UNIVERSITÉ DE STRASBOURG

École Doctorale Mathématiques, Sciences de l'Information et de l'Ingénieur

Institut de Recherche Mathématique Avancée, UMR 7501

THÈSE présentée par

Thomas Saigre

soutenue le 20 décembre 2024

pour obtenir le grade de Docteur de l'Université de Strasbourg

Spécialité : Mathématiques Appliquées

Modélisation mathématique, simulation et réduction d'ordre de flux oculaires et leurs interactions : Construire le jumeau numérique de l'œil

Mathematical modeling, simulation and order reduction of ocular flows and their interactions: Building the digital twin of the eye

---

THÈSE DIRIGÉE PAR :

Christophe PRUD'HOMME	Professeur, Directeur de thèse, Université de Strasbourg
Marcela SZOPOS	Professeure, Directrice de thèse, Université Paris-Cité

RAPPORTEURS :

Florian DE VUYST	Professeur, Université de Technologie de Compiègne
Rodolfo REPETTO	Professeur, Università di Genova

EXAMINATEURS :

Stéphane COTIN	Directeur de Recherche, INRIA Strasbourg
Yannick HOARAU	Professeur, Université de Strasbourg
Damiano LOMBARDI	Chargé de Recherche, INRIA Paris

Hergé, *Les Cigares du Pharaon*, 1955

Remerciements

En premier lieu, je tiens à remercier mes deux directeurs de thèse, Christophe Prud'homme et Marcela Szopos, pour m'avoir donné l'opportunité de travailler sur ce sujet passionnant. Merci pour votre soutien, vos conseils, et tout le temps que vous m'avez consacré, que ce soit pour discuter des travaux, pour relire mes écrits, ou plus spécifiquement pour passer des heures avec moi pour trouver le − qui manquait dans les équations ou dans le code !

Je remercie Florian De Vuyst et Rodolfo Repetto d'avoir accepté de rapporter ma thèse, et pour tous les retours qu'ils ont pu faire sur mon travail. Merci aussi à Stéphane Cotin, Yannick Hoarau et Damiano Lombardi d'avoir pris part au jury de ma thèse.

Je remercie aussi les collègues avec qui j'ai eu plaisir à travailler au cours de ces dernières années, et qui m'ont parfois sorti d'impasses, que ce soit mathématiquement ou dans la programmation avec Feel++. En particulier, merci à toute l'équipe de Cemosis, avec Céline, Vincent, Philippe, Christophe, Patrick, Gwennolé, Javier, Maryam ; mais aussi celles et ceux qui sont partis vers d'autres horizons, comme Luca, Zohra ou Ali.

Merci aussi à Javan, Lucas, Lorenzo, Thibault et Béatrice avec qui j'ai pu travailler pendant tout un été sur un sujet différent de mon sujet de thèse, mais non moins intéressant ! Merci aussi à toutes celles et ceux que j'ai pu croiser au CEMRACS, qui ont fait de ces cinq semaines à Marseille une expérience inoubliable !

Je remercie aussi le personnel administratif et technique de l'IRMA et de l'UFR, sans qui ce séjour à Strasbourg n'aurait pas été aussi agréable.

Durant ces trois années, j'ai eu la chance d'encadrer le Cercle Mathématique de Strasbourg, et je tiens à remercier tous les élèves et aussi les autres encadrants avec qui j'ai pu travailler, Tatiana, Xiaolin, Christophe et Marianne. Je sais que maintenant les élèves sont entre de bonnes mains, car la relève est assurée en les personnes de Lauriane et Thomas !

Ces années de dur labeur n'auraient pas été les mêmes sans la présence de tous mes ~~collègues~~ amis doctorants et post-doctorants avec qui j'ai vécu des moments incroyables, que ce soit au travail ou en dehors. Merci pour les longues pauses !kafé, les discussions interminables au Télégraphe (ou au Comptoir pour les plus anciens), les cinés, les randonnées, les parties de tarot ou de coinche, les soirées et tous ces moments incroyables que j'ai vécu avec vous. Merci donc à Brieuc, Léopold, Roméo, Clarence, Victoria, Céline, Antoine, Thomas (tu seras toujours le premier des Thomas !), Adam, Guillaume, Paul, Tom, Ludovic, Romane, Mickaël :mb:, Basile, Raoul, Claire, Roxana, Vincent, Anna, Robin, Colin, Nicolas, Jules, Rym, Clément, Louise, Victor, Christopher, Lauriane, Florian, Thibault, Yohann, Alex, Jean-Pierre *et al.* Merci à la team Click, anciennement tribu des BO'réliens, pour les innombrables séances d'escalade et aussi aux grimpeurs du SUAPS (Victoria, Damien, Mélissa, Alba, Enzo, ...) avec qui j'ai toujours autant de plaisir à grimper.

Si par malheur j'ai oublié de citer votre nom dans ces quelques lignes, c'est simplement parce que la marge est trop petite pour le contenir, mais vous avez toutes et tous été géniaux, quand

j'ai choisi de faire ma thèse ici, on peut dire que oui, j'ai vraiment fait le bon choix !

Merci aussi à Nicolas pour avoir été un si bon ami, et pour tous ces précieux moments qu'on a passés ensemble, entre l'UGC et le PMC.

Je remercie aussi ma famille, qui me supporte (dans tous les sens du terme) depuis plus de 25 ans maintenant, merci à mes parents Christophe et Muriel, à mes frères et sœurs, Chloé, Hugoooo, Éva et à tout le monde pour avoir grandement contribué à faire ce que je suis maintenant.

Enfin, je termine ces quelques lignes en ayant une pensée particulièrement émue pour ma maman, qui d'où elle est, doit certainement être fière du chemin parcouru.

Contents

Notations

Abreviations

AC Anterior Chamber.
AH Aqueous Humor.
CAD Computer Aided Design.
CFD Computational Fluid Dynamics.
CRB Certified Reduced Basis.
DSA Deterministic Sensitivity Analysis.
FE Finite Element.
FEM Finite Element Method.
GAMG Geometric Algebraic MultiGrid.
GCC Geometrical Center of the Cornea.
GLV Generalized Lotka-Volterra.
GSA General Smoothing Algorithm.
HF High Fidelity.
HPC High Performance Computing.
IOP Intraocular Pressure.
KSP Krylov Subspace Projection.
LS Least Squares.
MOR Model Order Reduction.
nDof number of degrees of freedom.
NIRB Non Intrusive Reduced Basis.
ODE Ordinary Differential Equation.
OMVS Ocular Mathematical Virtual Simulator.
PC Posterior Chamber.
PDE Partial Differential Equation.
PINN Physics-Informed Neural Network.
POD Proper Orthogonal Decomposition.
RBM Reduced Basis Method.
RIC Relative Information Content.
SA Sensitivity Analysis.
SSA Stochastic Sensitivity Analysis.
TM Trabecular Meshwork.
UA Uncertainty Analysis.
UQ Uncertainty Quantification.
WSS Wall Shear Stress.

Mathematical notations

$\vec{v}$ vector representing a physical quantity.

$\underline{\boldsymbol{v}}$ vector representing an algebraic quantity.

$\underline{\underline{A}}$ matrix or tensor.

$\underline{\underline{\boldsymbol{A}}}$ matrix representing an algebraic quantity.

Abstract

The human body is a complex system, and the human eye is no exception. Despite significant advancements in medical research, many questions regarding ocular pathologies remain unanswered. The use of mathematical and computational models has revealed intricate mechanisms underlying human physiology. Due to its special connection with the brain, the eye is considered a window into the brain, providing non-invasive access to a range of biological markers that can aid in diagnosing neurodegenerative diseases. Therefore, understanding the eye's behavior, the diseases that affect it, and the potential treatments is crucial.

This thesis focuses on the mathematical modeling and numerical simulation of ocular fluid dynamics within the human eye, particularly on heat transfer and aqueous humor flow. These methods must undergo validation with clinical data to ensure their reliability. Bio-physical models involve numerous parameters that may be patient-specific or influenced by external conditions. The objective of this sensitivity analysis is to understand how these parameters affect the eye's behavior and how they can be used for disease diagnosis, potentially assisting clinicians in selecting the best treatment.

Such a study requires numerous simulations, which can be computationally expensive for the complex models used in this thesis. To reduce this computational cost, we have developed model reduction methods that decrease the number of equations to solve while maintaining result accuracy.

In the first part of the thesis, we present the geometric and bio-physical models of the eyeball concerning heat transfer. Next, we discuss the numerical discretization methods implemented to simulate this model. The third chapter introduces the model reduction techniques used to lower the computational cost of these simulations, particularly through the certified reduced basis method. The fourth chapter addresses a specific issue related to the study of pointwise quantities of interest within the framework of reduced bases for elliptic problems with a Dirac source term. In the fifth chapter, we present the sensitivity analysis results obtained from our models. The sixth chapter extends the initial thermal model by incorporating the flow of aqueous humor in the anterior and posterior chambers of the eye. Finally, the seventh chapter provides an overview of the implementation contributions of this thesis.

Résumé

Le corps humain est un système complexe, l'œil humain n'y fait pas exception. Malgré les avancées significatives de la recherche dans le domaine médical, de nombreuses questions sur les pathologies restent sans réponse. En complément des études cliniques, l'utilisation de modèles mathématiques et computationnels a permis de révéler des mécanismes complexes de la physiopathologie humaine. Grâce à sa connexion spéciale avec le cerveau, l'œil est considéré comme une fenêtre vers celui-ci, et permet un accès non invasif à un ensemble de marqueurs biologiques qui pourraient aider au diagnostic de maladies neurodégénératives. C'est pour cela qu'il est crucial de comprendre le comportement de l'œil, des maladies qui y sont liées et des traitements qui peuvent être appliqués.

Dans cette thèse, nous nous concentrons sur la modélisation mathématique et la simulation numérique de flux oculaire dans l'œil humain, particulièrement sur le transfert de chaleur et le flux d'humeur aqueuse. Ces méthodes doivent par ailleurs passer par une phase de validation avec des données cliniques, pour s'assurer de leur validité. Dans les modèles bio-physiques, de nombreux paramètres interviennent, qui peuvent être spécifique à chaque patient ou venir de conditions extérieures. Afin de comprendre comment ces paramètres influent sur le comportement de l'œil, et comment ils peuvent être utilisés pour diagnostiquer des maladies, et éventuellement aider les cliniciens à choisir le meilleur traitement, une analyse de sensibilité globale a été menée.

Une telle étude requiert un grand nombre de simulations, qui se révèle être très coûteux en temps de calcul pour des modèles complexes tels que ceux utilisés dans le cadre de cette thèse. Pour réduire ce coût, nous avons développé des méthodes de réduction de modèle certifiées, qui permettent de réduire le nombre d'équations à résoudre, tout en conservant la précision des résultats.

Dans une première partie, nous présentons les modèles géométriques et bio-physiques du globe oculaire dans le cadre du transfert thermique. Ensuite, nous présentons les moyens de discrétisations numériques mis en place pour simuler ce modèle. Dans un troisième chapitre, nous présentons les méthodes de réduction de modèle utilisées pour réduire le coût de calcul de ces simulations, en particulier avec la méthode des bases réduites certifiées. Dans le quatrième chapitre, nous nous intéressons à une problématique soulevée par l'étude de quantités d'intérêt ponctuelles dans le cadre des bases réduites à propos des problèmes elliptiques en présence d'un terme source de type Dirac. Ensuite, dans le cinquième chapitre, nous présentons les résultats de l'analyse de sensibilité obtenus via nos modèles. Dans un sixième chapitre, nous présentons une extension de ce premier modèle de thermique, en prenant en compte le couplage avec le flux d'humeur aqueuse dans les chambres antérieures et postérieures de l'œil. Enfin, dans un septième chapitre, nous présentons un aperçu des contributions implémentées dans le cadre de cette thèse.

Introduction en français

Avec le développement des nouvelles technologies, l'intérêt pour les modèles mathématiques en biologie et en médecine s'est considérablement accru. En effet, l'utilisation de modèles informatiques peut aider à comprendre des systèmes biologiques complexes et à prédire le comportement de ces systèmes dans diverses conditions.

En raison de sa connexion spéciale avec le cerveau, l'œil est considéré comme une fenêtre sur le cerveau, offrant un accès non invasif à un large ensemble de biomarqueurs qui pourraient faciliter, par exemple, le diagnostic précoce des maladies neurodégénératives ou des conditions cardiovasculaires [Che+24]. Cependant, la caractérisation des biomarqueurs oculaires comme substituts de l'état vasculaire cérébral ou du corps humain est loin d'être triviale [Gui+20]. Les mesures cliniques sont influencées par de nombreux facteurs qui varient d'un individu à l'autre et ne peuvent être isolés *in vivo*, ce qui complique considérablement leur interprétation.

Comme reporté dans [Fly+23], la distribution actuelle des sources de preuves scientifiques est également répartie entre les études humaines, animales et de laboratoire, avec seulement 1% s'appuyant sur des modèles informatiques. Toutefois, les projections basées sur le programme *Computational Modeling and Simulation* suggèrent une évolution future vers une plus grande utilisation des simulations informatiques, qui devraient représenter 45% des preuves scientifiques, comme illustré dans le graphique en Figure 1. Cette tendance reflète l'intégration croissante des techniques informatiques avancées dans le développement des dispositifs médicaux, réduisant ainsi la dépendance aux méthodes traditionnelles.

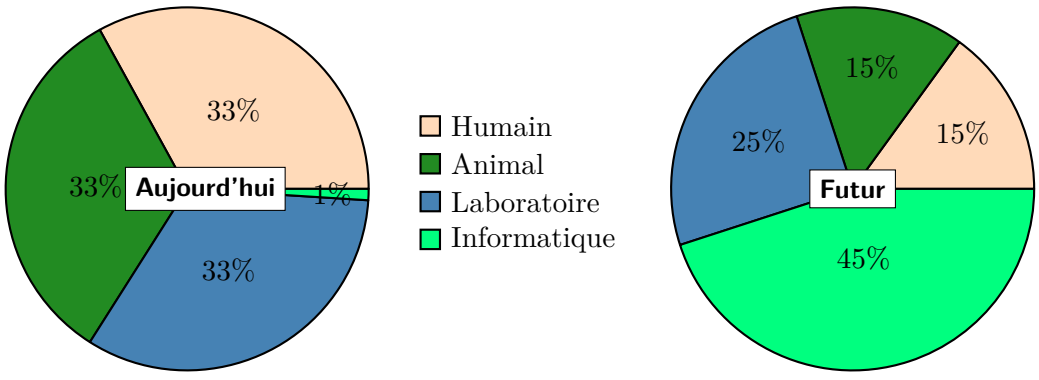


Figure 1: Sources de preuves scientifiques, extrait de [Fly+23].

Les modèles computationnels jouent un rôle crucial dans la recherche scientifique, notamment en biologie et en médecine. En effet, ils permettent de limiter le nombre d'expériences physiques nécessaires pour développer un produit ou approfondir la compréhension d'une maladie, ce qui réduit considérablement les coûts et diminue la dépendance vis-à-vis des données issues de tests sur des animaux et des humains, en favorisant les données de patients virtuels.

L'étude des biofluides par des modèles mathématiques et informatiques n'est pas une approche récente. Le système cardiovasculaire fut le premier système humain ainsi étudié, d'abord par

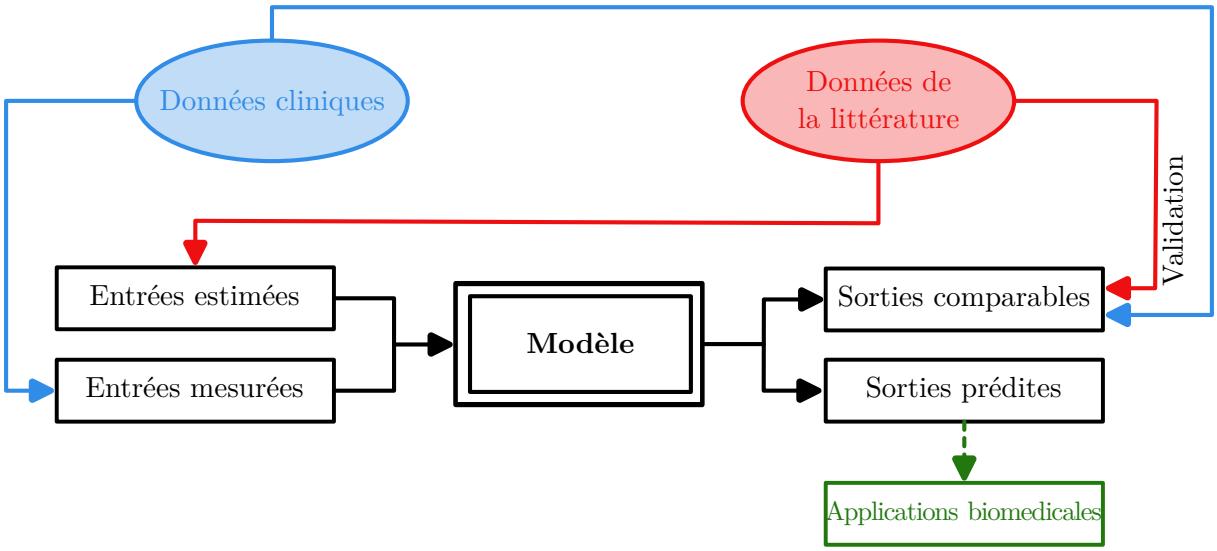


Figure 2: Méthodologie pour le développement de modèles spécifiques aux patients, adaptée de [Sal+23].

une approche purement mathématique au XVIII^{e} siècle [Eul75 ; Hal+33], puis sous un angle computationnel [NVB63]. Par la suite, d'autres organes comme les poumons [Mau13], le cerveau [Mar+22], *etc.* ont également été explorés.

Pour être complémentaires à la recherche traditionnelle, les modèles mathématiques et informatiques doivent être basés sur des données cliniques, combinées à des connaissances biophysiques et doivent être validés pour garantir leur fiabilité. Ce type de modèles a déjà été développé pour des applications biomédicales spécifiques, telles que le système cardiovasculaire [Upd+17 ; Art+21] ou l'hémodynamique cérébrale [Mar+22]. La Figure 2 illustre la méthodologie pour le développement de modèles spécifiques au patient, en soulignant l'importance de l'intégration des données à la fois du point de vue clinique et numérique.

L'intérêt pour l'œil est plus récent, mais a connu une croissance significative au cours des dernières décennies. Plusieurs maladies oculaires sont encore mal comprises par la communauté scientifique et le développement de modèles informatiques peut aider à mieux comprendre les mécanismes sous-jacents [Gui+22]. Plusieurs modèles ont été proposés, comme étudié dans [GHS19 ; HRE23]. En particulier, un modèle tenant compte des effets combinés du flux sanguin oculaire et des différents tissus oculaires a été proposé dans [Sal+23]. L'*Ocular Mathematical Virtual Simulator* (OMVS) [Sal19 ; Sal+23], est un cadre mathématique et informatique qui permet de simuler des phénomènes multiphysiques impliqués dans l'œil, à partir de données spécifiques au patient. Pour intégrer les incertitudes et la variabilité inhérentes aux modèles, une analyse de propagation d'incertitude et de sensibilité sur le composant simulant les flux de fluides dans l'œil a été menée dans [PSS21]. Ces méthodes de calcul et ces cadres *in silico* peuvent être utilisés pour les patients virtuels et pour développer des modèles spécifiques aux patients ou des jumeaux numériques œil-patient [Sal+21a].

Dans ce contexte, cette thèse s'inscrit dans le cadre du projet *Eye2Brain* [Sal16], dont l'objectif ambitieux est de connecter les environnements cérébral et oculaire et de contribuer à long terme à une meilleure compréhension des maladies neurodégénératives [Gui+20]. Bien que l'étude du transfert de chaleur dans l'œil puisse ne pas sembler directement liée au cerveau, ce travail s'aligne sur les objectifs du projet en améliorant notre compréhension des processus physiologiques au sein de l'environnement oculaire, ce qui pourrait fournir des informations précieuses sur des conditions systémiques et neurologiques.

Alors que des études antérieures se sont concentrées sur l'interaction entre l'hémodynamique et

la biomécanique dans l'œil, nous nous concentrons dans ce travail sur le transfert de chaleur dans l'œil et son couplage avec l'écoulement de l'humeur aqueuse. Plusieurs modèles mathématiques ont été proposés pour l'étude de divers aspects de la distribution de la température dans le globe oculaire humain et de ses variations dues au champ d'écoulement dans les chambres antérieure et postérieure. En particulier, le transfert de chaleur dans l'œil a été étudié dans [Sco88 ; NO06 ; NO07 ; ON08 ; Li+10], ainsi que l'écoulement de l'humeur aqueuse dans la chambre antérieure et son couplage avec le transfert de chaleur [HB02 ; Wan+16 ; Mur+23]. Tous ces aspects sont importants dans la perspective des connexions œil-cerveau [Gui+20], visant à prédire la distribution des contraintes et des déformations survenant au cours de procédures diagnostiques ou thérapeutiques, telles que le traitement par hyperthermie des tumeurs oculaires [Li+10] ou le traitement par injection de cellules pour guérir la kératopathie bulleuse [Kin+18].

Alors que des études invasives ont été menées sur des animaux [PW05], les mesures non invasives sur des sujets humains sont rares, complexes à réaliser et peuvent donner des résultats inexacts [RF77]. La plupart des études se concentrent sur les mesures de température à la surface de l'œil [Map68 ; EYB89], mais elles font état de différences significatives et identifient plusieurs sources d'incertitude. Par ailleurs, les simulations numériques peuvent fournir des informations complémentaires, éventuellement non disponibles pour les mesures expérimentales, mais essentielles pour l'administration de médicaments et les évaluations pré- et post-opératoires, telles que la contrainte de cisaillement des parois internes de l'œil.

L'analyse de sensibilité [Raz+21], en particulier par l'utilisation des indices de Sobol [Sob93], permet d'explorer de façon systématique la façon dont les variations des paramètres du modèle contribuent à la variabilité globale des résultats. Cela permet d'identifier les paramètres les plus influents et de comprendre leurs interactions.

L'effort de calcul nécessaire pour résoudre des problèmes réalistes dans ce contexte est considérable. Tout d'abord, la représentation de géométries 3D réalistes implique la manipulation d'une quantité importante de données, ce qui accroît la complexité des simulations. Ensuite, l'inclusion de modèles multiphysiques complexes, tels que le couplage du transfert de chaleur avec la dynamique des fluides, ajoute des difficultés de calcul supplémentaires. En outre, l'incorporation d'incertitudes dans les paramètres d'entrée par la quantification d'incertitudes et l'analyse de sensibilité nécessite l'exécution de plusieurs simulations, chacune avec des paramètres variables. Ce processus augmente considérablement la charge de calcul, ce qui rend essentielle l'utilisation de ressources informatiques de haute performance (HPC), et le développement de méthodes numériques efficaces pour réduire les coûts de calcul [QMN16 ; Pru+02].

Conformément à ces exigences de calcul, un autre aspect de cette thèse fait partie du projet Exa-MA (*Methods and Algorithms for Exascale*) du programme PEPR NumPEx [Num24], qui se concentre sur les défis exascale des méthodes numériques et des algorithmes.

Contributions

Dans cette thèse, nous développons un modèle mathématique et informatique en trois dimensions de l'œil humain, en nous concentrant d'abord sur le transfert de chaleur, puis sur son couplage avec l'écoulement de l'humeur aqueuse. Les modèles proposés intègrent divers paramètres biomécaniques et géométriques. Nous nous concentrons ici sur les paramètres biomécaniques, qui couvrent un large éventail de valeurs, y compris des conditions extrêmes potentielles. La variation de ces paramètres peut avoir un impact significatif sur les résultats. Pour quantifier ces impacts, nous mettons en place une étude de quantification d'incertitude, accompagné d'une analyse de sensibilité approfondie.

Une validation rigoureuse est par ailleurs une étape clé pour garantir la fiabilité des résultats. Nous confrontons nos résultats aux données disponibles dans la littérature, issues soit de mesures réalisées sur des sujets en bonne santé [EYB89], soit de simulations numériques [NO06 ; NO07 ;

Li+10 ; KS10], afin de confirmer la robustesse et la validité de notre approche.

Alors que les indices de Sobol mesurent efficacement les impacts et les interactions entre paramètres, la complexité et le temps de calcul élevé requis par notre modèle posent des défis considérables. Pour surmonter ces difficultés, nous adoptons la méthode des bases réduites certifiées (RBM) [Pru+02 ; QMN16], une approche mathématique conçue pour accélérer les simulations de systèmes complexes sans sacrifier la précision. Par essence, le RBM crée une version simplifiée du modèle original en identifiant et en conservant uniquement ses caractéristiques les plus essentielles, ce qui nous permet de construire un modèle réduit qui conserve sa nature 3D tout en réduisant considérablement les coûts de calcul et en fournissant des résultats fiables.

La RBM se compose de deux phases : une phase dite *hors ligne*, au cours de laquelle la version simplifiée (ou espace réduit) est construite, et une phase *en ligne*, au cours de laquelle le modèle réduit est utilisé pour calculer efficacement les solutions. Bien que la phase hors ligne soit coûteuse en termes de calcul, elle n'est exécutée qu'une seule fois. Par la suite, la phase en ligne permet des simulations rapides, ce qui rend cette méthode particulièrement adaptée aux scénarios nécessitant des évaluations répétées, telles que l'analyse de sensibilité ou la quantification de l'incertitude. Cette méthodologie s'aligne sur le paradigme observé dans les modèles mathématiques spécifiques aux patients appliqués aux problèmes biomédicaux, garantissant une approche globale impliquant l'intégration des données, la dérivation du modèle, la résolution numérique, la validation et la quantification d'incertitude. Ces paradigmes sont bien établis dans des domaines tels que les simulations cardiovasculaires et l'hémodynamique cérébrale. En ophtalmologie, un cadre similaire s'impose, en raison de la diversité et de l'hétérogénéité des données disponibles, et requiert des stratégies innovantes pour optimiser le diagnostic et le suivi des patients.

Pour compléter la méthode RBM, nous mettons également en œuvre la méthode des bases réduites non intrusives (NIRB) [CM09b ; CM09a ; Gro22], une variante de la méthode RBM qui est particulièrement utile lorsque la complexité du modèle empêche l'application directe de cette dernière. Contrairement au modèle RBM standard, le modèle NIRB utilise une stratégie de discrétisation à deux niveaux, dans laquelle un modèle de substitution simplifié est construit et résolu sans nécessiter de modifications directes du code de simulation original. Par ailleurs, le paradigme hors ligne/en ligne reste pleinement applicable dans le cadre de la méthode NIRB.

Pour garantir une convergence quadratique rapide des résultats, la méthode RBM repose sur une formulation duale du problème, dans laquelle chaque sortie est exprimée comme une fonction linéaire de la solution. Cependant, dans certaines quantités d'intérêt, comme celles qui sont étudiées dans cette thèse, la sortie peut impliquer des évaluations ponctuelles de la solution. Cela engendre un problème dual intégrant un terme source de Dirac δ_x, localisé au point x où la solution est évaluée. Dans de telles situations, la théorie classique de RBM n'est pas directement applicable. Néanmoins, nos résultats numériques mettent en évidence des propriétés de convergence en accord avec les prédictions théoriques dans le cas continu. Les aspects théoriques et numériques du problème du laplacien avec un terme source de Dirac ont été étudiés dans [KW14 ; Ber+18]. Bien que les résultats théoriques soient établis sous des hypothèses spécifiques, nous observons des comportements similaires même lorsque ces conditions ne sont pas strictement satisfaites. Nous proposons une analyse théorique et une exploration numérique du problème laplacien avec un terme source de Dirac, en examinant diverses conditions aux limites ainsi que différents ordres de discrétisation.

Nous étendons également le modèle thermique initial en intégrant l'écoulement d'humeur aqueuse dans la chambre antérieure (AC) et postérieure (PC) de l'œil, en le couplant aux processus thermiques affectant l'ensemble du globe oculaire. Ce couplage est crucial, car le mouvement d'humeur aqueuse joue un rôle déterminant dans la répartition de la chaleur et la régulation de la pression intraoculaire, deux facteurs essentiels pour la santé globale de l'œil. Par ailleurs, la contrainte de cisaillement exercée sur les parois par l'écoulement de l'humeur aqueuse constitue un facteur biomécanique clé. Elle influence directement la santé des tissus

oculaires et peut affecter les voies de drainage, un aspect particulièrement pertinent dans des pathologies comme le glaucome. Plusieurs études antérieures ont exploré divers aspects de cette interaction complexe. Par exemple, [HB02 ; Wan+16 ; Mur+23] ont modélisé l'écoulement couplé au transfert de chaleur dans les chambres AC et PC, tandis que [Sac+23] a analysé l'impact de la pression sur l'écoulement et le drainage de l'humeur aqueuse. D'autres travaux [ON08 ; BBS20 ; Abd+21 ; Dvo+22] ont étudié la dynamique thermo-fluide de l'écoulement d'humeur aqueuse dans l'œil, en considérant des conditions aux limites spécifiques. Cependant, ces études se concentrent souvent sur des géométries simplifiées ou négligent le couplage complet avec le transfert de chaleur dans l'ensemble du globe oculaire. Pour résoudre efficacement le problème numérique, nous adoptons des techniques avancées de préconditionnement multiphysique. Ces approches combinent des préconditionneurs tels que GAMG [Bal+24] ou des préconditionneurs par blocs efficaces pour les équations considérées [ESW14], afin de résoudre numériquement le problème couplé de manière optimale. Enfin, dans un cadre de calcul haute performance, nous analysons les propriétés d'extensibilité du modèle pour évaluer ses performances à grande échelle.

Au cours de ces trois années de thèse, de nombreuses contributions ont été apportées aux codes de calculs, menant à la production de plusieurs ensembles de données. Pour garantir que ces résultats puissent être validés et reproduits de manière indépendante, toutes ces contributions ont été publiées en libre accès, favorisant la transparence et la réutilisation des travaux [Cam+24].

Plan du manuscrit

Chapitre 1 : Modélisation mathématique des mécanismes de transfert de chaleur dans l'œil humain Dans ce premier chapitre, nous présentons dans un premier temps l'anatomie et la physiologie du globe oculaire humain, ainsi que les différents défis pharmacologiques et médicaux liés aux modèles considérés. Précisément, à partir de publications précédentes, nous développons un modèle de transfert de chaleur dans l'œil humain, en prenant en compte les différents mécanismes de transfert de chaleur, les conditions aux limites et les propriétés thermophysiques des tissus oculaires :

$$\nabla \cdot (k \nabla T) = 0 \quad \text{dans l'œil,} \tag{1a}$$

$$k \frac{\partial T}{\partial \vec{n}} = h_{\text{amb}}(T - T_{\text{amb}}) + \sigma \varepsilon (T^4 - T_{\text{amb}}^4) + E \quad \text{sur la surface externe de l'œil,} \tag{1b}$$

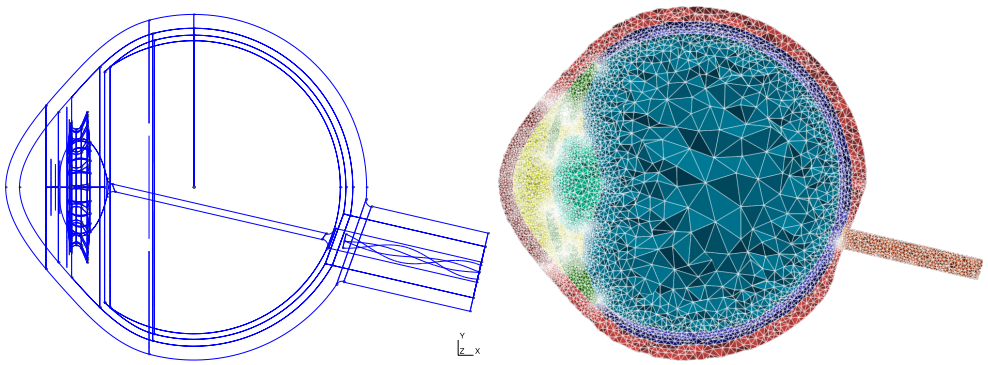
$$k \frac{\partial T}{\partial \vec{n}} = h_{\text{bl}}(T - T_{\text{bl}}) \quad \text{sur les surfaces internes.} \tag{1c}$$

Dans le modèle développé, de nombreux paramètres $\mu = [E, h_{\text{bl}}, h_{\text{amb}}, T_{\text{bl}}, T_{\text{amb}}, k]$ interviennent, qui peuvent être spécifiques à chaque patient ou provenir de conditions extérieures, et c'est précisément l'étude de l'impact de ces paramètres sur le comportement de l'œil qui est l'objet de cette thèse. Nous nous intéresserons en particulier à des quantités d'intérêt, telles que la température à un point donné ou la température moyenne dans une région de l'œil.

Enfin nous présentons le modèle mathématique qui découle de ces considérations, sous une formulation variationnelle de la forme : Trouver le champ de température $T(\mu) \in V$ tel que pour toute fonction test $v \in V$, l'égalité suivante soit satisfaite :

$$a(T(\mu), v; \mu) = f(v; \mu). \tag{2}$$

Chapitre 2 : Cadre computationnel du modèle haute fidélité : méthodes, vérifications et validations Dans ce chapitre, nous présentons les méthodes numériques utilisées pour résoudre le modèle mathématique développé dans le premier chapitre. Tout d'abord, nous



(a) Géométrie réaliste de l'œil...

(b) ... convertie en un maillage raffiné au niveau des zones d'intérêt (les chambres antérieures et postérieures)

Figure 3: Modèle géométrique : maillage à partir de données CAO.

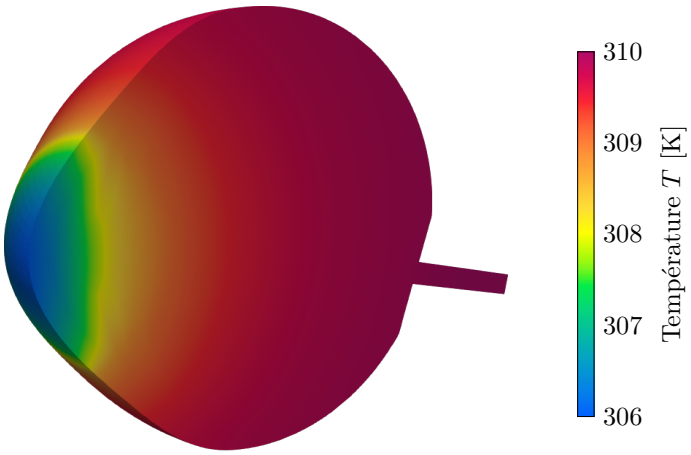


Figure 4: Distribution de la température sur l'œil, obtenue par la méthode des éléments finis.

commençons par présenter le modèle géométrique discret, c'est-à-dire la construction du maillage à partir des données CAO (Conception Assistée par Ordinateur) de l'œil, mais aussi les techniques de raffinement, comme illustré en Figure 3. Une étape de vérification des maillages générés est aussi effectuée.

Ensuite, nous présentons la méthode des Éléments Finis, qui permet de calculer une solution numérique au problème mathématique (2). Précisément, grâce à la discrétisation de la géométrie, nous obtenons un équivalent discret au problème (2), via l'introduction d'un espace de fonction $V_h \subset V$ de dimension finie $\mathcal{N}$: trouver $T_h(\mu) \in V_h$ tel que pour toute fonction test $v_h \in V_h$, l'égalité suivante soit satisfaite :

$$a(T_h(\mu), v_h; \mu) = f(v_h; \mu). \tag{3}$$

À partir de ce problème discrétisé, nous pouvons obtenir une solution numérique, comme celle qui est donnée en Figure 4. Nous menons une étude de convergence et de stabilité de la méthode, mais aussi une étape de vérification et de validation du modèle avec des données simulées d'autres publications, et aussi de données cliniques mesurées sur des patients.

Chapitre 3 : Cadre computationnel du modèle réduit : méthodes et vérifications
Le modèle haute fidélité présenté dans le chapitre précédent est coûteux en temps de calcul et

les études d'analyse de sensibilité que l'on va effectuer dans la suite nécessitent un grand nombre de simulations. Pour cette raison, nous mettons en place des méthodes de réduction d'ordre, permettant de réduire le temps de calcul tout en conservant la précision des résultats.

Dans un premier temps, nous présentons la méthode des bases réduites, avec bornes d'erreur certifiées, qui permet d'explorer l'espace des paramètres du modèle, et de calculer des quantités d'intérêt en temps réel. Précisément, une espace de fonction $V_N \subset V_h$ est construit, de dimension $N \ll \mathcal{N}$, et une solution approchée $T_N(\mu) \in V_N$ est obtenue, telle que pour toute fonction test $v_N \in V_N$, l'égalité suivante soit satisfaite :

$$a(T_N(\mu), v_N; \mu) = f(v_N; \mu). \tag{4}$$

Une discussion est faite sur la manière de construire cet espace V_N, qui est permis grâce à une bornée d'erreur certifiée Δ_N qui assure la précision des résultats :

$$\|T(\mu) - T_N(\mu)\|_V \leqslant \Delta_N(\mu). \tag{5}$$

Une seconde méthode de réduction d'ordre est présentée, la méthode des bases réduites non intrusives, qui permet de répondre à certaines limitations de la méthode des bases réduites classiques. Cette méthode est basée sur l'utilisation de deux niveaux de discrétisation, un fin V_h et un grossier V_H. Ainsi, les calculs haute fidélité sont effectués sur la grille grossière et la solution est reconstruite sur la grille fine grâce à des opérations d'interpolation.

Ces deux méthodes ont un paradigme similaire : une étape coûteuse dite *hors-ligne* est réalisée pour construire l'espace réduit V_N. Cette première étape n'est faite qu'une seule fois. Dans une seconde étape, dite *en ligne*, la solution réduite T_N est obtenue en résolvant le problème réduit (4), cette étape est beaucoup moins coûteuse en temps de calcul et permet de calculer des quantités d'intérêt en temps réel.

Chapitre 4 : Résolution de problème elliptique en présence d'une source de type Dirac La méthode des bases réduites introduite dans le chapitre précédent nous donne des résultats cohérents avec la théorie usuelle, mais dans le cas où des sorties d'intérêts ponctuelles sont considérées, nous sommes amenés à résoudre des problèmes avec un terme de type Dirac, qui invalide l'utilisation de certains théorèmes de convergence. Dans ce chapitre, nous nous intéressons à la résolution de problèmes elliptiques du type

$$\Delta u = \delta_0 \quad \text{dans } \Omega, \tag{6}$$

avec différents types de conditions aux bords : Dirichlet, Neumann, Robin ainsi que des conditions mixtes. Le logiciel Feel++ permet de résoudre ce type de problème. Nous nous intéressons particulièrement à la convergence de la solution numérique sous différentes conditions aux bords, mais aussi pour des positions différentes du Dirac.

Chapitre 5 : Analyse de sensibilité Dans ce chapitre, nous présentons les résultats de l'analyse de sensibilité qui a été menée sur le modèle de transfert de chaleur dans l'œil humain, introduit dans le Chapitre 1. L'objectif de cette étude est de comprendre comment les paramètres du modèle influent sur les quantités d'intérêt, et comment ces quantités peuvent être utilisées pour diagnostiquer des maladies. La Figure 5 montre la méthodologie suivie pour cette étude.

Deux études sont menées, une analyse déterministe suivant des résultats de précédentes études, pour permettre une comparaison avec le modèle développé dans cette thèse. Une seconde étude stochastique est menée, avec le calcul des indices de Sobol, permettant de quantifier l'impact des paramètres sur les quantités d'intérêt.

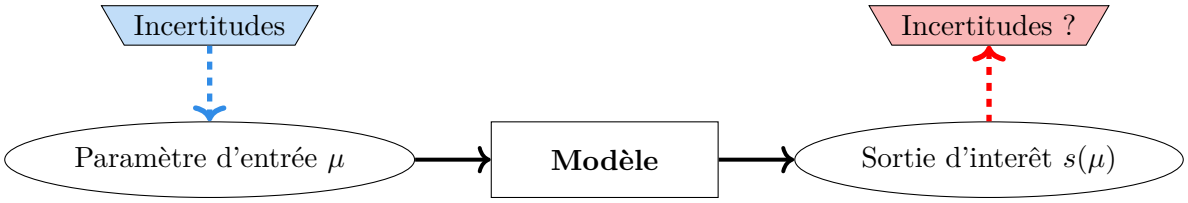


Figure 5: Propagation d'incertitudes.

Chapitre 6 : Modélisation couplée des flux d'humeur aqueuse et de transfert de chaleur Dans ce chapitre, nous présentons un modèle couplé de transfert de chaleur et de flux d'humeur aqueuse dans l'œil humain. Ce modèle est donné par le couplage des équations de transfert de chaleur avec les équations de Navier-Stokes pour l'humeur aqueuse, sous l'approximation de Boussinesq :

Équations de Navier-Stokes — Approximation de Boussinesq

$$\rho(\vec{u}\cdot\nabla\vec{u}) - \mu\nabla^2\vec{u} + \nabla p = -\rho\beta(T - T_{\text{ref}})\vec{g} \quad \text{dans } \Omega_{\text{AH}}, \tag{7a}$$

Conservation de la masse

$$\nabla\cdot\vec{u} = 0 \quad \text{dans } \Omega_{\text{AH}}, \tag{7b}$$

Transfert thermique

$$\rho C_p\vec{u}\cdot\nabla T - k\nabla^2 T = 0 \quad \text{dans } \Omega. \tag{7c}$$

La méthodologie des éléments finis est utilisée pour résoudre ce problème couplé, mais nécessite une utilisation de préconditionneurs adaptés pour résoudre ce type d'équations. À partir de ce modèle numérique, nous présentons des résultats de simulations, comprenant une étude de l'impact de la position du sujet et aussi de la température extérieure sur diverses quantités d'intérêt telles que la vitesse ou la pression du fluide, mais aussi la contrainte de cisaillement exercée sur les parois de la chambre antérieure.

Chapitre 7 : Contributions logicielles Dans ce chapitre, nous présentons les contributions logicielles qui ont été développées dans le cadre de cette thèse. Ces contributions s'inscrivent dans une démarche de reproductibilité scientifique et de science ouverte, conformément aux principes de transparence et de validation exposés dans [Cam+24]. Ces contributions ont été faites à la librairie open-source Feel++[1] [Pru+24b].

Chapitre A : Effet du refroidissement de la surface oculaire sur la sédimentation des cellules endothéliales dans la thérapie par injection de cellules : Perspectives par la Mécanique des fluides numérique Ce chapitre annexe reprend le travail qui a été proposé à la conférence ARVO 2025, qui porte sur l'étude de l'effet du refroidissement de la surface oculaire sur la sédimentation des cellules endothéliales dans la thérapie par injection de cellules, en vue d'une application clinique à la thérapie de sédimentation des cellules endothéliales. Il s'agit d'un extension du travail effectué dans le Chapitre 6, en collaboration avec Vincent Chabannes (Cemosis, Université de Strasbourg), Giovanna Guidoboni (University of Maine) Christophe Prud'homme (Université de Strasbourg) Marcela Szopos (Université Paris Cité), et and Sangly P. Srinivas (School of Optometry, Indiana University).

Chapitre B : Simulation des interactions dans les communautés microbiennes à l'aide de PINN Un chapitre annexe est consacré à la présentation des travaux réalisés lors du CEMRACS 2023, qui porte sur la simulation numérique des interactions de populations microbiennes à l'aide de réseaux de neurones. Ce travail a été réalisé en collaboration avec Javan

[1] https://github.com/feelpp/feelpp

Hossie (Université d'Orléans), Béatrice Laroche (INRAE), Thibault Malou (INRAE), Lucas Perrin (Universität Konstanz) et Lorenzo Sala (INRAE).

Les micro-organismes forment des communautés complexes, des microbiotes, qui influencent divers aspects de la santé de leur hôte. Le modèle de Lotka-Volterra généralisé (GLV) est couramment utilisé pour comprendre la dynamique des populations de micro-organismes, mais son application au microbiote se heurte à des difficultés dues au manque de données sur les bactéries et à la complexité des interactions. Ce travail préliminaire se concentre sur l'utilisation d'un *Physics-Informed Neural Network* (PINN) et de données synthétiques pour simuler l'évolution d'espèces bactériennes décrite par un modèle GLV. Cette approche est calibrée et testée sur plusieurs modèles qui diffèrent par leur taille et leur comportement dynamique.

L'avantage des PINN est qu'ils peuvent être entraînés à la fois sur des données, mais aussi sur des équations différentielles, ce qui permet de capturer le comportement dynamique des populations microbiennes.

Les réseaux de neurones permettent de mettre en place une méthodologie similaire à celle des méthodes de réduction d'ordre du Chapitre 3, avec une première phase d'entraînement hors ligne et une seconde phase d'inférence en ligne.

Publications et présentations durant la période de thèse

Publications

- Thomas Saigre, Christophe Prud'homme et Marcela Szopos. "Model order reduction and sensitivity analysis for complex heat transfer simulations inside the human eyeball". en. In : *International Journal for Numerical Methods in Biomedical Engineering* 40.11 (sept. 2024), e3864. ISSN : 2040-7939, 2040-7947. DOI : 10.1002/cnm.3864. URL : https://onlinelibrary.wiley.com/doi/10.1002/cnm.3864.

- Paguiel Javan Hossie, Béatrice Laroche, Thibault Malou, Lucas Perrin, Thomas Saigre et Lorenzo Sala. "Surrogate modeling of interactions in microbial communities through Physics-Informed Neural Networks." Fév. 2024. URL : https://hal.inrae.fr/hal-04440736, à paraître dans *ESAIM: Proceedings and Surveys*.

- Thomas Saigre, Vincent Chabannes, Giovanna Guidoboni, Christophe Prud'homme, Marcela Szopos et Sangly P Srinivas. *Effect of Cooling of the Ocular Surface on Endothelial Cell Sedimentation in Cell Injection Therapy: Insights from Computational Fluid Dynamics*. 2024, soumis.

- Thomas Saigre, Vincent Chabannes, Christophe Prud'homme et Marcela Szopos. "A coupled fluid-dynamics-heat transfer model for 3D simulations of the aqueous humor flow in the human eye". In preparation.

- Silvia Bertoluzza, Christophe Prud'homme, Thomas Saigre et Marcela Szopos. "Low to high order finite element resolution for elliptic problems in the presence of a Dirac source term". In preparation.

Proceedings de conférence à comité de lecture

- Thomas Saigre, Christophe Prud'homme, Marcela Szopos et Vincent Chabannes. "A coupled fluid-dynamics-heat transfer model for 3D simulations of the aqueous humor flow in the human eye". In : *8th International Conference on Computational and Mathematical Biomedical Engineering – CMBE2024 Proceedings*. T. 2. Arlington (Virginia), United

States : P. Nithiarasu et R. Löhner (Eds.), juin 2024, p. 508-512. ISBN : 978-0-9562914-7-9. URL : https://hal.science/hal-04558924.

Présentations dans des conférences internationales

- 26 juin 2024. **8th International Conference on Computational and Mathematical Biomedical Engineering**, Arlington, USA : A coupled fluid-dynamics-heat transfer model for 3D simulations of the aqueous humor flow in the human eye.
- 13 septembre 2023. **CompBioMed Conference 2023**, Munich, Allemagne : Model Order Reduction and Sensitivity Analysis for complex ocular simulations inside the human eyeball.
- 28 février 2023. **SIAM Conference on Computational Science and Engineering**, Amsterdam, Pays-Bas : Model Order Reduction for Complex Ocular Simulations Inside the Human Eyeball.

Présentations orales dans des conférences nationales, groupes de travail et séminaires

- 13 décembre 2024. **Groupe de travail Modélisation, Analyse et Simulation**, Paris, France : Mathematical modeling, simulation and reduced order modeling of ocular flows and their interactions: Building the Eye's Digital Twin.
- 28 mai 2024. **CANUM**, Île de Ré, France : Low to high order finite element resolution for elliptic problems in the presence of a Dirac source term.
- 30 janvier 2024. **Journées Numériques de Besançon**, Besançon, France : Model order reduction and sensitivity analysis for complex heat transfer simulations inside the human eyeball.
- 16 janvier 2024. **Séminaire EDP à l'IRMA**, Strasbourg, France : Modèle de réduction d'ordre et analyse de sensibilité pour les simulations de transfer de chaleur de chaleur à l'intérieur du globe oculaire humain.
- 23 août 2023. **CEMRACS 2023**, Marseille, France : Estimation of interactions in microbial communities via a neural network-based generalized smoothing algorithm.
- 1er décembre 2022. **Séminaire Jeunes Chercheurs**, Reims, France : Modèles de réduction d'ordre des simulations oculaires complexes dans l'œil humain.

Présentations de posters

- 5 octobre 2022. **Journée Math Bio Santé 2022**, Besançon, France : Model order reduction for complex ocular simulations inside the human eyeball.
- 13 juin 2022. **CANUM**, Évian-les-Bains, France : Modèles de réduction d'ordre des simulations oculaires complexes dans l'œil humain.

Rapports techniques

- Christophe Prud'homme, Pierre Alliez, Vincent Chabannes, Rudy Chocat, Emmanuel Franck, Vincent Fraucher, Floriant Faucher, Clément Gauchy, Christos Georgiadis, Luc Giraud, Frédéric Hecht, Guillaume Helbecque, Pierre Jolivet,

Olivier Jamond, Pierre Ledac, Nouredine Melab, Victor Michel-Dansac, Frédéric Nataf, Lucas Palazzolo, Yannick Privat, Thomas Saigre-Tardif, El-Ghazali Talbi, Pierre Henri Tournier, Christophe Trophime, Céline Van Landeghem et Raphael Zanella. *Benchmarking analysis report*. Deliverable Report D7.1. tex.version: v1.1.0. U. Strasbourg, INRIA, CEA, U. Lille, U. Luxembourg, CNRS, Sorbonne U., oct. 2024. URL : https://github.com/numpex/exa-ma-d7.1/.

Logiciels et jeux de données

- Christophe Prud'homme, Vincent Chabannes, Thomas Saigre, Christophe Trophime, Luca Berti, Abdoulaye Samaké, Céline Van Landeghem, Marcela Szopos, Laetitia Giraldi, Silvia Bertoluzza et Yvon Maday. *feelpp/feelpp: Feel++ Release V111 preview.10*. Juill. 2024. DOI : 10.5281/ZENODO.12742155. URL : https://zenodo.org/doi/10.5281/zenodo.12742155
- Vincent Chabannes, Christophe Prud'homme, Thomas Saigre, Sala Lorenzo, Marcela Szopos et Christophe Trophime. *A 3D geometrical model and meshing procedures for the human eyeball*. Sept. 2024. DOI : 10.5281/ZENODO.13829740. URL : https://zenodo.org/doi/10.5281/zenodo.13829740
- Thomas Saigre, Christophe Prud'homme, Marcela Szopos et Vincent Chabannes. *Mesh and configuration files to perform coupled heat+fluid simulations on a realistic human eyeball geometry with Feel++*. Oct. 2024. DOI : 10.5281/ZENODO.13886143. URL : https://zenodo.org/doi/10.5281/zenodo.13886143
- Thomas Saigre, Christophe Prud'homme et Marcela Szopos. *Model order reduction and sensitivity analysis for complex heat transfer simulations inside the human eyeball*. Oct. 2024. DOI : 10.5281/ZENODO.13907890. URL : https://zenodo.org/doi/10.5281/zenodo.13907890

Introduction

With the development of new technologies, the interest on mathematical models in biology and medicine has greatly increased. Indeed, the use of computational models can help to understand complex biological systems, and to predict the behavior of these systems under various conditions.

Because of the special connection to the brain, the eye is generally considered as a window to the brain [Kel+24; Gui+20], that offers non-invasive access to a large set of biomarkers that could help for instance in the early diagnosis of neurodegenerative diseases, or cardiovascular conditions [Che+24]. However, characterizing ocular biomarkers as surrogates of cerebral or systemic vascular status is far from trivial [Gui+20]. Clinical measurements are influenced by many factors that vary among individuals and cannot be isolated *in vivo*, thereby posing serious challenges for the interpretation of such measurements.

As pointed out in [Fly+23], the current distribution of scientific evidence sources is evenly split between human, animal, and laboratory studies, with only 1% relying on computational models. However, projections based on Computational Modeling and Simulation suggests a future shift towards a heavier reliance on computer simulations, expected to account for 45% of the evidence, as pointed out in Figure 6. This trend reflects the increasing integration of advanced computational techniques in medical device development, reducing dependence on traditional methods.

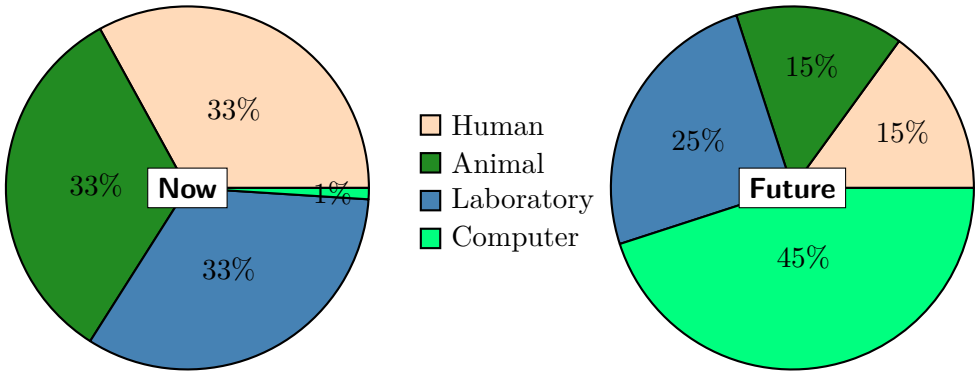


Figure 6: Sources of scientific evidence, from [Fly+23].

Indeed, computational models are crucial for scientific research, especially in biology and medicine due to the fact they can decrease the number of physical tests necessary for product development or disease understanding, thus reducing tremendously the economic cost and relying less on animal and human data with a greater influence of data from virtual patients.

The study of biofluids using mathematical and computational models is not a novel approach. The cardiovascular system was the first human system to be examined in this way, beginning with a purely mathematical perspective in the 18th century [Eul75; Hal+33], and later from a computational standpoint [NVB63]. Subsequently, other organs such as the lungs [Mau13], brain [Mar+22], *etc.* were also investigated.

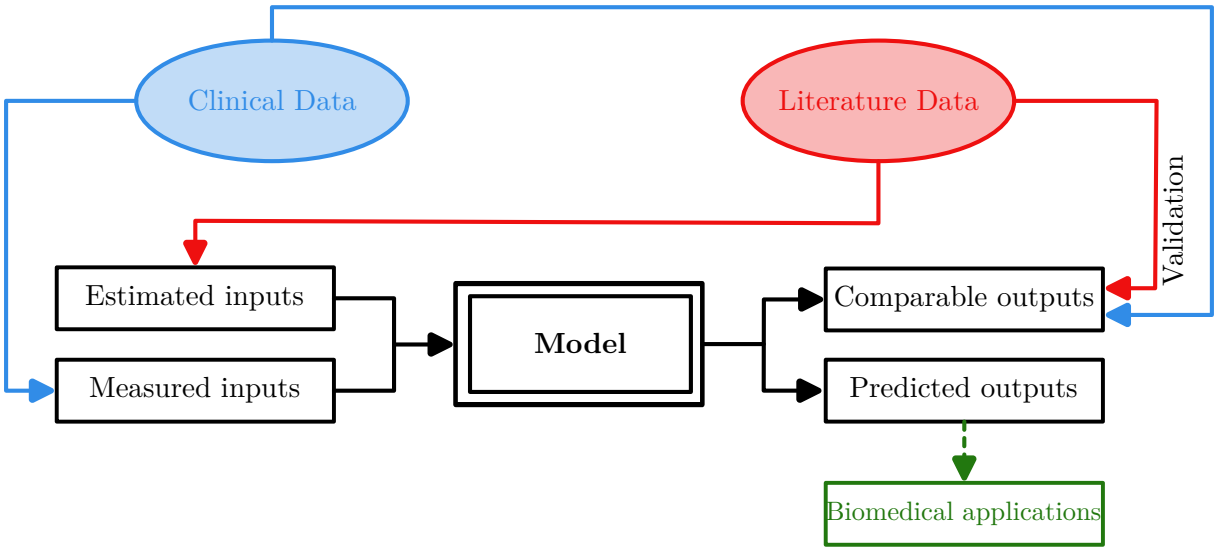


Figure 7: Methodology for the development of patient-specific models, adapted from [Sal+23].

To be complementary to traditional research, mathematical and computational models should stem from clinical data, combined with biophysical knowledge, and should be validated to assure their reliability. This kind of models has already been developed for specific biomedical applications, such as the cardiovascular system [Upd+17; Art+21] or cerebral hemodynamics [Mar+22]. Figure 7 illustrates the methodology for the development of patient-specific models, enlightening the importance of data integration from both clinical and numerical standpoints.

Interest in the eye, however, is more recent but has seen significant growth in the past few decades. Several ocular diseases are still not well understood by the scientific community, and the development of computational models can help to better understand the underlying mechanisms [Gui+22]. Several models have been proposed, as reviewed in [GHS19; HRE23]. In particular, a model accounting for the combined effects of ocular blood flow and different ocular tissues was proposed in [Sal+23]. The Ocular Mathematical Virtual Simulator (OMVS) [Sal19; Sal+23], is a mathematical and computational framework that allows to simulate multi-physic phenomena involved in the eye, from patient-specific data. To incorporate inherent uncertainties and variability, an uncertainty propagation and sensitivity analysis on the component simulating the fluid flows in the eye was developed in [PSS21]. Such computational methods and *in silico* frameworks can be used for virtual patients, and to develop patient-specific models, or patient eye digital twins [Sal+21a].

In this challenging context, the present thesis is part of the *Eye2Brain* project [Sal16], which has the ambitious objective of connecting the cerebral and ocular environments and contributing in the long term to a better understanding of neurodegenerative diseases [Gui+20]. Although the study of heat transfer in the eye may not seem directly related to the brain, this work aligns with the project by enhancing our understanding of physiological processes within the ocular environment, which could provide valuable insights into systemic and neurological conditions, as well as on cardiovascular aspects [Kel+24].

While previous studies have focused on the interplay between hemodynamics and biomechanics in the eye, in this work we concentrate on heat transfer in the eye and its coupling with the fluid flow of the aqueous humor. Several mathematical models have been proposed for the study of various aspects of temperature distribution in the human eyeball and its variations due to the flow field in the anterior and posterior chambers. In particular, heat transfer in the eye has been studied in [Sco88; NO06; NO07; ON08; Li+10], as well as the fluid flow of aqueous humor in the anterior chamber and its coupling with heat transfer [HB02; Wan+16; Mur+23]. All these

aspects are important in the perspective of eye-brain connections [Gui+20], aiming to predict the distribution of stress and strains occurring during diagnostic or therapeutic procedures, such as hyperthermia treatment of eye tumors [Li+10] or cell injection treatment to cure bullous keratopathy [Kin+18].

While invasive studies on animals have been conducted [PW05], non-invasive measurements on human subjects are scarce, complex to perform, and may yield inaccurate results [RF77]. Most studies focus on temperature measurements at the eye's surface [Map68; EYB89] but report significant differences and identify several sources of uncertainty. Alternatively, numerical simulations can provide complementary information, eventually not available to experimental measurements but critical for drug delivery and pre- and post-surgery assessments such as the wall shear stress.

Sensitivity analysis [Raz+21], particularly through the use of Sobol' indices [Sob93], allows for the systematic exploration of how variations in the model parameters contribute to the overall variability of the outcomes. This helps in identifying which parameters are the most influential and understanding their interactions.

The computational effort required for solving realistic problems in this context is considerable. First, representing realistic 3D geometries involves handling a significant amount of data, which increases the complexity of the simulations. Second, the inclusion of complex multiphysics models, such as coupling heat transfer with fluid dynamics, adds further computational challenges. Additionally, incorporating uncertainties in the input parameters through uncertainty quantification and sensitivity analysis necessitates running multiple simulations, each with varying parameters. This process dramatically increases the computational burden, making the use of high-performance computing (HPC) resources essential, and to develop efficient reduced order models to mitigate the computational cost [QMN16; Pru+02].

In line with these computational demands, another aspect of this thesis is part of the ExaMA (*Methods and Algorithms for Exascale*) project of the PEPR NumPEx program [Num24], concentrating on the exascale challenges of numerical methods and algorithms.

Contributions

In this thesis, we develop a three-dimensional mathematical and computational model of the human eye, focusing initially on heat transfer and subsequently on its coupling with the fluid flow of the aqueous humor. The presented models incorporate numerous biomechanical and geometrical parameters. Our primary focus is on biomechanical parameters, spanning a broad range, including potential extreme conditions. The variation of these parameters can significantly impact the results. To quantify these impacts, we establish a framework for forward uncertainty quantification, complemented by sensitivity analysis.

A rigorous validation step is essential to ensure the reliability of these results. We compare our findings with data available in the literature, derived either from measurements on healthy subjects [EYB89] or from simulations [NO06; NO07; Li+10; KS10], to confirm the validity of our approach.

While Sobol' indices effectively measure parameter impacts and interactions, the complexity and significant computational time required by our model pose considerable challenges. To address this, we adopt the certified Reduced Basis Method (RBM) [Pru+02; QMN16], a mathematical approach designed to speed up simulations of complex systems without sacrificing accuracy. This approach is particularly well-suited for uncertainty quantification, sensitivity analysis and real-time simulations, as it allows for the rapid evaluation of the model's response to varying parameters. In essence, the RBM creates a simplified version of the original model by identifying and retaining only its most essential features, allowing us to construct a reduced model that

maintains its 3D nature while significantly lowering computational costs and providing reliable results.

The RBM operates in two stages: an *offline phase*, during which the simplified version (or reduced space) is built, and an *online phase*, where the reduced model is used to compute solutions efficiently. Although the offline phase is computationally expensive, it is performed only once. Subsequently, the online phase enables rapid simulations, making this method particularly suited for scenarios requiring repeated evaluations, such as sensitivity analysis or uncertainty quantification. This methodology aligns with the workflow observed in patient-specific mathematical models applied to biomedical problems, encompassing data integration, model derivation, numerical solving, validation, and uncertainty quantification. Such paradigms are well-established in fields like cardiovascular simulations and cerebral hemodynamics. In ophthalmology, a similar approach is needed due to the richness and heterogeneity of the available data, which call for innovative methods for diagnosis and monitoring.

To complement the RBM, we also implement the non-intrusive reduced basis method (NIRB) [CM09b; CM09a; Gro22], a variant of the RBM that is particularly useful when model complexity prevents direct application of the latter. Unlike the standard RBM, the NIRB uses a two-level discretization strategy, where a simplified surrogate model is constructed and solved without requiring direct modifications to the original simulation code. The offline/online paradigm remains applicable in this context, ensuring computational efficiency even for highly complex models.

To achieve fast quadratic convergence in the output, the RBM relies on a dual problem formulation, wherein each output is expressed as a linear functional of the solution. However, in certain cases of interest, as the one considered in this thesis, the output functional may involve pointwise evaluations of the solution. This leads to a dual problem featuring a Dirac source term δ_x at the point x where the solution is evaluated for the output of interest. In such cases, the conventional theory of RBM does not apply, but our numerical results exhibit favorable convergence properties, consistent with theoretical predictions in the continuous case. Theoretical and numerical aspects of the Laplacian problem with a Dirac source term have been studied in [KW14; Ber+18]. While theoretical results are derived under specific assumptions, we observe similar outcomes even when these conditions are not strictly met. We provide a theoretical review and a numerical exploration of the Laplacian model with a prescribed Dirac source term under various boundary conditions and discretization orders.

Additionally, we extend the initial thermal model by incorporating the flow of the aqueous humor (AH) in the anterior chamber (AC) and posterior chamber (PC) of the eye, coupling it with the thermal processes throughout the eyeball. This coupling is critical because the motion of AH plays a significant role in heat distribution and intraocular pressure regulation, both of which are vital for overall ocular health. Furthermore, the wall shear stress generated by the AH flow is an important biomechanical factor, influencing ocular tissue health and potentially impacting the drainage pathways, which are relevant to conditions such as glaucoma. Previous studies have investigated various aspects of this complex interaction. For example, [HB02; Wan+16; Mur+23] modeled flow coupled with heat transfer in the AC and PC, while [Sac+23] examined the impact of pressure on AH flow and drainage. Other works [ON08; BBS20; Abd+21; Dvo+22] explored the thermo-fluid dynamics of AH flow in the AC under specific boundary conditions. However, these studies often focus on simplified geometries or do not fully couple heat transfer within the entire eyeball. To efficiently solve the numerical problem, we employ advanced multiphysics preconditioning techniques, combining preconditioner such as GAMG [Bal+24] or efficient block preconditioners for the fluid equations [ESW14], to efficiently solve the conjugate heat transfer problem. Additionally, within the context of HPC, we investigate the scalability properties of the model.

Over these three years of intensive development, substantial code contributions and datasets

have been produced. Ensuring that these results can be independently validated and reproduced is critical [Cam+24]. To this end, all contributions have been published in open-access repositories.

Outline of the manuscript

The manuscript is structured as follows: In Chapter 1, we introduce the anatomical structure of the eye, alongside the medical context and challenges that motivate this study. This chapter also presents the underlying model for heat transfer and the associated mathematical framework. Chapter 2 focuses on the discretization of both the geometrical and biophysical models. In particular, it details the implementation of the Finite Element Method. In Chapter 3, we explore Model Order Reduction (MOR) techniques designed to mitigate the computational expense of solving the full model, enabling a thorough sensitivity analysis. Specifically, we present two methodologies: the Reduced Basis Method and the Non-Intrusive Reduced Basis Method, with corresponding results derived for the thermal model of the eye. The latter method has been developed in collaboration with Ali Elarif (Cemosis, Université de Strasbourg). Chapter 4 addresses a particular issue concerning the computation of pointwise quantities of interest within the MOR framework for elliptic problems, particularly those involving a Dirac source term. The work presented in this chapter is the result of a collaboration with Silvia Bertoluzza (CNR). The frameworks and the results of the sensitivity analysis, applied to the models developed throughout the manuscript, are presented in Chapter 5. In Chapter 6, we extend the initial thermal model by incorporating the dynamics of aqueous humor flow in both the anterior and posterior chambers of the eye, in collaboration with Vincent Chabannes (Cemosis, Université de Strasbourg). A dedicated chapter, Chapter 7, details the software contributions developed during this thesis work. Finally, we present the conclusions and the outlook for future work in Chapter 8.

Moreover, we present in Appendix A a work submitted to the ARVO 2025 Meeting, which is an extension of Chapter 6. Lastly, Appendix B documents my participation in the CEMRACS 2023 project, where we investigated microbial community interactions using a neural network-based generalized smoothing algorithm. This project was proposed by Lorenzo Sala, Béatrice Laroche, Thibault Malou (INRAE) and was in collaboration with Paguiel Javan Hossie (Université d'Orléans) and Lucas Perrin (Universität Konstanz).

Feel++

All the methods presented and developed in this manuscript have been implemented in the framework of the Feel++ library [Pru+24b], the *Finite Element Embedded Library in C++*. As the models presented are described by Partial Differential Equations, the Finite Element Method (FEM) is used to solve them.

Feel++ is an open-source library that provides a high-level interface to the FEM, and allows solving complex problems in a simple way, by the use of a syntax close to the mathematical formulation. It has been described in [Pru+12], the latest version being `v0.111` [Pru+24b] at the time of redaction of this manuscript. The documentation is available online[2], as well as the source code on GitHub[3]. Recently, the library has been ported to the Python language, allowing to use the library in a more user-friendly way, and to benefit from the numerous libraries available in Python.

To efficiently solve the problems, Feel++ is relying on PETSc [Bal+24], and efficient meshing libraries such as GMSH [GR09] and SALOME [RBG17; CAS22]. One can also note that the

[2] https://docs.feelpp.org/
[3] https://github.com/feelpp/feelpp/

parallelization process is transparent to the user, allowing to easily run the code on a personal computer up to a EuroHPC supercomputer [Pru+24a].

During the thesis, some developments have been made in the Feel++ library. First, the implementation of the Model Order Reduction applied to the thermal model of the eye, as presented in Chapter 3, as well as the sensitivity analysis, presented in Section 5.3. Then the implementation of the NIRB method, presented in Section 3.2, has been done in collaboration with Ali Elarif. A more in-depth discussion about these contributions is presented in Chapter 7.

All the results presented in this manuscript are obtained on one of the six nodes of `Gaya`, the computing resource cluster at IRMA and Cemosis. The node is equipped with two CPU `AMD EPYC 7713 64-Core`, with 64 cores each, and 512 Go of RAM.

Publications and presentations during the thesis period

Publications

- Thomas Saigre, Christophe Prud'homme, and Marcela Szopos. "Model order reduction and sensitivity analysis for complex heat transfer simulations inside the human eyeball". en. In: *International Journal for Numerical Methods in Biomedical Engineering* 40.11 (Sept. 2024), e3864. ISSN: 2040-7939, 2040-7947. DOI: 10.1002/cnm.3864. URL: https://onlinelibrary.wiley.com/doi/10.1002/cnm.3864.

- Paguiel Javan Hossie, Béatrice Laroche, Thibault Malou, Lucas Perrin, Thomas Saigre, and Lorenzo Sala. "Surrogate modeling of interactions in microbial communities through Physics-Informed Neural Networks." Feb. 2024. URL: https://hal.inrae.fr/hal-04440736, to appear in *ESAIM: Proceedings and Surveys*.

- Thomas Saigre, Vincent Chabannes, Giovanna Guidoboni, Christophe Prud'homme, Marcela Szopos, and Sangly P Srinivas. *Effect of Cooling of the Ocular Surface on Endothelial Cell Sedimentation in Cell Injection Therapy: Insights from Computational Fluid Dynamics*. 2024, submitted.

- Thomas Saigre, Vincent Chabannes, Christophe Prud'homme, and Marcela Szopos. "A coupled fluid-dynamics-heat transfer model for 3D simulations of the aqueous humor flow in the human eye". In preparation.

- Silvia Bertoluzza, Christophe Prud'homme, Thomas Saigre, and Marcela Szopos. "Low to high order finite element resolution for elliptic problems in the presence of a Dirac source term". In preparation.

Peer-reviewed conference proceedings

- Thomas Saigre, Christophe Prud'homme, Marcela Szopos, and Vincent Chabannes. "A coupled fluid-dynamics-heat transfer model for 3D simulations of the aqueous humor flow in the human eye". In: *8th International Conference on Computational and Mathematical Biomedical Engineering – CMBE2024 Proceedings*. Vol. 2. Arlington (Virginia), United States: P. Nithiarasu and R. Löhner (Eds.), June 2024, pp. 508–512. ISBN: 978-0-9562914-7-9. URL: https://hal.science/hal-04558924.

Presentations at international conferences

- 26th June 2024. **8th International Conference on Computational and Mathematical Biomedical Engineering**, Arlington, USA: A coupled fluid-dynamics-heat transfer model for 3D simulations of the aqueous humor flow in the human eye.
- 13th September 2023. **CompBioMed Conference 2023**, Munich, Germany: Model Order Reduction and Sensitivity Analysis for complex ocular simulations inside the human eyeball.
- 28th February 2023. **SIAM Conference on Computational Science and Engineering**, Amsterdam, the Netherlands: Model Order Reduction for Complex Ocular Simulations Inside the Human Eyeball.

Oral presentations at national conferences, working groups and seminars

- 13th December 2024. **Groupe de travail Modélisation, Analyse et Simulation**, Paris, France : Mathematical modeling, simulation and reduced order modeling of ocular flows and their interactions: Building the Eye's Digital Twin.
- 28th May 2024. **CANUM**, Île de Ré, France: Low to high order finite element resolution for elliptic problems in the presence of a Dirac source term.
- 30th January 2024. **Journées Numériques de Besançon**, Besançon, France: Model order reduction and sensitivity analysis for complex heat transfer simulations inside the human eyeball.
- 16th January 2024. **Séminaire EDP à l'IRMA**, Strasbourg, France: Modèle de réduction d'ordre et analyse de sensibilité pour les simulations de transfer de chaleur de chaleur à l'intérieur du globe oculaire humain.
- 23rd August 2023. **CEMRACS 2023**, Marseille, France: Estimation of interactions in microbial communities via a neural network-based generalized smoothing algorithm.
- 1st December 2022. **Séminaire Jeunes Chercheurs**, Reims, France: Modèles de réduction d'ordre des simulations oculaires complexes dans l'œil humain.

Poster presentations

- 5th October 2022. **Journée Math Bio Santé 2022**, Besançon, France: Model order reduction for complex ocular simulations inside the human eyeball.
- 13th June 2022. **CANUM**, Évian-les-Bains, France: Modèles de réduction d'ordre des simulations oculaires complexes dans l'œil humain.

Technical reports

- Christophe Prud'homme, Pierre Alliez, Vincent Chabannes, Rudy Chocat, Emmanuel Franck, Vincent Fraucher, Floriant Faucher, Clément Gauchy, Christos Georgiadis, Luc Giraud, Frédéric Hecht, Guillaume Helbecque, Pierre Jolivet, Olivier Jamond, Pierre Ledac, Nouredine Melab, Victor Michel-Dansac, Frédéric Nataf, Lucas Palazzolo, Yannick Privat, Thomas Saigre-Tardif, El-Ghazali Talbi, Pierre Henri Tournier, Christophe Trophime, Céline Van Landeghem, and Raphael Zanella. *Benchmarking analysis report*. Deliverable Report D7.1. tex.version: v1.1.0. U. Strasbourg, INRIA, CEA, U. Lille, U. Luxembourg, CNRS, Sorbonne U., Oct. 2024. URL: https://github.com/numpex/exa-ma-d7.1/.

Software and datasets

- Christophe Prud'homme, Vincent Chabannes, Thomas Saigre, Christophe Trophime, Luca Berti, Abdoulaye Samaké, Céline Van Landeghem, Marcela Szopos, Laetitia Giraldi, Silvia Bertoluzza, and Yvon Maday. *feelpp/feelpp: Feel++ Release V111 preview.10*. July 2024. DOI: 10.5281/ZENODO.12742155. URL: https://zenodo.org/doi/10.5281/zenodo.12742155

- Vincent Chabannes, Christophe Prud'homme, Thomas Saigre, Sala Lorenzo, Marcela Szopos, and Christophe Trophime. *A 3D geometrical model and meshing procedures for the human eyeball*. Sept. 2024. DOI: 10.5281/ZENODO.13829740. URL: https://zenodo.org/doi/10.5281/zenodo.13829740

- Thomas Saigre, Christophe Prud'homme, Marcela Szopos, and Vincent Chabannes. *Mesh and configuration files to perform coupled heat+fluid simulations on a realistic human eyeball geometry with Feel++*. Oct. 2024. DOI: 10.5281/ZENODO.13886143. URL: https://zenodo.org/doi/10.5281/zenodo.13886143

- Thomas Saigre, Christophe Prud'homme, and Marcela Szopos. *Model order reduction and sensitivity analysis for complex heat transfer simulations inside the human eyeball*. Oct. 2024. DOI: 10.5281/ZENODO.13907890. URL: https://zenodo.org/doi/10.5281/zenodo.13907890

Chapter 1

Mathematical modeling of heat transfer mechanisms in the human eye

In this chapter, we focus on the modeling of the human eyeball. In the first section, we describe the anatomy and physiology of the eye, focusing on the parts that are relevant for the studies that will be performed in the sequel. Next, we develop the connections with the general pharmacological aspects, that are of interest in applications, especially in the context of heat transfer in the human eye, and its coupling with aqueous humor flow in the anterior and posterior chambers. Finally, we present the geometrical and physical models that were considered in the present work, with a particular emphasis on the heat transfer phenomena in the eye, stemming from previous works [Sco88; NO06; NO07]. Chapter 6 will be dedicated to the coupling of the heat transfer in the eye with the aqueous humor flow. We develop a mathematical framework that will be employed in the context of the finite element method, and we present the continuous model and its variational formulation.

This chapter is organized as follows: in Section 1.1, we present the anatomy of the human eyeball, focusing on the parts that are relevant for the heat transfer studies that will be performed in the sequel, while Section 1.2 focuses on the medical challenges associated with the work developed in this manuscript. Then, in Section 1.3 and Section 1.4 present respectively the geometrical and biophysical models that were considered in the present work. Finally, in Section 1.5, we present the mathematical model, showing the continuous model and its variational formulation.

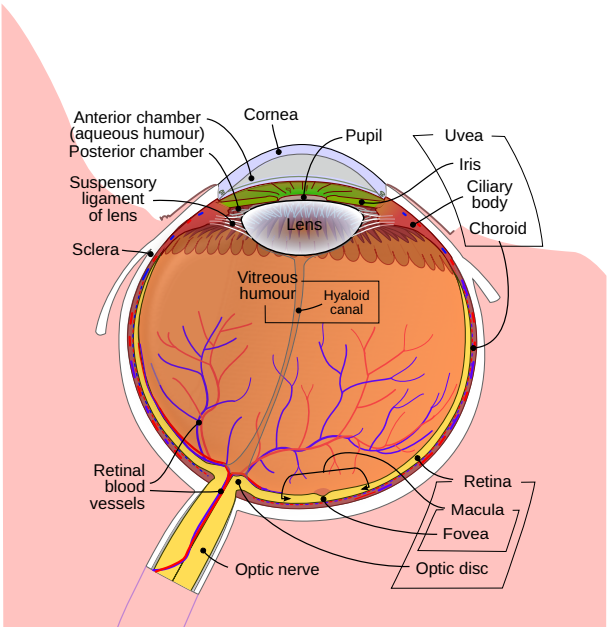


Figure 1.1: Schematic diagram of the human eye. Courtesy of Rhcastilhos, on Wikipedia.

1.1 Anatomy and physiology of the human eye

The human eye is a complex organ that enables light perception and image formation. While this manuscript does not focus on the visual functions of the eye, understanding its anatomy and physiology is essential for developing accurate mathematical and computational models.

Figure 1.1 presents a schematic diagram of the human eyeball. The outermost layer of the eye consists of the sclera and the cornea. The *sclera*, commonly referred to as the "white of the eye," maintains the shape of the eyeball and is metabolically inactive. The *cornea*, a transparent front structure, interfaces with the external environment and allows non-invasive measurement of surface temperature [PW05; EYB89].

Beneath the sclera lies the *choroid*, a vascular layer responsible for supplying nutrients to the retina. At the front of the eye, the *iris* controls the amount of light entering the eye by regulating the pupil's size. The *lens*, located just behind the iris, is a transparent biconvex structure that, along with the cornea, focuses light onto the retina. Its shape is adjusted by ligaments to adapt focus.

The *retina* forms the innermost layer of the eye, converting light into electrical signals transmitted to the brain via the optic nerve. It is composed of several layers of cells, including photoreceptors, which convert light into electrical signals that are sent to the brain through the optic nerve. In this work, we do not focus on the light perception aspect of the eye, hence we consider the retina as a region composed of blood vessels. Between the lens and the retina lies the vitreous humor, a gel-like substance that allows light to pass through to the retina. In the anterior part of the eye, the aqueous humor (AH) fills the anterior and posterior chambers, located between the cornea and the lens.

At the back of the eye, the *lamina cribrosa* is a porous structure through which the optic nerve exits the eye. It connects the retina to the optic nerve and links the eyeball to the body's circulatory system, playing a crucial role in ocular physiology. Although it is not directly involved in the models presented here, the lamina cribrosa has been studied in related projects, such as Eye2brain [Sal+23; Sal19], due to its importance in the overall function of the eye.

The AH is a transparent fluid produced by the ciliary body that flows from the posterior

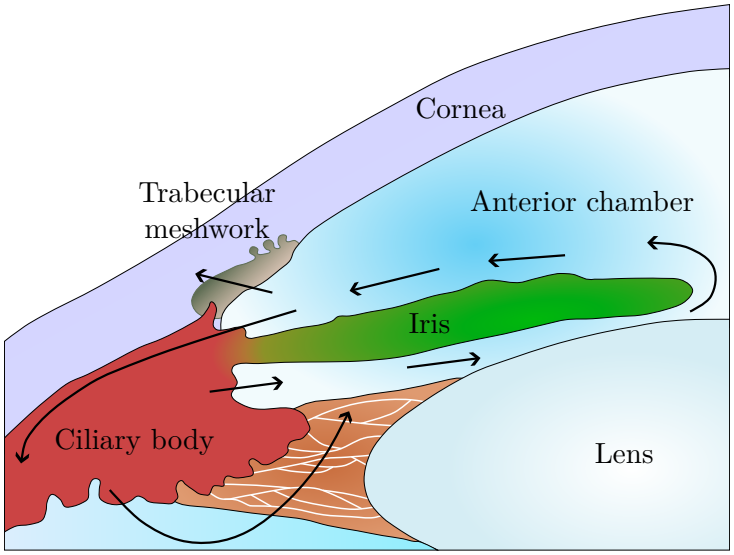


Figure 1.2: Production and drainage of AH in the front part of the eye, adapted from [RRK13].

chamber to the anterior chamber. It is drained via two pathways: the trabecular meshwork (TM) and the uveoscleral pathway [Dvo+19]. This flow maintains intraocular pressure (IOP) [RRK13]. Mechanically, the AH behaves as a Newtonian fluid, with its dynamics influenced by the pressure difference due to production and drainage, as well as convective effects driven by the temperature gradient between the corneal surface and the inner eye, which is at body temperature. Figure 1.2 illustrates the production and drainage pathways of AH.

1.2 Medical challenges and previous studies

The human eye is a highly complex organ, where multiple physiological factors interact, making it a challenging area for both medical research and clinical treatment. Understanding these interactions, particularly with respect to heat transfer, intraocular pressure, and aqueous humor flow, is crucial for addressing various ocular conditions and optimizing treatments.

The temperature of the eyeball may influence drug distribution in the eye, particularly due to age-related changes in tissue properties [Bha21]. Aging of the eye causes various geometrical and morphological changes, such as the reduction in the size of the anterior chamber, variations in lens thickness, and alterations in corneal curvature. These modifications impact both the temperature distribution and the aqueous humor (AH) flow, thereby affecting drug dispersion, as studied numerically in [Bha21].

Hyperthermia is a common treatment for ocular tumors [Li+10; Nar17], and a thorough understanding of heat transfer mechanisms can enhance the efficacy of ophthalmic treatments such as retinal laser therapy [Mas04]. Heat transfer is also essential when studying the effects of electromagnetic radiation on the eye [Hir+07; ON09]. The model initially introduced in [Sco88] to study temperature rises caused by infrared radiation exposure has since been expanded in various studies [NO06; NO07; ON08; Li+10].

Numerous parameters influence the heat transfer model. These parameters can be classified into two groups: *(i)* geometrical parameters, which depend on factors like eye anatomy, subject ethnicity [SLG06; Bou11], and age [BBS20], and *(ii)* physical parameters, which refer to the thermal properties of ocular tissues, such as thermal conductivity, heat capacity, and tissue density. Additionally, external factors like ambient temperature and air humidity may also affect

the eye's temperature distribution.

Another key factor in the eye is intraocular pressure (IOP), which is the pressure of the fluids inside the eye, and is determined by the balance between AH production and drainage. Parameters have also an impact on the IOP. Changes in IOP, particularly in the anterior chamber, can lead to conditions like glaucoma [WL10], a degenerative disease that results in the loss of retinal ganglion cells, ultimately causing blindness [RRK13].

While invasive studies on animals have provided some insights [PW05], non-invasive measurements in humans are rare, complex, and often yield inconsistent results [RF77]. Most studies focus on surface temperature measurements [Map68; EYB89], where variability is significant. IOP can be measured using the Goldmann Applanation Tonometer [Mos58], although this method is influenced by factors such as corneal thickness and curvature [Bha+02]. Numerical simulations, if properly validated against experimental data, can provide a useful alternative [EYB89; NO06; Li+10].

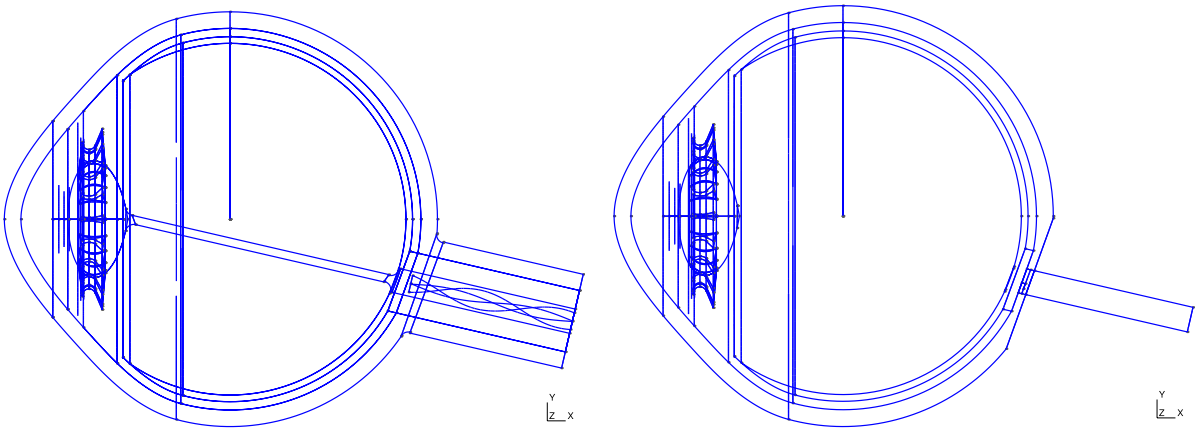
The present thesis aims to contribute to these developments, by means of a mathematical and computational modeling approach, combined with a sensitivity analysis study performed thanks to a model reduction technique. The comparison with data available in the literature, obtained either by measurement on healthy subjects [EYB89] or by other simulations [NO06; NO07; Li+10] will ensure the validity of the approach.

The model developed in this current chapter concerns only heat transfer in the eye, but in a second step, we will couple this model with the AH flow in the anterior and posterior chambers of the eye, as presented in Chapter 6. The understanding of the heat transfer in the eye is crucial to develop a reliable model of the AH flow, as the temperature of the eye has an impact on the flow of the AH [RRK13]. Some studies have been performed to couple the heat transfer with the AH flow [Can+02; FG06], as reported in [Dvo+19]. IOP also fluctuates over time, at various time scales [TTP19]: *(i)* at a short timescale, it is influenced by the cardiac cycle, the postural position of the subject of its activities, *(ii)* at a longer timescale, it is also influenced by aging related changes in the eye. Moreover, [SSS79; Har+23] indicate that while normal daily fluctuations in ambient temperature have a modest impact on intraocular pressure (IOP), larger temperature changes, such as those induced experimentally or encountered in extreme environments, can lead to significant alterations in IOP. These effects are often mediated by changes in aqueous humor dynamics and systemic factors such as blood pressure.

IOP is a key parameter since when abnormally elevated, it is a major risk factor for degenerative ocular diseases such as *glaucoma* [Kwo+09; WL10; RRK13]. This condition is known to be one of the most common causes of blindness [Kla98].

Mathematical and numerical investigation of the complex dynamic of the AH flow have been performed in previous studies [Dvo+22], also taking the coupling with heat transfer [ON08; Wan+16; Dvo+22; Sac+23]. Such model have proved to be useful tools for understanding the physiology of the eye, as well as the pathophysiology of ocular diseases [Dvo+19]. A better understanding of the AH flow is crucial to develop new treatments for glaucoma, as the IOP is a key parameter in the development of this disease [WL10; RRK13], or to model drug delivery in the eye [Ruf+24] and cell injection treatment to cure conditions such as bullous keratopathy [Kin+18].

A comprehensive understanding of the intricate relationships between heat transfer, aqueous humor dynamics, and intraocular pressure is essential for improving current therapeutic strategies and developing new treatments for various ocular diseases.



(a) Initial CAD before modifications. (b) CAD after modifications.

Figure 1.3: CAD of the human eyeball from the STEP file, before and after modifications with SALOME framework.

1.3 Geometrical model construction for the human eyeball

In this section, we describe the realistic three-dimensional geometry that will be used in the sequel. The model we employ in the present work stems from [Sal+23], and was constructed using a CAD (Computer Aided Design) module from SALOME [RBG17; CAS22]. As presented in Section 1.1, the eye is composed of several regions, which have different physical properties. In the present model, the optic nerve domain is assumed to be homogeneous, the contribution of the inner vessels is not directly taken into account in heat transfer [ON08].

Several more simplified geometrical descriptions were already utilized in the literature to study heat transport in the eye; primarily in 2D [Sco88; NO06] and in 3D [NO07; Li+10]. In particular, the 3D model developed in [NO07] did not incorporate a detailed description of the vascular beds, although previous studies [Sco88] and our further sensitivity analysis pointed out the importance of the influence of the blood temperature on the heat distribution. The 3D geometry in [NO07] was generated by revolving a 2D model around the optical axis. In contrast, a more realistic geometry, derived from magnetic resonance imaging data, was developed in [SLG06].

The initial CAD, presented in Figure 1.3(a), is a 3D model of the human eyeball saved in the STEP format (Standard for the Exchange of Product Data), and is composed of several domains: the cornea, the vitreous body, the iris, the ligament, the lens, the sclera, the choroid, the retina, and the central retinal artery and vein.

As such, the geometry is not corresponding to the one that was described in Section 1.1. Specifically, several modifications were performed to have a geometrical model corresponding to the needs of our current model: the lamina cribrosa was not present in the initial geometry, and had to be constructed. The geometry of the lamina can be parametrized, more details are provided in the thesis of Lorenzo Sala [Sal19, Ch. III.1]. As the construction of the lamina created a new domain, the surrounding tissues (retina, sclera, optic nerve) had to be modified to take into account the new structure.

