturelle* et *Forcée*

```
p_0 = [angle_off_set_l, angle_off_set_r]
bounds = [(angle_off_set_l,angle_l_max), (angle_off_set_r,angle_r_max)]

ANGLE_R = np.zeros(nb_time_step)
ANGLE_L = np.zeros(nb_time_step)
V_OPT = np.zeros(nb_time_step)
for _ in range(nb_time_step):

    print(_/nb_time_step*100)
    def objective(p):
        angle_l, angle_r = p
        return abs(V_mv(step = _ , angle_l = angle_l, angle_r = angle_r) - V[_])/np.max(V)  +                ...
        ↪  0.05*np.linalg.norm(p - p_0)

    res = minimize(objective, p_0 , bounds=bounds, method='L-BFGS-B',  tol=1e-12 )
    angle_l_opt, angle_r_opt =  res.x

    p_0 = [angle_l_opt, angle_r_opt]
    ANGLE_L[_] = angle_l_opt
    ANGLE_R[_] = angle_r_opt

    V_opt = V_mv(step = _ , angle_l = angle_l_opt, angle_r = angle_r_opt)
    V_OPT[_] = V_opt

    print(f"Closest point to f0 = {V[_]}:")
    print(f"angle_l = {angle_l_opt:.4f}, angle_r = {angle_r_opt:.4f}, f(x0, y0) = {V_opt:.4f}")
    print(f"Absolute error = {abs(V_opt - V[_]):.4e}")

ANGLE_L[:np.argmin(area_mod)+1] = angle_off_set_l
```

```
ANGLE_R[:np.argmin(area_mod)+1] = angle_off_set_r
V_OPT[:np.argmin(area_mod)+1]  = V_0

```

Listing 10 Problème Inverse - Optimisation des angles

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
import pandas as pd
from scipy.spatial import cKDTree, Delaunay
from scipy.interpolate import splev, splrep
import os
import glob
import time as timer
from skspatial.objects import Line
from scipy.interpolate import griddata, LinearNDInterpolator, interp1d
from shapely.geometry import Polygon, Point
from scipy.signal import argrelextrema
from scipy.interpolate import SmoothBivariateSpline, RectBivariateSpline
from pint import Quantity
import shutil
from sklearn.metrics import mean_squared_error
from scipy.integrate import simpson
import math
from sklearn.metrics import mean_squared_error
from scipy.sparse import diags, kron, identity, linalg

workingFolder = 'Y:\\Recherche\\DEV\\PyAnsys\\case4\\'
resultFolder = workingFolder + 'results\\'

def load_points(motionPath):

    data = np.loadtxt(motionPath, delimiter='\t')
    nb_time_step_tt, total_columns = data[1:,:].shape
    nb_points = total_columns // 2
    curve_mesh = data[1:,:].reshape(nb_time_step_tt, nb_points, 2)

    return curve_mesh

path_in = workingFolder + 'data\\' + 'fluent\\LV_IN_motion.dat'
path_out = workingFolder + 'data\\' + 'fluent\\LV_OUT_motion.dat'

parts = ['MV','AV','LVW_in','LVW_out','MVadd_in','MVadd_out','AVadd_in','AVadd_out']

coord_parts = {}

for _ in range(len(parts)):
    x = np.loadtxt(f'Y:\\Recherche\\DEV\\PyAnsys\\case4\\data\\geom\\xcoord_{parts[_]}.csv',                ...
    ↪  delimiter=',',skiprows=2)/1000
    y = np.loadtxt(f'Y:\\Recherche\\DEV\\PyAnsys\\case4\\data\\geom\\ycoord_{parts[_]}.csv',                ...
    ↪  delimiter=',',skiprows=2)/1000
```

```
45      coord_parts[parts[_]] = [x[:,:], y[:,:]]### x(t,p); y(t,p)
46
47
48  curve_mesh = np.concatenate((load_points(path_in)[:,::-1,:], load_points(path_out)), axis=1)
49  nb_time_step = x.shape[0]
50  dt = 0.000943
51  T = dt*910
52  t = np.array([k*dt for k in range(2*nb_time_step)])
53
54  area= np.zeros(nb_time_step)
55  for time_step in range(nb_time_step):
56      pgon = Polygon(curve_mesh[time_step])
57      area[time_step] = pgon.area
58
59
60  area_diff = np.zeros_like(area)
61  area_diff = np.diff(area,axis=0, n=1, append=area[0])
62  area_diff = area_diff/dt
63
64  spl_area = splrep(np.array([k*dt for k in range(2*nb_time_step)]),np.concatenate((area,area)),k=5, s=5e-10)
65
66  area_splined = splev(t, spl_area)
67  area_splined_diff = np.diff(area_splined,axis=0, n=1, append=area_splined[0])/dt
68  area_splined_acc = np.diff(area_splined_diff,axis=0, n=1, append=area_splined_diff[0])
69
70  local_extreme_max = argrelextrema(data=np.abs(area_diff), comparator=np.greater, order=100)[0].astype(int)
71
72  local_extreme_min = argrelextrema(data=np.abs(area_diff), comparator=np.less, order=100)[0].astype(int)
73
74  i_dia = np.argmin(area)
75
76
77  t_mid_systole, t_E_wave, t_A_wave = t[local_extreme_max]
78  t_start_diastole, t_diastasis = t[local_extreme_min]
79  t_end_diastole = t[nb_time_step-1]
80  id_ref = [local_extreme_max[0], local_extreme_min[0], local_extreme_max[1],local_extreme_min[1],          ...
    ↪  local_extreme_max[2], nb_time_step -1 ]
81  t_ref = [t_mid_systole, t_start_diastole, t_E_wave, t_diastasis, t_A_wave, t_end_diastole]
82  moment_ref = ['mid_systole', 'start_diastole', 'E_wave', 'diastasis', 'A_wave', 'end_diastole']
83
84  # Create a regular mesh grid
85  d = 1e-4  # Define grid spacing
86
87
88  x_min, x_max = np.min(curve_mesh[:,:,0]), np.max(curve_mesh[:,:,0])
89
90  x_c, x_d = (x_min + x_max)/2, (x_max - x_min)/2
91  x_min, x_max = x_c - 1.05*x_d, x_c + 1.05*x_d
92  print(x_min, x_max)
93  y_min, y_max = np.min(curve_mesh[:,:,1]), np.max(curve_mesh[:,:,1])
94
95  y_c, y_d = (y_min + y_max)/2, (y_max - y_min)/2
96  y_min, y_max = y_c - 1.05*y_d, y_c + 1.15*y_d
97  print(y_min, y_max)
98
```

```
x_grid, y_grid = np.meshgrid(np.arange(x_min, x_max, d), np.arange(y_min, y_max, d))

case = ['check']
dt = [0.000943]
dict_case={}
for _ in range(len(case)):
    files= np.array(sorted(glob.glob(resultFolder + case[_] + "\\velocity_csv\\fluent_export_csv-*")))


    t = np.array([dt[_]*k for k in range(files.shape[0])])

    idx = [np.abs(t - time).argmin() for time in t_ref]