Moreover, we notice in Figure 1.3(a) the presence of the hyaloid canal. Since it does not seem to significantly impact the thermal properties of the vitreous body (see Chapter 2), we do not include it here. To create a volume corresponding to the full domain containing the vitreous humor, a tedious work consisting in manually selecting the various faces forming the boundary

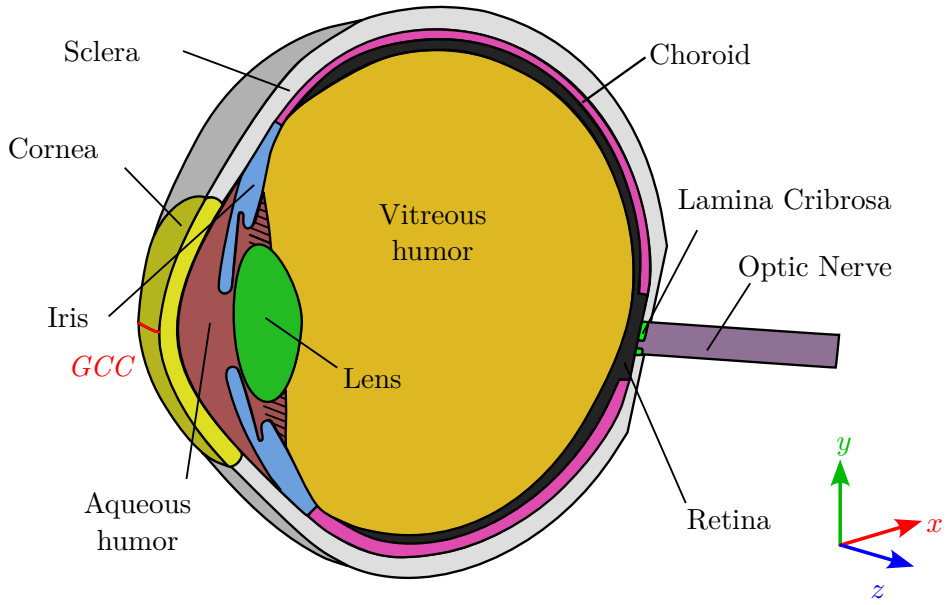


Figure 1.4: Vertical cut of the geometrical model of the human eye, with the different regions of the eye.

of the vitreous humor was performed. Finally, the posterior and anterior chambers were not included in the initial geometry. If we generated the mesh in this state, the mesh would contain a huge hole at this specific location. In the same spirit as for the vitreous humor, we manually selected the 56 faces forming the boundary of the anterior and posterior chambers among the face of the lens, the aqueous humor, the iris, the choroid, and the retina (373 faces in total !). After all these modifications, we get a new CAD of the eyeball that is presented in Figure 1.3(b).

We present in Section 7.1.1 some details on the implementation and the SALOME script. The dataset containing the CAD and the whole script has been made available in open access on GitHub [Cha+24].

Figure 1.4 shows a cut-away view along a vertical plane of the reconstructed eye anatomy. The geometry is now devoid of holes and includes all the parts of the eye that are relevant for the studies that will be performed in the sequel. To set up the geometry for meshing purposes, we need to define the interfaces between the different tissues. More details on the script are provided in Section 7.1.1.

To reduce computational effort, we further simplified the geometric description by removing the structures corresponding to the suspensory ligaments of the lens (hatched in Figure 1.4). The remaining region was initially included in the PC and considered to be filled with AH. However, on closer inspection, this anatomical description would have to be modified, as it appears that this zone is more likely to be included in the part corresponding to the vitreous humor.

We now provide some measures of the generated geometry: *(i)* The eyeball has a depth of 26 mm and a diameter of 25 mm, while the nominal value ranges from 22.0 to 24.8 mm [Wik24]. *(ii)* The cornea is 990 µm thick. According to the literature [Wei24], it ranges from 500 to 600 µm at the center and increases towards the periphery. *(iii)* The iris is 393 µm thick, and the choroid is 498 µm thick, that lies in the range given in literature, respectively around 420 µm [You+22] and 363 µm.

In this thesis, we focus on the corneal surface, where non-invasive temperature measurements can be performed [PW05; EYB89]. The geometrical center of the cornea (GCC), defined as an imaginary line horizontally bisecting the cornea, serves as the reference location for pointwise temperature evaluations, as shown in red in Figure 1.4. Outputs of interest include temperature

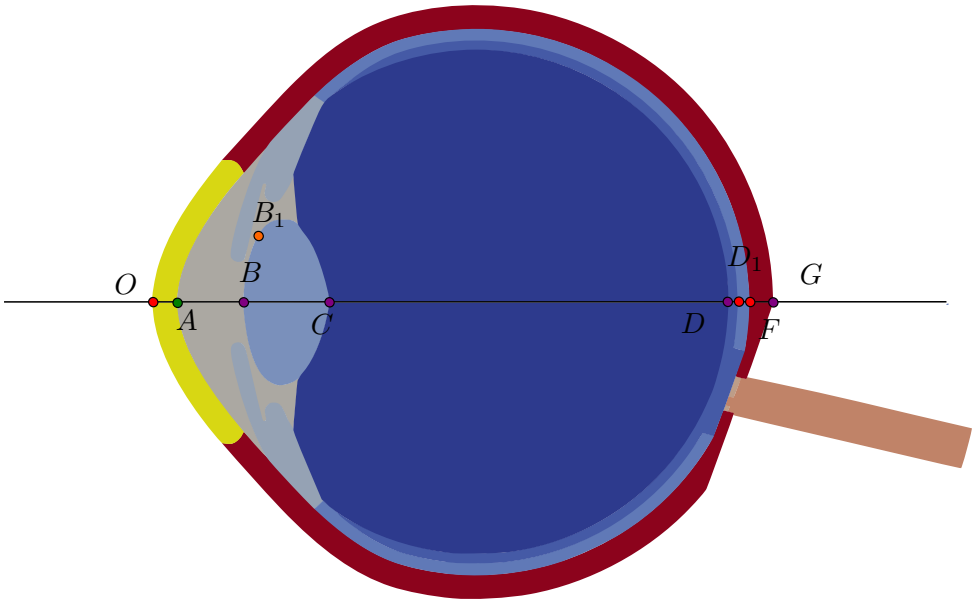


Figure 1.5: Featured geometrical locations for the output of interest (pointwise temperature).

values at specific anatomical interfaces and the mean temperature over domains such as the corneal surface. These locations are visually depicted in Figure 1.5.

1.4 Three-dimensional biophysical model for heat transfer

In this section, we introduce a biophysical approach to model the heat transfer in the human eye, which is crucial for understanding temperature regulation and its effects on ocular health.

1.4.1 Biomechanical modeling: non-linear continuous model and its linearization

Based on Section 1.3, the geometry of the eye can be written as a disjoint union of ten distinct regions: $\Omega = \bigsqcup_{i=1}^{10} \Omega_i$, where each subdomain Ω_i corresponds to one of the following regions: cornea, vitreous humor, aqueous humor, retina, iris, choroid, lens, sclera, lamina cribrosa, and optic nerve.

We focus on heat transfer in this domain, for an eyeball unexposed to external thermal sources. Hence, the temperature distribution will be constant in time [Sco88]. Following [Sco88; NO06] the steady-state condition of the heat transfer in the human eye can be described by the following system

$$\nabla \cdot (k_i \nabla T) = 0 \qquad \text{in } \Omega = \bigsqcup_{i=1}^{10} \Omega_i, \tag{1.1a}$$

where:

- i is the volume index (cornea, vitreous humor...),
- T [K] is the temperature in the domain Ω_i,
- k_i [$\mathrm{W\,m^{-1}\,K^{-1}}$] is the thermal conductivity of the tissue defined in Ω_i. We set the global thermal conductivity k [$\mathrm{W\,m^{-1}\,K^{-1}}$] as a discontinuous piece-wise constant function: $k = k_i$ on Ω_i.

Following [Can+02; NO06], we neglect the metabolic heat generation in the eye due to blood perfusion as there is no literature data available on this topic. However, such a term could be added in further analysis to enhance the model's accuracy.

The boundary $\partial\Omega$ is decomposed as: $\partial\Omega = \Gamma_{\text{amb}} \cup \Gamma_{\text{body}}$ (see Figure 1.6), where Γ_{amb} corresponds to the boundary region exposed to the ambient environment and Γ_{body} the boundary of the internal domain. Denote by $\vec{n}$ the outward normal vector to the domain Ω. The following boundary conditions are adopted:

- To model the exchange between the eye and the ambient air, and incorporate radiative heat transfer we impose the following non-linear Neumann condition:

$$-k\frac{\partial T}{\partial \vec{n}} = \underbrace{h_{\text{amb}}(T - T_{\text{amb}})}_{(i)} + \underbrace{\sigma\varepsilon(T^4 - T_{\text{amb}}^4)}_{(ii)} + \underbrace{E}_{(iii)} \quad \text{on } \Gamma_{\text{amb}}. \tag{1.1b}$$

  Three terms are present in this condition to describe different heat loss mechanisms occurring on the cornea:

  (i) The first term in the equation represents the convective heat transfer between the surface of the eye and the surrounding air. The parameter h_{amb} [$\text{W m}^{-2}\,\text{K}^{-1}$] is the air convective coefficient, and T_{amb} [K] is the ambient temperature;

  (ii) the second term represents the radiative heat transfer between the surface of the eye and the surrounding environment, where the parameter σ is the Stefan-Boltzmann constant ($\sigma = 5.67 \times 10^{-8}\,\text{W m}^{-2}\,\text{K}^{-1}$), and ε [–] is the emissivity of the surface;

  (iii) the third term represents the heat loss due to tear evaporation. Tear evaporation is a natural process that occurs at the surface of the eye, where tears constantly evaporate into the surrounding air. This process causes a cooling effect on the surface of the eye, which can be significant in dry environments or cases of reduced tear production. The parameter E [W m^{-2}] depends on the environmental conditions and the tear film characteristics.

- To model the thermal exchanges between the eye and the body, we impose:

$$-k\frac{\partial T}{\partial \vec{n}} = h_{\text{bl}}(T - T_{\text{bl}}) \quad \text{on } \Gamma_{\text{body}}, \tag{1.1c}$$

  where the parameter h_{bl} [$\text{W m}^{-2}\,\text{K}^{-1}$] is the blood convection coefficient and T_{bl} [K] is the blood temperature.

Finally, to ensure a continuous transfer of heat flux without any temperature jump, we impose the following conditions at the interface between the two adjacent regions Ω_i and Ω_j:

$$\begin{cases} T_i &= T_j \\ k_i(\nabla T_i \cdot \vec{n}_i) &= -k_j(\nabla T_j \cdot \vec{n}_j) \end{cases} \text{on } \partial\Omega_i \cap \partial\Omega_j, \tag{1.1d}$$

where $\vec{n}_i$ (resp. $\vec{n}_j$) denotes the outward normal vector to the domain Ω_i (resp. Ω_j).

System (1.1a) - (1.1d) defines a non-linear problem, denoted $\mathcal{E}_{\text{NL}}$ in the sequel.

Remark 1.4.1. Note that the condition (1.1b) modeling radiative transfer is non-linear, because of the term in T^4, which requires a more complex treatment, both from the mathematical standpoint, for the reduced basis method; and from the numerical standpoint, due to extra computational cost. As an alternative, a linearization of the condition (1.1b) was proposed in [Sco88]:

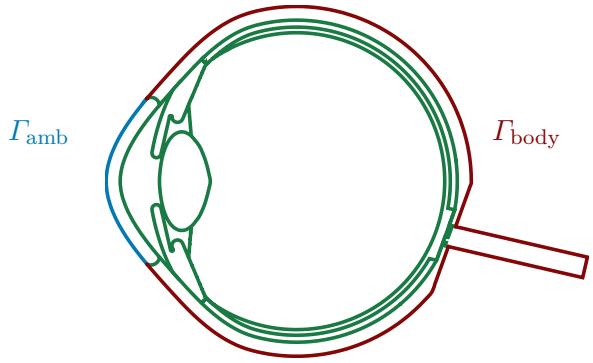


Figure 1.6: Description of the physical boundaries and interfaces of the domain Ω.

$$\sigma\varepsilon(T^4 - T_{\text{amb}}^4) = (T - T_{\text{amb}}) \underbrace{\sigma\varepsilon(T^2 + T_{\text{amb}}^2)(T + T_{\text{amb}})}_{=:h_{\text{r}}}, \tag{1.2}$$

which leads to a linear Robin condition. The value h_{r} stands for the *radiation heat transfer coefficient* and is approximately equal to $6\,\text{W}\,\text{m}^{-2}\,\text{K}^{-1}$ [Sco88], corresponds to a range of tissue temperatures T_{bl} between 30 and 45°C, and the usual ambient room temperature.

Condition (1.1b) can hence be rewritten as:

$$-k\frac{\partial T_i}{\partial \vec{n}} = h_{\text{amb}}(T - T_{\text{amb}}) + h_{\text{r}}(T - T_{\text{amb}}) + E \qquad \text{on } \Gamma_{\text{amb}}. \tag{1.3}$$

The linearized model described by Equations (1.1a)-(1.3)-(1.1c)-(1.1d) is further denoted $\mathcal{E}_{\text{L}}$.

1.4.2 Key parameters for the biophysical model

In the model presented in the previous section, many parameters are involved, but not all of them are directly measurable. Moreover, inherent uncertainties due to noise and individual variability must be taken into account in the modeling process. We therefore fixed in a first stage a set of baseline values, corresponding to the nominal values for the human body, according to the literature [Sco88; NO06] (see Table 1.1). In a second step, we split the total set of parameters into two subsets: a first part kept fixed to baseline values, and a second part that varies in a certain range (see Table 1.1). The aim is to perform a refined sensitivity analysis, that encompasses previously published studies [Sco88; NO06; NO07], and extends the analysis to a larger parameter space.

Specifically, we set the varying *parameter space* $D^\mu \subset \mathbb{R}^6$ as the Cartesian product of the intervals defined in the last column of Table 1.1. For the purpose of the sensitivity analysis, an element $\mu = \{T_{\text{amb}}, T_{\text{bl}}, h_{\text{amb}}, h_{\text{bl}}, E, k_{\text{lens}}\} \in D^\mu$ is called a *parameter*, and we denote $\bar{\mu}$ the baseline parameter, extracted from the corresponding column in Table 1.1. The dependence of the model concerning the parameter μ is emphasized by the notation $\mathcal{E}_{\text{L}}(\mu)$ and $\mathcal{E}_{\text{NL}}(\mu)$.

1.5 Mathematical framework for heat transfer modeling

This section outlines the mathematical framework, including the continuous model and its variational formulation derivation. The discrete formulation is presented in Section 2.2.

We compute the variational formulation of the linearized model $\mathcal{E}_{\text{L}}(\mu)$ described in Section 1.4.

Let $v \in H^1(\Omega)$ be an arbitrary function. As the union $\Omega = \bigsqcup_i \Omega_i$ is disjoint, we have:

Symbol	Name	Dimension	Baseline value	Range
T_{amb}	Ambient temperature	[K]	298	[283.15, 303.15]
T_{bl}	Blood temperature	[K]	310	[308.3, 312]
h_{amb}	Ambient air convection coefficient	$[\text{W m}^{-2}\text{ K}^{-1}]$	10[a]	[8, 100]
h_{bl}	Blood convection coefficient	$[\text{W m}^{-2}\text{ K}^{-1}]$	65[b]	[50, 110]
h_{r}	Radiation heat transfer coefficient	$[\text{W m}^{-2}\text{ K}^{-1}]$	6[c]	–
E	Evaporation rate	$[\text{W m}^{-2}]$	40[c]	[20, 320]
k_{lens}	Lens conductivity	$[\text{W m}^{-1}\text{ K}^{-1}]$	0.4[b]	[0.21, 0.544]
k_{cornea}	Cornea conductivity	$[\text{W m}^{-1}\text{ K}^{-1}]$	0.58[d]	–
$k_{\text{sclera}} = k_{\text{iris}} = k_{\text{lamina}} = k_{\text{opticNerve}}$	Eye envelope components conductivity	$[\text{W m}^{-1}\text{ K}^{-1}]$	1.0042[e]	–
$k_{\text{aqueousHumor}}$	Aqueous humor conductivity	$[\text{W m}^{-1}\text{ K}^{-1}]$	0.28[d]	–
$k_{\text{vitreousHumor}}$	Vitreous humor conductivity	$[\text{W m}^{-1}\text{ K}^{-1}]$	0.603[c]	–
$k_{\text{choroid}} = k_{\text{retina}}$	Vascular beds conductivity	$[\text{W m}^{-1}\text{ K}^{-1}]$	0.52[f]	–
ε	Emissivity of the cornea	[–]	0.975[a]	–

[a] [Map68] [b] [J J82] [c] [Sco88] [d] [Eme+75] [e] [NO07] [f] [ITI24]

Table 1.1: Parameters involved in the model, baseline values and ranges used in the sensitivity analysis.

$$\int_{\Omega} -\nabla \cdot (k\nabla T)v \,\mathrm{d}\vec{x} = \sum_i \int_{\Omega_i} -\nabla \cdot (k_i \nabla T)v \,\mathrm{d}\vec{x}. \tag{1.4}$$

Recall from Section 1.4 that $k = k_i$ over Ω_i. Hence, using Green's theorem:

$$\sum_i \int_{\Omega_i} -\nabla \cdot (k_i\nabla T)v \,\mathrm{d}\vec{x} = 0 \Leftrightarrow \sum_i \int_{\Omega_i} k_i \nabla T \cdot \nabla v \,\mathrm{d}\vec{x} - \int_{\partial\Omega_i} k_i \frac{\partial T}{\partial \vec{n}_i} v \,\mathrm{d}\sigma = 0, \tag{1.5a}$$

with boundary and interface conditions Equations (1.1c), (1.1d) and (1.3), we obtain

$$\begin{aligned} \sum_i k_i \int_{\Omega_i} \nabla T \cdot \nabla v \,\mathrm{d}\vec{x} + \int_{\Gamma_{\text{amb}}} [h_{\text{amb}} T + h_{\text{r}} T]\, v \,\mathrm{d}\sigma + \int_{\Gamma_{\text{body}}} h_{\text{bl}} T v \,\mathrm{d}\sigma = \\ \int_{\Gamma_{\text{amb}}} [h_{\text{amb}} T_{\text{amb}} + h_{\text{r}} T_{\text{amb}} - E]\, v \,\mathrm{d}\sigma + \int_{\Gamma_{\text{body}}} h_{\text{bl}} T_{\text{bl}} v \,\mathrm{d}\sigma, \end{aligned} \tag{1.5b}$$

where $\mathrm{d}\sigma$ denotes the surface measure on the boundary $\partial\Omega = \Gamma_{\text{body}} \sqcup \Gamma_{\text{amb}}$.

The previous equation is equivalent to:

$$a_L(T, v; \mu) = f_L(v; \mu), \tag{1.6a}$$

with:

$$\begin{aligned} a_L(T, v; \mu) := k_{\text{lens}} \int_{\Omega_{\text{lens}}} \nabla T \cdot \nabla v \,\mathrm{d}\vec{x} + \sum_{i \neq \text{lens}} k_i \int_{\Omega_i} \nabla T \cdot \nabla v \,\mathrm{d}\vec{x} + \\ \int_{\Gamma_{\text{amb}}} [h_{\text{amb}} T + h_{\text{r}} T]\, v \,\mathrm{d}\sigma + \int_{\Gamma_{\text{body}}} h_{\text{bl}} T v \,\mathrm{d}\sigma, \end{aligned} \tag{1.6b}$$

$$f_L(v; \mu) := \int_{\Gamma_{\text{amb}}} [h_{\text{amb}} T_{\text{amb}} + h_{\text{r}} T_{\text{amb}} - E]\, v \,\mathrm{d}\sigma + \int_{\Gamma_{\text{body}}} h_{\text{bl}} T_{\text{bl}} v \,\mathrm{d}\sigma. \tag{1.6c}$$

The problem statement is therefore: for $\mu \in D^\mu$ given, find the output of interest $s(\mu) \in \mathbb{R}$ given by

$$s(\mu) = \ell(T(\mu)), \tag{1.7a}$$

where $T(\mu) \in H^1(\Omega)$ is solution to problem

$$a_L(T(\mu), v; \mu) = f_L(v; \mu) \quad \forall v \in H^1(\Omega). \tag{1.7b}$$

Definition 1.5.1 (Trial and test functions)**.** In Equation (1.7b), T is called the *trial function* as it is the unknown to be determined, and v is the *test function*, which is a function used to test the solution.

The functional ℓ returns the desired output of interest, which can be the mean temperature in a selected region, *e.g.* $\ell(T(\mu)) = \frac{1}{|\Omega_{\text{cornea}}|} \int_{\Omega_{\text{cornea}}} T(\mu)\, \mathrm{d}\vec{x}$, or the temperature at a fixed point, *e.g.* $\ell(T(\mu)) = \langle \delta_O, T(\mu) \rangle$, where δ_O denotes the Dirac delta function centered at point O, and $\langle \cdot, \cdot \rangle$ represents the duality product between a functional and a function, namely $\langle \delta_O, T(\mu) \rangle = T(\mu)(O)$.

Remark 1.5.2. Note that when the output of interest is $s(\mu) = \langle \delta_O, T(\mu) \rangle$, a Dirac delta function appears in the equations. In this case, the Dirac delta only affects the output of interest, and since the temperature is assumed to be continuous, the quantity $\langle \delta_O, T(\mu) \rangle$ is well-defined. In Chapter 3, we will introduce the dual problem, where the Dirac delta appears on the right-hand side of the variational formulation, adding complexity to the problem's analysis. This observation will lead us to explore certain aspects of this issue in Chapter 4.

Theorem 1.5.3. *Let $\mu \in D^\mu$ fixed. The problem (1.6) is well-posed for $v \in H^1(\Omega)$: there exists a unique $T \in H^1(\Omega)$ such that $a_L(T, v; \mu) = f_L(v; \mu)$ for all $v \in H^1(\Omega)$.*

To prove this theorem we will need the following lemma.

Lemma 1.5.4. *Let $\Gamma \subset \partial\Omega$, and let $v \in H^1(\Omega)$. Then there exists a constant C_Γ such that:*

$$\|v\|_{L^2(\Omega)} \leqslant C_\Gamma \left(\frac{1}{2} \|\nabla v\|_{L^2(\Omega)} + \|v\|_{L^2(\Gamma)} \right). \tag{1.8}$$

Proof. By absurd, we suppose that for every $n \in \mathbb{N} \setminus \{0\}$, there exists $v_n \in H^1(\Omega)$ such that

$$\|v_n\|_{L^2(\Omega)} > n \left(\frac{1}{2} \|\nabla v\|_{L^2(\Omega)} + \|v\|_{L^2(\Gamma)} \right). \tag{1.9}$$

If we normalize, we can assume that $\forall n \in \mathbb{N} \setminus \{0\}\ \|v_n\|_{L^2(\Omega)} = 1$. The sequence $(v_n)_n$ is bounded in $H^1(\Omega)$. As the injection of $H^1(\Omega)$ into $L^2(\Omega)$ is compact, we can find a subsequence $(v_{n_k})_n$ converging to v in $L^2(\Omega)$. We still denote by $(v_n)_n$ the subsequence $(v_{n_k})_n$.

From Equation (1.9), we deduce that $\|\nabla v_n\|_{L^2(\Omega)} \to 0$, so $\nabla v_n \to 0$ in $L^2(\Omega)$. So $(v_n)_n$ is a Cauchy sequence of $H^1(\Omega)$, hence we have $v \in H^1(\Omega)$ and $\nabla v = 0$. So u is constant on Ω.

By continuity of the trace $u|_\Gamma = \lim u_n|_\Gamma = 0$ because $\|u_n\|_{L^2(\Gamma)} \to 0$ from (1.9). So $v = 0$ on Γ. As v is constant over Ω, we have $v = 0$ everywhere, which is a contradiction with $\|v\|_{L^2(\Omega)} = 1$. ∎

We can now prove the Theorem 1.5.3.

Proof. We show that the weak formulation (1.6) admits a unique solution $T \in H^1(\Omega)$. The form a_L is bilinear and for any $u, v \in H^1(\Omega)$ and $\mu \in D^\mu$, we have:

$$\begin{aligned} |a_L(u, v; \mu)| &\leqslant \sum_i k_i \left| \int_{\Omega_i} \nabla T_i \cdot \nabla v \, \mathrm{d}\vec{x} \right| + \left| \int_{\Gamma_{\text{amb}}} (h_{\text{amb}} T + h_{\text{r}} T)\, v \, \mathrm{d}\sigma \right| + \left| \int_{\Gamma_{\text{body}}} h_{\text{bl}} T v \, \mathrm{d}\sigma \right| \\ &\leqslant \sum_i k_i \|\nabla T_i\|_{L^2(\Omega_i)} \|\nabla v\|_{L^2(\Omega_i)} + (h_{\text{amb}} + h_{\text{r}}) \|T\|_{L^2(\Gamma_{\text{amb}})} \|v\|_{L^2(\Gamma_{\text{amb}})} \\ &\qquad + h_{\text{bl}} \|T\|_{L^2(\Gamma_{\text{body}})} \|v\|_{L^2(\Gamma_{\text{body}})}, \end{aligned}$$

by the Cauchy-Schwarz inequality. As the injections of $H^1(\Omega)$ into $L^2(\Gamma_N)$ and $L^2(\Gamma_R)$ are continuous, we have two constants $C_R, C_N > 0$ such that $\|u\|_{L^2(\Gamma_{\text{amb}})} \leqslant C_N \|u\|_{H^1(\Omega)}$ (idem on Γ_{body}), hence setting $k_{\max} = \max_i\{k_i\}$

$$\begin{aligned}|a_L(u,v;\mu)| &\leqslant k_{\max}\|\nabla T\|_{L^2(\Omega)}\|\nabla v\|_{L^2(\Omega)} + C_N(h_{\text{amb}}+h_{\text{r}})\|T\|_{L^2(\Gamma_{\text{amb}})}\|v\|_{L^2(\Omega)} \\ &\quad + C_R h_{\text{bl}}\|T\|_{L^2(\Gamma_{\text{body}})}\|v\|_{L^2(\Omega)} \\ &\leqslant (k_{\max} + C_N(h_{\text{amb}}+h_{\text{r}}) + C_R h_{\text{bl}})\|\nabla T\|_{H^1(\Omega)}\|\nabla v\|_{H^1(\Omega)},\end{aligned}$$

so the bilinear form a is continuous.

Now, for any $v \in H^1(\Omega)$ and $\mu \in D^\mu$, we have:

$$\begin{aligned}a_L(v,v;\mu) &= \sum_i k_i \int_{\Omega_i}\|\nabla v\|^2_{L^2(\Omega_i)}\,\mathrm{d}\vec{x} + \int_{\Gamma_{\text{amb}}}(h_{\text{amb}}+h_{\text{r}})\|v\|^2_{L^2(\Gamma_{\text{amb}})}\,\mathrm{d}\sigma + \int_{\Gamma_{\text{body}}} h_{\text{bl}}\|v\|^2_{L^2(\Gamma_{\text{body}})}\,\mathrm{d}\sigma \\ &\geqslant m\left[\|\nabla v\|^2_{L^2(\Omega)} + \|v\|^2_{L^2(\Gamma_{\text{amb}})} + \|v\|^2_{L^2(\Gamma_{\text{body}})}\right],\end{aligned}$$

with $m = \min\{k_i, h_{\text{amb}}+h_{\text{r}}, h_{\text{bl}}\}$. By applying the Cauchy-Schwarz inequality to the vectors $\left[\sqrt{\frac{m}{3}}^2, \sqrt{\frac{m}{3}}^2, \sqrt{\frac{m}{3}}^2\right]^T$ and $\left[\|\nabla v\|^2_{L^2(\Omega)}, \|v\|^2_{L^2(\Gamma_{\text{amb}})}, \|v\|^2_{L^2(\Gamma_{\text{body}})}\right]^T$, we get

$$\begin{aligned}a_L(v,v;\mu) &\geqslant \left(\sqrt{\frac{m}{2}}\left[\|\nabla v\|^2_{L^2(\Omega)} + \|v\|^2_{L^2(\Gamma_{\text{amb}})} + \|v\|^2_{L^2(\Gamma_{\text{body}})}\right]\right)^2 \\ &\geqslant \frac{m}{4}\left[\left(\frac{1}{2}\|\nabla v\|^2_{L^2(\Omega)} + \|v\|^2_{L^2(\Gamma_{\text{amb}})}\right) + \left(\frac{1}{2}\|\nabla v\|^2_{L^2(\Omega)} + \|v\|^2_{L^2(\Gamma_{\text{body}})}\right)\right] + \frac{m}{4}\|\nabla v\|^2_{L^2(\Omega)} \\ &\geqslant \frac{m}{4}\left[C_{\Gamma_{\text{amb}}}\|v\|_{L^2(\Omega)} + C_{\Gamma_{\text{body}}}\|v\|_{L^2(\Omega)}\right] + \frac{m}{4}\|\nabla v\|^2_{L^2(\Omega)},\end{aligned}$$

from Lemma 1.5.4. Setting $\bar{m} = \min\{C_N\frac{m}{4}, C_R\frac{m}{4}, \frac{m}{4}\}$, we get

$$a_L(v,v;\mu) \geqslant \bar{m}\|v\|^2_{H^1(\Omega)}$$

So the bilinear form a_L is coercive. Moreover, the application f_L is linear and continuous. So according to the Lax-Milgram theorem [LM54; EG21], there exists a unique solution $T \in H^1(\Omega)$ such that $a_L(T,v;\mu) = f_L(v;\mu)$ for all $v \in H^1(\Omega)$ and $\mu \in D^\mu$. ■

From the previous theorem, have the following properties on the bilinear form a_L:

Corollary 1.5.5. *For all $\mu \in D^\mu$, the bilinear form $a_L(\cdot,\cdot,\mu)$ is continuous and coercive. There exists constants $\alpha(\mu)$ and $\gamma(\mu)$ such that*

$$0 < \alpha(\mu) := \inf_{T\in H^1(\Omega)} \frac{a_L(T,T;\mu)}{\|T\|^2_{H^1(\Omega)}}, \tag{1.10a}$$

$$\gamma(\mu) := \sup_{T\in H^1(\Omega)}\sup_{v\in H^1(\Omega)} \frac{a_L(T,v;\mu)}{\|T\|_{H^1(\Omega)}\|v\|_{H^1(\Omega)}} < \infty. \tag{1.10b}$$

Theorem 1.5.6. *Let $T(\mu)$ be the unique solution to Problem (1.6). Assume that $T(\mu) \in H^1(\Omega)\cap H^2(\Omega)$. Then $T(\mu)$ is solution to problem $\mathcal{E}_L(\mu)$.*

Proof. We have for any $v \in H^1(\Omega)$, $a_L(T,v;\mu) = f_L(v;\mu)$, thus:

$$\int_\Omega\left[\sum_i k_i\nabla T_i\right]\cdot\nabla v\,\mathrm{d}\vec{x} + \int_{\Gamma_{\text{amb}}}\left[h_{\text{amb}}(T-T_{\text{amb}}) + h_{\text{r}}(T-T_{\text{amb}}) + E\right]v\,\mathrm{d}\sigma + \int_{\Gamma_{\text{body}}} h_{\text{bl}}(T-T_{\text{amb}})v\,\mathrm{d}\sigma = 0.$$

Applying Green's theorem:

$$-\int_{\Omega}\left[\sum_i \nabla\cdot(k_i\nabla T)\right]\cdot v\,\mathrm{d}\vec{x} = \int_{\Gamma_{\mathrm{amb}}}\left[h_{\mathrm{amb}}(T-T_{\mathrm{amb}})+h_{\mathrm{r}}(T-T_{\mathrm{amb}})+E-k\frac{\partial T}{\partial\vec{n}}\right]v\,\mathrm{d}\sigma + \int_{\Gamma_{\mathrm{body}}}\left[h_{\mathrm{bl}}(T-T_{\mathrm{amb}})-k\frac{\partial T}{\partial\vec{n}}\right]v\,\mathrm{d}\sigma.$$

By density of $C_c^\infty(\Omega)$ in $H^1(\Omega)$, we deduce that

$$\begin{cases}\sum_i \nabla\cdot(k_i\nabla T)=0 & \text{in } \Omega,\\ -k\frac{\partial T_i}{\partial\vec{n}} = h_{\mathrm{amb}}(T-T_{\mathrm{amb}})+h_{\mathrm{r}}(T-T_{\mathrm{amb}})+E & \text{on } \Gamma_{\mathrm{amb}},\\ -k\frac{\partial T}{\partial\vec{n}} = h_{\mathrm{bl}}(T-T_{\mathrm{bl}}) & \text{on } \Gamma_{\mathrm{body}}.\end{cases}$$

■

Remark 1.5.7. The well-posedness of the fully non-linear problem $\mathcal{E}_{\mathrm{NL}}(\mu)$ can also be obtained by the mean of a variational approach, in the spirit of [Mil93].

1.6 Conclusion

In this chapter, we presented the anatomy of the human eyeball, and the medical challenges that are associated with the heat transfer in the eye. We presented the geometrical and physical models that were considered in the present work, highlighting some parameters involved, as well as output of interest whose study will be performed in the sequel. As pointed out in the context and in literature [SLG06; Bou11; BBS20], the geometry of the eye is also influenced by the age, the ethnicity of the subject, *etc.* At this point these geometrical parameters are not taken into account in the model, a perspective of the work is to include them. Two approaches can be considered: *(i)* change these parameters in the CAD and generate new meshes, or *(ii)* use mesh adaptation post-process to adapt some regions of the mesh to take into account the variations of the geometrical parameters. Finally, we introduced the mathematical model, showing the continuous model and its variational formulation.

The realistic geometrical model of the human eye takes into account parts of the eye that are relevant for the heat transfer studies that will be performed in the sequel, and that were not necessarily considered in previous works, such as the vascular beds in the choroid and the retina.

Thanks to these models, we are now able to develop a discretization of the problem, and to perform numerical simulations, presented in Chapter 2.

Chapter 2

Full order computational framework: methods, verification and validation

The previous chapter introduced the geometrical and biophysical models of the eye. In this chapter, we focus on the discretization of these models, with a particular emphasis on mesh generation and the Finite Element Method (FEM), which are essential tools for transforming continuous models into discrete, computationally tractable forms.

Mesh generation involves constructing and refining grids that accurately represent the eye's complex geometrical structure. This section covers the generation of initial meshes, techniques for adapting them to preserve crucial geometric properties, and methods for verifying their accuracy. Several methods have been proposed to develop meshes from geometrical data [FGF08], and many library tools are available to generate and refine them, either in open-source software like GMSH [GR09], NETGEN [Sch97], or commercial software like MeshGems. We also evaluate the performance of the mesh generation and refinement processes. Two-dimensional meshes stemming from realistic geometries of the human eye have been employed in [Sco88; NO06], and three-dimensional meshes have been used in [NO07] by revolving the 2D mesh around the optical axis.

Subsequently, we delve into the Finite Element Method [ESW14], which leverages these meshes and a mathematical formulation of the problem to approximate the eye's biophysical behavior. Key topics include selecting appropriate finite element spaces, assessing their performance and scalability, and discussing verification and validation methods to ensure the reliability of FEM-based simulations. The FEM has successfully been applied to model heat transfer in the eye by several authors [Sco88; NO06; NO07], both in two and three dimensions. An extension of the FEM, the *alpha finite element method*, has also been used to model heat transfer in the eye [Li+10]. In [NOR08], the authors compared theirs results between a 2D and 3D model, to assess the impact of the dimensionality of the model on the results. Their conclusion is that a 3D model is indeed more precise, even though the 2D model in some case can provide satisfactory results.

In this chapter, we introduce the discretization of the geometrical model in Section 2.1, then we present the Finite Element Method in Section 2.2 used to discretize the biophysical model of the eye.

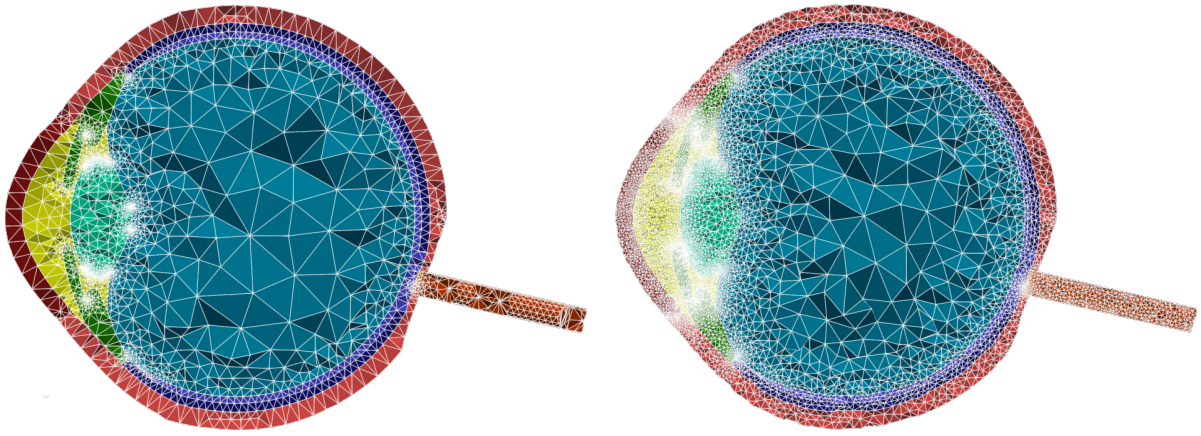
2.1 Discrete geometrical representation

2.1.1 Mesh construction

In Section 1.3, we presented the geometrical model of the eye and how to build it from the CAD data. In this section, we focus on the meshing of this model. This step is executed using Salome's meshing library [RBG17], which employs the NETGEN algorithm. NETGEN [Sch97] is an open source automatic 3D tetrahedral mesh generator that is able to create meshes of high quality.

The first step is to create the mesh object and to define the algorithm that will be used to create the mesh. Then we define the markers that will be used to define the different tissues. The code (truncated) performing these step is shown in Section 7.1.2, but is also available in full-length in the GitHub repository [Cha+24]. Finally, we compute the mesh and export it to a MED file. Note that we did not set any particular parameters for the meshing algorithm. The default parameters are used, but still we will handle the mesh quality by the use of a mesh refinement strategy, that is presented in the sequel, see Section 2.1.2.

The generated mesh is shown in Figure 2.1(a). Note that at this point, the mesh is quite coarse, except in parts where small domains are present, such as the lamina cribrosa, or when the shape of the geometry is complex, such as the iris and the lens.



(a) Original mesh generated by Salome, with $4.64 \cdot 10^5$ tetrahedrons: $h_{\min} = 5.77 \cdot 10^{-6}, h_{\max} = 5.76 \cdot 10^{-3}$.

(b) Mesh refined around the anterior and posterior chambers, with $9.4 \cdot 10^5$ elements: $h_{\min} = 5.09 \cdot 10^{-5}, h_{\max} = 3.12 \cdot 10^{-3}$.

Figure 2.1: Meshed geometry of the eye, over a vertical plane. Characteristic sizes are given in meters.

(a) Meshes `M`.

Mesh	# elements	$h_{\min}$	$h_{\max}$	h_{avg}
`M0`	$2.25 \cdot 10^5$	$8.25 \cdot 10^{-5}$	$3.64 \cdot 10^{-3}$	$8.6 \cdot 10^{-4}$
`M1`	$3.27 \cdot 10^5$	$7.28 \cdot 10^{-5}$	$3.52 \cdot 10^{-3}$	$7.39 \cdot 10^{-4}$
`M2`	$6.32 \cdot 10^5$	$7.48 \cdot 10^{-5}$	$1.65 \cdot 10^{-3}$	$5.11 \cdot 10^{-4}$
`M3`	$1.15 \cdot 10^6$	$4.84 \cdot 10^{-5}$	$1.12 \cdot 10^{-3}$	$4.69 \cdot 10^{-4}$
`M4`	$5.84 \cdot 10^6$	$5.11 \cdot 10^{-5}$	$6.01 \cdot 10^{-4}$	$2.94 \cdot 10^{-4}$
`M5`	$4.38 \cdot 10^7$	$3.03 \cdot 10^{-5}$	$3.54 \cdot 10^{-4}$	$1.5 \cdot 10^{-4}$

(b) Meshes `Mr`, remeshed around AH.

Mesh	# elements	$h_{\min}$	$h_{\max}$	h_{avg}
`Mr0`	$1.92 \cdot 10^5$	$1.25 \cdot 10^{-4}$	$4 \cdot 10^{-3}$	$9.23 \cdot 10^{-4}$
`Mr1`	$2.82 \cdot 10^5$	$1.37 \cdot 10^{-4}$	$3.63 \cdot 10^{-3}$	$7.72 \cdot 10^{-4}$
`Mr2`	$7.47 \cdot 10^5$	$6.54 \cdot 10^{-5}$	$1.6 \cdot 10^{-3}$	$4.67 \cdot 10^{-4}$
`Mr3`	$1.4 \cdot 10^6$	$3.29 \cdot 10^{-5}$	$9.59 \cdot 10^{-4}$	$4.17 \cdot 10^{-4}$
`Mr4`	$6.04 \cdot 10^6$	$2.55 \cdot 10^{-5}$	$5.29 \cdot 10^{-4}$	$2.88 \cdot 10^{-4}$
`Mr5`	$4.39 \cdot 10^7$	$3.12 \cdot 10^{-5}$	$1.5 \cdot 10^{-4}$	$2.77 \cdot 10^{-4}$

Table 2.1: Description of the families of meshes generated: number of elements and characteristic sizes, in meter.

2.1.2 Mesh refinement strategy

In Section 1.3, we detailed the geometry of the eyeball, derived from CAD data. As illustrated in Figure 2.1(a), certain regions exhibit greater complexity than others. For instance, the lamina cribrosa is notably thinner, while the iris presents a less uniform structure. Achieving an effective mesh requires a well-distributed arrangement of elements. This is attainable through the application of a specialized meshing algorithm designed to tailor the mesh according to the geometric intricacies. Utilizing the MMG library [MMG22], we have generated a family of meshes with varying levels of refinement. These meshes are used to our subsequent simulation processes for the heat transfer model, and are denoted `M0`, `M1`, `M2`, `M3`, `M4`, and `M5`. Their characteristics are detailed in Table 2.1(a).

In Chapter 6, we introduce a model where heat transfer in the eyeball is coupled to the AH flow in the anterior and posterior chambers of the eye. The geometry of the domain Ω_{AH} is quite complex, especially at the junction between the anterior and posterior chambers. As this is a very small region, we need to refine the mesh in this area, which can lead to a very large number of elements in the whole domain.

Geometrical mesh refinement can easily be implemented in the configuration files of Feel++ simulation, by *(i)* setting the metric used to refine the mesh, *(ii)* using the distance to a range of markers, and *(iii)* defining the metric as a function of this distance. The corresponding code is presented in Section 7.1.3.

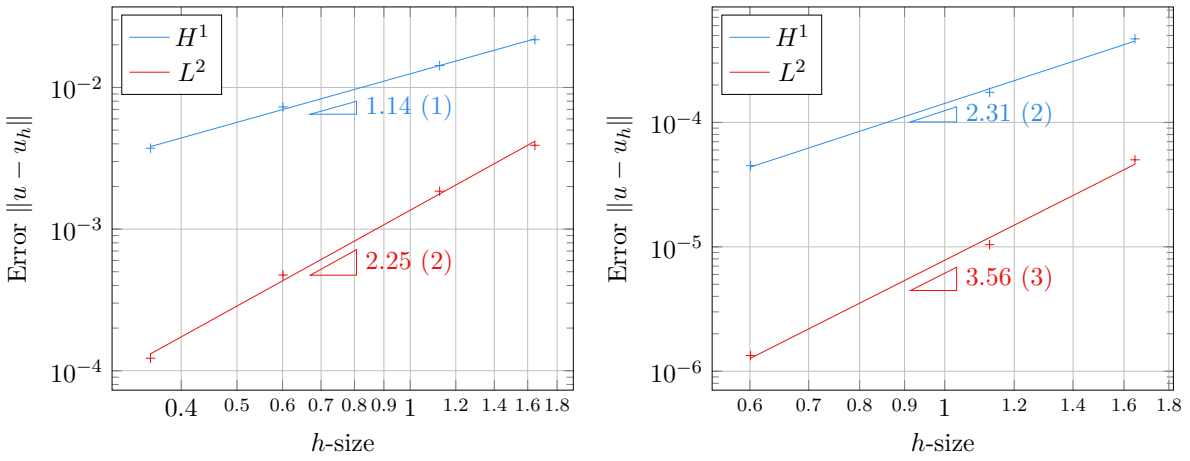
The result of this mesh refinement procedure, denoted `Mr` in the sequel, is shown in Figure 2.1(b), and is also available in [Cha+24]. We can see in the figure that the desired domains are indeed refined as expected. The mesh `M1` has 68,993 points, for an average element size of 0.74, a maximal element size of 3.52 and a minimal element size of $7.28 \cdot 10^{-2}$; while the refined mesh `Mr` has $3.36 \cdot 10^5$ points, for an average element size of $h_{\text{avg}} = 0.33$, and a maximal element size of $h_{\max} = 3.21$ and $h_{\min} = 4.03 \cdot 10^{-2}$.

In the next section, we will perform some tests on the meshes to ensure that the results are stable with respect to the mesh size. To set up these tests, we perform a mesh refinement process starting from the mesh `Mr`, and denote the resulting meshes as `Mr1`, `Mr2`, `Mr3`, `Mr4` and `Mr5`. Some characteristics of these meshes are gathered in Table 2.1(b). The number of elements is increasing with the level of refinement, as expected.

The dataset of the family `Mr` is available in open access [Sai+24c].

2.1.3 Verification of the generated meshes

Verification of generated meshes is a critical step in ensuring their accuracy and reliability for computational simulations. Two specific verification studies are conducted here: first, we solve a Laplacian toy problem. This involves solving a partial differential equation and comparing the numerical solution against an analytical one, and compare the behavior of the error against



(a) $\mathbb{P}_1$ discretization.

(b) $\mathbb{P}_2$ discretization.

Figure 2.2: Evolution of the error, according to the mesh size, for the Laplacian problem. The exact solution used in simulations is $u(x, y, z) = x^3y + z^4$. The theoretical convergence rate is indicated between parentheses.

what is expected from the theory. Second, we compute the volume of the meshed geometry to check the geometric fidelity of the mesh, by comparing the calculated volume with the known volume of the original geometry to ensure that the mesh accurately represents the physical space. These verification steps are essential to confirm that the generated meshes are suitable for further simulations and analyses.

A Poisson toy problem The first study consists of solving the following problem:

$$\begin{cases} -\Delta u = f & \text{on } \Omega, \\ u = g & \text{over } \partial\Omega, \end{cases} \tag{2.1}$$

where u is a manufactured solution. Given u, we can analytically compute the right-hand side f and the boundary conditions g, and then solve the problem numerically. This can be done with the application `feelpp_qs_laplacian_3d`, that solves Problem (2.1) using the finite element method. More details about this method are presented in Section 2.2. The application computes the norms of the error $\|u - u_h\|_{L^2(\Omega)}$ and $\|u - u_h\|_{H^1(\Omega)}$, where u_h is the numerical solution computed by the library.

The results are presented in Figure 2.2. The error is computed for the $\mathbb{P}_1$ and $\mathbb{P}_2$ discretizations, and the theoretical convergence rate is indicated on the plot. The results show that the error decreases as the mesh is refined, and the convergence rate is in agreement with the theoretical rate [EG21].

Preservation of the volume First, to ensure that the mesh refinement is well constructed, we need to verify that the geometrical properties of the domain are preserved. We focus here on the global volume of the domain, that can be easily computed using the Feel++ library. We then compare across all the meshes generated in Section 2.1.2. The results are presented in Figure 2.3. The main striking result is that when the mesh is too coarse, the volume is not accurately preserved, but for the finer mesh, it tends toward a constant value, around 8.02 mL. The geometric model overestimates this value, but still remains in an acceptable physiological range [Hey+16].

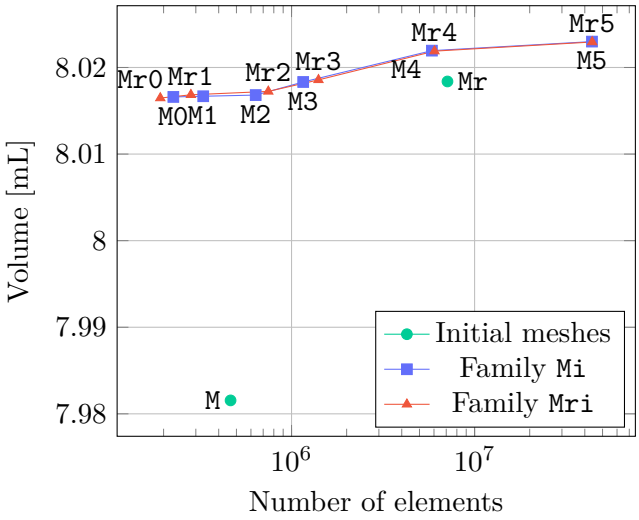


Figure 2.3: Evolution of the volume of the domain, depending on the level of refinement of the mesh.

Another check can be performed, by computing the volume of all regions of the eye described in Section 1.4, and comparing the sum to the whole volume. If a difference is observed, it can be an indication of a problem in the meshing process, and that there is a part of the geometry that is not well meshed, or not meshed at all. For the meshes that we generated, we compute a difference of the order 10^{-20}, which is negligible and can be attributed to the numerical precision of the computations.

From this study, we can conclude that the mesh is well constructed, and that it is suitable for further simulations and analyses. However, for the family of meshes `Mr`, we need to use the finer among them (`Mr4` and `Mr5`), as the volume is not preserved for the coarser ones.

2.1.4 Performance evaluation

In this section, we focus on the performance of the mesh generation and refinement processes.

The generation of the mesh, described in Section 2.1, is executed with the code shown in Listing 7.12. In the process, the time taken to compute the mesh by Salome [RBG17] is 49.01 s, while the whole pipeline going from the initial CAD to the exportation of the mesh takes 60.38 s, showing that the mesh generation is the most time-consuming part of the process.

Now, we look at the time taken to perform the mesh refinement, with the MMG library [MMG22], as described in Section 2.1.2. The results are presented in Table 2.2. The time is measured for the different mesh of the family `M`, and for different number of parallel processes: in sequential mode (`np1`), with 12 processes (`np12`), and with 24 processes (`np24`). The time to load the mesh `M` is also included in the table.

The first striking result is that when the number of parallel processes is increased, the time taken to load the mesh is also increased. This can be explained by the fact that the mesh is saved as a sequential mesh, and the application need to load the mesh and distribute it to the different processes, which takes more time. Moreover, we see that the time taken to refine the mesh is slightly identical when the number of parallel processes is increased, hence motivating the need to pre-partition the mesh for subsequent simulations. The Python script performing the mesh refinement, shown in Listing 7.14, export the refined mesh at the format `hdf5`, which is a binary format more efficient than the `msh` format used by Salome.

The step of mesh partitioning is actually quite fast: the time to read the initial `hdf5` mesh, partition it over 1, 2, 4 8, 12 and 24 parallel processes and export the results mesh goes from

	Mesh size (see Table 2.1)	Computational time [s] np1	np12	np24
Mesh loading	$4.64 \cdot 10^5$	6.14	10.94	11.28
M0	$2.25 \cdot 10^5$	113.28	129.52	125.95
M1	$3.27 \cdot 10^5$	120.22	120.16	117.06
M2	$6.32 \cdot 10^5$	135.3	134.6	132.48
M3	$1.15 \cdot 10^6$	158.05	156.84	154
M4	$5.84 \cdot 10^6$	384.33	384.12	367.24
M5	$4.38 \cdot 10^7$	2,088.45	2,061.73	2,045.92

Table 2.2: Performance evaluation of the mesh refinement process with MMG library, for various number of parallel processes, in sequential mode (`np1`), or in parallel (`np12` and `np24`).

2.79 s for the mesh `M0`, to 3.01 s for the mesh `M5`. For instance, pre-partitioned over 12 parallel processes, the mesh `M4` takes 0.13 s to be loaded, against 8.31 s for the `msh` format.

In conclusion, the performance evaluation highlights that the mesh generation process is the most time-consuming, accounting for the majority of the computational overhead. Although parallel processing speeds up mesh refinement, it introduces additional delays in mesh loading due to the need to distribute the mesh across multiple processes. The choice of file format, such as using `hdf5` over `msh`, also plays a crucial role in optimizing loading times.

2.2 Finite element method

In this section, we present the Finite Element Method (FEM) used to solve the heat transfer problem in the eye, and present results of simulation of the model, in order to verify and validate it. The FEM is particularly useful for solving problems with complex geometries, such as the one previously described, because it allows for the discretization of irregular shapes and domains into smaller, manageable elements. Additionally, FEM provides a framework for obtaining approximate solutions with controlled error bounds, enabling accurate predictions of physical phenomena while maintaining computational efficiency.

2.2.1 High fidelity FEM resolution

We briefly describe the Galerkin FEM. More details can be found in literature, such as [EG21; ESW14]. The main idea is to numerically approximate the solution $T(\mu)$ to Problem (1.7b), by taking a finite dimension subspace $V_h \subset V := H^1(\Omega)$. Such space is constructed thanks to the mesh discretization, as introduced in Section 2.1. Let $\{\varphi_j\}_{j=1}^{\mathcal{N}}$ be a basis of V_h, and we denote by $\mathcal{N}$ its dimension. The finite element approximation $T_h(\mu) \in V_h$ is then associated to a vector $\underline{\boldsymbol{T}}^{\text{fem}} \in \mathbb{R}^{\mathcal{N}}$ such that

$$T_h(\mu) = \sum_{i=1}^{\mathcal{N}} \underline{\boldsymbol{T}}_i^{\text{fem}}(\mu)\varphi_i. \tag{2.2}$$

The results of the Galerkin approximation is a finite-dimensional equivalent of the variational formulation Equation (1.7b): find $T_h \in V_h$ such that

$$a_L(T_h, v_h; \mu) = f_L(v_h; \mu) \quad \forall v_h \in V_h. \tag{2.3}$$

Consequently, as φ_j is a basis function of V_h for $1 \leqslant j \leqslant \mathcal{N}$, we have:

$$a_L\left(\sum_{i=1}^{\mathcal{N}} \underline{\boldsymbol{T}}_i^{\text{fem}} \varphi_i, \varphi_j; \mu\right) = f_L(\varphi_j; \mu),$$
$$\sum_{i=1}^{\mathcal{N}} \underline{\boldsymbol{T}}_i^{\text{fem}} a_L(\varphi_i, \varphi_j; \mu) = f_l(\varphi_j; \mu).$$

This latter system can be rewritten under the algebraic form:

$$\underline{\underline{\boldsymbol{A}}}_L(\mu)\, \underline{\boldsymbol{T}}^{\text{fem}}(\mu) = \underline{\boldsymbol{f}}_L(\mu), \tag{2.4a}$$

where

$$\underline{\underline{\boldsymbol{A}}}_L(\mu) = [a_{i,j}]_{i,j} \in \mathbb{R}^{\mathcal{N}\times\mathcal{N}} \quad a_{i,j} = a_L(\varphi_i, \varphi_j; \mu), \tag{2.4b}$$
$$\underline{\boldsymbol{f}}_L(\mu) = [f_i]_i \in \mathbb{R}^{\mathcal{N}} \quad f_i = f_L(\varphi_i; \mu). \tag{2.4c}$$

As presented in Section 1.1, we focus on various *outputs of interest*, $s_k(\mu)$, for $k \in [\![1, n_{\text{output}}]\!]$ given by the formula $s_k(\mu) = \ell_k(T(\mu); \mu)$, where ℓ is a bounded linear form and $T(\mu)$ is the solution to Problem (1.7b). The previous problem is equivalent to:

$$\underline{\underline{\boldsymbol{A}}}_L(\mu)\underline{\boldsymbol{T}}^{\text{fem}}(\mu) = \underline{\boldsymbol{f}}_L(\mu), \tag{2.5a}$$
$$s_k(\mu) = \underline{\boldsymbol{L}}_k(\mu)^T \underline{\boldsymbol{T}}^{\text{fem}}(\mu). \tag{2.5b}$$

The vector $\underline{\boldsymbol{L}}_k(\mu) \in \mathbb{R}^{\mathcal{N}}$ is defined in the same manner as $\underline{\boldsymbol{f}}_L$.

The steps run during resolution are recapitulated in Algorithm 1.

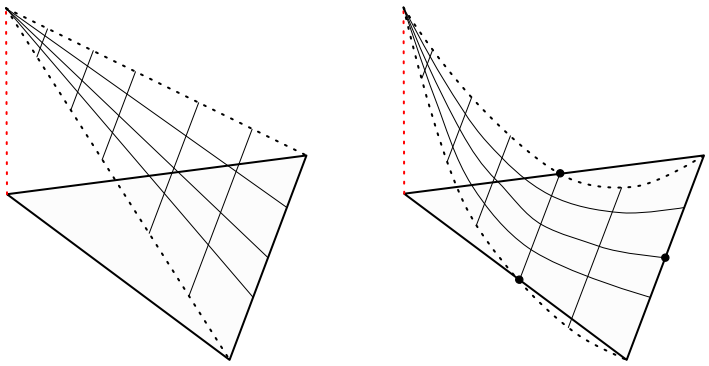
Algorithm 1: High fidelity resolution.

Input: $\mu \in D^\mu$.
Assemble $\underline{\underline{\boldsymbol{A}}}_L(\mu)$, $\underline{\boldsymbol{f}}_L(\mu)$, $\underline{\boldsymbol{L}}_k(\mu)$;
Solve $\underline{\underline{\boldsymbol{A}}}_L(\mu)\underline{\boldsymbol{T}}^{\text{fem}}(\mu) = \underline{\boldsymbol{f}}_L(\mu)$;
Compute outputs $s_k(\mu) = \underline{\boldsymbol{L}}_k(\mu)^T \underline{\boldsymbol{T}}^{\text{fem}}(\mu)$;
Output: Numerical solution $\underline{\boldsymbol{T}}^{\text{fem}}(\mu)$ and outputs $s_k(\mu)$.

Before presenting the first results of simulation, we need to discuss the choice of the basis if the finite element space V_h, with the standard Galerkin continuous method.

In the context of finite element methods, the $\mathbb{P}_1$ and $\mathbb{P}_2$ finite element spaces are commonly used [EG21]. The $\mathbb{P}_1$ finite element space, also known as the linear finite element space, consists of piece-wise linear functions. Each basis function in the $\mathbb{P}_1$ space is associated with a vertex of the mesh and is defined to be 1 at its associated vertex and 0 at all other vertices. This creates a continuous, piece-wise linear approximation over the entire mesh. An example of $\mathbb{P}_1$ basis function over one triangular element is shown in Figure 2.4(a). The $\mathbb{P}_2$ finite element space, on the other hand, consists of piece-wise quadratic functions. The basis functions in the $\mathbb{P}_2$ space are associated not only with the vertices but also with the midpoints of the edges of the mesh elements. Each basis function is defined to be 1 at its associated node (vertex or midpoint) and 0 at all other nodes. This results in a continuous, piece-wise quadratic approximation. An example is provided in Figure 2.4(b).

The choice of basis functions with support defined on a few number of elements is motivated by the desire to achieve sparsity in the matrix $\underline{\underline{\boldsymbol{A}}}$, as a sparse system is easier to invert than a dense one. This sparsity leads to reduced computational complexity and memory requirements, facilitating more efficient numerical solutions and enabling the handling of larger systems.



(a) $\mathbb{P}_1$ basis function.

(b) $\mathbb{P}_2$ basis function (taken from [EG21]).

Figure 2.4: Examples of finite element basis functions. This is presented for 2D elements, but the concept extends to 3D elements.

Resolution strategy We implement Algorithm 1 in the framework of the open-source library Feel++ [Pru+24b], and specifically the `heat` toolbox[1] where both models $\mathcal{E}_{\mathrm{NL}}(\mu)$ and $\mathcal{E}_{\mathrm{L}}(\mu)$ can be simulated. In the context of $\mathcal{E}_{\mathrm{NL}}$, non-linear iterations are required to achieve convergence by the mean of the Newton method, namely the resolution step of Algorithm 1 is performed iteratively until convergence is reached. More details about this methodology are provided in Section 6.2.1.

To numerically solve the system Equation (2.5), we use multigrid preconditioning, GAMG [Bra86; ADL00]. *GAMG* (for Geometric Algebraic Multigrid) efficiently handles large sparse matrices by recursively coarsening and solving the problem on multiple levels, significantly accelerating the convergence. The core idea behind GAMG is to generate a quick solution on a coarse grid and then interpolate it back to the finer grid, using it as an initial guess for the solution on the finer grid.

The implementation in Feel++ involves several key components that contribute to its effectiveness. These include a highly expressive embedded language, which allows for detailed and flexible problem description. Additionally, straightforward interpolation techniques are employed to handle the data between different levels of the multigrid method. Mesh adaptation techniques are used to refine or coarsen the computational grid as needed, improving the accuracy and efficiency of the solution process. Finally, parallelization capabilities are integrated into the framework, enabling the handling of large-scale problems by distributing the computational workload across multiple processors.

These features collectively enable Feel++ to manage the complexity of the models and effectively resolve the system, making it a powerful tool for solving large-scale, non-linear finite element problems.

2.2.2 Scalability

In this section, we explore the scalability properties of the developed computational framework. This involves measuring the time required to solve the model in relation to the number of MPI parallel processes utilized. The time measured pertains to the duration necessary for assembling the algebraic system and solving the problem, as per Algorithm 1. Our experiments utilized mesh `M3`, with both $\mathbb{P}_1$ and $\mathbb{P}_2$ discretizations.

[1] See documentation: https://docs.feelpp.org/toolboxes/latest/heat/toolbox.html

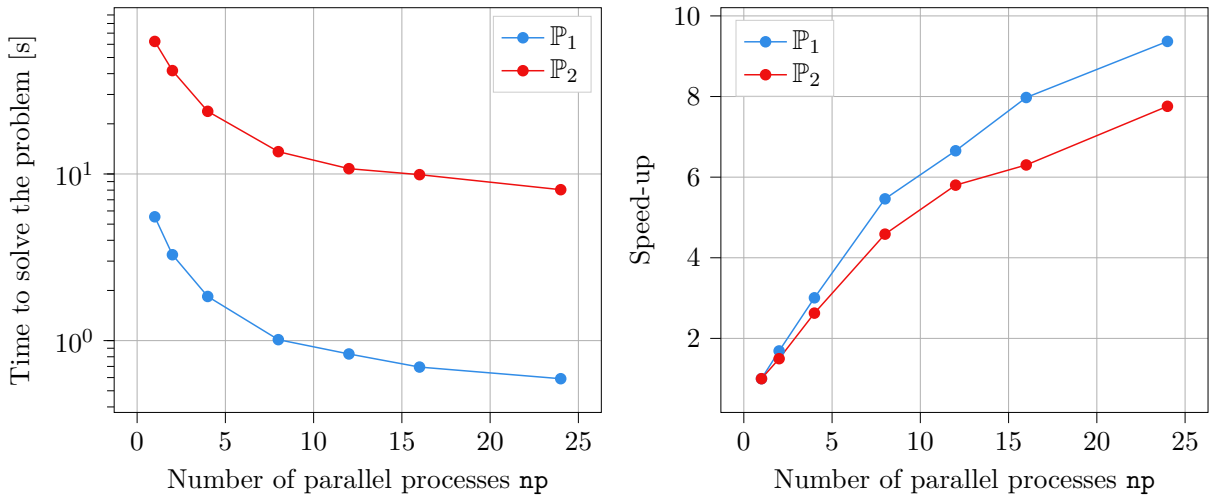


Figure 2.5: Time of execution to run $\mathcal{E}_{\mathrm{L}}^{\mathcal{N}}(\bar{\mu})$ and corresponding speed-up, for an increasing number of parallel processes. Simulations are performed on the mesh `M3`.

Definition 2.2.1. The *speed-up* is defined as the ratio of the time taken to solve the problem with a single process to the time taken with `np` processes:

$$\text{speed-up}(\texttt{np}) = \frac{\text{time}(1)}{\text{time}(\texttt{np})}. \tag{2.6}$$

The results, presented in Figure 2.5, demonstrate satisfactory scalability: the execution time decreases as the number of parallel processes increases. However, we observed that beyond 12 processes, the reduction in execution time becomes less significant. Consequently, for optimal efficiency, we have selected 12 processes for our subsequent analyses. This study sets the stage for a subsequent comparison with a reduced-order model, which employs a reduced basis with reliable, certified output bounds derived from the high-fidelity solutions, see Chapter 3. This comparison aims to highlight that, while parallel computing can accelerate the high-fidelity computation, the reduced-order approach offers even more substantial computational gains.

2.2.3 Verification: mesh convergence study

In this section, we detail the outcomes of our mesh convergence analysis. This study involves solving the given problem on various mesh configurations and subsequently comparing the resultant data. To conduct this analysis, we first solve the model denoted $\mathcal{E}_{\mathrm{NL}}(\bar{\mu})$. Following this, we compute the output $T^{\mathrm{fem}}_{\mathrm{NL},O}$ representing the temperature at the cornea's center as determined by the high-fidelity model.

Table 2.3 displays the characteristics of the meshes, including characteristic size h and the number of degrees of freedom (nDof) of the associated problem, for both $\mathbb{P}_1$ and $\mathbb{P}_2$ finite element discretizations. Since a 3D model is considered, the number of degrees of freedom is notably large, particularly with the $\mathbb{P}_2$ discretization, emphasizing the necessity of developing an efficient computational framework.

The primary objective of this analysis is to ascertain whether the obtained temperature values demonstrate convergence towards a consistent value. Figure 2.6 illustrates the results of our mesh convergence study, clearly indicating a pattern of satisfactory convergence. We select for further comparisons the values obtained for `M3` and for $\mathbb{P}_2$.

Mesh	h_{avg}	**nDof** $\mathbb{P}_1$	**nDof** $\mathbb{P}_2$
`M0`	$8.6 \cdot 10^{-4}$	47,284	$3.27 \cdot 10^5$
`M1`	$7.39 \cdot 10^{-4}$	68,993	$4.73 \cdot 10^5$
`M2`	$5.11 \cdot 10^{-4}$	$1.21 \cdot 10^5$	$8.83 \cdot 10^5$
`M3`	$4.69 \cdot 10^{-4}$	$2.08 \cdot 10^5$	$1.58 \cdot 10^6$
`M4`	$2.94 \cdot 10^{-4}$	$9.96 \cdot 10^5$	$7.87 \cdot 10^6$
`M5`	$1.5 \cdot 10^{-4}$	$7.36 \cdot 10^6$	$5.87 \cdot 10^7$

Table 2.3: Number of degree of freedom of the FE function space constructed from the meshes of the family `M`, for both $\mathbb{P}_1$ and $\mathbb{P}_2$ discretizations.

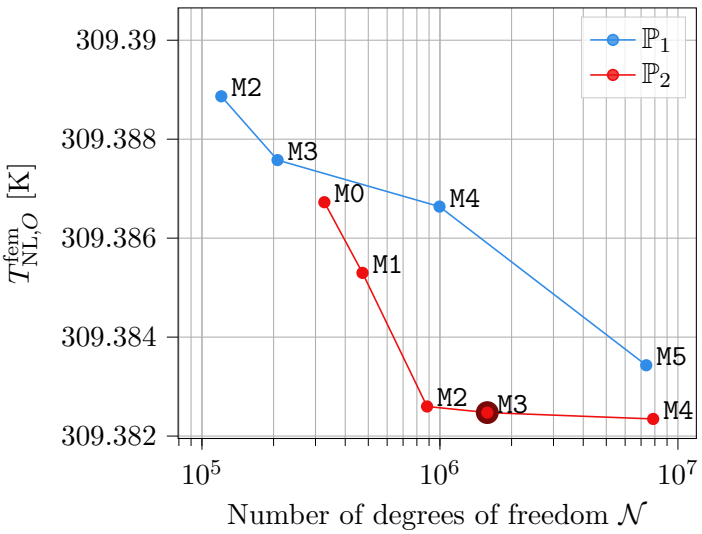


Figure 2.6: Temperature at the center of the cornea computed with the high-fidelity model $\mathcal{E}_{\text{NL}}(\bar{\mu})$, depending on the level of mesh refinement.

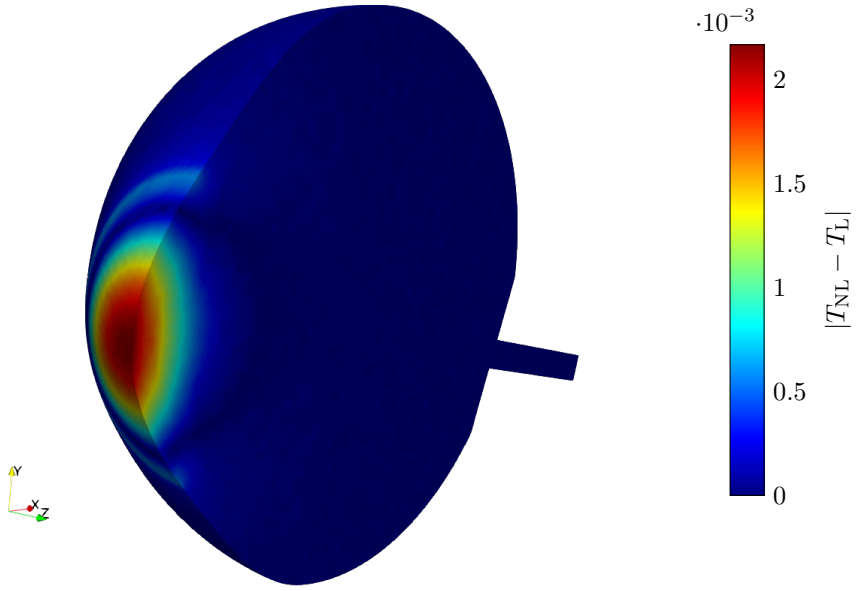


Figure 2.7: Difference of the temperature between the full model and the linearized model, computed on the mesh `M3` with $\mathbb{P}_2$ elements, and the baseline values $\bar{\mu}$ for the parameters.

2.2.4 Linearized model

We now compare the results obtained after solving $\mathcal{E}_{\mathrm{NL}}(\bar{\mu})$ against $\mathcal{E}_{\mathrm{L}}(\bar{\mu})$. We denote the solution of the nonlinear model $\mathcal{E}_{\mathrm{NL}}(\bar{\mu})$ by $T_{\mathrm{NL}}(\bar{\mu})$, and by $T_{\mathrm{L}}(\bar{\mu})$ the solution of the linearized model $\mathcal{E}_{\mathrm{L}}(\bar{\mu})$, and compute the relative error:

$$e_{\mathrm{lin}}(\mu) = \frac{\|T_{\mathrm{NL}}(\mu) - T_{\mathrm{L}}(\mu)\|_{L^2(\Omega)}}{\|T_{\mathrm{NL}}(\mu)\|_{L^2(\Omega)}}. \tag{2.7}$$

For $\mu = \bar{\mu}$, we get $e_{\mathrm{lin}}(\mu) = 4.176 \cdot 10^{-7}$. In Figure 2.7, we plot the difference between the two solutions $|T_{\mathrm{NL}}(\vec{x}) - T_{\mathrm{L}}(\vec{x})|$ for $\vec{x} \in \Omega$. We notice that the difference is the largest on the front of the eye, where the boundary condition has been changed, whereas at the back of the eye, the solutions are superposed. We also compute the maximal difference: $2.1667 \cdot 10^{-3}$ K. Therefore, we consider in the sequel that the linearized model does not induce a significant error in the results.

Figure 2.8 displays the results of the simulation of the linear model $\mathcal{E}_{\mathrm{L}}(\mu)$, for three parameters μ: $\bar{\mu}$ the baseline value parameters, $\mu_{\min}$ (resp. $\mu_{\max}$) where each component is the lowest (resp. highest) bound of its range of values, as defined in Table 1.1. This figure enlights the important impact of the parameters on the temperature distribution in the eye. This impact will be further analyzed in the sequel, in Chapter 5.

2.2.5 Validation and comparison with previous studies

We present in this section a thorough comparison between the results of this work and previously published data on the temperature of the eye, obtained either by experimental procedures or *via* computational modeling. Note that only scarce data are available for the entire human eyeball, since most of the measurement techniques estimated only the surface temperature of the cornea. In particular, [EYB89] gathers the outputs of 19 studies conducted with various instruments (mercury bulbs, liquid crystal thermometers or infrared thermometers), and the

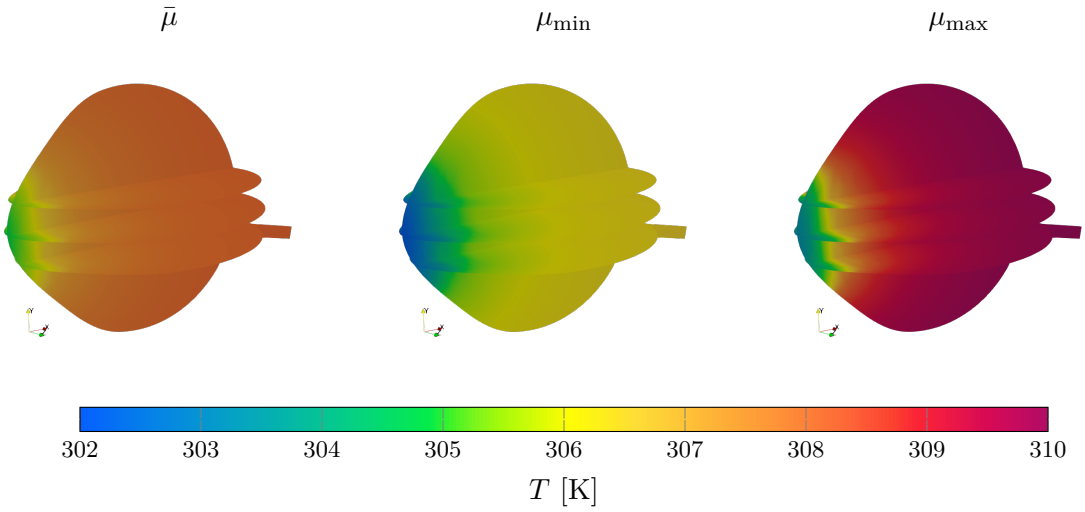


Figure 2.8: Distribution of the temperature in the eyeball from the linear model $\mathcal{E}_{\mathrm{L}}(\mu)$ for three different values of the parameters: the baseline values $\bar{\mu}$, the "minimal" parameter $\mu_{\min}$ and the "maximal" parameter $\mu_{\max}$.

mean value reported, according to [NO06], is $T_O^{\mathrm{exp}} = 307.15\,\mathrm{K}$. The temperature at the center of the cornea computed with baseline value from our model is $T_O^{\mathrm{fem}}(\bar{\mu}) = 306.02$ K, which lies in the interval of results from the literature (see [EYB89, Table 1] and [NO06, Table 9]).

Additionally, in [EYB89], the temperature is measured along an imaginary horizontal line, the *Geometrical Center of the Cornea* (GCC), as described in Figure 1.4, on a panel of 21 subjects. The experimental data are displayed in Figure 2.9, together with the findings of the present work. On the horizontal axis, the distance to the center of the eye is represented, and on the vertical axis is the temperature difference to the central one (mean value and standard deviation). Note that as the geometry of the simulated eye is not the same as the one used in the experiment, we scaled the results over the x-axis. The result shows that the high fidelity model is able to closely replicate the same behavior as the one experimentally measured, and the model $\mathcal{E}_{\mathrm{L}}(\bar{\mu})$ provides very close values (see Section 2.2.4). Moreover, thanks to the error bound introduced in Section 3.1.2 for the RBM, the approach is considered to be valid for the sensitivity analysis procedure hereafter.

In Figure 2.10, we present a comparative analysis between the results of our current study and various numerical findings reported in existing literature. This comparison features temperatures calculated along a line traversing the eye's center, the specific location of which is depicted in Figure 1.5. This comparative approach is crucial as it verifies the accuracy of our computed values, encompassing not just the corneal surface but also the eye's internal tissue structures. It is noteworthy that our analysis includes a mix of both 2D and 3D results, derived from both non-linear and linearized models. The results computed for the present model are shown for the linearized model $\mathcal{E}_{\mathrm{L}}(\bar{\mu})$ and is superposed to the results of the non-linear one $\mathcal{E}_{\mathrm{NL}}(\bar{\mu})$, as the difference between the two is negligible (see Section 2.2.4).

The results show a very good agreement between the findings of the present study and previously reported temperature results, along the different locations in the eyeball.

2.3 Conclusion

In this chapter, we have discussed the discretization of the geometrical model, permitting to generate meshes that accurately represent the eye's complex structure. Two families of mesh

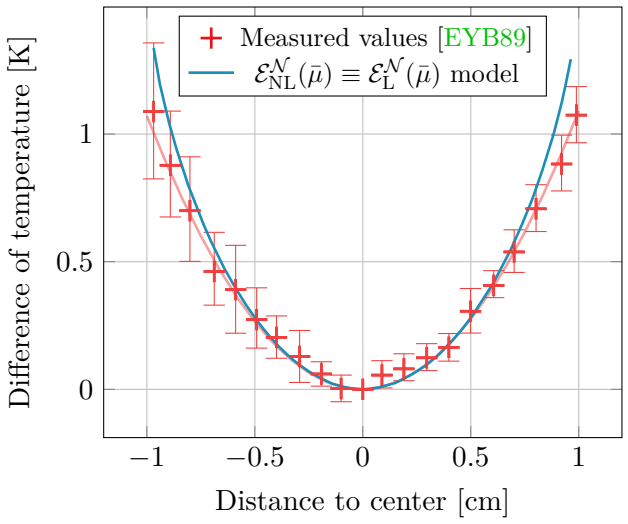


Figure 2.9: Temperature on the GCC: experimental data (mean and standard deviation) vs. numerical results. In this analysis, we cannot distinguish graphical difference between the linear and non-linear models.

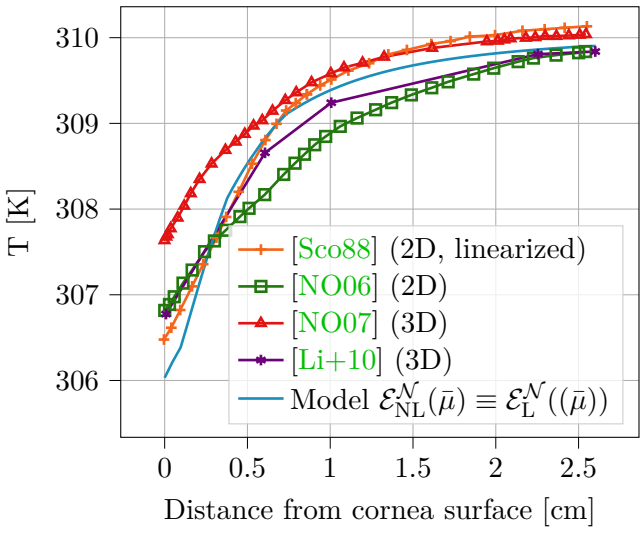


Figure 2.10: Temperature on a line going through the center of the eye (see Figure 1.5), comparison with numerical results from literature.

have been generated, of various refinement: A first one directly generated from the CAD data, for the heat transfer model presented in Chapter 1, and a second one, refined around the anterior and posterior chambers, that will be used for a coupled model of heat transfer and aqueous humor flow, presented in Chapter 6. The work committed to the mesh generation and refinement was published as an open-source dataset [Cha+24].

Then, we have presented the Finite Element Method, which is used to solve the partial differential equations governing the heat transfer in the eye. We also justified the choice of the FEM for this problem, and presented results of verification and validation of the method. The FEM has been successfully applied to models of heat transfer in the eye.

Thanks to this framework, we are now able to set up the Reduced Basis framework, presented in Chapter 3, to reduce the computational cost of the simulations, and to perform a sensitivity analysis, presented in Chapter 5.

Chapter 3

Reduced order computational framework: methods and verification

The framework of the finite element method presented in Section 2.2 is a powerful tool to solve complex problems, as it stems from a strong mathematical background. However, any change in the model parameters requires a new solve pf the problem from scratch and induces a high computational cost. This challenge becomes particularly pronounced in scenarios such as optimization, uncertainty quantification, and sensitivity analysis—areas we intend to explore in the context of the heat transfer eye model introduced in Chapter 1.

To mitigate these computational challenges, we can develop Model Order Reduction (MOR) strategies. MOR techniques have evolved significantly, offering powerful tools to reduce the computational complexity of high-dimensional models across various fields. These methods aim to approximate the solution of complex systems by reducing the dimensionality of the problem, while maintaining an acceptable level of accuracy. This reduction is often achieved by constructing surrogate models that capture the most significant features of the full-order model, enabling faster simulations without a complete loss of fidelity. On one hand, the physical model itself can be simplified, as we did in the process of linearization in Remark 1.4.1. Even though we verified that such linearization does not result in significant loss of accuracy (see Section 2.2.4), this valid simplification is not always feasible. On the other hand, methods can be applied where the model equations are projected onto a lower-dimensional space. For instance, we can utilize Proper Generalized Decomposition [CLC11], Proper Orthogonal Decomposition (POD) [Ker+05], the Reduced Basis Method (RBM), or the Non-Intrusive Reduced Basis (NIRB) method. Recently, data-driven approaches, such as those incorporating dynamical models and snapshot-based techniques [DDS23], have emerged to further enhance prediction accuracy and efficiency. Alongside these, machine learning techniques, especially Physics-Informed Neural Networks (PINNs) [RPK19], are becoming a promising alternative for complex systems by integrating physical laws with data-driven models. This hybrid approach allows for real-time simulation and accurate estimations in fields like biomedical engineering and beyond. In Appendix B, we explore the application of PINNs to microbial community interactions, demonstrating how machine learning can be harnessed for dynamic system modeling and parameter estimation.

In this chapter, we focus on the Reduced Basis Method (Section 3.1) and the Non-Intrusive Reduced Basis method (Section 3.2), as the model developed for thermal transfer in the eye model is well-suited for these methods. The Reduced Basis method has been applied to the heat transfer problem in the eye model, and we have shown that the method is efficient to significantly reduce the computational cost of solving the parametric problem, and that the results are accurate, thanks to the error bound provided by the method.

3.1 Reduced order modeling with the reduced basis method

The Reduced Basis Method (RBM) is a computational technique for solving parametrized partial differential equations efficiently. It reduces the problem's complexity by constructing a low-dimensional approximation space from precomputed "snapshot" solutions, obtained during an offline phase. These snapshots capture the essential features of the solution space. In the online phase, the RBM uses this reduced basis to approximate solutions rapidly, making it ideal for applications requiring repeated simulations, such as optimization and uncertainty quantification, while maintaining high accuracy.

The RBM has been introduced in the 1970s for nonlinear structural analysis [ASB78; NP80], and has been extended later to multiparameter problems [FR83; Bal96; Rhe93], or to problems from other fields such as incompressible flows [Pet89]. In the 2000s, the method has been improved by the development of efficient error estimator, such as the Certified Reduced Basis (CRB) [Pru+02; VPP03; RHP08; QMN16], ensuring the reliability of the reduced basis approximation. The error certified by the CRB allow setting up an offline/online strategy that will be detailed in this section. Such strategy allows to explore the parametric domain to efficiently build the reduced basis.

3.1.1 The Reduced Basis Method

We introduce in this section the reduced basis approach [Pru+02; VPP03; RHP08; QMN16]. The goal of the reduced basis method (RBM) is to approximate the solution of the parametrized-PDE described by a system of equations. For complex geometries and biomechanical problems, such as the one described in Section 1.3, numerical solving has a prohibitive cost, especially for studies of uncertainty quantification, requiring the resolution of the system for many parameters. We present the implemented strategy, following [Pru+02]. In this section, we describe the methodology in a general layout, the results presented will be computed using the thermal transfer model $\mathcal{E}_{\mathrm{L}}$ described in Section 1.4. The general framework of this method was already implemented in the Feel++ library [Vey14], we adapted this framework to the specific problem of heat transfer in the eye. More details about the implementation within the Feel++ library are provided in Section 7.2.

We are looking to the solution to a variational problem stemming from a parametrized PDE written as: find $u(\mu) \in V$ such that

$$a(u(\mu), v; \mu) = f(v; \mu) \quad \forall v \in V, \tag{3.1}$$

and evaluate the output of interest $s(\mu)$ defined by

$$s(\mu) = \ell(u(\mu); \mu). \tag{3.2}$$

The solution of the PDE $u(\mu)$, also called the *exact solution*, belongs to the *solution manifold* $\mathcal{M}$ defined as

$$\mathcal{M} = \{u(\mu) | \mu \in D^\mu\}. \tag{3.3}$$

We aim at approximating this solution manifold by a low-dimensional subspace V_N of small dimension, as shown in Figure 3.1.

As presented in Section 2.2, we employ the FEM to numerically solve this problem, so we have a finite dimensional space $V_h \subset V$, and we are computing the solution $u^{\text{fem}}(\mu) \in V_h$ to the weak formulation

$$a(u^{\text{fem}}(\mu), v; \mu) = f(v; \mu) \quad \forall v \in V_h. \tag{3.4}$$

Remark 3.1.1. In particular, we can choose the output such as $s(\mu) = f(u(\mu); \mu)$. In this context, the output is said to be *compliant*.

We denote by $\mathcal{N}$ the dimension of V_h, and we set $(\varphi_h^n)_{n \in [\![1, \mathcal{N}]\!]}$ a basis of V_h provided by the FEM. We call the solution u^{fem} to be the *high fidelity* (HF) solution. Under their algebraic representation, Equations (3.2) and (3.4) read:

$$\underline{\underline{\boldsymbol{A}}}(\mu)\underline{\boldsymbol{u}}^{\text{fem}}(\mu) = \underline{\boldsymbol{f}}(\mu), \tag{3.5a}$$

$$s(\mu) = \underline{\boldsymbol{L}}(\mu)^T \underline{\boldsymbol{u}}^{\text{fem}}(\mu). \tag{3.5b}$$

As the dimension $\mathcal{N}$ can be very large, it could be challenging to efficiently compute the solution, especially in the context of sensitivity analysis where numerous evaluation are required, for various parameters of the PDE. The main idea of RBM is to construct a low-dimension subspace $V_N \subset V_h$, of dimension N with $N \ll \mathcal{N}$, such that the approximation error is small: $\|u^{\text{fem}}(\mu) - u^{\text{rbm}}(\mu)\|_V \leqslant \varepsilon_{\text{tol}}$, while the procedure to compute $u^{\text{rbm}}(\mu)$ is efficient and stable. The *reduced solution* $u^{\text{rbm}}(\mu) \in V_N$ is computed by solving the reduced weak formulation: find $u^{\text{rbm}}(\mu) \in V_N$ such that given $\mu \in D^\mu$:

$$a(u^{\text{rbm},N}(\mu), v; \mu) = f(u; \mu) \quad \forall v \in V_N. \tag{3.6}$$

The space V_N is constructed to efficiently approximate the manifold $\mathcal{M}$. Precisely, it is constructed from *snapshots*, which are high fidelity solutions. The RBM consists of two main phases:

(i) the *offline stage*, where the reduced space is constructed, and

(ii) the *online stage*, where the reduced space is used to compute the solution of the system.

The first step is performed only once and can be costly, whereas the second step is performed for each parameter μ and is efficient.

We present in Figure 3.1 a schematic representation of the methodology, for a finite number of snapshots $u(\mu_1), \cdots, u(\mu_N)$.

During the offline stage, snapshots are computed for a set of parameters $\{\mu_i\}_{i=1}^N \subset D^\mu$. This provides a family of vectors, the *snapshots*, $\left(u^{\text{fem}}(\mu_i)\right)_{1 \leqslant i \leqslant N} \subset V_h$. The reduced space is defined by $V_N := \text{span}\left(\underline{\boldsymbol{\xi}}\right)$, where $(\underline{\boldsymbol{\xi}}_i)_{1 \leqslant i \leqslant N}$ is an orthonormal family of vectors, obtained by the Gram-Schmidt process applied to the snapshots $\{u^{\text{fem}}(\mu_i)\}_{1 \leqslant i \leqslant N}$. We define the snapshots matrix $\underline{\underline{\boldsymbol{Z}}}_N = \left[\underline{\boldsymbol{\xi}}_1, \cdots, \underline{\boldsymbol{\xi}}_N\right] \in \mathbb{R}^{\mathcal{N} \times N}$.

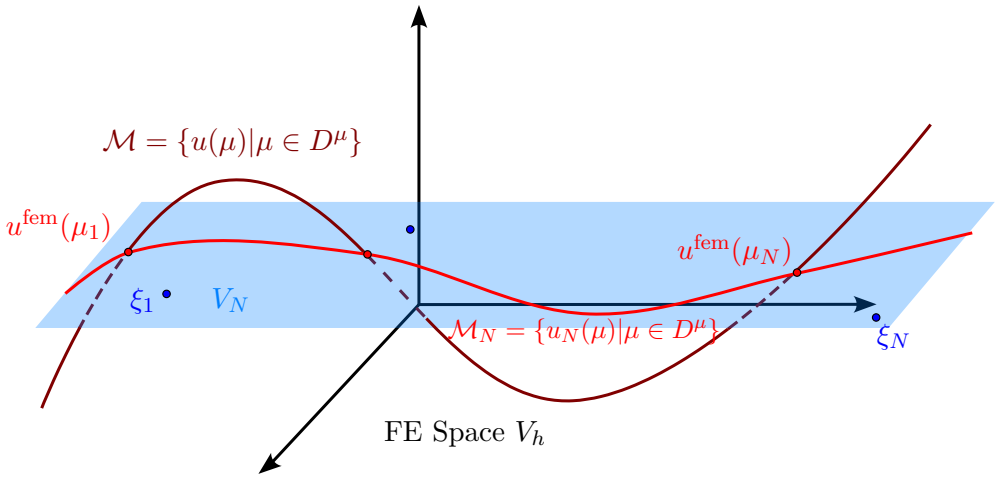


Figure 3.1: Schematic representation of the reduced basis method, with the construction of the reduced space V_N from the snapshots $u^{\text{fem}}(\mu_i)$.