    # Initialize dictionaries
    coords_dict = {}
    velocity_dict = {}
    velocity_interp = {}

    # Loop through files
    for time, file in enumerate(files):
        print(time/files.shape[0]*100)

        # Read CSV file
        df = pd.read_csv(file, skipinitialspace=True)

        # Store coordinate data
        coords_dict[time] = df[["x-coordinate", "y-coordinate"]].to_numpy()

        # Store vector data
        velocity_dict[time] = df[["x-velocity", "y-velocity", "velocity-magnitude"]].to_numpy()

        x_vel = griddata(coords_dict[time], velocity_dict[time][:,0], (x_grid, y_grid),
        ↪  method='linear',fill_value=0)
        y_vel = griddata(coords_dict[time], velocity_dict[time][:,1], (x_grid, y_grid),
        ↪  method='linear',fill_value=0)
        magn_vel = griddata(coords_dict[time], velocity_dict[time][:,2], (x_grid, y_grid),
        ↪  method='linear',fill_value=0)

        velocity_interp[time] = {"x-velocity":x_vel , "y-velocity":y_vel, "velocity-magnitude":magn_vel}

    dict_case[case[_]] = {'coords': coords_dict, 'velocity':velocity_dict, 'velocity_grid':velocity_interp}


n = 50

mv_addin = np.column_stack((np.linspace(coord_parts['MVadd_in'][0][0,0], coord_parts['MVadd_in'][0][0,1],
↪  n), np.linspace(coord_parts['MVadd_in'][1][0,0], coord_parts['MVadd_in'][1][0,1], n)))
mv_addout = np.column_stack((np.linspace(coord_parts['MVadd_out'][0][0,0],
↪  coord_parts['MVadd_out'][0][0,1], n), np.linspace(coord_parts['MVadd_out'][1][0,0],
↪  coord_parts['MVadd_out'][1][0,1], n)))
av_red = np.column_stack((np.linspace(coord_parts['AVadd_in'][0][0, 0], coord_parts['AVadd_out'][0][0, 1],
↪  n), np.linspace(coord_parts['AVadd_in'][1][0, 0], coord_parts['AVadd_out'][1][0, 1], n)))
mv  = np.column_stack((coord_parts['MV'][0][0,:], coord_parts['MV'][1][0,:]))
lv_in = load_points(path_in)
lv_out = load_points(path_out)
```

```
147
148
149 period_off_set = 2
150
151 id_end_backward, id_init, id_end_forward = nb_time_step*(period_off_set-1) + i_dia,                          ...
    ↪  nb_time_step*period_off_set, nb_time_step*period_off_set + i_dia
152
153 per = 0.1
154 id_red = np.ix_(np.linspace(0, x_grid.shape[0] - 1, int(per*x_grid.shape[0]) , dtype=int),np.linspace(0,     ...
    ↪  x_grid.shape[1] - 1, int(per*x_grid.shape[1]) , dtype=int))
155 x_red  = x_grid[id_red]
156 y_red  = y_grid[id_red]
157
158 mask_grid = np.zeros_like(x_red).astype(bool)
159 # Create the polygon from the closed curve
160
161 curve_mesh = np.concatenate((mv_addin,lv_in[id_init,::-1,:], av_red, lv_out[id_init,:,:],mv_addout, mv ),    ...
    ↪  axis=0)
162
163 polygon = Polygon(curve_mesh)
164
165 # Check if each point in the mesh is inside the polygon
166 for i in range(x_red.shape[0]):
167     for j in range(x_red.shape[1]):
168         point = Point(x_red[i, j], y_red[i, j])
169         dist  = polygon.exterior.distance(point)
170         if (dist>= 5*d) and (polygon.contains(point)):  # Check if point is inside the polygon
171             mask_grid[i, j] = 1
172
173 x_init = x_red[mask_grid]
174 y_init = y_red[mask_grid]
175
176 #prepare forward
177
178 points_2d = np.column_stack((x_grid.flatten(), y_grid.flatten()))
179 tri = Delaunay(points_2d)
180
181 start = timer.time()
182
183 INTERP_f = []
184 for _ in range(id_init, id_end_forward + 3):
185     vel_flat = np.column_stack((
186         dict_case[case[0]]['velocity_grid'][_]['x-velocity'].flatten(),
187         dict_case[case[0]]['velocity_grid'][_]['y-velocity'].flatten()
188     ))
189     INTERP_f.append(LinearNDInterpolator(tri, vel_flat, fill_value=0))
190
191 print(f"Precomputation interpolators took {timer.time() - start:.2f} seconds")
192
193 start = timer.time()
194
195 GEO_f = np.array([np.concatenate((mv_addin,lv_in[_,::-1,:], av_red, lv_out[_,:,:],mv_addout, mv ), axis=0)   ...
    ↪  for _ in range(id_init, id_end_forward + 3 )])
196
197
```

```
198 print(f"Precomputation geometries took {timer.time() - start:.2f} seconds")
199
200 start = timer.time()
201 POLYGON_f = [Polygon(GEO_f[_,:,:]) for _ in range(GEO_f.shape[0])]
202
203 print(f"Precomputation polygons took {timer.time() - start:.2f} seconds")
204
205 start = timer.time()
206
207 TREE_f = [cKDTree(GEO_f[_,:,:]) for _ in range(GEO_f.shape[0])]
208
209 print(f"Precomputation trees took {timer.time() - start:.2f} seconds")
210
211 INTERP_b = []
212 for _ in range(id_init, id_end_backward - 3,-1):
213     vel_flat = np.column_stack((
214         -dict_case[case[0]]['velocity_grid'][_]['x-velocity'].flatten(),
215         -dict_case[case[0]]['velocity_grid'][_]['y-velocity'].flatten()
216     ))
217     INTERP_b.append(LinearNDInterpolator(tri, vel_flat, fill_value=0))
218
219 print(f"Precomputation interpolators took {timer.time() - start:.2f} seconds")
220
221 start = timer.time()
222
223 GEO_b = np.array([np.concatenate((mv_addin,lv_in[_,::-1,:], av_red, lv_out[_,:,:],mv_addout, mv ), axis=0)   ...
    ↪  for _ in range(id_init, id_end_backward - 3, -1 )])
224
225
226 print(f"Precomputation geometries took {timer.time() - start:.2f} seconds")
227
228 start = timer.time()
229 POLYGON_b = [Polygon(GEO_b[_,:,:]) for _ in range(GEO_b.shape[0])]
230
231 print(f"Precomputation polygons took {timer.time() - start:.2f} seconds")
232
233 start = timer.time()
234
235 TREE_b = [cKDTree(GEO_b[_,:,:]) for _ in range(GEO_b.shape[0])]
236
237 print(f"Precomputation trees took {timer.time() - start:.2f} seconds")
238
239
240 def forward(p, time_step,  INTERP, dt):
241
242     interp_n, interp_np1, interp_np2 = INTERP[time_step], INTERP[time_step+1], INTERP[time_step+2]
243
244     k1 = interp_n(p)
245     k2 = interp_np1(p + dt*k1)
246     k3 = interp_np1(p + dt*k2)
247     k4 = interp_np2(p+ 2*dt*k3)
248
249     return p + dt*(k1 + 2*k2 + 2*k3 + k4)/3
250
251 def backward(p, time_step,  INTERP, dt):
```

```
252
253     interp_n, interp_np1, interp_np2 = INTERP[time_step], INTERP[time_step+1], INTERP[time_step+2]
254
255     k1 = interp_n(p)
256     k2 = interp_np1(p + dt*k1)
257     k3 = interp_np1(p + dt*k2)
258     k4 = interp_np2(p+ 2*dt*k3)
259
260     return p + dt*(k1 + 2*k2 + 2*k3 + k4)/3
261
262
263 # forward
264 p_f = []
265 p_f.append(p_init)
266
267 p0 = p_init
268 step  = id_init
269 it = 0
270 while step  <= id_end_forward :
271
272     print(f'iteration {it}')
273     p1 = forward(p=p0, time_step= it,  INTERP= INTERP_f, dt=dt[0])
274
275     curve_mesh = GEO_f[it + 2]
276     polygon = POLYGON_f[it + 2]
277     tree = TREE_f[it + 2]
278
279     p1_star = p1.copy()
280
281     for i in range(p1.shape[0]):
282         point = Point(p1[i,:])
283
284         if not (polygon.contains(point)):
285
286             tree = cKDTree(curve_mesh[:,:])
287             closest_distances, closest_indices = tree.query(p1[i,:], k=1)  # Find closest points
288             pt = curve_mesh[closest_indices,:]
289
290             p1_star[i,:] = pt
291
292
293     p_f.append(p1_star)
294
295     step+=2
296     it+=2
297     p0 = p1_star
298
299 p_f = np.array(p_f)
300
301 p_f_mod = p_f[:-1,:,:].copy()
302 out_wall = 0
303 out_av = 0
304 statut_f = np.ones_like(p_f)
305
306 dst_av = np.nan
```