The snapshots can be selected in different ways. The first approach is to select the snapshots randomly in the parameter space, but this could lead to a poor approximation of the solution [Buf+12]. Another approach is to select the snapshots greedily, by selecting the parameter that maximizes the error between the reduced solution and the high fidelity solution:

$$\mu_{N+1} = \arg\max_{\mu \in D^\mu} \left\| u^{\text{fem}}(\mu) - u^{\text{rbm},N}(\mu) \right\|_V . \tag{3.7}$$

More details about this greedy procedure are provided in Section 3.1.3, as this second strategy is the one implemented in our model.

Let $v_N \in V_N$. It can be expressed in the basis $(\underline{\boldsymbol{\xi}}_i)_{i \in [\![1,N]\!]}$

$$v_N = \sum_{i=1}^{N} v_{N,i} \underline{\boldsymbol{\xi}}_i = \underline{\underline{\boldsymbol{Z}}}_N \underline{\boldsymbol{v}}_N, \tag{3.8}$$

where $\underline{\boldsymbol{v}}_N$ is the algebraic representation of $v_N \in V_N$. Inserting this expression in the algebraic equation Equation (3.5b), we obtain

$$\begin{aligned} \underline{\underline{\boldsymbol{A}}}(\mu) \underline{\underline{\boldsymbol{Z}}}_N \underline{\boldsymbol{v}}_N &= \underline{\boldsymbol{f}}(\mu), \\ \underline{\underline{\boldsymbol{Z}}}_N^T \underline{\underline{\boldsymbol{A}}}(\mu) \underline{\underline{\boldsymbol{Z}}}_N \underline{\boldsymbol{v}}_N &= \underline{\underline{\boldsymbol{Z}}}_N^T \underline{\boldsymbol{f}}(\mu). \end{aligned} \tag{3.9}$$

Setting $\underline{\underline{\boldsymbol{A}}}_N(\mu) := \underline{\underline{\boldsymbol{Z}}}_N^T \underline{\underline{\boldsymbol{A}}}_L(\mu) \underline{\underline{\boldsymbol{Z}}}_N \in \mathbb{R}^{N \times N}$ and $\underline{\boldsymbol{f}}_N(\mu) := \underline{\underline{\boldsymbol{Z}}}_N^T \underline{\boldsymbol{f}}_L(\mu) \in \mathbb{R}^N$, we obtain the reduced algebraic system of size N:

$$\underline{\underline{\boldsymbol{A}}}_N(\mu) \underline{\boldsymbol{u}}^{\text{rbm},N}(\mu) = \underline{\boldsymbol{f}}_N(\mu), \tag{3.10a}$$

$$s_N(\mu) = \underline{\boldsymbol{L}}_N(\mu)^T \underline{\boldsymbol{T}}^{\text{rbm},N}(\mu), \tag{3.10b}$$

the same process applying for the outputs $\underline{\boldsymbol{L}}$.

We have constructed an algebraic system of size N to solve the problem, but still an issue remains: the cost to compute the reduced matrices $\underline{\underline{\boldsymbol{A}}}_N(\mu)$ and $\underline{\boldsymbol{f}}_N(\mu)$ is still high, because it still depends on the high fidelity dimension $\mathcal{N}$, involved in the matrix multiplications. The idea to overcome this issue is to exploit the *affine decomposition* of the problem, whenever possible.

Assume that we can decompose the forms $a(\cdot,\cdot;\mu)$, $f(\cdot;\mu)$, and $\ell(\cdot;\mu)$ in an affine way:

$$a(u,v;\mu) = \sum_{q=1}^{Q_a} \beta_A^q(\mu) a^q(u,v), \tag{3.11a}$$

$$f(v;\mu) = \sum_{p=1}^{Q_f} \beta_F^p(\mu) f^p(v), \tag{3.11b}$$

$$\ell(u;\mu) = \sum_{k=1}^{Q_\ell} \beta_\ell^k(\mu) \ell^k(u), \tag{3.11c}$$

where Q_a, Q_f, and Q_ℓ are the number of terms in the decomposition.

We furthermore define the algebraic matrices $\underline{\underline{\boldsymbol{A}}}_L^q \in \mathbb{R}^{\mathcal{N}\times\mathcal{N}}$ and vectors $\underline{\boldsymbol{f}}_L^p \in \mathbb{R}^{\mathcal{N}}$, so the following equality holds:

$$\underline{\underline{\boldsymbol{A}}}_L(\mu) = \sum_{q=1}^{Q_a} \beta_A^q(\mu) \underline{\underline{\boldsymbol{A}}}_L^q, \quad \underline{\boldsymbol{f}}_L(\mu) = \sum_{p=1}^{Q_f} \beta_F^p(\mu) \underline{\boldsymbol{f}}_L^p. \tag{3.12}$$

Moreover, the affine decomposition of the bilinear form a is *parametrically coercive*, meaning that:

$$\beta_A^q(\mu) > 0 \quad \forall q \in [\![1, Q_a]\!], \tag{3.13a}$$

$$a^q(u,u;\mu) \geqslant 0 \quad \forall q \in [\![1, Q_a]\!], \quad \forall u \in V_h, \forall \mu \in D^\mu. \tag{3.13b}$$

From this decomposition and Equation (3.10a), we obtain the following algebraic system:

$$\underline{\underline{\boldsymbol{A}}}_N(\mu) = \sum_{q=1}^{Q_a} \beta_A^q(\mu) \underbrace{\underline{\underline{\boldsymbol{Z}}}_N^T \underline{\underline{\boldsymbol{A}}}_L^q \underline{\underline{\boldsymbol{Z}}}_N}_{\underline{\underline{\boldsymbol{A}}}_N^q}. \tag{3.14}$$

We set $\underline{\underline{\boldsymbol{A}}}_N^q := \underline{\underline{\boldsymbol{Z}}}_N^T \underline{\underline{\boldsymbol{A}}}_L^q \underline{\underline{\boldsymbol{Z}}}_N \in \mathbb{R}^{N\times N}$. The matrices $\underline{\underline{\boldsymbol{A}}}_N^q \in \mathbb{R}^{N\times N}$ are independent of μ and can be computed only once and stored. The same process applies to $\underline{\boldsymbol{f}}_N(\mu)$ and $\underline{\boldsymbol{L}}_{k,N}(\mu)$:

$$\underline{\boldsymbol{f}}_N(\mu) = \sum_{q=1}^{Q_f} \beta_F^q(\mu) \underline{\boldsymbol{f}}_N^q, \quad \underline{\boldsymbol{L}}_N(\mu) = \sum_{q=1}^{Q_\ell} \beta_\ell^q(\mu) \underline{\boldsymbol{L}}_N^q. \tag{3.15}$$

This decomposition allows implementing an *offline/online procedure*. During the *offline phase*, the basis of V_N is constructed from the snapshots, as well as the matrices $\underline{\underline{\boldsymbol{A}}}_N^q$, $\underline{\boldsymbol{f}}_N^q$, and $\underline{\boldsymbol{L}}_N^q$ that are computed and stored. More details about this construction are given in Section 3.1.3. This procedure can be computationally costly, but it is performed only once for the problem. During the *online phase*, the reduced system Equation (3.10) is solved for any parameter μ. The entire procedure is synthesized in Algorithms 2 and 3.

During the offline stage, two approaches can be used to select the size of the reduced basis N:

(i) an approach where we set the size of the reduced basis N to a fixed value, and

(ii) an approach where we set a tolerance ε_{tol} on the error committed on the output.

The second approach is more interesting since it allows having a reduced basis of size N that is adapted to the desired tolerance, and it is the one implemented in our model.

Algorithm 2: Offline stage of the RBM.

Input: Parameters $\mu_1, \cdots, \mu_N \in D^\mu$.
Compute snapshots $\underline{\boldsymbol{u}}^{\text{fem}}(\mu_1), \cdots, \underline{\boldsymbol{u}}^{\text{fem}}(\mu_N)$;
Construct $\underline{\underline{\boldsymbol{Z}}}_N \leftarrow \left[\underline{\boldsymbol{\xi_1}}, \cdots, \underline{\boldsymbol{\xi_N}}\right]$ (orthonormal);
Construct the reduced matrices $(\underline{\underline{\boldsymbol{A}}}_N^q)_{1\leqslant q\leqslant Q_a}$, $(\underline{\boldsymbol{F}}_N^p)_{1\leqslant p\leqslant Q_f}$, $\underline{\boldsymbol{L}}_N$;
Output: Reduced basis and reduced matrices, stored.

Algorithm 3: Online stage of the RBM.

Input: $\mu \in D^\mu$.
Assemble $\underline{\underline{\boldsymbol{A}}}_N(\mu)$, $\underline{\boldsymbol{F}}_N(\mu)$, $\underline{\boldsymbol{L}}_N(\mu)$ using the saved matrices and the affine decomposition;
Solve $\underline{\underline{\boldsymbol{A}}}_N(\mu)\underline{\boldsymbol{u}}_N(\mu) = \underline{\boldsymbol{F}}_N(\mu)$;
Compute the output $s_N(\mu) = \underline{\boldsymbol{L}}_N(\mu)^T\underline{\boldsymbol{u}}_N(\mu)$;
Output: $\underline{\boldsymbol{u}}_N(\mu), s_N(\mu)$.

Remark 3.1.2. Note that in the context of heat transfer problem described in Section 1.4, such a decomposition is possible for the model $\mathcal{E}_{\text{L}}$: for $T, v \in V$,

$$a_L(T, v; \mu) = \sum_{q=1}^{Q_a} \beta_A^q(\mu) a_L^q(T, v), \tag{3.16a}$$

with

$$\beta_A^1(\mu) = k_{\text{lens}} \qquad a_L^1(T, v) = \int_{\Omega_{\text{lens}}} \nabla T \cdot \nabla v \, \mathrm{d}\vec{x}, \tag{3.16b}$$

$$\beta_A^2(\mu) = h_{\text{amb}} \qquad a_L^2(T, v) = \int_{\Gamma_{\text{amb}}} Tv \, \mathrm{d}\sigma, \tag{3.16c}$$

$$\beta_A^3(\mu) = h_{\text{bl}} \qquad a_L^3(T, v) = \int_{\Gamma_{\text{body}}} Tv \, \mathrm{d}\sigma, \tag{3.16d}$$

$$\beta_A^4(\mu) = 1 \qquad a_L^4(T, v) = \int_{\Gamma_{\text{amb}}} h_{\text{r}} Tv \, \mathrm{d}\sigma + \sum_{i\neq\text{lens}} k_i \int_{\Omega_i} \nabla T \cdot \nabla v \, \mathrm{d}\vec{x}, \tag{3.16e}$$

and

$$f_L(v; \mu) = \sum_{p=1}^{Q_f} \beta_F^p(\mu) f_L^p(v), \tag{3.17a}$$

with

$$\beta_F^1(\mu) = h_{\text{amb}} T_{\text{amb}} + h_{\text{r}} T_{\text{amb}} - E \qquad f^1(v) = \int_{\Gamma_{\text{amb}}} v \, \mathrm{d}\sigma, \tag{3.17b}$$

$$\beta_F^2(\mu) = h_{\text{bl}} T_{\text{bl}} \qquad f^2(v) = \int_{\Gamma_{\text{body}}} v \, \mathrm{d}\sigma, \tag{3.17c}$$

where $Q_a = 4$ and $Q_f = 2$. The various outputs of interest are already in a decomposed form, with only one term in the decomposition.

3.1.2 Error estimates

A priori convergence theory

Definition 3.1.3. We introduce the *inner scalar product* and the induced *energy norm*: for $u, v \in V$ and $\mu \in D^\mu$, $\langle u, v\rangle_\mu := a(u, v; \mu)$ and $\|v\|_\mu := \sqrt{\langle v, v\rangle_\mu}$.

Proposition 3.1.4 ([EG21]). *Let $\mu \in D^\mu$. Assume that the bilinear form $a(\cdot,\cdot;\mu)$ satisfies the hypothesis From the hypothesis of the bilinear form $a(\cdot,\cdot;\mu)$, then the solution of Equation (3.6) $u_N(\mu) \in V_N$ satisfies the following property:*

$$u_N(\mu) = \arg\min_{v \in V_N} \|u(\mu) - v\|_\mu . \tag{3.18}$$

Proof. Recall that the solution u_N minimizes the functional $J\colon V \to \mathbb{R}, w \mapsto \frac{1}{2}a(w,w;\mu) - \ell(w;\mu)$, as a consequence of the Lax-Milgram theorem. For clarity, the parameter dependence is omitted. Let $v_N \in V_N$. Using the optimality of u_N, we have $J(u_N) \leqslant J(v_N)$. Hence:

$$\begin{aligned}\|u - u_N\|_\mu &= a(u-u_N, u-u_N) = a(u,u) - 2a(u_N,u) + a(u_N,u_N)\\ &= a(u,u) - 2\ell(u_N) + a(u_N,u_N) = a(u,u) + J(u_N)\\ &\leqslant a(u,u) + J(v) = \|u - v\|_\mu .\end{aligned}$$

■

We focus on the convergence rate of the field $u_N(\mu)$ to $u(\mu)$, and of the *compliant* output $s_N(\mu)$ to $s(\mu)$, as $N \to \infty$.

Proposition 3.1.5 ([Pru+02]). *The following optimality property of the reduced basis approximation $u_N(\mu)$ stands:*

$$\|u(\mu) - u_N(\mu)\|_V = \sqrt{\frac{\gamma(\mu)}{\alpha(\mu)}} \inf_{v_N \in V_N} \|u(\mu) - v_N\|_V . \tag{3.19}$$

where $\alpha(\mu)$ and $\gamma(\mu)$ are the coercivity and continuity constants of $a(\cdot,\cdot;\mu)$, respectively.

Moreover, for a compliant output, $s_N(\mu)$ is a lower bound of $s(\mu)$, and the error $s(\mu) - s_N(\mu)$ converges as the square of the error in the best approximation:

$$s(\mu) - s_N(\mu) \leqslant \gamma(\mu) \|u(\mu) - u_N(\mu)\|_V^2 . \tag{3.20}$$

Proof. From Proposition 3.1.4, we have $a(u(\mu) - u_N(\mu), u(\mu) - u_N(\mu);\mu) \leqslant a(u(\mu) - v_N, u(\mu) - v_N;\mu)$ for all $v_N \in V_N$. Using coercivity and continuity of $a(\cdot,\cdot;\mu)$, we obtain:

$$\alpha(\mu) \|u(\mu) - u_N(\mu)\|_V^2 \leqslant \gamma(\mu) \|u(\mu) - v_N\|_V^2 . \tag{3.21}$$

That proves the first result.

Moreover, for compliant output, we have

$$\begin{aligned}s(\mu) - s_N(\mu) &= f(u(\mu);\mu) - f(u(\mu);\mu) = a(u(\mu),u(\mu);\mu) - a(u_N(\mu),u_N(\mu);\mu)\\ &= a(u - u_N, a - u_N;\mu),\end{aligned} \tag{3.22}$$

from symmetry of $a(\cdot,\cdot;\mu)$, and Galerkin orthogonality. The second result follows from this equation and from the coercivity of $a(\cdot,\cdot;\mu)$. ■

Error bound

Given the reduced basis approximation $u^{\mathrm{rbm},N}(\mu)$ of the HF solution $u^{\mathrm{fem}}(\mu)$ for $\mu \in D^\mu$, we define the *field error* as

$$e(\mu) := u^{\mathrm{fem}}(\mu) - u^{\mathrm{rbm},N}(\mu). \tag{3.23}$$

We aim to construct quantities $\Delta_N(\mu)$ and $\Delta_N^s(\mu)$ such that

$$\|e(\mu)\|_V \leqslant \Delta_N(\mu) \quad \text{and} \quad s(\mu) - s_N(\mu) \leqslant \Delta_N^s(\mu). \tag{3.24}$$

These quantities are called *a posteriori error bounds* [Pru+02; RHP07]. To quantify the *sharpness* and *rigor* properties of the error bound, we introduce the *effectivity*:

$$\eta_N(\mu) := \frac{\Delta_N(\mu)}{\|e(\mu)\|_V}, \qquad \eta_N^s(\mu) := \frac{\Delta_N^s(\mu)}{s(\mu) - s_N(\mu)}. \tag{3.25}$$

It has been proven in [Pru+02] that the error bound is *rigorous*, namely that it is always greater than the error; and *sharp*, namely that it is as close as possible to the actual error.

These properties can be summarized as:

$$1 \leqslant \eta_N(\mu) \leqslant \eta_{\text{ub}}(\mu) \quad \forall \mu \in D^\mu, \tag{3.26}$$

where $\eta_{\text{ub}}(\mu)$ is the sharpness of the bound and is proven to be bounded when N increases [Pru+02].

Finally, to construct the reduced space, we require the error bound to be *efficient*, that is, its evaluation is independent of the size of the high fidelity space $\mathcal{N}$. This is critical when heuristic algorithms are used to construct the reduced space, such as the Greedy algorithm discussed in Section 3.1.3.

Such an error bound can be constructed efficiently from the *residual* r of the problem (1.6), a lower bound $\alpha_{\text{lb}}(\mu)$ of the coercivity constant $\alpha(\mu)$ of $a_L(\cdot,\cdot;\mu)$, and the affine decomposition of a_L and f_L, as follows:

$$\alpha(\mu) = \inf_{v \in V} \frac{a(v,v;\mu)}{\|v\|_V^2}, \quad r(v,\mu) := \ell(v;\mu) - a(u_N(\mu),v;\mu) \quad \forall v \in V \quad \text{and} \quad \Delta_N := \frac{\|r(\cdot,\mu)\|_{V'}}{\alpha_{\text{lb}}(\mu)}. \tag{3.27}$$

For more details, we refer to [Pru+02]. The *Certified Reduced Basis* (CRB) method is characterized by the inclusion of this error bound. This guarantees not only the efficiency of the reduced basis approximation but also its reliability.

A posteriori error estimation

Thanks to the bilinearity of $a(\cdot,\cdot;\mu)$, the error $e(\mu)$ satisfies the following variational problem:

$$a(e(\mu),v;\mu) = \ell(v;\mu) - a(u_N(\mu),v;\mu) \quad \forall v \in V. \tag{3.28}$$

We set the *residual* $r(\mu)$ as

$$r(v,\mu) := \ell(v;\mu) - a(u_N(\mu),v;\mu) \quad \forall v \in V. \tag{3.29}$$

We set $\alpha_{\text{lb}}(\mu)$ as a lower bound of $\alpha(\mu)$ over D^μ and $\gamma_{\text{ub}}(\mu)$ as an upper bound of $\gamma(\mu)$ over D^μ. Such quantities are well-defined since $a(\cdot,\cdot;\mu)$ is coercive and $\gamma(\mu)$ is bounded.

Proposition 3.1.6 ([Pru+02])**.** *A more precise lower bound for $\alpha(\mu)$ can be given by:*

$$\alpha_{\text{lb}}(\mu) = \alpha(\bar{\mu}) \min_q \frac{\beta_a^q(\mu)}{\beta_a^q(\bar{\mu})}. \tag{3.30}$$

Proof. We have $\forall \mu \in D^\mu$:

$$\begin{aligned}\alpha(\mu) &= \min_{v \in V} \frac{a(v,v:\mu)}{a(v,v;\bar{\mu})} = \min_{v \in V} \frac{\sum_q \beta_a^q(\mu) a^q(v,v)}{a(v,v;\bar{\mu})} \times \frac{\beta_a^q(\bar{\mu})}{\beta_a^q(\bar{\mu})} \\ &\geqslant \min_q \frac{\beta_a^q(\mu)}{\beta_a^q(\bar{\mu})} \min_{v \in V} \frac{\sum_q \beta_a^q(\bar{\mu}) a^q(v,v)}{a(v,v;\bar{\mu})} = \min_q \frac{\beta_a^q(\mu)}{\beta_a^q(\bar{\mu})} \alpha(\bar{\mu}).\end{aligned}$$

■

This provides *a* lower bound of $\alpha(\mu)$, but it may not be *the* greatest lower bound. To find a better lower bound, we can use the following procedure: We take $\bar{\bar{\mu}}_i$ for $i \in [\![1, \bar{\bar{N}}]\!]$ a set of parameters in D^μ, with $\bar{\bar{N}} \in \mathbb{N} \setminus \{0\}$, and compute during the off-line stage the values of $\alpha(\bar{\bar{\mu}}_i)$. During the online stage for $\mu \in D^\mu$, we set $\alpha_{\text{lb}}(\mu) = \max_{i \in [\![1,\bar{\bar{N}}]\!]} \alpha(\bar{\bar{\mu}}_i) \min_q \frac{\beta_a^q(\mu)}{\beta_a^q(\bar{\bar{\mu}})}$. This quantity can be computed efficiently.

Then, we set the following error estimate

$$\frac{\|r(\mu)\|_{V'}}{\alpha_{\text{lb}}(\mu)} := \Delta_N(\mu). \tag{3.31}$$

Theorem 3.1.7 ([Pru+02])**.** *The error bound $\Delta_N(\mu)$ defined in Equation (3.31) is sharp, rigorous, and efficient.*

Proof. Using the continuity of $a(\cdot, \cdot; \mu)$, we have $\forall v \in V$

$$|r(v, \mu)| \leqslant \gamma(\mu) \|e(\mu)\|_V \|v\|_V . \tag{3.32a}$$

Hence, using the dual norm

$$\|r(\mu)\|_{V'} \leqslant \gamma(\mu) \|e(\mu)\|_V . \tag{3.32b}$$

So by definition of $\Delta_N(\mu)$ and of the effectivity, we have $\eta_N^{\text{en}}(\mu) \leqslant \frac{\gamma(\mu)}{\alpha(\mu)} \leqslant \frac{\gamma(\mu)}{\alpha_{\text{lb}}}$. Hence, the rigorous property.

Moreover, using the coercivity of $a(\cdot, \cdot; \mu)$, we have

$$\alpha(\mu) \|e(\mu)\|_V \leqslant \|r(\mu)\|_{V'} , \tag{3.32c}$$

and so $\eta_N^{\text{en}}(\mu) \geqslant 1$, hence the sharpness property.

Finally, $r(\mu)$ is only dependent of the reduced space V_N, and α_{lb} is independent of μ and $\mathcal{N}$, so the evaluation of $\Delta_N(\mu)$ is efficient.

■

We need to highlight the fact that the reduced problem as well as the computation of the error bound $\Delta_N(\mu)$ are independent of the size of the high fidelity space $\mathcal{N}$. They only depend on the size of the reduced basis N. This is a key property for the efficiency of the reduced basis method, as it allows to construct the reduced basis space using heuristic algorithms such as the Greedy algorithm, as discussed in Section 3.1.3.

Dual problem

The theory presented in the previous section is based on the fact that the output selected is compliant, meaning that $\ell(\cdot; \mu) = f(\cdot; \mu)$. This is not the case for the thermal model $\mathcal{E}_{\text{L}}(\mu)$, as ℓ is a general a bounded linear form on V. We have the following *a priori* error estimate:

$$|s(\mu) - s_N(\mu)| = |\ell(u(\mu)) - \ell(u_N(\mu))| \leqslant \|\ell\|_{V'} \|u(\mu) - u_N(\mu)\|_V .$$

We notice that we lost the square convergence of the error obtained in the complaint case.

To overcome this, we introduce the *dual problem* [Pru+02] associated to ℓ: find $\psi(\mu) \in V$ such that:

$$a(v, \psi(\mu); \mu) = -\ell(v; \mu) \quad \forall v \in V, \tag{3.33}$$

$\psi(\mu)$ is called the *adjoint* or *dual* field.

We introduce the *primal residual* $r^{\mathrm{pr}}(v;\mu) := f(v;\mu) - a(u^{\mathcal{N}}(\mu), v;\mu)$ for all $v \in V$. The problem to solve is therefore: given $\mu \in D^{\mu}$, find $u^{\mathcal{N}}(\mu) \in V$ such that

$$a(u^{\mathcal{N}}(\mu), v;\mu) = f(v;\mu) \quad \forall v \in V, \tag{3.34}$$

and evaluate the output $s^{\mathcal{N}}(\mu) = \ell(u^{\mathcal{N}}(\mu)) - r^{\mathrm{pr}}(\psi(\mu))$, where $\psi(\mu)$ is the solution of the dual problem.

In Section 3.1.2, we saw that for a compliant output, the reduced output s_N is a lower bound for $s(\mu)$. From that, we can introduce the following prediction for $\mu \in D^{\mu}$: $s_N^{\mathrm{pred}} = s_N(\mu) + \frac{\Delta_N^s(\mu)}{2}$, with the following bounds: $s_N^{\mathrm{pred}} \in [s_N(\mu), s_N(\mu) + \Delta]$.

We now have:

$$\begin{aligned} |s(\mu) - s_N(\mu)| &= |\ell(u(\mu)) - \ell(u_N(\mu))| \\ &= |a(u(\mu) - u_N(\mu), \psi(\mu);\mu) \\ &= |a(u(\mu) - u_N(\mu), \psi(\mu) - \psi_N(\mu);\mu)| \\ &\leqslant \gamma(\mu) \left\|u(\mu) - u_N(\mu)\right\|_V \left\|\psi(\mu) - \psi_N(\mu)\right\|_V , \end{aligned} \tag{3.35}$$

from Galerkin orthogonality, and from the continuity of $a(\cdot,\cdot;\mu)$.

From Equation (3.35) we understand why we solve the dual problem: in order to obtain a precise approximation of the output $s(\mu)$, we utilize the approximation solution of the dual problem $\psi_N(\mu)$ to recover the square convergence of the error in the output.

The error bound is then:

$$\Delta_N^s(\mu) := \frac{\left\|r^{\mathrm{pr}}(\cdot;\mu)\right\|_{V'}}{\sqrt{\alpha(\mu)}} \frac{\left\|r^{\mathrm{du}}(\cdot;\mu)\right\|_{V'}}{\sqrt{\alpha(\mu)}}, \tag{3.36}$$

where $r^{\mathrm{du}}(v;\mu) := -\ell(v;\mu) + a(v, \psi(\mu);\mu)$ is the *dual residual*.

The theory of certified reduced basis requires the linear form f to be L^2. As presented in Section 1.5, the outputs considered in the context of thermal model $\mathcal{E}_{\mathrm{L}}(\mu)$ deviate from this standard case, attributed to the utilization of the Dirac functional in output computation used for pointwise evaluation. Nevertheless, we will see in the result section that the results are still in good agreement with the theory in the regular case. Additional insights into this issue are provided in Chapter 4.

3.1.3 Greedy generation of the reduced basis

The a posteriori error estimator introduced earlier provides an efficient criterion to select the desired dimension of the reduced space N, in the offline phase. Given a fixed tolerance $\varepsilon_{\mathrm{tol}}$, we can greedily select the greatest N such that the error bound is smaller than the tolerance. In this section, we describe an algorithm to generate the reduced basis.

For this algorithm, a large set of parameters $\Xi_{\mathrm{train}} \subset D^{\mu}$ is required. This set is called the *training set*, and is generated log-randomly. A first snapshot is computed for a given parameter $\mu_0 \in D^{\mu}$. To get the $N+1$-th snapshot to be inserted in the basis, we select the parameter $\mu^{\star}$ that maximizes the error bound $\Delta_N(\mu)$, for $\mu \in \Xi_{\mathrm{train}}$. This step is performed until a selected tolerance for the maximal error bound is reached. This procedure is denoted as *greedy* because at each step, the parameter that maximizes the error bound is selected. The steps of the procedure are presented in Algorithm 4.

As pointed out in [Coh+20], the quality of the reduced space V_N is highly dependent on the solution manifold $\mathcal{M}$, and its Kolmorogov n-width [MPT02; Ngu20]. In the spirit of [Pru+02],

Algorithm 4: Greedy algorithm to construct the reduced basis.

Input: $\mu_0 \in D^\mu$, $\Xi_{\text{train}} \subset D^\mu$ and $\varepsilon_{\text{tol}} > 0$.
$S \leftarrow [\mu_0]$;
while $\Delta_N^{\max} > \varepsilon_{tol}$ **do**
  $\mu^\star \leftarrow \underset{\mu \in \Xi_{\text{train}}}{\arg\max}\, \Delta_N(\mu)$ (and $\Delta_N^{\max} \leftarrow \underset{\mu \in \Xi_{\text{train}}}{\max}\, \Delta_N(\mu)$);
  $V_{N+1} \leftarrow \left\{\xi = u^{\text{fem}}(\mu^\star)\right\} \cup V_N$;
  Append $\mu^\star$ to S;
  $N \leftarrow N + 1$;
end
Output: Sample S, reduced basis V_N.

we choose a train set containing parameters that are log-uniformly distributed in the parameter space.

In the sequel of this section, we present the results of the reduced basis method applied to the thermal transfer model $\mathcal{E}_{\text{L}}$ described in Section 1.4.

3.1.4 Accuracy, performance and verifications of the reduced basis model

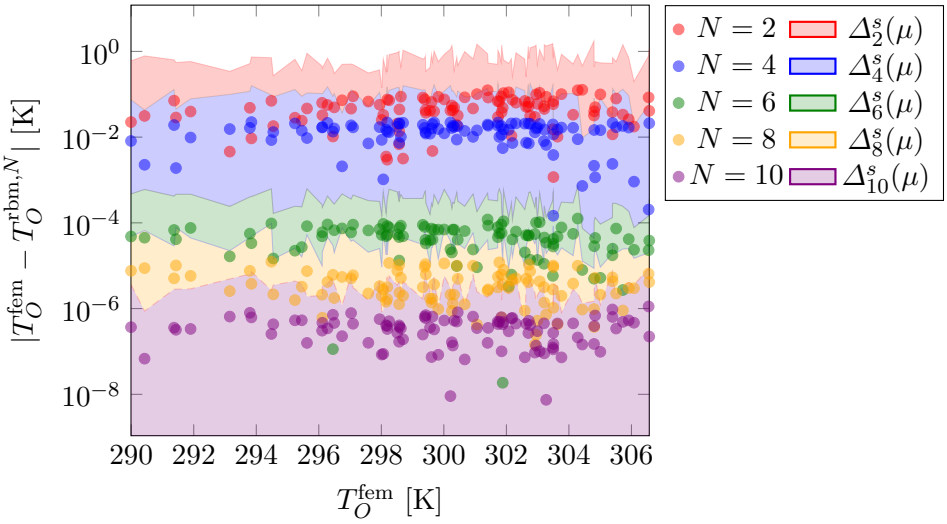
Convergence analysis We first compare the results of the reduced basis method with the output of the high fidelity FEM model. We generate a sample Ξ_{test} of 100 parameters in D^μ. For $\mu \in \Xi_{\text{test}}$, we compute on the one hand $T_O^{\text{fem}}(\mu)$, the value of the temperature at point O from the model $\mathcal{E}_{\text{L}}(\mu)$, and on the other hand $T_O^{\text{rbm},N}(\mu)$, the value of the temperature for the reduced basis model, with a basis of size N. In Figure 3.2(a), the value of the error $|T_O^{\text{fem}}(\mu) - T_O^{\text{rbm},N}(\mu)|$ is plotted for each $\mu \in \Xi_{\text{test}}$, for various reduced basis sizes N. Statistics on the error committed over the sample Ξ_{test} are displayed in Figure 3.2(b), as well as the effectivity $\eta_N(\mu)$ in Figure 3.2(c).

We observe that even for small values of N, the error on the output is tiny: an error of 10^{-4} is reached for $N = 6$. On the other hand, we find that the convergence rate on the output is approximately twice as fast as the convergence on the field, as predicted by the theoretical error estimate [Pru+02, Eq. (36)].

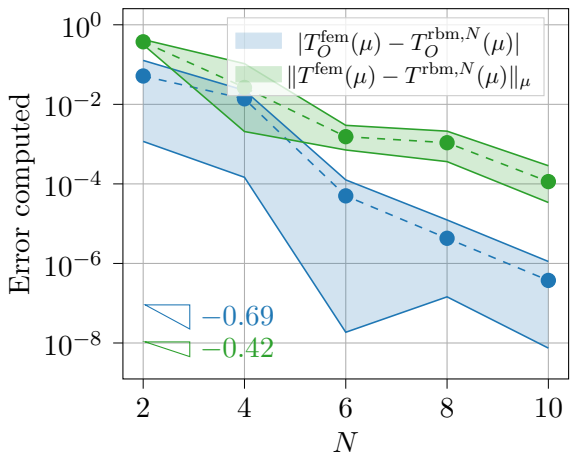
Note that the anticipated error behavior aligns with theory when the output functional maintains continuity. In this work, we deviate from the standard case, attributed to the utilization of the Dirac functional in output computation used for pointwise evaluation. Nevertheless, a similar behavior is observed, and additional insights into this phenomenon are provided in Chapter 4.

To infer the greatest possible error on the output, we compute the error bound $\Delta_N(\mu)$ for each $\mu \in \widetilde{\Xi}_{\text{test}}$, where $\widetilde{\Xi}_{\text{test}}$ is a very large sample of parameters in D^μ. Here we take a set whose size is 10^5. With $N = 10$, we find that the maximal error on the output is $\Delta_N^{\max} = 1.45 \cdot 10^{-4}$. Figure 3.3 shows the box plot of the error bound $\Delta_N(\mu)$ for $\mu \in \widetilde{\Xi}_{\text{test}}$, showing that as the size of the reduced basis increases, the error bound generally decreases.

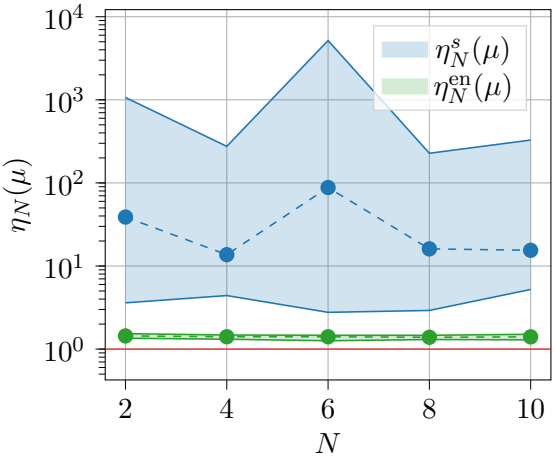
Execution time Table 3.1 displays a comparative analysis of execution times for solving the heat transfer problem. We first discuss the execution times for the high-fidelity solution, encompassing both $\mathbb{P}_1$ and $\mathbb{P}_2$ finite-element discretizations. The measured time, denoted as t_{exec}, includes assembling and solving the problem. In addition, we also evaluate the execution time of the online phase of our certified reliable reduced basis model. It corresponds to the time spent by the application to compute the solution of both primal and dual problems, as well as the error



(a) Error on the reduced solution $T^{\text{fem},N}(mu)$ for various reduced basis sizes N with the corresponding error bound $\Delta_N^s(\mu)$.



(b) Convergence of the errors on the field and the output on point O, for $\mu \in \Xi_{\text{test}}$, for various reduced basis sizes. The maximal, minimal, and mean values are represented.



(c) Stability of the effectivity $\eta_N^s(\mu) = \frac{\Delta_N^s(\mu)}{|T_O^{\text{fem}}(\mu) - T_O^{\text{rbm},N}(\mu)|}$ and $\eta_N^{\text{en}}(\mu) = \frac{\Delta_N^{\text{en}}(\mu)}{\|T^{\text{fem}}(\mu) - T^{\text{rbm},N}(\mu)\|_\mu}$ for $\mu \in \Xi_{\text{test}}$, for various reduced basis sizes. The full red line represents the theoretical lower bound of the effectivity, *i.e.* 1.

Figure 3.2: Comparison of the temperature between the full order model and the reduced basis model, tested over a sample $\Xi_{\text{test}} \subset D^\mu$ of 100 parameters.

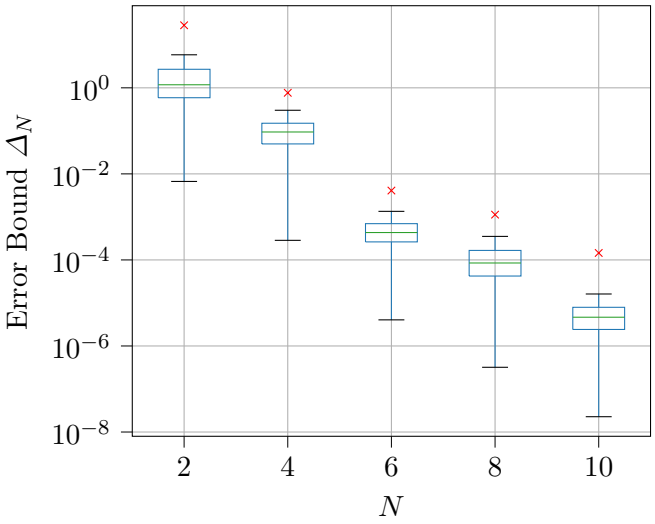


Figure 3.3: Box plot of the error bound $\Delta_N(\mu)$ for $\mu \in \widetilde{\Xi}_{\text{test}}$, and various reduced basis sizes N.

	Finite element resolution $T^{\text{fem}}(\mu)$			Reduced model $T^{\text{rbm},N}(\mu), \Delta_N(\mu)$
	$\mathbb{P}_1$	$\mathbb{P}_2$ (`np=1`)	$\mathbb{P}_2$ (`np=12`)	
Problem size	$\mathcal{N} = 207\,845$	$\mathcal{N} = 1\,580\,932$		$N = 10$
t_{exec}	5.534 s	62.432 s	10.76 s	2.88×10^{-4} s
speed-up	11.69	1	5.80	$\mathbf{2.17 \times 10^{5}}$

Table 3.1: Times of execution of the finite element model for both $\mathbb{P}_1$ and $\mathbb{P}_2$ discretization against the computation time of the reduced solution and error bound. The mesh `M3` is used for high fidelity simulations.

bound. This comparison highlights a significant reduction in the time required to assemble and solve the problem using our advanced reduced basis approach. Importantly, this efficiency does not compromise accuracy; the results from the reduced basis model are effectively sharp with respect to the high fidelity model. As anticipated in our earlier scalability analysis, we achieve remarkable computational gains with our model, reinforcing the benefits of our approach in both precision and performance.

The reduced bases constructed for the various outputs of interest are generated with the greedy algorithm (Algorithm 4), using a maximal tolerance $\varepsilon_{\text{tol}} = 1 \cdot 10^{-6}$ and a maximal size for the basis $N = 20$. In practice, the tolerance is reached for $N = 10$ to 12.

We presented the time required by the online stage to compute the solution of the reduced basis model, which we aim to minimize. In contrast, the offline stage, which is performed only once, can be computationally expensive. For a training set of size $N_{\text{train}} = 1000$ using the greedy algorithm, the offline stage takes approximately 3.5 hours to select the snapshots and construct the reduced basis. During the snapshot selection, the online solution was computed for each parameter in the training set, resulting in a total of over 11 000 computations, with an average time of 5.38×10^{-5} s per computation. The total time for these computations was 6.26×10^{-1} s. This highlights the importance of ensuring that the computation of the error bound is independent of the dimension of the HF problem.

3.2 Non-Intrusive Reduced Basis method

The Non-Intrusive Reduced Basis (NIRB) method is a variant of the Reduced Basis Method that achieves computational efficiency without requiring modifications to the underlying high-fidelity model. It constructs a low-dimensional approximation space using precomputed solutions ("snapshots") from the original model. The reduced basis is then combined with interpolation or regression techniques to approximate new solutions for different parameter values. This non-intrusive approach is particularly useful when the high-fidelity model is a black box or difficult to alter, enabling reduced-order modeling without compromising the integrity of the original solver.

While the RBM method introduced in the previous section proved highly efficient for the bioheat transfer problem, its applicability can be limited in certain contexts. For instance, RBM may face challenges with complex geometries, non-linear problems, or cases where access to the full-order model's internal structures is restricted, a common constraint in industrial applications. These limitations arise because RBM is an *intrusive* method, requiring direct interaction with the underlying model.

The *Non-Intrusive Reduced Basis* (NIRB) method addresses these constraints by decoupling the reduced-order model from the full-order model's internal details. This approach, introduced in [CM09b; CM09a] and extended to problems like CFD [CMP19] and parabolic systems [Gro22], uses a two-grid strategy to set up the MOR. Several error estimation techniques have also been proposed [GM23b; GM23b] to enhance its reliability. This approach still allows setting up an offline/online strategy, and is detailed in this second section. The implementation of the NIRB method within the Feel++ library is a novel contribution of this work. Developed in collaboration with Ali Elarif from Cemosis, this integration adapts the NIRB methodology to the modeling capabilities offered by Feel++, enabling its application to a broader class of problems, including those studied in this manuscript. Details of this contribution are discussed in this section and further elaborated in Section 7.3.

In this section, we first introduce the NIRB method and then present results obtained from various test cases. Lastly, we will explore the initial application of this method to a thermal model of the eye and discuss the outcomes.

We place ourselves in the same context as in Section 3.1, where we have a physical problem modeled by a system of PDE, with parameters involved. Starting from the PDE formulation, we can compute the associated variational formulation: for $\mu \in D^\mu$ given, find $u(\mu) \in V$ such that

$$\mathcal{F}(u(\mu), v; \mu) = 0, \quad \forall v \in V, \tag{3.37}$$

where V is a well-chosen functional space, and $\mathcal{F}\colon V \times V \times D^\mu \to \mathbb{R}$ is a functional with good properties so that the problem is well-posed; and focus on an *output of interest*:

$$s(\mu) = \ell(u(\mu); \mu). \tag{3.38}$$

We approximate the functional space by a finite element (FE) space V_h of dimension $\mathcal{N}_h$. The discretization of Equation (3.37) is then: for $\mu \in D^\mu$ given, find $u_h(\mu) \in V_h$ such that

$$\mathcal{F}(u_h, v; \mu) = 0, \quad \forall v \in V_h. \tag{3.39}$$

Similar to what was presented in the previous chapters, the algebraic equivalent to Equation (3.37) is: find $\underline{\boldsymbol{u}}_h(\mu) \in V_h \simeq \mathbb{R}^{\mathcal{N}_h}$ solution to the system

$$\underline{\underline{\boldsymbol{A}}}_h(\mu)\underline{\boldsymbol{u}}_h(\mu) = \underline{\boldsymbol{F}}_h(\mu). \tag{3.40}$$

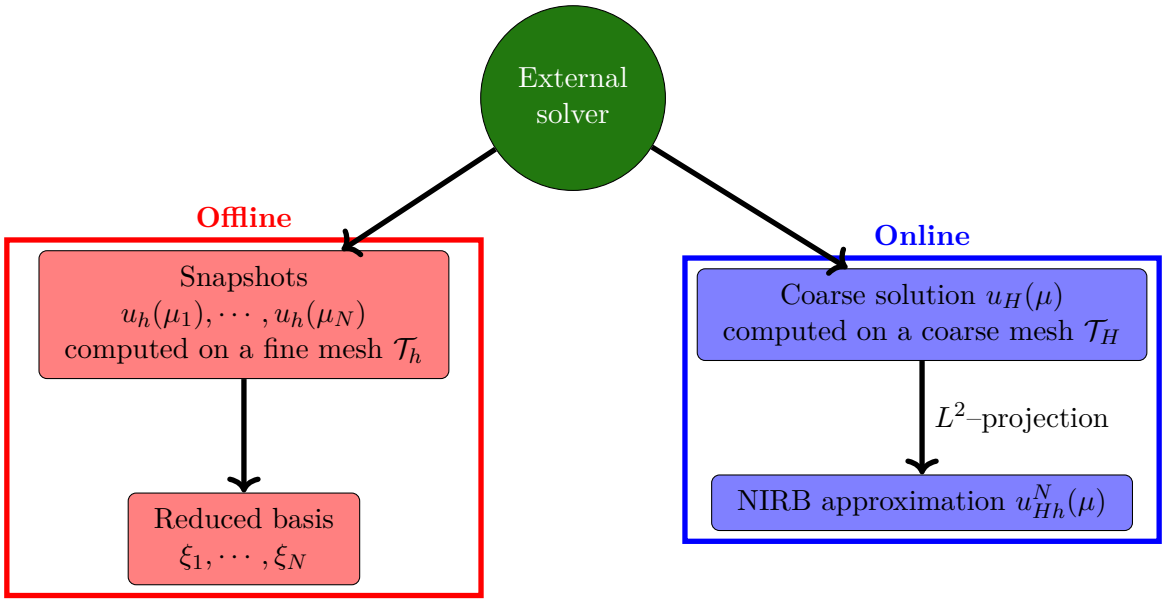


Figure 3.4: Two-grids strategy for the Non-Intrusive Reduced Basis method, adapted from [Gro22].

where $\underline{\underline{\boldsymbol{A}}}(\mu) \in \mathbb{R}^{\mathcal{N}_h \times \mathcal{N}_h}$ is the algebraic representation of $a(\cdot, \cdot; \mu)$ in V_h, and $\underline{\boldsymbol{F}}(\mu) \in \mathbb{R}^{\mathcal{N}_h}$ of $f(\cdot; \mu)$.

As presented in Section 3.1, the aim of RBM is to construct a problem of size N with $N \ll \mathcal{N}_h$ whose solution $\underline{\boldsymbol{u}}_N(\mu) \in V_N$ is a suitable approximation to the FE solution $u(\mu)$:

$$\underline{\underline{\boldsymbol{A}}}_N(\mu)\underline{\boldsymbol{u}}_N(\mu) = \underline{\boldsymbol{F}}_N(\mu), \tag{3.41}$$

with $\underline{\underline{\boldsymbol{A}}}_N(\mu) \in \mathbb{R}^{N \times N}$ and $\underline{\boldsymbol{F}}_N(\mu) \in \mathbb{R}^N$.

This method requires a linear decomposition such as $\underline{\underline{\boldsymbol{A}}}_h(\mu) = \sum_{q=1}^{Q_A} \beta_A^q(\mu)\underline{\underline{A}}_h^q$ [Pru+02] to enable fast online computation. For complex models, such a decomposition can be difficult to obtain, if not impossible. Moreover, in the context of industrial applications, the solver code may not be available, and therefore the intrusive method is not applicable.

In the following, we focus on a non-intrusive strategy, involving a multi-grid method, in which the functional space V is approximated by two different spaces of various refinement, V_H and V_h, where $H \gg h$ is the characteristic sizes used for the discretization. The two meshes used to discretize the problem are named $\mathcal{T}_H$ and $\mathcal{T}_h$ respectively. We denote by $\mathcal{N}_H$ (resp. $\mathcal{N}_h$) the dimension of V_H, (resp. V_h). We display in Figure 3.4 a schematic of the two-grids strategy for the Non-Intrusive Reduced Basis method.

Notations Before delving into the details of the method, we introduce some notations that will be used in the following. A notation in normal font (*e.g.* u) corresponds to a function in a functional space, while a notation in bold font (*e.g.* $\underline{\boldsymbol{u}}$) corresponds to the algebraic equivalent. A quantity noted with a superscript $\mathcal{N}$ means that such element is associated to the FE space, while a subscript N means that the element is associated to the reduced space.

Table 3.2 gathers all the notations that are used in the present section. Note that to avoid overloading the notation, the FE solution in the coarse mesh is written $u_H^{\mathcal{N}}$ instead of $u_H^{\mathcal{N}_H}$.

The scalar product $\langle \cdot, \cdot \rangle_{L^2}$ is the L^2-scalar product over the functional space V. In practice, we will use the L^2-scalar product over the FE space V_h, or V_H. We also define the H^1-scalar product over functional space, $\langle \cdot, \cdot \rangle_{H^1}$. Using the notation of Table 3.2, we have for $u, v \in V_h$:

$$\langle u, v \rangle_{L^2} = u^T \cdot M_{L^2} \cdot v \qquad \text{and} \qquad \langle u, v \rangle_{H^1} = u^T \cdot M_{H^1} \cdot v. \tag{3.42}$$

Notation	Description
V_h	FE functional space on the fine mesh $\mathcal{T}_h$
V_H	FE functional space on the coarse mesh $\mathcal{T}_H$
V_h^N	reduced functional space of the fine functional space V_h
V_H^N	reduced functional space of the coarse functional space V_H
$\mathcal{N}_H$	Dimension of the coarse FE space V_h
$\mathcal{N}_h$	Dimension of the fine FE space V_H
$u_h^{\mathcal{N}}$	FE solution on the fine mesh
u_h^N	Projection of $u_h^{\mathcal{N}}$ on the reduced function space V_h^N
$u_H^{\mathcal{N}}$	FE solution on the coarse mesh
$u_{H\to h}^{\mathcal{N}}$	FE solution on the coarse mesh, interpolated on the fine mesh
u_{Hh}^N	NIRB solution
$\underline{\underline{\boldsymbol{Z}}}_N$	reduced basis, stored in columns: $\underline{\underline{\boldsymbol{Z}}}_N \in \mathbb{R}^{\mathcal{N}_h \times N}$
$(\xi_h^i)_{i=1\dots N}$	basis functions, $\xi_h^i \in \mathbb{R}^{\mathcal{N}_h}$, and $\underline{\underline{\boldsymbol{Z}}}_N$`[:, i]` $= \xi_i$
$(\xi_H^i)_{i=1\dots N}$	coarse basis functions, $\xi_H^i \in \mathbb{R}^{\mathcal{N}_H}$
$\underline{\underline{\boldsymbol{M}}}_{L^2}$	matrix of mass (L^2 scalar product): $M_{L^2} \in \mathbb{R}^{\mathcal{N}_h \times \mathcal{N}_h}$
$\underline{\underline{\boldsymbol{M}}}_{H^1}$	matrix of stiffness (H^1 scalar product): $M_{H^1} \in \mathbb{R}^{\mathcal{N}_h \times \mathcal{N}_h}$
$\underline{\underline{\boldsymbol{M}}}_{Hh}$	interpolation matrix from coarse mesh to fine one: $M_{Hh} \in \mathbb{R}^{\mathcal{N}_h \times \mathcal{N}_H}$

Table 3.2: Notations for the NIRB framework.

3.2.1 Theoretical framework

The RBM has been presented in Section 3.1. We simply recall here the notation used in this part.

The reduced basis is composed of N basis functions $(\xi_h^i)_{i=1\dots N}$, which are L^2-orthonormalized. We set the matrix $\underline{\underline{\boldsymbol{Z}}}_N \in \mathbb{R}^{\mathcal{N}_h \times N}$ such that $\underline{\underline{\boldsymbol{Z}}}_N$`[:,i]` $= \xi_h^i$.

Projection step

In the RBM, during the online stage, we solve a low-dimensionnal system to get $\underline{\boldsymbol{u}}_N(\mu) \in \mathbb{R}^N$, and then construct the solution on the space V_h by means of the following formula:

$$u_h^N(\mu) = \sum_{i=1}^{N} \underline{\boldsymbol{u}}_{Ni}(\mu)\xi_h^i. \tag{3.43}$$

On the other hand, the NIRB method proposes constructing the coefficient $(u_i^N)_{i=1\dots N}$ by using a direct projection of the finite element approximation $u_h^{\mathcal{N}}$ in the space V_h^N. This projection ensures that the reduced coefficients $(u_i^N)_{i=1\dots N}$ accurately capture the essential features of the finite element approximation $u_h^{\mathcal{N}}$ within the reduced space V_h^N, preserving the fidelity of the reduced model. Denote by $\Pi u_h^{\mathcal{N}}$ the L^2-projection of FE approximation of $u_h^{\mathcal{N}}$ in the space V_N.

Proposition 3.2.1. *The following relation holds:*

$$u_h^N(\mu) = \Pi u_h^{\mathcal{N}}(\mu) = \sum_{i=1}^{N} \alpha_i^{N,h}(\mu)\xi_h^i, \tag{3.44}$$

with $\alpha_i^{N,h}(\mu)$ *defined by:*

$$\alpha_i^{N,h}(\mu) = \left\langle u_h^{\mathcal{N}}(\mu), \xi_h^i \right\rangle_{L^2}. \tag{3.45}$$

Proof. In this proof, $\mu \in D^\mu$ is fixed. The quantities $u_h^{\mathcal{N}}(\mu), \Pi u(\mu)$ are introduced in the following depending on μ, but to simplify the notation, this dependence will not appear explicitly.

We look for the L^2-projection of $u_h^{\mathcal{N}}$, $\Pi u \in V_h^N$, such that

$$\langle \Pi u, v \rangle_{L^2} = \left\langle u_h^{\mathcal{N}}, v \right\rangle_{L^2} \qquad \forall v \in V_N. \tag{3.46a}$$

This is true for all $v \in V_h^N$, in particular for the basis functions ξ_h^i, for $i \in [\![1..N]\!]$:

$$\forall i \in [\![1..N]\!], \left\langle \Pi u, \xi_h^i \right\rangle_{L^2} = \left\langle u_h^{\mathcal{N}}, \xi_h^i \right\rangle_{L^2}. \tag{3.46b}$$

On the one hand, we have:

$$\Pi u = \sum_{j=1}^{N} \Pi u_j \xi_j^h, \tag{3.46c}$$

and on the other hand, as the basis, $(\xi_h^i)_i$ is L^2-orthonormalized, $\langle \xi_i, \xi_j \rangle_{L^2} = \delta_{ij}$, hence

$$\left\langle \Pi u, \xi_h^i \right\rangle_{L^2} = \Pi u_i = \left\langle u_h^{\mathcal{N}}, \xi_h^i \right\rangle_{L^2}. \tag{3.46d}$$

Finally, using Equations (3.46c) and (3.46d), we get:

$$\Pi u = \sum_{i=1}^{N} \left\langle u_h^{\mathcal{N}}, \xi_h^i \right\rangle_{L^2} \xi_j^h. \tag{3.46e}$$

■

The main idea of the NIRB [CM09b; CM09a] method relies on a second triangulation $\{T_H\}_H$, whose mesh size H is much larger than the mesh size h of the fine grid $\{T_h\}_h$: $H \gg h$. This second triangulation is called *coarse mesh* and the corresponding finite element space is denoted by V_H, of finite dimension $\mathcal{N}_H \ll \mathcal{N}_h$. For $\mu \in D^\mu$, we denote by $u_H^{\mathcal{N}}(\mu) \in V_H$ the solution of the finite element approximation of Equation (3.37) on the coarse mesh T_H. As the time of computation on the coarse mesh is much smaller than on the fine mesh, instead of using $u_h^{\mathcal{N}}(\mu)$ to construct the interpolation coefficients in Equation (3.44), we use the following approximate solution:

$$u_{Hh}^N(\mu) = \sum_{i=1}^{N} \alpha_i^{N,H}(\mu) \xi_h^i, \tag{3.47}$$

with $\alpha_i^{N,H}(\mu)$ defined by:

$$\alpha_i^{N,H}(\mu) = \left\langle u_{H\to h}^{\mathcal{N}}(\mu), \xi_h^i \right\rangle_{L^2}, \tag{3.48}$$

where $u_{H\to h}^{\mathcal{N}}(\mu)$ is the finite element approximation of $u_H^{\mathcal{N}}(\mu)$, interpolated on the fine mesh T_h.

In the following, we describe how the NIRB basis functions are constructed, and how to ensure their orthonormalization. Recall that the NIRB method consists in building the basis functions by using preselected finite element solutions from the fine mesh, the *snapshots*. In this work, we use two kinds of preselection: a random selection and a Greedy algorithm, that will be implemented in the following sections.

3.2.2 Construction of the basis functions

Random selection This kind of selection is widely used in the reduced order method [Gro22; CMP19] due to the simplicity of implementation. It consists in taking a random sample of parameters $\mu_i \in D^\mu$, and compute the snapshots associated to these parameters. The basis functions are built using a Proper Orthogonal Decomposition (POD) method, which is a statistical technique used to reduce the dimensionality of a system by identifying patterns in the data. The steps involved in constructing the basis functions using POD are as follows:

1. A random choice of a sampling of N_{train} parameters $\{\mu_1, \cdots, \mu_{N_{\text{train}}}\} \subset D^\mu$,
2. Solving the equation (3.40) on the fine grid for each chosen parameters to get the N_{train} initial snapshots, $u_h^i := u_h^{\mathcal{N}}(\mu_i), 1 \leqslant i \leqslant N_{\text{train}}$,
3. Computation of the correlation matrix $\underline{\underline{\boldsymbol{C}}} \in \mathbb{R}^{N_{\text{train}} \times N_{\text{train}}}$ by:
$$\underline{\underline{\boldsymbol{C}}}_{i,j} = \left\langle u_h^i, u_h^j \right\rangle_{L^2} \qquad 1 \leqslant i, j \leqslant N_{\text{train}}. \tag{3.49}$$
4. Solving the eigenvalue problem:
$$\underline{\underline{\boldsymbol{C}}}\,\underline{\boldsymbol{v}} = \lambda \underline{\boldsymbol{v}}. \tag{3.50}$$
5. The POD basis functions are given by:
$$\xi_i^h = \frac{1}{\sqrt{\lambda_i}} \sum_{j=1}^{N_{\text{train}}} \underline{\boldsymbol{v}}^i[j] \cdot u_h^j, \qquad 1 \leqslant i \leqslant N_{\text{train}}, \tag{3.51}$$
where $(\lambda_i, \underline{\boldsymbol{v}}^i) \in \mathbb{R} \times \mathbb{R}^{N_{\text{train}}}$ is the i-th pair of eigenvalue and eigenvector. As $\underline{\underline{\boldsymbol{C}}}$ is symmetric, the eigenvectors are orthogonal, and the eigenvalues are real, and we set $\lambda_1 \leqslant \cdots \leqslant \lambda_{N_{\text{train}}}$.

The reduced space V_h^N is spanned by the POD basis functions $\{\xi_h^i\}_{i=1}^{N_{\text{train}}}$, and has the dimension N_{train}.

Proposition 3.2.2. *The POD basis functions $\{\xi_h^i\}_{i=1}^{N_{train}}$ are orthonormal in L^2. Namely:*
$$\left\langle \xi_h^i, \xi_h^j \right\rangle_{L^2} = \delta_{i,j}, \tag{3.52}$$
where $\delta_{i,j}$ is the Kronecker symbol.

Proof. Using Equation (3.51), we obtain:
$$\left\langle \xi_h^i, \xi_h^j \right\rangle_{L_2} = \frac{1}{\sqrt{\lambda_i \lambda_j}} \left\langle \sum_l \underline{\boldsymbol{v}}_i[l] u_h^l, \sum_{l'} \underline{\boldsymbol{v}}_j[l'] u_h^{l'} \right\rangle_{L^2} = \frac{1}{\sqrt{\lambda_i \lambda_j}} \sum_l \underline{\boldsymbol{v}}_i[l] \underbrace{\sum_{l'} \underline{\underline{\boldsymbol{C}}}_{l,l'} \underline{\boldsymbol{v}}_j[l']}_{\lambda_j \underline{\boldsymbol{v}}_j[l]}. \tag{3.53}$$
As the matrix $\underline{\underline{\boldsymbol{C}}}$ is positive definite, the eigenvectors are orthogonal: $\forall i, j \in [\![1, N]\!], \left\langle \underline{\boldsymbol{v}}_i, \underline{\boldsymbol{v}}_j \right\rangle_{L^2} = \delta_{i,j}$. Thus: $\left\langle \xi_h^i, \xi_h^j \right\rangle_{L_2} = \frac{\lambda_j}{\sqrt{\lambda_i \lambda_j}} \sum_l \underline{\boldsymbol{v}}_i[l] \underline{\boldsymbol{v}}_j[l] = \frac{\lambda_j}{\sqrt{\lambda_i \lambda_j}} \left\langle \underline{\boldsymbol{v}}_i, \underline{\boldsymbol{v}}_j \right\rangle_{L^2} = \delta_{i,j}$. ∎

In Item 5, all the eigenvectors are used to construct the POD basis functions. From a heuristic viewpoint, the eigenvectors with the fewer energy may be discarded to reduce the size of the basis because of their weak impact on the space generated. To this purpose we introduce the *Relative Information Content*.

Definition 3.2.3. Let $\{\lambda_1, \cdots, \lambda_{N_{\text{train}}}\}$ be the eigenvalues of the correlation matrix $\underline{\underline{\boldsymbol{C}}}$. The *Relative Information Content* (RIC) is defined as the ratio between the N first eigenvalues and their total sum:

$$\text{RIC}(N) := \frac{\sum_{j=1}^{N} \lambda_j}{\sum_{j=1}^{N_{\text{train}}} \lambda_j}. \tag{3.54}$$

Instead of computing the POD modes for all the eigenvalues, we stop when the RIC reach a threshold $1-\varepsilon$, for a given tolerance $\varepsilon > 0$. This will result in a reduced basis of size $N \leqslant N_{\text{train}}$.

This method is implemented in Algorithms 5 and 6.

Algorithm 5: Offline POD computation of basis function from snapshots.

Data: $\{u_h^1, \cdots, u_h^{N_{\text{train}}}\} \in V_h$ fine snapshots.

Compute correlation matrix $\underline{\underline{\boldsymbol{C}}} \in \mathbb{R}^{N_{\text{train}} \times N_{\text{train}}}$:
forall $1 \leqslant i, j \leqslant N_{train}$ **do**
  $\underline{\underline{\boldsymbol{C}}}_{i,j} = \left\langle u_h^i, u_h^j \right\rangle_{L^2}$ where $u_h^i = u_h^{\mathcal{N}}(\mu_i)$;
end

Solve eigenvalue problem: $\underline{\underline{\boldsymbol{C}}}\,\underline{\boldsymbol{v}} = \lambda \underline{\boldsymbol{v}}$, to get eigenvalues $\lambda_1 \leqslant \cdots \leqslant \lambda_{N_{\text{train}}}$ and eigenvectors $\underline{\boldsymbol{v}}_1, \cdots, \underline{\boldsymbol{v}}_{N_{\text{train}}} \in \mathbb{R}^{N_{\text{train}}}$;

Compute basis functions:
$N \leftarrow 0$;
forall $1 \leqslant i \leqslant N_{train}$ **do**
  $\xi_h^i = \dfrac{1}{\sqrt{\lambda_i}} \displaystyle\sum_{j=1}^{N_{\text{train}}} \underline{\boldsymbol{v}}^i[j] \times u_h^j$;
  $N \leftarrow N + 1$;
  Stop if RIC(N) is close enough to 1, for a given tolerance;
  $V_h^N = V_h^{N-1} \cup \{\xi_h^i\}$;
end
Result: ξ_h^i to save on Disk, $1 \leqslant i \leqslant N$.

Algorithm 6: Offline computation: random selection.

Data: Choose $S_\mu = \{\mu_k\}, 1 \leqslant k \leqslant N_{\text{train}}$ with $\mu_k \in D^\mu$.
for $k \in [\![1, N_{train}]\!]$ **do**
  $u_h^k := u_h^{\mathcal{N}}(\mu_k)$;
end
$V_h^N := \text{POD}\left(\left\{u_h^1, \ldots, u_h^N\right\}\right)$ (see Algorithm 5);
Result: Reduced basis $\{\xi_h^i\}_{1 \leqslant i \leqslant N}$ of V_h^N.

Greedy selection The arbitrary choice of sampling may result in a basis that is not well adapted. To compensate for this, we introduce a greedy procedure to select those parameters in a given space Ξ_{train}. In the method introduced in the previous section, the sample Ξ_{train} has the size of the expected reduced basis. In the greedy method, we start with a bigger sample, and we select the parameters that maximize a criterion.

The idea of this procedure is to select the parameters on this bigger sample, while minimizing the call to the fine solver. In each iteration, we select one parameter as the one maximizing

an estimator, that can be computed efficiently, without calling the solver on the fine grid. If N describes the size of the reduced space already constructed, we take $M < N$ (for instance, $M = N - p$ or $M = N/2$). For a given parameter, the estimator is obtained as

$$\left| s\left(u_H^N(\mu)\right) - s\left(u_H^M(\mu)\right) \right| \quad \text{or} \quad \left\| u_{Hh}^N(\mu) - u_{Hh}^M(\mu) \right\|_{L^2}, \tag{3.55}$$

where s is an output function. Recall that $u_{Hh}^N(\mu)$ is the online NIRB approximation computed in the offline stage.

In the implementation, we use the field error as the estimator of error. This choice is motivated by the fact that this quantity can capture the general behavior of the solution, while the output function would result in a basis more specific to the output-of-interest considered.

The quantity $u_H^N(\mu)$ is a reduced solution computed on the coarse mesh, so its computation is inexpensive compared to a fine FE resolution:

$$u_H^N\texttt{[i]} = \left\langle \xi_h^i, u_H^{\mathcal{N}} \right\rangle_{L^2} \text{ for } i \in [\![1, N]\!], \tag{3.56}$$

where ξ_h^i is the i-th coarse snapshot introduced in the basis. We have $\underline{\underline{\boldsymbol{Z}}}_N^H = \left[\xi_1^H, \ldots, \xi_N^H\right]$. In order to correctly project $u_H^{\mathcal{N}}$ into the reduced spaces, the matrix $\underline{\underline{\boldsymbol{Z}}}_N$ must be L^2-orthonormalized (according to the scalar product of the coarse function space V_H).

Another estimator for the algorithm is the difference between the NIRB approximation for two different sizes of reduced spaces:

$$\|u_{Hh}^N(\mu) - u_{Hh}^M(\mu)\|_{L^2}. \tag{3.57}$$

Then we construct the basis functions by following steps 3, 4, and 5 of the previous section, using the POD described earlier, in Algorithm 5. The whole process is summarized in Algorithm 7.

Algorithm 7: Offline computation: Greedy selection of the snapshots. Only N snapshots are computed on the fine grid.

Data: $\Xi_{\text{train}} \subset D^\mu$, N_{init}.
$S := \{\mu_1, \cdots \mu_{N_{\text{init}}}\} \subset \Xi_{\text{train}}$, randomly
Construct $V_h^{N_{\text{init}}} = \text{POD}(\{u_h^{\mathcal{N}}(\mu_1), \cdots, u_h^{\mathcal{N}}(\mu_{N_{\text{init}}})\})$;
$N \leftarrow N_{\text{init}}$;
while $\Delta > \varepsilon$ **do**
  $\mu_\star = \underset{\mu \in \Xi_{\text{train}}}{\arg\max} \left\| u_{Hh}^N(\mu) - u_{Hh}^M(\mu) \right\|_{L^2}$;
  $S \leftarrow S \cup \{\mu_\star\}$;
  $X_{N+1} := \text{POD}\left(V_N \cup \left\{u_h^{\mathcal{N}}(\mu_\star)\right\}\right)$;
end

3.2.3 Rectification post-process

The rectification post-process is a step that can be added to recover the accuracy of the approximation given by the optimal coefficients $\alpha_i^{N,H}(\mu)$, without losing computational time. Such a process has been introduced in [CM09b; CM09a], and can be easily added in addition to the classical NIRB method. It was observed from [CMP19] that this process increases the precision of the NIRB approximate solution.

The rectification post-process allows ensuring that for a given set of parameters $S_\mu = \{\mu_k\}, 1 \leqslant k \leqslant N$ used in the construction of basis function, the NIRB method returns *exactly* $u_h^{\mathcal{N}}(\mu_k)$ which is already pre-computed to construct the reduced basis.

We introduce the matrices $\underline{\underline{\boldsymbol{C}}}^h, \underline{\underline{\boldsymbol{C}}}^H \in \mathbb{R}^{N\times N}$:

$$\forall i = 1 \dots N \quad \forall \mu_k \in \mathcal{D}^\mu, \quad \underline{\underline{\boldsymbol{C}}}^H_{k,i} = \alpha_i^{N,H}(\mu_k) \quad \text{and} \quad \underline{\underline{\boldsymbol{C}}}^h_{k,i} = \alpha_i^{N,h}(\mu_k), \tag{3.58}$$

where $\alpha_i^{N,H}$ and $\alpha_i^{N,h}$ are defined respectively in Equation (3.48) and Equation (3.45) as the coefficients of $u^N_{Hh}(\mu)$ and $u^N_h(\mu)$ in the basis $\{\xi^i_h\}_{1\leqslant i\leqslant N}$.

The rectification post-process aims to find the matrix $\underline{\underline{\boldsymbol{\mathcal{R}}}} \in \mathbb{R}^{N\times N}$ which minimizes:

$$\left\| \sum_{i=1}^N \alpha_i^{N,h}(\mu_k)\xi^i_h - \sum_{i=1}^N \left(\sum_{j=1}^N \underline{\underline{\boldsymbol{\mathcal{R}}}}_{i,j} \alpha_j^{N,H}(\mu_k) \right) \xi^i_h \right\|_{L^2} \quad \forall k = 1 \dots N. \tag{3.59}$$

Using the L^2 orthonormalization property of the reduced basis, this is equivalent to minimizing the quantity [CMP19; Gro22]:

$$\left\| \sum_{i=1}^N \left(\alpha_i^{N,h}(\mu_k) - \sum_{j=1}^N \underline{\underline{\boldsymbol{\mathcal{R}}}}_{i,j} \alpha_j^{N,H}(\mu_k) \right) \right\|_2^2 \quad \forall k = 1 \dots N.$$

Namely,

$$\left\| \underline{\underline{\boldsymbol{C}}}^H \underline{\underline{\boldsymbol{\mathcal{R}}}} - \underline{\underline{\boldsymbol{C}}}^h \right\|_2^2.$$

Adding a Tikhonov regularization in order to have a more regular problem to solve, we minimize the following quantity:

$$\left\| \underline{\underline{\boldsymbol{C}}}^H \underline{\underline{\boldsymbol{\mathcal{R}}}} - \underline{\underline{\boldsymbol{C}}}^h \right\|_{L^2} + \lambda \left\| \underline{\underline{\boldsymbol{\mathcal{R}}}} \right\|_{L^2},$$

which gives an explicit expression of $\underline{\underline{\boldsymbol{\mathcal{R}}}}$ [Gro22]:

$$\underline{\underline{\boldsymbol{\mathcal{R}}}} = \left[(\underline{\underline{\boldsymbol{C}}}^H)^T \underline{\underline{\boldsymbol{C}}}^H + \lambda \underline{\underline{I}}_N \right]^{-1} (\underline{\underline{\boldsymbol{C}}}^H)^T \underline{\underline{\boldsymbol{C}}}^h. \tag{3.60}$$

So that to compute the approximate NIRB solution for a given parameter value μ, we replace the coefficient $\alpha_i^{N,h}$ by:

$$\alpha_i^{N,h,\mathrm{rec}}(\mu) = \sum_{j=1}^N \underline{\underline{\boldsymbol{\mathcal{R}}}}_{i,j}(\mu) \alpha_j^{N,H}(\mu),$$

and then

$$u_h^N(\mu) = \sum_{i=1}^N \alpha_i^{N,h,\mathrm{rec}}(\mu)\xi^i_h. \tag{3.61}$$

Remark 3.2.4. Numerically, we have $\alpha^{N,h,\mathrm{rec}}(\mu) = \underline{\underline{\boldsymbol{\mathcal{R}}}}\alpha^{N,H}(\mu)$. As the rectification matrix does not depend on the parameter μ, it can be computed once and for all, which is an advantage from the perspective of the computational cost.

If the rectification post-process is used, the offline procedure described in Algorithm 5 is completed by the computation and the storage of the matrices $\underline{\underline{\boldsymbol{C}}}^h$ and $\underline{\underline{\boldsymbol{C}}}^H$, as presented in Algorithm 8.

3.2.4 Online reconstruction

Once the basis functions have been generated, and possibly the rectification matrix, the online stage consists in computing the NIRB solution obtained using Equation (3.47). The steps are described in Algorithm 9.

Algorithm 8: Offline POD computation of basis function from snapshots, with the rectification post-process.

Data: $\{u_h^1, \cdots, u_h^{N_{\text{train}}}\} \in V_h$ fine snapshots,
$\{u_H^1, \cdots, u_H^{N_{\text{train}}}\} \in V_H$ corresponding coarse snapshots.

Compute the POD reduced basis functions as per Algorithm 5.
Get V_H^N in coarse mesh with Greedy or POD random;
forall $1 \leqslant i, j \leqslant N$ **do**

$$\underline{\underline{\boldsymbol{C}}}^h_{i,j} = \left\langle u_h^i, u_h^j \right\rangle_{L^2(V_h^N)} \quad u_h^k \in V_h^N;$$

$$\underline{\underline{\boldsymbol{C}}}^H_{i,j} = \left\langle u_H^i, u_H^j \right\rangle_{L^2(V_H^N)} \quad u_H^k \in V_H^N;$$

end
Result: Matrix $\underline{\underline{\boldsymbol{C}}}^h$ and $\underline{\underline{\boldsymbol{C}}}^H$ to save on Disk.

3.2.5 Test cases

In this section, we present the various tests that have been used to test the NIRB method.

2D case We consider a simple case of heat transfer, with regions of various thermal conductivities, that is adapted from [HRS16]. We study an adimensionalized heat transfer problem. The geometry is described in Figure 3.5, and is composed of four regions Ω_i with different thermal conductivities k_i. The boundary $\partial\Omega$ is split into two parts: Γ_N and Γ_R, where the Neumann and Robin boundary conditions are applied, respectively.

The square domain is subdivided into $n \times n$ parts, each one with a different thermal property. The temperature u is governed by the elliptic equation:

$$-k_i \Delta u = 0 \quad \text{in} \quad \Omega_i, \tag{3.62a}$$

with the boundary conditions:

$$-k_i \frac{\partial u_i}{\partial n} = -1 \quad \text{on } \Gamma_N, \tag{3.62b}$$

$$-k_i \frac{\partial u}{\partial n} = \text{Bi}\, u \quad \text{on } \Gamma_R. \tag{3.62c}$$

The domain described can also contain a different number of subdomains. A Python script can generate the configuration files for Feel++, see the script `case_generator_cube`[1].

Thermal fin model We first present a simple case of an adimensionalized heat problem of a thermal fin, adapted from [Pru+02; Kho+03]. We consider the geometry given in Figure 3.6, composed of 5 subdomains Ω_i, with various thermal conductivities k_i. The boundary $\partial\Omega$ is split into various parts: Γ_{root} at the bottom of the domain and Γ^i_{ext} for the external boundaries of the domain Ω_i. We also consider the internal boundaries Γ^i_{int} between the domains Ω_i and Ω_0.

The adimensionalized temperature u is governed by the elliptic PDE:

$$-k_i \Delta u = 0 \quad \text{in} \quad \Omega_i. \tag{3.63a}$$

Furthermore, we add the following boundary conditions:

[1] https://github.com/feelpp/feelpp/blob/c560758/toolboxes/generator/cases/heat_nirb/case_generator_cube.adoc

Algorithm 9: Online computations.

Data: $\mu \in D^{\mu}$ and N number of basis function required.
Load N basis functions from Disk,
if *With rectification post-process* **then**
  Load matrix $\underline{\underline{\boldsymbol{C}}}^h$ and $\underline{\underline{\boldsymbol{C}}}^H$ from Disk;
  $\underline{\underline{\boldsymbol{C}}}^h_N := \underline{\underline{\boldsymbol{C}}}^h[:N,:N]$;
  $\underline{\underline{\boldsymbol{C}}}^H_N := \underline{\underline{\boldsymbol{C}}}^H[:N,:N]$;
  $\underline{\underline{\boldsymbol{\mathcal{R}}}} := \left[(\underline{\underline{\boldsymbol{C}}}^H_N)^T \underline{\underline{\boldsymbol{C}}}^H_N + \lambda \underline{\underline{I}}_N\right]^{-1} (\underline{\underline{\boldsymbol{C}}}^H_N)^T \underline{\underline{\boldsymbol{C}}}^h_N$;
end
Compute the *coarse* FE solution $u^{\mathcal{N}}_H(\mu) \in V^{\mathcal{N}}_H$;
Interpolate coarse solution into the fine mesh: $u^{\mathcal{N}}_{H\to h}(\mu) \in V^{\mathcal{N}}_h$;
forall $1 \leqslant i \leqslant N$ **do**
  $\alpha_i^{N,H}(\mu) = \left\langle u^{\mathcal{N}}_{H\to h}(\mu), \xi^i_h \right\rangle_{L^2}$;
end
if *With rectification post-process* **then**
  $\alpha^{N,H}(\mu) \leftarrow \underline{\underline{\boldsymbol{\mathcal{R}}}} \times \alpha^{N,H}(\mu)$;
end
Result: NIRB solution $u^N_{Hh}(\mu) = \sum_{i=1}^{N} \alpha_i^{N,H}(\mu)\xi^i_h$, and the output $s\left(u^N_{Hh}(\mu)\right)$.

$$-\left(\nabla u_0 \cdot n_i\right) = -1 \qquad \text{on } \Gamma_{\text{root}}, \tag{3.63b}$$

$$-k_i\left(\nabla u_i \cdot n_i\right) = \text{Bi}\, u_i \qquad \text{on } \Gamma_i^{\text{ext}} \text{ for } i \in \{0,1,2,3,4\}, \tag{3.63c}$$

$$u_0 = u_i \qquad \text{on } \Gamma_i^{\text{int}} \text{ for } i \in \{1,2,3,4\}, \tag{3.63d}$$

$$-\left(\nabla u_0 \cdot n_i\right) = -k_i(\nabla u_i \cdot n_i) \qquad \text{on } \Gamma_i^{\text{int}}. \tag{3.63e}$$

The parameters involved in this problem are the conductivities $k_i \in [0.1, 10]$ and the Biot number $\text{Bi} \in [0.01, 1]$. Geometrical parameters can also be involved, such as the length of the fin, the number of regions, or the number of dimensions of the model, even though they are not considered in this study. A Python script can generate configuration files for Feel++, see `case_generator_thermal_fin.py`[2].

The present model has been present with a 2D geometry, but can be extended to a 3D geometry, with a central parallelepiped post and parallelepiped fins around it. The 3D geometry is presented in Figure 3.13(a), among results.

Thermal bridge benchmark ISO 10211:2007 [Sta07] sets out the specifications for a three-dimensional and a two-dimensional geometrical model of a thermal bridge for the numerical calculation of: *(i)* Heat flows, in order to assess the overall heat loss from a building or part of it; *(ii)* minimum surface temperatures, in order to assess the risk of surface condensation.

These specifications include the geometrical boundaries and subdivisions of the model, the thermal boundary conditions, and the thermal values and relationships to be used. ISO 10211:2007 is based upon the following assumptions: *(i)* all physical properties are independent of temperature, *(ii)* there are no heat sources within the building element.

[2] https://github.com/feelpp/feelpp/blob/9b42914/toolboxes/generator/cases/thermal_fin/case_generator_thermal_fin.adoc

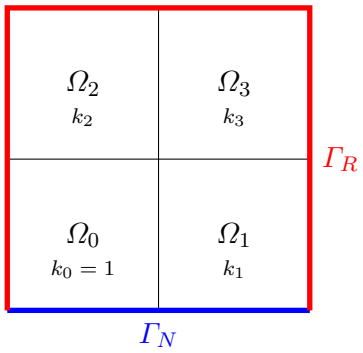


Figure 3.5: Square domain of the heat transfer problem.

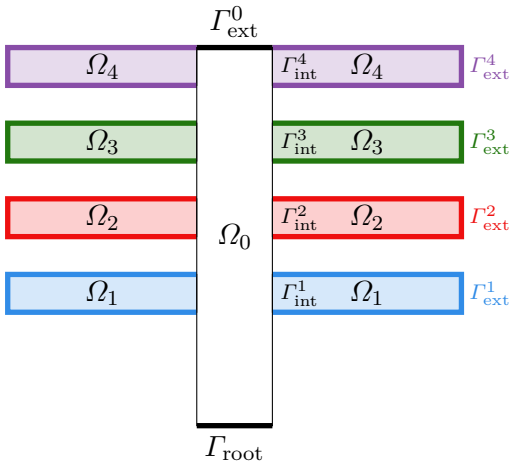


Figure 3.6: Geometry of the thermal fin, with a central post Ω_0 and 4 fins Ω_i.

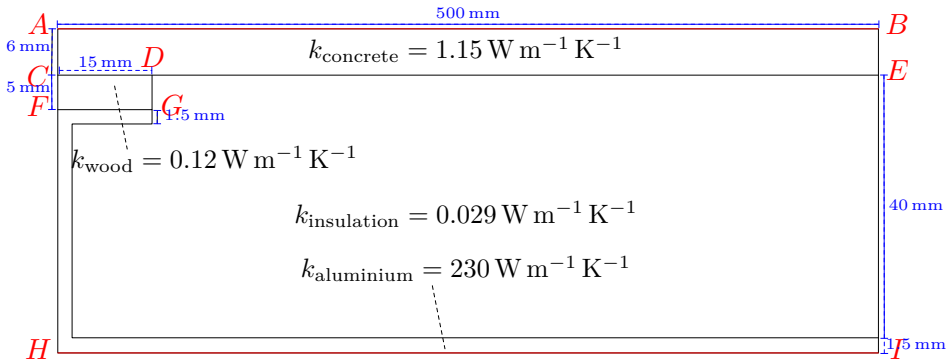


Figure 3.7: Thermal bridge geometry from [Sta07], not on scale.

The geometry is presented in Figure 3.7, as well as the points where the temperature is tabulated. The boundary $\partial\Omega$ is split into two parts: Γ_{top} and Γ_{bottom}.

The temperature T is governed by the elliptic PDE:

$$-\nabla \cdot (k\nabla T) = 0 \quad \text{in } \Omega_i, \tag{3.64a}$$

coupled with the following boundary conditions:

$$-k\nabla T \cdot n = h_{\text{top}}(T - T_0^{\text{top}}) \qquad \text{on } \Gamma_{\text{top}}, \tag{3.64b}$$

$$-k\nabla T \cdot n = h_{\text{bottom}}(T - T_0^{\text{bottom}}) \quad \text{on } \Gamma_{\text{bottom}}, \tag{3.64c}$$

with $h_{\text{top}} = 16.67\,\text{W m}^{-2}\,\text{K}^{-1}$, $h_{\text{bottom}} = 9.09\,\text{W m}^{-2}\,\text{K}^{-1}$, $T_0^{\text{top}} = 0\,°\text{C}$, $T_0^{\text{bottom}} = 20\,°\text{C}$, and k depending on the material, as presented in Figure 3.7.

No variability on the parameters is accounted for in [Sta07], we therefore consider the following range for the parameters:

- the thermal conductivities k of the materials are in the range of $\pm 10\%$ of the nominal values,
- the bottom temperature T_0^{bottom} lies in the interval $[10, 30]$, and
- the top temperature T_0^{top} lies in the interval $[-5, 5]$.

Norm	$\left\|u_h - u_{Hh}^{N,\text{NoRect}}\right\|$	$\left\|u_h - u_{Hh}^{N,\text{Rect}}\right\|$
L^2	$2.07 \cdot 10^{-3}$	$9.01 \cdot 10^{-4}$
L^∞	$8.01 \cdot 10^{-2}$	$1.85 \cdot 10^{-2}$

Table 3.3: Statistics on the error committed by the NIRB method in 2D thermal fin problem with a reduced basis of size 64, for $\mu = \{\text{Bi} : 0.1, k_1 : 0.1, k_2 : 0.5, k_3 : 0.1\}$.

3.2.6 Numerical results

In this section, we present some numerical results of the NIRB method. To highlight the efficiency of the method, we will present convergence results by studying the models described above. As the analytical solution cannot be computed for these problems, we use the FE solution computed on the fine mesh as a reference. To compute the error, we employ the following strategy: we select a random sample of $N_s = 50$ parameters in D^μ, and we compute the average error over this sample. The norm of the error is then defined by:

$$e_{\text{NIRB}} := \left\|u_h^{\mathcal{N}} - u_{Hh}^N\right\|_{H^k(\Omega)} = \frac{1}{N_s}\sum_{j=1}^{N_s}\left\|u_h^{\mathcal{N}}(\mu_j) - u_{Hh}^N(\mu_j)\right\|_{H^k(\Omega)}, \tag{3.65}$$

where $u_h^{\mathcal{N}}$ is the FE solution on the fine mesh, and u_{Hh}^N is the NIRB approximation. The *reference error* is defined as the difference between the FE solution and its projection in the reduced space:

$$e_{\text{ref}} := \left\|u_h^{\mathcal{N}} - u_h^N\right\|_{H^k(\Omega)} = \frac{1}{N_s}\sum_{j=1}^{N_s}\left\|u_h^{\mathcal{N}}(\mu_j) - \sum_{i=1}^{N}\left\langle u_h^{\mathcal{N}}(\mu_j), \xi_h^i\right\rangle_{H^k}\xi_i\right\|_{H^k(\Omega)}. \tag{3.66}$$

Note that for $k = 0$, $H^0(\Omega) = L^2(\Omega)$.

2D case

We perform an error convergence study of the NIRB method for the 2D thermal fin problem described above. For now, the procedure use to generate the basis is the random selection, without the greedy algorithm introduced in Algorithm 7. A comparison between both algorithms will be discussed in Section 3.2.6.

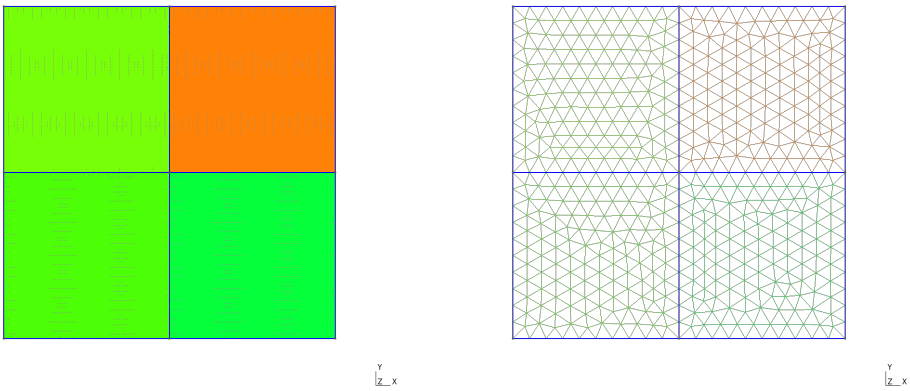
The geometrical parameter used on GMSH to generate those meshes are $h = h_{\text{fin}} = 0.0025$ and $H = h_{\text{coarse}} = 0.05$. We present the fine and coarse meshes used in Figure 3.8. A reduced basis of size $N = 64$ is generated. We set $\mu = \{\text{Bi} : 0.1, k_1 : 0.1, k_2 : 0.5, k_3 : 0.1\} \in D^\mu$. We compute on the one hand the FE solution on the fine mesh $u_h^{\mathcal{N}}(\mu)$, that is displayed in Figure 3.9(a). On the other hand, we compute the NIRB solution $u_{Hh}^N(\mu)$ with and without rectification, obtained with a basis of size 64. Figures 3.9(b) and 3.9(c) show the difference $|u_h^{\mathcal{N}}(\mu) - u_{Hh}^N(\mu)|$ with and without the rectification post-process. Statistics on the error committed are computed and provided in Table 3.3. We also present in Table 3.4 the computational time of the NIRB method, with and without rectification, in sequential and parallel. The associated speedup, namely the ratio of the computational time with respect to the sequential fine FE computational time, is also given. We remark a speedup of around 3.7 in sequential and 1.4 in parallel. Moreover, we remark that the computational time added by the rectification post-process is negligible compared to the time saved by the method.

A first analysis of the results shows that the NIRB approximation without rectification seems better on Figure 3.9(b) compared to the solution with it, but we notice that the error is not uniformly distributed on the domain, but localized on small domains, while the error with rectification is more uniformly distributed, and is 10 times smaller. The norms computed

Element computed		Computational time		Speedup	
		Sequential	Parallel (`np=4`)	Sequential	Parallel (`np=4`)
Fine snapshots	$u_h^{\mathcal{N}}$	1.49 s	0.610 s	–	2.44
Coarse snapshots	$u_H^{\mathcal{N}}$	0.0297 s	0.0255 s	50.1	58.4
NIRB w/ rectification	$u_{Hh}^{N,\text{Rect}}$	0.396 s	0.179 s	3.76	8.32
NIRB w/o rectification	$u_{Hh}^{N,\text{NoRect}}$	0.380 s	0.158 s	3.92	9.43

Table 3.4: Time of execution and associated speedup to run the NIRB method for the 2D thermal fin problem with $\mathbb{P}_1$ discretization, mean over 64 parameters, in sequential and parallel.

in Table 3.3 confirm this observation: the rectification post process leads to an error 10 times smaller.



(a) Fine mesh $\mathcal{N}_h = 185\,893$.

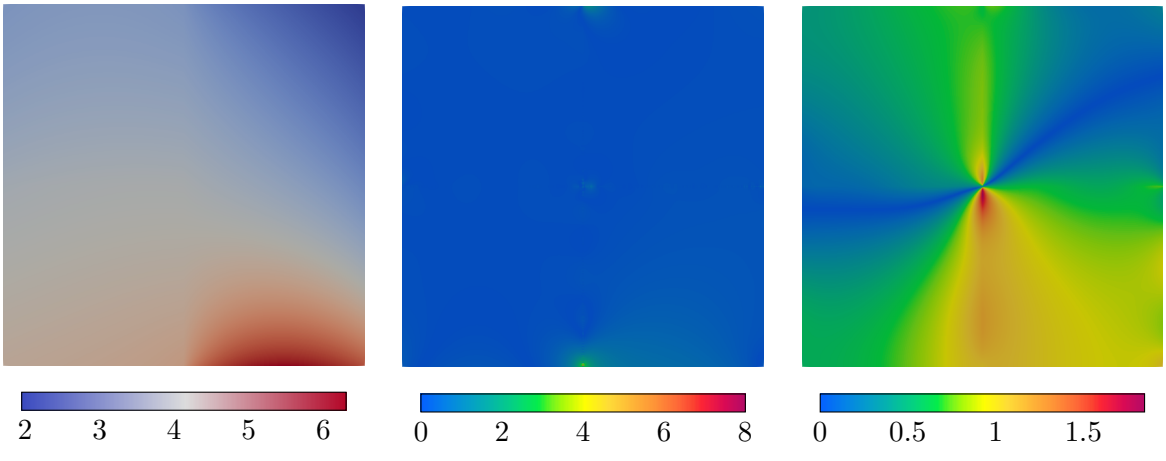
(b) Coarse mesh, $\mathcal{N}_H = 533$.

Figure 3.8: Coarse and fine mesh used in the NIRB simulation in 2D square domain.

In Figure 3.10 we present the convergence errors with respect to the number of basis functions. We notice that without rectification, the NIRB method does not improve that much the error compared to the interpolated solution. For the H^1 norm, we even notice a degradation of the error when the number of basis functions increases. On the other hand, with rectification, the error is 10 times smaller than the interpolated solution. In both cases, for high values of N, the error tend to stagnate on a plateau.