```
for i in range(p_f.shape[0]):
    for j in range(p_f.shape[1]):
        if np.isnan(dst_av): dst_av = np.min(np.linalg.norm(av_red - p_f[i,j,:], axis=1))
        else: dst_av = min(dst_av, np.min(np.linalg.norm(av_red - p_f[i,j,:], axis=1)))
print(dst_av)


for j in range(p_f.shape[1]):
    it = 0
    for i in range(p_f.shape[0]-1):

        if np.min(np.linalg.norm(av_red - p_f[i,j,:], axis=1)) <= 10*dst_av:
            p_f_mod[i:,j] = np.nan
            out_av +=1
            statut_f[i:,j] = 2
            break

        polygon = POLYGON_f[it + 2]
        point = Point(p_f[i,j,:])

        it+=2

print((out_av, p_f.shape[1]))

# backward
p_b = []
p_b.append(p_init)

p0 = p_init

step  = id_init
it = 0
while step  >= id_end_backward :

    print(f'iteration {step}')
    p1 = backward(p=p0, time_step= it,  INTERP= INTERP_b, dt=dt)

    curve_mesh = GEO_b[it + 2]
    polygon = POLYGON_b[it + 2]
    tree = TREE_b[it + 2]

    p1_star = p1.copy()
    count=0
    for i in range(p1.shape[0]):

        point = Point(p1[i,:])

        if not (polygon.contains(point)):
            count+=1
            tree = cKDTree(curve_mesh[:,:])
            closest_distances, closest_indices = tree.query(p1[i,:], k=1)  # Find closest points
            pt = curve_mesh[closest_indices,:]

            p1_star[i,:] = pt
    print(count)
```

```
    p_b.append(p1_star)


    step-=2
    it+=2
    p0 = p1_star

p_b = np.array(p_b[1:])


p_b_mod = p_b[:-1,:,:].copy()
out_wall = 0
out_mv = 0
statut_b = np.ones_like(p_b)

dst_mv = np.nan
for i in range(p_b.shape[0]):
    for j in range(p_b.shape[1]):
        if np.isnan(dst_mv): dst_mv = np.min(np.linalg.norm(mv - p_b[i,j,:], axis=1))
        else: dst_mv = min(dst_mv, np.min(np.linalg.norm(mv - p_b[i,j,:], axis=1)))



for j in range(p_b.shape[1]):
    it = 0
    for i in range(p_b.shape[0]-1):

        if np.min(np.linalg.norm(mv - p_b[i,j,:], axis=1)) <= d:
            p_b_mod[i:,j] = np.nan
            out_mv +=1
            statut_b[i:,j] = 3
            break

        polygon = POLYGON_b[it + 2]
        point = Point(p_b[i,j,:])


        it+=2

print((out_mv, p_b.shape[1]))

p = np.concatenate((p_b_mod[::-1,:,:], p_f_mod[1:,:,:]), axis=0)
statut = np.concatenate((statut_b[::-1,:,:], statut_f[1:,:,:]), axis=0)
GEO = np.concatenate((GEO_b[::-1,:,:], GEO_f[1:,:,:]), axis=0)

cat = np.ones(p.shape[1])
for i in range(p.shape[1]):
    if np.any(statut[:,i] == 2) and np.any(statut[:,i] == 3):
        cat[i] = 1 # in and out : direct flow
    if np.any(statut[:,i] == 2) and (not np.any(statut[:,i] == 3)):
        cat[i] = 2 # out and not in : delayed ejection flow
    if (not np.any(statut[:,i] == 2)) and (not np.any(statut[:,i] == 3)):
        cat[i] = 3 # not out and not in : residual volume
    if (not np.any(statut[:,i] == 2)) and (np.any(statut[:,i] == 3)):
        cat[i] = 4 # not out and in : retained inflow
```

```
417
418 print(f'direct flow :{np.sum(cat == 1)/cat.shape[0]*100}\n delayed ejection flow :{np.sum(cat ==  ...
    ↪  2)/cat.shape[0]*100}\n residual volume :{np.sum(cat == 3)/cat.shape[0]*100}\n retained inflow  ...
    ↪  :{np.sum(cat == 4)/cat.shape[0]*100}\n ')
419
420 fig, ax = plt.subplots(figsize=(16, 12))
421
422 curve_mesh = np.concatenate((mv_addin,lv_in[id_init,::-1,:], av_red, lv_out[id_init,:,:],mv_addout, mv ),  ...
    ↪  axis=0)
423
424
425 ax.plot(curve_mesh[:,0],curve_mesh[:,1], color = 'k')
426
427 ax.scatter(p_init[cat == 1,0], p_init[cat == 1,1], color='r', s = 1,label = 'Direct Flow')
428 ax.scatter(p_init[cat == 2,0], p_init[cat == 2,1], color='g', s = 1, label = 'Delayed Ejection Flow')
429 ax.scatter(p_init[cat == 3,0], p_init[cat == 3,1], color='b', s = 1, label = 'Residual Volume')
430 ax.scatter(p_init[cat == 4,0], p_init[cat == 4,1], color='y', s = 1, label = 'Retained Inflow')
431
432 plt.axis("equal")
433 plt.axis('off')
434 plt.legend(loc='best')
435 plt.savefig('Y:\\Recherche\\DEV\PyAnsys\\\case4\\results\\'+ case[0] + '\\ptt\\categorical_mapping.png',  ...
    ↪  dpi=300, format='png')
436 plt.show()
```

Listing 11 Intégration des trajectoires et catégorisation de l'écoulement

---

```
1  ########## Interpolation of the EchoPAC segmentation
2  import numpy as np
3  import matplotlib.pyplot as plt
4  from scipy.interpolate import SmoothBivariateSpline
5  from matplotlib.animation import FuncAnimation
6  import pandas as pd
7  import glob
8  import cv2 as cv
9  import skimage as skifrom
10 from shapely.geometry import Polygon
11 from scipy.spatial import cKDTree
12 from scipy.interpolate import griddata, LinearNDInterpolator, interp1d, CubicSpline, SmoothBivariateSpline,  ...
   ↪  BarycentricInterpolator, bisplrep, bisplev, make_interp_spline, RectBivariateSpline
13 import scipy.interpolate as spi
14 from scipy.interpolate import splrep, splev
15 from scipy.signal import argrelextrema
16 from shapely.geometry import Polygon, Point
17 import matplotlib
18
19 patient = '.\\patient_06_1'
20
21 # Initialize dictionary
22 data = {}
23
```

```
# Get all CSV files in the directory
csv_files = glob.glob(patient + "\*.csv")

for file in csv_files:
    with open(file, 'r') as f:
        first_line = f.readline()
        header_values = first_line.strip().split(',')

    variable = header_values[0]
    print(variable)
    dim1,dim2,dim3 = int(header_values[1]),int(header_values[2]),int(header_values[3])
    print(dim1,dim2,dim3)
    # Load the CSV file
    df = pd.read_csv(file,sep=',', skiprows = 1,  header=None, skip_blank_lines=True, lineterminator='\n')

    # Convert the DataFrame to a NumPy array
    value = df.values
    value[:,0]
    value = value.reshape(dim3, dim1, dim2).transpose(1, 2, 0)

    data[variable] = value

np.save(patient +   "\\patient_06_1.npy", data)

data = np.load(patient +  "\\patient_06_1.npy", allow_pickle=True).item()

wall = data['wall'].astype(bool)
lv_seg = data['lv_seg'].astype(bool)
x = data['grid_X'][:,:,0]
x_tissue = data['X'][:,:,0]
y = data['grid_Z'][:,:,0]
y_tissue = data['Z'][:,:,0]
tissue = data["Value"]
v_doppler = data['uD']
t = data['t'].flatten()
t = t - t[0]