Thermal bridge: ISO 10211:2007 We focus in this part on the benchmark test case provided by the ISO 10211:2007 norm [Sta07], that was described in Section 3.2.5. The discretization parameters taken are: $H = 0.02$ and $h = 0.0004$. This leads to a number of degrees of freedom of $\mathcal{N}_H = 158$ and $\mathcal{N}_h = 175\,621$. We show in Figure 3.11 the FE solution for the nominal parameter $\mu = \{T_0^{\text{bottom}} : 20, T_0^{\text{top}} : 0, k_{\text{aluminium}} : 230, k_{\text{concrete}} : 1.15, k_{\text{insulation}} : 0.029, k_{\text{wood}} : 0.12\}$.

In Table 3.5, we represent a comparison between reference values and those obtained with NIRB method. The first row provides the expected values at different points (A to I) from the benchmark [Sta07]. According to the norm, the error should not exceed 0.1 °C, that is not the case for all points concerning the NIRB approximation without rectification: the reported values lie outside the range of tolerable error at points D, F, and G. When the rectification is enabled, values reported exhibit slight improvements, especially noticeable at the three points D, F, and H, where the rectified results are closer to the expected values compared to the non-rectified



(a) FE solution $u_h^{\mathcal{N}}(\mu)$. (b) $\left|u_{Hh}^{N,\text{NoRect}}(\mu) - u_h^{\mathcal{N}}(\mu)\right|$. (c) $\left|u_{Hh}^{N,\text{Rect}}(\mu) - u_h^{\mathcal{N}}(\mu)\right|$.

Figure 3.9: Comparison between FE solution and NIRB solution, with and without rectification for the 2D test case, with a reduced basis of size 64.

NIRB results. We also provided the results obtained with the FE solution. The results indicate that the rectification process enhances the accuracy of the NIRB method, bringing it closer to the high-fidelity solution while maintaining computational efficiency.

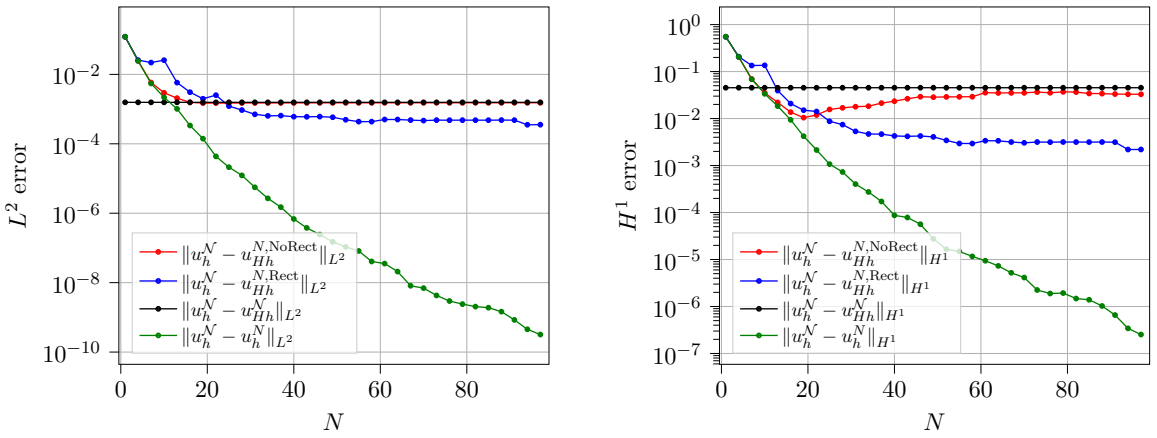
Point	A	B	C	D	E	F	G	H	I
Expected value	7.1	0.8	7.9	6.3	0.8	16.4	16.3	16.8	18.3
$u_{Hh}^{N,\text{NoRect}}$	7.089	0.741	7.91	**7.026**	0.822	**16.225**	**17.939**	16.746	18.33
$u_{Hh}^{N,\text{Rect}}$	7.065	0.761	7.898	6.274	0.827	16.408	16.334	16.767	18.334
$u_h^{\mathcal{N}}$	7.065	0.761	7.898	6.274	0.827	16.408	16.334	16.767	18.334

Table 3.5: Results of benchmark ISO 10211:2007 for NIRB method, $\mathcal{N}_h = 128$ and $\mathcal{N}_H = 175\,621$, with a basis of size $N = 30$. The quantities are presented in °C. The expected values are taken from [Sta07]. The quantities displayed in **red boldface** indicates the values that are outside the range of tolerable error of 0.1 °C.

We also present in Figure 3.12 the results of the convergence study performed with this test-case. They are in good agreement with what was previously observed in the thermal fin case: the error decreases as the number of basis functions increases and finally reaches a plateau. Moreover, the rectification post-process significantly improves the accuracy of the NIRB method, as the error is divided by a factor of 100 compared to the NIRB approximation without rectification.

We display in Table 3.6 the computational time of the NIRB method for the ISO 10211:2007 benchmark test case. We observe similar behavior as the previous test cases: the NIRB approximation is computed faster than the FE solution, with a speedup of 4.7 in sequential mode and 10.9 in parallel mode. Moreover, the computational cost of the rectification post process is negligible, as the time of execution is almost the same with or without rectification.

3D case Now we focus on a 3D case, the thermal fin, as described in Problem (3.63). The following discretization parameters are taken: $H = 0.25$ and $h = 0.0625$, leading to a number of degrees of freedom of $\mathcal{N}_H = 6785$ and $\mathcal{N}_h = 171\,493$. A basis of size $N = 64$ is generated using the POD algorithm.



(a) Norm $\|\cdot\|_{L^2}$. (b) Norm $\|\cdot\|_{H^1}$.

Figure 3.10: Convergence error in respect to basis size for the 2D thermal case, with $\mathcal{N}_h = 185894$ and $\mathcal{N}_H = 532$

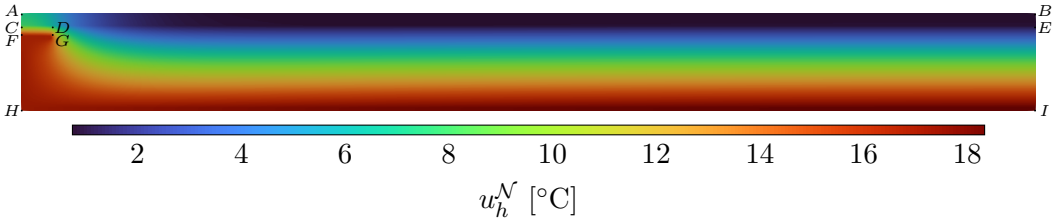


Figure 3.11: Solution $u_h^{\mathcal{N}}$ for the ISO 10211:2007 benchmark test case.

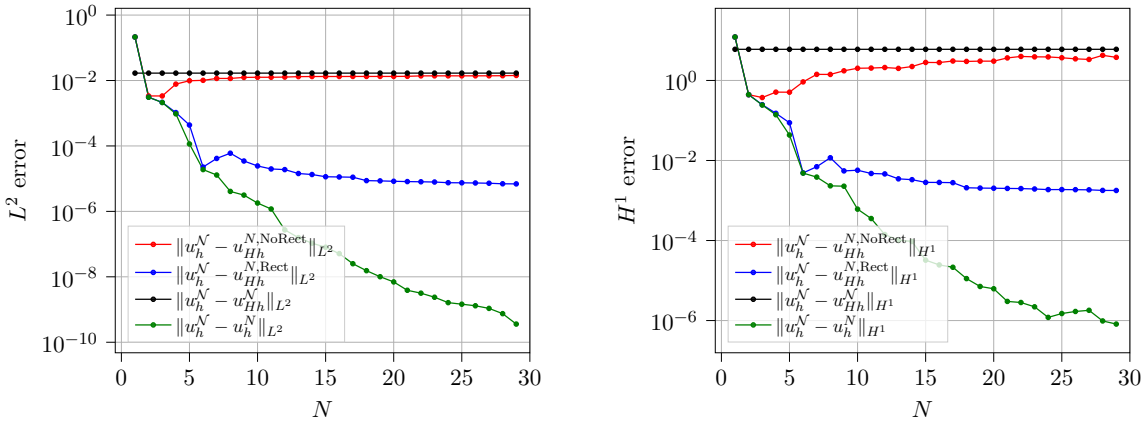
In Figure 3.13(a) we display the FE solution, among the error committed by the NIRB approximation with and without rectification in Figures 3.13(b) and 3.13(c). As for the results of the other test cases presented in the previous sections, the rectification post-process significantly improves the accuracy of the NIRB method, even though the error seems to be less uniformly distributed in the 3D case.

We also present in Figure 3.14 the convergence error in L^2 and H^1 norm in respect to basis function. The results are still in good agreement with what was observed in the 2D cases, as well as results from the literature for CFD problems [CMP19].

A comparison of the computational time is also provided in Table 3.7, showing that the NIRB method is faster than the FE solution, as we observe a speedup of 6.5 in sequential mode and 17 in parallel mode. We can also note that the rectification post-process does not significantly impact the computational time, as the speedup remains almost the same.

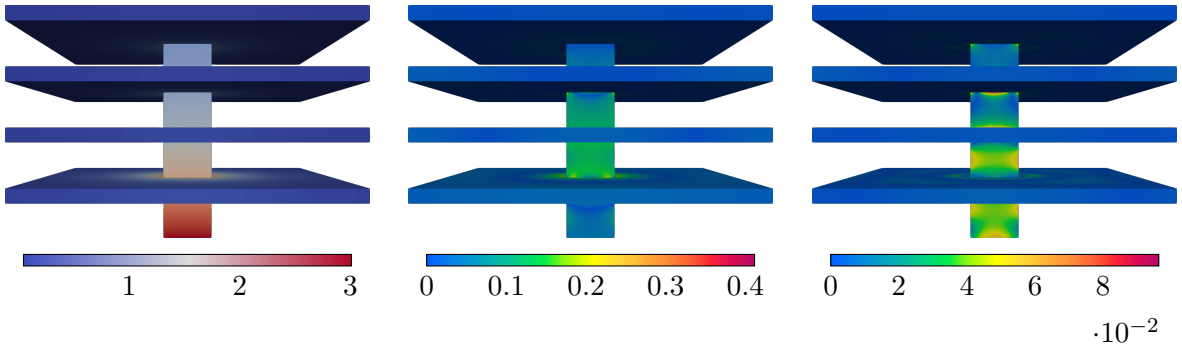
Element computed		Computational time		Speedup	
		Sequential	Parallel (`np=12`)	Sequential	Parallel (`np=12`)
Fine snapshots	$u_h^{\mathcal{N}}$	2.03 s	0.679 s	–	2.990
Coarse snapshots	$u_H^{\mathcal{N}}$	0.0508 s	0.0418 s	39.96	48.56
NIRB w/ rectification	$u_{Hh}^{N,\text{Rect}}$	0.4277 s	0.1861 s	4.746	10.91
NIRB w/o rectification	$u_{Hh}^{N,\text{NoRect}}$	0.4276 s	0.1859 s	4.747	10.92

Table 3.6: Time of execution and associated speedup to run the NIRB method benchmark ISO 10211 case, mean over 64 parameters, in sequential and parallel.



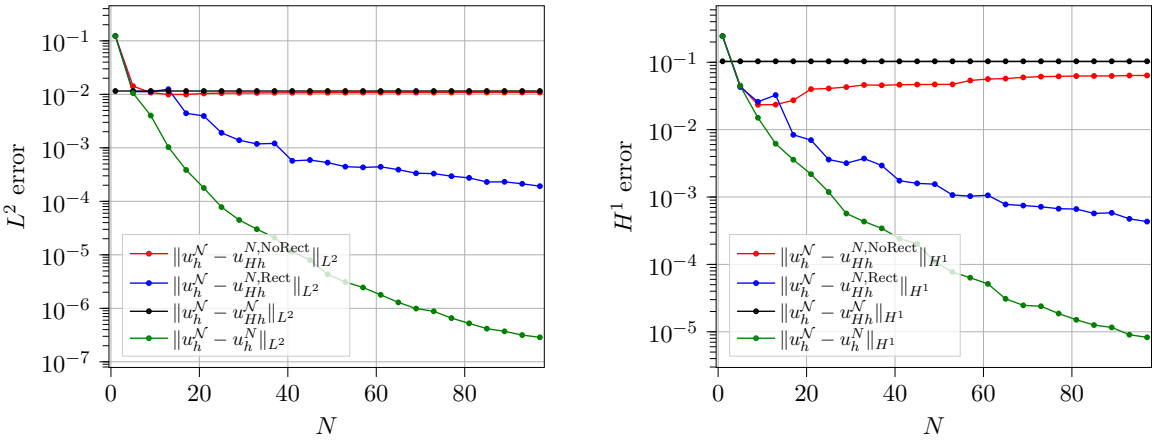
(a) L^2 norm in 2D. (b) H^1 norm in 2D.

Figure 3.12: Convergence error with respect to the basis size N for the 2D ISO 10211:2007 case, with $\mathcal{N}_h = 773\,810$ and $\mathcal{N}_H = 564$.



(a) FE solution $u_h^{\mathcal{N}}(\mu)$. (b) $\left|u_{Hh}^{N,\mathrm{NoRect}}(\mu) - u_h^{\mathcal{N}}(\mu)\right|$. (c) $\left|u_{Hh}^{N,\mathrm{Rect}}(\mu) - u_h^{\mathcal{N}}(\mu)\right|$.

Figure 3.13: High fidelity solution, and NIRB approximation for the 3D thermal fin case, with $\mathcal{N}_H = 6785$ and $\mathcal{N}_h = 171\,493$, and $N = 30$.



(a) L^2 norm in 2D. (b) H^1 norm in 2D.

Figure 3.14: Convergence error in respect to basis size for the 3D thermal fin case.

Element computed		Computational time		Speedup	
		Sequential	Parallel (`np=12`)	Sequential	Parallel (`np=12`)
Fine snapshots	$u_h^{\mathcal{N}}$	3.50 s	0.523 s	–	6.692
Coarse snapshots	$u_H^{\mathcal{N}}$	0.221 s	0.145 s	15.8	24.14
NIRB w/ rectification	$u_{Hh}^{N,\text{Rect}}$	0.5417 s	0.205 39 s	6.46	17.04
NIRB w/o rectification	$u_{Hh}^{N,\text{NoRect}}$	0.5424 s	0.205 40 s	6.45	17.03

Table 3.7: Time of execution and associated speedup to run the NIRB method for the 3D thermal fin problem with $\mathbb{P}_1$ discretization, mean over 64 parameters, in sequential and parallel.

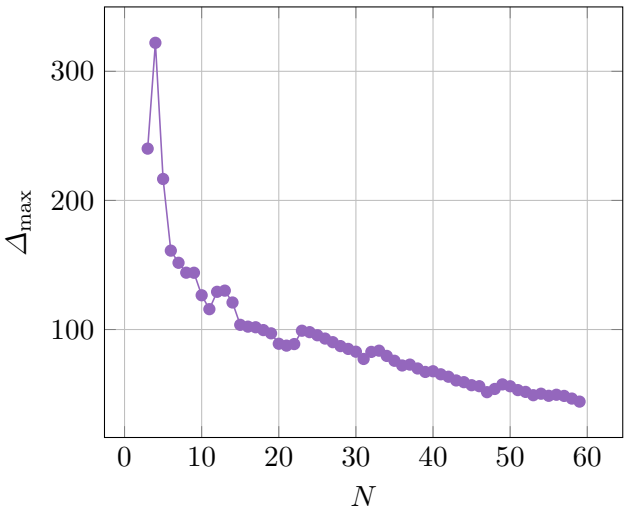


Figure 3.15: Convergence of the maximal error $\Delta_{\max}$ during greedy procedure.

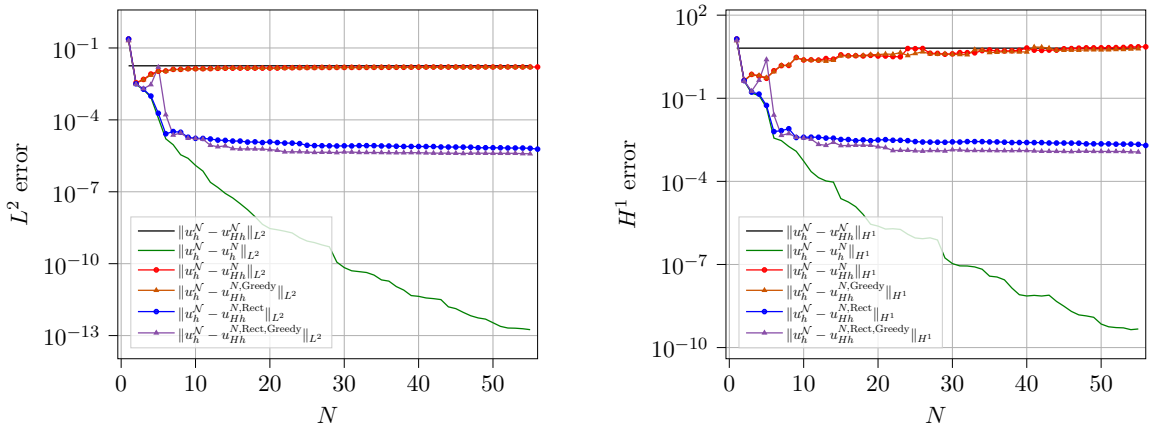
Greedy vs. random selections In Algorithm 7, we introduced a procedure to greedily select the functions that are added to the reduced basis. We are now comparing this approach to a random selection of the basis functions.

We perform the comparison on the ISO 10211:2007 benchmark test case, with a sample of training Ξ_{train} of size $N_{\text{train}} = 500$. Before the convergence criterion is reached, 57 basis functions have been selected. We present in Figure 3.15 the convergence of the maximal error $\Delta_{\max} = \max_{\mu \in \Xi_{\text{train}}} \Delta(\mu)$ during the greedy procedure. Note that as the criterion is performed on the online prediction, it is quite fast to compute the value of $\Delta(\mu)$ for each parameter $\mu \in \Xi_{\text{train}}$: on a run parallelized on 12 processors, about 10 seconds are necessary to compute the error for the 500 parameters.

In Figure 3.16 we present the convergence of the error in L^2 and H^1 norm for the two algorithms. When no rectification post-process is applied, we observe no difference between the two approaches, both converging to the same error are the interpolation error. On the other hand, when the rectification is enabled, the greedy algorithm allows to reach a slightly better result with a smaller number of basis functions: the limit error is divided by a factor of 2 when the greedy algorithm is used to select the snapshots.

3.2.7 NIRB method applied to the bioheat transfer model of the human eyeball

In this section, we present the results of the NIRB model applied to the model $\mathcal{E}_{\text{L}}$ of heat transfer in the human eyeball, introduced in Section 1.4. As this model is well-suited to the RBM, we do not expect NIRB to provide a significant improvement, but still, we can compare



(a) L^2 norm. (b) H^1 norm.

Figure 3.16: Comparison of convergence error for both greedy and random selection of basis functions, the triangle plots are obtained thought the greedy procedure, while dot plots come from a random sampling.

Element computed		Computational time			Speedup		
		`np = 12`	`np = 64`	`np = 128`	`np = 12`	`np = 64`	`np = 128`
Fine snapshots	$u_h^{\mathcal{N}}$	13.6 s	4.79 s	2.38 s	–	2.84	5.71
Coarse snapshots	$u_H^{\mathcal{N}}$	2.77 s	1.10 s	0.859 s	4.90	12.36	15.83
NIRB w/o rectification	$u_{Hh}^{N,\text{NoRect}}$	3.190 s	1.309 s	1.0687 s	4.26	10.39	12.73
NIRB w/ rectification	$u_{Hh}^{N,\text{Rect}}$	3.203 s	1.311 s	1.0691 s	4.25	10.37	12.72

Table 3.8: Time of execution and associated speedup to run the NIRB method for the 3D thermal fin problem with $\mathbb{P}_1$ discretization, mean over 64 parameters, in sequential and parallel.

the results of both methods.

The first point of comparison is the computational time of the online phases of both method. Previously, we have seen that the online phase of the RBM is very fast, as it only requires the evaluation of the reduced basis functions, see Table 3.1. On the other hand, the online phase of NIRB takes 3.34 s to compute the solution for the same problem, which is still an improvement compared to the high-fidelity resolution (10.76 s) when the online execution is performed on 12 parallel processes.

The second point of comparison is the convergence of the L^2 error, that is displayed in Figure 3.17. The field error is presented. For the RBM method, we only computed the error for basis of size to $N = 10$, while the convergence study of NIRB has been performed for greater values of N. Still we figure that the error of the NIRB method is greater than the error of the RBM method, especially when the rectification post process is not applied.

Compared to the previous convergence of the NIRB method, we notice another difference: the error of the NIRB approximation without rectification $\|u_h^{\mathcal{N}} - u_{Hh}^{N,\text{NoRect}}\|$ used to be superposed to the interpolation error $\|u_h^{\mathcal{N}} - u_{Hh}^{\mathcal{N}}\|$ for big values of N, which is not observed for this case: the NIRB error is 10 times smaller than the interpolation error.

In Figure 3.18, we display both the coarse and fine snapshots of the solution. The coarse mesh, denoted as `M0`, consists of 47,284 elements, while the fine mesh, `M3`, contains 207,845 elements. Additionally, Figure 3.19 shows the solution obtained using the NIRB method for the same problem, employing a reduced basis of size $N = 30$. Without the rectification post-process, we observe that the solution is highly inaccurate — the "norm of the eye" clearly illustrates this.

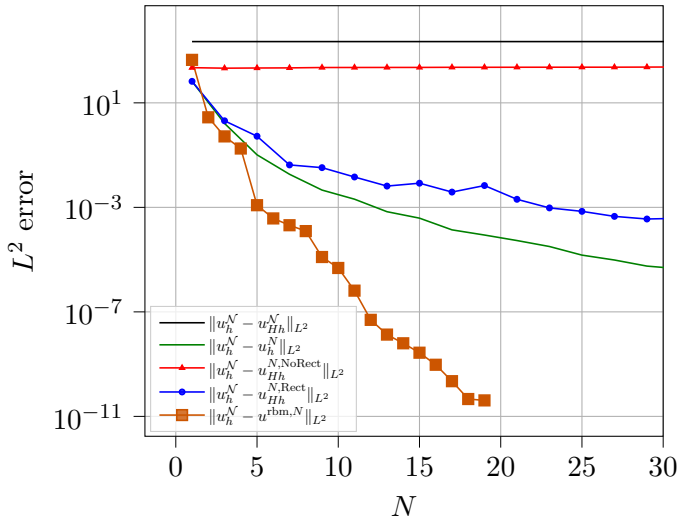
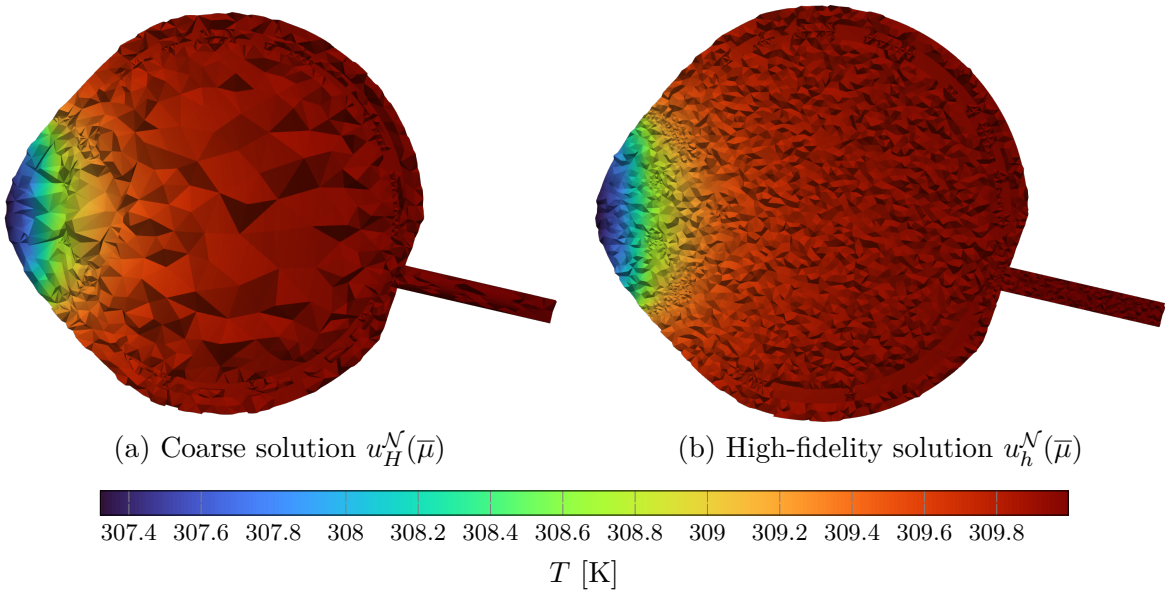


Figure 3.17: Convergence of the L^2 error for the NIRB model and the RBM model, applied to the 3D heat transfer model in the human eyeball.

However, after applying rectification, the solution aligns very closely with the high-fidelity result. The computed errors are as follows:

$$\|u_h^{\mathcal{N}} - u_{Hh}^{N,\mathrm{NoRect}}\|_{L^2} = 8.28 \cdot 10^{-3}, \quad \|u_h^{\mathcal{N}} - u_{Hh}^{N,\mathrm{Rect}}\|_{L^2} = 4.7 \cdot 10^{-7}. \tag{3.67}$$

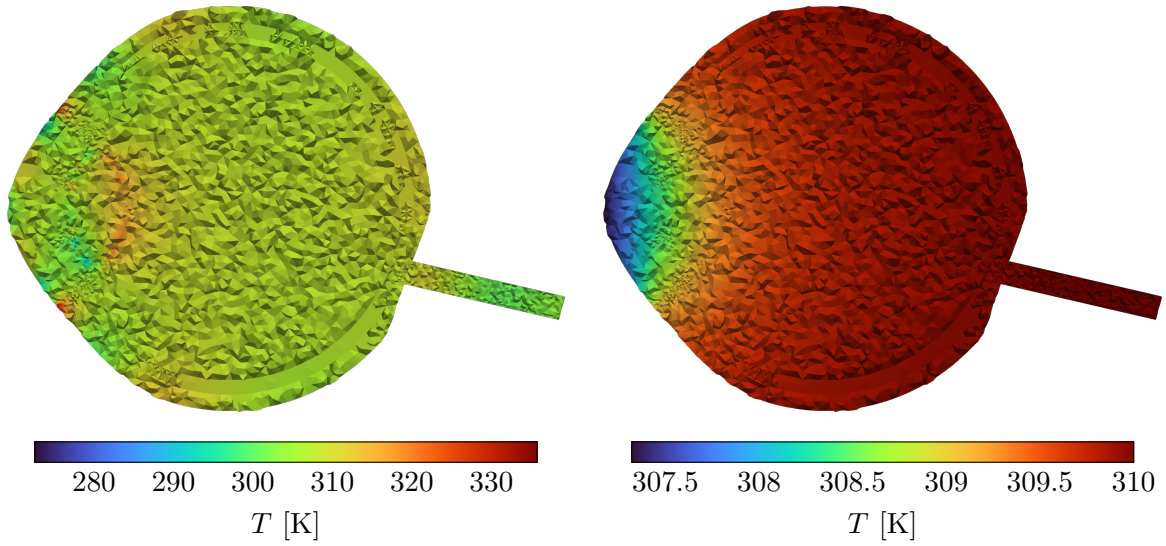


(a) Coarse solution $u_H^{\mathcal{N}}(\overline{\mu})$ (b) High-fidelity solution $u_h^{\mathcal{N}}(\overline{\mu})$

Figure 3.18: Comparison of the coarse and high-fidelity solutions of the NIRB model applied to the 3D heat transfer model in the human eyeball.

3.3 Conclusion and outlook

In this chapter, we have presented two methods to reduce the computational cost of solving parametric problems.



(a) NIRB approximation without rectification $u_{Hh}^{N,\text{NoRect}}(\overline{\mu})$. (b) NIRB approximation with rectification $u_{Hh}^{N,\text{Rect}}(\overline{\mu})$.

Figure 3.19: NIRB approximation, with and without the rectification post process for the bio-heat transfer model in the eyeball. For this test-case, we have $\mathcal{N}_H = 47\,284$ and $\mathcal{N}_h = 207\,845$, and $N = 30$.

The Certified Reduced Basis Method has been applied to the heat transfer problem in the eye model, and we have shown that the method is efficient to significantly reduce the computational cost of solving the parametric problem, and that the results are accurate, thanks to the error bound provided by the method. This significant computational gain allows us to explore the parametric domain of the problem and perform the sensitivity analysis, that requires to solve the problem for several parameters, and a high number of evaluations. The sensitivity analysis is presented in Chapter 5.

We also presented the Non-Intrusive Reduced Basis method, another model order reduction technique that allows to reduce the computational cost of solving parametric problems. We have presented the mathematical framework of the method, and the results obtained on the test cases, and shown that the method is efficient to reduce the computational cost of solving the parametric problem, and that the results are accurate. This method has only been applied to the heat transfer eye model in the context of this thesis, but it is a promising method to reduce the computational cost of solving the eye model for various parameters, especially in the context of non-linear problems, such as the coupled heat-fluid transfer problem that is described in Chapter 6, because of the non-intrusive nature of the method. Moreover, the NIRB method has been extended in [GM23a] by the use of domain truncation, allowing to further reduce the computational time of the online time. The *domain truncation*, also called *zoom*, consists in solving the problem on a smaller domain $\omega \subset \Omega$, and to reconstruct the solution on the full domain Ω. As a perspective, this strategy would be well suited in the context of point-wise outputs-of-interest or to capture the behavior of the solution in a specific region of the domain.

Chapter 4

Low to high order finite element resolution for elliptic problems in the presence of a Dirac source term

In the previous chapter, we applied a reduced order modeling to analyze heat transfer within the human eyeball, employing the reduced basis method (RBM) strategy [Pru+02]. To obtain fast quadratic convergence in the output, we relied on a dual problem formulation associated with each output expressed as a linear functional of the solution. However, in some cases of interest, the output functional ℓ can be a pointwise evaluation of the solution u, *e.g.* $\ell_x(u) = u(x) = \delta_x(u)$, for $x \in \Omega$. Thus, the formulation leads to a dual problem with Dirac source term δ_x with $x \in \Omega$, the point where we evaluate the solution of the primal problem. This approach was initially tailored for the heat transfer analysis problem described in Chapter 1, but more generally addresses similar challenges in other fields, such as acoustics and elasticity, where pointwise evaluations are frequently encountered.

We consider the context of a parametric problem, where we study a quantity of interest that depends on a set of parameters. Precisely, the problem reads: for a given *parameter* μ, find $s(\mu) = \ell(u(\mu))$, where $u(\mu) \in H^1(\Omega)$ is the solution to the following PDE:

$$\begin{cases} -\nabla \cdot (k \nabla u) = f & \text{in } \Omega, \\ \partial_{\vec{n}} u = k(u - u_0) & \text{on } \partial\Omega. \end{cases} \tag{4.1}$$

with $\ell(u) = \langle \delta_O, u \rangle$ is discontinuous functional (*non-compliant* case). The boundary condition $\partial_{\vec{n}} u = k(u - u_0)$ on $\partial\Omega$ represents a Robin boundary condition, where k is the thermal conductivity and u_0 is the ambient temperature. The notation μ enlightens the parametrical dependence of the initial problem, which represents in our context the thermal conductivities of biological regions of the eyeball, as well as external factors such as the ambient temperature, convection coefficient... The variational form of Problem (4.1) reads as: find $u(\mu) \in H^1(\Omega)$ such that

$$a(u(\mu), v; \mu) = f(v; \mu) \quad \forall v \in H^1(\Omega).$$

where $a(\cdot, \cdot; \mu)$ and $f(v; \mu)$ are obtained through the usual weak formulations of the underlying PDE.

In the context of the certified RBM presented in Section 3.1, to recover good properties of the reduced solution, we need to compute the solution of the dual problem [Pru+02]: find $\psi(\mu) \in H^1(\Omega)$ such that

$$a(u, \psi(\mu); \mu) = -\ell(u) \quad \forall u \in H^1(\Omega).$$

The issue raised is that the functional $u \mapsto \langle \delta_O, u \rangle$ is not well-defined in H^1, so the usual theory — Lax-Milgram, minimization results — does not apply. Nevertheless, we achieve quadratic

convergence in numerical resolution, consistent with theoretical predictions in the continuous case, as discussed in Section 3.1.4.

Theoretical and numerical aspects of the Laplacian problem with a Dirac source term and homogeneous Dirichlet boundary condition have been studied in [KW14; Ber+18]. While theoretical results are derived under specific hypotheses, we observe similar outcomes even when these conditions are not strictly met. In the present chapter, we provide a theoretical review and numerical exploration of these models under various boundary conditions and extend the analysis to generalized elliptic problems with discontinuous coefficients. Additionally, we examine the effect of the Dirac source term's position relative to the boundary domain $\partial\Omega$. Our simulations, spanning from low to high-order discretizations, are conducted using the finite element method within the open-source software Feel++ [Pru+24b], and the results are discussed in detail.

The work of this chapter is a collaboration with Silvia Bertoluzza (CNR), and is organized as follows: in Section 4.1, we present the benchmark model used to perform our numerical investigations, as well as the computational framework. Then, in Section 4.2, we review the theoretical results and numerical methods used to solve the problem. In Section 4.3, we present the numerical results obtained for the Laplacian problem with a Dirac source term, focusing on the influence of the distance between the Dirac source term and the boundary. Finally, we conclude in Section 4.5 and provide outlooks for future research directions.

4.1 Problem statement

We first introduce the benchmark problem stemming from the elliptic problem stated in Equation (4.1), and from available tests in the literature [Ber+18]. Let $\Omega \subset \mathbb{R}^2$ be a bounded domain. To begin with, we consider a circular domain described in Figure 4.1, even though in a subsequent part, we will study the behavior under geometrical singularities as well. Let $\vec{X}_0 = (x_0, y_0) \in \Omega$.

4.1.1 Benchmark problem

We focus on the problem defined by the Laplace equation with a Dirac delta distribution located at $\vec{X}_0$ in the right-hand side. We consider this problem under the three commonly encountered boundary conditions: Dirichlet, Neumann, and Robin. The problem reads: find u

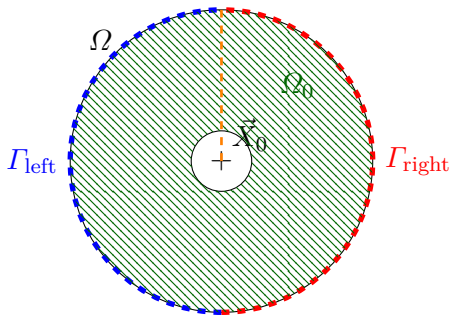


Figure 4.1: Description of the domain Ω, the dashed lines represent the names of the domains where mixed boundary conditions stand, and the green domain is the domain $\Omega_0 = \Omega \setminus B(\vec{X}_0, r)$ (for $r > 0$) where the error is computed.

such that

$$\begin{cases} -\Delta u = \delta_{\vec{X}_0} & \text{in } \Omega, \\ + \text{ boundary conditions} & \text{on } \partial\Omega. \end{cases} \tag{4.2}$$

The boundary conditions considered on $\partial\Omega$ can be of various types: *(i)* homogeneous Dirichlet ($u = 0$ on $\partial\Omega$), *(ii)* homogeneous Neumann ($\partial_{\vec{n}} u = 0$), *(iii)* Robin ($\partial_{\vec{n}} u + \mu u = g$), and *(iv)* mixed boundary conditions. Regarding the last case, we partition the boundary $\partial\Omega$ into two parts: Γ_{left} and Γ_{right}, where one of the three boundary conditions is applied.

Definition 4.1.1. Let $G\colon \Omega \to \mathbb{R}$ be the *Green's function* defined by

$$G(x, y) = -\frac{1}{2\pi} \log\left(\sqrt{(x - x_0)^2 + (y - y_0)^2}\right). \tag{4.3}$$

Proposition 4.1.2. *The Green function G satisfies*

$$-\Delta G = \delta_{\vec{X}_0} \quad \text{in } \Omega. \tag{4.4}$$

Moreover, the gradient of G reads:

$$\nabla G(x, y) = \begin{bmatrix} \frac{x - x_0}{2\pi\left((x-x_0)^2 + (y-y_0)^2\right)} \\ \frac{y - y_0}{2\pi\left((x-x_0)^2 + (y-y_0)^2\right)} \end{bmatrix}. \tag{4.5}$$

Proof. Without any loss of generality, we can assume that $\vec{X}_0 = (0, 0)$. We denote by δ_0 the Dirac distribution $\delta_{(0,0)}$. The solution $G(x, y)$ must be radially symmetric because the source δ_0 is located at a single point and is isotropic, meaning the heat (or other physical quantity) is distributed equally in all directions from the source. Consequently, the solution depends only on the distance from the origin and not on the specific direction. Thus, we have

$$G(x, y) = G(r) \quad \text{with } r = \sqrt{x^2 + y^2}.$$

Expressing the Laplacian in polar coordinates, we have $\Delta G(r) = G''(r) + \frac{1}{r} G'(r)$. The function G being solution of Equation (4.4), we have: $G''(r) + \frac{1}{r} G'(r) = 0$. So $G(r) = c_1 \log(r) + c_2$, for some constants $c_1, c_2 \in \mathbb{R}$. Without any loss of generality, we can assume that $c_2 = 0$. Let $R > 0$, and define the disk $B(0, R) := \{(x, y) \in \mathbb{R}^2 \mid x^2 + y^2 \leqslant R^2\}$. We also note $\vec{X} = (x, y)$. We have:

$$1 = \int_\Omega \delta_0(\vec{X})\, \mathrm{d}\vec{X} = -\int_\Omega \Delta G(\vec{X})\, \mathrm{d}\vec{X} = -\int_{B(0,R)} \Delta G(\vec{X})\, \mathrm{d}\vec{X} = -\int_{\partial B(0,R)} \partial_{\vec{n}} G(\vec{X})\, \mathrm{d}S,$$

from the divergence theorem. As the normal derivative of G along the boundary of the disk is equal to the derivative of G in the radial direction evaluated at R, we get $\partial_{\vec{n}} G(\vec{X}) = G'(R)$, for $\|\vec{X}\| = R$. Moreover, $G'(R)$ is constant over the boundary of the disk, therefore:

$$-\int_{\partial B(0,R)} \partial_{\vec{n}} G(\vec{X})\,\mathrm{d}S = G'(R)\int_{\partial B(0,R)} \mathrm{d}S = 2\pi R G'(R).$$

Finally, we have $1 = -2\pi R G'(R)$, so $G'(R) = -\dfrac{1}{2\pi R}$, and therefore $G(R) = -\dfrac{1}{2\pi}\log(R)$. We deduce the value $c_1 = -\frac{1}{2\pi}$.

In conclusion the function $G(x,y) = -\frac{1}{2\pi}\log(r) = \log\left(\sqrt{x^2+y^2}\right)$ is a solution of Equation (4.4). A straightforward computation shows that the gradient of G is as stated. ■

Note that is Ω is a circle centered in $\vec{X}_0$ and with radius $R > 0$, then G is precisely the solution to the Problem (4.2) endorsed with Dirichlet boundary conditions.

4.1.2 Computational Framework

We solve Problem (4.2) using the finite element method (FEM). Denote by $V = H^1(\Omega)$ the Sobolev space where the solution of Equation (4.2) belongs.

The domain Ω is discretized with a mesh size h, as illustrated in Figure 4.2(a). Note that the Dirac delta location in Ω is *not* associated with a mesh node. We introduce the Lagrange finite element space $V_h^k \subset V$, constructed using $\mathcal{C}^0$ polynomials of degree k. The finite element solution $u_h \in V_h^k$ is obtained by solving the following weak formulation:

$$\int_\Omega \nabla u_h \cdot \nabla v_h = \left\langle \delta_{\vec{X}_0}, v_h \right\rangle \quad \text{for all } v_h \in V_h^k. \tag{4.6}$$

Here, Equation (4.6) corresponds to the case where Dirichlet boundary conditions are considered. For other boundary conditions, additional terms are added to the weak formulation.

The numerical solution u_h is computed by solving the weak formulation (4.6) and the library Feel++ [Pru+24b] is used for the implementation. The details of the implementation are provided in Section 7.4.

To accurately capture the behavior near the singularity, we refine the mesh around $\vec{X}_0$. We apply a mesh refinement strategy based on the distance to the Dirac delta location, defined by the metric:

$$h_{\text{disc}}(x,y) = (x-x_0)^2 + (y-y_0)^2 \quad \text{for } (x,y) \in \Omega. \tag{4.7}$$

The final metric is given by:

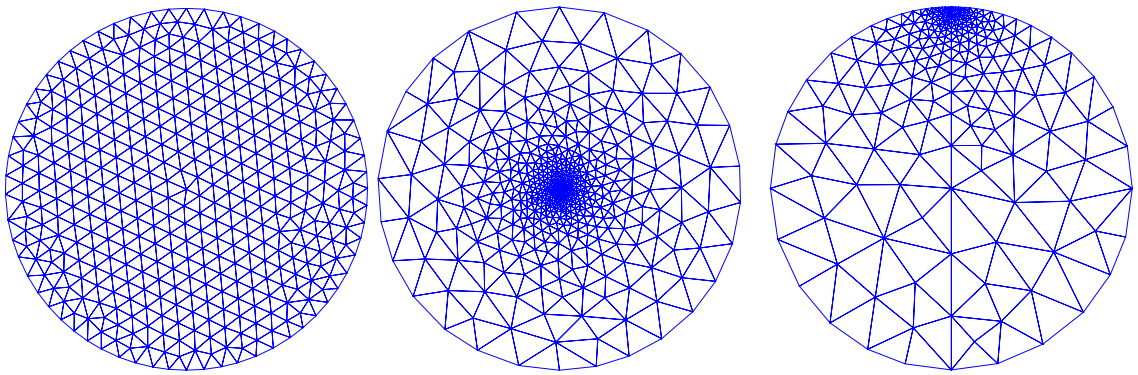
$$h(x,y) = h_{\text{disc}}(x,y) + \frac{h}{10}, \tag{4.8}$$

where h is the mesh size of the initial mesh. The factor 10 ensures adequate refinement near the singularity, and the additive constant $\frac{h}{10}$ avoids excessively small elements.

The mesh refinement is performed using the MMG library [MMG22]. Example of mesh refinement are shown in Figure 4.2(b).

4.2 State of the Art

In this section, we review the current state of the art regarding the Laplacian problem with a Dirac distribution as the source term. We focus on both theoretical results and numerical methods related to this problem. We start by examining the theoretical results concerning the



(a) Initial mesh with a uniform discretization.

(b) Mesh adaptation around the Dirac ($\vec{X}_0 = (0,0)$ in the left panel and $\vec{X}_0 = (0,1)$ on the right).

Figure 4.2: Example of mesh refinement near the singularity and around the boundary of the domain, with $h_{\text{avg}} = 0.1$ and $\vec{X}_0 = (0,0)$, $(0,1)$ respectively.

Laplace equation with a Dirac delta distribution $\delta_{\vec{X}_0}$ located at $\vec{X}_0$ in Ω. The primary challenge here is dealing with the singularity introduced by the Dirac distribution.

To circumvent the difficulties posed by the singularity, one approach is to use a regularized Dirac distribution [Oje24]. For instance, a Gaussian approximation is defined as:

$$\varphi(\vec{X}) = \exp\left(-\frac{\|\vec{X} - \vec{X}_0\|^2}{2r^2}\right), \tag{4.9}$$

where r is a given radius. While this method introduces an additional parameter to the problem, it simplifies the treatment of the singularity. However, in this work, we focus on the non-regularized Dirac distribution to study the behavior of the solution directly in the presence of the singularity.

4.2.1 Error Estimates

We briefly recall in this section two theoretical results available in the literature concerning convergence of the finite element method (FEM) for problems involving a Dirac distribution.

Theorem 4.2.1 ([NS74])**.** *Let $\Omega_0 \subsetneq \Omega_1 \subsetneq \Omega$, where Ω_1 and Ω_0 do not contain the singularity. If the solution u_h is sufficiently regular in Ω_1, then the error in the $H^1(\Omega_0)$ norm converges at order k as follows:*

$$\|u - u_h\|_{H^1(\Omega_0)} \leqslant C(\Omega_0, \Omega_1, \Omega)\left(h^k \|u\|_{H^{k-1}(\Omega_1)} + \|u - u_h\|_{H^{-p}(\Omega_1)}\right), \tag{4.10}$$

where $C(\Omega_0, \Omega_1, \Omega)$ is a constant that depends on the domains.

This theorem indicates that the error behaves well in domains away from the singularity. Extending this result, [Ber+18] demonstrates quasi-optimal convergence in the H^1 norm for finite element methods:

Theorem 4.2.2 ([Ber+18])**.** *For a $\mathbb{P}_k$-finite element method on a domain $\Omega_0 \subsetneq \Omega$ that does not contain the singularity, the convergence rate is given by:*

$$\|u - u_h\|_{H^1(\Omega_0)} \leqslant C(\Omega_0, \Omega_1, \Omega) h^k \sqrt{|\log(h)|}. \tag{4.11}$$

These results, are demonstrated for Dirichlet boundary conditions, and require further investigation for other boundary conditions. The theoretical framework generally assumes that the solution is regular away from the singularity, and the Galerkin orthogonality remains valid. However, variations in boundary conditions might affect the error estimates.

4.2.2 Handling Singularity Near the Boundary

The earlier error estimates assume that the singularity is sufficiently far from the boundary. When the singularity approaches the boundary, the assumptions underlying the theorems may no longer hold.

We recall that [Ber+18] provides an error estimate for the situation where the distance from the singularity to the boundary is of the order of h^2:

$$\|u - u_h\|_{H^1(\Omega_0)} \leqslant C, \tag{4.12}$$

where C is a constant independent of h. In particular, note that this estimate does not allow concluding with a convergence result when the singularity is too close to the boundary.

4.2.3 Open questions under investigation

The current state-of-the-art raises several questions and areas for investigation in the next sections:

- **Sharpness of the Theoretical Results:** How sharp are the theoretical bounds? Does the choice of domain Ω_0 where the error is computed affect the accuracy of the results?
- **Singularity Position Relative to the Boundary:** How does the position of the singularity relative to the boundary impact the convergence rates and error estimates?
- **Interplay Between Multiple Singularities:** What are the effects of having multiple singularities on the solution and its approximation?

Addressing these questions will help refine our understanding of the problem and improve numerical methods for handling singularities in various boundary conditions.

4.3 Sharpness of the theoretical results

The Laplace problem (4.2) is solved over the domain Ω defined as the unit disk

$$\Omega = \{\vec{X} \in \mathbb{R}^2 \,:\, \|\vec{X}\|_2 < 1\}. \tag{4.13}$$

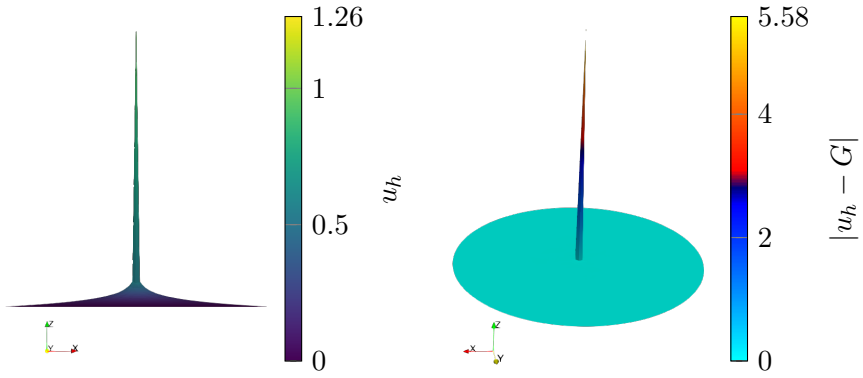
We present in Figure 4.3(a) the numerical solution u_h computed for the Dirichlet problem, with $\vec{X}_0 = (0, 0)$. Figure 4.3(b) shows the numerical error field $|u_h - G|$. This plot shows that the error is mainly concentrated near the singularity. Following [Ber+18] we compute the error on a domain "far" from the singularity, over the domain

$$\Omega_0 = \{\vec{X} \in \Omega \,:\, \|\vec{X} - \vec{X}_0\|_2 > r\}, \tag{4.14}$$

where r is a given radius: the error is computed over the domain Ω where a small disk around the discontinuity is removed. The domain Ω_0 is drawn in green on Figure 4.1. A more in-depth discussion about the choice of r is given in Section 4.3.2.

4.3.1 Convergence for various mesh sizes

It has been established in previous studies that, away from the singularity, the convergence rate of the finite element method matches that of the standard case: specifically, the method converges at an order of $k + 1$ for the L^2 error [KW14] and at an order of k for the H^1 error [Ber+18], when $\mathbb{P}_k$ finite elements are utilized.



(a) Numerical solution u_h.

(b) Pointwise error on the numerical solution u_h compared to the exact solution G.

Figure 4.3: Numerical solution (left) and pointwise error (right) compared to the exact solution G. In both figures, the solution is warped on the z-axis to represent the value of the plotted field.

We compare our results with those obtained in [Lac16; Ber+18]. The results from the literature are plotted alongside our results in Figure 4.4, and we observe a strong agreement: the convergence rates for both L^2 and H^1 errors are consistent, and our simulations demonstrate a more accurate approximation.

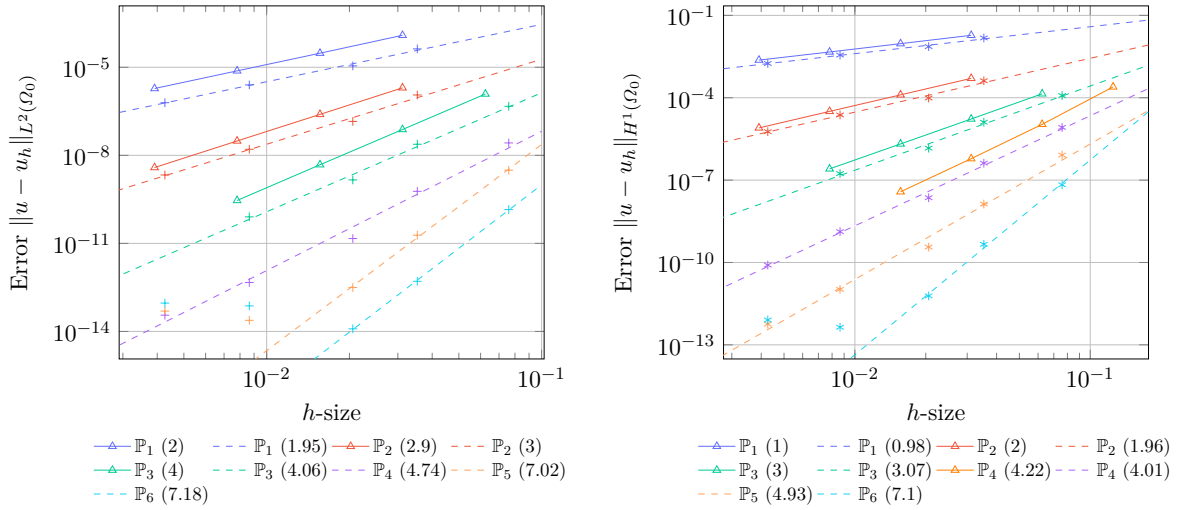
It is important to note that the same domain, Ω_0, with $r = 0.2$, is used for the computation, although the meshes differ. For high orders of discretization, machine precision limits are reached, which accounts for the observed plateau in the error for high mesh refinement. These limitations are not factored into the computation of the convergence rate.

4.3.2 Error computational domain influence

We note from Figure 4.3(b) that the error on the solution computed is concentrated in a small region around the singularity. We introduced the domain Ω_0 to compute the error, which is defined as $\Omega_0 = \{\vec{X} \in \Omega \, : \, \|\vec{X} - \vec{X}_0\|_2 > r\}$. To the best of our knowledge, no theoretical result that provides insights into the choice of the radius r has been established. Nevertheless, in [KW14], the authors performed a convergence study with various values of r. The theoretical results they presented hold only if $r > h$. More precisely, in theoretical result of [Ber+18] presented in Theorem 4.2.2, the value of r intervenes in the constant $C(\Omega_0, \Omega_1, \Omega)$, and this constant increases when r gets close to 0.

We consider the Dirichlet problem, with the discontinuity located at $\vec{X}_0 = (0, 0)$. We measure the error for various values of r, going from the mesh size h to 0.1. The results are presented in Figure 4.5, where the study has been performed for two mesh discretization sizes: $h = 0.0125$ and $h = 0.00625$. Vertical lines are drawn to show the value of h and $2h$ for both meshes considered. The plot is presented for the L^2 error, but a similar behavior is observed for the H^1 error.

The first observation is that for a higher discretization order, the error computed is smaller than for a lower one for the same value of r, meaning that the error, is concentrated on the singularity. Moreover, we observe that when r gets higher, as the number of elements where the error is computed, the error decreases faster for a higher order of discretization, reaching the machine precision for $r \approx 0.1$ with the $\mathbb{P}_6$ discretization. Finally, we notice that for $r \in [h, 2h]$, the error stagnates on a plateau, and then decreases, independently of the discretization order. This bound of $2h$ numerically found give a first insight on the choice of r, while theorems are not available for this choice.



(a) Error L^2. (b) Error H^1.

Figure 4.4: Convergence of the error for the Dirichlet problem, for various orders of discretization. Comparison of the results from [Lac16; Ber+18] (plain lines) with the present word (dashed). The number in parentheses is the slope of the linear regression from the plot data. Remark that for high-order discretization, the points reaching a plateau at machine precision are not taken into account in the regression. The error is computed over the domain Ω_0 with $r = 0.2$.

These results motivate our choice of taking $r = 0.1$ for the computation of the error: the error is satisfactory enough for all the discretization considered, and the error is computed over a large enough domain to capture the behavior of the solution.

Remark 4.3.1. When choosing $r = 0$ (*i.e.*, $\Omega_0 = \Omega$), the computed H^1 error remains constant across different mesh sizes, regardless of the discretization order. For the L^2 error, we observe a convergence rate of 1 for all discretization orders, as illustrated in Figure 4.6. These findings support the theoretical results presented in [KW14; Ber+18], which state that the error must be evaluated over a domain excluding the singularity.

4.4 Numerical experiments

4.4.1 Influence of Boundary located Dirac Delta

Convergence for various positions of the Dirac source The primary result of [Ber+18] regarding the convergence of the finite element method assumes that the discontinuity is sufficiently far from the boundary of the domain. Specifically, the hypothesis requires that there exists a small ball $B_\varepsilon \subset T_0$, where T_0 is the triangle of the mesh containing the point $\vec{X}_0$. However, in specific applications, the point of interest may be located close or on the boundary of the domain: for example, as the measures of temperature of the eyeball are mostly located at the level of the cornea (see Section 1.1), the points located on the boundary play an important role in the sensitivity analysis presented in Chapter 5.

In this section, we therefore analyze the behavior of the error as the Dirac source is moved within the domain Ω, using a fixed mesh size. We specifically consider positions along the segment $[(0, 0), (0, 1)]$. The results are illustrated in Figure 4.7, where the Robin problem is examined. While [Ber+18] focuses solely on the Dirichlet problem, we observe similar behavior in our results for the Robin problem.

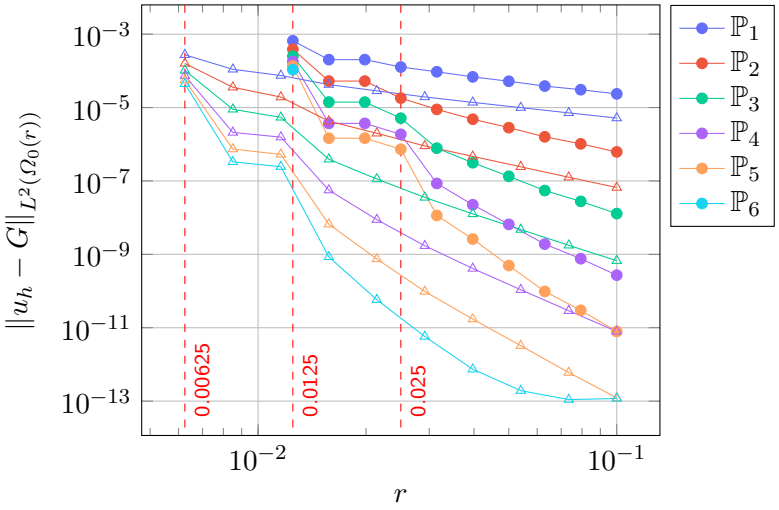


Figure 4.5: Evolution of the error over the domain Ω_0, depending on the radius r used for domain Ω_0. Two mesh sizes h are used: $h = 0.025$ (dot plot) and $h = 0.00625$ (triangle). The Dirichlet problem is considered. The radius varies from h to 0.1.

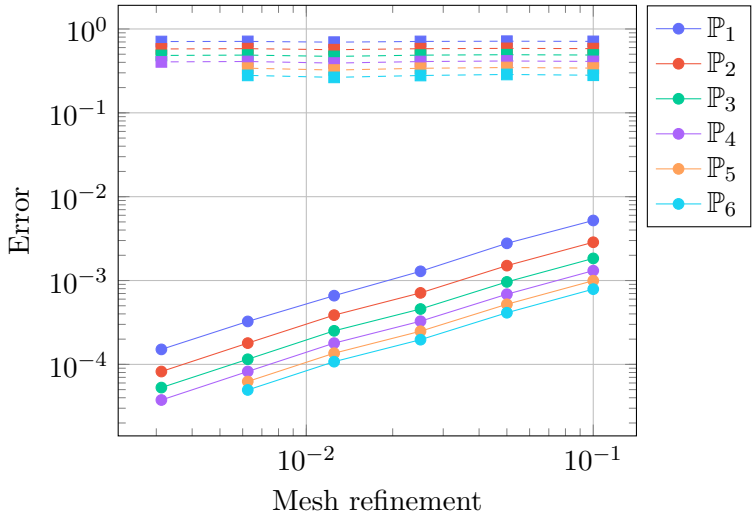
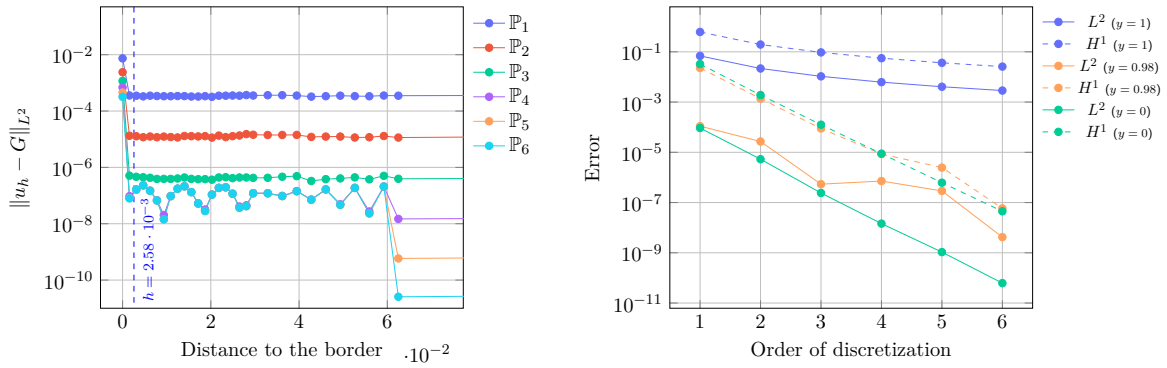


Figure 4.6: Convergence of the error $H^1(\Omega_0)$ (dashed) and $L^2(\Omega_0)$ (plain lines) for the Dirichlet problem, for various orders of discretization, when $r = 0$.



(a) Evolution of the error L^2 for $\mathrm{d}(\vec{X}_0, \partial\Omega) \in [0, 1]$. The vertical line shows the size of the mesh around the point $\vec{X}_0$.

(b) Evolution of the error against the order of discretization, for various y positions of the Dirac source.

Figure 4.7: Evolution of the error over the domain Ω_0, dependence on the distance of the discontinuity to the border of the domain, for the Robin problem.

Figure 4.7(a) shows how the error evolves as the point $\vec{X}_0$ approaches the boundary $\partial\Omega$. When the point is sufficiently far from the boundary, the error remains constant across all discretization orders considered. However, as the point nears the boundary, for higher orders of discretization, we observe a plateau around 10^{-7}. When the distance becomes smaller than the mesh size h, the error increases sharply to approximately 10^{-3}. Despite this, higher discretization orders still yield smaller errors.

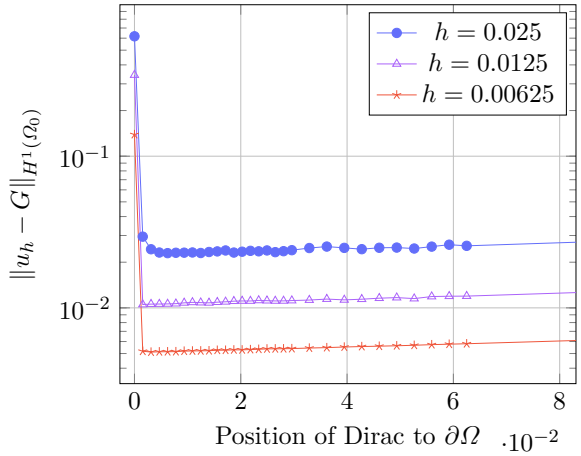
Figure 4.7(b) presents the error for various discretization orders when the Dirac source is fixed, considering both L^2 and H^1 errors. The error is 25 times smaller for $\mathbb{P}_6$ discretization compared to $\mathbb{P}_1$. The figure also includes an analysis of different positions of the Dirac on the segment. When $y = 0$, the higher-order discretization achieves an improvement of 6 orders of magnitude. However, if the Dirac is positioned close to the boundary ($y = 0.98$), there is a slight degradation in convergence for higher orders.

From the previous results, we can draw the following conclusions: *(i)* Even when the Dirac source is positioned exactly on the boundary of the domain, higher-order discretization improves precision, as shown in Figure 4.8. *(ii)* As the Dirac source approaches the boundary, the benefit of higher-order discretization diminishes, though it still provides some improvement. *(iii)* When the Dirac source is very close to the boundary, the error reaches a plateau, indicating limited further reduction in precision.

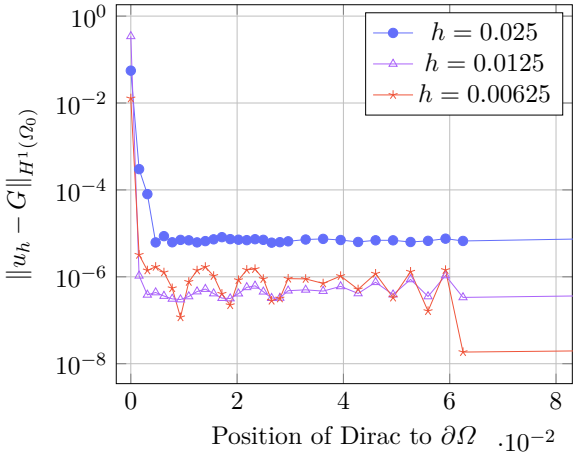
Convergence for various mesh sizes In the preceding part, we analyzed the behavior of the error for a fixed mesh size while varying the position of the Dirac source within the domain. We now extend this analysis to examine how the error evolves with different mesh sizes. The results, depicted in Figure 4.8 (panels (a) and (b)) for the Robin problem, reveal that the error exhibits oscillations as the Dirac source moves closer to the domain boundary. At this point, we cannot explain the origin of these oscillations, and further investigations are required.

Results presented in Figure 4.8 show that with a low order of discretization, the error plateaus until the discontinuity is positioned on the boundary. In contrast, with high-order discretization, the error exhibits oscillations, and further mesh refinement becomes less effective, although the error remains substantially smaller compared to lower-order discretization.

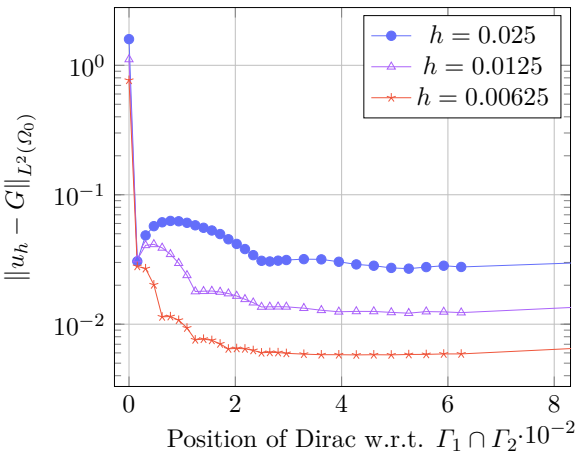
For mixed boundary conditions, similar trends are observed (Dirichlet-Neumann in panel (c) and Robin-Neumann in panel (d)). However, when Dirichlet is one of the boundary conditions, an unusual feature is noted: the error tends to be higher with a finer mesh at high order of



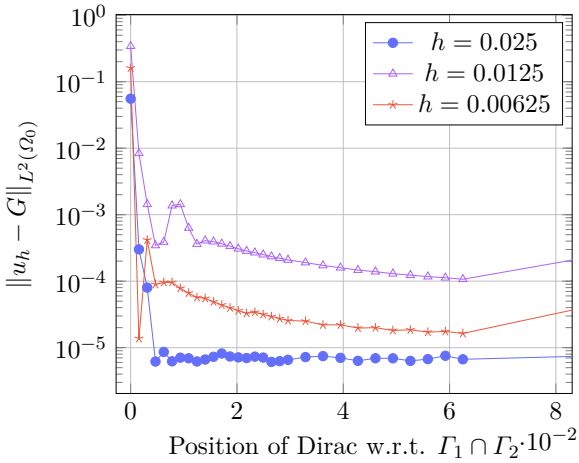
(a) Discretization $\mathbb{P}_1$, for the Robin problem.



(b) Discretization $\mathbb{P}_4$, for the Robin problem.



(c) Discretization $\mathbb{P}_1$ for mixed Dirichlet-Neumann problem.



(d) Discretization $\mathbb{P}_4$ for mixed Robin-Neumann problem.

Figure 4.8: Comparison of the error for various mesh sizes, for the Robin problem, Dirichlet-Neumann and Robin-Neumann problem.

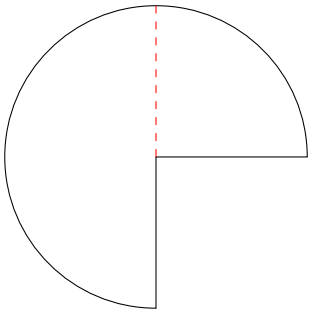


Figure 4.9: Circular domain with entering corner. The Dirac source is located on the red dashed line.

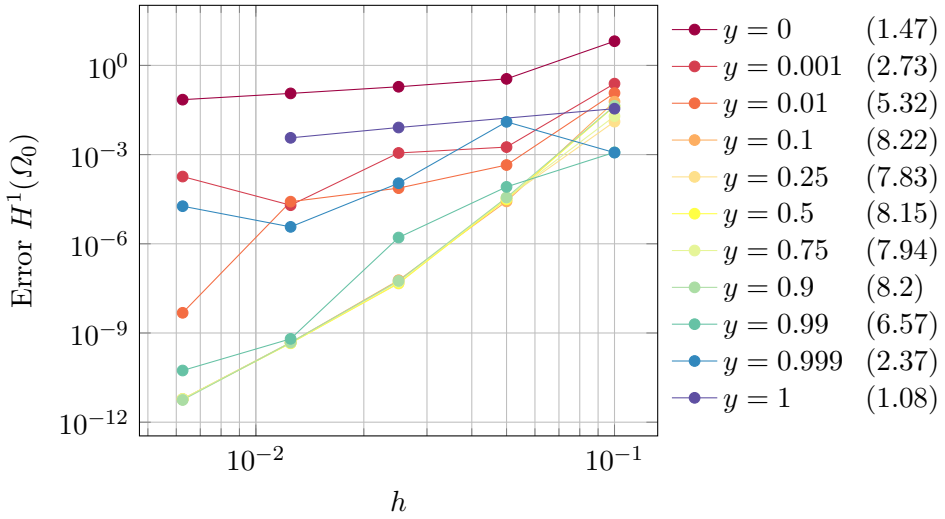


Figure 4.10: Convergence curve with $\mathbb{P}_6$ discretization for Dirac position going from the internal corner ($y = 0$) to the border of the domain ($y = 1$). The number displayed in parentheses is the slope of the linear regression from the plot data.

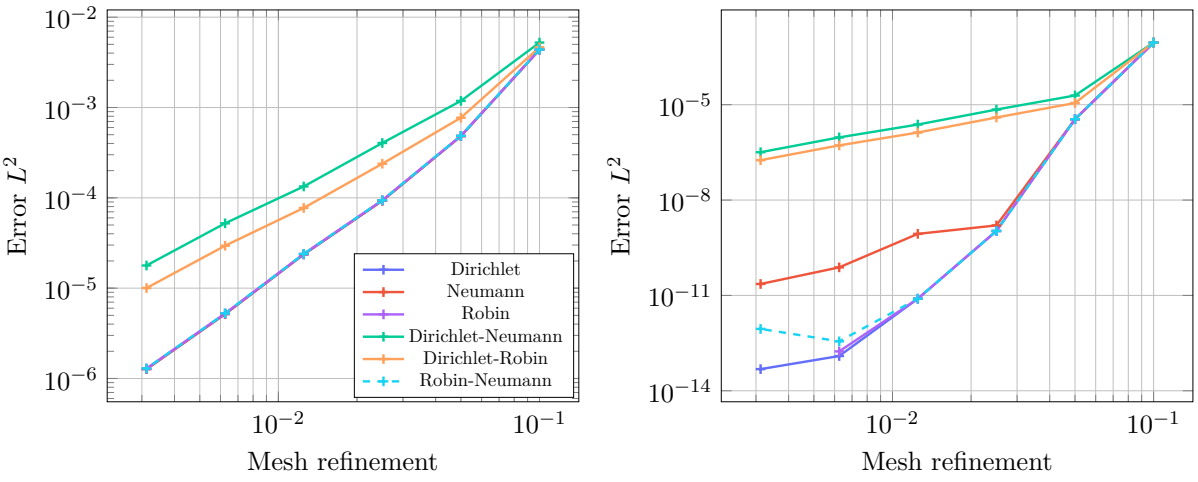
discretization. At this stage, we are not able to identify the source of this behavior, and further investigation are in progress.

4.4.2 Interplay between Dirac sources and geometric singularities

In Section 4.3.1, we noted that the convergence rate of the finite element method for the Dirac problem remains consistent with that of the standard case. This section examines how the position of the discontinuity influences the convergence rate. Specifically, we analyze the Dirichlet problem by varying the location of the Dirac source within the domain Ω, considering two distinct cases: *(i)* the circular domain illustrated in Figure 4.1, and *(ii)* the circular domain with a reentrant corner, as depicted in Figure 4.9.

The results presented in Figure 4.10 reveal several key insights into the impact of the discontinuity's location on the convergence rates. When the discontinuity is placed on the border of the domain, the observed convergence rates are notably poorer. This issue is most pronounced when the discontinuity is positioned at the corner of the domain, where both the slope and the error are at their worst.

A similar pattern is observed across other problem setups. For example, in the case of a full disk domain, the optimal results are achieved when the discontinuity is located at the center of the disk. This observation aligns with theoretical expectations, which suggest that for each



(a) $\mathbb{P}_1$ discretization.

(b) $\mathbb{P}_5$ discretization.

Figure 4.11: Evolution of the error for various boundary conditions considered, for various discretization orders.

discretization degree, there must be at least one layer of elements between the discontinuity and the domain boundary.

The theoretical framework [NS74; Ber+18] supports this by indicating that error control can be achieved if there exists a domain Ω_1 such that $\Omega_0 \subseteq \Omega_1 \subseteq \Omega$ (where Ω_0 is the domain over which the error is computed). The error estimate is given by:

$$\|e\|_{H^1(\Omega_0)} \leqslant h^k \|u\|_{H^{k+1}(\Omega_1)} + \|e\|_{H^{-p}(\Omega_1)} .$$

If Ω is sufficiently regular, we have:

$$\|e\|_{H^{-p}(\Omega_1)} \leqslant h^{p+1} \|e\|_{H^1(\Omega)} .$$

When the Dirac source is positioned on the border, theoretical predictions suggest a convergence rate of 4/3. In practice, the observed rates are somewhat lower, with values of 1.47 for the corner and 1.08 for the border, indicating a deviation from the theoretical expectations.

4.4.3 Impact of the boundary conditions type

In the previous section, we presented results for a single type of boundary condition at each time. However, it is important to note that similar behavior has been observed across all the problems considered. We present in Figure 4.11 the evolution of the L^2 error for various boundary conditions, for different discretization orders. We note that at small orders of discretization, a similar error is observed for all types of boundary conditions, except for the coupled one where a Dirichlet Boundary conditions is considered. This can be explained by the fact that on the interface between the two boundary conditions ($\Gamma_1 \cap \Gamma_2$), the solution is not smooth, and the error is not well controlled. But still, we observe a convergence of the error when the mesh is refined.

A similar behavior is observed at high orders of discretization, even though the difference is larger in this case.

4.5 Conclusion and outlook

In this chapter, we investigated the performance of finite element methods (FEM) for solving the Laplacian problem with a singularity at the origin in a unit disk domain. Our numerical results reveal important insights into the behavior of FEM near singularities and the associated convergence properties.

The theoretical analysis suggests that the FEM might face challenges due to the nature of the singularity, especially as the polynomial degree increases. This pessimistic outlook stems from the type of proof used, which predicts a more significant degradation in accuracy close to the singularity, necessitating the computation of the error farther away as the polynomial order increases. However, our numerical experiments present a more optimistic picture. Specifically, we observed that the solution's accuracy remains high in a region around the singularity with a radius greater than $2h$, provided we stay sufficiently away from the domain boundary.

Close to the boundary, we do encounter very slow convergence for all polynomial orders. Nonetheless, the constants associated with the convergence rates improve with higher polynomial orders, indicating that while convergence is slow near the boundary, higher-order elements can still provide better approximation quality. These findings align with the practical observations made in similar studies, such as those by [Ber+18] and [Lac16], but highlight the nuances that theoretical predictions alone may not fully capture.

Several avenues for future research emerge from this study:

(i) **Refinement of theoretical analysis:** Revisiting the theoretical framework to account for the observed discrepancy between theory and practice could offer more accurate predictions of FEM performance near singularities. Developing new proofs or refining existing ones to better match numerical findings would be valuable.

(ii) **Adaptive mesh refinement:** Implementing adaptive mesh refinement strategies to focus on regions near the singularity and boundary could enhance accuracy and efficiency. Such techniques are critical to dynamically adjust the mesh to better capture the solution's behavior where the error is most significant.

(iii) **Analysis of the effect of the boundary conditions:** Detailed examination of the boundary effects on convergence could lead to more effective strategies for mitigating slow convergence. Understanding how boundary proximity influences solution accuracy can inform better mesh design and solution strategies.

Another interesting direction for future research is to explore the case where several singularities are present in the domain, as presented in [KW14].

In conclusion, while theoretical predictions may suggest a more challenging scenario for FEM in the presence of singularities, our numerical results demonstrate that practical performance can be significantly better than expected. Continued research in refining theoretical models and exploring advanced numerical techniques will further enhance the efficacy of FEM for solving complex problems involving singularities.

Chapter 5

Sensitivity Analysis

In the models introduced in Chapter 1, numerous parameters, both biomechanical and geometrical, are involved. We concentrate here on biomechanical parameters, in a large range that potentially include extreme conditions. The variation of these parameters might have a significant influence on the results. Therefore, in order to quantify their impact, we set up a framework to perform a forward uncertainty quantification study, complemented by a sensitivity analysis. Deterministic sensitivity analysis has already been documented in [Sco88; NO06; NO07; Li+10], using various numerical methods. In this work, we reproduce and extend these results, to incorporate the effect of blood flow, as suggested for instance in [Sco88]. We also run a global sensitivity analysis, that accounts for stochastic effects, and discriminate among different factors by means of Sobol' indices [Sob93].

The present chapter is organized as follows: Section 5.1 provides an introduction to sensitivity analysis and uncertainty quantification, while Section 5.2 and Section 5.3 detail the deterministic and stochastic sensitivity analyses, respectively.

5.1 Introduction to sensitivity analysis

To understand the model and the impact of uncertainties on the output, two main tools are commonly employed: Sensitivity Analysis and Uncertainty Analysis.

Sensitivity Analysis (SA) [Sal02; Sal+21b] is the study of how the uncertainty in the output of a model can be apportioned to different sources of uncertainty in the model input. It is used for various purposes, including quantifying the contributions of model inputs (or groups of them) to the output and identifying the input factors that contribute the most to model uncertainty. On the other hand, *Uncertainty Analysis* (UA), characterizes the uncertainty in model predictions without necessarily identifying which assumptions are primarily responsible. These assumptions may include the choice of model structure, statistical distributions of inputs, independence of

parameters, simplifications of boundary conditions, and numerical approximations, all of which can significantly influence the reliability of the analysis [Sal+21b]. The process of UA typically involves running the model multiple times and extracting relevant statistical data such as the mean, standard deviation, and other related metrics. Uncertainty analysis can be further divided into *forward* and *backward* types, where forward UQ assesses how input uncertainties propagate to outputs, and backward UQ determines the input uncertainties required to achieve a specific level of output uncertainty.

According to [Sal+19], UA should ideally be conducted before Sensitivity Analysis. This sequential approach ensures that the uncertainties are well characterized before their sources are investigated.

However, some problems are commonly observed in the practical application of SA and UA. One such issue is the misuse of the terms UA and SA, leading to confusion and misinterpretation of results. Additionally, SA methods that rely on local techniques can be invalid for non-linear models, which require more robust methods to ensure accurate and meaningful analysis. These challenges are explored in greater detail in [Sal+19].

In [Raz+21], the authors emphasize that SA is on the path to becoming an essential component of mathematical modeling, outlining areas of research that need further attention, and among them, they address the computational cost of SA.

The *uncertainty quantification* (UQ) allows quantifying the uncertainty of the model parameters on various outputs. In the present work, we focus on *forward* UQ, that is, we want to quantify the uncertainty and the sensitivity of the output of the model, given the uncertainty of the input parameters. More precisely, two studies are performed:

(i) an *uncertainty propagation*, to understand how the uncertainties of the inputs of the model are propagated to the output *via* the computational model, and

(ii) a *sensitivity analysis* (SA) to assess the impact of varying selected parameters on several outputs of interest, namely temperature at specific locations in the eye, that are detailed in Figure 1.5.

5.1.1 Uncertainty propagation framework

Given a computational model, the uncertainties on the inputs are propagated to the output, as depicted in Figure 5.1. However, we do not have any information on this propagation, and we need to quantify the level of confidence on the output.

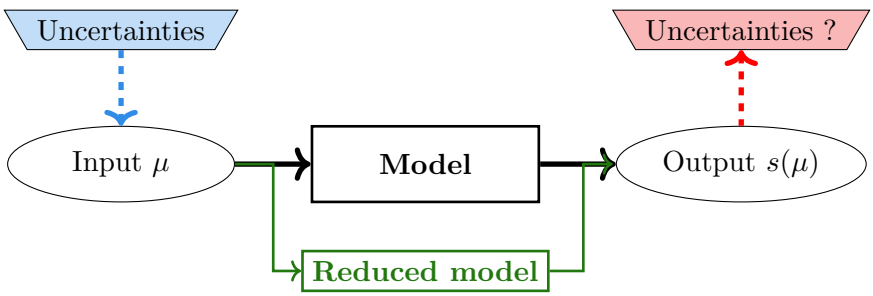


Figure 5.1: Uncertainties propagation framework.

We consider an output quantity Y depending on a set of input parameters $\mu \in D^{\mu}$, and we estimate the sensitivity of Y to each parameter μ_i for $i \in [\![1, d]\!]$, where d is the dimension of the parametric space. To this end, we compute the *Sobol' sensitivity indices* introduced in [Sob93; Sob01] as follows. We assume that each component μ_i of μ follows a random variable X_i, independent of the others. The first-order indices are defined as:

$$S_i := \frac{\text{var}\left(\mathbb{E}\left[Y|X_i\right]\right)}{\text{var}(Y)}, \tag{5.1}$$

where $\text{var}(Y)$ corresponds to the variance of Y including the eventual non-linearity effect of the coefficient on the output, and $\text{var}\left(\mathbb{E}\left[Y|X_i\right]\right)$ is the variance of the conditional expectation of Y given X_i, corresponding to the first order effect of the parameter μ_i on the output: if the parameter modeled by the distribution X_j has a great impact on the output Y, then $\mathbb{E}\left[Y|X_j\right]$ will vary has well, and so its variance.

We also define the *total Sobol' index*:

$$S_i^{\text{tot}} := \frac{\text{var}\left(\mathbb{E}\left[Y\middle|X_{(-i)}\right]\right)}{\text{var}(Y)} = 1 - S_{-i}, \tag{5.2}$$

where $X_{(-i)} = (X_1, \cdots, X_{i-1}, X_{i+1}, \cdots, X_d)$ is the set of parameters without the parameter X_i, and S_{-i} is the sum of the indices where X_i is not present.

The study of such indices is very useful to understand the impact of the parameters on the output, and to identify the most influential parameters. However, some drawbacks are observed, such as the high computational cost of the method. Indeed, the computation of the Sobol' indices requires numerous model evaluations, which can be prohibitive for complex models such as the HF model described in Chapter 1. To overcome this issue, we have implemented in Section 3.1 a reduced model thanks to the Reduced Basis Method, which allows a significant gain in terms of computational cost, while maintaining certified and accurate results. This process is highlighted in green in Figure 5.1.

5.1.2 Deterministic versus stochastic sensitivity analysis

In the present manuscript, the SA is conducted in two different approaches. First, to recapitulate findings from the literature, we performed a *deterministic sensitivity analysis* (DSA), where for each simulation, only one parameter is allowed to vary in a given range, whereas the others are fixed to their baseline value. In a second stage, we extended the SA to a stochastic framework, where each selected parameter follows a given random distribution and the impact on the quantity of interest is assessed via sensitivity indices. We denote as *Stochastic Sensitivity Analysis* (SSA) this second approach. The advantages of the latter are the global perspective provided by this method and its ability to capture high-order interactions among several input parameters.

The integration of deterministic and stochastic methods has emerged as a highly effective approach in the field of uncertainty quantification. This combined methodology has been successfully applied in various recent studies. For example, [Dod+14] investigates the impact of uncertainties on the distribution of the electromagnetic field in ocular tissue. Similarly, broader applications include uncertainty analysis in the human head [Šuš+22; Šuš+21], as well as in the cardiovascular domain [CQR13; BDL17]. A SA on the component simulating the fluid flows in the eye was also developed in [PSS21].

While Sobol' indices are effective in measuring parameter impact and interactions, the complexity and the significant computational time of our model are very challenging. To overcome this, we adopt the certified reduced basis method [Pru+02; QMN16] to obtain a reduced model, maintaining its 3D nature while significantly reducing computational demands, as detailed in Section 3.1. This method aligns with the paradigm observed in patient-specific mathematical models applied to biomedical problems, ensuring a comprehensive approach involving data integration, model derivation, numerical solving, validation, and uncertainty quantification, as seen in mature research fields like cardiovascular simulations or cerebral hemodynamics. In

ophthalmology, a similar paradigm is imperative due to the richness and heterogeneity of available data, requiring innovative approaches for diagnosis and monitoring.

5.2 Deterministic sensitivity analysis

Our initial investigation of the impact of varying selected parameters is conducted through a *deterministic sensitivity analysis* (DSA). Specifically, we choose a parameter among the ones defined in Table 1.1, and we set the other to their baseline value. Next, we vary the selected parameter among pre-defined values and compute the outputs of the high fidelity model. Similar studies were performed in [Sco88; NO06; Li+10]. We gather in the present study information about several parameters of interest for the heat transfer model from these studies, namely baseline values and ranges. These variations correspond not only to physiological conditions but also include some extreme situations. We postpone a more in-depth discussion on this topic to Section 5.3, where the random distributions characterizing these parameters are set up. Note that in this case, we do not need to use the reduced model, since only a relatively small number of simulations is required. The Python code that performs this study using the toolbox `heat` of Feel++ is presented in Section 7.5.1. The full-length code is available in the GitHub repository [SPS24a].

In Figure 5.2, we present the DSA results for the parameters $\mu = \{E, h_{\text{amb}}, h_{\text{bl}}, k_{\text{lens}}, T_{\text{amb}}, T_{\text{bl}}\}$, focusing on the temperature at point O, located at the surface of the cornea. The plain-line curves correspond to the results reported in the literature, which we compare with the results of our simulations; the vertical dashed line corresponds to the baseline value for each parameter. The results are in very good agreement with previous findings and show that temperature at the level of the cornea is strongly influenced by h_{amb}, T_{amb}, E, and T_{bl}, whereas the influence of h_{bl} and k_{lens} is less significant. For instance, high air conductivity can result in a temperature 7 K lower than the baseline value, while the difference obtained for h_{bl} and k_{lens} in the computed temperature is at most of 1 K.

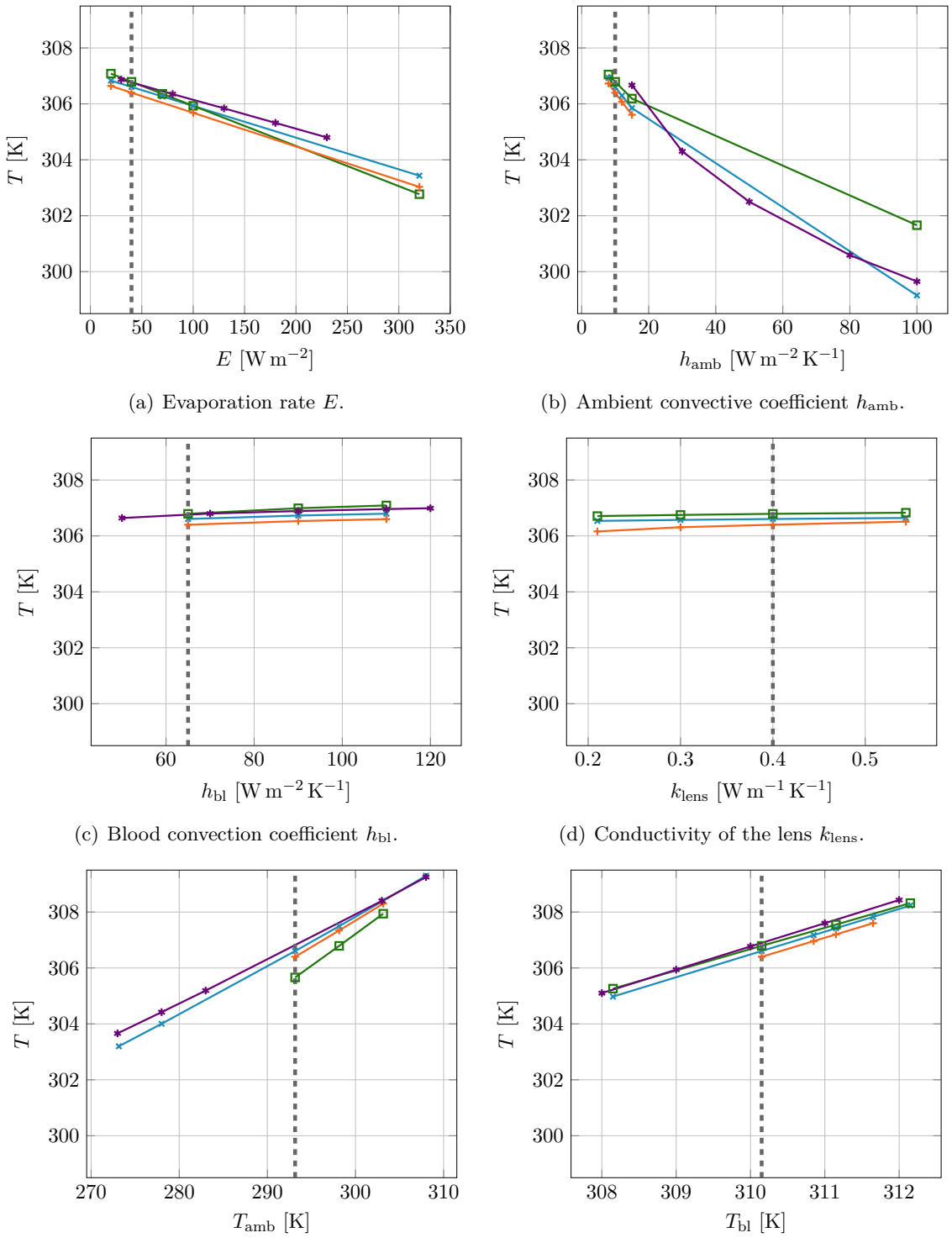
5.3 Stochastic sensitivity analysis

Following the DSA, a *stochastic sensitivity analysis* (SSA) is conducted to account for the uncertainty and variability in the input parameters, using Sobol' indices. By incorporating probability distributions, this approach provides a more comprehensive understanding of how random fluctuations can impact the outcomes of the model.

To compute the Sobol' indices introduced in Equations (5.1) and (5.2), we use an algorithm of functional chaos, implemented in the library OpenTURNS [Bau+16] provided by the class `FunctionalChaosAlgorithm` [Ope], using a bootstrap method for the confidence intervals. More details about the method can be found in [MS18] as well as in the documentation of the library. Note that this method require a huge number of evaluations of the model, that is why the reduced model is crucial in this context. The code implemented to compute the Sobol' indices is presented in Section 7.5.2, and strongly relies on the framework of the Reduced Basis Method previously implemented.

5.3.1 Choice of the distributions

We discuss in this section the prior distributions for each parameter. This choice is crucial [Rod+19] and needs to be done carefully, in light of the quantitative and qualitative information available data. The sources of uncertainties are diverse, but initially, based on the deterministic analysis present in the literature, we proposed this first step of global sensitivity analysis, which incorporates stochastic effects and identifies interactions.



(a) Evaporation rate E.

(b) Ambient convective coefficient h_{amb}.

(c) Blood convection coefficient h_{bl}.

(d) Conductivity of the lens k_{lens}.

(e) Ambient temperature T_{amb}.

(f) Blood temperature T_{bl}.

Figure 5.2: Results of the DSA for the 6 parameters studied, among previous studies from the literature (markers × $\mathcal{E}_{\text{NL}}(\mu)$, □ [NO06], + [Sco88], ∗ [Li+10]). The vertical dashed line corresponds to the baseline value of the parameter.

Each parameter does not depend on the others, resulting in a family of 6 random independent variables. Figure 5.3 shows the probability density function (PDF) of distributions of the parameters, associated with the baseline values (see Table 1.1), where the parameters used in the literature for the deterministic sensitivity analysis are represented with a vertical line. We present hereafter some details on how the random distributions were constructed.

Definition 5.3.1 (Log-Normal distribution)**.** Let $\mu \in \mathbb{R}$, and $\sigma > 0$. We say that a distribution X has Log-Normal distribution with parameters μ and σ^2 if the random variable $Y := \log(X)$ follows a normal distribution $\mathcal{N}(\mu, \sigma^2)$. We denote $X \sim \log\text{-}\mathcal{N}(\mu, \sigma^2)$. The support of this law is $[0, +\infty[$. Let $\gamma \in \mathbb{R}$. We define the log-Normal distribution of parameters (μ, σ^2, γ) the translation of a $\log\text{-}\mathcal{N}(\mu, \sigma^2)$, over $[\gamma, +\infty[$.

The probability density function of $X \sim \log\text{-}\mathcal{N}(\mu, \sigma, \gamma)$ is defined, for $x \in [\gamma, +\infty[$, as

$$f_X(x) = \frac{1}{\sqrt{2\pi}\sigma(x-\gamma)} \exp\left(-\frac{1}{2}\left(\frac{\log(x-\gamma)-\mu}{\sigma}\right)^2\right). \tag{5.3}$$

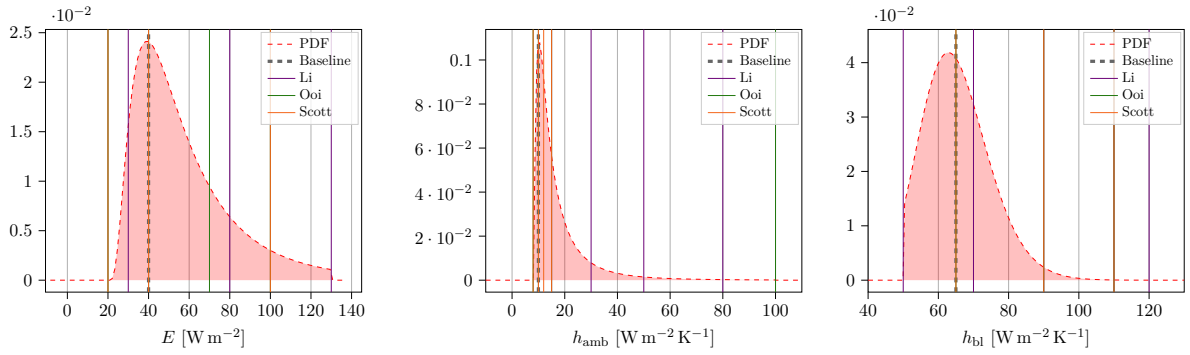
In particular, we have $\mathbb{E}[X] = \exp\left(\mu + \frac{\sigma^2}{2} + \gamma\right)$, and $\mathrm{var}[X] = \exp\left(2\mu + \sigma^2\right)\left(\exp\left(\sigma^2 - 1\right)\right)$. A class in OpenTURNS, `OT::LogNormal`, implements this distribution.

- **Evaporation rate** E**:** According to [Sco88], the evaporation rate's range is from 40 to 100 $\mathrm{W\,m^{-2}}$, using data from literature [Adl53]. The value $E = 40\,\mathrm{W\,m^{-2}}$ is chosen as the baseline value. Some high values are also considered, to study the impact of important evaporation rates. The values used in the literature run from 20 to 320 $\mathrm{W\,m^{-2}}$. As this parameter varies by several orders of magnitude, we decided to use a log-normal distribution to represent it. More precisely we set $E \sim \log\text{-}\mathcal{N}(\mu_E, \sigma_E, \gamma_E)$, with $\sigma_E = 0.7$, $\mu_E = \log(40) - \frac{0.15^2}{2}$ and $\gamma_E = 20$, restricted to $[20, 130]$. The distribution is presented in Figure 5.3(a). This choice of the distribution leads to a mean value of $\overline{E} = 55.8\,\mathrm{W\,m^{-2}}$.

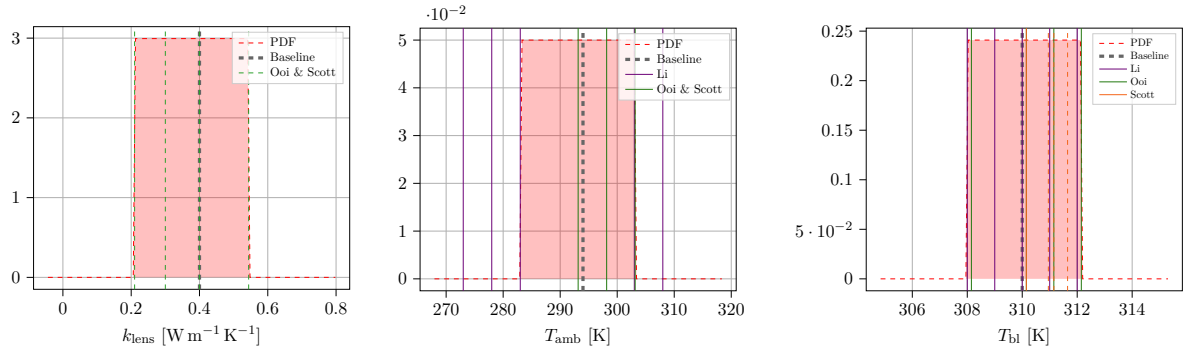
- **Ambient air convection coefficient** h_{amb}**:** In [Sco88], the sole value given for the ambient air convection coefficient is $10\,\mathrm{W\,m^{-2}\,K^{-1}}$, and similar values are used to run the DSA, from 8 to 15 $\mathrm{W\,m^{-2}\,K^{-1}}$. Other results in the literature corroborate this value: [Kos+13, Table 12.2] reports a range of 2.5 to 25 $\mathrm{W\,m^{-2}\,K^{-1}}$ for a free convection, and 10 to 500 $\mathrm{W\,m^{-2}\,K^{-1}}$ for a forced convection. [EL] proposes a range of 10 to 100 $\mathrm{W\,m^{-2}\,K^{-1}}$ for the air. In their DSA, [NO06] and [Li+10] use higher values of h_{amb}, up to 100 $\mathrm{W\,m^{-2}\,K^{-1}}$ to simulate a forced convection condition. As high values are not a common case, such a coefficient should not have a high frequency in the distribution. We chose to use a log-normal distribution: $h_{\mathrm{amb}} \sim \log\text{-}\mathcal{N}\left(\log(10) - \frac{1}{2}, 1, 8\right)$. In Remark 1.4.1, we discussed the linearization process of the model, inducing the usage of a fixed parameter h_{r} chosen to fit temperature in usual ambient room conditions, which leads to a restriction of the distribution to the interval $[8, 100]$ $\mathrm{W\,m^{-2}\,K^{-1}}$. The distribution is presented in Figure 5.3(b). The mean value of the distribution is $\overline{h_{\mathrm{amb}}} = 17.6\,\mathrm{W\,m^{-2}\,K^{-1}}$.

- **Blood convection coefficient** h_{bl}**:** A control value of 65 $\mathrm{W\,m^{-2}\,K^{-1}}$, derived from experimental data [J 82] is provided in [Sco88]. For the DSA, the values used run from 50 to 120 $\mathrm{W\,m^{-2}\,K^{-1}}$. This leads us to the following assumption for the distribution of the parameter: $h_{\mathrm{bl}} \sim \log\text{-}\mathcal{N}\left(\log(65) - \frac{0.15^2}{2}, 0.15, 0\right)$, restricted over $[50, 120]$, see Figure 5.3(c). The mean of this distribution is $\overline{h_{\mathrm{bl}}} = 65.8\,\mathrm{W\,m^{-2}\,K^{-1}}$.

- **Lens conductivity** k_{lens}**:** This parameter is chosen among all the conductivities since the water content of the lens varies with aging [Sco88]. [Sco88] and [NO06] run the DSA with this parameter, using values from 0.21 to 0.544 $\mathrm{Wm^{-1}K^{-1}}$. As the range of values



(a) Distribution of E. (b) Distribution of h_{amb}. (c) Distribution of h_{bl}.



(d) Distribution of k_{lens}. (e) Distribution of T_{amb}. (f) Distribution of T_{bl}.

Figure 5.3: Distributions of the parameters. The vertical lines represent the values chosen in literature for the DSA.

is not very large, it seems reasonable to use a uniform distribution for this parameter: $k_{\text{lens}} \sim \mathcal{U}(0.21, 0.544)$, see Figure 5.3(d).

- **Ambient temperature T_{amb}:** The baseline value of this parameter is taken to a usual room temperature of 294.15 K (20 °C). The values used for the DSA vary from extreme conditions of 273.15 K (0 °C) to 308 K (35 °C). As these extreme values are not very common, we choose to restrict the values taken by T_{amb} from 283.15 K (10 °C) to 303.15 K (30 °C): $T_{\text{amb}} \sim \mathcal{U}(283.15, 303.15)$. The distribution is presented in Figure 5.3(e).

- **Blood temperature T_{bl}:** The temperature of human blood is commonly accepted to be 310 K (37 °C). For the DSA, cases of hypothermia and hyperthermia are considered, with a range from 308 K to 312.15 K (35 °C to 39 °C). We therefore take $T_{\text{bl}} \sim \mathcal{U}(308, 312.15)$, see Figure 5.3(f).

5.3.2 Uncertainty propagation

We analyze the distribution of outputs of interest from a random sample of the input parameters obtained from the distributions presented. We use 10,000 points, leading to 10,000 simulations. The computational cost of the high fidelity simulations becomes prohibitive in this case; therefore, we employ the reduced basis metamodel developed in Section 3.1. Figure 5.4 presents the distribution of three outputs, namely the mean of the temperature over the cornea T_{cornea}, and the temperature at points O and G respectively, which are located at the front and the back of the eyeball, respectively. Note that T_O and T_{cornea} display a Gaussian distribution,

	T_{cornea}	T_O	T_G
Mean	305.590082	303.185187	310.028526
Standard deviation	1.788358	2.457063	1.055978
Skewness	-0.483163	-0.536222	-0.087999
Kurtosis	0.563015	0.598157	-1.222492

Table 5.1: Statistics of the outputs.

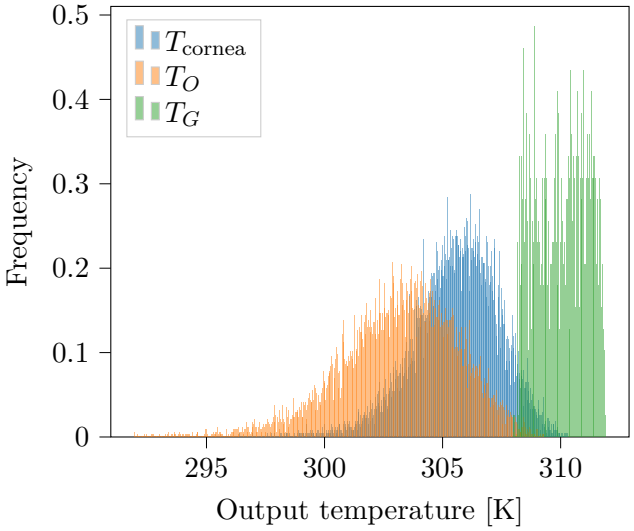


Figure 5.4: Distribution of the output, from the composed input distribution.

whereas T_G is more difficult to interpret, but might correspond to a uniform or bi-modal distribution.

In addition, we extracted the *Skewness* and *kurtosis*, statistical measures that provide insights into the shape and distribution of a dataset. They are useful in understanding the shape of data and can provide insights into the underlying characteristics of a dataset. However, it's important to note that these measures should be interpreted along with other descriptive statistics and visualizations to get a comprehensive understanding of the data.

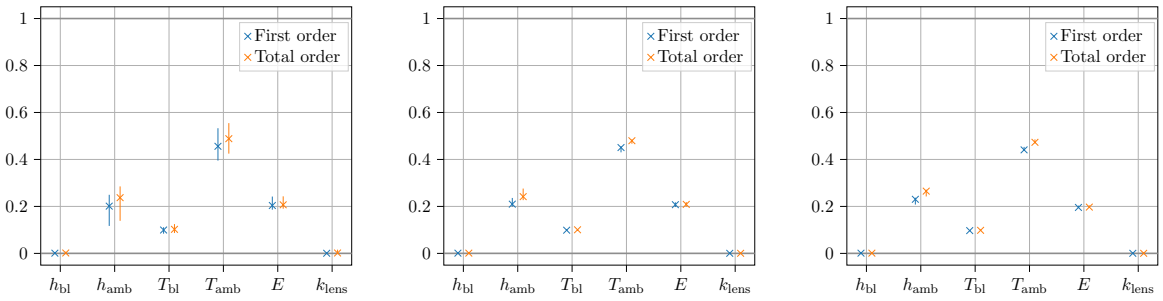
We provide in Table 5.1 results about mean values and standard deviation for the same quantities. We note that the mean values of T_O and T_{cornea} are of the same order of magnitude as the experimental data in the validation section (Section 2.2.5): the difference of temperature is about 2 K, and standard deviations are in the same ranges. The mean value of T_G is very close to results reported in Figure 2.10 from the literature with a small standard deviation.

5.3.3 Results of the Stochastic Sensitivity Analysis

Before going through the interpretation of the SA results via Sobol' indices, we present a convergence analysis we performed by varying the sampling size N_{param}. We report in Table 5.2 the maximal deviation of these indices, the time taken by the application to compute the 6 sets of Sobol' indices, and the predictivity factor Q_2.

The *predictivity factor* Q_2 for the polynomial chaos metamodel is defined as:

$$Q_2 := 1 - \frac{\sum_{l=1}^{N} \left(Y_l - \widehat{f}(X_l)\right)^2}{\text{var}(Y)}, \tag{5.4}$$



(a) $N_{\text{param}} = 60$. (b) $N_{\text{param}} = 100$. (c) $N_{\text{param}} = 200$.

Figure 5.5: Convergence of the Sobol' indices.

N_{param}	Max deviation	t_{exec}	Q_2
60	0.18102	0.756 09 s	0.999153
100	0.03698	1.996 51 s	0.992648
150	0.02969	2.837 43 s	0.99986
200	0.02923	4.160 46 s	0.998926
400	0.00739	8.367 01 s	0.999931
600	0.00496	15.7947 s	0.9998
1000	0.00248	22.364 s	0.999904

Table 5.2: Convergence of the Sobol' indices.

measuring how accurate the metamodel $\widehat{f}$ is at predicting the output Y from the input X. The closer Q_2 is to 1, the better the metamodel is. In the context of the Sobol' indices experiment, the metamodel $\widehat{f}$ is the polynomial chaos expansion of the output Y. The test of convergence is performed using the temperature on point O as the output. The convergence of Sobol' indices is reached for $N_{\text{param}} = 200$ with a 10^{-2} accuracy, which is a threshold used in the sequel.

Figure 5.6 displays the results of the Sobol analysis for different outputs of interest. Recall that Figure 1.5 shows where the points are in the eye.

In the deterministic sensitivity analysis conducted in Section 5.2, the impact of the variation of a sole parameter on the temperature at point O was studied. Using Sobol' indices, we are now able to measure the impact when all of them are varying. The results of Sobol analysis at point O presented in Figure 5.6(a) are in very good agreement with the deterministic findings: the temperature at the level of the cornea is strongly influenced by external factors such as h_{amb}, as well subject-specific parameters such as T_{amb}, E, and T_{bl}, Moreover, it is minimally influenced by the lens conductivity k_{lens} and the blood convection coefficient h_{bl}.

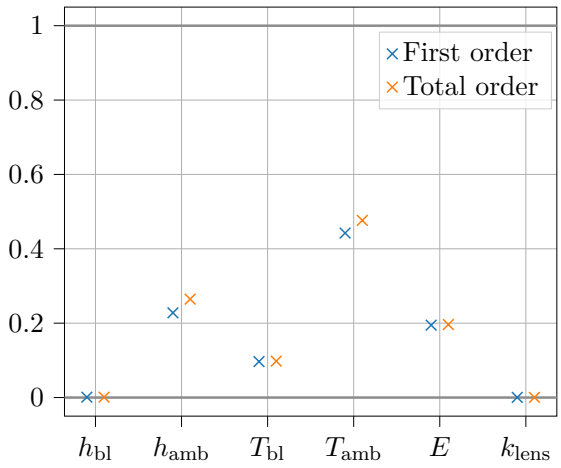
Sobol' indices for several other locations are gathered in Figure 5.6(b–f). From these results, we can infer the following ranking of the influential parameters: T_{amb}, h_{amb}, E, and T_{bl}. In particular, the dependence of the ambient temperature T_{amb} decreases when we go deeper inside the eye. Precisely, the impact of T_{amb} is still significant for the mean temperature of the cornea, but the other parameters are equally influential. These behaviors are coherent with physiological conditions. Moreover, regardless of the output studied, the parameters k_{lens} and h_{bl} are minimally influencing the output. Consequently, in future simulations, their value can be set at baseline. Surprisingly, the temperature at B_1, on the lens, is minimally influenced by k_{lens}, but this parameter has a minimal role in the modeling process. On the other hand, T_{bl} is very influential at D_1 and G, close to vascular beds, again in a coherent manner with the physiological situation. Finally, we can notice a slight difference between the first-order and

total-order indices, mostly for h_{amb} and T_{amb}, implying that there are high-order interactions among these selected parameters. To measure the impact of coupled parameters, second-order Sobol' indices computation is required, but the polynomial chaos expansion does not directly provide these values. Alternatively, a Monte-Carlo based method could be implemented, but this strategy would be very costly from the computational viewpoint.

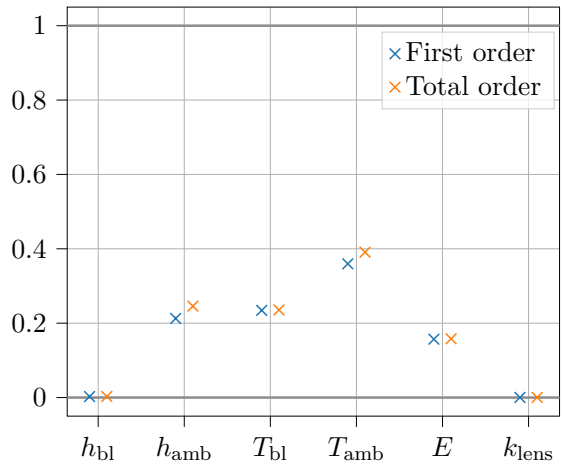
5.4 Conclusion

In this chapter, we have presented a sensitivity analysis framework to investigate the impact of biomechanical parameters on the output of model of heat transfer in the human eye introduced in Chapter 1. On the one hand, we performed a deterministic sensitivity analysis to assess the influence of individual parameters on the output, and compared our result to previous studies, hence validating our model. Then, we extended the analysis to a stochastic framework, performed an uncertainty propagation, and computed Sobol' indices to quantify the impact of each parameter and their interactions. The SSA nor only confirmed the findings of the DSA, but also identified high-order interactions among selected parameters that should be further explored.

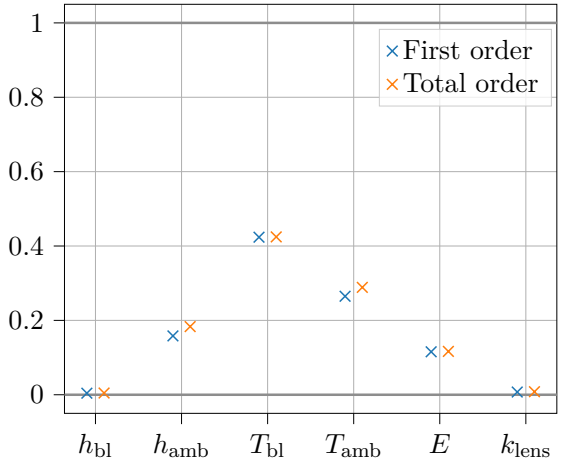
These studies provide valuable insights into the behavior of the model and the sensitivity of the output to the input parameters, and could be extended to explore more complex models or other applications.



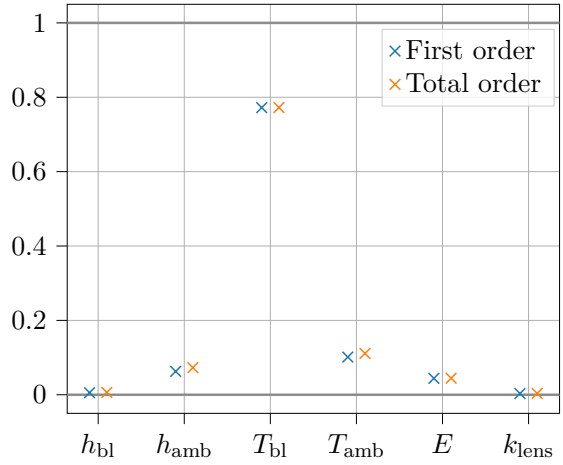
(a) Output T_{O}.



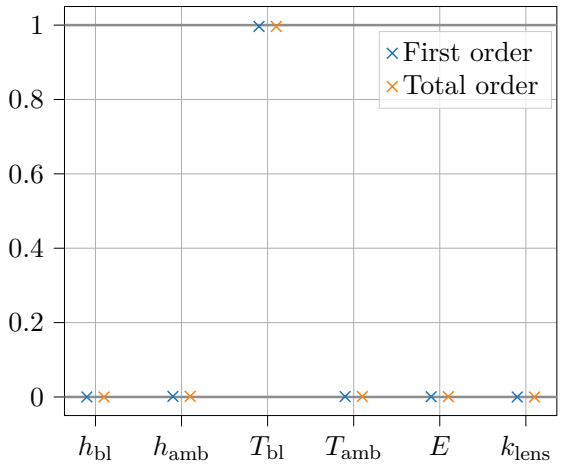
(b) Mean temperature on the cornea.



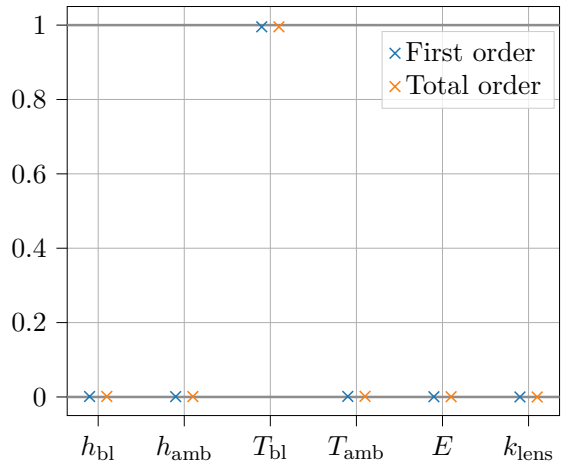
(c) Output T_{B_1}.



(d) Output T_C.



(e) Output T_{D_1}.



(f) Output T_G.

Figure 5.6: SSA results: Sobol' indices for different outputs.

Chapter 6

Modeling heat transfer coupled with aqueous humor flow

The model presented in this chapter is an extension of the thermal described in Chapter 1, which focused on heat transfer within the ocular tissues without considering fluid dynamics. In the present work, we integrate the flow of the aqueous humor (AH) in the anterior chamber (AC) and posterior chamber (PC) of the eye, coupling it with the thermal processes inside the eyeball. This coupling is crucial, as the movement of AH plays a significant role in heat distribution and intraocular pressure regulation, affecting overall ocular health. Moreover, the wall shear stress generated by the AH flow is an important biomechanical factor, influencing the health of ocular tissues and potentially impacting the drainage pathways, which are relevant to conditions such as glaucoma.

Previous studies have investigated aspects of this complex interaction, as reviewed in [Dvo+19]. For instance, [HB02; Wan+16; Mur+23] modeled flow coupled with heat transfer in the AC and PC, while [Sac+23] explored the impact of pressure on AH flow and drainage. Other works, like [ON08; BBS20; Abd+21; Dvo+22] examined the thermo-fluid dynamics of AH flow in the AC with specified boundary conditions. However, these studies often focused on simplified geometries or did not fully couple the heat transfer within the entire eyeball.

To address these gaps, our work develops a three-dimensional computational model that simulates heat transfer throughout the whole human eyeball, accounting for the dynamic flow of AH in both the AC and PC. The model aims to provide deeper insights into the ocular thermal environment and its interaction with fluid dynamics, which is essential for understanding physiopathological condition such as glaucoma and for improving drug delivery methods.

Solving the coupled three-dimensional thermo-fluid dynamics model of the eye numerically poses significant computational challenges due to the complexity and nonlinearity of the governing equations. The interaction between heat transfer and fluid flow requires solving large, sparse linear systems that can be computationally expensive and time-consuming. To address these challenges, it is essential to employ adapted preconditioners that can enhance the convergence rate of iterative solvers. Preconditioners transform the original system into a form that is easier to solve, reducing the number of iterations needed and improving overall computational efficiency. By using tailored preconditioning techniques such as GAMG or the Schur complement [ESW14], we achieve faster and more stable solutions, enabling the simulation of more detailed and realistic models of ocular physiology.

The work of this chapter is a collaboration with Vincent Chabannes (Cemosis, Université de Strasbourg). It is organized as follows: in Section 6.1, we present the biophysical model of the human eye, focusing on the detailed geometrical representation and the mechanisms governing AH flow. Section 6.2 describes the discretization techniques and computational framework employed, including the numerical methods and preconditioning strategies used to solve the

coupled equations efficiently. In Section 6.3, we present and discuss the results obtained from our simulations, highlighting the model's capabilities and potential applications. Finally, Section 6.5 summarizes our findings and outlines future research directions in the field of mathematical and computational ophthalmology.

6.1 Biophysical model

In this section, we present the biophysical model of the human eye that we aim to simulate. We describe the biomechanical behavior involved in heat transfer and fluid dynamics.

We briefly recall the geometrical model of the human eye presented in Section 1.3. The computational domain, denoted by Ω, represents the entire human eye as shown in Figure 1.4. This domain is subdivided into ten subdomains, each representing different anatomical parts of the eye such as the cornea, lens, vitreous body, retina, *etc.*, each with its own physical properties. Specifically, we denote by Ω_{AH} the domain corresponding to the anterior and posterior chambers of the eye (highlighted in brown in Figure 1.4), which are filled with the *aqueous humor* (AH). The dynamics of the AH were previously described in Figure 1.2. We furthermore decompose the boundary of Ω_{AH} into five parts, as shown in Figure 6.1: *(i)* Denote by Γ_C the interface of Ω_{AH} with the cornea, *(ii)* Γ_I with the iris, *(iii)* Γ_L with the lens, *(iv)* Γ_{VH} with the vitreous humor, and *(v)* Γ_{Sc} with the sclera. As discussed in Section 1.3, the region corresponding to the suspensory ligaments of the lens was excluded from the geometrical model and incorporated into the PC, resulting in a domain that is slightly larger than in reality.

Bio-heat and fluid dynamics model Following the approach of Abdelhafid et al. [Abd+21] and Wang et al. [Wan+16], we make the following assumptions:

- **Incompressible fluid:** The aqueous humor is considered an incompressible Newtonian fluid due to its low compressibility and viscosity.
- **Boussinesq approximation:** Density variations in the fluid are small and can be neglected except in the buoyancy term. This approximation allows us to model the buoyancy effects due to temperature differences without accounting for full density variations.

Under these assumptions, the steady flow of AH is governed by the incompressible Navier-Stokes equations coupled with heat transfer:

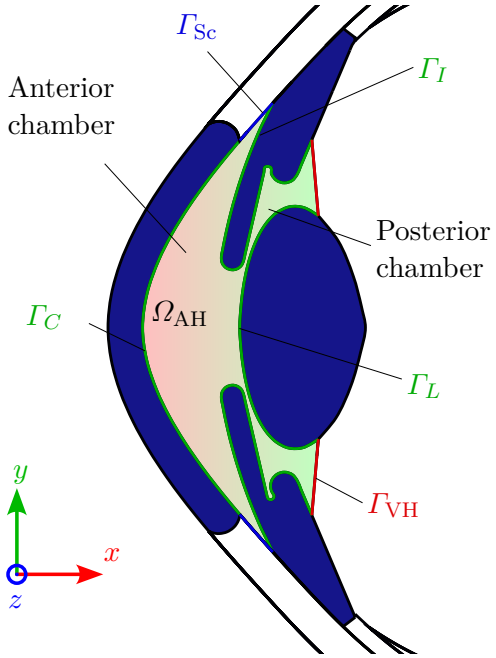


Figure 6.1: Vertical cut of the anterior and posterior chambers of the eye.

$$\rho(\vec{u}\cdot\nabla)\vec{u} - \nabla\cdot\underline{\underline{\sigma}} = -\rho\beta(T - T_{\text{ref}})\vec{g} \quad \text{in } \Omega_{\text{AH}}, \tag{6.1a}$$

$$\nabla\cdot\vec{u} = 0 \quad \text{in } \Omega_{\text{AH}}, \tag{6.1b}$$

$$\rho C_p\vec{u}\cdot\nabla T - k\nabla^2 T = 0 \quad \text{in } \Omega, \tag{6.1c}$$

where μ [N s/m^2] is the dynamic viscosity of the fluid, ρ [kg/m^3] its density (both at reference temperature T_{ref} [K]), C_p [J kg^{-1} K^{-1}] its specific heat, k [W m^{-1} K^{-1}] its thermal conductivity. The quantity T [K] is the temperature of the eye, while p [Pa] is the pressure of the aqueous humor fluid (also expressed in mmHg in a biologic context), and $\vec{u}$ [m s^{-1}] is its velocity. As discussed in Section 6.1, we neglect the metabolic heat generation in the eye due to blood perfusion.

The behavior of the fluid is characterized by the Cauchy stress tensor $\underline{\underline{\sigma}}$ defined as

$$\underline{\underline{\sigma}}(\vec{u}, p) = -p\underline{\underline{I}} + 2\mu\underline{\underline{D}}(\vec{u}), \tag{6.2}$$

where $\underline{\underline{I}}$ is the identity tensor, and $\underline{\underline{D}}(\vec{u}) = \frac{1}{2}\left(\nabla\vec{u} + \nabla\vec{u}^T\right)$ is the *strain rate tensor*.

The right-hand side term in Equation (6.1a) represents the gravitational force per unit volume, along with the Boussinesq approximation [DR04], utilized to account for the buoyancy effects due to temperature variations. This approximation states that the fluid's density varies with temperature but remains virtually unaffected by pressure, as discussed above. The coefficient β [K^{-1}] is the fluid volume expansion coefficient, and $\vec{g}$ [m s^{-2}] the gravitational acceleration vector. As the variation of temperature is small, we can consider that the density is constant in the fluid domain, and the Boussinesq approximation is valid. Depending on the position of the patient (standing, laying supine or prone respectively), $\vec{g}$ can be either vertical ($\vec{g} = [0, -g, 0]^T$) or horizontal ($\vec{g} = [g, 0, 0]^T$, $\vec{g} = [-g, 0, 0]^T$ respectively), where g is the gravitational acceleration.

We impose no-slip boundary conditions for the fluid velocity on the boundaries of Ω_{AH}:

$$\vec{u} = \vec{0} \quad \text{on } \Gamma_C \cup \Gamma_I \cup \Gamma_L \cup \Gamma_{\text{VH}} \cup \Gamma_{\text{Sc}}. \tag{6.3}$$

These boundary conditions, together with Equation (6.1), model the AH flow driven by thermal and gravitational effects. The hydraulic pressure difference due to the production and drainage of AH is not explicitly accounted for, as previous studies have indicated that buoyancy is the dominant mechanism driving convective motion in the AH regardless of postural orientation [ON08; Kum+06].

Note that alternative boundary conditions, such as non-homogeneous Dirichlet conditions [HB02; Abd+21] or specified flow rates and pressures [Wan+16], could also be considered but are beyond the scope of the present study.

As presented in Section 1.4, the heat transfer within the eye is governed by Equation (6.1c), endowed with the following boundary conditions:

$$-k\frac{\partial T}{\partial \vec{n}} = h_{\text{bl}}(T - T_{\text{bl}}) \qquad \text{on } \Gamma_{\text{body}}, \tag{6.4a}$$

$$-k\frac{\partial T}{\partial \vec{n}} = \underbrace{h_{\text{amb}}(T - T_{\text{amb}})}_{(i)} + \underbrace{\sigma\varepsilon(T^4 - T_{\text{amb}}^4)}_{(ii)} + \underbrace{E}_{(iii)} \qquad \text{on } \Gamma_{\text{amb}}. \tag{6.4b}$$

Precisely, Equation (6.4a) models the convective heat transfer between the eye and the surrounding body, where h_{bl} [$\text{W m}^{-2}\,\text{K}^{-1}$] is the heat transfer coefficient between the eye and the surrounding body, and T_{bl} [K] is the blood temperature. On the other hand, Equation (6.4b) models three types of exchanges that are involved between the eyeball and the ambient air: *(i)* **Convective heat transfer**, where h_{amb} [$\text{W m}^{-2}\,\text{K}^{-1}$] is the heat transfer coefficient between the eye and the ambient air, and T_{amb} [K] is the ambient temperature; *(ii)* **radiative heat transfer**, where $\sigma = 5.67 \times 10^{-8}\,\text{W/m}^2/\text{K}^4$ represents the Stefan-Boltzmann constant, and ε [–] is the emissivity of the cornea; and *(iii)* **evaporative heat loss**, due to tear evaporation at the surface of the eye, where E [W m^{-2}] is the evaporation rate of the tear film.

Remark 6.1.1. One might consider simplifying the momentum equations by using the Stokes equations, omitting the non-linear convective term $\rho(\vec{u}\cdot\nabla)\vec{u}$, especially since the Reynolds number in ocular flow is low [Wan+16]. However, due to the coupling with the heat equation through buoyancy effects, non-linearities remain in the system, necessitating an iterative solution strategy regardless.

We performed a comparative study of different modeling approaches, comparing computational times for assembly and solution phases, as summarized in Table 6.1. While the use of linearized boundary conditions in the sense of Equation (1.3) and the Stokes equations slightly reduced the computational time, the differences were not substantial. Thus, we opted to use the fully non-linear Navier-Stokes equations to maintain the accuracy of the model.

The computational times indicate that while linear boundary conditions and the Stokes model reduce solution time, the fully non-linear Navier-Stokes model provides a more accurate representation of the AH dynamics with only a modest increase in computational effort.

Table 6.1: Comparison of computational times for different models, using mesh `Mr4` parallelized on 256 cores.

(a) Time to assembly the algebraic objects.

Model	Boundary condition	Time [s]
Navier-Stokes	Non-linear	0.24
Navier-Stokes	Linear	0.25
Stokes	Non-linear	0.24
Stokes	Linear	0.26

(b) Time to solve the problem.

Model	Boundary condition	Time [s]
Navier-Stokes	Non-linear	68.89
Navier-Stokes	Linear	43.7
Stokes	Non-linear	69.14
Stokes	Linear	46.2

6.2 Mathematical and computational framework

In this section, we present the mathematical and computational framework used to solve the coupled fluid dynamics and heat transfer model described in (6.1–6.3–6.4). We detail the variational formulation, finite element discretization, and the solution strategy employed.

We begin by deriving the variational formulation of the model. Let us introduce the following functional spaces:

(i) The velocity space $\boldsymbol{V} := [H_0^1(\Omega_{\mathrm{AH}})]^3$, consisting of vector fields $\vec{u}$ with square-integrable derivatives that vanish on the boundary;

(ii) the pressure space $Q := L_0^2(\Omega_{\mathrm{AH}}) = \left\{ p \in L^2(\Omega_{\mathrm{AH}}) \middle| \int_{\Omega_{\mathrm{AH}}} p \,\mathrm{d}x = 0 \right\}$, consisting of square-integrable scalar fields with zero mean; and

(iii) the temperature space $W := H^1(\Omega)$.

Let $\vec{v} \in \boldsymbol{V}$, $q \in Q$, and $\varphi \in W$ be test functions. Multiplying Equations (6.1a), (6.1b), and (6.1c) by $\vec{v}$, q, and φ, respectively, and integrating over the appropriate domains yield:

$$\rho \int_{\Omega_{\mathrm{AH}}} (\vec{u} \cdot \nabla)\vec{u} \cdot \vec{v} \,\mathrm{d}\vec{x} + \mu \int_{\Omega_{\mathrm{AH}}} \underline{\underline{D}}(\vec{u}) : \nabla \vec{v} \,\mathrm{d}\vec{x} - \int_{\Omega_{\mathrm{AH}}} p \cdot \nabla \vec{v} \,\mathrm{d}\vec{x} + \int_{\Omega_{\mathrm{AH}}} \rho_0 \beta T \vec{g} \cdot \vec{v} \,\mathrm{d}\vec{x} = \int_{\Omega_{\mathrm{AH}}} \rho_0 \beta T_{\mathrm{ref}} \vec{g} \cdot \vec{v} \,\mathrm{d}\vec{x}, \tag{6.5a}$$

$$\int_{\Omega_{\mathrm{AH}}} \nabla \cdot \vec{u} q \,\mathrm{d}\vec{x} = 0, \tag{6.5b}$$

$$\rho C_p \int_{\Omega} \vec{u} \cdot \nabla T \varphi \,\mathrm{d}\vec{x} + k \int_{\Omega} \nabla T \cdot \nabla \varphi \,\mathrm{d}\vec{x} + \int_{\Gamma_{\mathrm{amb}}} \left(h_{\mathrm{amb}} T \,\mathrm{d}\sigma + \sigma \varepsilon T^4 \right) \varphi \,\mathrm{d}\sigma + \int_{\Gamma_{\mathrm{body}}} h_{\mathrm{bl}} T \varphi \,\mathrm{d}\sigma = \int_{\Gamma_{\mathrm{amb}}} \left(h_{\mathrm{amb}} T_{\mathrm{amb}} + \sigma \varepsilon T_{\mathrm{amb}}^4 \right) \varphi \,\mathrm{d}\sigma + \int_{\Gamma_{\mathrm{body}}} h_{\mathrm{bl}} T_{\mathrm{bl}} \varphi \,\mathrm{d}\sigma. \tag{6.5c}$$

We define the following bilinear and trilinear forms:

$$a_1(\vec{z}, \vec{u}, \vec{v}) = \rho \int_{\Omega_{\mathrm{AH}}} (\vec{z} \cdot \nabla)\vec{u} \cdot \vec{v} \,\mathrm{d}\vec{x}, \tag{6.6a}$$

$$a_2(\vec{u}, \vec{v}) = \mu \int_{\Omega_{\mathrm{AH}}} \underline{\underline{D}}(u) : \nabla \vec{v} \,\mathrm{d}\vec{x}, \tag{6.6b}$$

$$b(p, \vec{v}) = -\int_{\Omega_{\mathrm{AH}}} p \cdot \nabla \vec{v} \,\mathrm{d}\vec{x}, \tag{6.6c}$$

$$d(T, \vec{v}) = -\int_{\Omega_{\mathrm{AH}}} \rho_0 \beta T \vec{g} \cdot \vec{v} \,\mathrm{d}\vec{x}, \tag{6.6d}$$

$$e(\vec{u}, T, \varphi) = \rho C_p \int_{\Omega} \vec{u} \cdot \nabla T \varphi \,\mathrm{d}\vec{x}, \tag{6.6e}$$

$$f(T, \varphi) = k \int_{\Omega} \nabla T \cdot \nabla \varphi \,\mathrm{d}\vec{x} + \int_{\Gamma_{\mathrm{amb}}} \left(h_{\mathrm{amb}} T + \sigma \varepsilon T^4 \right) \varphi \,\mathrm{d}\sigma + \int_{\Gamma_{\mathrm{body}}} h_{\mathrm{bl}} T \varphi \,\mathrm{d}\sigma, \tag{6.6f}$$

$$\ell_1(\vec{v}) = \int_{\Omega_{\mathrm{AH}}} \rho_0 \beta T_{\mathrm{ref}} \vec{g} \cdot \vec{v} \,\mathrm{d}\vec{x}, \tag{6.6g}$$

$$\ell_2(\varphi) = \int_{\Gamma_{\mathrm{amb}}} \left(h_{\mathrm{amb}} T_{\mathrm{amb}} + \sigma \varepsilon T_{\mathrm{amb}}^4 \right) \varphi \,\mathrm{d}\sigma + \int_{\Gamma_{\mathrm{body}}} h_{\mathrm{bl}} T_{\mathrm{bl}} \varphi \,\mathrm{d}\sigma. \tag{6.6h}$$

The variational formulation of the problem is then: Find $(\vec{u}, p, T) \in \boldsymbol{V} \times Q \times W$ such that for all $(\vec{v}, q, \varphi) \in \boldsymbol{V} \times Q \times W$:

$$\begin{aligned} a_1(\vec{u},\vec{u},\vec{v}) + a_2(\vec{u},\vec{v}) + b(p,\vec{v}) + d(T,\vec{v}) &= \ell_1(\vec{v}), \\ b(q,\vec{u}) &= 0, \\ e(\vec{u},T,\varphi) + f(T,\varphi) &= \ell_2(\varphi). \end{aligned} \tag{6.7}$$

Theorem 6.2.1 (Existence and Uniqueness)**.** *The variational problem* (6.7) *has a unique solution under appropriate assumptions on the data.*

Proof. A detailed proof is beyond the scope of this section. However, the existence and uniqueness can be established using fixed-point arguments and standard results for the Navier-Stokes equations coupled with heat transfer, as discussed in [Tsu15]. ■

6.2.1 Finite element setting

We discretize the variational problem using the finite element method (FEM). The computational domain is discretized as described in Section 2.1.2, with mesh refinement in the anterior and posterior chambers to accurately capture the flow and thermal dynamics.

We define the finite element spaces for velocity, pressure, and temperature as follows:

- $\boldsymbol{V}_h := \{\vec{v} \in [\mathbb{P}_2(\Omega_{\text{AH}})]^3 \,|\, \vec{v} = 0 \text{ on } \partial\Omega\}$, the space of vector-valued piece-wise quadratic polynomials that vanish on the boundary for velocity.
- $Q_h := \mathbb{P}_1(\Omega_{\text{AH}})$, the space of piece-wise linear polynomials for pressure.
- $W_h := \mathbb{P}_1(\Omega)$, the space of piece-wise linear polynomials for temperature.

This choice corresponds to the $\mathbb{P}_1$–$\mathbb{P}_2\mathbb{P}_1$ Taylor-Hood element for the velocity-pressure pair, which satisfies the LBB (Ladyzhenskaya-Babuška-Brezzi) stability condition. Other discretization strategies could be employed, such as $\mathbb{P}_1$–$\mathbb{P}_1\mathbb{P}_1$ or $\mathbb{P}_2$–$\mathbb{P}_2\mathbb{P}_1$. The former requires a stabilization term to ensure stability, while the latter is more computationally expensive and may not provide significant improvements in accuracy.

The discrete problem reads: Find $(\vec{u}_h, p_h, T_h) \in \boldsymbol{V}_h \times Q_h \times W_h$ such that for all $(\vec{v}_h, q_h, \varphi_h) \in \boldsymbol{V}_h \times Q_h \times W_h$:

$$\begin{aligned} a_1(\vec{u}_h,\vec{u}_h,\vec{v}_h) + a_2(\vec{u}_h,\vec{v}_h) + b(p_h,\vec{v}_h) + d(T_h,\vec{v}_h) &= \ell_1(\vec{v}_h), \\ b(q_h,\vec{u}_h) &= 0, \\ e(\vec{u}_h,T_h,\varphi_h) + f(T_h,\varphi_h) &= \ell_2(\varphi_h). \end{aligned} \tag{6.8}$$

The system (6.8) is non-linear because of the terms a_1 and e. We employ Newton's method to solve it iteratively. Precisely, it consists of starting from an initial guess $(\vec{u}^0, p^0, T^0)$, and iteratively compute $(\vec{u}^{k+1}, p^{k+1}, T^{k+1})$ as the solution of the non-linear system at each iteration. We set the correction terms $\delta\vec{u}^k := \vec{u}^{k+1} - \vec{u}^k$, $\delta p^k := p^{k+1} - p^k$, and $\delta T^k := T^{k+1} - T^k$. Given $(\vec{u}^k, p^k, T^k)$, we define the *nonlinear residual* associated with the variational formulation (6.8) as:

$$r_{\vec{u}}^k(\vec{v}) := \ell_1(\vec{v}) - a_1(\vec{u}^k,\vec{u}^k,\vec{v}) - a_2(\vec{u}^k,\vec{v}) - b(p^k,\vec{v}) - d(T^k,\vec{v}), \tag{6.9a}$$

$$r_p^k(q) := -b(q,\vec{u}^k), \tag{6.9b}$$

$$r_T^k(\varphi) := \ell_2(\varphi) - e(\vec{u}^k,T^k,\varphi) - f(T^k,\varphi). \tag{6.9c}$$

In the spirit of [ESW14, Ch. 8], by dropping the quadratics terms, the correction terms verify the following weak linear problem: $\forall(\vec{v}, q, \varphi) \in \boldsymbol{V} \times Q \times T$:

$$a_1(\delta\vec{u}^k, \vec{u}^k, \vec{v}) + a_1(\vec{u}^k, \delta\vec{u}^k, \vec{v}) + a_2(\delta\vec{u}^k, \vec{v}) + b(\delta p^k, \vec{v}) + d(\delta T^k, \vec{v}) = r_{\vec{u}}^k(\vec{v}), \tag{6.10a}$$

$$b(\delta p^k, \vec{u}^k) = r_p^k(q), \tag{6.10b}$$

$$e(\delta\vec{u}^k, T^k, \varphi) + e(\vec{u}^k, \delta T^k, \varphi) + f(\delta T^k, \varphi) = r_T^k(\varphi). \tag{6.10c}$$

In the discretized spaces $\boldsymbol{V}_h \times Q_h \times T_h$, the same variational problem is solved. To define the corresponding linear algebra problem, we set the basis of the discrete spaces: $\{\vec{\lambda}_i\}_{i=1}^{N_u}$ is a basis of $\boldsymbol{V}_h$, $\{\mu_j\}_{j=1}^{N_p}$ is a basis of Q_h, and $\{\xi_l\}_{l=1}^{N_T}$ is a basis of T_h. Setting $\underline{\boldsymbol{u}}$, $\underline{\boldsymbol{\Delta u}}$, $\underline{\boldsymbol{p}}$, $\underline{\boldsymbol{\Delta p}}$, $\underline{\boldsymbol{T}}$, and $\underline{\boldsymbol{\Delta T}}$ the vectors of the coefficients of the basis functions in the corresponding basis of $\vec{u}$, $\delta\vec{u}$, p, δp, T, and δT respectively, the algebraic problem reads:

$$\begin{bmatrix} \underline{\underline{\boldsymbol{V}}}^k + \underline{\underline{\boldsymbol{W}}}^k + \underline{\underline{\boldsymbol{N}}} & \underline{\underline{\boldsymbol{B}}}^T & \underline{\underline{\boldsymbol{D}}} \\ \underline{\underline{\boldsymbol{B}}} & \underline{\underline{\boldsymbol{0}}} & \underline{\underline{\boldsymbol{0}}} \\ \underline{\underline{\boldsymbol{E}}}_1^k & \underline{\underline{\boldsymbol{0}}} & \underline{\underline{\boldsymbol{E}}}_2^k + \underline{\underline{\boldsymbol{F}}} \end{bmatrix} \begin{bmatrix} \underline{\boldsymbol{\Delta u}} \\ \underline{\boldsymbol{\Delta p}} \\ \underline{\boldsymbol{\Delta T}} \end{bmatrix} = \begin{bmatrix} \underline{\boldsymbol{r}}_{\vec{u}}^k \\ \underline{\boldsymbol{r}}_p^k \\ \underline{\boldsymbol{r}}_T^k \end{bmatrix}, \tag{6.11}$$

where all elements of this system are defined on the basis of the discrete spaces:

$$\begin{gathered} \underline{\underline{\boldsymbol{V}}}^k = \left[a_1(\vec{u}^k, \vec{\lambda}_i, \vec{\lambda}_j)\right] \in \mathbb{R}^{N_u \times N_u}, \ \underline{\underline{\boldsymbol{W}}}^k = \left[a_1(\vec{\lambda}_i, \vec{u}^k, \vec{\lambda}_j)\right] \in \mathbb{R}^{N_u \times N_u}, \ \underline{\underline{\boldsymbol{N}}} = \left[a_2(\vec{\lambda}_i, \vec{\lambda}_j)\right] \in \mathbb{R}^{N_u \times N_u}, \\ \underline{\underline{\boldsymbol{B}}} = \left[b(\mu_l, \vec{\lambda}_j)\right] \in \mathbb{R}^{N_p \times N_u}, \quad \underline{\underline{\boldsymbol{D}}} = \left[d(\vec{\lambda}_i, \xi_l)\right] \in \mathbb{R}^{N_u \times N_T}, \\ \underline{\underline{\boldsymbol{E}}}_1^k = \left[e(\vec{\lambda}_i, T^k, \xi_l)\right] \in \mathbb{R}^{N_u \times N_T}, \quad \underline{\underline{\boldsymbol{E}}}_2^k = \left[e(\vec{u}^k, \xi_l, \xi_m)\right] \in \mathbb{R}^{N_T \times N_T}, \quad \underline{\underline{\boldsymbol{F}}} = \left[f(\xi_l, \xi_m)\right] \in \mathbb{R}^{N_T \times N_T}, \\ \underline{\boldsymbol{r}}_{\vec{u}}^k = \left[r_{\vec{u}}^k(\vec{\lambda}_i)\right] \in \mathbb{R}^{N_u}, \quad \underline{\boldsymbol{r}}_p^k = \left[r_p^k(\mu_j)\right] \in \mathbb{R}^{N_p}, \quad \underline{\boldsymbol{r}}_T^k = \left[r_T^k(\xi_l)\right] \in \mathbb{R}^{N_T}. \end{gathered} \tag{6.12}$$

From the initial guess, we iteratively solve the linear algebra problem (6.11) at each Newton iteration, until the relative increment $\mathrm{Crit}^k := \max\{\|\delta\vec{u}^k\|/\|\delta\vec{u}^0\|, \|\delta p^k\|/\|\delta p^0\|, \|\delta T^k\|/\|\delta T^0\|\} \leqslant \varepsilon_{\mathrm{tol}}$ is reached for a given tolerance $\varepsilon_{\mathrm{tol}} > 0$. Algorithm 10 summarizes the Newton iteration loop.

Algorithm 10: Newton iteration loop.

Input: $\{\underline{\boldsymbol{u}}^0, \underline{\boldsymbol{p}}^0, \underline{\boldsymbol{T}}^0, \varepsilon_{\mathrm{tol}}\}$.
$(\underline{\boldsymbol{u}}^0, \underline{\boldsymbol{p}}^0, \underline{\boldsymbol{T}}^0) \leftarrow$ initial guess;
Assemble $\underline{\underline{\boldsymbol{N}}}$, $\underline{\underline{\boldsymbol{B}}}$, $\underline{\underline{\boldsymbol{D}}}$, $\underline{\underline{\boldsymbol{F}}}$;
while $\mathrm{Crit}^k > \varepsilon_{\mathrm{tol}}$ **do**
  Assemble $\underline{\underline{\boldsymbol{V}}}^k$, $\underline{\underline{\boldsymbol{W}}}^k$, $\underline{\underline{\boldsymbol{E}}}_1^k$, $\underline{\underline{\boldsymbol{E}}}_2^k$, $\underline{\boldsymbol{r}}_{\vec{u}}^k$, $\underline{\boldsymbol{r}}_p^k$, $\underline{\boldsymbol{r}}_T^k$;
  $(\underline{\boldsymbol{\Delta u}}, \underline{\boldsymbol{\Delta p}}, \underline{\boldsymbol{\Delta T}}) \leftarrow$ solution of System (6.11);
  $\underline{\boldsymbol{u}}^{k+1} \leftarrow \underline{\boldsymbol{u}}^k + \alpha\underline{\boldsymbol{\Delta u}}$, $\underline{\boldsymbol{p}}^{k+1} \leftarrow \underline{\boldsymbol{p}}^k + \alpha\underline{\boldsymbol{\Delta p}}$, $\underline{\boldsymbol{T}}^{k+1} \leftarrow \underline{\boldsymbol{T}}^k + \alpha\underline{\boldsymbol{\Delta T}}$;
end
Output: $(\underline{\boldsymbol{u}}^{k+1}, \underline{\boldsymbol{p}}^{k+1}, \underline{\boldsymbol{T}}^{k+1})$.

The parameter $\alpha \in [0, 1]$ is a relaxation factor that can be adjusted, using the line search method of PETSc [Bal+24], to improve the convergence of the Newton iteration.

6.2.2 Solution strategy

We implement the computational framework using the heatfluid toolbox of Feel++[1] [Pru+24b] to solve Algorithm 10. Efficiently solving the resulting linear systems at each Newton iteration is

[1] See documentation: https://docs.feelpp.org/toolboxes/latest/heatfluid/

crucial.

Direct solvers become impractical for large-scale problems due to computational and memory constraints. Therefore, we employ iterative solvers with appropriate preconditioners to enhance convergence. The preconditioner is a key component in the iterative solver, as it actually enables the solver to converge. We utilize a *field-split* preconditioning strategy, where the global system is partitioned into smaller blocks corresponding to different physical fields (*e.g.*, fluid and thermal fields). This allows us to apply specialized solvers and preconditioners to each block.

By transforming the original linear system $\underline{\underline{\boldsymbol{A}}}\,\underline{\boldsymbol{x}} = \underline{\boldsymbol{b}}$ into an equivalent one $\underline{\underline{\boldsymbol{P}}}^{-1}\underline{\underline{\boldsymbol{A}}}\,\underline{\boldsymbol{x}} = \underline{\underline{\boldsymbol{P}}}^{-1}\underline{\boldsymbol{b}}$ with more favorable properties for numerical solution, such as a better condition number, preconditioners play a critical role in enhancing the performance and robustness of the solver. Here, $\underline{\underline{\boldsymbol{P}}}^{-1}$ approximates $\underline{\underline{\boldsymbol{A}}}^{-1}$, making $\underline{\underline{\boldsymbol{P}}}^{-1}\,\underline{\underline{\boldsymbol{A}}}$ easier to invert than $\underline{\underline{\boldsymbol{A}}}$ and having a spectrum close to that of $\underline{\underline{\boldsymbol{A}}}$.

For clarity, we rewrite the system (6.11) with block notation, omitting the superscript k:

$$\left[\begin{array}{cc|c} \underline{\underline{\widetilde{\boldsymbol{A}}}} & \underline{\underline{\boldsymbol{B}}}^T & \underline{\underline{\boldsymbol{D}}} \\ \underline{\underline{\boldsymbol{B}}} & \underline{\underline{\boldsymbol{0}}} & \underline{\underline{\boldsymbol{0}}} \\ \hline \underline{\underline{\boldsymbol{E}}} & \underline{\underline{\boldsymbol{0}}} & \underline{\underline{\widetilde{\boldsymbol{F}}}} \end{array}\right] \left[\begin{array}{c} \underline{\boldsymbol{\Delta u}} \\ \underline{\boldsymbol{\Delta p}} \\ \hline \underline{\boldsymbol{\Delta T}} \end{array}\right] = \left[\begin{array}{c} \underline{\boldsymbol{r}}_{\vec{u}} \\ \underline{\boldsymbol{r}}_p \\ \hline \underline{\boldsymbol{r}}_T \end{array}\right] \quad \Longleftrightarrow: \quad \underbrace{\left[\begin{array}{c|c} \underline{\underline{\boldsymbol{K}}}_{0,0} & \underline{\underline{\boldsymbol{K}}}_{0,1} \\ \hline \underline{\underline{\boldsymbol{K}}}_{1,0} & \underline{\underline{\boldsymbol{K}}}_{1,1} \end{array}\right]}_{=:\underline{\underline{\boldsymbol{K}}}} \begin{bmatrix} \underline{\boldsymbol{\Delta}}_{\text{fluid}} \\ \underline{\boldsymbol{\Delta}}_{\text{heat}} \end{bmatrix} = \begin{bmatrix} \underline{\boldsymbol{r}}_{\text{fluid}} \\ \underline{\boldsymbol{r}}_{\text{heat}} \end{bmatrix}. \tag{6.13}$$

The main idea of *additive fieldsplit* preconditioner is to approximate the inverse of the matrix $\underline{\underline{K}}$ by the matrix

$$\begin{bmatrix} \underline{\underline{\boldsymbol{K}}}_{0,0}^{-1} & \underline{\underline{\boldsymbol{0}}} \\ \underline{\underline{\boldsymbol{0}}} & \underline{\underline{\boldsymbol{K}}}_{1,1}^{-1} \end{bmatrix}, \tag{6.14}$$

where the inverses of the diagonal blocks are applied separately, with appropriate solvers and associated preconditioners.

The heat block $\underline{\underline{\boldsymbol{K}}}_{1,1}$ inverse is approximated using a few iterations of *GAMG* — Geometric Algebraic Multigrid — from PETSc [Bal+24]. This preconditioner efficiently handles large sparse matrices by recursively coarsening and solving the problem on multiple levels, significantly accelerating the convergence.

On the other hand, concerning the fluid block $\underline{\underline{\boldsymbol{K}}}_{0,0}$, the inverse is approximated using the Schur complement, as proposed in [ESW14, Ch. 9], where another field split preconditioner is implemented: the degrees of freedom are now divided into velocity and pressure blocks. Then, the LDU decomposition of the matrix $\underline{\underline{\boldsymbol{K}}}_{0,0}$ is computed as follows:

$$\underline{\underline{\boldsymbol{K}}}_{0,0} = \left[\begin{array}{c|c} \underline{\underline{\widetilde{\boldsymbol{A}}}} & \underline{\underline{\boldsymbol{B}}}^T \\ \hline \underline{\underline{\boldsymbol{B}}} & \underline{\underline{\boldsymbol{0}}} \end{array}\right] = \underbrace{\begin{bmatrix} \underline{\underline{\boldsymbol{I}}} & \underline{\underline{\boldsymbol{0}}} \\ \underline{\underline{\boldsymbol{B}}}\,\underline{\underline{\widetilde{\boldsymbol{A}}}}^{-1} & \underline{\underline{\boldsymbol{I}}} \end{bmatrix}}_{\underline{\underline{\boldsymbol{L}}}} \underbrace{\begin{bmatrix} \underline{\underline{\widetilde{\boldsymbol{A}}}} & \underline{\underline{\boldsymbol{0}}} \\ \underline{\underline{\boldsymbol{0}}} & \underline{\underline{\boldsymbol{S}}} \end{bmatrix}}_{\underline{\underline{\boldsymbol{D}}}} \underbrace{\begin{bmatrix} \underline{\underline{\boldsymbol{I}}} & \underline{\underline{\widetilde{\boldsymbol{A}}}}^{-1}\underline{\underline{\boldsymbol{B}}}^T \\ \underline{\underline{\boldsymbol{0}}} & \underline{\underline{\boldsymbol{I}}} \end{bmatrix}}_{\underline{\underline{\boldsymbol{U}}}}, \tag{6.15}$$

where $\underline{\underline{\boldsymbol{S}}} = -\underline{\underline{\boldsymbol{B}}}\,\underline{\underline{\widetilde{\boldsymbol{A}}}}^{-1}\underline{\underline{\boldsymbol{B}}}^T$ is the *Schur complement operator*. We employ an *upper* strategy, namely we approximate the inverse of $\underline{\underline{\boldsymbol{K}}}_{0,0}$ by the matrix $(\underline{\underline{\boldsymbol{D}}}\,\underline{\underline{\boldsymbol{U}}})^{-1}$:

$$\underline{\underline{\boldsymbol{K}}}_{0,0}^{-1} \approx \begin{bmatrix} \underline{\underline{\boldsymbol{I}}} & -\underline{\underline{\widetilde{\boldsymbol{A}}}}^{-1}\underline{\underline{\boldsymbol{B}}}^T \\ \underline{\underline{\boldsymbol{0}}} & \underline{\underline{\boldsymbol{I}}} \end{bmatrix} \begin{bmatrix} \underline{\underline{\widetilde{\boldsymbol{A}}}}^{-1} & \underline{\underline{\boldsymbol{0}}} \\ \underline{\underline{\boldsymbol{0}}} & \underline{\underline{\boldsymbol{S}}}^{-1} \end{bmatrix}. \tag{6.16}$$

Note that these two matrices are not computed explicitly, as we are computing the action of the inverse on a vector. Finally, another fieldsplit is applied to the velocity block $\underline{\underline{\widetilde{\boldsymbol{A}}}}$, where the velocity components are solved separately using a block Jacobi preconditionner, decoupling the velocity components.

Block	Description	Preconditioner	Krylov solver
$\underline{\underline{\boldsymbol{K}}}$	Overall coupled problem	`fieldsplit additive`	`gmres`
$\underline{\underline{\boldsymbol{K}}}_{1,1}$	Heat split subproblem	`gamg`	`gmres`
$\underline{\underline{\boldsymbol{K}}}_{0,0}$	Fluid split subproblem	`fieldsplit schur`	`fgmres`
$\underline{\underline{\widetilde{\boldsymbol{A}}}}$	Velocity block (split)	`jacobi`	`preonly`
$\underline{\underline{\boldsymbol{S}}}$	Schur complement	`gamg`	`preonly`

Table 6.2: Description of the solvers and preconditioners used for the different blocks of the system (6.11). In monospaced font, we indicate the actual PETSc component used for the solver and preconditioner used.

Parameter	Value	Dimension
μ	0.001	$[\mathrm{kg\,m^{-1}\,s^{-1}}]$
ρ	1000	$[\mathrm{kg\,m^{-3}}]$
C_p	4178	$[\mathrm{J\,kg^{-1}\,K^{-1}}]$
β	$3 \cdot 10^{-4}$	$[\mathrm{K^{-1}}]$
k_{AH}	0.576	$[\mathrm{W\,m^{-1}\,K^{-1}}]$
g	9.81	$[\mathrm{m\,s^{-2}}]$
T_{ref}	298	[K]
h_{bl}	65	$[\mathrm{W\,m^{-2}\,K^{-1}}]$
h_{amb}	10	$[\mathrm{W\,m^{-2}\,K^{-1}}]$
E	40	$[\mathrm{W\,m^{-3}}]$
T_{bl}	310	[K]
T_{amb}	294	[K]

Table 6.3: Parameters used for the simulations, representing nominal physiological values. Sources can be found in [ON08; Wan+16].

We summarize in Table 6.2 the preconditioners used for the different blocks of the system (6.11), as well as the Krylov Subspace Method (KSP) employed to solve the linear system. These methods can be either the *Generalized Minimal Residual* (GMRES) or the *Flexible GMRES* (FGMRES) methods, depending on the block being solved. In some cases, the `preonly` solver is used, which indicates that no Krylov subspace solver is executed, rather only the associated preconditioner is applied once. More details on the setting of these preconditioners can be found in the configuration file presented in Section 7.6, or in [Sai+24c].

6.3 Verification and validation of the proposed model

In this section, we present the verification and validation of our proposed model through numerical simulations conducted using the computational framework detailed in Section 6.2. The simulations aim to verify the accuracy and reliability of our model in replicating the physiological conditions of the human eye. The set of parameters used in the simulations is listed in Table 6.3, representing nominal physiological values corresponding to a healthy subject.

To ensure the accuracy and reliability of our numerical solutions, we first perform a mesh convergence analysis, followed by a scalability study to assess the computational performance of our framework. We also compare our results with those from prior research to further validate our model.

M	$h_{\min}$	$h_{\max}$	h_{mean}	# elements	# Degree of freedom		
					T	$\vec{u}$	p
Mr0	$1.25 \cdot 10^{-4}$	$4 \cdot 10^{-3}$	$9.23 \cdot 10^{-4}$	$1.92 \cdot 10^{5}$	37,470	84,966	4,615
Mr1	$1.37 \cdot 10^{-4}$	$3.63 \cdot 10^{-3}$	$7.72 \cdot 10^{-4}$	$2.82 \cdot 10^{5}$	51,753	$1.17 \cdot 10^{5}$	6,155
Mr2	$6.54 \cdot 10^{-5}$	$1.6 \cdot 10^{-3}$	$4.67 \cdot 10^{-4}$	$7.47 \cdot 10^{5}$	$1.31 \cdot 10^{5}$	$5.9 \cdot 10^{5}$	28,548
Mr3	$3.29 \cdot 10^{-5}$	$9.59 \cdot 10^{-4}$	$4.17 \cdot 10^{-4}$	$1.4 \cdot 10^{6}$	$2.42 \cdot 10^{5}$	$7.08 \cdot 10^{5}$	34,304
Mr4	$2.55 \cdot 10^{-5}$	$5.29 \cdot 10^{-4}$	$2.88 \cdot 10^{-4}$	$6.04 \cdot 10^{6}$	$1.03 \cdot 10^{6}$	$1.02 \cdot 10^{6}$	48,534
Mr5	$3.12 \cdot 10^{-5}$	$1.5 \cdot 10^{-4}$	$2.77 \cdot 10^{-4}$	$4.39 \cdot 10^{7}$	$7.37 \cdot 10^{6}$	$4.62 \cdot 10^{6}$	$2.05 \cdot 10^{5}$
Mr6	$2.82 \cdot 10^{-5}$	$9.94 \cdot 10^{-7}$	$1.84 \cdot 10^{-4}$	$1.51 \cdot 10^{8}$	$2.52 \cdot 10^{7}$	$1.47 \cdot 10^{7}$	$6.37 \cdot 10^{5}$

Table 6.4: Characteristics of meshes used for the convergence study and number of degrees of freedom for temperature T, velocity $\vec{u}$, and pressure p respectively, using $\mathbb{P}_1$–$\mathbb{P}_2\mathbb{P}_1$ elements.

6.3.1 Mesh convergence analysis: ensuring accuracy and reliability

To verify the accuracy of our numerical solution, we conduct a mesh convergence study. We generate a series of meshes, denoted `Mr0` to `Mr5`, with progressively increasing levels of refinement, following the procedure described in Section 2.1.2. The characteristics of these meshes, including the minimum, maximum, and mean element sizes, the number of elements, and the degrees of freedom for temperature, velocity, and pressure fields, are summarized in Table 6.4.

Using these meshes, we solve the model equations with consistent parameters and boundary conditions. We extract quantities of interest from the solutions, namely the maximum temperature of the cornea and the mean velocity of the aqueous humor in the anterior chamber. The evolution of these quantities with respect to the number of degrees of freedom is presented in Figure 6.2.

As shown in Figure 6.2, both the maximum corneal temperature and the mean aqueous humor velocity converge toward asymptotic values as the mesh is refined. This convergence indicates that our numerical solution becomes independent of the mesh size, confirming the accuracy and reliability of the simulation results. We observe that beyond a certain mesh density, further refinement results in negligible changes in the computed quantities, suggesting that an optimal mesh size can be selected to balance accuracy and computational cost.

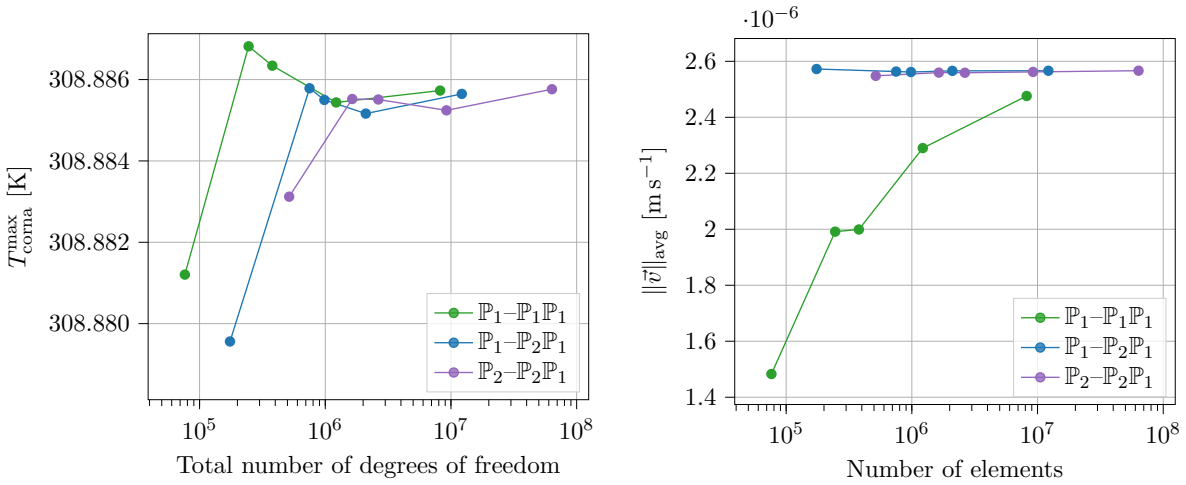
Based on these results, we select mesh `Mr4` for subsequent simulations, using the $\mathbb{P}_1$–$\mathbb{P}_2\mathbb{P}_1$ discretization, as it provides a good compromise between accuracy and computational efficiency.

6.3.2 Speed-up and scalability study

We assess the scalability of our computational framework by measuring the execution time required to solve the model as a function of the number of MPI parallel processes utilized. This analysis provides insights into the efficiency and performance of our implementation on parallel computing architectures.

Impact of the postural orientation We first examine the scalability in the context of different postural orientations of the eye—standing, prone, and supine. The execution times required to assemble and solve the non-linear problem (as outlined in Algorithm 10) for each posture are presented in Figure 6.3(a). As anticipated, the execution time decreases with an increasing number of parallel processes, demonstrating the benefits of parallelization.

Notably, the simulation for the standing position requires more time compared to the prone and supine positions. This difference is attributed to the higher fluid velocities observed in the standing position, which necessitate more non-linear iterations for convergence (five iterations compared to four for the other positions).



(a) Maximal temperature of the cornea.

(b) Mean fluid velocity.

Figure 6.2: Results of the mesh convergence study. For each curve, the point on the left corresponds to mesh `Mr1`, and the point on the right to mesh `Mr5`. The reference quantity use to compute the relative error as been obtained with the mesh `Mr5` and the discretization $\mathbb{P}_2$–$\mathbb{P}_2\mathbb{P}_1$, and a higher order of accuracy required to the solver.

We compute the speed-up $s = t_1/t_{\texttt{np}}$, where t_1 is the execution time with a single process, and $t_{\texttt{np}}$ is the time with `np` parallel processes. The results, depicted in Figure 6.3(b), show that while the speed-up improves with additional processes, it deviates from the ideal linear speed-up, particularly at higher process counts.

An in-depth analysis of the speed-up, using various numbers of processes as reference points, is provided in Figure 6.3(c). The plots reveal that although the speed-up is not perfectly optimal, it tends to improve relatively as the number of processes increases.

An interactive version of this plot is available online[2]. For each case, we compute the slope of the linear regression of the speed-up, and we scale it so that the optimal speed-up is 1. The values are presented on the right of the plot, for each postural orientation.

Overall, our computational framework demonstrates effective scalability across different simulation scenarios, with parallelization yielding significant reductions in execution time.

Scalability analysis of simulation components Beyond the assembly and solution phases, we investigate the scalability of other key components of the simulation. Focusing on the three largest meshes (`Mr4`, `Mr5`, and `Mr6`), we measure the execution times for the following steps:

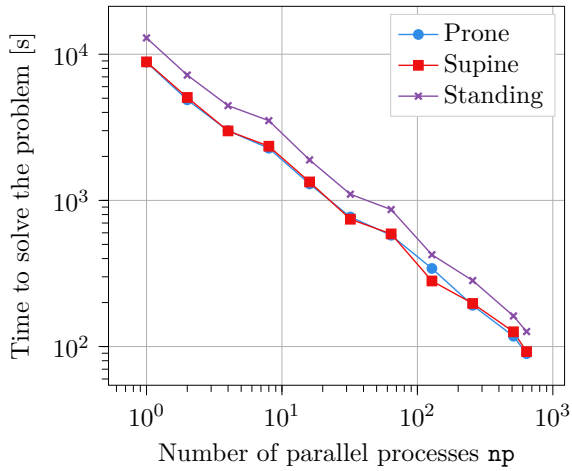
(i) Load and initialize the mesh that is already partitioned on the disk: this step involves reading the pre-partitioned mesh data from the disk and initializing the mesh structure in memory.

(ii) Initialize the data structures: this step sets up the necessary data structures required for the simulation, including matrices, vectors, and other computational entities.

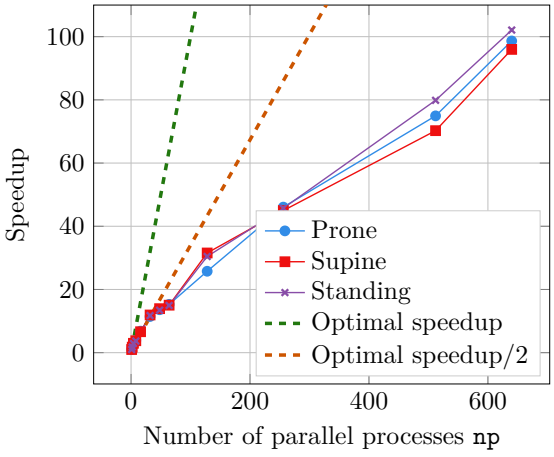
(iii) Assembly the algebraic objects of the linear system: this step involves assembling the linear system of equations that arise from the discretization of the governing equations.

(iv) Solve the non-linear algebraic system: this step involves solving the non-linear equations that arise from the discretization of the governing equations using iterative solvers.

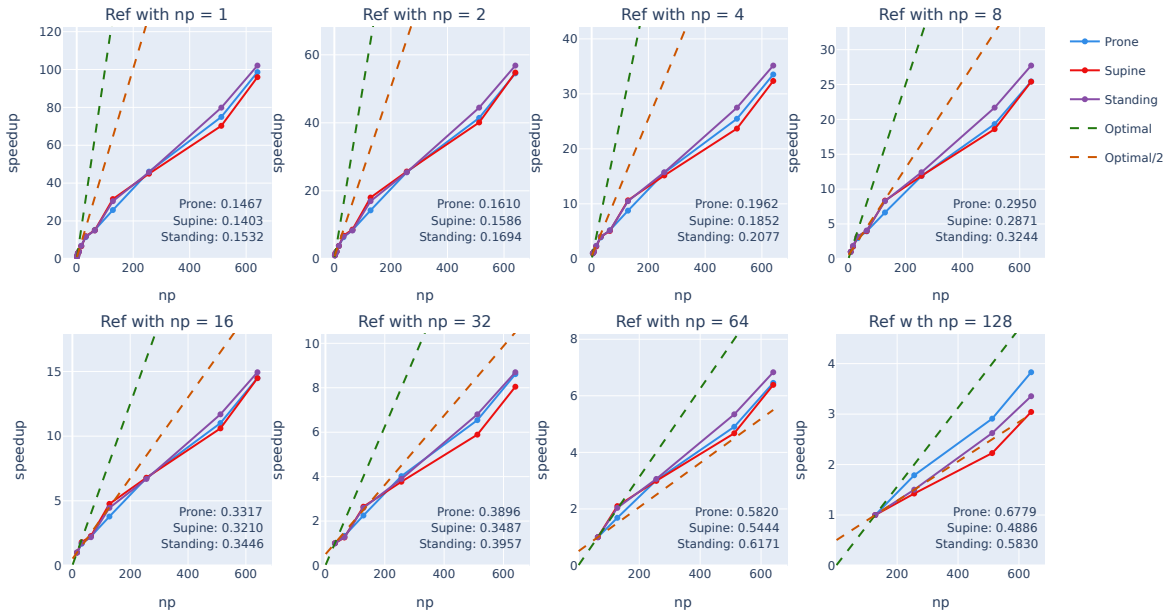
[2] https://irma.math.unistra.fr/~saigre/ressources/speedup.html



(a) Time to solve the non-linear problem (Algorithm 10).

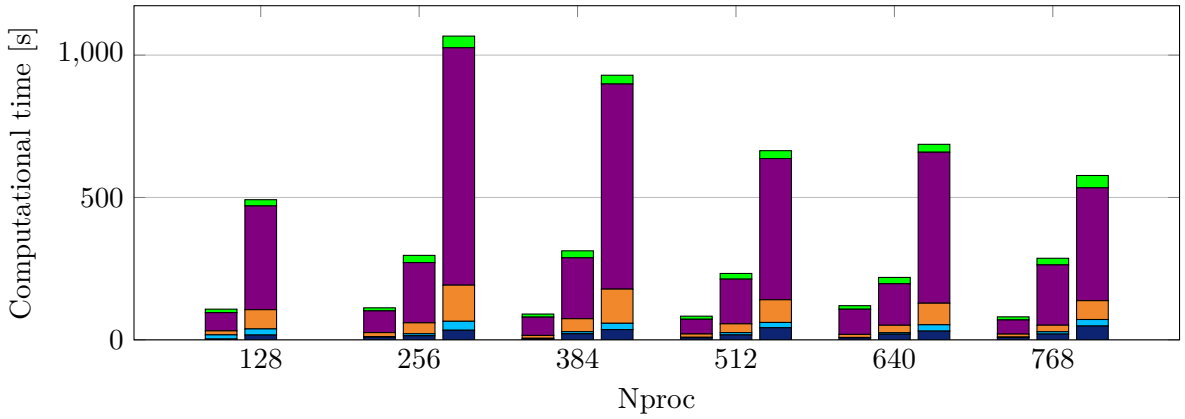


(b) Speed-up $s = t_1/t_{\mathtt{np}}$.

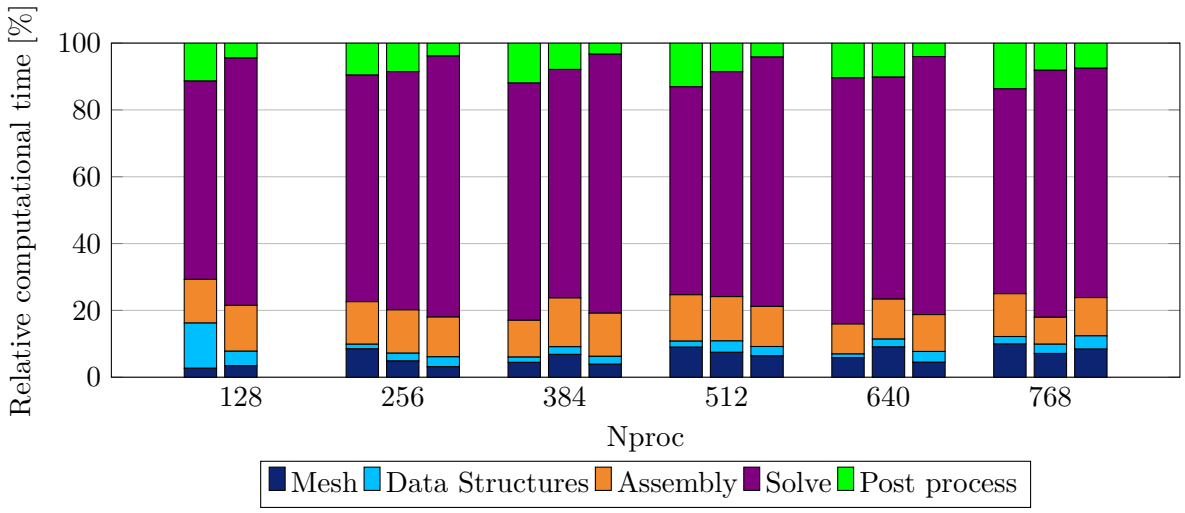


(c) Speed-up with for various number of processes as reference time, interactive figure is accessible[2]. The optimal speed-up is 1, and the values presented are the slope of the linear regression of the speed-up.

Figure 6.3: Scalability results: time of execution to simulate coupled model, and corresponding speed-up, for an increasing number of parallel processes, and for the three subject positions.



(a) Absolute computational time in each component.



(b) Relative time in each component.

Figure 6.4: Absolute and relative computational time for the coupled heat-fluid test case in the standing position, performed on Gaya with the meshes `Mr4` (left), `Mr5` (middle), and `Mr6` (right).

(v) Export the results: this step writes the computed results, such as temperature, velocity, and pressure fields, to disk in the specified output format.

The execution times and their relative contributions are summarized in Figure 6.4. We selected for this analysis the standing position, which is the most challenging case. The resolution of the non-linear system is the most time-consuming step, followed by the assembly phase. As the number of processes increases, we observe a decrease in execution times for most components, except for the result export phase, where I/O operations become a bottleneck due to the larger data volumes and potential disk access contention.

These findings highlight the importance of optimizing not only the computational algorithms but also the data management strategies, particularly for large-scale simulations.

6.3.3 Validation in comparison with previous studies

To validate our model, we compare our results with those from previous studies that have investigated temperature distributions and AH flow in the human eye. Notably, [Sco88] developed a 2D heat transfer model, [ON08] extended this to a 2D model coupling heat transfer with AH

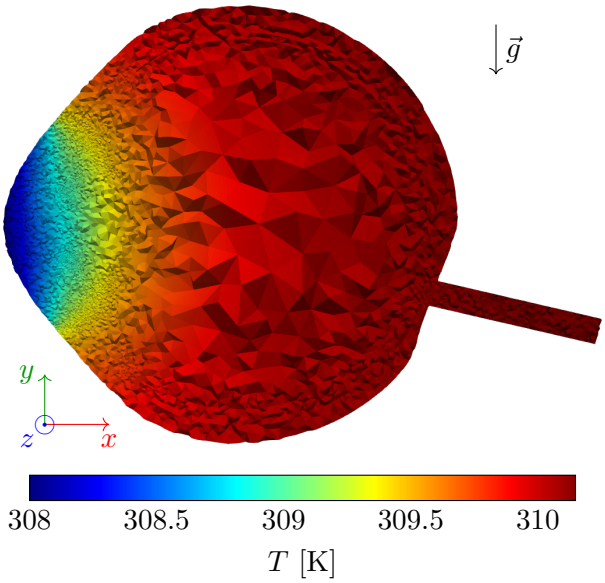


Figure 6.5: Distribution of the computed temperature over the eyeball in the standing position, on a vertical cut. Mesh discretization is also presented.

flow, and [KS10; Wan+16] presented a 3D coupled model, providing corneal temperatures for various ambient temperatures (T_{amb}). Note that the first two studies only consider the standing position of the subject.

Figure 6.5 illustrates the computed temperature distribution over the eyeball in the standing position, depicted on a vertical cross-sectional plane. As expected, the temperature is higher in the posterior part of the eye, which is insulated within the body, and lower in the anterior region, where heat exchange with the ambient air occurs through the cornea. This temperature gradient aligns with observations reported in previous studies [ON08; Wan+16], reinforcing the validity of our model.

Table 6.5 presents the corneal surface temperatures from these studies alongside our findings for the three postural configurations. Consistent with previous research, we observe that coupling the AH flow leads to an increase in the corneal surface temperature in the standing position. While [KS10] reported a temperature difference of approximately 0.2 K due to the flow, our model predicts a smaller difference of about 0.05 K. This discrepancy may be attributed to differences in model assumptions, boundary conditions, or computational methods.

Moreover, our results indicate that the lying positions (prone and supine) have minimal impact on the corneal temperature, with differences less than 0.01 K. Interestingly, we observe a slight decrease in corneal temperature in the prone position compared to the supine position, which can be explained by the flow patterns influenced by gravity, see Section 6.3.

Additionally, experimental studies compiled by [EYB89] and [NO06] report an average corneal surface temperature of approximately 307.15 K, which falls within the range of our simulation results, further supporting the validity of our model. Note that the value of T_{amb} was not specified in the publications, but still, our findings lie within the interval of results reported.

We present in Figure 6.6 the velocity field and pressure distribution in the AC and PC for the three postural orientations. Note that as the pressure is defined up to a constant, we adjust it to a nominal value of 15.5 mmHg for comparison purposes, as this is a typical value for the intraocular pressure in healthy eyes. In [Wan+16], the authors reported the pressure and velocity distributions for two postural orientations (standing and supine), we present in Table 6.6 a comparison of our results with theirs. Even though the values are different, we observe similar

Author	T_{amb}	No AH flow	AH flow coupled		
			Prone	Supine	Standing
Scott [Sco88] (2D)	293.15	306.4	–	–	–
Ooi and Ng [ON08] (2D)	298	306.45	–	–	306.9
Karampatzakis and Samaras [KS10] (3D)	293	306.81	–	–	307.06
	296	307.33	–	–	307.51
	298	307.69	–	–	307.83
Current model (3D)	293	306.5647	306.56915	306.55899	306.63672
	296	307.09845	307.10175	307.09436	307.14651
	298	307.45746	307.46008	307.45432	307.49222

Table 6.5: Corneal surface temperature for various configurations. Temperatures are given in K. A red value represents the highest temperature, and a blue value the lowest.

Position	Reference	Maximum velocity [$m\,s^{-1}$]	Average velocity [$m\,s^{-1}$]	Pressure [mmHg]
Supine	[Wan+16]	$9.44 \cdot 10^{-4}$	$4.1 \cdot 10^{-5}$	13.50 – 13.58
	[Mur+23]	$6 \cdot 10^{-5}$	n/a	n/a
	[BBS20]	n/a	$9.88 \cdot 10^{-6}$	n/a
	Current model	$2.59 \cdot 10^{-5}$	$3.21 \cdot 10^{-6}$	15.42 – 15.59
Standing	[Wan+16]	$9.6 \cdot 10^{-4}$	$2.5 \cdot 10^{-4}$	13.50 – 13.59
	[BBS20]	n/a	$5.88 \cdot 10^{-5}$	n/a
	Current model	$2.76 \cdot 10^{-4}$	$5.23 \cdot 10^{-5}$	15.28 – 15.72

Table 6.6: Comparison of the results of the current model with the literature.

orders of magnitude and trends in the pressure and velocity distributions, which further validate our model.

6.4 Numerical results

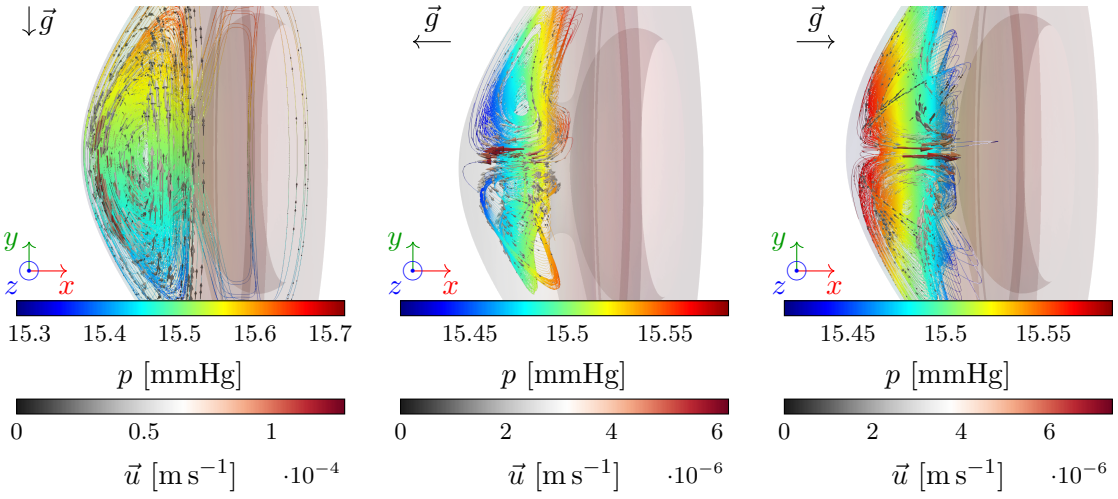
In this section, we present the results of our numerical simulations conducted using the heatfluid toolbox of Feel++ [Pru+24b]. The simulations aim to investigate the impact of the subject's position on the aqueous humor flow and temperature distribution within the eye, as well as to analyze the wall shear stress (WSS) distributions and their implications for ocular physiology.

For further details on the configuration files used to set the preconditioner and solver, we refer to Section 7.6. The configuration files and the mesh family `Mr` utilized in the simulations are available as open-access resources [Sai+24c].

6.4.1 Impact of the position of the subject

We examine how different postural orientations—standing, prone, and supine—affect the flow of the AH in the AC. Figure 6.6 presents the simulation results for each position, illustrating the flow patterns and pressure distributions.

In the standing position (Figure 6.6(a)), gravity significantly influences the flow, resulting in higher velocities and a pronounced downward movement of the aqueous humor. This enhanced



(a) Standing position. (b) Prone position. (c) Supine position.

Figure 6.6: Results of simulation for various postural orientations of the eye. Streamlines are colored according to the pressure, and the arrows show the fluid velocity magnitude.

flow contributes to the formation of characteristic patterns such as Krukenberg's spindle and recirculation zones within the AC, consistent with observations in the literature [Abd+21; Wan+16; Mur+23].

In the prone and supine positions (Figures 6.6(b) and 6.6(c)), the flow patterns are altered due to the change in the direction of gravity relative to the eye. In the prone position, the flow moves from the back of the AC towards the cornea, slightly warming the cornea. Conversely, in the supine position, the flow moves from the cornea towards the back of the AC, leading to a slight cooling effect on the cornea.

These variations in flow patterns and velocities directly impact the temperature distribution within the eye, as discussed in Section 6.3.3, and have potential implications for ocular health and treatment strategies.