#load and order initial segmentation

seg = []

for _ in range(t.shape[0]):

    mesh = np.column_stack((x[wall[:,:,_]],y[wall[:,:,_]]))
    mesh = mesh[mesh[:, 1].argsort()]
    mesh = mesh[::-1,:]
    points = mesh
    idx = []
    ordered_points = []
    # Start with the extremity
    start_idx = 0
    ordered_points.append(points[start_idx,:])
    remaining_points = np.delete(points, start_idx, axis=0)
    while len(remaining_points) > 0:
        # Find the closest point to the last point in the ordered list
```

```
79          tree = cKDTree(remaining_points)
80          search_points = ordered_points[-1]
81          closest_distances, closest_indices = tree.query(search_points, k=1)  # Find closest points
82          closest_point = remaining_points[closest_indices,:]
83          ordered_points.append(closest_point)
84          remaining_points = np.delete(remaining_points, closest_indices, axis=0)
85
86      ordered_points = np.array(ordered_points)
87
88      indexes = [np.where((points == row).all(axis=1))[0][0] for row in ordered_points]
89
90      mesh = mesh[indexes,:]
91
92      seg.append(mesh)
93
94  nb_nodes = 100
95  nb_time_step = 1000
96  period = 1
97
98  area= np.zeros(t.shape[0])
99  for _ in range(t.shape[0]):
100     pgon = Polygon(seg[_])
101     area[_] = pgon.area
102
103 #locate and remove anormalities
104
105 local_extreme_min = argrelextrema(data=area, comparator=np.less, order=3)[0]
106 remove_index = local_extreme_min.shape[0] > 1
107 if remove_index:
108     mask = np.ones_like(area)
109     mask[local_extreme_min[1:]]=0
110     mask = mask.astype(bool)
111
112 seg_mod = []
113 for _ in range(mask.shape[0]):
114     if mask[_]:
115         seg_mod.append(seg[_])
116 seg_mod.append(seg[0])
117
118 t_mod = np.append(t[mask][:],t[-1]+t[1]-t[0])
119
120 s_grid = np.linspace(0,1,nb_nodes)
121 t_grid = np.linspace(t_mod[0],3*t_mod[-1],3*nb_time_step)
122
123 #interpolate in time and space
124
125 S_list = []
126 for time_step in range(t_mod.shape[0]):
127     S=[]
128     s=0
129     S.append(0)
130     for pts in range(seg_mod[time_step].shape[0] - 1):
131         dx = seg_mod[time_step][pts+1,0]- seg_mod[time_step][pts,0]
132         dy = seg_mod[time_step][pts+1,1]- seg_mod[time_step][pts,1]
133         s+= np.sqrt(dx**2 +dy**2)
```

```
            S.append(s)
        S=np.array(S)
        S = S/S[-1]
        S_list.append(S)

points  = []
f = []

count =0
for _ in range(3):
    if _==0:
        for time_step in range(t_mod.shape[0]):
            for pts in range(seg_mod[time_step].shape[0]):
                points.append([S_list[time_step][pts],t_mod[time_step]])
                f.append([seg_mod[time_step][pts,0],seg_mod[time_step][pts,1]])

                count+=1
    else:
        for time_step in range(1,t_mod.shape[0]):
            for pts in range(seg_mod[time_step].shape[0]):
                points.append([S_list[time_step][pts],t_mod[time_step]+_*t_mod[-1]])
                f.append([seg_mod[time_step][pts,0],seg_mod[time_step][pts,1]])

                count+=1

f= np.array(f)
points = np.array(points)

#interp_x = LinearNDInterpolator(points=points, values=f[:,0])
#interp_y = LinearNDInterpolator(points=points, values=f[:,1])

grid_s, grid_t = np.meshgrid(s_grid,t_grid)

interp_x = SmoothBivariateSpline(points[:,0],points[:,1], f[:,0], kx=3, ky=5, s=0.002425)#check around 0.0025
interp_y = SmoothBivariateSpline(points[:,0],points[:,1], f[:,1], kx=3, ky=5, s=0.002425)

grid_x = interp_x(x=grid_s.flatten(order='C'), y=grid_t.flatten(order='C'), grid=False)
grid_y = interp_y(x=grid_s.flatten(order='C'), y=grid_t.flatten(order='C'), grid=False)

grid_x = grid_x.reshape(grid_s.shape,order='C')
grid_y = grid_y.reshape(grid_s.shape,order='C')

area_select = np.zeros(grid_x.shape[0])

for _ in range(grid_x.shape[0]):
    pgon = Polygon(np.column_stack((grid_x[_,:],grid_y[_,:])))
    area_select[_] = pgon.area

area_diff = np.diff(area_select, append=area_select[0])

i_0 = argrelextrema(data=area_select, comparator=np.greater, order=200)[0][1]

mask_diff = (area_diff <= 0)
mask_period = np.zeros_like(mask_diff)
mask_period[ i_0 - int(1*nb_time_step): i_0] = 1
```

```
mask_sup = (area_select - area_select[i_0]) >=0

if np.any(mask_diff*mask_period*mask_sup):
    i_1 = np.abs(area_select[mask_diff*mask_period*mask_sup] - area_select[i_0]).argmin() + np.argmax(mask_diff*mask_period*mask_sup)

    mesh = np.zeros((i_0-i_1, nb_nodes, 2))
    mesh[:,:,0] = grid_x[i_1:i_0,:]
    mesh[:,:,1] = grid_y[i_1:i_0,:]

else:
    i_0 = argrelextrema(data=area_select, comparator=np.greater, order=200)[0][0]
    mask_diff = (area_diff >= 0)
    mask_period = np.zeros_like(mask_diff)
    mask_period[ i_0: i_0 + nb_time_step] = 1
    mask_sup = (area_select - area_select[i_0]) <=0

    i_1 = np.abs(area_select[mask_diff*mask_period*mask_sup] - area_select[i_0]).argmin() + np.argmax(mask_diff*mask_period*mask_sup)

    mesh = np.zeros((i_1-i_0, nb_nodes, 2))
    mesh[:,:,0] = grid_x[i_0:i_1,:]
    mesh[:,:,1] = grid_y[i_0:i_1,:]


t_0 = np.linspace(t_mod[0],t_mod[-1],mesh.shape[0])

t_new =  np.linspace(t_mod[0],t_mod[-1],nb_time_step)
s_new = s_grid


interp_x = RectBivariateSpline(t_0, s_grid, mesh[:, :, 0], kx=5, ky=4)
interp_y = RectBivariateSpline(t_0, s_grid, mesh[:, :, 1], kx=5, ky=4)

grid_x = interp_x(t_new, s_new)
grid_y = interp_y(t_new, s_new)

mesh = np.zeros((nb_time_step, nb_nodes, 2))
mesh[:,:,0] = grid_x
mesh[:,:,1] = grid_y

########## Extraction of the upper part from mode B ECG

import cv2
import numpy as np
import matplotlib.pyplot as plt
import skimage as ski
from skimage.restoration import denoise_nl_means, estimate_sigma, rolling_ball
from skimage.exposure import equalize_hist
from scipy.ndimage import distance_transform_edt
from skimage.segmentation import watershed, chan_vese, morphological_chan_vese, morphological_geodesic_active_contour
from skimage.filters import sobel, try_all_threshold, threshold_isodata, threshold_li, threshold_local, threshold_mean, threshold_minimum, threshold_multiotsu, threshold_niblack, threshold_otsu, threshold_sauvola, threshold_triangle, threshold_yen
```

```
from skimage.morphology import flood
from skimage.feature import peak_local_max
from fil_finder import FilFinder2D, Filament2D
import astropy.units as u