6.4.2 Wall shear stress and its implications in ocular physiology

The wall shear stress (WSS) is a critical parameter representing the tangential force per unit area exerted by the fluid on the wall due to viscous effects. The insights gained from WSS analysis can inform clinical practices, contributing to personalized medicine, surgical optimization, and improved device design [Yan+22; Fer+18]. Understanding the WSS distribution has significant implications in ocular physiology, including:

- **Drug delivery:** High WSS regions may enhance the mixing and transport of drug particles within the AH, potentially increasing drug absorption rates through ocular tissues, see [Xu+13; Kou16; SZM13]. By identifying these regions, drug delivery systems can be designed to target specific areas within the eye, improving therapeutic outcomes.

- **Surgical procedures:** WSS analysis provides valuable insights for optimizing surgical interventions, [Kud+20; Bas+24]: *(i)* **Design optimization:** Knowledge of WSS distributions helps in planning procedures that minimize mechanical stresses on ocular tissues, reducing the risk of tissue damage; *(ii)* **Implantable devices:** Designing intraocular lenses and drainage devices that account for WSS can help prevent endothelial cell loss

and maintain corneal health [Rep+15; Bas+24]; *(iii)* **Postoperative outcomes:** Monitoring changes in WSS after surgery can help predict healing responses and the risk of complications.

Definition of wall shear stress

The *wall shear stress* $\vec{\tau}_w$ at a point on the wall is defined as the magnitude of the tangential component of the stress tensor acting on the wall:

$$\vec{\tau}_w(\vec{u}, p) = \underline{\underline{\sigma}}(\vec{u}, p)\vec{n}\Big|_{\text{wall}} - \left(\underline{\underline{\sigma}}(\vec{u}, p)\vec{n}\Big|_{\text{wall}} \cdot \vec{n}\right)\vec{n}, \tag{6.17}$$

where:

- $\underline{\underline{\sigma}}$ is the Cauchy stress tensor,
- $\vec{n}$ is the unit outward normal vector.

The stress tensor $\underline{\underline{\sigma}}(\vec{u}, p)$ has been defined in Equation (6.2). Substituting the expression for $\underline{\underline{\sigma}}$ into Equation (6.17), the wall shear stress becomes:

$$\vec{\tau}_w(\vec{u}, p) = 2\mu\, \underline{\underline{D}}(\vec{u})\vec{n}\Big|_{\text{wall}} - 2\mu \left(\underline{\underline{D}}(\vec{u})\vec{n}\Big|_{\text{wall}} \cdot \vec{n}\right)\vec{n}. \tag{6.18}$$

Under the no-slip boundary condition at the wall, the fluid velocity relative to the wall is zero, but the velocity gradient (shear rate) perpendicular to the wall is generally non-zero. Therefore, the wall shear stress computation simplifies to:

$$\vec{\tau}_w(\vec{u}, p) = \mu \left.\frac{\partial \vec{u}_\tau}{\partial \vec{n}}\right|_{\text{wall}}, \tag{6.19}$$

where: *(i)* $\vec{u}_\tau$ is the tangential component of the velocity vector at the wall, and *(ii)* $\partial/\partial n$ denotes the derivative normal to the wall surface.

Computational considerations

It is difficult to measure WSS experimentally, and numerical simulations are a valuable tool for investigating the complex interactions between the different physical phenomena in the eye, and in particular the WSS distribution, [Kum+06; Yam+10; Qin+21].

From a computational standpoint, accurately determining the WSS necessitates *(i)* a sufficiently fine mesh resolution near the walls to capture the sharp velocity gradients present in these regions, and *(ii)* a consistent discretization strategy to ensure the accuracy of the computed WSS.

To achieve the former, we apply the mesh discretization strategy outlined in Section 2.1.2, ensuring the mesh is adequately refined near the boundaries of Ω_{AH}, as depicted in Figure 6.7. The mesh used in the following simulation is the mesh `Mr5`, described in Table 6.4.

Regarding the latter, the mesh elements at the interfaces between the domain Ω_{AH} and the surrounding tissues exhibit varying normal vectors, whose direction is not defined uniformly across all elements. Since the velocity field is approximated using a $\mathbb{P}_2$ finite element space, its gradient is naturally of order 1, but discontinuous. Accordingly, we employ a $\mathbb{P}_{1,\text{disc}}$ approximation space to compute the WSS, ensuring consistency and accuracy. In addition, we compute a continuous piece-wise linear approximation ($[L^2]^3$-projection of the discontinuous approximation) of the WSS to avoid numerical oscillations and improve the visualization of the results.

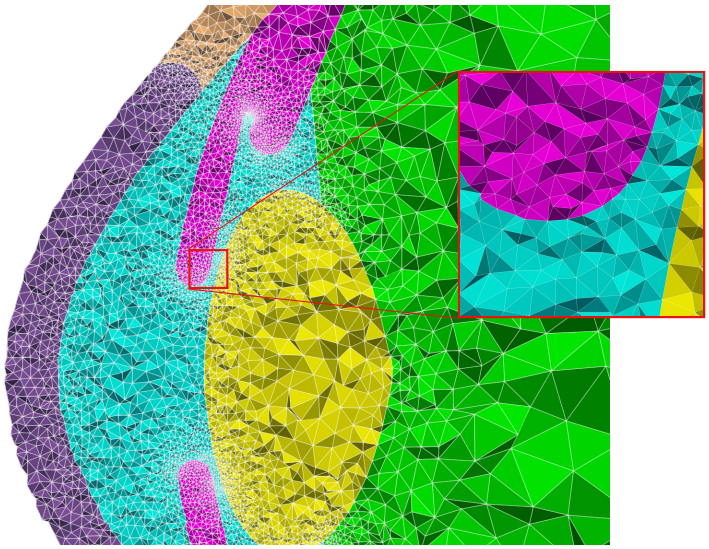
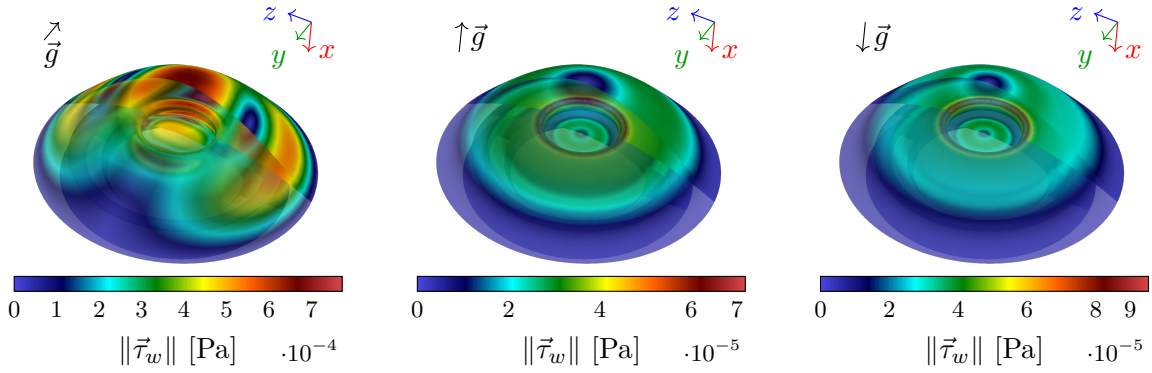


Figure 6.7: Refinement of mesh `Mr5` near the wall boundary of Ω_{AH}.

Simulation results

We compute the WSS on the surfaces of the anterior chamber to analyze the shear stresses resulting from the aqueous humor flow under different postural orientations. Figure 6.8 presents the WSS magnitude over the corneal endothelium for the standing, prone, and supine positions.



(a) Standing. (b) Prone. (c) Supine.

Figure 6.8: Wall shear stress distribution on the corneal endothelium for the three postural orientations.

We first compare its magnitude with results from the literature. Note that the authors did not employ the same model as the one presented in this work: *(i)* in [Fer+18; Qin+21], an inlet and outlet flow boundary condition was used, *(ii)* in [Kud+20], the authors imposed an inlet flow and outlet pressure boundary condition, and *(iii)* in [Rep+15], a no-slip velocity boundary condition on the wall of the anterior chamber was applied. Recall that the model presented in this work uses a no-slip boundary condition, as in [Rep+15]. The results are presented in Table 6.7.

Despite these differences in boundary conditions, our results show a similar order of magnitude for WSS compared to the literature. The slightly lower values in our model may be attributed to the no-slip condition applied at the walls, which tends to reduce shear stress near the boundaries. Moreover, when comparing the distribution of the WSS, shown in Figures 6.8(a) and 6.8(c), with

results presented in [Qin+21, Figure 5], we observe a similar pattern, with higher values near the corneal endothelium and lower values near the iris and lens surfaces.

Author	Boundary conditions	Orientation	Range of WSS
Fernández-Vigo et al. [Fer+18]	Inlet/outlet flow	n/a	$[10^{-5}, 1.7 \cdot 10^{-3}]$
Kudsieh et al. [Kud+20]	Inlet flow/outlet pressure	n/a	$[0, 1.36 \cdot 10^{-3}]$
Repetto et al. [Rep+15]	No-slip velocity	Standing	$[0, 1.63 \cdot 10^{-3}]$
Qin et al. [Qin+21]	Inlet/outlet flow	Standing Supine	$[5 \cdot 10^{-4}, 3.5 \cdot 10^{-3}]$ $[10^{-4}, 10^{-3}]$
Current model	No-slip velocity	Standing Supine Prone	$[0,\ 7.7 \cdot 10^{-4}]$ $[0,\ 9.5 \cdot 10^{-5}]$ $[0,\ 7.2 \cdot 10^{-5}]$

Table 6.7: Comparison of the magnitude of the WSS results with results from the literature [Fer+18; Kud+20; Rep+15; Qin+21], all in 3D. In [Fer+18; Kud+20], the authors did not specify the orientation of the eye (n/a). The values are given in Pa.

Building on the sensitivity analysis (SA) presented in Chapter 5, we examine the effect of ambient temperature on the WSS magnitude. Specifically, we calculate the average WSS magnitude across three surfaces within the anterior chamber as a function of ambient temperature for each of the three postural orientations: *(i)* on the corneal endothelium, denoted by Γ_{cornea}, *(ii)* on the iris surface, denoted by Γ_{iris}, and *(iii)* on the entire boundary of the anterior chamber, denoted by $\partial\Omega_{\text{AH}}$.

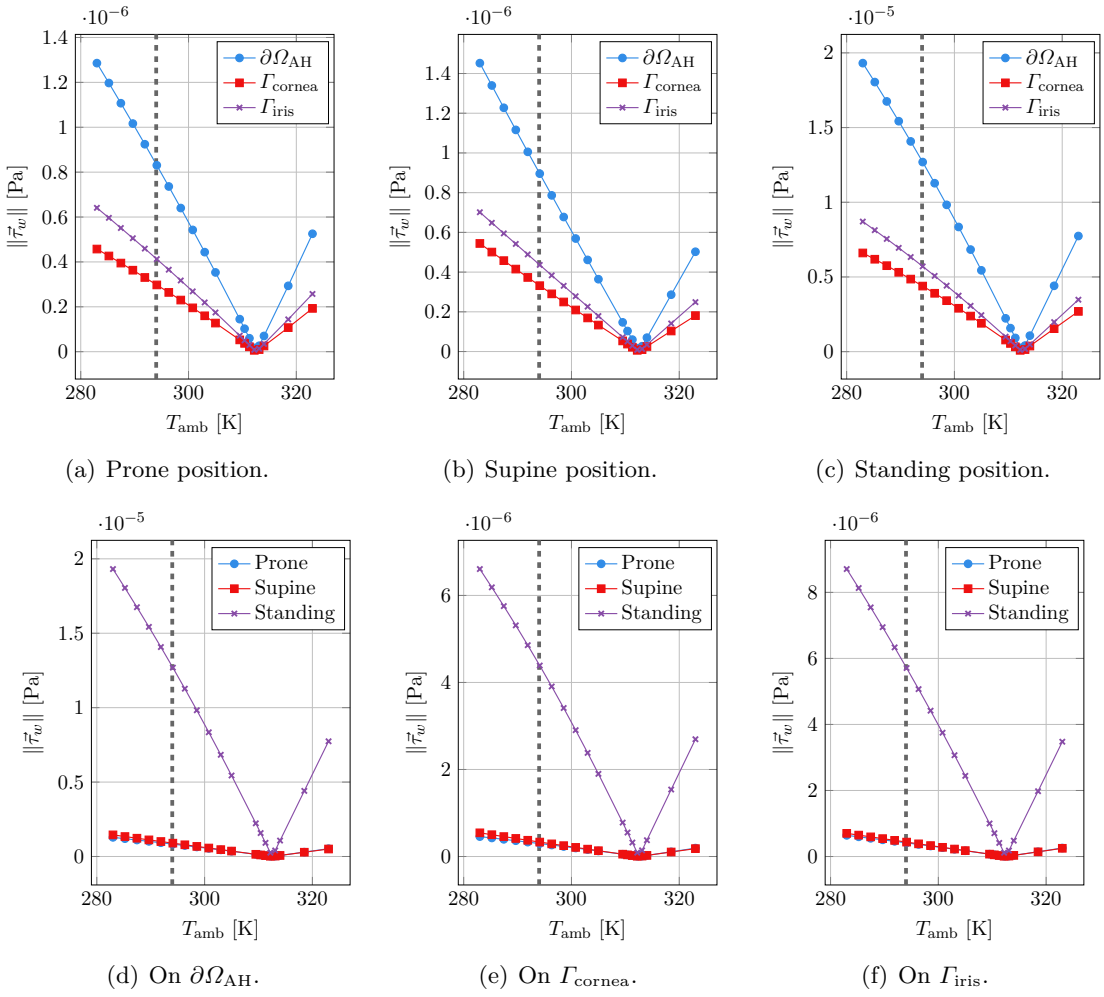
The results are presented in Figure 6.9. The first striking result is that the WSS magnitude is significantly influenced by the postural orientation or the subject: in horizontal positions (prone and supine), the WSS magnitude is ten times lower than in the standing position. This observation is coherent with the fact that for corneal surgery, after the injection of endothelial cells inside the aqueous humor or the patient, the patient is placed in prone position for three hours to enhance the adhesion of the cells to the cornea [Kin+18].

In addition, the results indicate a linear dependency of the WSS magnitude on ambient temperature, although the model is non-linear. In the three positions, the WSS magnitude reaches a minimum around $T_{\text{amb}} = 310\,\text{K}$, corresponding to the body temperature. This insight suggests that, in clinical contexts, adjusting the ambient temperature might influence cell adhesion by modulating the WSS. This minimum is reached because as there is no inflow or outflow, the temperature difference is the sole responsible for fluid flow. It raises a limitation of our model, to assess the effect of the temperature on the WSS, we should consider the effect of the inflow and outflow of the aqueous humor.

A conclusion from this study is that the WSS magnitude is significantly influenced by both postural orientation and ambient temperature. These factors should be considered in the design of ocular devices and drug delivery systems to optimize therapeutic outcomes.

6.5 Conclusion and future works

In this chapter, we have presented a comprehensive modeling and computational framework for simulating heat transfer within the human eyeball, coupled with the flow of AH in both the anterior and posterior chambers of a healthy eye. Our complex model has undergone rigorous verification and validation against numerical results from existing literature, demonstrating its accuracy and reliability.



(a) Prone position. (b) Supine position. (c) Standing position.

(d) On $\partial\Omega_{\text{AH}}$. (e) On Γ_{cornea}. (f) On Γ_{iris}.

Figure 6.9: Average of the wall shear stress magnitude as a function of the ambient temperature for the three postural orientations. The vertical line at $T_{\text{amb}} = 294\,\text{K}$ represents the baseline value as per Table 6.3.

An important novelty of our work lies in the integration of high-performance computing (HPC) techniques to solve the coupled heat transfer and fluid flow equations on the entire eye geometry, including both the anterior and posterior chambers. This holistic approach allows for a more accurate representation of intraocular phenomena compared to models that focus solely on the anterior chamber or neglect the posterior chamber. By leveraging HPC resources efficiently, we can handle the computational demands of such detailed simulations, enabling high-resolution analyses that were previously impractical.

Additionally, we have computed the wall shear stress (WSS) distributions within the eye, providing a foundational layer for future applications in drug delivery and surgical planning. The WSS analysis is crucial for understanding the mechanical forces acting on ocular tissues, which can influence drug absorption rates, endothelial cell health, and surgical outcomes. Our ability to accurately model WSS opens new avenues for optimizing therapeutic strategies that minimize adverse mechanical effects. We could even envision including the design of intraocular devices.

The simulation results show flow patterns and temperature distributions that align closely with previous numerical studies, reinforcing the validity of our approach. Notably, the model accurately captures the impact of postural orientation on flow recirculations within the eye, providing valuable insights into ocular physiology and the effects of gravity on intraocular fluid

dynamics.

Despite these successes, a primary drawback of the present model is its computational cost, which remains relatively high—requiring several minutes on the specified hardware to perform a single simulation. This computational intensity limits the feasibility of real-time simulations, which are desirable for clinical applications and interactive studies.

Future Work

To address the computational challenges, we are working on improving the preconditioner for the conjugate heat transfer problem, in particular the Schur complement preconditioner for the fluid block. The challenge is to significantly reduce the computational cost while enable many parameter or real-time evaluations. We are currently developing model order reduction techniques tailored to our problem, extending our previous work [SPS24b]. These methods, such as the (certified) reduced basis method, aim to enable real-time simulations of coupled flow and heat transfer inside the human eyeball by reducing the computational complexity while preserving essential dynamics. Implementing such techniques will significantly enhance the model's applicability in clinical settings, allowing for rapid simulations that can assist in diagnosis and treatment planning.

Another extension of this work involves incorporating AH inflow and outflow mechanisms, which were neglected in the current model under the assumption of minimal influence on overall flow dynamics. Including AH production and drainage would provide a better understanding of intraocular fluid dynamics, especially under pathological conditions such as glaucoma, where these processes are disrupted. This requires modeling AH production in the ciliary body and the trabecular meshwork's drainage function using appropriate boundary conditions and source terms.

From a clinical perspective, our framework holds significant potential for assessing the effects of topical administration of ophthalmic drugs, such as eye drops, which are the standard therapeutic approach for ocular diseases like glaucoma. Additionally, it could be instrumental in evaluating cell injection treatments for internal pathologies, such as bullous keratopathy [Kin+18]. Future research will focus on integrating drug transport models and cell injection into our simulations, enabling the study of diffusion, absorption, and interaction with ocular tissues. This integration will facilitate the development of personalized medicine approaches and improve therapeutic strategies by predicting drug efficacy and optimizing dosing regimens.

In summary, this work lays the foundation for advanced computational modeling of ocular fluid dynamics and heat transfer, with promising applications in both research and clinical practice. The ongoing developments aim to enhance the model's capabilities and usability, bringing us closer to real-time, patient-specific simulations that can inform diagnosis, treatment planning, and potentially device design in ophthalmology.

Chapter 7

Computation framework: contributions

In this chapter, we present the code contributions developed in the context of this thesis, emphasizing their alignment with principles of reproducibility and open science, as outlined in [Cam+24]. These principles are crucial for ensuring that our results can be independently validated and built upon by others. By adhering to these practices, we aim to support transparency and foster collaboration within the research community. Given the extensive nature of the full codebase, we focus on specific components that are essential for reproducing the results presented. Links to the complete, open-access code repositories are provided throughout the chapter.

The chapter is organized as follows: We present in Section 7.1 the script used to generate the geometry introduced in Section 1.3 and the mesh families as per Section 2.1. Then, Section 7.2 introduces the implementation of the Reduced Basis Method for the heat transfer model in the eye, as presented in Section 3.1. Section 7.3 presents the implementation of the Non-Intrusive Reduced Basis method, as introduced in Section 3.2. In Section 7.4, we present the code used to solve the elliptic problem in the presence of a Dirac source term, as discussed in Chapter 4. Next, Section 7.5.2 introduces the code used to perform the sensitivity analysis of the heat transfer model in the eye, as presented in Chapter 5. Finally, Section 7.6 presents the code used to solve the coupled heat transfer and aqueous humor flow model, as introduced in Chapter 6.

7.1 Geometrical model and discretization of the eye

In this section, we present the scripts implemented to generate the families of meshes that are used in simulations. The whole process is schematized in Figure 7.1. Some parts have been cropped here, but the full code, as well as the pipeline to generate the meshes from the CAD file, are available in [Cha+24]. The dataset containing the mesh families has also been published in open-access [Sai+24c].

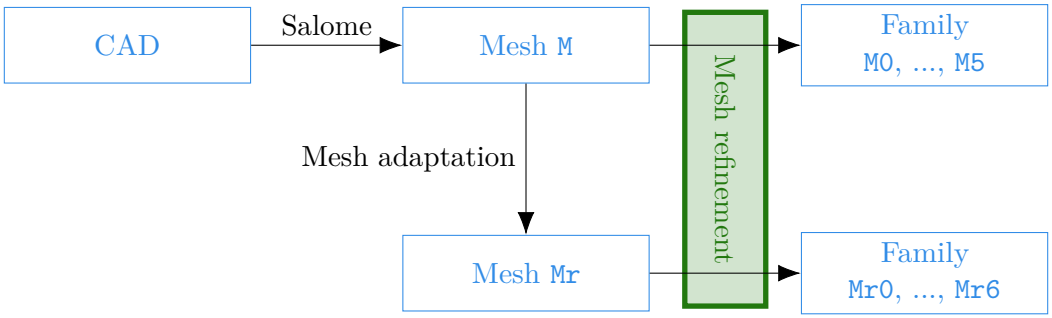


Figure 7.1: Pipeline to generate the geometry and mesh families.

7.1.1 Geometrical construction of the human eyeball

To begin with, we present the Salome script used to generate the geometry of the human eyeball, as introduced in Section 1.3.

The initial CAD model of the human eyeball, presented in Figure 1.3(a), is saved in the STEP format. First, we initialize the Salome environment, as shown in Listing 7.1. Note that a check on the version of Salome is performed to ensure compatibility with the code, as with new algorithm implemented in the software, the code may not work as expected[1].

Listing 7.1: Initialisation of Salome.

```
import salome, salome_version
from eye_utils import *
salome.salome_init()
print("This code is supposed to run with salome version 9.12.0")
print("Current Salome Version is :", salome_version.getVersion(), '\n')
```

Then, we import the CAD file and extract the different solids that compose the eye. As pointed out in Section 1.3, some volumes need to be modified to fit our needs, and some of them do not, such as the cornea, the lens, the choroid, and the iris. Listing 7.2 shows how to import the CAD file and extract the different solids.

Listing 7.2: Import the CAD.

```
Human_Eye = geompy.ImportSTEP(path_to_step_file, True)
[Cornea_h, Iris___Ciliary_Body_h, Suspensory_Ligament_h, Lens_Body_h, Vitreous_humor_h, Sclera_h,
↪  Choroid_h, Retina_h, Vein_h, Artery_h] = geompy.ExtractShapes(Human_Eye, geompy.ShapeType["SOLID"],
↪  True)

# Solids that do not need to be modified
Cornea = Cornea_h
Iris = Iris___Ciliary_Body_h
Lens = Lens_Body_h
Choroid = Choroid_h
```

[1] We actually had this issue, thanks to Christophe Bourcier and Christophe Trophime for their help!

The next step is to build the other volumes. Some auxiliary functions have been implemented, such as `build_vitreous_humor`, that builds the vitreous humor volume from the different solids extracted from the CAD file. For instance, we consider the vitreous humor, the same process applying to the other volumes. To create a volume corresponding to the full domain containing the vitreous humor, a tedious work consisting in manually selecting the various faces forming the boundary of the vitreous humor was performed. The Salome script used to build the vitreous humor domain is presented in Listing 7.3.

Listing 7.3: Script to build the vitreous humor domain.

```
def build_vitreous_humor(geompy, lens, aqueousHumor, retina, iris, choroid):
    Faces_lens = geompy.ExtractShapes(lens, geompy.ShapeType["FACE"], True)
    Faces_aqueousHumor = geompy.ExtractShapes(aqueousHumor, geompy.ShapeType["FACE"], True)
    Faces_iris = geompy.ExtractShapes(iris, geompy.ShapeType["FACE"], True)
    Faces_choroid = geompy.ExtractShapes(choroid, geompy.ShapeType["FACE"], True)
    Faces_retina = geompy.ExtractShapes(retina, geompy.ShapeType["FACE"], True)

    ShellNewVitreousHumor = geompy.MakeShell( [Faces_lens[26], Faces_lens[27], Faces_lens[28],
        Faces_lens[29], ..., Faces_iris[114], Faces_iris[122], Faces_choroid[1], Faces_choroid[2],
        Faces_choroid[3], Faces_choroid[5], Faces_retina[2], Faces_retina[3], Faces_retina[6],
        Faces_retina[7]] )
    VitreousHumor = geompy.MakeSolid([ShellNewVitreousHumor])

    return VitreousHumor
```

All the volumes are built one after another. The functions implemented to construct them are present in the file `eye_utils.py`[2]. Note that at some points, a first partition needs to be constructed to glue the different faces together. We present in Listing 7.4 the list of instruction necessary to build all the volumes.

Listing 7.4: Build the different volumes.

```
# Build the lamina cribrosa
Lamina0 = build_lamina(geompy, Retina_h, Sclera_h, distance_from_sclera, depth, shift, hole)
# Build the sclera, and the retina
Sclera0, _ = build_sclera(geompy, Sclera_h)
Retina0, Pia0, InterFaceRetina = build_retina(geompy, Retina_h, Choroid_h)

Lamina1 = translate_lamina(geompy, Lamina0, InterFaceRetina)    # translate the lamina
Retina = fill_retina(geompy, Retina0)       # fill the retina
Sclera1 = fill_sclera(geompy, Sclera0)      # fill the sclera
wholeOpticNerve, OpticNerve = build_optic_nerve(geompy, Pia0, Lamina1) # build the optic nerve

# fit the volumes
Sclera2 = modify_sclera(geompy, Sclera1, wholeOpticNerve)
Lamina = fit_lamina_to_optic_nerve(geompy, Retina, OpticNerve, wholeOpticNerve)
Sclera = fit_sclera_to_lamina(geompy, Sclera2, Lamina)

# Gather the built volumes in a partition
eye_partition0 = build_first_partition(geompy, Sclera, Retina, OpticNerve, Cornea, Iris,
↪  Suspensory_Ligament_h, Lens, Vitreous_humor_h, Choroid, Lamina, scale_factor)
[Cornea, Iris, Ligament, Lens, Vitreous_humor0, Sclera, Choroid, Retina, Lamina, OpticNerve] =
↪  geompy.ExtractShapes(eye_partition0, geompy.ShapeType["SOLID"], True)
# build the Aqueous and Vitreous Humor from the partition
Aqueous_humor = build_aqueous_humor(geompy, Cornea, Sclera, Iris, Lens, Ligament, Vitreous_humor0)
Vitreous_humor = build_vitreous_humor(geompy, Lens, Aqueous_humor, Retina, Iris, Choroid)
```

Finally, the partition with all the constructed volumes can be assembled to create the full geometry, as shown in Listing 7.5. At this point we export it to another STEP file, that will be used in a second Salome script to generate the mesh.

Listing 7.5: Build the full geometry.

[2] https://github.com/feelpp/mesh.eye/blob/main/eye_utils.py

```
eye0 = geompy.MakePartition( [Cornea, Aqueous_humor, Iris, Lens, Vitreous_humor, Sclera, Choroid, Retina,
↪  Lamina, OpticNerve], [], [], [], geompy.ShapeType["SOLID"], 0, [], 0)
geompy.ExportSTEP(eye0, "Eye.step", GEOM.LU_METER )
```

To generate the new STP file, the script can be run with the following command. The optional argument `-t` is used to run the script in the terminal mode, without the graphical interface of Salome. The geometrical arguments can be passed as optional arguments to the script. In the work of this thesis, we did not incorporate them, but more details about them can be found in [Sal+23].

```
$ salome [-t] construct-eye.py [--GEOMETRICAL_ARGUMENTS]
```

After the execution of this first script, the new geometry, depicted in Figure 1.3(b), is saved in the file `Eye.step`. This new CAD file is used in the next step to generate the mesh.

7.1.2 Discrete representation: mesh construction

Now, we delve into the mesh generation described in Section 2.1.1. Another Salome script, `eye.py` is used to generate the mesh. The whole script is also available in the repository [Cha+24]. After having initialized the Salome environment like in Listing 7.1, we import the geometry from the newly generated STP file, and extract the different solids that compose the eye, as shown in Listing 7.6.

Listing 7.6: Import the geometry.

```
Eye = geompy.ImportSTEP("Eye.step", True)
[Cornea, AqueousHumor, Iris, Lens, VitreousHumor, Sclera, Choroid, Retina, Lamina, OpticNerve] =
↪  geompy.ExtractShapes(Eye, geompy.ShapeType["SOLID"], True)
Solids = [Cornea, AqueousHumor, Iris, Lens, VitreousHumor, Sclera, Choroid, Retina, Lamina, OpticNerve]
Solids = sorted(Solids, key=lambda solid: solid.GetName() )
```

To set up the geometry for meshing purposes, we need to define the interfaces between the different tissues. The Salome script used to build the interfaces is presented in Listing 7.7. The main idea of this script is to loop over all the tissues and to find the faces that are shared between two of them. To ensure that the interfaces are defined once, we use the lexical order of the names of the tissues to define the name of the interface. Finally, we create groups of faces for the external faces of the tissues, that will be used to define the boundary conditions in the simulations. The following convention of naming the interfaces is used: `tissue1_tissue2`, where `tissue1` and `tissue2` are the names of the tissues in the lexical order.

Listing 7.7: Script to build the interfaces between the different tissues.

```
Solids = [Cornea, AqueousHumor, Iris, Lens, VitreousHumor, Sclera, Choroid, Retina, Lamina, OpticNerve]
Solids = sorted(Solids, key=lambda solid: solid.GetName() )
Interfaces = []
Others = []
Interface_ = {}

for i, solid1 in enumerate(Solids):
  Interfaces_solid1 = []
  Name1 = solid1.GetName()
  for j in range(0, len(Solids)):
    if i != j:
      solid2 = Solids[j]
      Name2 = solid2.GetName()
      faces = geompy.GetSharedShapesMulti([solid1, solid2], geompy.ShapeType["FACE"], False)
      if len(faces) > 0:
        objName = Name2 + "_" + Name1 if Name1 > Name2 else Name1 + "_" + Name2
        Interface_[(Name1, Name2)] = geompy.CreateGroup(solid1, geompy.ShapeType["FACE"], objName)
        geompy.UnionList(Interface_[(Name1, Name2)], faces)
        Interfaces_solid1.append(Interface_[(Name1, Name2)])
  Interfaces.append(Interfaces_solid1)
```

After creating the interfaces, we gather all the other faces that are not shared between two tissues, the *free groups*. The script used to build the free groups is presented in Listing 7.8.

Listing 7.8: Script to build the free groups.

```
for i, solid1 in enumerate(Solids):
    Name1 = solid1.GetName()
    if "Cornea" in Name1:
        FCor = geompy.ExtractShapes(solid1, geompy.ShapeType["FACE"], True)
        optic_faces = geompy.CreateGroup(solid1, geompy.ShapeType["FACE"], "BC_%s"%(Name1))
        geompy.UnionList(optic_faces, [FCor[0], FCor[1]])
        Others.append(optic_faces)
    elif "OpticNerve" in Name1:
        FOpt = geompy.ExtractShapes(solid1, geompy.ShapeType["FACE"], True)
        optic_faces = geompy.CreateGroup(solid1, geompy.ShapeType["FACE"], "BC_%s"%(Name1))
        geompy.UnionList(optic_faces, [FOpt[5], FOpt[6], FOpt[7]])
        Others.append(optic_faces)
    elif "Sclera" in Name1:
        FScl = geompy.ExtractShapes(solid1, geompy.ShapeType["FACE"], True)
        optic_faces = geompy.CreateGroup(solid1, geompy.ShapeType["FACE"], "BC_%s"%(Name1))
        geompy.UnionList(optic_faces, [FScl[6], FScl[7], FScl[12], FScl[13], FScl[19]])
        Others.append(optic_faces)
    elif "Lamina" in Name1:
        [Hole, In, Lateral1, Lateral0, Out] = geompy.ExtractShapes(solid1, geompy.ShapeType["FACE"],
        ↪  True)
        Lateral = geompy.CreateGroup(Lamina, geompy.ShapeType["FACE"], "Lateral")
        geompy.UnionList(Lateral, [Lateral0, Lateral1])
        In_ = geompy.CreateGroup(Lamina, geompy.ShapeType["FACE"], "In")
        geompy.UnionList(In_, [In])
        Hole_ = geompy.CreateGroup(Lamina, geompy.ShapeType["FACE"], "Hole")
        geompy.UnionList(Hole_, [Hole])
        Out_ = geompy.CreateGroup(Lamina, geompy.ShapeType["FACE"], "Out")
        geompy.UnionList(Out_, [Out])
```

The final step is to create the mesh. First, we create the mesh object, and set up the meshing algorithm, namely NETGEN. Then, we create the groups of faces that will be used to define the boundary conditions in the simulations. The script used to set up the mesh is presented in Listing 7.9.

Listing 7.9: Mesh generation.

```
import SMESH, SALOMEDS
from salome.smesh import smeshBuilder
smesh = smeshBuilder.New()
EyeMesh = smesh.Mesh(Eye)
EyeMesh.SetName("Eye_Mesh")
NETGEN_1D_2D_3D = EyeMesh.Tetrahedron(algo=smeshBuilder.NETGEN_1D2D3D)
```

Then, we set the markers for the different tissues and interfaces, for the volumes elements (Listing 7.10) and for the interfaces and free groups (Listing 7.11).

Listing 7.10: Set markers for the different tissues and interfaces.

```
Retina_mesh = EyeMesh.GroupOnGeom(Retina, 'Retina', SMESH.VOLUME)
Choroid_mesh = EyeMesh.GroupOnGeom(Choroid, 'Choroid', SMESH.VOLUME)
Vitreous_humor_mesh = EyeMesh.GroupOnGeom(VitreousHumor, 'VitreousHumor', SMESH.VOLUME)
Lens_mesh = EyeMesh.GroupOnGeom(Lens, 'Lens', SMESH.VOLUME)
Iris_mesh = EyeMesh.GroupOnGeom(Iris, 'Iris', SMESH.VOLUME)
AqueousHumor_mesh = EyeMesh.GroupOnGeom(AqueousHumor, 'AqueousHumor', SMESH.VOLUME)
Lamina_mesh = EyeMesh.GroupOnGeom(Lamina, 'Lamina', SMESH.VOLUME)
OpticNerve_mesh = EyeMesh.GroupOnGeom(OpticNerve, 'OpticNerve', SMESH.VOLUME)
Cornea_mesh = EyeMesh.GroupOnGeom(Cornea, 'Cornea', SMESH.VOLUME)
Sclera_mesh = EyeMesh.GroupOnGeom(Sclera, 'Sclera', SMESH.VOLUME)
```

Listing 7.11: Set markers for the interfaces and free groups.

```
Out_1 = EyeMesh.GroupOnGeom(Out,'Out',SMESH.FACE)
Out_1.SetName( 'Lamina_Out' )
Hole_1 = EyeMesh.GroupOnGeom(Hole,'Hole',SMESH.FACE)
Hole_1.SetName( 'Lamina_Hole' )
In_1 = EyeMesh.GroupOnGeom(In,'In',SMESH.FACE)
In_1.SetName( 'Lamina_In' )
Lateral_1 = EyeMesh.GroupOnGeom(Lateral,'Lateral',SMESH.FACE)
Lateral_1.SetName( 'Lamina_Lateral' )

Done = []
for item in Others:
    Name = item.GetName()
    if Name not in Done:
        Done.append(Name)
        BC_Group_Mesh = EyeMesh.GroupOnGeom(item,item.GetName(),SMESH.FACE)
Done = ["Lamina"]
for interface in Interfaces:
    for item in interface:
        Name = item.GetName()
        if not Name in Done and "Lamina" not in Name:
            Done.append(Name)
            BC_Group_Mesh = EyeMesh.GroupOnGeom(item,Name,SMESH.FACE)
```

Finally, we actually generate the mesh in Listing 7.12.

Listing 7.12: Mesh generation.

```
try:
    isDone = EyeMesh.Compute()
    EyeMesh.ExportMED( "mesh/Eye_Mesh3D.med", 0, SMESH.MED_V2_2, 1, None, 1 )
    print(EyeMesh.Dump())
    print('Mesh built successfully')
except Exception as e:
    print(f"Failed to create Mesh : {e}")
    sys.exit(1)
```

The command to run the script is the following:

```
$ salome [-t] eye.py
```

It results in a mesh in the MED format. As is, this mesh cannot be used in the framework of Feel++, so we convert it to the `msh` format, with GMSH [GR09]:

```
$ gmsh Eye_Mesh3D.med -0 -o Eye_Mesh3D.msh
```

The mesh generated is shown in Figure 2.1(a). We present in Listing 7.13 the characteristics of the mesh, such as the number of nodes, edges, faces, and volumes.

Listing 7.13: Dump content of the mesh.

```
========================== Dump contents of mesh ==========================
1) Total number of nodes:       81634
2) Total number of edges:       8388
3) Total number of faces:       86622
4) Total number of polygons:    0
5) Total number of volumes:     464417
6) Total number of polyhedrons: 0

7) Total number of linear edges:    8388
8) Total number of linear faces:    86622
8.1) Number of linear triangles:    86622
8.2) Number of linear quadrangles:  0
9) Total number of linear volumes:  464417
9.1) Number of linear hexahedrons:  0
9.2) Number of linear tetrahedrons: 464417
9.3) Number of linear prisms:       0
9.4) Number of linear pyramids:     0

10) Total number of quadratic edges:   0
```

```
11) Total number of quadratic faces:   0
12) Total number of quadratic volumes: 0
```

7.1.3 Mesh refinement strategy

In Section 2.1.2, we introduced the mesh refinement strategy, consisting in *(i)* refining the mesh around the PC and AC, and *(ii)* generating two families of meshes with different refinement levels. The Python script provided in Listing 7.14 allows to generate a family of meshes, of various mesh refinement levels. It consists in loading the initial mesh, and refining it with a metric that is defined by the refinement level. The metric is only set by a characteristic length.

Listing 7.14: Generation of the mesh family `M`.

```python
import sys, os
import feelpp.core as fppc
cwd = os.getcwd()
e = fppc.Environment(sys.argv, config=fppc.localRepository("remesh"))
m = fppc.mesh(dim=3, realdim=3)
old_mesh = fppc.load(m, "Eye_Mesh3D.msh")
for idx, refinement in enumerate(["10", "5", "1", "0.5", "0.25", "0.125"]):
    mesh, cpt = fppc.remesh(old_mesh, metric=refinement)
    export_dir = os.path.join(cwd, "M", f"M{idx}")
    mesh_r.saveHDF5(os.path.join(export_dir, "Eye_Mesh3D.json"))
```

For the coupled model presented in Chapter 6, a second family of meshes is generated, with a specific refinement around the PC and AC. This kind of refinement can be achieved with options set in the Feel++ toolboxes. The refinement presented in Listing 7.15 is defined as follows:

- `hfar` is the size of the elements far from the PC and AC,
- `hclose` is the size of the elements close to the PC and AC,
- `d2r_wall` is the distance to the range of the wall,
- `mymetric_wall` is the metric used for the mesh adaptation.

Listing 7.15: Mesh refinement around the PC and AC.

```
{
  "Parameters": {
    "hfar": "h:h",
    "hclose": "hfar/3:hfar",
    "d2r_wall": "min(meshes_heatfluid_distanceToRange_wall_normalized_min_max, 0.2) /
    ↪  0.2:meshes_heatfluid_distanceToRange_wall_normalized_min_max",
    "mymetric_wall": "hclose+(hfar-hclose)*d2r_wall:hclose:hfar:d2r_wall"
  },
  "Meshes":
  {
    "heatfluid": {
      "DistanceToRange": {
        "wall": {
          "markers": ["AqueousHumor_Lens", "AqueousHumor_Cornea", "AqueousHumor_Iris"],
          "normalization": "min_max"
        }
      },
      "MeshAdaptation": [{
        "metric":"mymetric_wall:mymetric_wall",
        "events":{ "after_import": {} }
      }]
    }
  }
}
```

The command to run the script is the following:

```
$ mpirun -np 12 feelpp_toolbox_heat --config-file refinement_aqueous_humor.cfg
```

Note that this code runs a whole simulation of heat transfer, but the only purpose is to generate the adapted mesh. At this point, the tool to adapt the mesh is only included in the simulation toolboxes, and not as a standalone tool.

Table 2.2 presents the performance of the mesh refinement strategy, and the characteristics of the families of meshes are reported in Table 2.1. The resulting mesh, denoted `Mr`, is shown in Figure 2.1(b).

The families of meshes generated are available in open-access [Sai+24c].

7.2 Reduced order modeling with the reduced basis method

In this section, we present the code developed to build the reduced basis model of the eye model, as described in Section 3.1. The complete code can be found in the Feel++ repository[3].

To begin with, we need to define the configuration structure `Eye2BrainConfig`, and introduce some type definitions that are useful in the following, namely:

- `value_type` is the type of the values of the model,
- `mesh_type` is the type of the mesh,
- `basis_type` is the type of the basis functions. In the present, we use Lagrange basis functions of order `Order`,

see Listing 7.16. Note that this class is constructed as a template class, with two template parameters `Order` and `Dim`, which are the order of the Lagrange basis functions and the dimension of the mesh, respectively. For the results presented in this manuscript, we used `Order = 2` and `Dim = 3`, but the model can be run for any values of `Order` and `Dim` (by providing a mesh of the right dimension).

Listing 7.16: Configuration structure `Eye2BrainConfig` for the reduced basis model of the eye model.

```
template<int Order, int Dim> struct FEELPP_EXPORT Eye2BrainConfig
{
    using value_type = double;
    using mesh_type = Mesh<Simplex<Dim>>;
    using basis_type = bases< Lagrange<Order, Scalar> >;
    using space_type = FunctionSpace<mesh_type, basis_type, value_type>;
};
```

Then, the main class that builds the reduced basis model is defined as a template class `Eye2Brain`, see Listing 7.17. This class inherits from the Feel++ class `ModelCrbBase` already implemented to build reduced basis models.

Listing 7.17: Template class `Eye2Brain` to build the reduced basis model of the eye model.

```
template<int Order, int Dim>
class FEELPP_EXPORT Eye2Brain: public ModelCrbBase<ParameterSpace<>, typename Eye2BrainConfig<Order,
↪  Dim>::space_type>
{
    using super_type = ModelCrbBase<ParameterSpace<>, typename Eye2BrainConfig<Order, Dim>::space_type>;
public:
    Eye2Brain();
    void initBetaQ();
```

[3] https://github.com/feelpp/feelpp/tree/develop/mor/examples/eye2brain

```
    typename super_type::betaq_type computeBetaQ( parameter_type const& mu );
    void updateSpecificityModelPropertyTree(boost::property_tree::ptree& ptree) const;
    void initModel();
    double output(int output_index, parameter_type const& mu, element_type& u, bool need_to_solve=false);
private:
    const std::vector<std::vector<double>> m_coordinates = {{-0.013597, 0, 0}, ...};
    const std::vector<std::string> m_outputNames = {"O", "A", "B", "B1", "C", "D", "D1", "F", "G"};
};
```

In the following, we present the implementation of the method initializing the RBM model `void Eye2Brain<Order, Dim>::initModel()`. For clarity, the code has been split in a few parts.

In the affine decomposition, presented in Equations (3.16) and (3.17), the parameters β_A^q and β_F^p can be expressed as a function of the component of the parameter μ. In Feel++, the class `ModelCrbBase` is implemented to work with the parameters β_A^q and β_F^p, hence in the class `Eye2Brain`, we need to convert the parameters μ to the parameters β_A^q and β_F^p. This implementation is presented in Listing 7.18.

Listing 7.18: Initialization of the model parameters: conversion of the parameters μ to the parameters β_A^q and β_F^p.

```
double E_min = 20        , E_max = 320,
       h_amb_min = 8     , h_amb_max = 100,
       h_bl_min = 50     , h_bl_max = 110,
       h_r_min = 6       , h_r_max = 6,
       T_amb_min = 283.15, T_amb_max = 303.15,
       T_bl_min = 308.3  , T_bl_max = 312,
       k_lens_min = 0.21 , k_lens_max = 0.544;
double k_lens_ref = 0.4, h_amb_ref = 10, h_bl_ref = 65, h_r_ref = 6;

this->Dmu->setDimension( 7 );
auto mu_min = this->Dmu->element();
mu_min << k_lens_min, h_amb_min, h_bl_min, h_r_min, 1, h_amb_min*T_amb_min + h_r_min*T_amb_min - E_max,
↪  h_bl_min*T_bl_min;
this->Dmu->setMin( mu_min );
auto mu_max = this->Dmu->element();
mu_max << k_lens_max, h_amb_max, h_bl_max, h_r_max, 1, h_amb_max*T_amb_max + h_r_max*T_amb_max - E_min,
↪  h_bl_max*T_bl_max;
this->Dmu->setMax( mu_max );
```

Then, we set up the function spaces, and the energy matrix that will be used to compute norm of elements, see Listing 7.19.

Listing 7.19: Initialization of the function spaces and the energy matrix.

```
auto mesh = loadMesh( _mesh = new typename Eye2BrainConfig<Order, Dim>::mesh_type,
                      _update = MESH_UPDATE_FACES | MESH_UPDATE_EDGES | MESH_NO_UPDATE_MEASURES );
this->setFunctionSpaces(Eye2BrainConfig<Order, Dim>::space_type::New( mesh ));
auto u = this->Xh->element();   // Test function
auto v = this->Xh->element();   // Trial function
std::vector<double> muRef = {k_lens_ref, h_amb_ref, h_bl_ref, h_r_ref, 1};
auto energy = backend()->newMatrix( _test = this->Xh, _trial = this->Xh );
```

Finally, we define the affine decomposition computed in Equations (3.16) and (3.17) by implementing each term of it, see Listing 7.20. We present here the implementation for the sole bilinear form a_0, and the linear form f_0, as the process is similar for the other terms. We also present the specific case of a_4, as it is a sum over different regions. The functions `addLhs` and `addRhs` are used to add the bilinear and linear forms to the left-hand side and right-hand side of the equation, respectively.

Listing 7.20: Implementation of the affine decomposition for the bilinear form a_L and the linear form f_L.

```
auto a0 = form2( _trial = this->Xh, _test = this->Xh );
a0 = integrate( _range=markedelements(mesh, "Lens"), _expr=gradt(u)*trans(grad(v)) );
a0.matrixPtr()->close();
this->addLhs( { a0 , "mu0" } );
energy->addMatrix(muRef[0], a0.matrixPtr() );
[...]
auto a4 = form2( _trial = this->Xh, _test = this->Xh );
std::map<std::string, double> regions = { {"Cornea", 0.58}, {"Sclera", 1.0042}, ... };
for (auto const& [key, val] : regions)
    a4 += integrate(_range=markedelements(mesh,key), _expr=val*gradt(u)*trans(grad(v)));
a4.matrixPtr()->close();
this->addLhs( { a4 , "mu4" } );
energy->addMatrix( muRef[4], a4.matrixPtr() );

auto f0 = form1( _test = this->Xh );
f0 = integrate( _range = markedfaces( mesh, "BC_Cornea" ), _expr = id( v ) );
f0.vectorPtr()->close();
this->addRhs( { f0, "mu5" } );
[...]

energy->close();
this->addEnergyMatrix( energy );
```

A similar treatment is done for the outputs of the model. In the current implementation of the class `CrModelBase`, only one output can be defined, the output of interest is hence defined once and provided to the application as an option `"measure-index"`. Depending on the value of this option, we select to output of interest. The code is given in Listing 7.21. Note that in the complete code, other outputs are also implemented, namely the outputs using regularized Gaussian sensors.

Listing 7.21: Implementation of the output of the model.

```
using form1_type = vf::detail::LinearForm<typename Eye2BrainConfig<Order, Dim>::space_type, typename
↪  backend_type::vector_type, typename backend_type::vector_type>;
form1_type out1;
int measure_index = ioption(_name = "measure-index");
Feel::cout << "Measure index = " << measure_index << std::endl;
if (measure_index >= 1)    // sensor pointwise output
{
    std::string name = m_outputNames[measure_index-1];
    std::vector<double> coord = m_coordinates[measure_index-1];
    Feel::cout << "[Eye2brain] Output " << name << " at coord " << coord << std::endl;
    node_type n(Eye2BrainConfig<Order, Dim>::space_type::nDim);
    for( int i = 0; i < Eye2BrainConfig<Order,Dim>::space_type::nDim; ++i ) n(i) = coord[i];
    auto s = std::make_shared<SensorPointwise<space_type>>(this->Xh, n, name);
    out1 = form1(_test = this->Xh, _vector = s->containerPtr());
    out1.vectorPtr()->close();
}
else if ( measure_index == 0 )   // mean over cornea
{
    Feel::cout << "[Eye2brain] Output mean over cornea" << std::endl;
    out1 = form1( _test = this->Xh );
    double meas = integrate( _range = markedelements(mesh, "Cornea"), _expr = cst(1.) ).evaluate()(0,0);
    out1 = integrate( _range=markedelements(mesh, "Cornea"), _expr=id(u)/cst(meas) );
}
else
    throw std::logic_error("[Eye2Brain] error with output_index: between 0 and 9");

this->addOutput( { out1, "1" } );
```

To build the reduced basis model, after having compiled the directory in the Feel++ repository, the following command can be used:

```
$ ./feelpp_mor_eye2brain_3dP2app --config-file eye2brain/eye2brain-3d.cfg
```

Once the reduced basis is constructed, other library of Feel++ can be used to run the online model. More details are provided about it in Section 7.5.2.

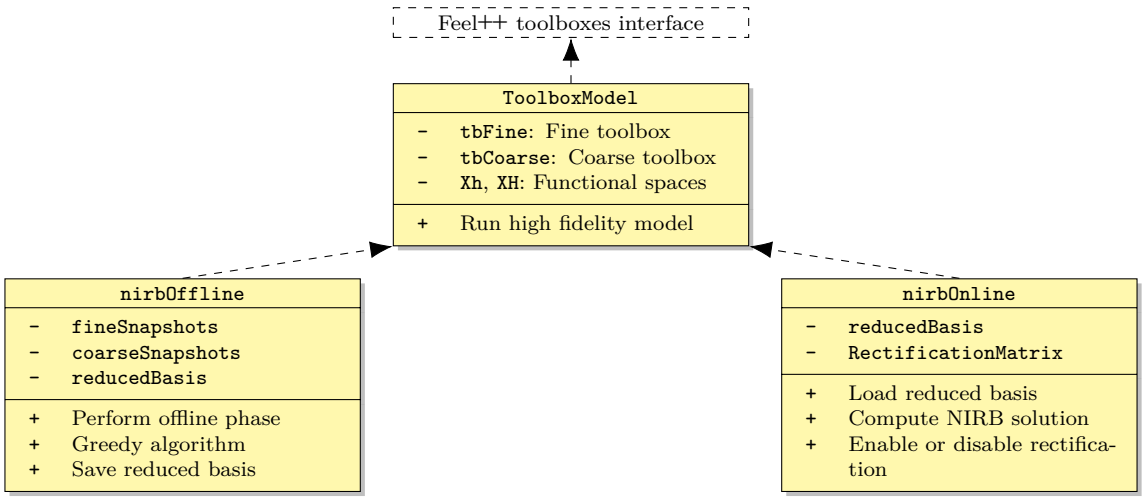


Figure 7.2: UML diagram of the NIRB module.

The offline step of the reduced basis model is quite time-consuming, for instance it took about 3.5 hours to build the reduced basis model for the eye model presented in Section 1.4. The results of the reduced basis model are presented in Section 3.1.4.

7.3 Non-Intrusive Reduced Basis method

In this section, we present the contribution to the Feel++ library for the NIRB method, introduced in Section 3.2. Note that this code was co-developed in collaboration with Ali Elarif. The whole code is available in the Feel++ repository[4].

The NIRB module consists in a Python library, included in the component `feelpp-mor` or Feel++. All the examples treated during in its development used the model of heat transfer, but the library is designed to be easily adaptable to other kind of models provided by the toolboxes of Feel++. Precisely, we define a first class `ToolboxModel` that deals with the model and run the simulation. Specifically, this class contains two attributes `tbFine` and `tbCoarse` that are object implementing the `Toolbox` interface of Feel++. The, two classes are defined: `nirbOffline` and `nirbOnline`, both inheriting from the class `ToolboxModel`. Each of this class is dedicated to the offline and online phases of the NIRB method, respectively. The UML diagram of the NIRB module is presented in Figure 7.2.

7.3.1 Implementation of the offline stage

Depending on the method chosen by the user to generate the reduced basis (random or greedy, refer to Section 3.2), the offline phase is performed differently. First, we present the method `initProblem` that initializes the problem using the random sampling method. This function, presented in Listing 7.22, takes as argument:

- `numberOfInitSnapshots` the number of initial snapshots to generate,
- `Xi_train` the training set of parameters, if `None` is given, the function will generate the parameter randomly,
- `computeCoarse` a boolean to indicate if the coarse snapshots should be computed. This step is crucial especially when the rectification is performed.

[4] https://github.com/feelpp/feelpp/tree/develop/python/pyfeelpp-mor/feelpp/mor/nirb

Note that the implementation takes into account the possibility to run the code in parallel.

Listing 7.22: Definition of the `nirbOffline` class.

```
def initProblem(self, numberOfInitSnapshots, Xi_train=None, computeCoarse=False):
    if self.doRectification: computeCoarse=True
    self.fineSnapShotList = []
    self.coarseSnapShotList = []

    if Xi_train is None:
        s = self.Dmu.sampling()
        s.sample(numberOfInitSnapshots)
        vector_mu = s.getVector()
    else:   # Take randomly the right number of parameter in Xi_train
        if self.worldcomm.isMasterRank():
            indice_mu = random.sample(range(len(Xi_train)), k=numberOfInitSnapshots)
            indice_mu = np.array(indice_mu, dtype='i')
        else :
            indice_mu = np.empty(numberOfInitSnapshots, dtype='i')
        self.worldcomm.globalComm().Bcast(indice_mu, root=0)
        vector_mu = [Xi_train[i] for i in list(indice_mu)]

    if computeCoarse:
        for mu in vector_mu:
            fineSnapshot = self.getToolboxSolution(self.tbFine, mu)
            self.fineSnapShotList.append(fineSnapshot)
            coarseSnapshot = self.getToolboxSolution(self.tbCoarse, mu)
            self.coarseSnapShotList.append(coarseSnapshot)
    else: [...] # Compute only fine snapshots

    return vector_mu
```

On the other hand, for the greedy procedure, we first define a function `getReducedSolution` that computes the online approximation used for the estimator `onlineSol`$(\mu)=\sum_{i=1}^{N} \langle \xi_i, u_H(\mu) \rangle_{L^2}$. This function is defined in Listing 7.23.

Listing 7.23: Definition of the `getReducedSolution` function.

```
def getReducedSolution(self, uH, mu, N):
    onlineSol = self.XH.element()
    onlineSol.setZero()
    for i in range(N):
        uHN_i = self.l2ScalarProductMatrixCoarse.energy(self.coarseSnapShotList[i], uH)
        onlineSol.add(uHN_i, self.coarseSnapShotList[i])
    return onlineSol
```

Then the main function `initProblemGreedy` is defined as follows, see Listing 7.24. Note that in the implementation, we compute a first time the coarse snapshots $u_H(\mu)$ for all $\mu \in \Xi_{\text{train}}$ only once.

Listing 7.24: Definition of the `initProblemGreedy` function.

```
def initProblemGreedy(self, Ntrain, eps, Xi_train=None, Nmax=50):
    if self.tbCoarse is None: raise Exception("Coarse toolbox needed for computing coarse Snapshot. set
    ↪  initCoarse->True in initialization")
    if Xi_train is None:
        s = self.Dmu.sampling()
        s.sample(Ntrain)
        Xi_train = s.getVector()
    Xi_train_copy = Xi_train.copy()

    Delta_star = eps+1
    Deltas_conv = []
    S = []
    self.fineSnapShotList = []
    self.coarseSnapShotList = []
    N = 0
    coarseSolutions = {}
```

```
    for mu in Xi_train_copy:    # Computation of coarse solutions
        coarseSolutions[mu] = self.getToolboxSolution(self.tbCoarse, mu)
    for i in range(2):  # First two snapshots included in the basis
        mu0 = self.Dmu.element()
        S.append(mu0)
        self.fineSnapShotList.append(self.getToolboxSolution(self.tbFine, mu0))
        uH = self.getToolboxSolution(self.tbCoarse, mu0)
        self.coarseSnapShotList.append(uH)
        N += 1

    while Delta_star > eps and N < Nmax:
        M = N - 1
        Delta_star = -float('inf')
        for i, mu in enumerate(Xi_train_copy,):
            uHN = self.getReducedSolution(coarseSolutions[mu], N)
            uHM = self.getReducedSolution(coarseSolutions[mu], M)
            diff = uHN - uHM
            Delta = self.l2ScalarProductMatrixCoarse.energy( diff, diff )
            if Delta > Delta_star:
                Delta_star = Delta
                mu_star = mu
                idx = i
        S.append(mu_star)
        del Xi_train_copy[idx]
        fineSnapshot = self.getToolboxSolution(self.tbFine, mu_star)
        self.fineSnapShotList.append(fineSnapshot)
        coarseSnapshot = self.getToolboxSolution(self.tbCoarse, mu_star)
        self.coarseSnapShotList.append(coarseSnapshot)
        N += 1
        Deltas_conv.append(Delta_star)
    return S, Xi_train, Deltas_conv
```

Once the step of choosing the snapshots is done, the function `generateReducedBasis` (Listing 7.25) is called to compute the reduced basis, precisely implementing Algorithm 5.

Listing 7.25: Definition of the `generateReducedBasis` function.

```
def generateReducedBasis(self, tolerance=1.e-12):
    self.reducedBasis, RIC = self.PODReducedBasis(tolerance=tolerance)
    self.N = len(self.reducedBasis)
    self.orthonormalizeL2()     # Gram-Schmidt orthonormalization
    if self.doRectification:
        self.coeffCoarse, self.coeffFine = self.coeffRectification()
    return RIC
```

The function `PODReducedBasis` is defined as in Listing 7.26. Note that a check is performed to avoid recomputing all the matrix if the function is called several times. This check is particularly useful when the greedy algorithm is used, as snapshots are added iteratively.

Listing 7.26: `PODReducedBasis` function: correlation matrix.

```
if self.correlationMatrix == None :
        self.correlationMatrix = np.zeros((Nsnap, Nsnap))
        for i, snap1 in enumerate(self.fineSnapShotList):
            for j, snap2 in enumerate(self.fineSnapShotList):
                if i > j:
                    corr = self.l2ScalarProductMatrix.energy(snap1, snap2)
                    self.correlationMatrix[i, j] = corr
                    self.correlationMatrix[j, i] = corr
            self.correlationMatrix[i, i] = self.l2ScalarProductMatrix.energy(snap1, snap1)
        self.correlationMatrix /= Nsnap
    else:
        lastSnap = self.fineSnapShotList[-1]
        lastCol = np.zeros(Nsnap)
        for i, snap in enumerate(self.fineSnapShotList[:-1]):
            lastCol[i] = self.l2ScalarProductMatrix.energy(snap, lastSnap)
        lastCol /= Nsnap
        self.correlationMatrix = np.vstack((self.correlationMatrix, lastCol[:-1]))
```

```
        self.correlationMatrix = np.column_stack((self.correlationMatrix, lastCol))
```

Then we compute in Listing 7.27 the eigenvalues and eigenvectors of the correlation matrix, to finally compute the vectors of the reduced basis.

Listing 7.27: `PODReducedBasis` function: compute the basis vectors.

```
eigenValues, eigenVectors = eigh(self.correlationMatrix)
idx = eigenValues.argsort()[::-1]
eigenValues = eigenValues[idx]
eigenVectors = eigenVectors[:, idx]
Nmode = len(eigenValues)
for i in range(Nmode):
    eigenVectors[:,i] /= math.sqrt(abs(eigenValues[i]))

reducedBasis = []
sum_eigenValues = eigenValues.sum()
RIC = []
for i in range(Nmode):
    vec = self.Xh.element()
    vec.setZero()
    for j in range(Nsnap):
        vec.add(eigenVectors[j,i], self.fineSnapShotList[j])
    reducedBasis.append(vec)
    RIC.append(eigenValues[:i].sum() / sum_eigenValues)
    if abs(1. - RIC[i])<= tolerance: break
```

Moreover, if the user wants to perform the rectification post-process introduced in Section 3.2.3, the matrices $\underline{\underline{C}}^h$ and $\underline{\underline{C}}^H$ are computed though the function `coeffRectification`, as in Listing 7.28:

Listing 7.28: Definition of the `coeffRectification` function.

```
def coeffRectification(self):
    assert len(self.reducedBasis) !=0, f"need computation of reduced basis"
    interpolateOperator = self.createInterpolator(self.tbCoarse, self.tbFine)
    InterpCoarseSnaps = []
    for snap in self.coarseSnapShotList:
        InterpCoarseSnaps.append(interpolateOperator.interpolate(snap))
    coeffCoarse = np.zeros((self.N, self.N))
    coeffFine = np.zeros((self.N, self.N))
    for i in range(self.N):
        for j in range(self.N):
            coeffCoarse[i,j] = self.l2ScalarProductMatrix.energy(InterpCoarseSnaps[i],
            ↪  self.reducedBasis[j])
            coeffFine[i,j] = self.l2ScalarProductMatrix.energy(self.fineSnapShotList[i],
            ↪  self.reducedBasis[j])
    return coeffCoarse, coeffFine
```

Finally, all the data computed in the offline phase are stored in a directory, to be loaded afterwards for the online phase. The function `saveData` is defined in Listing 7.29. A safeguard is implemented to avoid overwriting the data, unless the user specifies the `force` parameter.

Listing 7.29: Definition of the `saveData` function.

```
def saveData(self, path="./", force=False):
    reducedPath = os.path.join(path, 'reducedBasis')
    reducedFilename = 'reducedBasis'
    l2productPath = os.path.join(path, 'l2productBasis')
    l2productFilename = 'l2productBasis'

    if self.worldcomm.isMasterRank():
        if os.path.isdir(path) and not force:
            print(f"[NIRB] Directory {path} already exists. Run with force = True to force saving")
            return
        if not os.path.isdir(path): os.makedirs(path)
        if not os.path.isdir(reducedPath): os.makedirs(reducedPath)
```

```
        if not os.path.isdir(l2productPath): os.makedirs(l2productPath)
    self.worldcomm.globalComm().Barrier()

    for i in range(len(self.reducedBasis)):
        self.reducedBasis[i].save(reducedPath, reducedFilename, suffix=str(i))
    for i in range(len(self.l2ProductBasis)):
        vec = self.Xh.element(self.l2ProductBasis[i])
        vec.save(l2productPath, l2productFilename, suffix=str(i))

    coeffCoarseFile = os.path.join(path, "coeffcoarse")
    coeffFineFile = os.path.join(path, "coefffine")
    if self.doRectification and self.worldcomm.isMasterRank():
        np.save(coeffCoarseFile, self.coeffCoarse)
        np.save(coeffFineFile, self.coeffFine)

    self.outdir = os.path.abspath(path)
    if self.worldcomm.isMasterRank(): print(f"[NIRB] Data saved in {self.outdir}")
```

The global pipeline of the offline phase is implemented this way:

Listing 7.30: Global pipeline of the offline phase.

```
config_nirb = fppc.readJson(nirb_file)['nirb']
nirb_off = nirbOffline(initCoarse=True, **config_nirb)
nirb_off.initModel()
Xi_train = generatedAndSaveSampling(nirb_off.Dmu, 200, path=Xi_train_path, samplingMode="log-random")
nirb_off.generateOperators(coarse=True)

if doGreedy: _, Xi_train, _ = nirb_off.initProblemGreedy(500, 1e-5, Nmax=config_nirb['nbSnapshots'],
↪  Xi_train=Xi_train, computeCoarse=True, samplingMode="random")
else: Xi_train = nirb_off.initProblem(nbSnap, Xi_train=Xi_train)
RIC = nirb_off.generateReducedBasis(tolerance=1.e-12)
nirb_off.saveData(RESPATH, force=True)
```

The user can then run the offline phase with the following command:

```
$ mpirun -np 12 python3 nirbOffline.py --config-file data/eye/heat/eye.cfg --N 30
```

Other parameters can be provided to the script and are set a JSON file, as in Listing 7.31.

Listing 7.31: Example of JSON configuration file for the offline phase.

```
"nirb":
{
    "H": 0.25,
    "h": "H**2:H",
    "dim": 3,
    "order": 1,
    "toolboxType": "heat",
    "finemesh_path": "$cfgdir/fin.geo",
    "coarsemesh_path": "$cfgdir/fin.geo",
    "model_path": "$cfgdir/thermal-fin.json",
    "doRectification": true,
    "nbSnapshots": 10,
    "greedy-generation": false,
}
```

7.3.2 Implementation of the online stage

The online phase presented in Section 3.2.4 is implemented in the `nirbOnline` class, also inheriting from the `ToolboxModel` class. The main function to compute the NIRB approximation $u_{Hh}^N(\mu)$ described in Equation (3.47) is the `getOnlineSol` function, defined in Listing 7.32. Some optional parameters can be provided to the function, such as the number of basis vectors to use, or if the rectification post-process should be performed (*c.f.* Section 3.2.3). If no such parameters are provided, the application will use the default values defined in the class. Moreover, when this

function is called for many times, some elements such as the rectification matrix $\underline{\underline{\mathcal{R}}}$ are stored in the class to avoid recomputing them.

Listing 7.32: Definition of the `getOnlineSol` function.

```
def getOnlineSol(self, mu, doRectification=None, Nb=None):
    if Nb is None: Nb = self.N
    onlineSol = self.Xh.element()    # Set a zero vector from the fine functional space
    onlineSol.setZero()
    compressedSol = self.getCompressedSol(mu=mu, Nb=Nb) # coefficients |α^{N,H}(μ)|
    if doRectification is None: doRectification = self.doRectification
    if doRectification:
        if Nb not in self.RectificationMat:
            self.RectificationMat[Nb] = self.getRectification(self.coeffCoarse, self.coeffFine, Nb=Nb)
        coef = self.RectificationMat[Nb] @ compressedSol
        for i in range(Nb): onlineSol.add(float(coef[i]), self.reducedBasis[i])
    else:
        for i in range(Nb): onlineSol.add(float(compressedSol[i]), self.reducedBasis[i])
    return onlineSol
```

The method `getCompressedSol` presented in Listing 7.33 computes the coefficients $\alpha^{N,H}(\mu)$, as per Equation (3.48). The user can provide either the parameter μ or the solution interpolated solution $u^{\mathcal{N}}_{H\to h}(\mu)$.

Listing 7.33: Definition of the `getCompressedSol` function.

```
def getCompressedSol(self, mu=None, solution=None, Nb=None):
    assert (mu != None) or (solution != None), f"One of the arguments must be given: solution or mu"
    if Nb is None: Nb = self.N
    coarseSol = self.getToolboxSolution(self.tbCoarse, mu)
    interpolatedSol = self.interpolationOperator.interpolate(coarseSol)
    compressedSol = np.zeros(Nb)
    for i in range(Nb):
        compressedSol[i] = self.l2ProductBasis[i].to_petsc().dot(interpolatedSol.to_petsc())
    return compressedSol
```

Note that there is still room to improving this code, especially by getting rid of the `for` loop. This could be achieved with the method `maxpy` and `mdot` of the PETSc library. A crucial point in this code is the interpolation of the coarse solution to the fine space, that is performed by the `interpolationOperator` object, initialized in Listing 7.34:

Listing 7.34: Initialization of the interpolation operator.

```
Vh_image = image_tb.spaceTemperature()
Vh_domain = domain_tb.spaceTemperature()
self.interpolationOperator=fpp.interpolator(domain = Vh_domain, image = Vh_image, range =
↪  image_tb.rangeMeshElements())
```

The object `self.interpolationOperator`, implemented in the Python library of Feel++, allows to interpolate the solution both in sequential and parallel, transparently for the user. Finally, the pipeline of the online phase is implemented as in Listing 7.35.

Listing 7.35: Global pipeline of the online phase.

```
config_nirb = fppc.readJson(nirb_file)['nirb']
nirb_on = nirbOnline(**config_nirb)
nirb_on.initModel()

mu = nirb_on.Dmu.element()
mu.setParameters([0.01, 0.1, 0.1, 0.1, 0.1])
err = nirb_on.loadData(path=RESPATH, nbSnap=nbSnap)
assert err == 0, "Error while loading data"

uHh_r = nirb_on.getOnlineSol(mu, doRectification=True)
uHh = nirb_on.getOnlineSol(mu, doRectification=False)
```

If wanted, we can also compute the high-fidelity solution, and export all the results for visualization with Paraview:

Listing 7.36: Export the results for visualization.

```
uh = nirb_on.getToolboxSolution(nirb_on.tbFine, mu)
error_r = nirb_on.normMat(uHh_r - uh)
error = nirb_on.normMat(uHh - uh)
if nirb_on.worldcomm.isMasterRank():
    print(f"[NIRB] L2 norm between nirb online rectified and toolbox sol = {error_r}")
    print(f"[NIRB] L2 norm between nirb online and toolbox sol = {error}")

if exporter:
    dirname = "nirbSol"
    nirb_on.initExporter(dirname, toolbox="fine")
    nirb_on.exportField(uh, "uh")
    nirb_on.exportField(uHh_r, "uNirb_r")
    nirb_on.exportField(uHh, "uNirb")
    nirb_on.saveExporter()
```

The user can then run the online phase with the following command:

```
$ mpirun -np 12 python3 nirbOnline.py --config-file data/eye/heat/eye.cfg --N 30
↪  --exporter 1
```

Results of the methods, as well as the convergence of the method, are presented in Sections 3.2.6 and 3.2.7.

7.4 Elliptic problems with a Dirac sources term

In the spirit of the application `feelpp_qs_laplacian`[5], we developed a new application, `feelpp_laplacian_dirac`, that solves the Laplacian problem with a Dirac in the right hand-side, as described in Chapter 4. At the time of redaction of this manuscript, the preprint and the code is not yet available, but it will be as soon as the preprint is published, on the GitHub repository[6]. The implementation stems directly from the computation of the variational form of the problem.

The main part of the code, computing the numerical solution is shown in Listing 7.37.

Listing 7.37: Elliptic problem with a Dirac source.

```
/**
 * @param Vh The space of trial and test functions.
 * @param u The solution element.
 * @param v The test element.
 * @param s The sensor.
 * @param mesh The mesh.
 * @param boundary_condition The type of boundary condition.
 * @param G_expr The expression for the boundary condition.
 * @param f The expression for the right-hand side.
 * @param un The expression for the Neumann boundary condition.
 * @param mu The expression for the Robin boundary condition.
 * @param r_2 The expression for the Robin boundary condition.
 */
void computeSolution(_space_ptr_type Vh, _element_type &u, _element_type &v, _container_ptr_type s,
↪  _mesh_ptr_type mesh, int boundary_condition, _expr_scal_type &G_expr, _expr_scal_type &f,
↪  _expr_scal_type &un, _expr_scal_type &mu, _expr_scal_type &r_2)
{
    auto a = form2( _trial = Vh, _test = Vh);
    auto l = form1( _test = Vh, _vector = s );
    a = integrate(_range = elements(mesh), _expr = gradt(u) * trans(grad(v)) );
    switch (boundary_condition)
    {
```

[5] See source code: https://github.com/feelpp/feelpp/blob/develop/feelpp/quickstart/laplacian/qs_laplacian.cpp

[6] https://github.com/feelpp/article.dirac

```
    case 0:         // Dirichlet boundary condition
        a += on(_range = boundaryfaces(mesh), _rhs = l, _element = u, _expr = G_expr);
        break;
    case 1:         // Neumann boundary condition
        l += integrate(_range = boundaryfaces(mesh), _expr = un * id(v));
        break;
    case 2:         // Robin boundary condition
        a += integrate(_range = boundaryfaces(mesh), _expr = mu * idt(u) * id(v));
        l += integrate(_range = boundaryfaces(mesh), _expr = r_2 * id(v));
        break;
    case 3:         // Mixed boundary condition
        [...]       // Handle mixed boundary conditions
        break;
    default:
        if (Environment::isMasterRank()) std::cout << "No valid BC" << std::endl;
    }
    a.solve(_rhs = l, _solution = u);
}
```

Note that because of geometrical discretization of the domain Ω, we impose on the boundary $\partial\Omega$ a homogeneous Dirichlet boundary condition (*e.g.* `Gexpr` for Dirichlet boundary conditions) that is computed from the expected solution using symbolic computation. This happens in the context where we plan to compute the numerical error of the solution. If not, then the selected boundary condition is applied to the boundary of the domain.

The Dirac source term `s` is implemented using the `SensorPointwise` class, which is a pointwise sensor that is able to compute the value of the solution at a given point, as shown in Listing 7.38.

Listing 7.38: Dirac source term.

```
using _sensor_type = SensorPointwise<_space_type>;
using _container_ptr_type = std::shared_ptr<Vector<double, unsigned int>>;
_container_ptr_type s = std::make_shared<_sensor_type>(Vh, n, "0")->containerPtr();
```

The application can be run using the following command line. The argument `bc` sets up the type of boundary condition used, Dirichlet in this example.

```
$ mpirun -np 16 feelpp_phd.thomas_laplacian_dirac_2DP6 \
    --bc 0 --coord.y 0.5 --do-remesh 0 \
    --config-file laplacian_dirac.cfg --gmsh.hsize 0.00625\
    --export-solution "fine-dirac"
```

Results of convergence of this program are presented in Sections 4.3 and 4.4.1 to 4.4.3.

7.5 Sensitivity Analysis

We provide in this section the implementation of the sensitivity analysis carried out in Chapter 5, on the one hand with the deterministic sensitivity analysis (DSA) and on the other hand with the stochastic sensitivity analysis (SSA). The whole implementation is available in the dataset published among the preprint: *(i)* the DSA, implemented in Python, is available in [SPS24a], *(ii)* the SSA, implemented thought the Feel++ and OpenTURNS framework is available in the main repository [Pru+24b][7].

7.5.1 Deterministic Sensitivity Analysis

We first present the code developed to perform the DSA described in Section 5.2. This code is written in Python and uses the Feel++ library to solve the high-fidelity model, for each selected

[7] https://github.com/feelpp/feelpp/blob/develop/mor/examples/eye2brain/sensitivity_analysis.cpp

combination of input parameters. First, we set the environment up, load the model, and define the parameters to be varied, as shown in Listing 7.39.

Listing 7.39: Setup the DSA.

```
heatBox = heat.heat(dim=3, order=2)
heatBox.init()

crb_model_properties = mor.CRBModelProperties(worldComm=fppc.Environment.worldCommPtr())
crb_model_properties.setup(PWD + '/../crb_param.json')
crb_model_parameters = crb_model_properties.parameters()
D = mor._mor.ParameterSpace.New(crb_model_parameters, fppc.Environment.worldCommPtr())

baseline = {"k_lens":0.40, "E":40, "h_bl":65, "h_amb":10, "T_amb":293.15, "T_bl":310.15}
mu = D.element()
mu.setParameters(baseline)

params = D.parameterNames()
values = {
    "E": [20, 40, 70, 100, 320],
    "T_amb": [273.15, 278, 293.15, 298.15, 303.15, 308],
    "T_bl": [308.15, 310.15, 310.85, 311.15, 311.65, 312.15],
    "h_amb": [8, 10, 12, 15, 100],
    "h_bl": [65, 90, 110],
    "k_lens": [0.21, 0.30, 0.40, 0.544]
}
df_points = pd.read_csv(os.path.join(PWD, "points_coord.csv"))  # Cooridnates of the output of interest
```

We implemented a function `updateParameters` that updates the parameters of the toolbox object. With this function, we run the deterministic sensitivity analysis using the high-fidelity model, presented in Listing 7.40.

Listing 7.40: Run the DSA.

```
for param in params:
    mu.setParameters(baseline)      # Reset the parameters to the baseline
    df_res = pd.DataFrame(columns=points_name, dtype=float)
    for value in values[param]:
        mu.setParameterNamed(param, value)
        updateParameters(heatBox, mu)
        heatBox.solve()
        heatBox.exportResults()
        measures = heatBox.postProcessMeasures().values()   # Get all the outputs computed
        df_res.loc[value] = res_of_meas(meas, points_name)  # Post-process the outputs
    df_res.to_csv(os.path.join(PWD, "results", f"{param}_feel.csv"), index_label=param)
```

The DSA is performed by running the whole script:

```
$ mpirun -np 12 python3 run-SA.py
```

The results of this analysis are presented in Section 5.2.

7.5.2 Stochastic Sensitivity Analysis with OpenTURNS

In this section, we present the code developed to perform the Stochastic Sensitivity Analysis described in Section 5.3. This code is written in C++ and uses the OpenTURNS library [Bau+16]. Precisely, it uses the methods implemented provided by the class `FunctionalChaosAlgorithm` [Ope], using a bootstrap method for the confidence intervals.

We present here only some selected parts of the code, the full code can be found in the Feel++ repository[8]. Before introducing the function, we define some macro types and constants in Listing 7.41, useful in the following.

[8] https://github.com/feelpp/feelpp/blob/develop/mor/examples/eye2brain/sensitivity_analysis.cpp

Listing 7.41: Definition of the macro types and constants.

```
using element_t = Feel::ParameterSpaceX::element_type;
using plugin_ptr_t = std::shared_ptr<Feel::CRBPluginAPI>;
using parameter_space_ptr_t = std::shared_ptr<Feel::ParameterSpaceX>;

const std::vector<std::string> NAME = {"h_bl", "h_amb", "T_bl", "T_amb", "E", "k_lens"};
const std::vector<double> MINS = {50, 8, 308.3, 283.15, 20, 0.21};
const std::vector<double> MAXS = {110, 100, 312, 303.15, 320, 0.544};
const size_t SIZE = 6;
```

In Listing 7.42, we show the construction of the random distribution for the input parameters, stemming from the model introduced in Section 5.3.1. Then Listing 7.43 presents how to compute the output sample from a given input sample.

Listing 7.42: Construction of the random distribution for the input parameters.

```
OT::ComposedDistribution composedFromModel()
{
    OT::Collection<OT::Distribution> marginals( SIZE );
    for (size_t d = 0; d < SIZE; ++d)
    {
        OT::Distribution dist;
        std::string name = NAME[d];
        if (name == "h_bl")
        {
            double s = 0.15; double mu = log(65) - 0.5*s*s;
            dist = OT::TruncatedDistribution(OT::LogNormal(mu, s, 0), OT::Interval(50, 110));
        }
        else if (name == "h_amb")
        {
            double s = 1; double mu = log(10) - 0.5*s*s;
            dist = OT::TruncatedDistribution(OT::LogNormal(mu, s, 8), OT::Interval(8, 100));
        }
        else if (name == "E")
        {
            double Emin = 20, Emax = 320;
            dist = OT::Uniform( MINS[d], MAXS[d] );
        }
        else
        {
            dist = OT::Uniform( MINS[d], MAXS[d] );
        }
        dist.setDescription( {name} );
        marginals[d] = dist;
    }
    return OT::ComposedDistribution( marginals );
}
```

Listing 7.43: Computation of the output sample.

```
/**
 * @brief Generate the output sample from a given input sample
 *
 * @param input Sample of input parameters
 * @param plugin loaded plugin
 * @param time_crb collection of timers
 * @param online_tol online tolerance
 * @param rbDim size of the reduced basis
 * @return OT::Sample
 */
OT::Sample output(OT::Sample const& input, plugin_ptr_t const& plugin, Eigen::VectorXd &time_crb, double
↪  online_tol, int rbDim)
{
    size_t n = input.getSize();
    double s_output, s_errorBound;
    OT::Sample output(n, 1);
    parameter_space_ptr_t Dmu = plugin->parameterSpace();
    element_t muMin = Dmu->min(), muMax = Dmu->max();
```

```
    double k_lens, h_amb, h_bl, h_r=6, T_amb, T_bl, E;
    for (size_t i: tqdm::range(n))
    {
        element_t mu = Dmu->element();
        OT::Point X = input[i];          // X = [ h_bl, h_amb, T_bl, T_amb, E, k_lens ]
        k_lens = X[5]; h_amb = X[1]; h_bl = X[0]; T_amb = X[3]; T_bl = X[2]; E = X[4];
        mu.setParameter(0, k_lens);                          // mu_0 = k_lens
        mu.setParameter(1, h_amb);                           // mu_1 = h_amb
        mu.setParameter(2, h_bl);                            // mu_2 = h_bl
        mu.setParameter(3, h_r);                             // mu_3 = h_r
//      mu.setParameter(4, 1);                               // mu_4 = 1
        mu.setParameter(5, h_amb*T_amb + h_r*T_amb - E); // mu_5 = h_amb T_amb + h_r T_amb - E
        mu.setParameter(6, h_bl*T_bl);                       // mu_6 = h_bl T_bl
        Feel::CRBResults crbRes = plugin->run(mu, time_crb, online_tol, rbDim, false);
        s_output = crbRes.output();
        s_errorBound = crbRes.errorbound();
        OT::Point P(1); P[0] = s_output;
        output[i] = P;
    }
    return output;
}
```

Before presenting the main function responsible for running the sensitivity analysis, we explain how the reduced output and error bounds are computed. As shown in Listing 7.43, we use a specific plugin of type `CRBPluginAPI`, to execute the online stage for a given parameter μ. This plugin, a shared library, is created during the offline stage and is loaded when the program starts. In scenarios where multiple outputs of interest are considered, as is the case in this work, a distinct plugin is generated for each output.

Listing 7.44 presents the main function running the sensitivity analysis:

Listing 7.44: Main function running the sensitivity analysis.

```
/**
 * @param plugin the plugin from load_plugin
 * @param sampling_size size of the input sample used for computation of sobol indices
 * @param rbDim size of the reduced basis
 * @param computeSecondOrder boolean to compute second order sobol indices
 * @param online_tol tolerance for online computation of reduced basis, default to 1e-2
 * @param print_rb_matrix boolean to print the reduced basis matrix, default to false
 */
void runSensitivityAnalysis( plugin_ptr_t plugin, size_t sampling_size, int rbDim, bool
↪  computeSecondOrder=true, double online_tol=1e-2, bool print_rb_matrix=false )
{
    Eigen::VectorXd/*typename crb_type::vectorN_type*/ time_crb;
    parameter_space_ptr_t muspace = plugin->parameterSpace();

    OT::ComposedDistribution composed_distribution = composedFromModel();
    OT::Sample input_sample = composed_distribution.getSample(sampling_size);
    OT::Sample output_sample = output(input_sample, plugin, time_crb, online_tol, rbDim);
    computeSobolIndicesBootstrap(plugin, composed_distribution, sampling_size, input_sample,
    ↪  output_sample, NAMES, time_crb, online_tol, rbDim);
}
```

We introduce in Listing 7.45 the function `computeSobolIndicesBootstrap`, that actually computes the Sobol indices from an input and an output samples.

Listing 7.45: Compute the Sobol indices with bootstrap.

```
/**
 * @brief Compute the Sobol indices using bootstrap
 *
 * @param plugin plugin loaded
 * @param composed_distribution compused distribution of the input parameters
 * @param sampling_size size of the sampling
 * @param input_sample input sample of parameters
 * @param output_sample output sample computed from the input sample
```

```
 * @param tableRowHeader names of the input parameters
 * @param time_crb time collection
 * @param online_tol online tolerance
 * @param rbDim dimension of the reduced basis
 */
void computeSobolIndicesBootstrap(plugin_ptr_t plugin, OT::ComposedDistribution composed_distribution,
↪  size_t sampling_size,
    OT::Sample input_sample, OT::Sample output_sample, std::vector<std::string> tableRowHeader,
    ↪  Eigen::VectorXd &time_crb, double online_tol, int rbDim)
{
    using namespace Feel;
    size_t dim = composed_distribution.getDimension();
    size_t bootstrap_size = ioption(_name="algo.bootstrap-size");

    OT::Collection<OT::Distribution> marginals(dim);
    for ( size_t d = 0; d < dim; ++d )
        marginals[d] = composed_distribution.getMarginal(d);
    auto basis = OT::OrthogonalProductPolynomialFactory( marginals );
    OT::UnsignedInteger total_degree = 3;

    Results res( dim, tableRowHeader, "polynomial-chaos-bootstrap", sampling_size );

    size_t N = input_sample.getSize();
    auto [fo_sample, to_sample] = computeBootstrapChaosSobolIndices(X, Y, basis, total_degree,
    ↪  distribution, bootstrap_size);
    auto [fo_interval, to_interval] = computeSobolIndicesConfidenceInterval(fo_sample, to_sample, alpha);
    res.setIndices( fo_sample.computeMean(), 1 );
    res.setIndices( to_sample.computeMean(), 2 );
    res.setInterval( fo_interval, 1 );
    res.setInterval( to_interval, 2 );
    res.exportValues( soption( _name="save.path" ) + "-bootstrap.json" );
}
```

The class `Results` is a simple self-implemented class that deals with the Sobol' indices and interval of confidence computed in the process, and exports them to a JSON file. Regarding the functions `computeBootstrapChaosSobolIndices` and `computeSobolIndicesConfidenceInterval`, they are not shown here, as they are a translation in C++ of the Python code provided by the developers of OpenTURNS [Ope].

The sensitivity analysis is performed by running the following command:

```
$ ./feelpp_mor_sensitivity_analysis --crbmodel.name <model-name> --plugin.dir
↪  ${build_directory}/install/bin/
```

The argument `<model-name>` is the name of the model used for the computation of the reduced basis, and corresponds to the choice of the output of interest (*e.g.* `eye2brain_3d_O_dirac`). The results of this SSA are presented in Section 5.3.3.

7.6 Model of heat transfer coupled with aqueous humor flow

We present in this section some configuration files used to run the simulations of the model of heat transfer coupled with aqueous humor flow presented in Chapter 6. All the files are available online [SPS24a]

We present in Listings 7.46 and 7.47 the configuration file used to set up the fieldsplit preconditioning.

Listing 7.46: Set-up fieldsplit preconditioning.

```
[heat-fluid]
ksp-type=gmres
pc-type=fieldsplit
fieldsplit-fields=0->(0,1),1->(2)       # We split the system into two blocks: (u,p) and T
fieldsplit-type=additive
```

Then we present how each block is preconditioned, the block temperature on the one hand, and the block velocity-pressure on the other hand.

Listing 7.47: Set-up preconditioning for each block.

```
# block temperature
[heat-fluid.fieldsplit-1]
ksp-type=gmres
pc-type=gamg
ksp-maxit=100

# block velocity - pressure
[heat-fluid.fieldsplit-0]
ksp-type=fgmres
fgmres-restart=100
ksp-maxit=100
ksp-rtol=1e-8
pc-type=fieldsplit
#ksp-monitor=1
fieldsplit-fields=0->(0),1->(1)      # We split the system into two blocks: (u,p)
fieldsplit-type=schur
fieldsplit-schur-fact-type=upper#full
fieldsplit-schur-precondition=user#selfp

# block velocity
[heat-fluid.fieldsplit-0.fieldsplit-0]
ksp-type=gmres#preonly#gmres
ksp-maxit=1
ksp-rtol=1e-8
pc-type=fieldsplit
fieldsplit-use-components=1
fieldsplit-fields=0->(0),1->(1),2->(2)  # We split each component of the velocity
fieldsplit-type=additive

# block velocity 0
[heat-fluid.fieldsplit-0.fieldsplit-0.fieldsplit-0] # Idem for blocks 1 and 2
ksp-type=preonly
pc-type=jacobi

# block pressure
[heat-fluid.fieldsplit-0.fieldsplit-1]
ksp-type=fgmres
ksp-maxit=1
ksp-rtol=1e-3
pc-type=gamg
```

The Slurm script that can be run to launch the simulation on a cluster is presented in Listing 7.48. Note that the subject position can be set with the variable `POSITION`, and the mesh with the variable `MESH_INDEX` (see Section 2.1.2 for more details).

Listing 7.48: Run the job.

```
#!/bin/bash
#SBATCH -J eye_heatfluid                    # name of the job
#SBATCH -N 3                                # number of nodes
#SBATCH --ntasks-per-node=128               # number of MPI tasks per node
#SBATCH --threads-per-core=1                # no hyperthreading
#SBATCH -o log/%j-eye_heatfluid-o.log       # standard output
#SBATCH -e log/%j-eye_heatfluid-e.log       # standard error

export OMP_NUM_THREADS=1
module load hpcx

MESH_INDEX=M4               # M1 M2 M3 M4 M5
SOLVER_TYPE=simple          # simple lsc
POSITION=prone              # prone supine standing

mpiexec -bind-to core feelpp_toolbox_heatfluid \
        --config-files eye-${POSITION}.cfg pc_${SOLVER_TYPE}.cfg \
```

```
18          --heat-fluid.json.patch='{ "op": "replace", "path": "/Meshes/heatfluid/Import/filename", "value":
            ↪  "$cfgdir/mesh/Mr/'${MESH_INDEX}'/Eye_Mesh3D_p$np.json" }'
```

The results of the simulation are presented in Sections 6.3 and 6.4.

Chapter 8

Conclusion and perspectives

In this thesis, we have developed a comprehensive mathematical and computational framework for simulating complex biomedical processes in the context of the human eye.

Ocular conditions present numerous biological challenges, making it crucial for clinicians to gain a deeper understanding of the underlying mechanisms. Driven by this motivation, we formulated a mathematical and computational model of the human eye, which incorporates its intricate geometry and various tissues, each with distinct physiological properties. In Section 1.4, we established a model for heat transfer within the eye, which we subsequently extended in Chapter 6 to account for the flow of aqueous humor in both the anterior and posterior chambers. These models have undergone rigorous validation, including comparisons with experimental data and previously published numerical studies.

The numerical solution of these models was achieved through the Finite Element Method, as described in Chapter 2. To mitigate the computational expense of these simulations without sacrificing accuracy, we developed a model order reduction framework, which is detailed in Chapter 3. Two approaches were employed for this purpose: the Certified Reduced Basis Method (Section 3.1) and the Non-Intrusive Reduced Basis Method (Section 3.2). The methods employed, grounded in strong mathematical theory, demonstrated efficiency and accuracy, as discussed in Sections 2.2 and 3.1. However, due to the complexity of the problem and the large-scale simulations required for realistic eye models, significant computational effort was necessary. High-performance computing resources were crucial to handle the complexity, enabling efficient exploration of parameter spaces and handling large-scale multiphysics simulations, as discussed in the scalability of the coupled model in Chapter 6.

Even though the RBM is not directly applicable to problems involving Dirac source terms due to the singularity at the evaluation point, our approach demonstrates that it can still yield accurate and efficient approximations, with numerical results showing favorable convergence properties consistent with theoretical predictions in the continuous case. Still, we investigated in Chapter 4 both theoretical and numerical aspects of the problem, providing insights into the behavior of the FEM in the presence of singularities, under various boundary conditions, and for various orders of discretization.

Within the RBM framework, we conducted a global sensitivity analysis on the eye model, which is described in Chapter 5. This allowed us to quantify the impact of key parameters on quantities of interest, such as the temperature distribution and aqueous humor flow in the anterior chamber. Our sensitivity analysis pinpointed four main physiological parameters as most influential in affecting the results: blood temperature, ambient temperature, the ambient air convection coefficient, and the evaporation rate. These findings build upon and enrich prior studies, such as those highlighted in [Sco88; NO06; Li+10], underscoring the vital role of blood flow characteristics and environmental conditions, particularly in the inner ocular tissues. Additionally, through Sobol' indices analysis, we identified the significant impact of parameter

interactions, particularly those related to ambient temperature. From a clinical standpoint, our insights into heat transport in the human eye could inform studies on the effects of electromagnetic wave radiation, as explored in [Hir+07; NO07; ON09] and related references.

Perspectives

Several aspects of the developed framework could be enhanced to improve its capabilities:

- The geometrical model developed in Section 1.3 is currently based on a CAD model of a general subject. A more precise analysis could be achieved by considering variations in geometrical parameters due to factors such as aging [BBS20] or ethnicity [SLG06; Bou11].
- Further computational improvements to the NIRB method (Section 3.2) could use the zoom functionalities [GM23a] that would considerably accelerate the online prediction and, code-wise, optimize computation time by removing the use of loops in the Python code.
- About the resolution of elliptic problem in presence of a Dirac source term (Chapter 4), our findings call for refinement of the currently available theoretical results and conduct more extensive numerical explorations of the phenomena.
- The coupled model introduced in Chapter 6, could be further developed by incorporating inflow (production) and outflow (drainage) boundary conditions and assessing their impact on the outcome of the model. Moreover, further developing reduced order modeling techniques would be beneficial in the spirit of the strategy developed in Chapter 3.

The work presented in this thesis opens several avenues of future research, including clinical aspects such as:

- Applying the developed framework in conjunction with a Kalman filter to evaluate optical properties during thermal laser-tissue interactions [AF22],
- Investigating cell displacement in the aqueous humor flow to improve our understanding of corneal cell sedimentation [Kin+18].

These advancements, combined with the ongoing efforts to refine both the geometrical and computational models, as well as the exploration of more efficient numerical techniques, contribute to the ultimate goal of creating a highly accurate digital twin of the eye. Such a digital twin would not only incorporate patient-specific anatomical variations but also simulate physiological behaviors with greater precision, paving the way for enhanced predictive modeling and personalized medical applications.

Appendix A

Effect of Cooling of the Ocular Surface on Endothelial Cell Sedimentation in Cell Injection Therapy: Insights from Computational Fluid Dynamics

The Association for Research in Vision and Ophthalmology (ARVO) Annual Meeting is a leading conference in vision research. Its prominence is shared with other major conferences like the European Society of Cataract and Refractive Surgeons Congress, the American Academy of Ophthalmology Annual Meeting, and the World Ophthalmology Congress. ARVO emphasizes basic and clinical research, making it particularly valuable for scientists and clinicians involved in investigative ophthalmology. ARVO encourages collaboration across various scientific disciplines, promoting a holistic understanding of vision science. Finally, ARVO draws a diverse international audience, facilitating the exchange of innovative ideas and research findings.

The work presented here is an extension of Chapter 6 and has been submitted to the ARVO 2025 Meeting. This work has been conducted by Thomas Saigre, Vincent Chabannes (Cemosis, Université de Strasbourg), Giovanna Guidoboni (University of Maine) Christophe Prud'homme (Université de Strasbourg) Marcela Szopos (Université Paris Cité), and Sangly P. Srinivas (School of Optometry, Indiana University).

Purpose

Cell injection therapy is a promising treatment for Fuchs endothelial corneal dystrophy by delivering cultured human corneal endothelial cells into the anterior chamber (AC). After injection, patients are rotated from a supine to a prone position for over three hours to promote sedimentation. This study aims to develop a computational framework to understand how convection currents influence the wall shear stress (WSS), thereby informing strategies to optimize sedimentation and improve therapeutic outcomes.

Method

A computational fluid dynamics framework investigated aqueous humor (AH) flow, focusing on velocity and WSS distributions. The model governed by the Navier-Stokes equations with

Zone	Orientation	Mean of magnitude of WSS		
		$T_{\text{amb}} = 25°\text{C}$	$T_{\text{amb}} = 15°\text{C}$	$T_{\text{amb}} = 33°\text{C}$
Whole domain	Prone	$6.4 \cdot 10^{-7}$ Pa	+72.96 %	−44.82 %
	Supine	$6.77 \cdot 10^{-7}$ Pa	+81.20 %	−46.23 %
Cornea	Prone	$2.3 \cdot 10^{-7}$ Pa	+71.63 %	−44.55 %
	Supine	$2.5 \cdot 10^{-7}$ Pa	**+83.40** %	**−46.89** %
Iris	Prone	$3.17 \cdot 10^{-7}$ Pa	+73.67 %	−44.96 %
	Supine	$3.31 \cdot 10^{-7}$ Pa	+79.76 %	−46.01 %

Table A.1: Evolution of the magnitude of the WSS for different zones of the eye on different regions, for various ambient temperature.

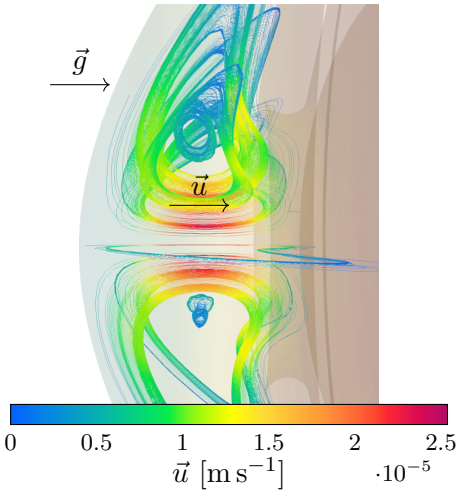
the Boussinesq approximation and gravitational effect account for heat exchange between the eye, surrounding tissues, and ambient environment. Posture (supine and prone) and ambient temperature (15 °C, 25 °C, and 33 °C) were varied to assess their impact. Elevated WSS may impede cell adhesion, while efficient mixing can enhance sedimentation and redistribution of injected cells. AH secretion and outflow were neglected, as previous work [ON08] indicated buoyancy-driven flows dominate regardless of posture. Simulations were performed using the Feel++ framework [Pru+24b], employing no-slip boundary conditions and solving for AH velocity and WSS distributions.

Results

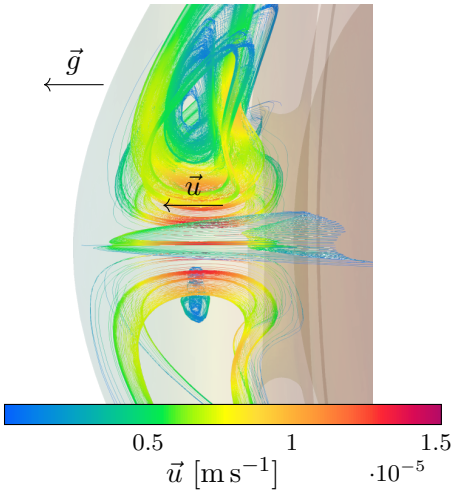
The simulations revealed that orientation and ambient temperature influence AH flows and WSS (Figure A.1). Changes from supine to prone altered gravitational alignment, reducing AH velocities and affecting sedimentation. At cooler ocular surface temperatures (15 °C), enhanced AH circulation increased WSS by up to 83.4 %, promoting more uniform cell dispersion. Warmer conditions (33 °C) reduced WSS by 46 %, resulting in less uniform sedimentation patterns potentially hindering optimal cell adhesion (Table A.1).

Conclusion

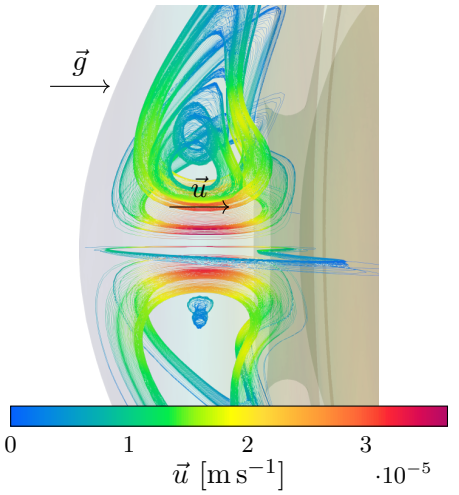
Our findings highlight the crucial interplay of thermal and postural factors in shaping AH dynamics and mechanical stresses within the AC. Cooler ocular surface conditions foster higher WSS and uniform cell dispersion, suggesting thermal modulation can improve endothelial cell therapy outcomes. By adjusting body orientation and ocular surface temperature, clinicians may guide cell sedimentation and adhesion, enhancing postoperative results in anterior segment procedures.



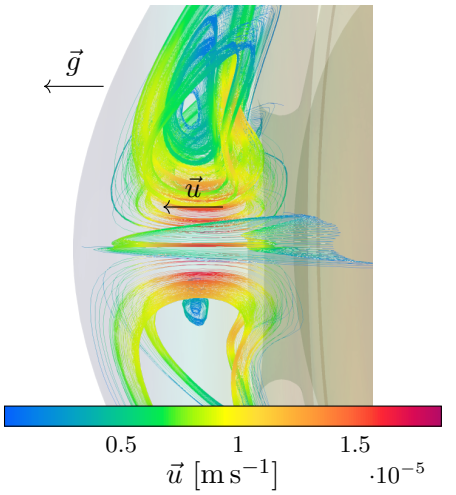
(a) Supine position with $T_{\text{amb}} = 25\,^{\circ}\text{C}$.



(b) Prone position with $T_{\text{amb}} = 25\,^{\circ}\text{C}$.



(c) Supine position with $T_{\text{amb}} = 15\,^{\circ}\text{C}$.



(d) Prone position with $T_{\text{amb}} = 15\,^{\circ}\text{C}$.

Figure A.1: AH flow in AC for various orientations and ambient temperatures.

Appendix B

Surrogate modeling of interactions in microbial communities through Physics-Informed Neural Networks

The content of this chapter is not directly linked to the work of the thesis. It is a work that was conducted during the CEMRACS[1] during summer 2023, whose thematic was *Scientific Machine Learning*. It was done in collaboration with Javan Hossie (Université d'Orléans), Béatrice Laroche (INRAE), Thibault Malou (INRAE), Lucas Perrin (Universität Konstanz) and Lorenzo Sala (INRAE). The content of this chapter has been submitted to the proceedings of the CEMRACS [Hos+24].

Microorganisms play an essential role as abundant and diverse entities within ecosystems, exerting a significant influence on biological functioning. They often come together in complex communities called microbiota, establishing symbiotic relationships with their environment to maintain a state of equilibrium. However, the spatial distribution of microorganisms in the human body is not homogeneous, varying according to habitat or anatomical site [San11]. In the digestive tract, for example, the concentration of microorganisms increases from the stomach, where the acidity and the presence of digestive enzymes are unfavorable for the development of bacteria, towards the colon where conditions are optimal for their growth, the temperature is constant (37 °C), the environment is not very acidic and is rich in water, transit is slow and food is abundant [Gau+16]. The gut microbiota found mainly in the human colon is a dynamic ecosystem whose development is influenced by several factors: genetics, age, geographical location, stress, diet, exposure to infectious agents or pollutants, and antibiotic intake [MD17; TJ17]. Numerous studies continue to reveal that the gut microbiota plays a crucial role in various aspects of our well-being [Arr+14], such as digestion, regulation of the immune system [HLM12], protection against infection, vitamin synthesis, and even influences on cerebral and metabolic functions [But+19]. Imbalances or alterations in this ecosystem can be associated with a wide range of health problems, from digestive disorders and autoimmune diseases to obesity and neurological disorders [CD12]. Certain disturbances in the composition and functions of the microbiota resulting in dysbiosis can lead to certain syndromes such as irritable bowel syndrome [Gau+16]. The microbiota can also be used as a medicine (using fecal transplants) to treat certain illnesses such as antibiotic-resistant diarrhea caused by *Clostridium difficile* [Gau+16]. In natural ecosystems, the soil microbiota actively participates in the decomposition of organic matter, releasing essential nutrients for plants, and some bacteria can establish symbiotic relationships with plant roots, promoting their growth by providing additional nutrients [Fie17;

[1] http://smai.emath.fr/cemracs/cemracs23/

MS15]. A better understanding of the microbiota could have a major impact on environmental sustainability, improving human health and managing ecosystems but despite advances in studies of bacterial ecological dynamics, the question of the nature of bacterial interactions in the intestinal microbiota remains open.

The Generalized Lotka-Volterra (GLV) model is commonly used to model and anticipate variations in microorganism populations within an ecosystem [VB31; Vol27; Lot56]. In the case of the microbiota, it could provide information on how changes in the composition and abundance of a microorganism population could influence the ecosystem [Ger14; BX14]. Furthermore, by analyzing the model parameters, we can identify the microbial species that have a significant influence on the dynamics of the microbiota as a whole [Seg+11; Fai+10], predict interspecies interactions [Ste+13], species coexistence [FHG17], and even community structure and dynamics [Buc+16].

However, when modeling large ecosystems with multiple species, simulating the GLV model may lead to computational challenges. This is particularly true in the context of parameter estimation, where many approaches (for instance Maximum Likelihood or MCMC estimation) explore the parameter space and simulate the model accordingly, potentially generating extensive simulation times and numerically unstable behaviors along the exploration process. Alternative approaches have been proposed in the literature, based on the so-called "metamodelisation" of the system trajectories, for instance through splines [Ram+07; Lar+18] or Gaussian processes [Don+13; Wen+19], embedded in a parameter estimation procedure. The main advantages of these approaches are to avoid numerical issues related to dynamical system simulation, and also to allow an easy integration of prior information in the trajectory reconstruction process, such as experimental data. The objective of the present work, in a first step towards parameter estimation, is to investigate the ability of Physics-Informed Neural Networks (PINNs) in providing such metamodels in the specific context of GLV ODEs, based on some typical issues they may raise.

PINNs is a recent and appealing approach that allows the fast and accurate simulation of large and complex dynamical systems [RPK19]. A PINN is a machine learning framework that combines neural networks with physics-based principles to efficiently solve complex physical problems. PINNs are designed to handle supervised learning tasks while respecting prescribed physical laws expressed through partial differential equations (PDEs) or ordinary differential equations (ODEs). They provide a powerful approach to solving problems based on PDEs and ODEs, but training PINN for certain ODE models with either sensitivity to initial conditions, or complex behavior, such as the ones presented above, requires an adequate architecture as well as a significant amount of synthetic training data and computational effort or an accurate emulation [Ant+22].

In the present preliminary work, we will design a PINN method, and apply it to the GLV model to simulate the evolution of bacterial species in presence of experimental data, potentially noisy. The paper is organized as follows: in Section B.1, we present the governing equations of the GLV model and provide some examples applied to the context of bacterial populations. Section B.2 introduces the PINN framework, and presents some preliminary results about the architecture employed in this study. The Section 6.3 shows the results obtained using the PINN to solve the GLV model, applied to synthetic cases of bacterial populations with and without added noise to the generated observations. Finally, in Section B.4 we present the GSA algorithm and its adaptation using PINN, and report our findings on the estimation of the parameters of the GLV model.

B.1 Generalized Lotka-Volterra model

In this section, we introduce the Generalized Lotka-Volterra model [Vol27]. This model has been developed to mathematically model the coexistence of various numbers of species in a closed system in a competitive or predator environment. Section B.1.1 is devoted to the introduction of the notation and the description of such model, while Section B.1.2 presents two examples that will be used in the sequel for numerical applications.

B.1.1 Description of the model

Assuming large and well-mixed bacteria population, with only bidirectional interactions between bacteria populations, and also assuming that their composition, diversity, and dynamics are not influenced by external factors or environmental conditions (*e.g.* physical and chemical parameters such as temperature, pH, humidity, light intensity, nutrient availability, and oxygen levels, host interactions, ...), then the evolution of N_s different species of bacteria population over a horizon of time $t_{\max}$ can be described by a Generalized Lotka-Volterra (GLV) model [VB31]:

$$\frac{\mathrm{d}x_i}{\mathrm{d}t}(t) = \mu_i x_i(t) + \sum_{j=1}^{N_s} a_{ij} x_i(t) x_j(t), \quad \forall t \in [0, t_{\max}], 1 \leqslant i \leqslant N_s, \tag{B.1}$$

with the initial condition $x_i(t=0) = x_{i0}$. The scalar $x_i(t)$ represents the abundance of the bacterial population of species i at time t, μ_i represents the intrinsic growth rate of the bacterial population i, and a_{ij} describes the interaction coefficient representing the direct effect of one unit of biomass of the bacterial population of species j on the growth rate of the bacterial population of species i.

Bidirectional interactions among these bacterial populations can be categorized into competition, cooperation, antagonism, and mutualism, but they often lack symmetry due to differences in species characteristics, environmental conditions, and ecological roles. For instance, a negative coefficient a_{ij} means that the population j has a negative impact on the growth of the population i, *e.g.* competition or predation by population j on population i. On the other hand, a positive coefficient a_{ij} means that the presence of the population j enables the population i to thrive, *e.g.* cooperation between the populations i and j or predation by population i on population j.

These asymmetric interactions are crucial for maintaining ecosystem balance; beneficial bacteria can inhibit pathogens through competition and antagonism, while cooperative relationships enhance resilience against infections. However, since a_{ij} and μ_i can take any signs and values, several behaviors can be observed [Har20; KG88; FS95; Jia98] such as oscillations, stable equilibrium with coexisting species, extinction of some species or even demographic explosion or chaos.

The simulation of GLV models can raise numerical issues, to prevent them and avoid unrealistic negative solutions, we assume positive initial conditions and use the logarithmic formulation of the model. For $t \in [0, t_{\max}]$, dividing by $x_i(t)$ in Equation (B.1) leads to the following formulation:

$$\frac{\mathrm{d}\log(x_i(t))}{\mathrm{d}t} = \mu_i + \sum_{j=1}^{N_s} a_{ij} x_j(t). \tag{B.2}$$

Setting $\underline{\boldsymbol{\mu}} = [\mu_1, \cdots, \mu_{N_s}]^T$, $\underline{\underline{\boldsymbol{A}}} = (a_{ij})_{1 \leqslant i,j \leqslant N_s}$ and $\underline{\boldsymbol{u}} = [u_1, \cdots, u_{N_s}]^T$ with $u_i = \log(x_i)$, Equation (B.2) can be written under the matrix form:

$$\frac{\mathrm{d}\underline{\boldsymbol{u}}(t)}{\mathrm{d}t} = \underline{\boldsymbol{\mu}} + \underline{\underline{\boldsymbol{A}}} \exp\left(\underline{\boldsymbol{u}}(t)\right), \tag{B.3}$$

with the initial condition $\underline{\boldsymbol{u}}(0) = \underline{\boldsymbol{u}}_0$.

In the following, the matrix $\underline{\underline{\boldsymbol{\theta}}}$ denotes the matrix of the GLV parameters, which contains the intrinsic growth rate and the interaction coefficients:

$$\underline{\underline{\boldsymbol{\theta}}} = \begin{bmatrix} \mu_1 & a_{11} & \dots & a_{1,N_s} \\ \vdots & \vdots & \ddots & \vdots \\ \mu_{Ns} & a_{N_s 1} & \dots & a_{N_s N_s} \end{bmatrix}.$$

In this study, the classical solver of the GLV model employed as reference is implemented in Python using the `odeint` function from the `scipy.integrate` library. The default solver of the library is used, namely `LSODA`. In the following, we denote by *exact solution*, or *true solution* $\underline{\boldsymbol{u}}_{\text{truth}}$ the trajectories computed with such solver knowing *a priori* the values of the parameters $\underline{\underline{\boldsymbol{\theta}}}$.

B.1.2 Illustrative cases for numerical simulation of the GLV model

As mentioned above, the solution of the GLV model can exhibit very different behavior. While GLV models with three species or fewer have been extensively studied [Jia98], there are few theoretical results on GLV models in higher dimensions, except for models with specific structures such as bounded competitive GLV (negative off-diagonal terms in the interaction matrix), cooperative systems (positive off-diagonal terms). We refer the reader to [Bai16] for a complete survey of available theoretical results.

If the parameters $(\underline{\boldsymbol{\mu}}, \underline{\underline{\boldsymbol{A}}})$ are taken randomly, the probability to reach a situation where only one species outlives while all the others go extinct increases with the system dimension. Species co-existence through stable oscillations or steady state is indeed non-generic.

A possible approach to obtain an oscillating model consists of generating a matrix $\underline{\underline{\boldsymbol{A}}}$ that possesses specific properties:

(i) its diagonal elements and the vector $\underline{\boldsymbol{\mu}}$ are set to zero, effectively nullifying any self-impact of species;

(ii) the matrix $\underline{\underline{\boldsymbol{A}}}$ is antisymmetric, guaranteeing that the interaction between species i and species j is equal in magnitude but opposite in impact to the interaction from species j to species i (predation interaction);

(iii) the sum of each row in $\underline{\underline{\boldsymbol{A}}}$ is precisely zero, thereby guaranteeing a compensatory effect for the impact of one species on another.

Assumptions (i) and (ii) ensure the total population to be constant over time (see [Bai16, Chap. 3]), whereas (iii) ensures that vectors with equal, strictly positive coordinates are fixed points of the system. In the case of systems with even dimensions, after [Bai16, Chap. 3], such systems are Hamiltonian, and a strictly convex Hamiltonian can be constructed, which guarantees (see e.g. [Mos76]) that the corresponding GLV model exhibits oscillatory dynamics. For systems exhibiting Hamiltonian behavior in even dimensions, exploring the use of symplectic numerical methods [Jin+20] provide a structured approach to preserve system properties such as energy conservation.

For the general GLV model, [Bai16, Theorem 5] provides a sufficient condition for the asymptotic convergence towards a stable, coexistence equilibrium, which refers to a situation where the interactions between the different species reach an equilibrium point where they coexist sustainably. Indeed, if the model parameters ensure that the growth rates and interactions among different species balance each other, resulting in a strictly positive equilibrium denoted as $-\underline{\underline{\boldsymbol{A}}}^{-1}\underline{\boldsymbol{\mu}}$, and if there exists a non-negative diagonal matrix $\underline{\underline{\boldsymbol{D}}}$ satisfying $\underline{\underline{\boldsymbol{A}\boldsymbol{D}}} + \underline{\underline{\boldsymbol{D}\boldsymbol{A}}}^{\top}$ being negative-definite, then the positive equilibrium is both stable and globally attractive in the positive region of the space. This characterization suggests a heuristic to generate such parameters:

1. Randomly generate a pair $(\underline{\boldsymbol{\mu}}, \underline{\underline{\boldsymbol{A}}}) \in \mathbb{R}^{N_s} \times \mathbb{R}^{N_s \times N_s}$, with $\underline{\boldsymbol{\mu}} \geq 0$ and the diagonal of $\underline{\underline{\boldsymbol{A}}} < 0$, and imposing 20 to 40 % of the extra-diagonal terms of $\underline{\underline{\boldsymbol{A}}}$ to be zero.

2. Solve the equation $\underline{\boldsymbol{\mu}} + \underline{\underline{\boldsymbol{A}\boldsymbol{X}}} = 0$. If any element of $\underline{\boldsymbol{X}}$ is negative, then go back to step (1), else continue to step (3).

3. Set $\underline{\underline{\boldsymbol{D}}} := \mathrm{diag}(\underline{\boldsymbol{X}})$, if $\underline{\underline{\boldsymbol{A}\boldsymbol{D}}} + \underline{\underline{\boldsymbol{D}\boldsymbol{A}}}^{\top}$ is negative-definite, then keep the generated parameters, else eliminate it and go back to step (1).

Note that in the limit case where all the extra-diagonal terms of $\underline{\underline{\boldsymbol{A}}}$ are zero, the GLV model becomes a set of uncoupled logistic growth equation, with a trivial positive stable steady state satisfying the proposed heuristic. Hence, by playing on the percentage of zeros in the extra-diagonal part of $\underline{\underline{\boldsymbol{A}}}$ or on their magnitude, the heuristic should provide adequate parameters, even for high dimensional models.

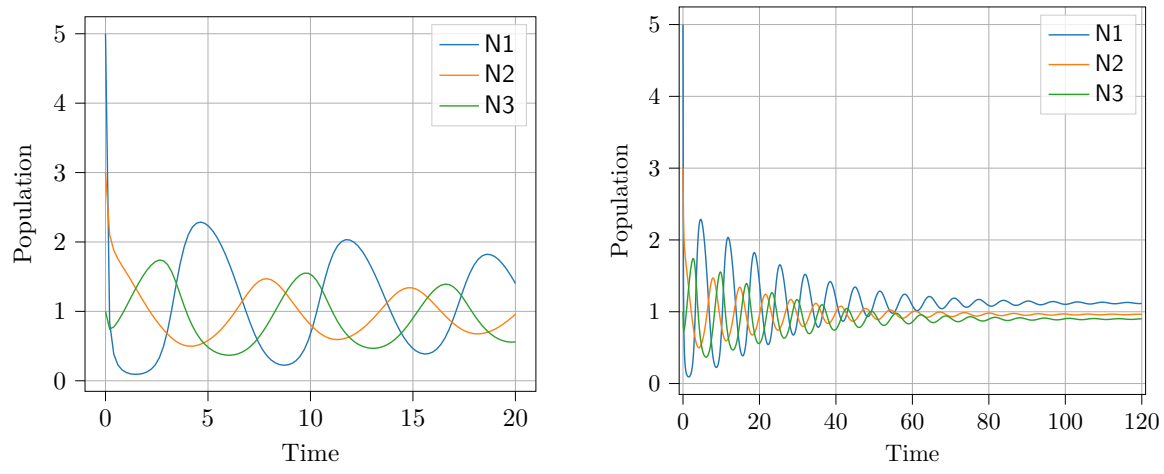
To illustrate the capability of the PINN to accurately capture the expected outcomes, two illustrative test cases with distinct behaviors will be considered. In this section, the test cases and the related numerical simulations are presented.

Example B.1.1. We introduce a first example with $N_s = 3$ bacterial species. The interaction matrix, the intrinsic growth rate, and the initial population are chosen as follows:

$$\underline{\underline{\boldsymbol{A}}}_3 = \begin{bmatrix} -2 & -5 & -0.5 \\ -0.5 & -1 & -1.2 \\ -1 & -0.5 & -1 \end{bmatrix}, \quad \underline{\boldsymbol{\mu}}_3 = [7.5, 2.6, 2.5]^T \text{ and } \underline{\boldsymbol{u}}_0 = [5, 3, 1]^T. \tag{B.4}$$

The growth of one species triggers a reduction in another, fostering a reciprocal cycle until a state of stationary equilibrium is achieved. Figure B.1(a) illustrates the evolution of these three populations over the time window of interest for this illustrative example, while Figure B.1(b) highlights that stationary equilibrium is achieved in the long term. The evolution over time of these three populations for this illustrative example is presented in Figure B.1.

Example B.1.2. We consider a bacterial population with $N_s = 20$ species. Using the algorithm described previously, we generate a parameter matrix $\underline{\underline{\boldsymbol{\theta}}}_{20} = (\underline{\boldsymbol{\mu}}_{20}, \underline{\underline{\boldsymbol{A}}}_{20})$ such that the system



(a) Time interval $(0, 10)$ s.

(b) Time interval $(0, 120)$ s.

Figure B.1: Results of the simulation of the GLV model with three populations of microbes for an initial population $u_0 = [5, 3, 1]^T$.

possess at stationary state $-\underline{\underline{\boldsymbol{A}}}_{20}^{-1}\underline{\boldsymbol{\mu}}_{20}$. The obtained matrix is available at https://gist.github.com/thomas-saigre/4f92bbb02221a335c4cbafd74b2441fb. An example of the simulation of the system, using random initial states is presented in Figure B.2. The theoretical limits for the populations are also shown. For this case, we select a final simulation time $t_{\max} = 20$ s.

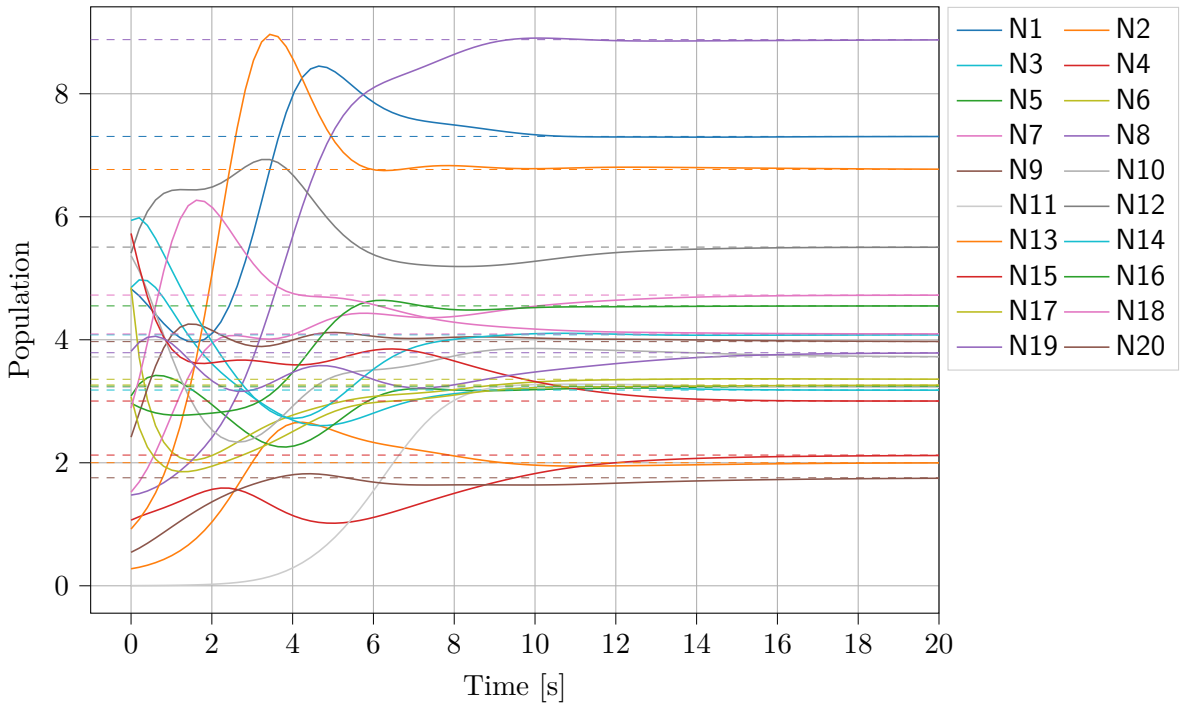


Figure B.2: Solution of the GLV model with a convergent state, where 20 species are considered, over the time interval $[0, 20]$ s. A random initial condition is considered. The dashed lines represent the theoretical values of the stationary stage $-\underline{\underline{\boldsymbol{A}}}_{20}^{-1}\underline{\boldsymbol{\mu}}_{20}$.

B.2 Physics-Informed Neural Network

This section introduces the neural network framework that is used to simulate the GLV model: Physics-Informed Neural Networks and the discussion about its architecture.

B.2.1 PINN framework

Physics-Informed Neural Networks or *PINNs* have been introduced in [RPK19] as neural networks designed to address supervised learning tasks while adhering to specified laws of physics outlined by nonlinear partial differential equations. It combines both supervised and unsupervised learning. In traditional supervised learning, the network learns from a labeled training dataset, where inputs and outputs are matched: the network aims at minimizing a loss function that measures the difference between its predictions and the data labels. In unsupervised learning, the network is exposed to unlabeled data, and its objective is to identify patterns or relationships within the data without explicit guidance from labeled examples. The PINNs are trained to simultaneously minimize the gap between the predictions and the training dataset and satisfy the governing physics equations, thereby incorporating both types of learning to achieve a comprehensive and physics-informed model.

The goal of the PINN is to construct a neural network approximation $\underline{\hat{\boldsymbol{u}}}_{\underline{\underline{\boldsymbol{\theta}}}}(t)$ of the solution $\underline{\boldsymbol{u}}(t)$ of Equation (B.3) based on the knowledge of this equation, where $\underline{\hat{\boldsymbol{u}}}_{\underline{\underline{\boldsymbol{\theta}}}} \colon [0, t_{\max}] \to \mathbb{R}^{N_s}$ denotes the function predicted by the network for a given parameter $\underline{\underline{\boldsymbol{\theta}}}$. For $t \in [0, t_{\max}]$, we introduce the *residual* $\mathcal{L}$ of the GLV model (B.3) with respect to the prediction $\underline{\hat{\boldsymbol{u}}}_{\underline{\underline{\boldsymbol{\theta}}}}$ at time t defined as

$$\mathcal{L}(\underline{\hat{\boldsymbol{u}}}_{\underline{\underline{\boldsymbol{\theta}}}}; t) := \frac{\mathrm{d}\underline{\hat{\boldsymbol{u}}}_{\underline{\underline{\boldsymbol{\theta}}}}(t)}{\mathrm{d}t} - (\underline{\boldsymbol{\mu}} + \underline{\underline{\boldsymbol{A}}} \exp(\underline{\hat{\boldsymbol{u}}}_{\underline{\underline{\boldsymbol{\theta}}}}(t))). \tag{B.5}$$

In the present work, we have in view a specific situation where, unlike in the usual framework of PINNs, the initial condition of the GLV model is not exactly known, and it is assumed to be observed together with other observations at other time points. For this reason, we add second information based on a sample of data $\underline{\underline{U}}^{(e)}$. The data, in this specific contribution, are generated numerically, but eventually experimental data should be provided. By averaging measurement errors, introducing these data should reduce the sensitivity to errors on the initial condition and improve the accuracy of our predictions given a fixed grid time points. Thus, the loss function should both satisfy the model (B.3) and fit the data $\underline{\underline{U}}^{(e)}$.

The neural network approach proceeds by using an optimizer to update the weights and bias by minimizing a loss function. In our implementation, we employ the optimizer Adam, implemented in the PyTorch library. This loss function is defined as a linear combination of quantities that measures the quality of our prediction. Specifically in the context of GLV model, we introduce two types of errors for the development of our PINN:

- MSE_{data}: the mean squared misfit by the data, also called the *data loss*, which is used to assess the extent to which the model can faithfully reproduce the data presented

$$MSE_{\text{data}}(t^{(e)}) = \frac{1}{N_s N^e_{\text{obs}}} \sum_{i=1}^{N_s} \sum_{k=1}^{N^e_{\text{obs}}} \left(\underline{\hat{\boldsymbol{u}}}^i(t^{(e)}_k) - \underline{\underline{U}}^{(e)}_{i,k} \right)^2, \tag{B.6}$$

  for an experiment e, $t^{(e)} = (t^{(e)}_i)_{i=1}^{N^e_{\text{obs}}}$ and N^e_{obs} are respectively the time of observations and the number of observations, $\underline{\underline{U}}^{(e)}_{i,k}$ represent the quantity of bacterial population of species i observed at time $t^{(e)}_k$ and $\underline{\hat{\boldsymbol{u}}}^i_\theta$ is the neural network prediction of the bacterial population of species i: $\underline{\hat{\boldsymbol{u}}} = \left(\underline{\hat{\boldsymbol{u}}}^i\right)_{i=1}^{N_s}$;

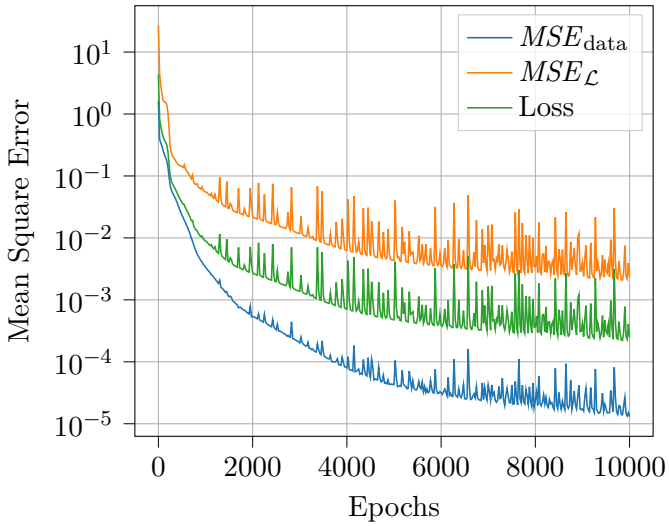


Figure B.3: Evolution of the data loss MSE_{data} (B.8), physical loss $MSE_{\mathcal{L}}$ (B.7) and loss function Loss (B.8), over the epochs during the training of the PINN for the Example B.1.2

- $MSE_{\mathcal{L}}$: the mean squared residual, also called the *physical loss*, which enforces the structure imposed by (B.5) at a finite set of collocation points $t_r = \{t_j\}_{j=1}^{N_f} \subset [0, t_{\max}]$

$$MSE_{\mathcal{L}}(t_r) = \frac{1}{N_s N_f} \sum_{i=1}^{N_s} \sum_{j=1}^{N_f} \mathcal{L}_i(\hat{\underline{\boldsymbol{u}}}; t_j)^2, \tag{B.7}$$

  where $\mathcal{L}(\hat{\underline{\boldsymbol{u}}}; t) = (\mathcal{L}_i(\hat{\underline{\boldsymbol{u}}}; t))_{i=1}^{N_s}$.

Note that the influence of the initial conditions is included in the model, by the loss MSE_{data}, but, as mentioned above, due to potential measurement errors, there is no reason to distinguish it from other available data. We introduce the loss function to be minimized, involving a hyperparameter $\lambda^{\mathsf{PINN}} > 0$:

$$\text{Loss} = MSE_{\text{data}}(t^{(e)}) + \lambda^{\mathsf{PINN}} MSE_{\mathcal{L}}(t_r). \tag{B.8}$$

For all our numerical experiments, unless otherwise specified, we take $N_f = 100$ collocations points and $N_{obs} = 10$ observational data points. We present in Figure B.3 an example of the evolution of the MSEs of the loss, over the iteration during the learning process of the PINN. The loss employing a value of $\lambda^{\mathsf{PINN}} = 0.1$ is also presented. The results presented are obtained from the case with 20 bacterial species, introduced in Example B.1.2. We remark that after a certain number of epochs, the global loss decreases steadily, demonstrating the learning process of the PINN, while the data loss stabilizes at a lower value, and the physical loss remains higher, likely due to the complexity of the governing equations.

B.2.2 PINN architecture

In this section, we discuss the chosen architecture for the neural network trained within the PINN framework exposed above. A single neural network will correspond to a single experiment e, and will therefore be trained to predict, for a time t as input, the population of the N_s species in the given experiment. By adopting a single neural network, the model is tasked with predicting the population dynamics of all N_s species at a given time t within the specific experiment. The

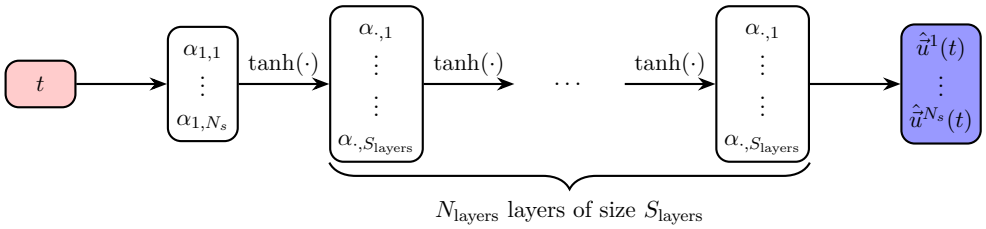


Figure B.4: Proposed architecture for the PINN framework: an input layer t, a second layer of size N_s, N_{layers} of size S_{layers}, and an output layer of size N_s to predict the population of the different species evaluated at time t. We use a hyperbolic tangent activation function for all our layers.

rationale behind this decision is to streamline the training process and enhance efficiency instead of considering a different neural network for each species. This consolidated architecture simplifies the complexity of the overall framework and promotes a more unified and manageable training procedure. This strategic choice aimed at achieving a balance between computational efficiency and predictive accuracy, providing a pragmatic solution for the modeling objectives within the PINN framework. Thus, the proposed architecture, outlined in Figure B.4, is composed of successive neural layers of various sizes and utilizes the hyperbolic tangent as activation function. The hyperparameters governing the number and size of intermediate layers, denoted as N_{layers} and S_{layers} respectively, are subject to tuning, which is detailed in Section B.2.3.

This chosen architecture aims to strike a balance between model complexity and predictive accuracy, leveraging the interconnectedness of species dynamics while maintaining computational feasibility. The choice of layer sizes being proportional to the number of species N_s allows the network to effectively handle the complexity of interspecies interactions, maintaining a sufficiently rich feature space. Additionally, it is possible to improve the network's ability to capture oscillatory dynamics by incorporating Fourier features [Tan+20], which could be particularly beneficial for cases like Example B.1.1, where periodic behaviors are prominent. This enhancement will be explored in future work.

In the following, we present two different aspects that have been investigated to increase the effectiveness of the PINN framework, notably the strategy to tune the architecture hyperparameters and the application of time normalization to the classical GLV model.

B.2.3 Selection of hyperparameters

Setting up the PINN described in Section B.2.1 requires choosing λ^{PINN} as well as the architecture of the multilayer perceptron (the number of layers N_{layers}, and the size of these layers S_{layers}) and the number of epochs used in training, N_{epochs}, which corresponds to how many times the entire dataset is passed through the neural network during the training process. We decided to use Optuna [Aki+19], an open-source hyperparameter optimization framework, to search the hyperparameter space to find the best value for these hyperparameters with respect to a chosen metric. This metric, E_{PINN}, will be the relative error of our prediction concerning the true solution at each collocation point:

$$E_{\text{PINN}} = \frac{1}{N_s} \sum_{j=1}^{N_s} \frac{\left\| \hat{\boldsymbol{u}}^j - \boldsymbol{u}^j_{\text{truth}} \right\|^2_{L^2([0,t_{\max}])}}{\left\| \boldsymbol{u}^j_{\text{truth}} \right\|^2_{L^2([0,t_{\max}])}}. \tag{B.9}$$

We explore the tuning of the hyperparameters N_{epochs}, N_{layers} and S_{layers}. The objective value optimized by Optuna, presented in the following results, is the logarithm of the error E_{PINN}.

When selecting the architecture of the PINN, we also need to keep in mind the computational cost that would be required by a more complex architecture: the more parameters are involved, which are directly influenced by N_s, N_{layers} and S_{layers}, the more time will be needed to perform the training of the neural network. Unlike the other hyperparameters, we did not perform specific tuning for λ^{PINN}, as its value depends on the confidence in the model versus the data. In this study, λ^{PINN} was chosen deterministically based on prior knowledge, and a thorough exploration of its optimization is beyond the scope of this work.

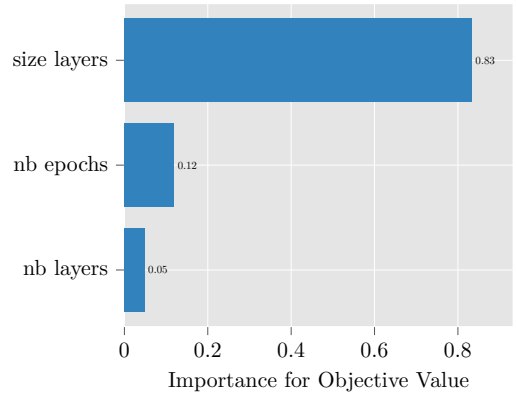
We present in Figure B.5 various findings of our investigation performed on the oscillatory test case with $N_s = 3$. Precisely, Figure B.5(a) shows the impact of the hyperparameters on the prediction error E_{PINN}, while Figure B.5(b) illustrates the error distribution map observed as the size S_{layers} and number of layers N_{layers} vary. As expected, the results indicate that the error is smaller when these two parameters are higher. The interesting finding is that the S_{layers} affects more the relative error than N_{layers}, see Figure B.5(a). Moreover, when $N_{\text{layers}} \geq 2$ we do not see such an improvement in the results, this behavior is also notable for $S_{\text{layers}} > 20$. In Figure B.5(c) we display the evolution of the prediction error E_{PINN} according to the value of the most influential hyperparameter S_{layers}. We recover the fact that the wider the layers are, the more precise the prediction is and the threshold $S_{\text{layers}} > 20$ where no appreciable increase of accuracy is highlighted. In order to understand the operations carried out by Optuna, we additionally use different colors to illustrate each stage within the trial process. Finally, we performed the same study, focusing on the computational time to find a balance in terms of efficiency for the architecture of the PINN, as presented in Figure B.5(d). The impact is measured on two metrics: the prediction error E_{PINN} and the computational time needed to train the neural network, at a fixed number of epochs. We recover that the size of the layers impacts the relative error, but less affects the time of simulation: the training of a deeper neural network will take more time than the training of a wide one.

In light of all these results, we hereafter select the hyperparameters $N_{\text{epochs}} = 2000$, $N_{\text{layer}} = 2$ and $S_{\text{layer}} = 7 \times N_s$, giving the following architecture to the neural network: $[1, N_s, 7 \times N_s, 7 \times N_s, N_s]$.

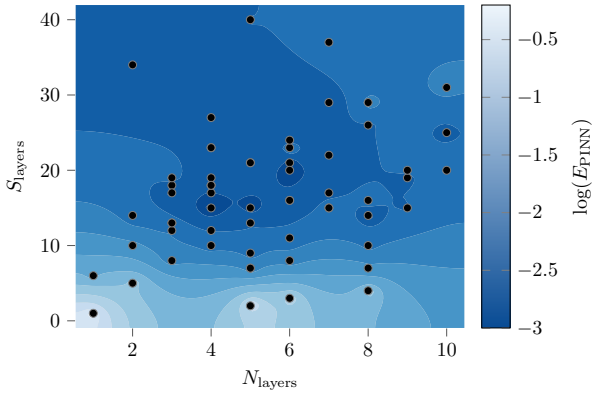
The final number of trainable parameters of the network is then dependent on the number of species. For the two examples presented in this work, with $N_s = 3$ and $N_s = 20$, we have respectively 571 and 25,221 trainable parameters in the neural network.

While we utilized Optuna for hyperparameter optimization, our experiments emphasized the more challenging oscillatory dynamics with $N_s = 3$, as this presents a greater challenge for neural networks compared to stationary dynamics. The architecture's proportional scaling with N_s ensures that the increase in the number of species does not impede the network's effectiveness across different system sizes.

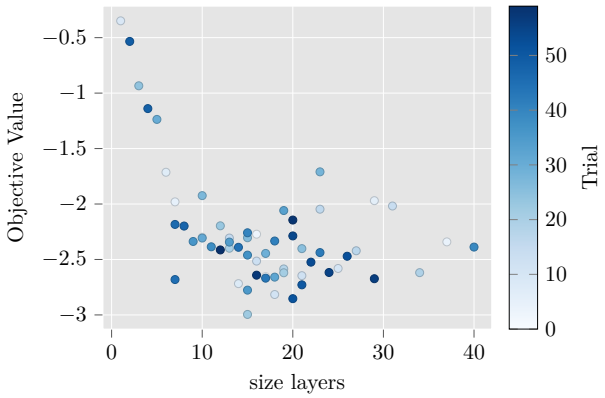
To enhance the stability of numerical computations during the training of PINN, expedite the convergence of PINN learning by mitigating potential issues like vanishing gradients and gradient explosions as discussed in [Kur20; Yas21], and facilitate the optimization process by improving the handling of diverse scales in model parameters, we propose a modification to the GLV model introduced in Section B.1.1. Originally defined over a finite time interval $[0, t_{\max}]$, the model is reformulated by normalizing the time, leading to a formulation defined over a normalized time interval $[0, 1]$. Normalization is a widely adopted approach in machine learning for enhancing neural network performance. In this context, specifically, data contain significant variations from one timescale to another, time normalization would help the model to focus on the underlying patterns (dynamics of competition, predation, cooperation, or other interactions between species that persist over significant time scales despite noise or short-term variations in the data) rather than on variations in amplitude. With this rationale, to obtain the normalized version of the GLV model, we perform the following change of variable $T := t/t_{\max}$. Hence, the GLV model



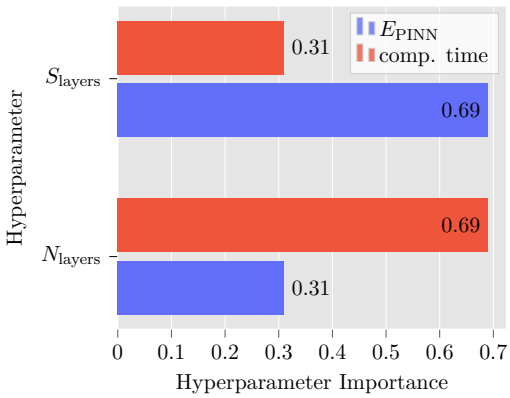
(a) Hyperparameters importance: impact of the architecture of the neural network (numbers N_{layers} and size S_{layers} of the layers), and number of epochs for the training set.



(b) Contour plot showing the distribution of the relative error of the PINN according to its architecture.

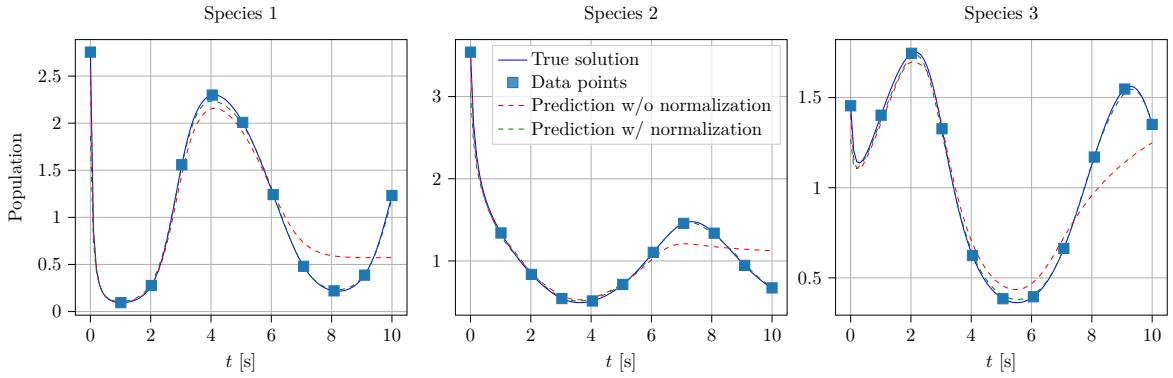


(c) PINN relative error against size of layers S_{layers}. The color bar represents the trial stage of Optuna execution.

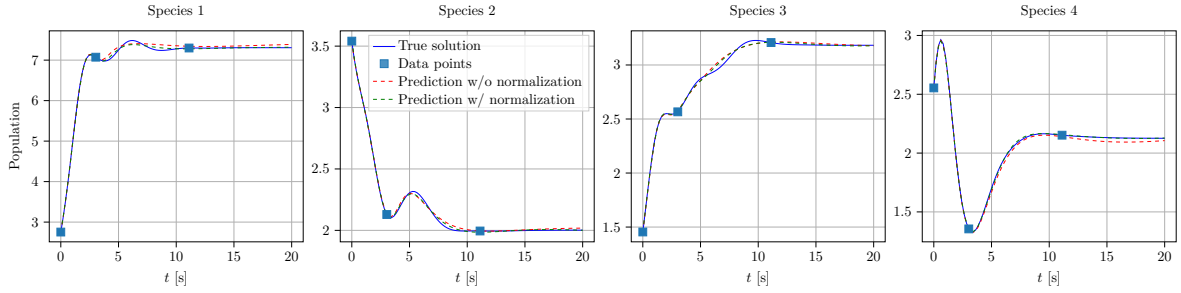


(d) Effectiveness of the hyperparameters: impact of neural network dimensions on the prediction error E_{PINN} and the computational time.

Figure B.5: Results obtained with Optuna on the PINN architecture: tuning of hyperparameters: number of layers N_{layers}, layer size S_{layers}, and number of epochs N_{epochs}.



(a) $N_s = 3$.



(b) $N_s = 20$, only 4 species are presented.

Figure B.6: Comparison of the PINN predictions with and without time normalization. The full blue line is the true solution of the model, and the dashed lines are the predictions of the two PINNs, with time normalization (green) or without it (red). The appropriate scaling for $\lambda_2^{\mathsf{PINN}}$ is applied.

(B.3) is rewritten as:

$$\frac{\mathrm{d}\underline{\boldsymbol{u}}(T)}{\mathrm{d}T} = t_{\max}(\underline{\boldsymbol{\mu}} + \underline{\underline{\boldsymbol{A}}} \cdot \exp(\underline{\boldsymbol{u}}(T))) \quad \text{for } T \in [0,1]. \tag{B.10}$$

This change of variable leads to a change also in the loss function (B.8), specifically in the physical loss $MSE_{\mathcal{L}}$ (B.7) leading to change of the hyperparameter λ^{PINN}:

$$\lambda^{\mathsf{PINN}}_{\text{normalized}} = \lambda^{\mathsf{PINN}}(t_{\max})^2. \tag{B.11}$$

On Figure B.6, the PINN predictions $\underline{\hat{\boldsymbol{u}}}$ obtained with the original GLV model (B.3) and with the normalized GLV model (B.10) using the appropriate scaling of λ^{PINN} are compared for both cases introduced in Section B.1.2.

These results suggest that the prediction is better when time normalization is used, confirming our hypotheses. Specifically, for the oscillatory test-case presented, see Figure B.6(a), the mean relative error on the predicted trajectories (B.9) is $E_{\text{PINN}}^{\text{w/o norm}} = 0.14$ without the normalization and $E_{\text{PINN}}^{\text{w/ norm}} = 2.04 \cdot 10^{-2}$ with the normalization. For the second test case with $N_s = 20$ species, see Figure B.6(b), even though it is less striking than in the oscillatory case, the time normalization enables to improve the prediction accuracy. The mean relative error on the predicted trajectories is $E_{\text{PINN}}^{\text{w/o norm}} = 0.11$ without the normalization and $E_{\text{PINN}}^{\text{w/ norm}} = 7 \cdot 10^{-2}$ with the normalization.

We figure that with the time normalization, the error gains one order of magnitude. Hence, we will consider the normalized version of the PINN for further simulations. Our tests indicate

that logarithmic normalization of concentrations does not significantly increase training duration while improving the accuracy of the results, making it a suitable choice for our analysis. Hereafter, by abuse of notation, t will refer to the normalized time variable and λ^{PINN} to the normalized hyperparameter $\lambda^{\mathsf{PINN}}_{\text{normalized}}$.

B.3 Numerical results over the influence of data

This section presents some numerical results obtained with the PINN described in the previous section. The cases introduced in Examples B.1.1 and B.1.2 will be both used in the following to compare $\underline{\hat{\boldsymbol{u}}}$ the approximation computed by the PINN with the reference solution $\underline{\boldsymbol{u}}_{\text{truth}}$ computed by solving the GLV model introduced in Section B.1 for the two specific sets of given parameters $\underline{\underline{\boldsymbol{\theta}}}$. In the present study, using the reference solution of the GLV model, we generate synthetic data to train the PINN. Still, the impact of different behavior of the dataset on $\underline{\hat{\boldsymbol{u}}}$ is highlighted in this study in order to test the robustness of the proposed approach to potential issues that may arise when working with real data. To do so, the following situations are considered hereafter:

1. generating data on a fixed equidistant grid over the time interval $[0, t_{\max}]$ or selecting those times randomly from a uniform distribution;
2. intentionally reducing the number of available data, to simulate missing or few data in actual experiments;
3. adding noise to the data, to simulate the noise in actual experiments.

In the following we analyze the accuracy and the efficiency of the PINN with respect to these three situations for the two illustrative examples presented in Section B.1.2.

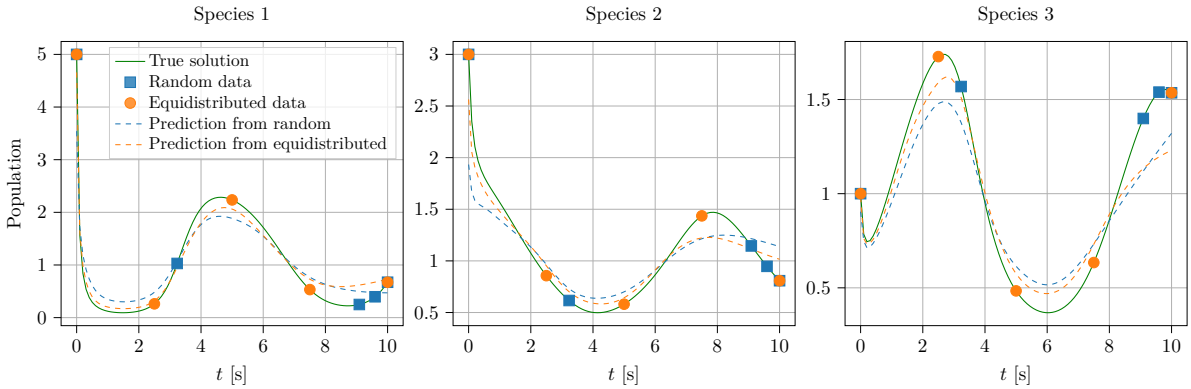
B.3.1 Impact of data sampling strategy

In this section, we focus on the impact of the data sampling on the accuracy of the prediction of the PINN. To begin with, we look at the distribution of the data over the time interval $[0, t_{\max}]$. Two data sets of size $N_{\text{obs}} = 5$ are generated. In the first data set, the data are generated over a uniform grid of the time interval. In the second dataset, the data are generated at times randomly sampled following a uniform distribution in the time interval. The predictions of the PINNs trained with these two datasets and for the two test cases are shown in Figure B.7. During the training process, all the collocation points are taken identical and equidistributed over the interval $[0, t_{\max}]$. For the case with 3 species, see Figure B.7(a), the errors are $E_{\text{PINN}} = 0.17$ for random times and $E_{\text{PINN}} = 0.13$ for equidistant time ones. On the other hand, the case with 20 species results in $E_{\text{PINN}} = 3.56 \cdot 10^{-3}$ for random times and $E_{\text{PINN}} = 3.37 \cdot 10^{-3}$ with equidistant time ones, see Figure B.7(b).

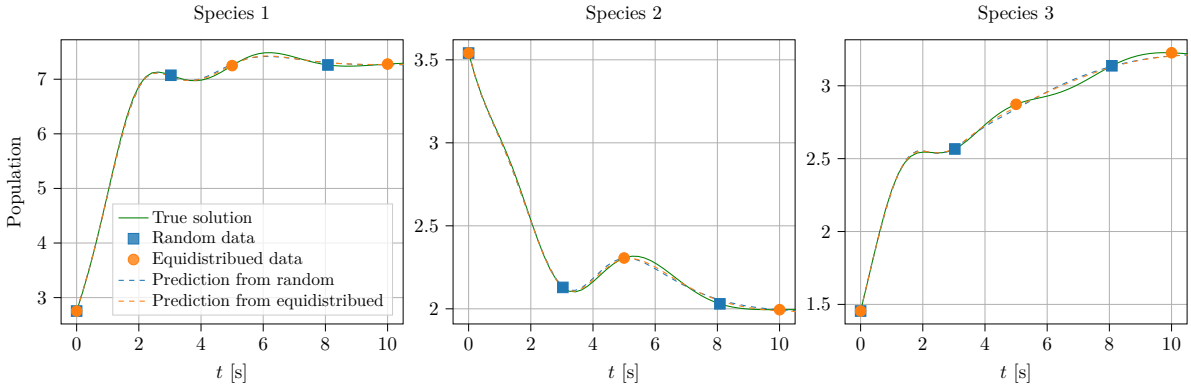
For both cases, the impact of the distribution of the data on the PINN's accuracy is not remarkable: the resulting PINN approximations $\underline{\hat{\boldsymbol{u}}}$ are similar using different strategies. This outcome is interesting from an experimental point of view, where the sampling strategy can be constrained.

B.3.2 Influence of the number of data used in the training on the network's prediction

First, we focus on the oscillatory example with $N_s = 3$ species, described in Example B.1.1. From the model parameters $\underline{\underline{\boldsymbol{\theta}}}_3$, two sets of data with various sizes are generated: one with $N_{\text{obs}} = 2$ and $N_{\text{obs}} = 12$. It should be noted that the case $N_{\text{obs}} = 1$ (usually the initial condition) corresponds to the standard framework of PINNs for ODEs, thoroughly investigated in the

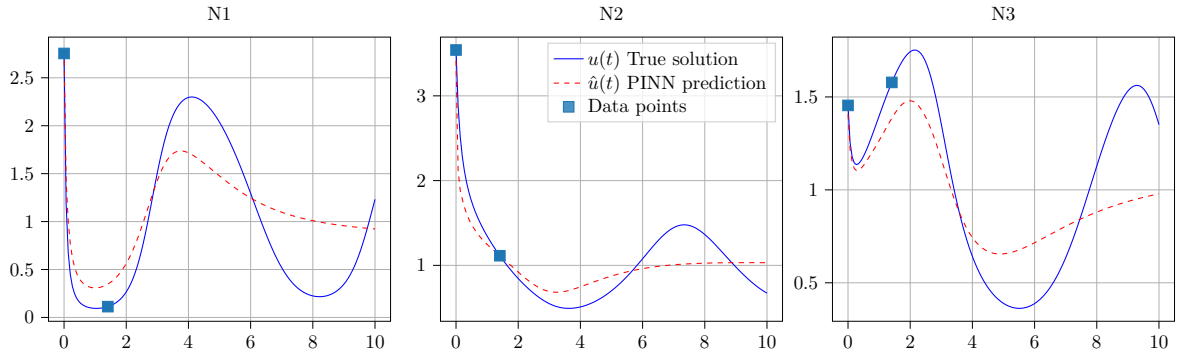


(a) Case 1 with $N_s = 3$.

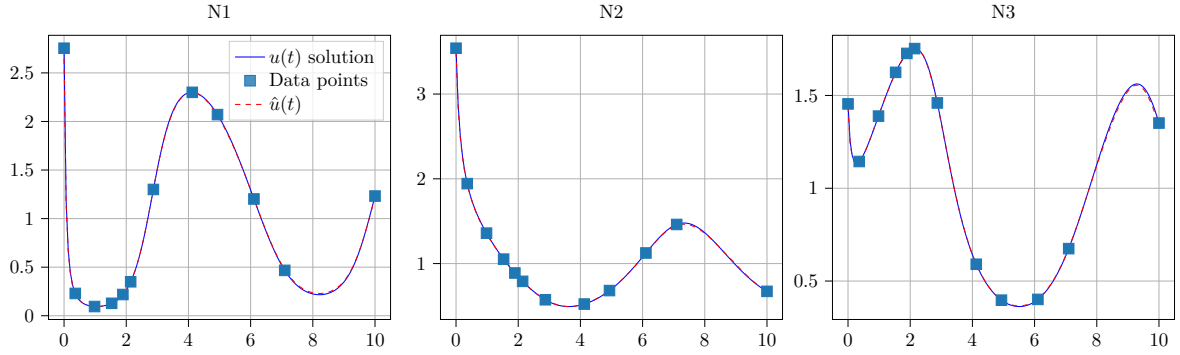


(b) Case 2 with $N_s = 20$, only 3 species are shown. Similar results stand for all other species.

Figure B.7: PINNs predictions trained with data generated on a fixed equidistant time grid (orange dashed line) and on times randomly selected from a uniform distribution (blue dashed line) over the time interval $[0, t_{\max}]$.



(a) With 2 data points in the training set.



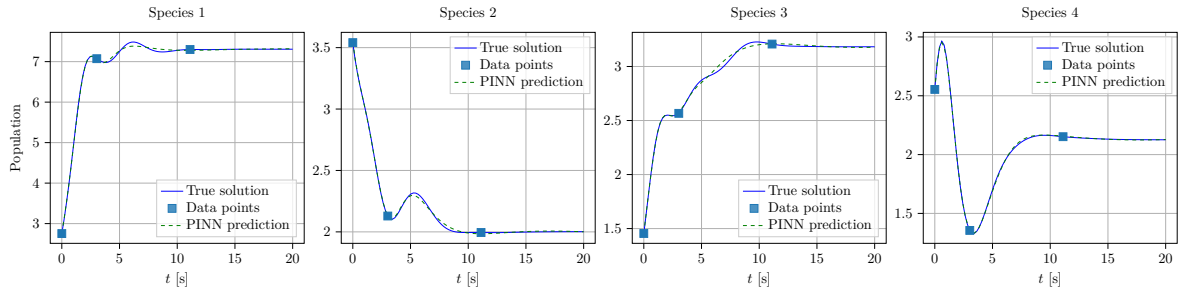
(b) With 12 data points in the training set.

Figure B.8: Prediction of the PINN with various numbers of points used for the training set. The population is composed of three species (Example B.1.1). The full blue line is the true solution of the model, the dashed red line is the model's prediction. The square points are the data used for the training of the PINN.

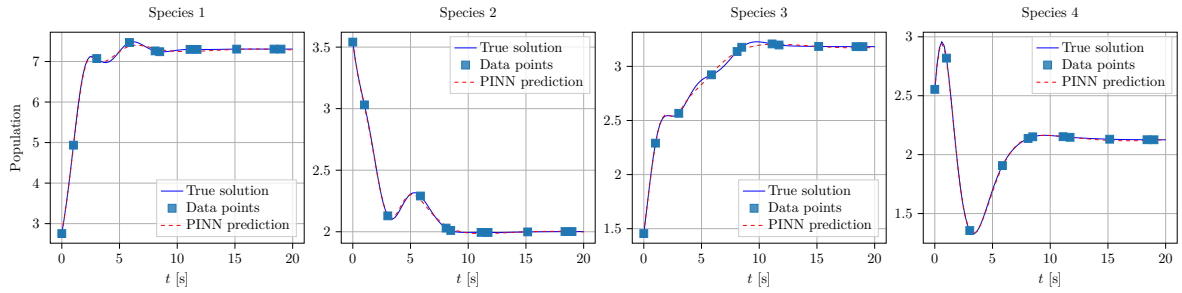
existing literature [BB23]. The results of the prediction of the PINN for both sets of data are presented in Figure B.8. For a small number of data, the PINN is not able to properly approximate the reference solution and the predicted trajectories are not able to capture the behavior of the true solution. However, as the number of data increases, the prediction becomes more accurate. It is also highlighted by the mean relative errors (B.9) that is $E^{(2)}_{\text{PINN}} = 0.25$ for the PINN trained with 2 observation data and $E^{(12)}_{\text{PINN}} = 7 \cdot 10^{-2}$ when 12 observations are provided.

Secondly, we analyze the test case where 20 species are considered, with the convergence toward a stationary state (Example B.1.2). The results are presented in Figure B.9 for two different training sets: one with $N_{\text{obs}} = 3$ (Figure B.9(a)) and one with $N_{\text{obs}} = 11$ (Figure B.9(b)). Note that only 4 species are presented. We remark that, with a small number of observations, unlike the oscillatory case, the PINN tends to approximate fairly well the trajectories and fits perfectly the theoretical limit. Precisely, the error is $E^{(3)}_{\text{PINN}} = 5.16 \cdot 10^{-3}$. Doing the same with more data ($N_{\text{obs}} = 11$), see Figure B.9(b), we remark that predictions of the PINN are similar to the previous case with $E^{(11)}_{\text{PINN}} = 5.55 \cdot 10^{-3}$. One can conclude that in this case, the number of observations does not influence the PINN approximation and especially the estimation of the stationary state.

In analyzing the results, it is noteworthy that the oscillatory test case exhibits poorer accuracy compared to the stationary counterpart. This disparity can be attributed to the chosen activation function, which inherently struggles to replicate periodic behaviors without



(a) $N_{\text{obs}} = 3$.



(b) $N_{\text{obs}} = 11$.

Figure B.9: Prediction of the PINN with various numbers of points used for the training set N_{obs}, for the test case with $N_s = 20$ (only 4 species are presented). The full blue line is the true solution of the model, predictions of the PINN with and without time-normalization are drawn in dashed lines, green and red respectively.

additional constraints from more data to guide the predicted trajectories of PINNs. It is worth mentioning that alternative activation functions (*e.g.*, cosine) could successfully capture periodic patterns but might fall short in reproducing stationary behaviors [RML23]. Given the need to make this activation function choice "offline" before knowing the specific parameters and the system behavior, we have consciously adhered to the hyperbolic tangent choice, prioritizing the reproduction of stationary behavior. This decision aligns with our focus on biological applications that often involve steady-state scenarios. In specific circumstances, however, alternative activation functions can be employed in the proposed approach to better simulate oscillating behaviors.

In the oscillatory test case, the inclusion of data points, such as through data loss, is crucial to avoid local minima and prevent constant incorrect predictions, even when no noise is present in the data. Conversely, in the stationary case, while the model can function without such data, adding data points enhances accuracy and reduces computational time, as shown in our preliminary tests. These findings are consistent with previous results reported in [BB23].

B.3.3 Adding noise

In order to evaluate the proposed methodology on more realistic data, we conducted some tests with noisy synthetic data.

From deterministic generated data y, we define the following multiplicative noise y_{noisy} as:

$$y_{\text{noisy}} = \text{Log-}\mathcal{N}\left(\mu = \ln(y) - \frac{1}{2}\sigma^2, \sigma = \ln(1+\nu^2)\right),$$

where ν is the desired ratio between the standard deviation and the value ($\nu = \frac{\text{std}}{y}$). In the context of noisy data for the GLV model, we select a value for ν between 0.1 and 0.3.

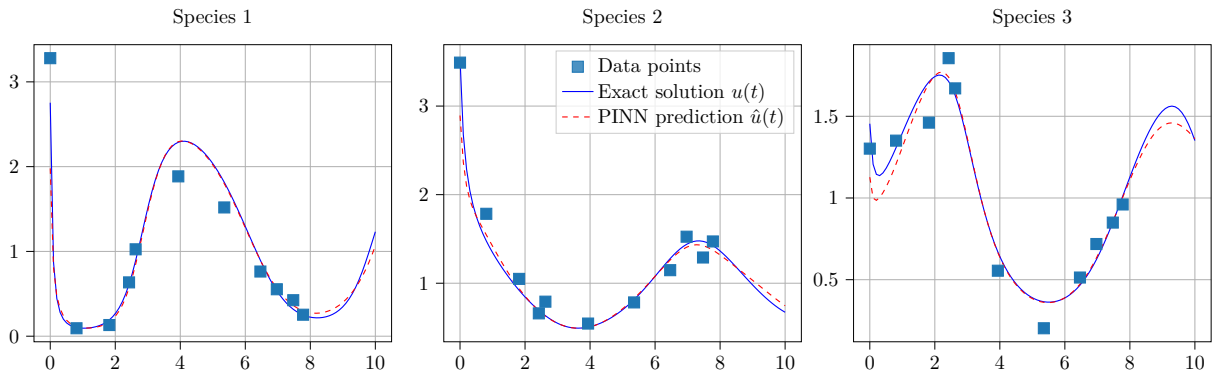


Figure B.10: Prediction of the PINN with noisy data, for the oscillatory case with $N_s = 3$ species.

To begin with, we present the results for the oscillatory case with 3 bacterial species, introduced in Example B.1.1, see Figure B.10. In the figure, the exact solution is still plotted, even if the data are noisy, as we do not expect the predicted trajectory to be equal to the exact solution. The error resulting from the PINN approximation is $E^{(3)}_{\text{PINN}} = 3.18 \cdot 10^{-2}$.

Now we focus on the second example introduced in Example B.1.2 with 20 species, see Figure B.11. The error resulting from the PINN approximation is $E^{(20)}_{\text{PINN}} = 4.89 \cdot 10^{-2}$.

For both test cases, the predicted trajectories are quite close to the reference solutions and the errors between the PINN approximations and reference solutions are fairly low. This tells that the developed PINN model is robust to noisy data, which is a positive outcome of the method. Note that such results are obtained to a fine-tuning of the hyperparameter λ^{PINN}.

B.4 Generalized smoothing approach

The *Generalized smoothing approach* (GSA) was introduced by Ramsay and co-authors in [Ram+07; RS05] as a method to estimate parameters in a model involved in a parameter-dependant nonlinear differential equation of the form

$$\dot{\underline{\boldsymbol{u}}}(t) = \underline{\boldsymbol{f}}(\underline{\boldsymbol{u}}, t; \underline{\boldsymbol{\theta}}). \tag{B.12}$$

The method described permits estimating the parameters $\underline{\boldsymbol{\theta}}$ of the model using noisy data. The work from [Ram+07] has been adapted in [Lar+18] for the GLV model presented in Section B.1.1, in the context of microbial population estimation.

B.4.1 GSA Least squares algorithm

In this section, we describe the GSA algorithm, as adapted in [Lar+18] for the GLV model. As a step of minimization is performed using the least squares (LS) method, we will denote by *GSA-LS* this algorithm.

It uses splines in order to represent the abundances of bacterial species across time, which should be a solution of the GLV model Equation (B.3), as well as being close to the experimental data provided. Consequently, the proposed method employs a joint estimation strategy for the spline coefficients and model parameters. This is achieved through alternate minimization of a goal function that considers three key components: *(i)* the proximity of splines to the data, *(ii)* a penalty associated with the deviation of the splines as a solution to a GLV model with respect to the estimated parameters, and *(iii)* a sparsity penalty applied to these parameters. This approach is adopted to effectively represent bacterial species abundances across time using

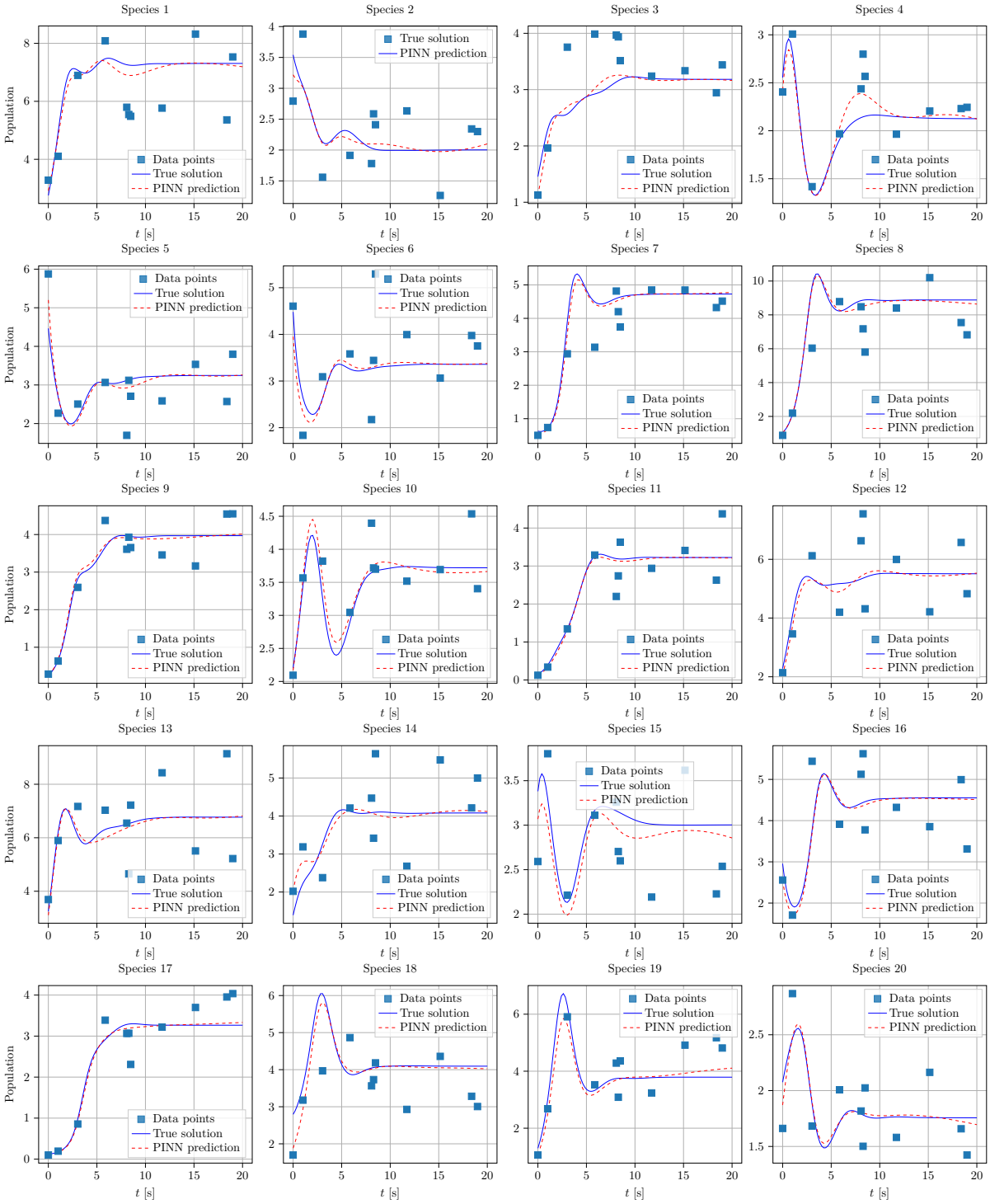


Figure B.11: Prediction of the PINN with noisy data for the stationary test case with $N_s = 20$ species.

splines, ensuring their alignment with the GLV model Equation (B.3) while closely adhering to the provided experimental data.

In particular, the algorithm is defined by the following steps :

Step 0 In this step, a basis $\underline{\underline{\boldsymbol{\Phi}}}$ of spline functions is constructed. These splines act as flexible building blocks that capture the underlying trends in the data while minimizing undesirable oscillations and artifacts. The construction of these spline functions forms the foundation for subsequent parameter estimation and coefficient optimization. The coefficients of the spline function fitting the data are stored in a matrix $\underline{\underline{\boldsymbol{C}}}$.

Step 1 In this step, an estimate of the model parameters $\underline{\boldsymbol{\theta}}$ is performed using the proximal gradient descent technique. Proximal gradient descent combines the iterative nature of gradient descent with a proximity operator that enforces a specific constraint on the estimated parameters. This constraint can be chosen to promote sparsity or other forms of regularization, thereby preventing overfitting and enhancing the model's ability to capture essential data features.

Step 2 In this phase, new coefficients of the spline basis are computed using the least squares minimization approach. The primary objective is to minimize the discrepancy between the model's predictions and the actual data points. By iteratively adjusting the coefficients, the algorithm ensures that the spline functions adeptly capture the underlying data distribution while minimizing the squared residuals. This optimization process contributes to refining the model's representation of the data. In [Lar+18], this step of optimization is performed using the least square minimization.

Precisely, at each step, a linear combination of functions is minimized. To lighten the notation, we consider here that data from only one experiment is used. To quantify the proximity between the fitted trajectories and the data, we introduce the function J_1, which assesses this closeness while incorporating a term to account for the smoothing applied.

$$J_1(\underline{\underline{\boldsymbol{C}}}) = \sum_{e=1}^{N_{\text{exp}}} \sum_{k=1}^{N_{\text{obs}}^{(e)}} \sum_{i=1}^{N_s} \left| \tilde{u}_i^{(e)}(t_k^{(e)}) - \underline{\underline{\boldsymbol{U}}}_{i,k}^{(e)} \right|^2, \tag{B.13a}$$

where $\tilde{u}_i^{(e)}(t) = C^{(e)}\underline{\underline{\boldsymbol{\Phi}}}(t)$ is the spline reconstructed solution for the species i, and the experiment e.

Moreover, to ensure the fitted trajectories fit the model Equation (B.3), we introduce the function J_2:

$$J_2(\underline{\underline{\boldsymbol{C}}}, \underline{\boldsymbol{\theta}}) = \sum_{e=1}^{N_{\text{exp}}} \frac{1}{N_f} \sum_{j=1}^{N_f} \left\| \dot{\underline{\tilde{\boldsymbol{u}}}}^{(e)}(t_j) - \underline{\boldsymbol{f}}(\underline{\tilde{\boldsymbol{u}}}^{(e)}(t_j), t_j, \underline{\boldsymbol{\theta}}) \right\|_2^2, \tag{B.13b}$$

where $(t_j)_{j=1}^{N_f}$ is a family of collocation points, equidistributed over $[0, 1]$.

Finally, to avoid $\underline{\boldsymbol{\theta}}$ taking abnormally huge values, we add a penalty term on it, which is simply the following :

$$J_3(\underline{\boldsymbol{\theta}}) = \text{Pen}(\underline{\boldsymbol{\theta}}), \tag{B.13c}$$

where Pen is any convex penalty.

Steps 1 and 2 are alternated one after another while a stop criterion is reached [Poy+06]. Such criterion is further discussed in Section B.4.3. The algorithm developed in the context of GSA is summarized in Algorithm 11. Some hyperparameters $\lambda_1, \lambda_2^{\mathsf{LS}} > 0$ are also introduced to weigh a functional against another during the corresponding step.

Alternate minimization is a preferred approach when dealing with large-sized optimization problems, particularly in scenarios where model equations exhibit linearity in the parameters.

In the first step, which involves a convex quadratic optimization problem, the highly efficient Nesterov accelerated proximal gradient method proves effective for obtaining solutions. This method is especially suitable for linear dynamical systems that are linear both in the state and the parameters. In cases like bi-convex optimization problems, this approach ensures guaranteed convergence towards a stationary point, further enhancing its appeal in tackling complex optimization challenges.

Algorithm 11: Generalized Smoothing Algorithm GSA-LS.

Input: $(t_i, \underline{\boldsymbol{u}}(t_i))_i$.
$\underline{\underline{\boldsymbol{C}}}^{[0]} \leftarrow$ spline smoothing of data *Step 0*;
while *error criterion* **do**
  $\underline{\underline{\boldsymbol{\theta}}}^{[n+1]} \leftarrow \arg\min_{\underline{\underline{\boldsymbol{\theta}}}} \left(J_2(\underline{\underline{\boldsymbol{C}}}^{[n]}, \underline{\underline{\boldsymbol{\theta}}}) + \lambda_1 J_3(\underline{\underline{\boldsymbol{\theta}}}) \right)$; *Step 1*
  $\underline{\underline{\boldsymbol{C}}}^{[n+1]} \leftarrow \arg\min_{\underline{\underline{\boldsymbol{C}}}} \left(J_1(\underline{\underline{\boldsymbol{C}}}) + \lambda_2^{\mathsf{LS}} J_2(\underline{\underline{\boldsymbol{C}}}, \underline{\underline{\boldsymbol{\theta}}}^{[n+1]}) \right)$; *Step 2*
end
Output: Parameters of the ODE $\underline{\underline{\boldsymbol{\theta}}}^{[N]}$.

B.4.2 GSA with PINN

In the present work, we replace the usage of the least square minimization performed on the spline coefficients with a PINN, as introduced in Section B.2. The splines are now used to generate the trajectories of each species for each experiment, those trajectories are used in the proximal gradient to estimate the parameters $\underline{\underline{\boldsymbol{\theta}}}$. Then during Step 2, the PINN is fine-tuned from the parameters newly estimated, to predict new trajectories of the bacterial populations.

Precisely, the PINN is trained again to minimize the loss function, but using the weight previously estimated at the precedent steps. During Step 1, the parameters updated may not differ that much from the previous ones, so we do not need to train again with as many epochs as the first time. The strategy we adopted is the following : during the first iteration, we train the PINN with 5000 initial epochs. Then in the following, we dynamically determine the number of epochs by considering Err_n the relative error committed on the parameters between the two iterations:

$$\texttt{epoch}_n = \min \left\{ 1 + \left\lfloor 10^3 \exp\left(\frac{1}{2} \log(err_{\underline{\underline{\boldsymbol{\theta}}}}^{[n]}) \right) \right\rfloor, \texttt{nb\_epochs} \right\} \tag{B.14}$$

where `nb_epochs` is a constant parameter, setting a maximal number of epochs at each iteration, which has a default value of 100.

On top of that, we add a final step consisting in using another PINN to predict the value of the parameters $\underline{\underline{\boldsymbol{\theta}}}$ from the trajectories. This step ensures that the parameters returned fit well the trajectories, which are constructed to fit both model and data provided.

B.4.3 Stop criterion

In Algorithms 11 and 12, we iterate steps 1 and 2 while an error criterion is reached. The purpose of this section is to describe this criterion. From one iteration n to the next one $n+1$, we compute the relative error made on the trajectory predicted and the parameter estimated :

$$err^{[n]} = \frac{\|\underline{\boldsymbol{u}}^{[n]} - \underline{\boldsymbol{u}}^{[n+1]}\|_{L^2[0,t_{\max}]}}{\|\underline{\boldsymbol{u}}^{[n]}\|_{L^2[0,t_{\max}]}} + \frac{\|\underline{\underline{\boldsymbol{\theta}}}^{[n]} - \underline{\underline{\boldsymbol{\theta}}}^{[n+1]}\|_{\mathrm{F}}}{\|\underline{\underline{\boldsymbol{\theta}}}^{[n]}\|_{\mathrm{F}}}, \tag{B.15}$$

Algorithm 12: Generalized Smoothing Algorithm GSA-PINN.

Input: $(t_i, \underline{\boldsymbol{u}}(t_i))_i$.
$\underline{\boldsymbol{u}}^{[0]} \leftarrow$ spline smoothing of data;
while *error criterion* **do**
  $\underline{\underline{\boldsymbol{\theta}}}^{[n+1]} \leftarrow \arg\min_{\underline{\underline{\boldsymbol{\theta}}}} \left(J_2(\underline{\boldsymbol{u}}^{[n]}, \underline{\underline{\boldsymbol{\theta}}}) + \lambda_1 J_3(\underline{\underline{\boldsymbol{\theta}}}) \right)$; *Step 1, same as Algorithm 11*
  Fine tune the $\text{PINN}_{\underline{\underline{\boldsymbol{\theta}}}^{[n]}}$ with $\underline{\underline{\boldsymbol{\theta}}}^{[n+1]}$;
  $\underline{\boldsymbol{u}}^{[n+1]} \leftarrow \text{PINN}_{\underline{\underline{\boldsymbol{\theta}}}^{[n+1]}}$ prediction;
end
Output: Parameters of the ODE $\underline{\underline{\boldsymbol{\theta}}}^{[N]}$.

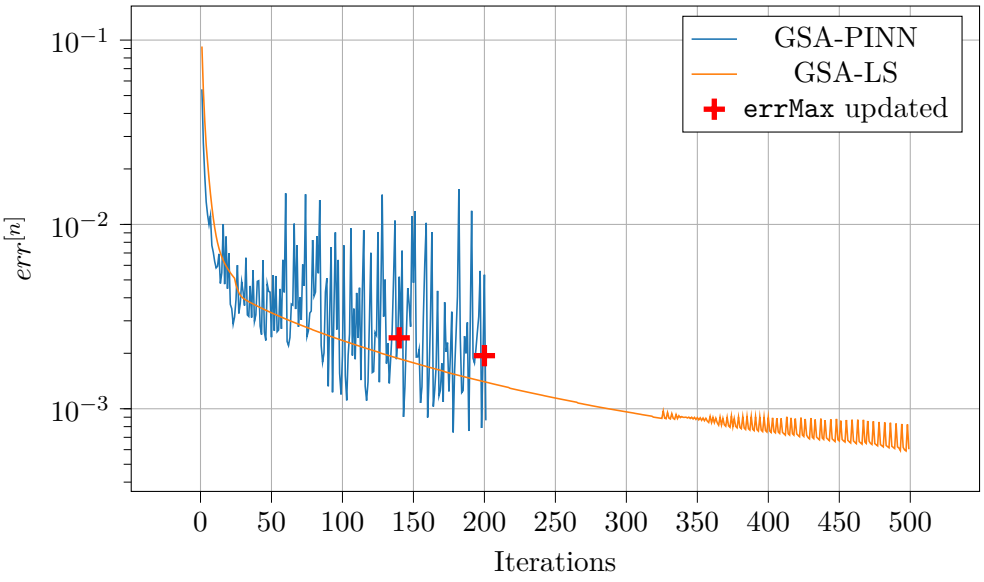


Figure B.12: Evolution of the error in the GSA-PINN algorithm, with test case described in Example B.1.1.

and we stop when $err^{[n]}$ reaches a given tolerance `errMax`.

During the execution of the GSA algorithm, we figure that error tends to stagnate, depending on the test case considered. We present in Figure B.12 the evolution of the error over the iterations, for the test case with 3 populations introduced in Example B.1.1. We figure that a plateau around 10^{-3} is present. The consequence of that is that we will reach the maximal iteration number, without getting any improvement in the relative error. To counter this, we introduce the following procedure : every 20 iterations, we look at the last 20 iterations. If during this lap of time, we have improved the global minimal error reached, then we continue 20 more iterations. Else we multiply the tolerance `errMax` by 10 and go on. In Figure B.12, the iterations where the tolerance has been updated are represented by a red cross. We notice that in this specific case, the algorithm stops after 200 iterations, over the 500 specified as a maximal number of iterations.

On the other hand, with this specific criterion, the algorithm GSA-LS runs until the maximal number is reached because it keeps improving the relative error over the iterations.