# interpolation of B mode image on regular grid of resolution .1mm
d = 1e-4  # Define grid spacing

x_min, x_max = np.min(x_tissue), np.max(x_tissue)

x_c, x_d = (x_min + x_max)/2, (x_max - x_min)/2
x_min, x_max = x_c - 1.05*x_d, x_c + 1.05*x_d
print(x_min, x_max)
y_min, y_max = np.min(y_tissue), np.max(y_tissue)

y_c, y_d = (y_min + y_max)/2, (y_max - y_min)/2
y_min, y_max = y_c - 1.05*y_d, y_c + 1.05*y_d
print(y_min, y_max)

x_grid, y_grid = np.meshgrid(np.arange(x_min, x_max, d), np.arange(y_min, y_max, d))


image = griddata(np.column_stack((x_tissue.flatten(),y_tissue.flatten())), tissue[:,:,i_systole].flatten(),  ...
↪  (x_grid, y_grid), method='linear', fill_value=0 )/255
plt.pcolormesh(x_grid, y_grid, image, shading='gouraud', cmap='gray')
plt.show()

#thickening of the interpolated segmentation

thickness = 5e-3
dp = np.gradient(seg_interp[i_systole][:,:], axis=0)
normals = -np.stack([-dp[:, 1], dp[:, 0]], axis=1)
norms = np.linalg.norm(normals, axis=1, keepdims=True)
normals /= norms
thickened_points = seg_interp[i_systole][:,:] + thickness * normals
points = np.concatenate((seg_interp[i_systole][:,:],thickened_points[::-1,:],[seg_interp[i_systole][0,:]]))

mask_grid = np.zeros_like(x_grid).astype(int)
polygon = Polygon(points)

for i in range(x_grid.shape[0]):
    for j in range(x_grid.shape[1]):
        point = Point(x_grid[i, j], y_grid[i, j])
        if polygon.contains(point):  # Check if point is inside the polygon
            mask_grid[i, j] = 1

mask_thickness = mask_grid.astype(bool)

# first filtering
thresh = image.copy()
thresh[image <=  0.2] = 0
equalized = ski.exposure.rescale_intensity(thresh,in_range = (0,1))
filtered_image = ski.filters.butterworth(equalized, cutoff_frequency_ratio=0.5, high_pass=False, order=2.0,  ...
↪  squared_butterworth=True, npad=0)
filtered_image = ski.exposure.rescale_intensity(filtered_image, in_range = (0,1))
```

```
filtered_image = ski.exposure.adjust_sigmoid(filtered_image, cutoff=0.5, gain = 10, inv=False)

plt.pcolormesh(x_grid, y_grid, filtered_image , shading='gouraud', cmap='gray')
plt.show()

# second filtering + adding the thickened ventricular wall

filtered_image_bis = filtered_image.copy()
equalized = ski.exposure.rescale_intensity(filtered_image_bis,in_range = (0,1))
filtered_image_bis = ski.filters.butterworth(equalized, cutoff_frequency_ratio=0.5, high_pass=False,          ...
↪  order=2.0, squared_butterworth=True, npad=0)
filtered_image_bis = ski.exposure.rescale_intensity(filtered_image_bis, in_range = (0,1))
filtered_image_rd = ski.exposure.adjust_sigmoid(filtered_image_bis, cutoff=0.5, gain = 10, inv=False)
filtered_image_bis = ski.exposure.adjust_sigmoid(filtered_image_bis, cutoff=0.15, gain = 10, inv=False)

barriere_image = filtered_image_bis.copy()

barriere_image[mask_thickness] = 1
filtered_image_rd[mask_thickness] = 1

plt.pcolormesh(x_grid, y_grid, barriere_image , shading='gouraud', cmap='gray')
plt.show()

plt.pcolormesh(x_grid, y_grid, filtered_image_rd, shading='gouraud', cmap='gray')
plt.show()

# first gross segmentation using flood on bright region

max_idx = np.unravel_index(np.argmax(barriere_image), filtered_image_bis.shape)
init_pt = (max_idx[0],max_idx[1])

disk_mask = np.zeros_like(barriere_image).astype(bool)

centre = [int(disk_mask.shape[0]*0.55),int(disk_mask.shape[1]*0.5)]
R = 0.75*min(disk_mask.shape[0],disk_mask.shape[1])/2

for i in range(disk_mask.shape[0]):
    for j in range(disk_mask.shape[1]):
        if ((i - centre[0])**2 + (j - centre[1])**2)<=R**2:
            disk_mask[i,j]= True

segmentation = flood(barriere_image, init_pt, connectivity=10, tolerance=0.4)*disk_mask
morphological_filtered = ski.morphology.binary_opening(segmentation, ski.morphology.disk(10))
morphological_filtered = ski.morphology.binary_closing(segmentation, ski.morphology.disk(20))
edge = ski.segmentation.find_boundaries(morphological_filtered,mode='thick', connectivity=10).astype(bool)

plt.pcolormesh(x_grid, y_grid, barriere_image , shading='gouraud', cmap='gray')
plt.scatter(x_grid[morphological_filtered], y_grid[morphological_filtered], s=2, color='b')
plt.scatter(x_grid[segmentation], y_grid[segmentation], s=2, color='r')
plt.scatter(x_grid[edge], y_grid[edge], s=2, color='g')
plt.scatter(x_grid[init_pt[0],init_pt[1]], y_grid[init_pt[0],init_pt[1]], s=20, color='k')
plt.axis('equal')
plt.show()

init_level_set = morphological_filtered
```

```
346
347 #Raffinement of the segmented region: filtred with inverse_gaussian_gradient and newly segmented using ...
    ↪ morphological active contour
348 filtered_image_rd = ski.segmentation.inverse_gaussian_gradient(filtered_image_rd, alpha=50.0, sigma=15.0)
349
350 segmentation = morphological_geodesic_active_contour(gimage = filtered_image_rd, num_iter=100, ...
    ↪ init_level_set= init_level_set, smoothing=4, threshold=0.75, balloon=0)
351
352 #Extraction of the skeleton of the segmented area
353
354 skeleton = ski.morphology.skeletonize(segmentation, method='lee')
355 fig, axes = plt.subplots(1, 2, figsize=(8, 8))
356
357 axes[0].pcolormesh(x_grid, y_grid, filtered_image_rd, shading='gouraud', cmap='gray')
358 axes[0].axis('equal')
359 axes[0].set_axis_off()
360
361 axes[1].imshow(barriere_image[::-1,:],cmap ='gray')
362 axes[1].contour(segmentation[::-1,:] , [0.5], colors='r')
363 axes[1].contour(skeleton[::-1,:] , [0.5], colors='g')
364 axes[1].axis('equal')
365 axes[1].set_axis_off()
366
367 fig.tight_layout()
368 plt.show()
369
370 #Cleaning of the skeleton
371
372 fil = FilFinder2D(image=barriere_image, beamwidth= 10*u.pix,ang_scale=1 * u.deg, distance=None, ...
    ↪ mask=segmentation)
373 fil.preprocess_image(flatten_percent=100)
374 fil.create_mask(use_existing_mask=True)
375 fil.medskel(verbose=False)
376 fil.analyze_skeletons(prune_criteria='length',relintens_thresh=0.8, branch_thresh=200* u.pix, ...
    ↪ skel_thresh=10 * u.pix, max_prune_iter=50, verbose=False)
377
378 branches = fil.filaments[0].branch_pts()
379 branch_lenghts = fil.branch_lengths()
380 centre_skeleton_x, centre_skeleton_y  = np.mean(np.where(fil.skeleton)[0]),np.mean(np.where(fil.skeleton)[1])
381
382 index = np.where(branch_lenghts[0]>=200*u.pix)[0]
383 skeleton_cleaned = []
384 for i in index:
385     skeleton_cleaned.append(branches[i])
386
387 skeleton_cleaned = np.concatenate(skeleton_cleaned, axis=0)
388
389 centre_x, centre_y  = np.mean(skeleton_cleaned[:,0]),np.mean(skeleton_cleaned[:,1])
390 dpl = np.array([centre_skeleton_x - centre_x, centre_skeleton_y - centre_y]).astype(int)
391 print(dpl)
392
393 skeleton_cleaned = skeleton_cleaned + dpl
394
395 #Extraction of the upper part
396
```