In the same spirit, we compute and plot the error committed by the two algorithm compared to the true solution of the benchmark model of example Example B.1.1. The results are shown in Figure B.13. During this run, we disabled the adaptive evolution of `errMax` to see the evolution

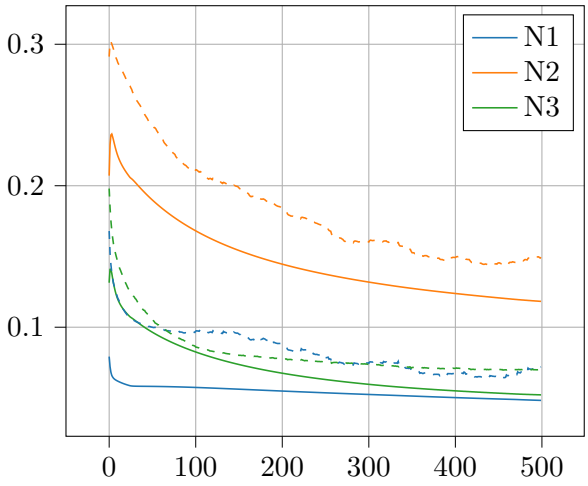


Figure B.13: Evolution of the relative error on the solution computed by the algorithms GSA-LS (plain) and GSA-PINN (dashed), for the example with 3 species of batteries. Each curve is the error on the corresponding species, averaged over the two experiments.

of the error until the maximal number of iterations is reached. The solutions compared are, on the one hand, the PINN-predicted trajectories of the three species, and on the other hand, the solution given by the splines. We figure that the error is decreasing quite slowly, and seem to reach a plateau. Moreover, the results returned by GSA-LS are closer to the exact solution, but the order of magnitude of the errors from both algorithms is the same. This similarity in error magnitudes allows us to effectively assess their respective efficiency in approximating the true solution.

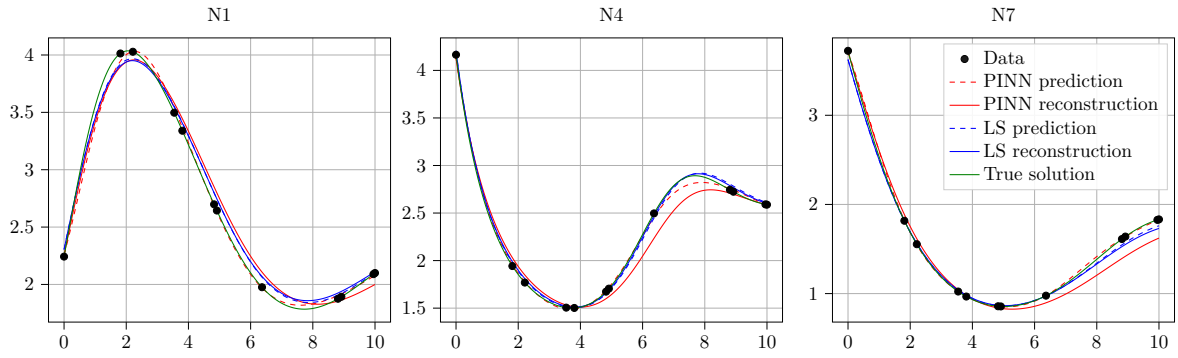
B.4.4 Results and comparisons

In this section, we present the results obtained with the GSA-LS and GSA-PINN algorithms, for test case introduced in Example B.1.2. We use data manually generated from a known set of parameters, and perform the comparison for data with and without noise. The results are presented in Figure B.14.

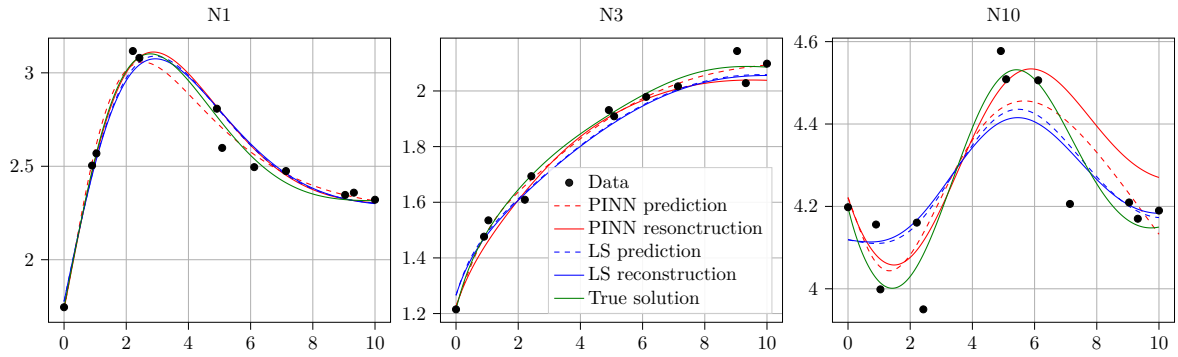
To assess the performance of the two algorithms, we introduce the following error metrics, measuring either the error committed on the estimated parameters or on the predicted trajectories.

- $Err_{\underline{\underline{\boldsymbol{\theta}}},1} := \dfrac{\|\hat{\underline{\underline{\boldsymbol{\theta}}}} - \underline{\underline{\boldsymbol{\theta}}}_{\text{truth}}\|_{\mathsf{F}}}{\|\underline{\underline{\boldsymbol{\theta}}}_{\text{truth}}\|_{\mathsf{F}}}$.
- $Err_{\underline{\underline{\boldsymbol{\theta}}},2}$ defined as the number of coefficients where $\hat{\underline{\underline{\boldsymbol{\theta}}}}$ and $\underline{\underline{\boldsymbol{\theta}}}_{\text{truth}}$ have the same sign, divided by the number of coefficients of the matrices. This definition is motivated by the fact that in biologic context, we are more interested qualitatively in the impact of population on each other, rather than quantitatively.
- $Err_{\underline{\boldsymbol{u}},1} := \dfrac{1}{N_s N_{\text{exp}}} \displaystyle\sum_{i=1}^{N_s} \sum_{e=1}^{N_{\text{exp}}} \dfrac{\|\hat{\underline{\boldsymbol{u}}}_j^{(e)} - \underline{\boldsymbol{u}}_j^{(e)}(\underline{\underline{\boldsymbol{\theta}}}_{\text{truth}})\|_2^2}{\|\underline{\boldsymbol{u}}_j^{(e)}(\underline{\underline{\boldsymbol{\theta}}}_{\text{truth}})\|_2^2}$.
- $Err_{\underline{\boldsymbol{u}},2} := \dfrac{1}{N_s N_{\text{exp}}} \displaystyle\sum_{i=1}^{N_s} \sum_{e=1}^{N_{\text{exp}}} \dfrac{\|\underline{\boldsymbol{u}}_j^{(e)}(\hat{\underline{\underline{\boldsymbol{\theta}}}}) - \underline{\boldsymbol{u}}_j^{(e)}(\underline{\underline{\boldsymbol{\theta}}}_{\text{truth}})\|_2^2}{\|\underline{\boldsymbol{u}}_j^{(e)}(\underline{\underline{\boldsymbol{\theta}}}_{\text{truth}})\|_2^2}$.

In the sequel, we also perform a comparison on the time taken by the two algorithms to run. Especially, the code used in the tests is for the GSA-LS algorithm is a Matlab code provided by



(a) Data without noise.



(b) Noisy data.

Figure B.14: Comparison of the results obtained with the GSA-LS and GSA-PINN algorithms, for the test case with 20 species of bacteria. Only 3 species are presented.

(a) Uniformly distributed data (non-random), 10 data, 10 species, 1 experiment, no noise.

Algo.	Mean $Err_{\underline{\underline{\boldsymbol{\theta}}},1}$	Mean $Err_{\underline{\underline{\boldsymbol{\theta}}},2}$	Mean $Err_{\underline{\boldsymbol{u}},1}$	Mean $Err_{\underline{\boldsymbol{u}},2}$	Elapsed time
GSA-LS	1.02	0.69	$4.73 \cdot 10^{-2}$	$4.94 \cdot 10^{-2}$	5.54 sec
GSA-PINN	1.06	0.6	$\mathbf{1.62 \cdot 10^{-2}}$	$\mathbf{2.2 \cdot 10^{-2}}$	**4.46 s**

(b) Uniformly distributed data (non-random), 10 data, 20 species, 1 experiment, no noise.

Algo.	Mean $Err_{\underline{\underline{\boldsymbol{\theta}}},1}$	Mean $Err_{\underline{\underline{\boldsymbol{\theta}}},2}$	Mean $Err_{\underline{\boldsymbol{u}},1}$	Mean $Err_{\underline{\boldsymbol{u}},2}$	Elapsed time
GSA-LS	1.09	0.8	$3.4 \cdot 10^{-2}$	$3.53 \cdot 10^{-2}$	**8.75 s**
GSA-PINN	0.98	0.74	$\mathbf{1.35 \cdot 10^{-2}}$	$\mathbf{1.92 \cdot 10^{-2}}$	10.48 sec

(c) Uniformly distributed data (non-random), 10 data, 10 species, 10 experiments, no noise.

Algo.	Mean $Err_{\underline{\underline{\boldsymbol{\theta}}},1}$	Mean $Err_{\underline{\underline{\boldsymbol{\theta}}},2}$	Mean $Err_{\underline{\boldsymbol{u}},1}$	Mean $Err_{\underline{\boldsymbol{u}},2}$	Elapsed time
GSA-LS	0.19	0.21	0.32	0.26	**16.81 s**
GSA-PINN	0.22	0.18	$\mathbf{1.5 \cdot 10^{-2}}$	$\mathbf{2.18 \cdot 10^{-2}}$	35.53 sec

(d) Uniformly distributed data (non-random), 10 data, 10 species, 20 experiments, with some noise.

Algo.	Mean $Err_{\underline{\underline{\boldsymbol{\theta}}},1}$	Mean $Err_{\underline{\underline{\boldsymbol{\theta}}},2}$	Mean $Err_{\underline{\boldsymbol{u}},1}$	Mean $Err_{\underline{\boldsymbol{u}},2}$	Elapsed time
GSA-LS	0.16	0.13	0.44	0.28	98.65 sec
GSA-PINN	0.18	0.15	$\mathbf{1.49 \cdot 10^{-2}}$	$\mathbf{2.08 \cdot 10^{-2}}$	**70 s**

Table B.1: Comparison of the results obtained with the GSA-LS and GSA-PINN algorithms, for various test cases.

the authors of [Lar+18], while our code for the GSA-PINN algorithm is written in Python, using the library PyTorch.

The results are gathered in Table B.1, for various configurations of the test case. The first striking results is that both algorithm perform similarly in the estimation of the parameters, as they have the same order on the errors $Err_{\underline{\underline{\boldsymbol{\theta}}},1}$ and $Err_{\underline{\underline{\boldsymbol{\theta}}},2}$. On the other hand, the GSA-PINN algorithm performs better on the prediction of the trajectories, with a lower error $Err_{\underline{\boldsymbol{u}},1}$ and $Err_{\underline{\boldsymbol{u}},2}$. Finally, regarding the time of execution, we find out similar times for both algorithms, but time elapsed by both algorithms depending on the configuration of the data provided: number of training data, species, experiments, and noise.

B.5 Conclusions and perspectives

In conclusion, our utilization of the GLV model within this study has provided valuable insights into the dynamics of microorganism populations. Despite its widespread use, conventional methods may encounter challenges in solving the GLV model due to the intricate behaviors associated with specific parameter sets. By employing PINNs in our research, we successfully simulated the evolution of bacterial species governed by the GLV model. Our proposed approach relies on a loss function that effectively combines the constraints of the physical model with data.

We experimented with various architectures and numerical strategies to enhance the efficiency and accuracy of this methodology. Subsequently, to illustrate the capability of the developed PINN in capturing expected outcomes, we considered two test cases with distinct behaviors. Finally, considering potential experimental applications, we discuss the influence of data, including different sampling, missing data, and noise, on the robustness of this method.

This study on the application of a PINN to the GLV model is a preliminary study to assess the capability of such approach to simulate the evolution of bacterial species. Looking forward, our perspective involves extending this approach into a more complex framework, particularly for the estimation of GLV parameters. Traditionally, algorithms designed for parameter estimation tasks require repeated solving of the model, making computational efficiency a crucial consideration. In this context, the adoption of a fast yet robust method, such as the proposed PINN, holds the potential to be pioneering in the field of parameter estimation for the GLV model. With such methodological developments, we expect to improve the optimization of parameter estimation processes, thus gaining a deeper understanding of microorganism dynamics. However, further investigations, that are out of the scope of this preliminary study, are required to adapt the proposed approach for various parameter values.

Index

Bibliography

Thesis References

[Abd+21] Farah Abdelhafid, Giovanna Guidoboni, Naoki Okumura, Noriko Koizumi, and Sangly P. Srinivas. "Operator Splitting for the Simulation of Aqueous Humor Thermo-Fluid-Dynamics in the Anterior Chamber". In: *Recent Developments in Mathematical, Statistical and Computational Sciences*. Ed. by D. Marc Kilgour, Herb Kunze, Roman Makarov, Roderick Melnik, and Xu Wang. Vol. 343. Series Title: Springer Proceedings in Mathematics & Statistics. Cham: Springer International Publishing, 2021, pp. 489–499. ISBN: 978-3-030-63590-9. DOI: 10.1007/978-3-030-63591-6_45. URL: https://link.springer.com/10.1007/978-3-030-63591-6_45.

[ADL00] P.R. Amestoy, I.S. Duff, and J.-Y. L'Excellent. "Multifrontal parallel distributed symmetric and unsymmetric solvers". In: *Computer Methods in Applied Mechanics and Engineering* 184.2-4 (Apr. 2000), pp. 501–520. ISSN: 00457825. DOI: 10.1016/S0045-7825(99)00242-X. URL: https://linkinghub.elsevier.com/retrieve/pii/S004578259900242X.

[Adl53] Francis Heed Adler. *Physiology of the eye, clinical applications*. St. Louis Mosby, 1953. URL: https://archive.org/details/physiologyofeyece2adle/page/450/mode/2up.

[AF22] Andrea Arnold and Loris Fichera. "Identification of tissue optical properties during thermal laser-tissue interactions: An ensemble Kalman filter-based approach". In: *International Journal for Numerical Methods in Biomedical Engineering* 38.4 (Apr. 2022), e3574. ISSN: 2040-7939, 2040-7947. DOI: 10.1002/cnm.3574. URL: https://onlinelibrary.wiley.com/doi/10.1002/cnm.3574.

[Art+21] Christopher J. Arthurs, Rostislav Khlebnikov, Alex Melville, Marija Marčan, Alberto Gomez, Desmond Dillon-Murphy, Federica Cuomo, Miguel Silva Vieira, Jonas Schollenberger, Sabrina R. Lynch, Christopher Tossas-Betancourt, Kritika Iyer, Sara Hopper, Elizabeth Livingston, Pouya Youssefi, Alia Noorani, Sabrina Ben Ahmed, Foeke J. H. Nauta, Theodorus M. J. Van Bakel, Yunus Ahmed, Petrus A. J. Van Bakel, Jonathan Mynard, Paolo Di Achille, Hamid Gharahi, Kevin D. Lau, Vasilina Filonova, Miquel Aguirre, Nitesh Nama, Nan Xiao, Seungik Baek, Krishna Garikipati, Onkar Sahni, David Nordsletten, and C. Alberto Figueroa. "CRIMSON: An open-source software framework for cardiovascular integrated modelling and simulation". In: *PLOS Computational Biology* 17.5 (May 2021). Ed. by Dina Schneidman-Duhovny, e1008881. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1008881. URL: https://dx.plos.org/10.1371/journal.pcbi.1008881.

[ASB78] B. O. Almroth, P. Stern, and F. A. Brogan. "Automatic choice of global shape functions in structural analysis". In: *AIAA Journal* 16.5 (May 1978), pp. 525–528. ISSN: 0001-1452, 1533-385X. DOI: 10.2514/3.7539. URL: https://arc.aiaa.org/doi/10.2514/3.7539.

[Bal+24] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Steven Benson, Jed Brown, Peter Brune, Kris Buschelman, Emil M. Constantinescu, Lisandro Dalcin, Alp Dener, Victor Eijkhout, Jacob Faibussowitsch, William D. Gropp, Václav Hapla, Tobin Isaac, Pierre Jolivet, Dmitry Karpeev, Dinesh Kaushik, Matthew G. Knepley, Fande Kong, Scott Kruger, Dave A. May, Lois Curfman McInnes, Richard Tran Mills, Lawrence Mitchell, Todd Munson, Jose E. Roman, Karl Rupp, Patrick Sanan, Jason Sarich, Barry F. Smith, Stefano Zampini, Hong Zhang, Hong Zhang, and Junchao Zhang. *PETSc Web page*. 2024. URL: https://petsc.org/.

[Bal96] Etienne Balmès. "PARAMETRIC FAMILIES OF REDUCED FINITE ELEMENT MODELS. THEORY AND APPLICATIONS". In: *Mechanical Systems and Signal Processing* 10.4 (July 1996), pp. 381–394. ISSN: 08883270. DOI: 10.1006/mssp.1996.0027. URL: https://linkinghub.elsevier.com/retrieve/pii/S0888327096900278.

[Bas+24] Nicol Basson, Chao-Hong Surachai Peng, Patrick Geoghegan, Tshilidzi Van Der Lecq, David Steven, Susan Williams, An Eng Lim, and Wei Hua Ho. "A computational fluid dynamics investigation of endothelial cell damage from glaucoma drainage devices". In: *Scientific Reports* 14.1 (Feb. 2024), p. 3777. ISSN: 2045-2322. DOI: 10.1038/s41598-023-50491-9. URL: https://www.nature.com/articles/s41598-023-50491-9.

[Bau+16] Michaël Baudin, Anne Dutfoy, Bertrand Iooss, and Anne-Laure Popelin. "OpenTURNS: An Industrial Software for Uncertainty Quantification in Simulation". In: *Handbook of Uncertainty Quantification*. Ed. by Roger Ghanem, David Higdon, and Houman Owhadi. Cham: Springer International Publishing, 2016, pp. 1–38. ISBN: 978-3-319-11259-6. DOI: 10.1007/978-3-319-11259-6_64-1. URL: https://doi.org/10.1007/978-3-319-11259-6_64-1.

[BBS20] Ajay Bhandari, Ankit Bansal, and Niraj Sinha. "Effect of aging on heat transfer, fluid flow and drug transport in anterior human eye: A computational study". In: *Journal of Controlled Release* 328 (Dec. 2020), pp. 286–303. ISSN: 01683659. DOI: 10.1016/j.jconrel.2020.08.044. URL: https://linkinghub.elsevier.com/retrieve/pii/S0168365920304855.

[BDL17] Antoine Brault, Laurent Dumas, and Didier Lucor. "Uncertainty quantification of inflow boundary condition and proximal arterial stiffness–coupled effect on pulse wave propagation in a vascular network". In: *International Journal for Numerical Methods in Biomedical Engineering* 33.10 (Oct. 2017), e2859. ISSN: 2040-7939, 2040-7947. DOI: 10.1002/cnm.2859. URL: https://onlinelibrary.wiley.com/doi/10.1002/cnm.2859.

[Ber+] Silvia Bertoluzza, Christophe Prud'homme, Thomas Saigre, and Marcela Szopos. "Low to high order finite element resolution for elliptic problems in the presence of a Dirac source term". In preparation.

[Ber+18] Silvia Bertoluzza, Astrid Decoene, Loïc Lacouture, and Sébastien Martin. "Local error estimates of the finite element method for an elliptic problem with a Dirac source term". In: *Numerical Methods for Partial Differential Equations* 34.1 (Jan. 2018), pp. 97–120. ISSN: 0749-159X, 1098-2426. DOI: 10.1002/num.22186. URL: https://onlinelibrary.wiley.com/doi/10.1002/num.22186.

[Bha+02] Archana Bhan, Andrew C. Browning, Sunil Shah, Robin Hamilton, Dinesh Dave, and Harminder S. Dua. "Effect of corneal thickness on intraocular pressure measurements with the pneumotonometer, Goldmann applanation tonometer, and Tono-Pen". In: *Investigative Ophthalmology & Visual Science* 43.5 (May 2002), pp. 1389–1392. ISSN: 0146-0404.

[Bha21] Ajay Bhandari. "Ocular Fluid Mechanics and Drug Delivery: A Review of Mathematical and Computational Models". In: *Pharmaceutical Research* 38.12 (Dec. 2021), pp. 2003–2033. ISSN: 0724-8741, 1573-904X. DOI: 10.1007/s11095-021-03141-6. URL: https://link.springer.com/10.1007/s11095-021-03141-6.

[Bou11] R R A Bourne. "Ethnicity and ocular imaging". In: *Eye* 25.3 (Mar. 2011), pp. 297–300. ISSN: 0950-222X, 1476-5454. DOI: 10.1038/eye.2010.187. URL: https://www.nature.com/articles/eye2010187.

[Bra86] Achi Brandt. "Algebraic multigrid theory: The symmetric case". In: *Applied Mathematics and Computation* 19.1-4 (July 1986), pp. 23–56. ISSN: 00963003. DOI: 10.1016/0096-3003(86)90095-0. URL: https://linkinghub.elsevier.com/retrieve/pii/0096300386900950.

[Buf+12] Annalisa Buffa, Yvon Maday, Anthony T. Patera, Christophe Prud'homme, and Gabriel Turinici. "A priori convergence of the Greedy algorithm for the parametrized reduced basis method". In: *ESAIM: Mathematical Modelling and Numerical Analysis* 46.3 (May 2012), pp. 595–603. ISSN: 0764-583X, 1290-3841. DOI: 10.1051/m2an/2011056. URL: http://www.esaim-m2an.org/10.1051/m2an/2011056.

[Cam+24] Sorina Camarasu-Pop, Gaël Vila, Tristan Glatard, Axel Bonnet, Carole Frindel, and Hélène Ratiney. "Fundamentals on Transparency, Reproducibility and Validation". In: *Trustworthy AI for Medical Imaging*. 2024. URL: https://hal.science/hal-04649249.

[Can+02] C. R. Canning, M. J. Greaney, J. N. Dewynne, and A. D. Fitt. "Fluid flow in the anterior chamber of a human eye". In: *IMA journal of mathematics applied in medicine and biology* 19.1 (Mar. 2002), pp. 31–60. ISSN: 0265-0746.

[CAS22] Open CASCADE. *SALOME: The Open Source Integration Platform for Numerical Simulation.* 2022. URL: https://salome-project.org.

[Cha+24] Vincent Chabannes, Christophe Prud'homme, Thomas Saigre, Sala Lorenzo, Marcela Szopos, and Christophe Trophime. *A 3D geometrical model and meshing procedures for the human eyeball.* Sept. 2024. DOI: 10.5281/ZENODO.13829740. URL: https://zenodo.org/doi/10.5281/zenodo.13829740.

[Che+24] Emily Y. Chew, Stephen A. Burns, Alison G. Abraham, Mathieu F. Bakhoum, Joshua A. Beckman, Toco Y. P. Chui, Robert P. Finger, Alejandro F. Frangi, Rebecca F. Gottesman, Maria B. Grant, Henner Hanssen, Cecilia S. Lee, Michelle L. Meyer, Damiano Rizzoni, Alicja R. Rudnicka, Joel S. Schuman, Sara B. Seidelmann, W. H. Wilson Tang, Bishow B. Adhikari, Narasimhan Danthi, Yuling Hong, Diane Reid, Grace L. Shen, and Young S. Oh. "Standardization and clinical applications of retinal imaging biomarkers for cardiovascular disease: a Roadmap from an NHLBI workshop". In: *Nature Reviews Cardiology* (July 2024). ISSN: 1759-5002, 1759-5010. DOI: 10.1038/s41569-024-01060-8. URL: https://www.nature.com/articles/s41569-024-01060-8 (visited on 10/16/2024).

[CLC11] Francisco Chinesta, Pierre Ladeveze, and Elías Cueto. "A Short Review on Model Order Reduction Based on Proper Generalized Decomposition". In: *Archives of Computational Methods in Engineering* 18.4 (Nov. 2011), pp. 395–404. ISSN: 1134-3060, 1886-1784. DOI: 10.1007/s11831-011-9064-7. URL: http://link.springer.com/10.1007/s11831-011-9064-7.

[CM09a] Rachida Chakir and Yvon Maday. "A two-grid finite-element/reduced basis scheme for the approximation of the solution of parameter dependent PDE". In: *9e Colloque national en calcul des structures.* Giens, France: CSMA, May 2009. URL: https://hal.science/hal-01420726.

[CM09b] Rachida Chakir and Yvon Maday. "Une méthode combinée d'éléments finis à deux grilles/bases réduites pour l'approximation des solutions d'une E.D.P. paramétrique". In: *Comptes Rendus. Mathématique* 347.7-8 (2009). Publisher: Elsevier, pp. 435–440. DOI: 10.1016/j.crma.2009.02.019.

[CMP19] R. Chakir, Y. Maday, and P. Parnaudeau. "A non-intrusive reduced basis approach for parametrized heat transfer problems". In: *Journal of Computational Physics* 376 (Jan. 2019), pp. 617–633. ISSN: 00219991. DOI: 10.1016/j.jcp.2018.10.001. URL: https://linkinghub.elsevier.com/retrieve/pii/S0021999118306570.

[Coh+20] Albert Cohen, Wolfgang Dahmen, Ronald DeVore, and James Nichols. "Reduced Basis Greedy Selection Using Random Training Sets". In: *ESAIM: Mathematical Modelling and Numerical Analysis* 54.5 (Sept. 2020), pp. 1509–1524. ISSN: 0764-583X, 1290-3841. DOI: 10.1051/m2an/2020004. URL: https://www.esaim-m2an.org/10.1051/m2an/2020004 (visited on 08/20/2024).

[CQR13] Peng Chen, Alfio Quarteroni, and Gianluigi Rozza. "Simulation-based uncertainty quantification of human arterial network hemodynamics". In: *International Journal for Numerical Methods in Biomedical Engineering* 29.6 (June 2013), pp. 698–721. ISSN: 2040-7939, 2040-7947. DOI: 10.1002/cnm.2554. URL: https://onlinelibrary.wiley.com/doi/10.1002/cnm.2554.

[DDS23] Claire Dupont, Florian De Vuyst, and Anne-Virginie Salsac. "Data-driven kinematics-consistent model-order reduction of fluid–structure interaction problems: application to deformable microcapsules in a Stokes flow". In: *Journal of Fluid Mechanics* 955 (Jan. 2023), A2. ISSN: 0022-1120, 1469-7645. DOI: 10.1017/jfm.2022.1005. URL: https://www.cambridge.org/core/product/identifier/S0022112022010059/type/journal_article.

[Dod+14] H. Dodig, S. Lalléchère, P. Bonnet, D. Poljak, and K. El Khamlichi Drissi. "Stochastic sensitivity of the electromagnetic distributions inside a human eye modeled with a 3D hybrid BEM/FEM edge element method". In: *Engineering Analysis with Boundary Elements* 49 (Dec. 2014), pp. 48–62. ISSN: 09557997. DOI: 10.1016/j.enganabound.2014.04.005. URL: https://linkinghub.elsevier.com/retrieve/pii/S0955799714000708.

[DR04] P. G. Drazin and W. H. Reid. *Hydrodynamic Stability*. 2nd ed. Cambridge University Press, Aug. 2004. ISBN: 978-0-521-52541-1 978-0-511-61693-8. DOI: 10.1017/CBO9780511616938. URL: https://www.cambridge.org/core/product/identifier/9780511616938/type/book.

[Dvo+19] Mariia Dvoriashyna, Jan O. Pralits, Jennifer H. Tweedy, and Rodolfo Repetto. "Mathematical Models of Aqueous Production, Flow and Drainage". In: *Ocular Fluid Dynamics*. Ed. by Giovanna Guidoboni, Alon Harris, and Riccardo Sacco. Series Title: Modeling and Simulation in Science, Engineering and Technology. Cham: Springer International Publishing, 2019, pp. 227–263. ISBN: 978-3-030-25886-3. DOI: 10.1007/978-3-030-25886-3_9. URL: http://link.springer.com/10.1007/978-3-030-25886-3_9.

[Dvo+22] Mariia Dvoriashyna, Alexander J. E. Foss, Eamonn A. Gaffney, and Rodolfo Repetto. "A Mathematical Model of Aqueous Humor Production and Composition". In: *Investigative Opthalmology & Visual Science* 63.9 (Aug. 2022), p. 1. ISSN: 1552-5783. DOI: 10.1167/iovs.63.9.1. URL: https://iovs.arvojournals.org/article.aspx?articleid=2783533.

[EG21] Alexandre Ern and Jean-Luc Guermond. *Finite Elements II: Galerkin Approximation, Elliptic and Mixed PDEs*. Vol. 73. Texts in Applied Mathematics. Cham: Springer International Publishing, 2021. ISBN: 978-3-030-56922-8. DOI: 10.1007/978-3-030-56923-5. URL: https://link.springer.com/10.1007/978-3-030-56923-5.

[EL] Engineers Edge and Engineers Edge LLC. *Convective Heat Transfer Coefficients Table Chart*. URL: https://www.engineersedge.com/heat_transfer/convective_heat_transfer_coefficients__13378.htm.

[Eme+75] A. F. Emery, P. Kramar, A. W. Guy, and J. C. Lin. "Microwave Induced Temperature Rises in Rabbit Eyes in Cataract Research". In: *Journal of Heat Transfer* 97.1 (Feb. 1975), pp. 123–128. ISSN: 0022-1481, 1528-8943. DOI: 10.1115/1.3450259. URL: https://asmedigitalcollection.asme.org/heattransfer/article/97/1/123/431306/Microwave-Induced-Temperature-Rises-in-Rabbit-Eyes.

[ESW14] Howard C. Elman, David J. Silvester, and Andrew J. Wathen. *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*. 2. ed. Numerical mathematics and scientific computation. Oxford: Oxford Univ. Press, 2014. ISBN: 978-0-19-967879-2.

[Eul75] Leonhard Euler. "Principles for Determining the Motion of the Blood through Arteries". In: *Opera Omnia* 2.23 (1775). Publisher: Societatis Scientiarum Naturalium, pp. 810–815.

[EYB89] N. Efron, G. Young, and N. A. Brennan. "Ocular surface temperature". In: *Current Eye Research* 8.9 (Sept. 1989), pp. 901–906. ISSN: 0271-3683.

[Fer+18] José Ignacio Fernández-Vigo, Alfonso C. Marcos, Rafael Agujetas, José María Montanero, Inés Sánchez-Guillén, Julián García-Feijóo, Adrián Pandal-Blanco, José Ángel Fernández-Vigo, and Ana Macarro-Merino. "Computational simulation of aqueous humour dynamics in the presence of a posterior-chamber versus iris-fixed phakic intraocular lens". In: *PLOS ONE* 13.8 (Aug. 2018). Ed. by Paloma B. Liton, e0202128. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0202128. URL: https://dx.plos.org/10.1371/journal.pone.0202128.

[FG06] A. D. Fitt and G. Gonzalez. "Fluid Mechanics of the Human Eye: Aqueous Humour Flow in The Anterior Chamber". In: *Bulletin of Mathematical Biology* 68.1 (Jan. 2006), pp. 53–71. ISSN: 0092-8240, 1522-9602. DOI: 10.1007/s11538-005-9015-2. URL: http://link.springer.com/10.1007/s11538-005-9015-2 (visited on 12/10/2024).

[FGF08] Pascal Jean Frey, Paul L. George, and Pascal Jean Frey. *Mesh generation: application to finite elements*. 2. ed. London: ISTE [u.a.], 2008. ISBN: 978-1-84821-029-5.

[Fly+23] David Flynn, Mark Palmer, Randall Schiestl, Steven Levine, and Thomas Maeder. *Landscape Report & Industry Survey on the Use of Computational Modeling & Simulation in Medical Device Development*. Tech. rep. Medical Device Innovation Consortium (MDIC), 2023. URL: https://mdic.org/resource/cmslandscapereport/.

[FR83] J. P. Fink and W. C. Rheinboldt. "On the Error Behavior of the Reduced Basis Technique for Nonlinear Finite Element Approximations". In: *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik* 63.1 (Jan. 1983), pp. 21–28. ISSN: 0044-2267, 1521-4001. DOI: 10.1002/zamm.19830630105. URL: https://onlinelibrary.wiley.com/doi/10.1002/zamm.19830630105.

[GHS19] G. Guidoboni, A. Harris, and R. Sacco. *Ocular Fluid Dynamics: Anatomy, Physiology, Imaging Techniques, and Mathematical Modeling*. Modeling and Simulation in Science, Engineering and Technology. Springer International Publishing, 2019. ISBN: 978-3-030-25886-3. URL: https://books.google.fr/books?id=Mz_ADwAAQBAJ.

[GM23a] Elise Grosjean and Yvon Maday. "A doubly reduced approximation for the solution to PDE's based on a domain truncation and a reduced basis method: Application to Navier-Stokes equations". Mar. 2023. URL: https://hal.science/hal-03588508.

[GM23b] Elise Grosjean and Yvon Maday. "Error estimate of the Non-Intrusive Reduced Basis (NIRB) two-grid method with parabolic equations". In: *The SMAI Journal of computational mathematics* 9 (Dec. 2023), pp. 227–256. ISSN: 2426-8399. DOI: 10.5802/smai-jcm.100. URL: https://smai-jcm.centre-mersenne.org/articles/10.5802/smai-jcm.100/.

[GR09] Christophe Geuzaine and Jean-François Remacle. "Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities". In: *International Journal for Numerical Methods in Engineering* 79.11 (Sept. 2009), pp. 1309–1331. ISSN: 0029-5981, 1097-0207. DOI: 10.1002/nme.2579. URL: https://onlinelibrary.wiley.com/doi/10.1002/nme.2579.

[Gro22] Elise Grosjean. "Variations and further developments on the Non-Intrusive Reduced Basis two-grid method". Issue: 2022SORUS019. Theses. Sorbonne Université, Mar. 2022. URL: https://tel.archives-ouvertes.fr/tel-03715386.

[Gui+20] Giovanna Guidoboni, Riccardo Sacco, Marcela Szopos, Lorenzo Sala, Alice Chandra Verticchio Vercellin, Brent Siesky, and Alon Harris. "Neurodegenerative Disorders of the Eye and of the Brain: A Perspective on Their Fluid-Dynamical Connections and the Potential of Mechanism-Driven Modeling". In: *Frontiers in Neuroscience* 14 (Nov. 2020), p. 566428. ISSN: 1662-453X. DOI: 10.3389/fnins.2020.566428. URL: https://www.frontiersin.org/articles/10.3389/fnins.2020.566428/full.

[Gui+22] G Guidoboni, R Nunez, J Keller, C Wikle, El Robinson, A Verticchio, B Siesky, F Oddone, L Quaranta, B Wirostko, F Topouzis, C-Y Cheng, I Januleviciene, A Wegner, G Antman, C Jones, and A Harris. "Precision medicine and glaucoma management: how mathematical modeling and artificial intelligence help in clinical practice". In: *Expert Review of Ophthalmology* 17.5 (Sept. 2022), pp. 299–301. ISSN: 1746-9899, 1746-9902. DOI: 10.1080/17469899.2022.2130249. URL: https://www.tandfonline.com/doi/full/10.1080/17469899.2022.2130249.

[Hal+33] Stephen Hales, William Innys, Richard Manby, and Thomas Woodward. *Statical essays, containing haemastaticks, or, An account of some hydraulick and hydrostatical experiments made on the blood and blood vessels of animals : also an account of some experiments on stones in the kidneys and bladder : with an enquiry into the nature of those anomalous concretions : to which is added, an appendix, containing observations and experiments relating to several subjects in the first volume, the greater part of which were read at several meetings before the Royal Society : with an index to both volumes Vol. II*. Vol. v.2 (1733). London, Printed for W. Innys, R. Manby, at the west-end of St. Paul's, and T. Woodward, at the Half-Moon between Temple-Gate, Fleetstreet, 1733, 1733. URL: https://www.biodiversitylibrary.org/item/188045.

[Har+23] Alica Hartmann, Stephanie D. Grabitz, Esther M. Hoffmann, Philipp S. Wild, Irene Schmidtmann, Karl J. Lackner, Manfred E. Beutel, Thomas Münzel, Oliver Tüscher, Jörn M. Schattenberg, Norbert Pfeiffer, and Alexander K. Schuster. "Intraocular Pressure and Its Relation to Climate Parameters—Results From the Gutenberg Health Study". In: *Investigative Opthalmology & Visual Science* 64.7 (June 2023), p. 15. ISSN: 1552-5783. DOI: 10.1167/iovs.64.7.15. URL: https://iovs.arvojournals.org/article.aspx?articleid=2785685.

[HB02] Jeffrey J. Heys and Victor H. Barocas. "A boussinesq model of natural convection in the human eye and the formation of Krukenberg's spindle". In: *Annals of Biomedical Engineering* 30.3 (Mar. 2002), pp. 392–401. ISSN: 0090-6964. DOI: 10.1114/1.1477447.

[Hey+16] Steven B Heymsfield, Cristina Gonzalez M, Diana Thomas, Kori Murray, and Guang Jia. "Adult Human Ocular Volume: Scaling to Body Size and Composition". In: *Anatomy & Physiology* 6.5 (2016). ISSN: 21610940. DOI: 10.4172/2161-0940.1000239. URL: https://www.omicsonline.org/open-access/adult-human-ocular-volume-scaling-to-body-size-and-composition-2161-0940-1000239.php?aid=78889.

[Hir+07] A Hirata, S Watanabe, O Fujiwara, M Kojima, K Sasaki, and T Shiozawa. "Temperature elevation in the eye of anatomically based human head models for plane-wave exposures". In: *Physics in Medicine and Biology* 52.21 (Oct. 2007). Publisher: IOP Publishing, pp. 6389–6399. ISSN: 1361-6560. DOI: 10.1088/0031-9155/52/21/003. URL: http://dx.doi.org/10.1088/0031-9155/52/21/003.

[HRE23] Rémi J Hernandez, Paul A Roberts, and Wahbi K El-Bouri. "Advancing treatment of retinal disease through in silico trials". In: *Progress in Biomedical Engineering* 5.2 (Apr. 2023), p. 022002. ISSN: 2516-1091. DOI: 10.1088/2516-1091/acc8a9. URL: https://iopscience.iop.org/article/10.1088/2516-1091/acc8a9.

[HRS16] Jan S Hesthaven, Gianluigi Rozza, and Benjamin Stamm. *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*. SpringerBriefs in Mathematics. Cham: Springer International Publishing, 2016. ISBN: 978-3-319-22470-1. DOI: 10.1007/978-3-319-22470-1. URL: https://link.springer.com/10.1007/978-3-319-22470-1 (visited on 09/27/2024).

[ITI24] IT'IS Foundation. *Thermal Conductivity*. 2024. URL: https://itis.swiss/virtual-population/tissue-properties/database/thermal-conductivity/.

[J J82] J J W Lagendijk. "A mathematical model to calculate temperature distributions in human and rabbit eyes during hyperthermic treatment". In: *Physics in Medicine & Biology* 27.11 (Nov. 1982), pp. 1301–1311. ISSN: 0031-9155, 1361-6560. DOI: 10.1088/0031-9155/27/11/001. URL: https://iopscience.iop.org/article/10.1088/0031-9155/27/11/001.

[Kel+24] Rebecca L. Kellner, Alon Harris, Lauren Ciulla, Giovanna Guidoboni, Alice Verticchio Vercellin, Francesco Oddone, Carmela Carnevale, Mohamed Zaid, Gal Antman, Jeffrey T. Kuvin, and Brent Siesky. "The Eye as the Window to the Heart: Optical Coherence Tomography Angiography Biomarkers as Indicators of Cardiovascular Disease". In: *Journal of Clinical Medicine* 13.3 (Jan. 2024), p. 829. ISSN: 2077-0383. DOI: 10.3390/jcm13030829. URL: https://www.mdpi.com/2077-0383/13/3/829.

[Ker+05] Gaetan Kerschen, Jean-claude Golinval, Alexander F. Vakakis, and Lawrence A. Bergman. "The Method of Proper Orthogonal Decomposition for Dynamical Characterization and Order Reduction of Mechanical Systems: An Overview". In: *Nonlinear Dynamics* 41.1-3 (Aug. 2005), pp. 147–169. ISSN: 0924-090X, 1573-269X. DOI: 10.1007/s11071-005-2803-2. URL: http://link.springer.com/10.1007/s11071-005-2803-2.

[Kho+03] Boo Cheong Khoo, Jacob White, Jaime Peraire, and Anthony T. Patera. *Numerical Methods for Partial Differential Equations (SMA 5212), Spring 2003*. 2003. URL: https://ocw.mit.edu/courses/16-920j-numerical-methods-for-partial-differential-equations-sma-5212-spring-2003/.

[Kin+18] Shigeru Kinoshita, Noriko Koizumi, Morio Ueno, Naoki Okumura, Kojiro Imai, Hiroshi Tanaka, Yuji Yamamoto, Takahiro Nakamura, Tsutomu Inatomi, John Bush, Munetoyo Toda, Michio Hagiya, Isao Yokota, Satoshi Teramukai, Chie Sotozono, and Junji Hamuro. "Injection of Cultured Cells with a ROCK Inhibitor for Bullous Keratopathy". In: *New England Journal of Medicine* 378.11 (Mar. 2018), pp. 995–1003. ISSN: 0028-4793, 1533-4406. DOI: 10.1056/NEJMoa1712770. URL: http://www.nejm.org/doi/10.1056/NEJMoa1712770.

[Kla98] Caroline C. W. Klaver. "Age-Specific Prevalence and Causes of Blindness and Visual Impairment in an Older Population: The Rotterdam Study". In: *Archives of Ophthalmology* 116.5 (May 1998), p. 653. ISSN: 0003-9950. DOI: 10.1001/archopht.116.5.653. URL: http://archopht.jamanetwork.com/article.aspx?doi=10.1001/archopht.116.5.653.

[Kos+13] Philip Kosky, Robert Balmer, William Keat, and George Wise. "Mechanical Engineering". In: *Exploring Engineering*. Elsevier, 2013, pp. 259–281. ISBN: 978-0-12-415891-7. DOI: 10.1016/B978-0-12-415891-7.00012-1. URL: https://linkinghub.elsevier.com/retrieve/pii/B9780124158917000121.

[Kou16] Aristotle G. Koutsiaris. "Wall shear stress in the human eye microcirculation in vivo, segmental heterogeneity and performance of in vitro cerebrovascular models". In: *Clinical Hemorheology and Microcirculation* 63.1 (July 2016), pp. 15–33. ISSN: 13860291, 18758622. DOI: 10.3233/CH-151976. URL: https://www.medra.org/servlet/aliasResolver?alias=iospress&doi=10.3233/CH-151976.

[KS10] Andreas Karampatzakis and Theodoros Samaras. "Numerical model of heat transfer in the human eye with consideration of fluid dynamics of the aqueous humour". In: *Physics in Medicine and Biology* 55.19 (Oct. 2010), pp. 5653–5665. ISSN: 0031-9155, 1361-6560. DOI: 10.1088/0031-9155/55/19/003. URL: https://iopscience.iop.org/article/10.1088/0031-9155/55/19/003.

[Kud+20] Bachar Kudsieh, Jose Ignacio Fernández-Vigo, Rafael Agujetas, Jose María Montanero, Jose María Ruiz-Moreno, Jose Ángel Fernández-Vigo, and Julián García-Feijóo. "Numerical model to predict and compare the hypotensive efficacy and safety of minimally invasive glaucoma surgery devices". In: *PLOS ONE* 15.9 (Sept. 2020). Ed. by Ahmed Awadein, e0239324. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0239324. URL: https://dx.plos.org/10.1371/journal.pone.0239324.

[Kum+06] Satish Kumar, Sumanta Acharya, Roger Beuerman, and Arto Palkama. "Numerical Solution of Ocular Fluid Dynamics in a Rabbit Eye: Parametric Effects". In: *Annals of Biomedical Engineering* 34.3 (Mar. 2006), pp. 530–544. ISSN: 0090-6964, 1573-9686. DOI: 10.1007/s10439-005-9048-6. URL: http://link.springer.com/10.1007/s10439-005-9048-6.

[KW14] T. Köppl and B. Wohlmuth. "Optimal A Priori Error Estimates for an Elliptic Problem with Dirac Right-Hand Side". In: *SIAM Journal on Numerical Analysis* 52.4 (Jan. 2014), pp. 1753–1769. ISSN: 0036-1429, 1095-7170. DOI: 10.1137/130927619. URL: http://epubs.siam.org/doi/10.1137/130927619.

[Kwo+09] Young H. Kwon, John H. Fingert, Markus H. Kuehn, and Wallace L.M. Alward. "Primary Open-Angle Glaucoma". In: *New England Journal of Medicine* 360.11 (Mar. 2009), pp. 1113–1124. ISSN: 0028-4793, 1533-4406. DOI: 10.1056/NEJMra0804630. URL: http://www.nejm.org/doi/abs/10.1056/NEJMra0804630.

[Lac16] Loïc Lacouture. "Modelling and simulation of the movement of thin structures in a viscous fluid : application to the muco-ciliary transport". PhD thesis. Université Paris-Saclay, 2016. URL: https://theses.hal.science/tel-01366885v1.

[Lem87] Jean Lemaire. *The All-Seeing Eye: A Subtle Twist*. 1987. URL: https://youtu.be/dQw4w9WgXcQ.

[Li+10] Eric Li, G. R. Liu, Vincent Tan, and Z. C. He. "Modeling and simulation of bioheat transfer in the human eye using the 3D alpha finite element method (αFEM)". In: *International Journal for Numerical Methods in Biomedical Engineering* 26.8 (2010), pp. 955–976. DOI: https://doi.org/10.1002/cnm.1372. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/cnm.1372.

[LM54] P. D. Lax and A. N. Milgram. “Parabolic equations”. In: *Contributions to the theory of partial differential equations*. Annals of Mathematics Studies, no. 33. Princeton University Press, Princeton, N. J., 1954, pp. 167–190.

[Map68] R. Mapstone. “Measurement of corneal temperature”. In: *Experimental Eye Research* 7.2 (Apr. 1968), 237–IN29. ISSN: 00144835. DOI: 10.1016/S0014-4835(68)80073-9. URL: https://linkinghub.elsevier.com/retrieve/pii/S0014483568800739.

[Mar+22] Kent-André Mardal, Marie E. Rognes, Travis B. Thompson, and Lars Magnus Valnes. *Mathematical Modeling of the Human Brain: From Magnetic Resonance Images to Finite Element Simulation*. Vol. 10. Simula SpringerBriefs on Computing. Cham: Springer International Publishing, 2022. ISBN: 978-3-030-95136-8. DOI: 10.1007/978-3-030-95136-8. URL: https://link.springer.com/10.1007/978-3-030-95136-8.

[Mas04] Barry R. Masters. “Franz Frankhauser and Sylwia Kwasniewska (eds): Lasers in Ophthalmology—Basic, Diagnostic and Surgical Aspects: A Review: Kugler Publications, The Hague, The Netherlands, 2003, pp. 450 (ISBN 90-6299-189-0)”. In: *Graefe’s Archive for Clinical and Experimental Ophthalmology* 242.5 (May 2004), pp. 446–447. ISSN: 0721-832X, 1435-702X. DOI: 10.1007/s00417-004-0873-3. URL: http://link.springer.com/10.1007/s00417-004-0873-3.

[Mau13] Bertrand Maury. *The Respiratory System in Equations*. Milano: Springer Milan, 2013. ISBN: 978-88-470-5214-7. DOI: 10.1007/978-88-470-5214-7. URL: http://link.springer.com/10.1007/978-88-470-5214-7.

[Mil93] Zdeněk Milka. “Finite element solution of a stationary heat conduction equation with the radiation boundary condition”. In: *Applications of Mathematics* 38.1 (1993). Publisher: Institute of Mathematics, Academy of Sciences of the Czech Republic, pp. 67–79. URL: http://eudml.org/doc/15737.

[MMG22] MMG tools. *MMG: a mesh-based finite element library*. 2022. URL: https://www.mmgtools.org/.

[Mos58] Robert A. Moses. “The Goldmann Applanation Tonometer*”. In: *American Journal of Ophthalmology* 46.6 (Dec. 1958), pp. 865–869. ISSN: 00029394. DOI: 10.1016/0002-9394(58)90998-X. URL: https://linkinghub.elsevier.com/retrieve/pii/000293945890998X.

[MPT02] Yvon Maday, Anthony T. Patera, and G. Turinici. “Global a priori convergence theory for reduced-basis approximations of single-parameter symmetric coercive elliptic partial differential equations”. In: *Comptes Rendus Mathematique* 335.3 (Jan. 2002), pp. 289–294. ISSN: 1631073X. DOI: 10.1016/S1631-073X(02)02466-4. URL: https://linkinghub.elsevier.com/retrieve/pii/S1631073X02024664.

[MS18] Stefano Marelli and Bruno Sudret. “An active-learning algorithm that combines sparse polynomial chaos expansions and bootstrap for structural reliability analysis”. In: *Structural Safety* 75 (Nov. 2018), pp. 67–74. ISSN: 01674730. DOI: 10.1016/j.strusafe.2018.06.003. URL: https://linkinghub.elsevier.com/retrieve/pii/S0167473017302977.

[Mur+23] Javier Murgoitio-Esandi, Benjamin Y. Xu, Brian J. Song, Qifa Zhou, and Assad A. Oberai. “A Mechanistic Model of Aqueous Humor Flow to Study Effects of Angle Closure on Intraocular Pressure”. In: *Translational Vision Science & Technology* 12.1 (Jan. 2023), p. 16. ISSN: 2164-2591. DOI: 10.1167/tvst.12.1.16. URL: https://doi.org/10.1167/tvst.12.1.16.

[Nar17] Arunn Narasimhan. “Heat and Mass Transfer Processes in the Eye”. In: *Handbook of Thermal Science and Engineering*. Ed. by Francis A. Kulacki. Cham: Springer International Publishing, 2017, pp. 1–35. ISBN: 978-3-319-32003-8. DOI: 10.1007/978-3-319-32003-8_72-2. URL: http://link.springer.com/10.1007/978-3-319-32003-8_72-2.

[Ngu20] Van Kien Nguyen. “Greedy algorithms and Kolmogorov widths in Banach spaces”. In: *Journal of Approximation Theory* 251 (Mar. 2020), p. 105344. ISSN: 00219045. DOI: 10.1016/j.jat.2019.105344. URL: https://linkinghub.elsevier.com/retrieve/pii/S0021904519301510.

[NO06] E.Y.K. Ng and E.H. Ooi. "FEM simulation of the eye structure with bioheat analysis". In: *Computer Methods and Programs in Biomedicine* 82.3 (June 2006), pp. 268–276. ISSN: 01692607. DOI: 10.1016/j.cmpb.2006.04.001. URL: https://linkinghub.elsevier.com/retrieve/pii/S0169260706000708.

[NO07] E.Y.K. Ng and E.H. Ooi. "Ocular surface temperature: A 3D FEM prediction using bioheat equation". In: *Computers in Biology and Medicine* 37.6 (June 2007), pp. 829–835. ISSN: 00104825. DOI: 10.1016/j.compbiomed.2006.08.023. URL: https://linkinghub.elsevier.com/retrieve/pii/S0010482506001557.

[NOR08] Eddie-Yin-Kwee Ng, Ean-Hin Ooi, and U. Rajendra Archarya. "A comparative study between the two-dimensional and three-dimensional human eye models". In: *Mathematical and Computer Modelling* 48.5-6 (Sept. 2008), pp. 712–720. ISSN: 08957177. DOI: 10.1016/j.mcm.2007.11.011. URL: https://linkinghub.elsevier.com/retrieve/pii/S0895717707003548 (visited on 09/25/2024).

[NP80] Ahmed K. Noor and Jeanne M. Peters. "Reduced Basis Technique for Nonlinear Analysis of Structures". In: *AIAA Journal* 18.4 (Apr. 1980), pp. 455–462. ISSN: 0001-1452, 1533-385X. DOI: 10.2514/3.50778. URL: https://arc.aiaa.org/doi/10.2514/3.50778.

[NS74] Joachim A. Nitsche and Alfred H. Schatz. "Interior estimates for Ritz-Galerkin methods". In: *Mathematics of Computation* 28.128 (1974), pp. 937–958. ISSN: 0025-5718, 1088-6842. DOI: 10.1090/S0025-5718-1974-0373325-9. URL: https://www.ams.org/mcom/1974-28-128/S0025-5718-1974-0373325-9/.

[Num24] NumPEx Consortium. *EXAMa: Methods and Algorithms for Exascale*. 2024. URL: https://numpex.org/exama-methods-and-algorithms-for-exascale/.

[NVB63] Abraham Noordergraaf, Pieter D. Verdouw, and Herman B.K. Boom. "The use of an analog computer in a circulation model". In: *Progress in Cardiovascular Diseases* 5.5 (Mar. 1963), pp. 419–439. ISSN: 00330620. DOI: 10.1016/S0033-0620(63)80009-2. URL: https://linkinghub.elsevier.com/retrieve/pii/S0033062063800092.

[Oje24] Ignacio Ojea. "Anisotropic regularity for elliptic problems with Dirac measures as data". In: *Journal of Mathematical Analysis and Applications* 535.1 (July 2024), p. 128104. ISSN: 0022247X. DOI: 10.1016/j.jmaa.2024.128104. URL: https://linkinghub.elsevier.com/retrieve/pii/S0022247X24000258 (visited on 10/11/2024).

[ON08] Ean-Hin Ooi and Eddie Yin-Kwee Ng. "Simulation of aqueous humor hydrodynamics in human eye heat transfer". In: *Computers in Biology and Medicine* 38.2 (Feb. 2008), pp. 252–262. ISSN: 00104825. DOI: 10.1016/j.compbiomed.2007.10.007. URL: https://linkinghub.elsevier.com/retrieve/pii/S001048250700176X.

[ON09] E. H. Ooi and E. Y. K. Ng. "Ocular temperature distribution: a mathematical perspective". In: *Journal of Mechanics in Medicine and Biology* 09.02 (June 2009), pp. 199–227. ISSN: 0219-5194, 1793-6810. DOI: 10.1142/S0219519409002936. URL: https://www.worldscientific.com/doi/abs/10.1142/S0219519409002936.

[Ope] OpenTURNS. *Compute Sobol' indices confidence intervals — OpenTURNS 1.23 documentation*. URL: https://openturns.github.io/openturns/latest/auto_meta_modeling/polynomial_chaos_metamodel/plot_chaos_sobol_confidence.html.

[Pet89] Janet S. Peterson. "The Reduced Basis Method for Incompressible Viscous Flow Calculations". In: *SIAM Journal on Scientific and Statistical Computing* 10.4 (July 1989), pp. 777–786. ISSN: 0196-5204, 2168-3417. DOI: 10.1137/0910047. URL: http://epubs.siam.org/doi/10.1137/0910047.

[Pru+02] C. Prud'homme, D. V. Rovas, K. Veroy, L. Machiels, Y. Maday, A. T. Patera, and G. Turinici. "Reliable Real-Time Solution of Parametrized Partial Differential Equations: Reduced-Basis Output Bound Methods". In: *Journal of Fluids Engineering* 124.1 (Mar. 2002), pp. 70–80. ISSN: 0098-2202, 1528-901X. DOI: 10.1115/1.1448332. URL: https://asmedigitalcollection.asme.org/fluidsengineering/article/124/1/70/462808/Reliable-RealTime-Solution-of-Parametrized-Partial.

[Pru+12] Christophe Prud'homme, Vincent Chabannes, Vincent Doyeux, Mourad Ismail, Abdoulaye Samake, and Goncalo Pena. "Feel++ : A computational framework for Galerkin Methods and Advanced Numerical Methods". In: *ESAIM: Proceedings* 38 (Dec. 2012). Ed. by F. Coquel, M. Gutnic, P. Helluy, F. Lagoutière, C. Rohde, and N. Seguin, pp. 429–455. ISSN: 1270-900X. DOI: 10.1051/proc/201238024. URL: http://www.esaim-proc.org/10.1051/proc/201238024.

[Pru+24a] Christophe Prud'homme, Pierre Alliez, Vincent Chabannes, Rudy Chocat, Emmanuel Franck, Vincent Fraucher, Floriant Faucher, Clément Gauchy, Christos Georgiadis, Luc Giraud, Frédéric Hecht, Guillaume Helbecque, Pierre Jolivet, Olivier Jamond, Pierre Ledac, Nouredine Melab, Victor Michel-Dansac, Frédéric Nataf, Lucas Palazzolo, Yannick Privat, Thomas Saigre-Tardif, El-Ghazali Talbi, Pierre Henri Tournier, Christophe Trophime, Céline Van Landeghem, and Raphael Zanella. *Benchmarking analysis report*. Deliverable Report D7.1. tex.version: v1.1.0. U. Strasbourg, INRIA, CEA, U. Lille, U. Luxembourg, CNRS, Sorbonne U., Oct. 2024. URL: https://github.com/numpex/exa-ma-d7.1/.

[Pru+24b] Christophe Prud'homme, Vincent Chabannes, Thomas Saigre, Christophe Trophime, Luca Berti, Abdoulaye Samaké, Céline Van Landeghem, Marcela Szopos, Laetitia Giraldi, Silvia Bertoluzza, and Yvon Maday. *feelpp/feelpp: Feel++ Release V111 preview.10*. July 2024. DOI: 10.5281/ZENODO.12742155. URL: https://zenodo.org/doi/10.5281/zenodo.12742155.

[PSS21] Christophe Prud'homme, Lorenzo Sala, and Marcela Szopos. "Uncertainty propagation and sensitivity analysis: results from the Ocular Mathematical Virtual Simulator". In: *Mathematical Biosciences and Engineering* 18.3 (2021), pp. 2010–2032. ISSN: 1551-0018. DOI: 10.3934/mbe.2021105. URL: http://www.aimspress.com/article/doi/10.3934/mbe.2021105.

[PW05] Christine Purslow and James S. Wolffsohn. "Ocular Surface Temperature: A Review". In: *Eye & Contact Lens: Science & Clinical Practice* 31.3 (May 2005), pp. 117–123. ISSN: 1542-2321. DOI: 10.1097/01.ICL.0000141921.80061.17. URL: https://journals.lww.com/00140068-200505000-00006.

[Qin+21] Zhangrong Qin, Lingjuan Meng, Fan Yang, Chaoying Zhang, Binghai Wen, Guangxi Key Lab of Multi-source Information Mining & Security, Guangxi Normal University, Guilin 541004, China, and Ophthalmology Department, Nanxishan Hospital of Guangxi Zhuang Autonomous Region, Guilin 541002, China. "Aqueous humor dynamics in human eye: A lattice Boltzmann study". In: *Mathematical Biosciences and Engineering* 18.5 (2021), pp. 5006–5028. ISSN: 1551-0018. DOI: 10.3934/mbe.2021255. URL: http://www.aimspress.com/article/doi/10.3934/mbe.2021255.

[QMN16] Alfio Quarteroni, Andrea Manzoni, and Federico Negri. *Reduced Basis Methods for Partial Differential Equations*. Vol. 92. UNITEXT. Cham: Springer International Publishing, 2016. ISBN: 978-3-319-15430-5. DOI: 10.1007/978-3-319-15431-2. URL: http://link.springer.com/10.1007/978-3-319-15431-2.

[Raz+21] Saman Razavi, Anthony Jakeman, Andrea Saltelli, Clémentine Prieur, Bertrand Iooss, Emanuele Borgonovo, Elmar Plischke, Samuele Lo Piano, Takuya Iwanaga, William Becker, Stefano Tarantola, Joseph H.A. Guillaume, John Jakeman, Hoshin Gupta, Nicola Melillo, Giovanni Rabitti, Vincent Chabridon, Qingyun Duan, Xifu Sun, Stefán Smith, Razi Sheikholeslami, Nasim Hosseini, Masoud Asadzadeh, Arnald Puy, Sergei Kucherenko, and Holger R. Maier. "The Future of Sensitivity Analysis: An essential discipline for systems modeling and policy support". In: *Environmental Modelling & Software* 137 (Mar. 2021), p. 104954. ISSN: 13648152. DOI: 10.1016/j.envsoft.2020.104954. URL: https://linkinghub.elsevier.com/retrieve/pii/S1364815220310112.

[RBG17] Alejandro Ribes, Adrien Bruneton, and Anthony Geay. *SALOME: an Open-Source simulation platform integrating ParaView*. 2017. DOI: 10.13140/RG.2.2.12107.08485. URL: http://rgdoi.net/10.13140/RG.2.2.12107.08485.

[Rep+15] Rodolfo Repetto, Jan O. Pralits, Jennifer H. Siggers, and Paolo Soleri. "Phakic Iris-Fixated Intraocular Lens Placement in the Anterior Chamber: Effects on Aqueous Flow". In: *Investigative Opthalmology & Visual Science* 56.5 (May 2015), p. 3061. ISSN: 1552-5783. DOI: 10.1167/iovs.14-16118. URL: http://iovs.arvojournals.org/article.aspx?doi=10.1167/iovs.14-16118.

[RF77] Robert F. Rosenbluth and Irving Fatt. "Temperature measurements in the eye". In: *Experimental Eye Research* 25.4 (Oct. 1977), pp. 325–341. ISSN: 00144835. DOI: 10.1016/0014-4835(77)90100-2. URL: https://linkinghub.elsevier.com/retrieve/pii/0014483577901002.

[Rhe93] Werner C. Rheinboldt. "On the theory and error estimation of the reduced basis method for multi-parameter problems". In: *Nonlinear Analysis: Theory, Methods & Applications* 21.11 (Dec. 1993), pp. 849–858. ISSN: 0362546X. DOI: 10.1016/0362-546X(93)90050-3. URL: https://linkinghub.elsevier.com/retrieve/pii/0362546X93900503 (visited on 07/30/2024).

[RHP07] G. Rozza, D. B. P. Huynh, and A. T. Patera. "Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations". In: *Archives of Computational Methods in Engineering* 15.3 (Sept. 2007), pp. 1–47. ISSN: 1134-3060, 1886-1784. DOI: 10.1007/BF03024948. URL: http://link.springer.com/10.1007/BF03024948.

[RHP08] G. Rozza, D. B. P. Huynh, and A. T. Patera. "Reduced Basis Approximation and a Posteriori Error Estimation for Affinely Parametrized Elliptic Coercive Partial Differential Equations: Application to Transport and Continuum Mechanics". In: *Archives of Computational Methods in Engineering* 15.3 (Sept. 2008), pp. 229–275. ISSN: 1134-3060, 1886-1784. DOI: 10.1007/s11831-008-9019-9. URL: http://link.springer.com/10.1007/s11831-008-9019-9.

[Rod+19] Rocío Rodríguez-Cantano, Henrik N. Finsberg, Samuel T. Wall, and Joakim Sundnes. "A Bayesian Approach for Parameter Estimation in Computational Models of Cardiac Mechanics". In: *6th International Conference on Computational and Mathematical Biomedical Engineering – CMBE2019 Proceedings*. Tokyo, Japan: P. Nithiarasu, M. Ohta, M. Oshima (Eds.), June 2019, pp. 535–538.

[RRK13] R Ramakrishnan, Shalmali Ranaut, and Mona Khurana. "Aqueous Humor Dynamics". In: *Diagnosis and Management of Glaucoma*. Jaypee Brothers Medical Publishers (P) Ltd., 2013, pp. 76–76. ISBN: 978-93-5025-578-0. DOI: 10.5005/jp/books/11801_9. URL: https://www.jaypeedigital.com/book/9789350255780/chapter/ch9.

[Ruf+24] Alessia Ruffini, Alessia Casalucci, Caterina Cara, C. Ross Ethier, and Rodolfo Repetto. "Drug Distribution After Intravitreal Injection: A Mathematical Model". In: *Investigative Opthalmology & Visual Science* 65.4 (Apr. 2024), p. 9. ISSN: 1552-5783. DOI: 10.1167/iovs.65.4.9. URL: https://iovs.arvojournals.org/article.aspx?articleid=2793527.

[Sac+23] Riccardo Sacco, Greta Chiaravalli, Gal Antman, Giovanna Guidoboni, Alice Verticchio, Brent Siesky, and Alon Harris. "The role of conventional and unconventional adaptive routes in lowering of intraocular pressure: Theoretical model and simulation". In: *Physics of Fluids* 35.6 (June 2023), p. 061902. ISSN: 1070-6631, 1089-7666. DOI: 10.1063/5.0151091. URL: https://pubs.aip.org/pof/article/35/6/061902/2895206/The-role-of-conventional-and-unconventional.

[Sai+] Thomas Saigre, Vincent Chabannes, Christophe Prud'homme, and Marcela Szopos. "A coupled fluid-dynamics-heat transfer model for 3D simulations of the aqueous humor flow in the human eye". In preparation.

[Sai+24a] Thomas Saigre, Vincent Chabannes, Giovanna Guidoboni, Christophe Prud'homme, Marcela Szopos, and Sangly P Srinivas. *Effect of Cooling of the Ocular Surface on Endothelial Cell Sedimentation in Cell Injection Therapy: Insights from Computational Fluid Dynamics*. 2024.

[Sai+24b] Thomas Saigre, Christophe Prud'homme, Marcela Szopos, and Vincent Chabannes. "A coupled fluid-dynamics-heat transfer model for 3D simulations of the aqueous humor flow in the human eye". In: *8th International Conference on Computational and Mathematical Biomedical Engineering – CMBE2024 Proceedings*. Vol. 2. Arlington (Virginia), United States: P. Nithiarasu and R. Löhner (Eds.), June 2024, pp. 508–512. ISBN: 978-0-9562914-7-9. URL: https://hal.science/hal-04558924.

[Sai+24c] Thomas Saigre, Christophe Prud'homme, Marcela Szopos, and Vincent Chabannes. *Mesh and configuration files to perform coupled heat+fluid simulations on a realistic human eyeball geometry with Feel++*. Oct. 2024. DOI: 10.5281/ZENODO.13886143. URL: https://zenodo.org/doi/10.5281/zenodo.13886143.

[Sal+19] Andrea Saltelli, Ksenia Aleksankina, William Becker, Pamela Fennell, Federico Ferretti, Niels Holst, Sushan Li, and Qiongli Wu. "Why so many published sensitivity analyses are false: A systematic review of sensitivity analysis practices". In: *Environmental Modelling & Software* 114 (Apr. 2019), pp. 29–39. ISSN: 13648152. DOI: 10.1016/j.envsoft.2019.01.012. URL: https://linkinghub.elsevier.com/retrieve/pii/S1364815218302822.

[Sal+21a] Lorenzo Sala, Christophe Prud'homme, Giovanna Guidoboni, Marcela Szopos, and Alon Harris. "Mathematical Assessment of the Role of Three Factors Entangled in the Development of Glaucoma by Means of the Ocular Mathematical Virtual Simulator". In: *Numerical Mathematics and Advanced Applications ENUMATH 2019*. Ed. by Fred J. Vermolen and Cornelis Vuik. Vol. 139. Series Title: Lecture Notes in Computational Science and Engineering. Cham: Springer International Publishing, 2021, pp. 851–860. ISBN: 978-3-030-55874-1. DOI: 10.1007/978-3-030-55874-1_84. URL: https://link.springer.com/10.1007/978-3-030-55874-1_84 (visited on 10/16/2024).

[Sal+21b] Andrea Saltelli, Anthony Jakeman, Saman Razavi, and Qiongli Wu. "Sensitivity analysis: A discipline coming of age". In: *Environmental Modelling & Software* 146 (Dec. 2021), p. 105226. ISSN: 13648152. DOI: 10.1016/j.envsoft.2021.105226. URL: https://linkinghub.elsevier.com/retrieve/pii/S1364815221002681.

[Sal+23] Lorenzo Sala, Christophe Prud'homme, Giovanna Guidoboni, Marcela Szopos, and Alon Harris. "The ocular mathematical virtual simulator: A validated multiscale model for hemodynamics and biomechanics in the human eye". In: *International Journal for Numerical Methods in Biomedical Engineering* (Nov. 2023), e3791. ISSN: 2040-7939, 2040-7947. DOI: 10.1002/cnm.3791. URL: https://onlinelibrary.wiley.com/doi/10.1002/cnm.3791.

[Sal02] Andrea Saltelli. "Sensitivity Analysis for Importance Assessment". In: *Risk Analysis* 22.3 (June 2002), pp. 579–590. ISSN: 0272-4332, 1539-6924. DOI: 10.1111/0272-4332.00040. URL: https://onlinelibrary.wiley.com/doi/10.1111/0272-4332.00040.

[Sal16] Lorenzo Sala. *Eye2Brain*. July 2016. URL: https://www.cemosis.fr/projects/eye2brain/.

[Sal19] Lorenzo Sala. "Mathematical modelling and simulation of ocular blood flows and their interactions". Issue: 2019STRAD021. PhD Theses. Université de Strasbourg, Sept. 2019. URL: https://tel.archives-ouvertes.fr/tel-02284233.

[Sch97] Joachim Schöberl. "NETGEN An advancing front 2D/3D-mesh generator based on abstract rules". In: *Computing and Visualization in Science* 1.1 (July 1997), pp. 41–52. ISSN: 1432-9360, 1433-0369. DOI: 10.1007/s007910050004. URL: http://link.springer.com/10.1007/s007910050004.

[Sco88] J A Scott. "A finite element model of heat transport in the human eye". In: *Physics in Medicine and Biology* 33.2 (Feb. 1988), pp. 227–242. ISSN: 0031-9155, 1361-6560. DOI: 10.1088/0031-9155/33/2/003. URL: https://iopscience.iop.org/article/10.1088/0031-9155/33/2/003.

[SLG06] Krish D. Singh, Nicola S. Logan, and Bernard Gilmartin. "Three-Dimensional Modeling of the Human Eye Based on Magnetic Resonance Imaging". In: *Investigative Opthalmology & Visual Science* 47.6 (June 2006), p. 2272. ISSN: 1552-5783. DOI: 10.1167/iovs.05-0856. URL: http://iovs.arvojournals.org/article.aspx?doi=10.1167/iovs.05-0856.

[Sob01] I.M Sobol. "Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates". In: *Mathematics and Computers in Simulation* 55.1-3 (Feb. 2001), pp. 271–280. ISSN: 03784754. DOI: 10.1016/S0378-4754(00)00270-6. URL: https://linkinghub.elsevier.com/retrieve/pii/S0378475400002706.

[Sob93] Ilya M. Sobol. "Sensitivity Estimates for Nonlinear Mathematical Models". In: 1993. URL: https://api.semanticscholar.org/CorpusID:115460399.

[SPS24a] Thomas Saigre, Christophe Prud'homme, and Marcela Szopos. *Model order reduction and sensitivity analysis for complex heat transfer simulations inside the human eyeball*. Oct. 2024. DOI: 10.5281/ZENODO.13907890. URL: https://zenodo.org/doi/10.5281/zenodo.13907890.

[SPS24b] Thomas Saigre, Christophe Prud'homme, and Marcela Szopos. "Model order reduction and sensitivity analysis for complex heat transfer simulations inside the human eyeball". In: *International Journal for Numerical Methods in Biomedical Engineering* 40.11 (Sept. 2024), e3864. ISSN: 2040-7939, 2040-7947. DOI: 10.1002/cnm.3864. URL: https://onlinelibrary.wiley.com/doi/10.1002/cnm.3864.

[SSS79] A. Shapiro, Y. Shoenfeld, and Y. Shapiro. "The effect of exposure to heat on intraocular pressure". In: *Albrecht von Graefes Archiv für Klinische und Experimentelle Ophthalmologie* 210.3 (1979), pp. 183–185. ISSN: 0065-6100, 1435-702X. DOI: 10.1007/BF00414567. URL: http://link.springer.com/10.1007/BF00414567.

[Sta07] International Organization for Standardization. *Thermal bridges in building construction – Heat flows and surface temperatures – Detailed calculations*. Standard ISO 10211:2007. Geneva, CH: International Organization for Standardization, Dec. 2007. URL: https://www.iso.org/standard/62711.html.

[Šuš+21] Anna Šušnjara, Hrvoje Dodig, Dragan Poljak, and Mario Cvetkovic. "Stochastic - Deterministic Thermal Dosimetry Below 6 GHz for 5G Mobile Communication Systems". In: *IEEE Transactions on Electromagnetic Compatibility* 63.5 (Oct. 2021), pp. 1667–1679. ISSN: 0018-9375, 1558-187X. DOI: 10.1109/TEMC.2021.3098431. URL: https://ieeexplore.ieee.org/document/9522096/.

[Šuš+22] Anna Šušnjara, Ožbej Verhnjak, Dragan Poljak, Mario Cvetković, and Jure Ravnik. "Uncertainty quantification and sensitivity analysis of transcranial electric stimulation for 9-subdomain human head model". In: *Engineering Analysis with Boundary Elements* 135 (Feb. 2022), pp. 1–11. ISSN: 09557997. DOI: 10.1016/j.enganabound.2021.10.026. URL: https://linkinghub.elsevier.com/retrieve/pii/S0955799721003192.

[SZM13] Till Saxer, Andreas Zumbuehl, and Bert Müller. "The use of shear stress for targeted drug delivery". In: *Cardiovascular Research* 99.2 (July 2013), pp. 328–333. ISSN: 1755-3245, 0008-6363. DOI: 10.1093/cvr/cvt102. URL: https://academic.oup.com/cardiovascres/article-lookup/doi/10.1093/cvr/cvt102.

[Tsu15] Yutaka Tsuzuki. "Existence and Uniqueness of Solutions to Heat Equations with Hysteresis Coupled with Navier–Stokes Equations in 2D and 3D". In: *Journal of Mathematical Fluid Mechanics* 17.3 (Sept. 2015), pp. 577–597. ISSN: 1422-6928, 1422-6952. DOI: 10.1007/s00021-015-0210-0. URL: http://link.springer.com/10.1007/s00021-015-0210-0.

[TTP19] Carol B. Toris, George Tye, and Padmanabhan Pattabiraman. "Changes in Parameters of Aqueous Humor Dynamics Throughout Life". In: *Ocular Fluid Dynamics*. Ed. by Giovanna Guidoboni, Alon Harris, and Riccardo Sacco. Series Title: Modeling and Simulation in Science, Engineering and Technology. Cham: Springer International Publishing, 2019, pp. 161–190. ISBN: 978-3-030-25886-3. DOI: 10.1007/978-3-030-25886-3_6. URL: http://link.springer.com/10.1007/978-3-030-25886-3_6.

[Upd+17] Adam Updegrove, Nathan M. Wilson, Jameson Merkow, Hongzhi Lan, Alison L. Marsden, and Shawn C. Shadden. "SimVascular: An Open Source Pipeline for Cardiovascular Simulation". In: *Annals of Biomedical Engineering* 45.3 (Mar. 2017), pp. 525–541. ISSN: 0090-6964, 1573-9686. DOI: 10.1007/s10439-016-1762-8. URL: http://link.springer.com/10.1007/s10439-016-1762-8.

[Vey14] Stéphane Veys. "Un Framework de calcul pour la méthode des bases réduites : applications à des problèmes non-linéaire multi-physiques". Issue: 2014GRENM069. Theses. Université de Grenoble, Nov. 2014. URL: https://theses.hal.science/tel-01079415.

[VPP03] Karen Veroy, Christophe Prud'homme, and Anthony T. Patera. "Reduced-basis approximation of the viscous Burgers equation: rigorous a posteriori error bounds". In: *Comptes Rendus. Mathématique* 337.9 (Oct. 2003), pp. 619–624. ISSN: 1778-3569. DOI: 10.1016/j.crma.2003.09.023. URL: https://comptes-rendus.academie-sciences.fr/mathematique/articles/10.1016/j.crma.2003.09.023/.

[Wan+16] Wenjia Wang, Xiuqing Qian, Hongfang Song, Mindi Zhang, and Zhicheng Liu. "Fluid and structure coupling analysis of the interaction between aqueous humor and iris". In: *BioMedical Engineering OnLine* 15.S2 (Dec. 2016), p. 133. ISSN: 1475-925X. DOI: 10.1186/s12938-016-0261-3. URL: http://biomedical-engineering-online.biomedcentral.com/articles/10.1186/s12938-016-0261-3.

[Wei24] Robert W Weisenthal. *2024-2025 basic and clinical science course, section 8: External disease and cornea*. July 2024.

[Wik24] Wikipedia. *Human eye - Wikipedia — en.wikipedia.org*. 2024. URL: https://en.wikipedia.org/wiki/Human_eye.

[WL10] Robert N. Weinreb and Jeffrey Liebmann. *Medical Treatment of Glaucoma: the 7th Consensus Report of the World Glaucoma Association*. OCLC: 753576362. Amsterdam, Gilsum, NH: Kugler Publications ; Distributors for the USA and Canada, Pathway Book Service, 2010. ISBN: 978-90-6299-854-8.

[Xu+13] Yingqian Xu, Bochu Wang, Jia Deng, Zerong Liu, and Liancai Zhu. "A potential model for drug screening by simulating the effect of shear stress in vivo on endothelium". In: *Biorheology* 50.1-2 (2013), pp. 33–43. ISSN: 0006355X. DOI: 10.3233/BIR-130624. URL: https://www.medra.org/servlet/aliasResolver?alias=iospress&doi=10.3233/BIR-130624.

[Yam+10] Yasuaki Yamamoto, Toshihiko Uno, Takeshi Joko, Atsushi Shiraishi, and Yuichi Ohashi. "Effect of Anterior Chamber Depth on Shear Stress Exerted on Corneal Endothelial Cells by Altered Aqueous Flow after Laser Iridotomy". In: *Investigative Opthalmology & Visual Science* 51.4 (Apr. 2010), p. 1956. ISSN: 1552-5783. DOI: 10.1167/iovs.09-4280. URL: http://iovs.arvojournals.org/article.aspx?doi=10.1167/iovs.09-4280.

[Yan+22] Shu Yang, Jing Zhang, Youhua Tan, and Yan Wang. "Unraveling the mechanobiology of cornea: From bench side to the clinic". In: *Frontiers in Bioengineering and Biotechnology* 10 (Oct. 2022), p. 953590. ISSN: 2296-4185. DOI: 10.3389/fbioe.2022.953590. URL: https://www.frontiersin.org/articles/10.3389/fbioe.2022.953590/full.

[You+22] Yacoub A Yousef, Mona Mohammad, Ibrahim AlNawaiseh, Reem AlJabari, Mario Damiano Toro, Almutez Gharaibeh, Robert Rejdak, Katarzyna Nowomiejska, Sandrine Zweifel, Teresio Avitabile, Magdalena Rejdak, and Rashed Nazzal. "Ultrasound Biomicroscopy Measurements of the Normal Thickness for the Ciliary Body and the Iris in a Middle East Population". In: *Clinical Ophthalmology* Volume 16 (Jan. 2022), pp. 101–109. ISSN: 1177-5483. DOI: 10.2147/OPTH.S297977. URL: https://www.dovepress.com/ultrasound-biomicroscopy-measurements-of-the-normal-thickness-for-the--peer-reviewed-fulltext-article-OPTH.

CEMRACS References

[Aki+19] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. "Optuna: A Next-generation Hyperparameter Optimization Framework". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2019.

[Ant+22] Eric Aislan Antonelo, Eduardo Camponogara, Laio Oriel Seman, Eduardo Rehbein de Souza, Jean P. Jordanou, and Jomi F. Hubner. *Physics-Informed Neural Nets for Control of Dynamical Systems*. _eprint: 2104.02556. 2022.

[Arr+14] Marie-Claire Arrieta, Leah T Stiemsma, Nelly Amenyogbe, Eric M Brown, and Brett Finlay. "The intestinal microbiome in early life: health and disease". In: *Frontiers in immunology* 5 (2014). Publisher: Frontiers Media SA, p. 427.

[Bai16] Stephen Baigent. *Lotka-Volterra Dynamics: An Introduction*. 2016. URL: https://api.semanticscholar.org/CorpusID:124912570.

[BB23] Hubert Baty and Leo Baty. *Solving differential equations using physics informed deep learning: a hand-on tutorial with benchmark tests*. Version Number: 2. 2023. DOI: 10.48550/ARXIV.2302.12260. URL: https://arxiv.org/abs/2302.12260.

[Buc+16] Vanni Bucci, Belinda Tzen, Ning Li, Matt Simmons, Takeshi Tanoue, Elijah Bogart, Luxue Deng, Vladimir Yeliseyev, Mary L Delaney, Qing Liu, et al. "MDSINE: Microbial Dynamical Systems INference Engine for microbiome time-series analyses". In: *Genome biology* 17 (2016). Publisher: Springer, pp. 1–17.

[But+19] Mary I Butler, Sabrina Mörkl, Kiran V Sandhu, John F Cryan, and Timothy G Dinan. "The gut microbiome and mental health: what should we tell our patients?: le microbiote intestinal et la santé mentale: que devrions-nous dire à nos patients?" In: *The Canadian Journal of Psychiatry* 64.11 (2019). Publisher: SAGE Publications Sage CA: Los Angeles, CA, pp. 747–760.

[BX14] Vanni Bucci and Joao B Xavier. "Towards predictive models of the human gut microbiome". In: *Journal of molecular biology* 426.23 (2014). Publisher: Elsevier, pp. 3907–3916.

[CD12] John F Cryan and Timothy G Dinan. "Mind-altering microorganisms: the impact of the gut microbiota on brain and behaviour". In: *Nature reviews neuroscience* 13.10 (2012). Publisher: Nature Publishing Group UK London, pp. 701–712.

[Don+13] Frank Dondelinger, Dirk Husmeier, Simon Rogers, and Maurizio Filippone. "ODE parameter inference using adaptive gradient matching with Gaussian processes". In: *International Conference on Artificial Intelligence and Statistics*. 2013. URL: https://api.semanticscholar.org/CorpusID:15234282.

[Fai+10] Ali Faisal, Frank Dondelinger, Dirk Husmeier, and Colin M Beale. "Inferring species interaction networks from species abundance data: A comparative evaluation of various statistical and machine learning methods". In: *Ecological Informatics* 5.6 (2010). Publisher: Elsevier, pp. 451–464.

[FHG17] Jonathan Friedman, Logan M Higgins, and Jeff Gore. "Community structure follows simple assembly rules in microbial microcosms". In: *Nature ecology & evolution* 1.5 (2017). Publisher: Nature Publishing Group UK London, p. 0109.

[Fie17] Noah Fierer. "Embracing the unknown: disentangling the complexities of the soil microbiome". In: *Nature Reviews Microbiology* 15.10 (2017). Publisher: Nature Publishing Group UK London, pp. 579–590.

[FS95] HI Freedman and HL Smith. "Tridiagonal competitive-cooperative Kolmogorov systems". In: *Differential Equations and Dynamical Systems* 3.4 (1995), pp. 367–382.

[Gau+16] Dominique Gauguier, Michel Neunlist, Harry Sokol, and Zitvogel Laurence. *Microbiote intestinal (flore intestinale). Une piste sérieuse pour comprendre l'origine de nombreuses maladies*. 2016. URL: https://www.inserm.fr/dossier/microbiote-intestinal-flore-intestinale/.

[Ger14] Georg K Gerber. "The dynamic microbiome". In: *FEBS letters* 588.22 (2014). Publisher: Elsevier, pp. 4131–4139.

[Har20] Jérome Harmand. "Les modèles de Lotka Volterra Généralisés : intérêts et limites en écologie microbienne". In: *Webinaire "Comprendre et valoriser les communautés microbiennes"*. Web, France, Sept. 2020. URL: https://hal.inrae.fr/hal-02954427.

[HLM12] Lora V Hooper, Dan R Littman, and Andrew J Macpherson. "Interactions between the microbiota and the immune system". In: *science* 336.6086 (2012). Publisher: American Association for the Advancement of Science, pp. 1268–1273.

[Hos+24] Paguiel Javan Hossie, Béatrice Laroche, Thibault Malou, Lucas Perrin, Thomas Saigre, and Lorenzo Sala. "Surrogate modeling of interactions in microbial communities through Physics-Informed Neural Networks." Feb. 2024. URL: https://hal.inrae.fr/hal-04440736.

[Jia98] JF Jiang. "The complete classification of asymptotic behavior for bounded cooperative Lotka-Volterra systems with the assumption (SM)". In: *Quarterly of Applied Mathematics* 56.1 (1998), pp. 37–53.

[Jin+20] Pengzhan Jin, Zhen Zhang, Aiqing Zhu, Yifa Tang, and George Em Karniadakis. "SympNets: Intrinsic structure-preserving symplectic networks for identifying Hamiltonian systems". In: *Neural Networks* 132 (Dec. 2020), pp. 166–179. ISSN: 08936080. DOI: 10.1016/j.neunet.2020.08.017. URL: https://linkinghub.elsevier.com/retrieve/pii/S0893608020303063.

[KG88] George Karakostas and Istvan Györi. "Global stability in job systems". In: *Journal of mathematical analysis and applications* 131.1 (1988). Publisher: Academic Press, pp. 85–96.

[Kur20] Pykes Kurtis. *The Vanishing/Exploding Gradient Problem in Deep Neural Networks Understanding the obstacles that faces us when building deep neural networks*. 2020. URL: https://towardsdatascience.com/the-vanishing-exploding-gradient-problem-in-deep-neural-networks-191358470c11.

[Lar+18] B. Laroche, N. Brunel, D. Goujot, and S. Labarthe. "Parameter estimation for dynamical systems using an FDA approach". In: *11th International Conference of the ERCIM WG on Computational and Methodological Statistics (CMStatistics 2018)*. Pise, Italy, Dec. 2018. URL: https://hal.science/hal-02045430.

[Lot56] Alfred J Lotka. *Elements of mathematical biology*. Dover Publications, 1956.

[MD17] Philippe Marteau and Joël Doré. "Le microbiote intestinal". In: *EMC-Gastro-entérologie* 12 (2017), pp. 1–8.

[Mos76] Jürgen Moser. "Periodic orbits near an equilibrium and a theorem by Alan Weinstein". In: *Communications in Pure Applied Mathematics* 29 (1976), pp. 727–747.

[MS15] Ulrich G Mueller and Joel L Sachs. "Engineering microbiomes to improve plant and animal health". In: *Trends in microbiology* 23.10 (2015). Publisher: Elsevier, pp. 606–617.

[Poy+06] A. A. Poyton, M. S. Varziri, K. B. McAuley, P. J. McLellan, and J. O. Ramsay. "Parameter estimation in continuous-time dynamic models using principal differential analysis". In: *Computers & Chemical Engineering* 30.4 (2006), pp. 698–708. ISSN: 0098-1354. DOI: https://doi.org/10.1016/j.compchemeng.2005.11.008. URL: https://www.sciencedirect.com/science/article/pii/S0098135405003042.

[Ram+07] J. O. Ramsay, G. Hooker, D. Campbell, and J. Cao. "Parameter estimation for differential equations: a generalized smoothing approach". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 69.5 (2007), pp. 741–796. DOI: https://doi.org/10.1111/j.1467-9868.2007.00610.x. URL: https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9868.2007.00610.x.

[RML23] Jamshaid Ul Rahman, Faiza Makhdoom, and Dianchen Lu. "Amplifying Sine Unit: An Oscillatory Activation Function for Deep Neural Networks to Recover Nonlinear Oscillations Efficiently". In: *arXiv preprint arXiv:2304.09759* (2023).

[RPK19] M. Raissi, P. Perdikaris, and G. E. Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational Physics* 378 (2019), pp. 686–707. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2018.10.045. URL: https://www.sciencedirect.com/science/article/pii/S0021999118307125.

[RS05] James O Ramsay and Bernard W Silverman. *Fitting differential equations to functional data: Principal differential analysis*. Springer, 2005.

[San11] Philippe Sansonetti. "La relation hôte-microbiote : une insondable symbiose ?" In: *Lett. Du Coll. Fr.* 32 (Oct. 2011). Publisher: OpenEdition, pp. 14–15.

[Seg+11] Nicola Segata, Jacques Izard, Levi Waldron, Dirk Gevers, Larisa Miropolsky, Wendy S Garrett, and Curtis Huttenhower. "Metagenomic biomarker discovery and explanation". In: *Genome biology* 12 (2011). Publisher: Springer, pp. 1–18.

[Ste+13] Richard R Stein, Vanni Bucci, Nora C Toussaint, Charlie G Buffie, Gunnar Rätsch, Eric G Pamer, Chris Sander, and Joao B Xavier. "Ecological modeling from time-series inference: insight into dynamics and stability of intestinal microbiota". In: *PLoS computational biology* 9.12 (2013). Publisher: Public Library of Science San Francisco, USA, e1003388.

[Tan+20] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. "Fourier features let networks learn high frequency functions in low dimensional domains". In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NIPS '20. event-place: Vancouver, BC, Canada. Red Hook, NY, USA: Curran Associates Inc., 2020. ISBN: 978-1-71382-954-6.

[TJ17] Elizabeth Thursby and Nathalie Juge. “Introduction to the human gut microbiota”. In: *Biochemical journal* 474.11 (2017). Publisher: Portland Press Ltd., pp. 1823–1836.

[VB31] V. Volterra and M. Brelot. *Leçons sur la théorie mathématique de la lutte pour la vie*. Paris : Gauthier-Villars, 1931. URL: http://lib.ugent.be/catalog/rug01:000471833.

[Vol27] V. Volterra. *Variazioni e fluttuazioni del numero d'individui in specie animali conviventi*. Vol. 2. Societá anonima tipografica "Leonardo da Vinci", 1927.

[Wen+19] Philippe Wenk, Alkis Gotovos, Stefan Bauer, Nico S. Gorbach, Andreas Krause, and Joachim M. Buhmann. “Fast Gaussian process based gradient matching for parameter identification in systems of nonlinear ODEs”. In: *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*. Ed. by Kamalika Chaudhuri and Masashi Sugiyama. Vol. 89. Proceedings of Machine Learning Research. PMLR, Apr. 2019, pp. 1351–1360. URL: https://proceedings.mlr.press/v89/wenk19a.html.

[Yas21] Bohra Yash. *The Challenge of Vanishing/Exploding Gradients in Deep Neural Networks*. 2021. URL: https://www.analyticsvidhya.com/blog/2021/06/the-challenge-of-vanishing-exploding-gradients-in-deep-neural-networks/.