```
397 def order(x,y,mask, invert=False):
398     mesh = np.column_stack((x[mask],y[mask]))
399     mesh = mesh[mesh[:, 1].argsort()]
400     mesh = mesh[::-1,:]
401     points = mesh
402     idx = []
403     ordered_points = []
404     # Start with the extremity
405     start_idx = 0
406     ordered_points.append(points[start_idx,:])
407     remaining_points = np.delete(points, start_idx, axis=0)
408     while len(remaining_points) > 0:
409         # Find the closest point to the last point in the ordered list
410         tree = cKDTree(remaining_points)
411         search_points = ordered_points[-1]
412         closest_distances, closest_indices = tree.query(search_points, k=1)  # Find closest points
413         closest_point = remaining_points[closest_indices,:]
414         ordered_points.append(closest_point)
415         remaining_points = np.delete(remaining_points, closest_indices, axis=0)
416 
417     ordered_points = np.array(ordered_points)
418 
419     indexes = [np.where((points == row).all(axis=1))[0][0] for row in ordered_points]
420 
421     mesh = mesh[indexes,:]
422     if invert: mesh = mesh[::-1,:]
423     return mesh
424 
425 wall_seg = griddata(np.column_stack((x.flatten(),y.flatten())), wall[:,:,i_systole].flatten(), (x_grid,      ...
    ↪  y_grid), method='nearest',fill_value=0)
426 wall_seg = wall_seg ==1
427 mesh_seg = order(x_grid,y_grid,wall_seg)
428 
429 segmentation_cleaned = np.zeros_like(filtered_image).astype(bool)
430 segmentation_cleaned[skeleton_cleaned[:,0],skeleton_cleaned[:,1]]=True
431 mesh_skeleton = order(x_grid,y_grid,segmentation_cleaned, invert=True)
432 
433 tree = cKDTree(mesh_skeleton)
434 search_points = seg_interp[i_systole][[0,-1],:]
435 
436 closest_distances, closest_indices = tree.query(search_points, k=1)  # Find closest points
437 closest_point = mesh_skeleton[closest_indices,:]
438 
439 plt.pcolormesh(x_grid, y_grid, filtered_image, shading='gouraud', cmap='gray')
440 plt.scatter(mesh_skeleton[:,0],mesh_skeleton[:,1],s=2,color='r')
441 plt.scatter(seg_interp[i_systole][:,0],seg_interp[i_systole][:,1],s=2,color='b')
442 plt.scatter(seg_interp[i_systole][-1,0],seg_interp[i_systole][-1,1],s=20,color='g')
443 plt.scatter(seg_interp[i_systole][0,0],seg_interp[i_systole][0,1],s=20,color='y')
444 plt.scatter(closest_point[1,0],closest_point[1,1],s=20,color='g')
445 plt.scatter(closest_point[0,0],closest_point[0,1],s=20,color='y')
446 
447 
448 plt.axis('off')
449 plt.axis('equal')
450 plt.savefig('Y:\\Recherche\\BDD\\iVFM\\clean_skeleton.png', dpi=200, format='png')
```

```
plt.show()

segment = np.delete(mesh_skeleton, np.arange(closest_indices[1], closest_indices[0]),axis=0)
segment = np.roll(segment, - (closest_indices[1]),axis=0)
segment = segment[::-1,:]

segment_index = np.zeros_like(segment).astype(int)
for _ in range(segment.shape[0]):
    segment_index[_,0] = np.argmin(np.abs(y_grid - segment[_,1]), axis=0)[0]
    segment_index[_,1] = np.argmin(np.abs(x_grid - segment[_,0]), axis=1)[0]


plt.pcolormesh(x_grid, y_grid, filtered_image, shading='gouraud', cmap='gray')
plt.scatter(segment[:,0],segment[:,1],s=20,color='g')
plt.plot(x_grid[segment_index[:,0],segment_index[:,1]],
↪  y_grid[segment_index[:,0],segment_index[:,1]],color='r')
plt.axis('off')
plt.axis('equal')
plt.show()

#Thresholding and detection of AV and MV

def angle_between_vectors(a, b):
    # Compute dot product
    dot_product = np.dot(a, b)
    # Compute norms (magnitudes)
    norm_a = np.linalg.norm(a)
    norm_b = np.linalg.norm(b)
    # Compute angle in radians
    angle_rad = np.arccos(dot_product / (norm_a * norm_b))
    # Convert to degrees
    return angle_rad

tau = segment[0,:] - segment[-1,:]
tau = -tau/np.linalg.norm(tau)
n = -np.array([-tau[1], tau[0]])
segment_n = np.dot(segment,n)
interp_func = splrep(range(segment_n.shape[0]), segment_n, s=1e-6, k=5)
segment_n_splined = splev(range(segment_n.shape[0]), interp_func)
segment_n_diff = np.diff(segment_n_splined, append=segment_n_splined[0])

local_extreme_max = argrelextrema(data=filtered_image[segment_index[:,0],segment_index[:,1]],
↪  comparator=np.greater, order=40)[0]
local_extreme_min = argrelextrema(data=filtered_image[segment_index[:,0],segment_index[:,1]],
↪  comparator=np.less, order=30)[0]

plt.plot(segment_n,label='n', color='k')

mask_av = np.abs(segment_n - np.max(segment_n))<=0.15*(np.max(segment_n) - np.min(segment_n))

plt.plot(np.where(mask_av)[0],segment_n[mask_av], color='r')

idx = np.abs(segment_n - segment_n[-1])

idx[segment_n_diff>=0] = None
```

```
idx = np.nanargmin(idx)

mask_mv = np.zeros_like(segment_n).astype(bool)
mask_mv[np.where(mask_av)[0][-1]:local_extreme_max[-1]]=True
plt.plot(np.where(mask_mv)[0],segment_n[mask_mv], color='g')
plt.show()

plt.plot(filtered_image[segment_index[:,0],segment_index[:,1]])
plt.scatter(local_extreme_max,filtered_image[segment_index[local_extreme_max,0],segment_index[local_extreme_max,1]],
↪  color='r')
plt.scatter(local_extreme_min,filtered_image[segment_index[local_extreme_min,0],segment_index[local_extreme_min,1]],
↪  color='g')
plt.scatter(np.argmin(segment_n),filtered_image[segment_index[np.argmin(segment_n),0],segment_index[np.argmin(segment_n),1]],
↪  color='k')

idx_md_lf = local_extreme_min[(local_extreme_min  - np.argmin(segment_n))>=0]
idx_md_lf = idx_md_lf[0]
plt.scatter(idx_md_lf,filtered_image[segment_index[idx_md_lf,0],segment_index[idx_md_lf,1]], color='y')
plt.show()

plt.pcolormesh(x_grid, y_grid, filtered_image, shading='gouraud', cmap='gray')
plt.plot(segment[mask_mv,0],segment[mask_mv,1],color='g')
plt.plot(segment[mask_av,0],segment[mask_av,1],color='r')
plt.scatter(mesh_seg[:,0],mesh_seg[:,1],s=2,color='b')

p_mid_lf = segment[np.argmin(segment_n),:]
p_mid_lf_vf = segment[idx_md_lf,:]
p_db_av = segment[np.where(mask_av)[0][0],:]
p_fn_av = segment[np.where(mask_av)[0][-1],:]
p_fn_mv = segment[np.where(mask_mv)[0][0],:]
p_db_mv = segment[np.where(mask_mv)[0][-1],:]

per=0.9
p_fn_mv = p_db_mv + per*(p_fn_mv - p_db_mv)
p_fn_av = p_db_av + per*(p_fn_av - p_db_av)

bosse_value = 3e-3
p_lvin_mid = (p_fn_mv + p_fn_av)/2 + bosse_value*(-n)

points = np.vstack((p_fn_av,p_lvin_mid,p_fn_mv))
s = [0,0.5,1]
cs_x = CubicSpline(s, points[:,0], bc_type='not-a-knot')
cs_y = CubicSpline(s, points[:,1], bc_type='not-a-knot')

s_interp = np.linspace(0,1,10)

lv_in = np.column_stack((cs_x(s_interp),cs_y(s_interp)))

closed_angle_r = angle_between_vectors(p_fn_mv - p_db_mv , p_mid_lf_vf - p_db_mv)
closed_angle_l = angle_between_vectors(p_db_mv - p_fn_mv , p_mid_lf_vf- (p_fn_mv+p_lvin_mid)/2)

l_r = np.linalg.norm(p_mid_lf_vf - p_db_mv)
l_l = np.linalg.norm(p_mid_lf_vf - (p_fn_mv+p_lvin_mid)/2)

print(l_r,l_l)
```

```
555 print(closed_angle_r*180/np.pi,closed_angle_l*180/np.pi)
556
557 tau_mv = (p_db_mv - p_fn_mv)
558 norm_mv = np.linalg.norm(tau_mv)
559 tau_mv = tau_mv/norm_mv
560 n_mv = np.array([-tau_mv[1], tau_mv[0]])
561
562 beta_t = np.dot(p_mid_lf_vf - p_fn_mv, tau_mv )/norm_mv #compute parameter beta_t
563 beta_n = np.dot(p_mid_lf_vf - p_fn_mv, n_mv )/norm_mv #compute parameter beta_n
564 print(beta_t,beta_n)
565
566 plt.scatter(p_mid_lf[0],p_mid_lf[1],s=20,label='mid_lf')
567 plt.scatter(p_mid_lf_vf[0],p_mid_lf_vf[1],s=20,label='mid_lf_v2')
568
569 plt.scatter(p_db_av[0],p_db_av[1],s=20,label='db_av')
570 plt.scatter(p_fn_av[0],p_fn_av[1],s=20,label='fn_av')
571 plt.scatter(p_db_mv[0],p_db_mv[1],s=20,label='db_mv')
572 plt.scatter(p_fn_mv[0],p_fn_mv[1],s=20,label='fn_mv')
573 plt.scatter(p_lvin_mid[0],p_lvin_mid[1],s=20,label='lv_in_mid')
574
575 tau_mv = p_fn_mv - p_db_mv
576 tau_mv = -tau_mv/np.linalg.norm(tau_mv)
577 n_mv = -np.array([-tau_mv[1], tau_mv[0]])
578
579 off_set_mv = 3e-3
580
581 p_db_mv_os = p_db_mv + off_set_mv*n_mv
582 p_fn_mv_os = p_fn_mv + off_set_mv*n_mv
583
584 plt.plot([p_db_mv_os[0],p_fn_mv_os[0]],[p_db_mv_os[1],p_fn_mv_os[1]], label='MV')
585 plt.plot([p_db_av[0],p_fn_av[0]],[p_db_av[1],p_fn_av[1]], label='AV')
586 plt.plot(lv_in[:,0],lv_in[:,1], label='LV_in')
587
588 plt.plot([p_db_mv[0],p_db_mv_os[0]],[p_db_mv[1],p_db_mv_os[1]], label='MV_addout')
589 plt.plot([p_fn_mv[0],p_fn_mv_os[0]],[p_fn_mv[1],p_fn_mv_os[1]], label='MV_addin')
590 plt.axis('off')
591 plt.axis('equal')
592
593 plt.legend(loc='best')
594 plt.show()
595
596 ######### Join upper and lower part and interpolate
597
598 #interpolation of the lower with fixed extremities that match upper part
599
600 seg_vf = []
601
602 for _ in range(len(seg_mod)):
603     seg_vf.append(np.concatenate((np.array([p_db_mv]),seg_mod[_],np.array([p_db_av])),axis=0))
604
605 s_grid = np.linspace(0,1,nb_nodes)
606 t_grid = np.linspace(t_mod[0],3*t_mod[-1],3*nb_time_step)
607
608
609 S_list = []
```

```
610 for time_step in range(t_mod.shape[0]):
611     S=[]
612     s=0
613     S.append(0)
614     for pts in range(seg_vf[time_step].shape[0] - 1):
615         dx = seg_vf[time_step][pts+1,0]- seg_vf[time_step][pts,0]
616         dy = seg_vf[time_step][pts+1,1]- seg_vf[time_step][pts,1]
617         s+= np.sqrt(dx**2 +dy**2)
618         S.append(s)
619     S=np.array(S)
620     S = S/S[-1]
621     S_list.append(S)
622 
623 points  = []
624 f = []
625 
626 count =0
627 for _ in range(3):
628     if _==0:
629         for time_step in range(t_mod.shape[0]):
630             for pts in range(seg_vf[time_step].shape[0]):
631                 points.append([S_list[time_step][pts],t_mod[time_step]])
632                 f.append([seg_vf[time_step][pts,0],seg_vf[time_step][pts,1]])
633 
634                 count+=1
635     else:
636         for time_step in range(1,t_mod.shape[0]):
637             for pts in range(seg_vf[time_step].shape[0]):
638                 points.append([S_list[time_step][pts],t_mod[time_step]+_*t_mod[-1]])
639                 f.append([seg_vf[time_step][pts,0],seg_vf[time_step][pts,1]])
640 
641                 count+=1
642 
643 f= np.array(f)
644 points = np.array(points)
645 
646 
647 grid_s, grid_t = np.meshgrid(s_grid,t_grid)
648 
649 
650 interp_x = SmoothBivariateSpline(points[:,0],points[:,1], f[:,0], kx=3, ky=5, s=0.002575)#check around 0.0025
651 interp_y = SmoothBivariateSpline(points[:,0],points[:,1], f[:,1], kx=3, ky=5, s=0.002575)#0.002425
652 
653 grid_x = interp_x(x=grid_s.flatten(order='C'), y=grid_t.flatten(order='C'), grid=False)
654 grid_y = interp_y(x=grid_s.flatten(order='C'), y=grid_t.flatten(order='C'), grid=False)
655 
656 grid_x = grid_x.reshape(grid_s.shape,order='C')
657 grid_y = grid_y.reshape(grid_s.shape,order='C')
658 
659 grid_x[:,0],grid_y[:,0] = p_db_mv[0], p_db_mv[1]
660 grid_x[:,-1],grid_y[:,-1] = p_db_av[0], p_db_av[1]
661 
662 area_select = np.zeros(grid_x.shape[0])
663 
664 for _ in range(grid_x.shape[0]):
```

```
665         pgon = Polygon(np.column_stack((grid_x[_,:],grid_y[_,:])))
666         area_select[_] = pgon.area
667 
668     area_diff = np.diff(area_select, append=area_select[0])
669 
670     i_0 = argrelextrema(data=area_select, comparator=np.greater, order=200)[0][1]
671 
672     mask_diff = (area_diff <= 0)
673     mask_period = np.zeros_like(mask_diff)
674     mask_period[ i_0 - int(1*nb_time_step): i_0] = 1
675     mask_sup = (area_select - area_select[i_0]) >=0
676 
677     if np.any(mask_diff*mask_period*mask_sup):
678         i_1 = np.abs(area_select[mask_diff*mask_period*mask_sup] - area_select[i_0]).argmin() +                    ...
        ↪  np.argmax(mask_diff*mask_period*mask_sup)
679 
680         mesh = np.zeros((i_0-i_1, nb_nodes, 2))
681         mesh[:,:,0] = grid_x[i_1:i_0,:]
682         mesh[:,:,1] = grid_y[i_1:i_0,:]
683 
684     else:
685         i_0 = argrelextrema(data=area_select, comparator=np.greater, order=200)[0][0]
686         mask_diff = (area_diff >= 0)
687         mask_period = np.zeros_like(mask_diff)
688         mask_period[ i_0: i_0 + nb_time_step] = 1
689         mask_sup = (area_select - area_select[i_0]) <=0
690 
691         i_1 = np.abs(area_select[mask_diff*mask_period*mask_sup] - area_select[i_0]).argmin() +                    ...
        ↪  np.argmax(mask_diff*mask_period*mask_sup)
692 
693         mesh = np.zeros((i_1-i_0, nb_nodes, 2))
694         mesh[:,:,0] = grid_x[i_0:i_1,:]
695         mesh[:,:,1] = grid_y[i_0:i_1,:]
696 
697 
698     t_0 = np.linspace(t_mod[0],t_mod[-1],mesh.shape[0])
699 
700     t_new =  np.linspace(t_mod[0],t_mod[-1],nb_time_step)
701     s_new = s_grid
702 
703 
704     interp_x = RectBivariateSpline(t_0, s_grid, mesh[:, :, 0], kx=5, ky=4)
705     interp_y = RectBivariateSpline(t_0, s_grid, mesh[:, :, 1], kx=5, ky=4)
706 
707     grid_x = interp_x(t_new, s_new)
708     grid_y = interp_y(t_new, s_new)
709 
710     mesh_vf = np.zeros((nb_time_step, nb_nodes, 2))
711     mesh_vf[:,:,0] = grid_x
712     mesh_vf[:,:,1] = grid_y
713 
714     mesh_vf[:,0,:] = p_db_mv
715     mesh_vf[:,-1,:] = p_db_av
716 
717     #Discretize segments of the upper part
```

```
718
719 alpha = np.linspace(0, 1, 30)
720
721 mv = np.outer(alpha,p_db_mv_os) + np.outer(1-alpha,p_fn_mv_os)
722 av = np.outer(alpha,p_db_av) +np.outer(1-alpha,p_fn_av)
723 mv_addout = np.outer(alpha,p_db_mv_os) +np.outer(1-alpha,p_db_mv)
724 mv_addin = np.outer(alpha,p_fn_mv) +np.outer(1-alpha,p_fn_mv_os)
725
726 plt.plot(lv_in[:,0],lv_in[:,1], label='lv in')
727 plt.plot(mv[:,0],mv[:,1], label='mv')
728 plt.plot(av[:,0],av[:,1], label='av')
729 plt.plot(mv_addin[:,0],mv_addin[:,1], label='mv add in')
730 plt.plot(mv_addout[:,0],mv_addout[:,1], label='mv add out')
731 plt.plot(mesh_vf[0,:,0],mesh_vf[0,:,1],label='lv out')
732 plt.pcolormesh(x_grid, y_grid, filtered_image, shading='gouraud', cmap='gray')
733
734 plt.legend(loc='best')
735 plt.axis('off')
736 plt.axis('equal')
737 plt.show()
738
739 #Flip the curves to match simulation referential
740
741 filtered_image_flipped = np.fliplr(filtered_image)
742
743 x_min, x_max = np.min(x_grid), np.max(x_grid)
744
745 lv_in[:,0] = x_max - (lv_in[:,0] - x_min)
746 mv[:,0] = x_max - (mv[:,0] - x_min)
747 av[:,0] = x_max - (av[:,0] - x_min)
748 mv_addin[:,0] = x_max - (mv_addin[:,0] - x_min)
749 mv_addout[:,0] = x_max - (mv_addout[:,0] - x_min)
750 mesh_vf[:,:,0] = x_max - (mesh_vf[:,:,0] - x_min)
751
752
753 curve_name = ['MV','AV','LVW_in','MVadd_in','MVadd_out']
754 curve = [mv, av, lv_in, mv_addin, mv_addout]
755
756 # Writing curves
757
758 for i in range(len(curve)):
759     curve_iter = curve[i]*1000
760     x_file = patient + "\\geom" + f"\\xcoord_{curve_name[i]}.csv"
761     y_file = patient + "\\geom" + f"\\ycoord_{curve_name[i]}.csv"
762
763     # Write X coordinates
764     with open(x_file, 'w') as fx:
765         fx.write(f"xcoord_{curve_name[i]}\n\n")
766         for _ in range(mesh_vf.shape[0]):
767             fx.write(" ".join(f"{val:.6f}," for val in curve_iter[:-1,0]))
768             fx.write(" " + f"{curve_iter[-1,0]:.6f}" + "\n")
769
770     # Write Y coordinates
771     with open(y_file, 'w') as fy:
772         fy.write(f"ycoord_{curve_name}\n\n")
```

```
        for _ in range(mesh_vf.shape[0]):
            fy.write(" ".join(f"{val:.6f}," for val in curve_iter[:-1,1]))
            fy.write(" " + f"{curve_iter[-1,1]:.6f}" + "\n")

x_file = patient + "\\geom" + f"\\xcoord_LVW_out.csv"
y_file = patient + "\\geom" + f"\\ycoord_LVW_out.csv"
mesh_iter = mesh_vf * 1000
# Write X coordinates
with open(x_file, 'w') as fx:
    fx.write(f"xcoord_LVW_out\n\n")
    for _ in range(mesh_vf.shape[0]):
        fx.write(" ".join(f"{val:.6f}," for val in mesh_iter[_,:-1,0]))
        fx.write(" " + f"{mesh_iter[_,-1,0]:.6f}" + "\n")

# Write Y coordinates
with open(y_file, 'w') as fy:
    fy.write(f"ycoord_LVW_out\n\n")
    for _ in range(mesh_vf.shape[0]):
        fy.write(" ".join(f"{val:.6f}," for val in mesh_iter[_,:-1,1]))
        fy.write(" " + f"{mesh_iter[_,-1,1]:.6f}" + "\n" )


##### Extract opened angles from mode B ECG

%matplotlib tk

import matplotlib.pyplot as plt
import matplotlib.image as mpimg


idx = np.abs(t_mod - t_grid[diff_max[0]]).argmin()
fig, ax = plt.subplots(1,1, figsize=(16, 12))


img = tissue[:,:,idx].astype(np.uint8)

pts  = []
def onclick(event):
    ax.scatter(event.xdata, event.ydata)
    fig.canvas.draw()
    ix, iy = event.xdata, event.ydata
    pts.append([ix, iy])

    plt.scatter(ix, iy, color='r')

cid = fig.canvas.mpl_connect('button_press_event', onclick)
imgplot = ax.pcolormesh(x_tissue, y_tissue, tissue[:,:,idx].astype(np.uint8), shading='gouraud', cmap='gray')

x_min, x_max = np.min(x_grid), np.max(x_grid)
mv_rv, mv_addin_rv, mv_addout_rv = mv.copy(), mv_addin.copy(), mv_addout.copy()
mv_rv[:,0] = x_max - (mv_rv[:,0] - x_min)
mv_addin_rv[:,0] = x_max - (mv_addin_rv[:,0] - x_min)
mv_addout_rv[:,0] = x_max - (mv_addout_rv[:,0] - x_min)

ax.plot(mv_rv[:,0], mv_rv[:,1], label='mv')
```

```
828 ax.plot(mv_addin_rv[:,0], mv_addin[:,1], label='mv add in')
829 ax.plot(mv_addout_rv[:,0], mv_addout[:,1], label='mv add out')
830
831
832 ax.set_title("Tissue Mode B")
833 ax.axis("off")
834 ax.axis("equal")
835 plt.show()
836
837 pts= np.array(pts)
838 p_mv_out, p_lv_r, p_mv_in, p_lv_l = pts[0,:], pts[1,:], pts[2,:], pts[3,:]
839
840 opnened_angle_r = angle_between_vectors(p_lv_r - p_mv_out , p_mv_in - p_mv_out)
841 opnened_angle_l = angle_between_vectors(p_lv_l - p_mv_in , p_mv_out - p_mv_in)
842
843 print(opnened_angle_l*180/np.pi, opnened_angle_r*180/np.pi  )
844
845
```

Listing 12 Reconstruction de la géométrie du VG à partir de la segmentation EchoPAC et les données EchoCG mode B